Implementing Weighted Residual, Spectral and Finite Element Methods

Orthogonal Collocation, Pseudospectral, Differential Quadrature

Larry C. Young, tildentechnologies.com

Table of Contents

June 2020 edition

Preface

After having been involved during the early days of orthogonal collocation [Young and Finlayson, (1973, 1976), Young (1977, 1981)], I had reason to use it again more recently. A review of the last 50 years revealed that some aspects of the method appear to have never been explained clearly or never explained at all. If you look at posts on various internet sites, you reach the conclusion that much confusion exists amongst students and practitioners. I eventually published a more recent article to address the confusion and demonstrate correct formulations [Young 2019]. The present monograph extends that effort and further demonstrates proper implementation with many examples in a variety of computer languages.

The term *Method of Weighted Residual* (MWR) covers many methods for the approximate solution of differential equations. The name can be traced to comments by Richard Courant at the First International Congress on Applied Mechanics in 1924. In more recent years, the name has been almost supplanted by the newer name *Spectral Methods*. Fundamentally, there is no difference. MWR is not a specific method, but a category of methods based on the same principal. It is a flexible framework for developing approximations. The approximate solution is developed from an assumed trial solution or assumed form which contains a finite number of adjustable parameters. The assumed form is usually a linear series of trial or basis functions which are the first few members of a complete set. The method gets its name from the procedure used to determine the free parameters. The assumed solution is substituted into the differential equation to form the residual or error function. The parameters are determined by setting weighted averages of the residual to zero. This framework provides a great deal of flexibility in the selection of the trial solution and the weighting scheme. Many specific methods are fundamentally MWR. Some examples are: spectral method, pseudospectral method, orthogonal collocation, differential quadrature, finite element method and spectral element method.

This monograph is focused on the application of collocation-like (Orthogonal Collocation, Pseudospectral, Differential Quadrature) methods and other Methods of Weighted Residuals (MWR) to solve engineering problems. The problems are drawn mostly from the authors experience in heat and mass transfer, chemical reaction engineering, porous media flow and to a lesser extent fluid mechanics and mechanical vibration problems. Trial function representations are applied to problems in one or more spatial coordinates, such as in a boundary value problem. Both global approximations and finite element trial functions are used. Nonperiodic problems in finite domains are considered, but other cases are addressed in the many references provided. Computer codes are provided for the examples in Matlab/Octave, Python, Excel, Fortran and C++.

The monograph is written for anyone that understands mathematics at a level typical of an undergraduate engineering education, e.g. differential equations and some basic numerical methods. The text will be useful if you have that type of background and want to learn about the MWR or finite elements in a general way - not only the mathematics but also how to implement them in computer code. I assume you can start with an engineering differential

equation and convert it to a dimensionless form, recognizing what the terms mean. A knowledge of the solution of linear algebraic systems by Gaussian elimination or LU factorization is also assumed together with a basic understanding of eigenvalues and eigenvectors. For nonlinear problems, some knowledge of Newton-Raphson and other iterative methods is helpful. Many of these topics are briefly described in the text, so the text and computer code along with an outside reference source may be adequate to achieve understanding. Fully working codes are provided, so the reader does not need to implement the basic calculations, but if you understand the fundamentals you will be able to follow how alternative solution methods are evaluated in order to pick a good method.

Some goals of the monograph are:

1. improved understanding of the MWR - how to formulate them correctly (including boundary conditions) and how different methods are related
2. demonstrate clean and modern computer implementation of MWR for a variety of problems
3. explain how to analyze alternative solution schemes in order to pick an effective method
4. to provide code and a framework for the reader to experiment with and build upon
5. reduce confusion by using logical and consistent terminology and by highlighting equivalent methods which go by different names

One area of confusion is the implementation of boundary conditions, especially those involving derivatives. These are called *natural* boundary conditions in the Rayleigh-Ritz variational method and its generalization, the Galerkin method. Through the examples, I demonstrate the correct implementation of boundary conditions, and the consequences if they are not properly implemented.

One caveat about the codes is that they do not produce graphs or pretty pictures. I find most built in graphics to be clunky. I always want to compare this calculation with that one on the same graph and that requires too much effort. Since there is no way for me to predict what you (or I) want to look at, I prefer to print the results in text files that are tab delimited, so they can easily be exported to a spreadsheet like Excel or graphics program like Tecplot®.

Since I am not a mathematician or an academician, you may ask - What does this monograph offer? As an engineer, I have always taken a hands-on approach, so that is the approach taken here. Most engineers complain they have difficulty grasping mathematics, because it is often divorced from their reality. What better way to learn, than by applying the methods to solve problems? Theory is introduced when it helps to improve understanding. I have always felt that finite element methods are best learned after you are familiar with global methods - using simple polynomials of one type or another. That is the approach taken here, i.e. global methods first, then finite element methods. The easiest methods to understand are collocation methods. When correctly implemented, they are usually the most efficient methods, so they too are emphasized.

Most engineers are notoriously terrible coders. Could it have something to do with examples of engineering code they've learned from? I will not claim to be the best coder in the world, but I

do have a system gained from considerable experience. Most of my knowledge of programming is self-taught through reading, self-study, the occasional short course, but mostly experimentation. I've found one of the best ways to learn is to look at your own code after six months or a year. Can you read it? Or, do you ask, "What the heck was I thinking?" When I started developing my most successful code in the mid-1980s I organized it differently from other codes for the same type of problem. The code was a reservoir simulator, so I organized it along the lines of: rock properties, fluid properties, gridding, coefficients, linear solver, etc. Later, the concept of object-oriented programming (OOP) was introduced, and I realized that was what I was trying to achieve. The most important OOP concept is the packaging of data and procedures together - called encapsulation. OOP was difficult (impossible) to fully implement in Fortran 77, which was the only viable language on the targeted supercomputers. However, I managed to put the code in separate files, and the data into common blocks accessed by only those files. One of the greatest difficulties encountered with my Fortran 77 project was the lack of an ability to allocate memory. Eventually, I used two small C programs as portable assembly language and on top of them developed my own first-fit memory allocation code, all in Fortran 77. The code would compile and run (without manual modifications) on most of the supercomputers, workstations and PC's available in the 1990s. Mixed language programing is now easier. If I were writing a similar code today, it would use a mixture of languages.

What are my credentials? I am not an academic, but I have taught a few courses at the university level. I consider myself a researcher, but also an implementer. For my thesis I worked on the formulation and implementation of models for automotive catalytic converters. Those models were very advanced for the time and more comprehensive than most that appeared years after. Later, I worked for a large oil company research center developing numerical methods for petroleum reservoir simulation, specifically finite element methods and compositional models for enhanced oil recovery. When the oils companies began to downgrade their research programs in the 1980s, I formed my own company. There, my coworkers and I wrote, maintained and successfully licensed reservoir simulation software to oil companies all over the world. The code targeted the supercomputers available at the time. However, we discovered that once the code was streamlined for vector processing, it also performed quite well on personal computers and workstations. My 15 minutes of fame came at a meeting of the Society of Petroleum Engineers in 1987. A comparison of several simulators on the same problem revealed that ours was faster by orders of magnitude, while giving comparable results. Obviously, there was more to it than just vector processing. In the mid-1990s, we were the first to demonstrate a simulation with over a million grid cells. I used to consider myself – an engineer, mathematician and computer scientist in more or less equal measure. After successfully running my engineering software company, I added businessman to the list.

1. Introduction to the Basics

The Weighted Residual, Spectral and Finite Element methods are procedures for the approximate solution of differential equations. Although methods of this type are applied in the time domain for evolution or initial value problems, our attention here is restricted to applications in the spatial coordinates, such as in problems of boundary value type. These methods are often portrayed as being different, but they are basically the same or at least are based on the same principal. They all represent the solution by a trial solution or assumed form with adjustable parameters. The parameters are selected by forcing weighted averages of the equation residual to zero. Some of the methods produce excellent accuracy but with large computational complexity, while others are simple but have low accuracy. The primary focus of the monograph is on methods which give good accuracy, but with reduced complexity. These methods are collocation-like methods but have been given various names - *Orthogonal Collocation* (OC), *Pseudospectral Method* (PS), and *Differential Quadrature Method* (DQ).

By various names these methods have been applied extensively in such diverse fields as: solid and structural mechanics, fluid mechanics, elasticity, computational chemistry and chemical physics, quantum mechanics, biomedical modeling, optimal control, acoustics, geophysics and seismic modeling, heat and mass transfer, water resource modeling, petroleum reservoir simulation, chemical reaction engineering, meteorology and atmospheric modeling, and electromagnetism.

1.1 Trial Functions

When polynomial trial or basis functions are used, the form of the polynomials is one area of considerable confusion. Some texts state they are using orthogonal polynomials (called a *modal* approach) when they are not. Others imply orthogonal polynomials should always be used, warning of dire consequence otherwise. In fact, Lagrange interpolating polynomials are just as good and lead to a more intuitive *nodal* approach. Both approaches give identical answers, but they are in a different form. A nodal approach is used here, but all approaches are discussed in order to show their differences.

All Methods of Weighted Residuals (MWR) start with an assumed form or trial solution which contains a finite number of adjustable parameters. Although nonlinear trial solutions are possible, the usual procedure is to use a linear combination of trial or basis functions. For example, if one is solving for a variable u, the solution is approximated by:

$$u \cong \tilde{u} = \sum_{k=1}^{n} a_k \psi_k(x) \tag{1.1}$$

Where x represents one or more spatial coordinates, the $\boldsymbol{a}$ are adjustable parameters which could be time dependent, and the ψ are called *trial functions* or *basis functions*. The trial functions often, but not always, satisfy the boundary conditions. MWR, spectral methods and finite elements satisfy the differential equation approximately by forcing weighted averages of

the error or residual of the differential equation to vanish. This criterion produces values for the adjustable parameters.

We distinguish between several different types of trial functions. In this monograph, all trial functions are polynomials. In early applications, the trial functions were often simple *monomials*, x^k. Later applications used orthogonal polynomial trial functions, usually Legendre or Chebyshev polynomials. This approach is called a *modal* approximation since the adjustable parameters are analogous to the modes in a Fourier series. Modal trial functions are like:

$$\tilde{u} = \sum_{i=1}^{n} a_i P_{i-1}(x) \quad \text{or}$$
$$= x(1-x)\sum_{i=1}^{n} a_i P_{i-1}(x) \tag{1.2}$$

where P_i is an i^{th} degree orthogonal polynomials. The second form is one way of constructing modal trial functions for homogeneous boundary conditions $u(0) = u(1) = 0$. An example is presented in Fig. 1.1.

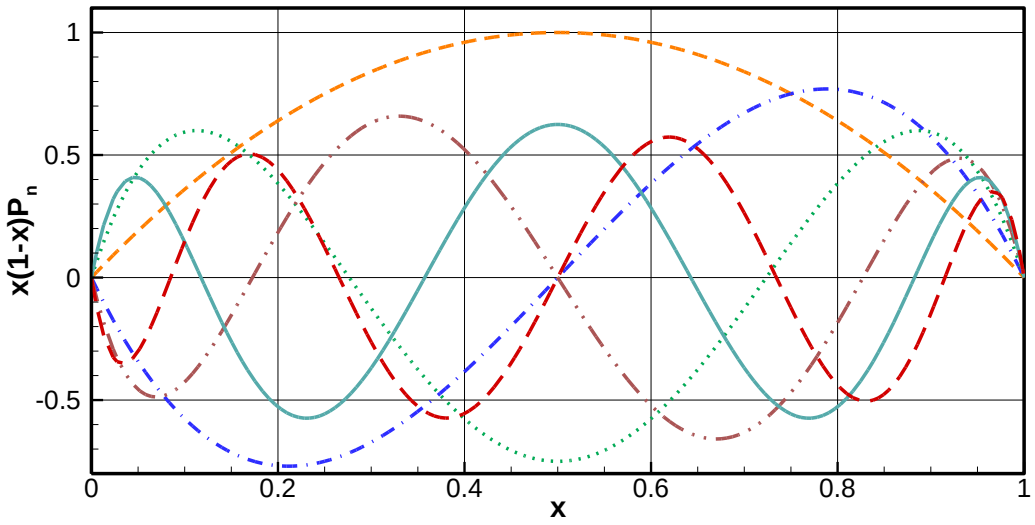


Fig. 1.1 Modal orthogonal polynomial trial functions, n = 6

The method is more intuitive and finite-difference-like if the adjustable parameters represent the approximate solution at the collocation or nodal points rather than polynomial coefficients, i.e. $a_k = \tilde{u}(x_k)$. In this case it is called a *nodal* approximation and the trial solution is:

$$\tilde{u} = \sum_{i=0}^{n+1} \tilde{u}(x_i)\ell_i(x) \tag{1.3}$$

where the trial functions, $\ell_i(x)$, are Lagrange interpolating polynomials:

$$\ell_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n+1} \frac{(x - x_j)}{(x_i - x_j)}$$

An example of nodal Lagrange trial functions is displayed in Fig. 1.2. Note these trial functions have the property $\ell_i(x_j) = \delta_{ij}$.

The choice of modal or nodal representations is largely a matter of convenience. In the absence of computer rounding errors all choices produce absolutely identical approximations. This important fact is obscured in many texts that emphasize the importance of orthogonal polynomial trial functions.

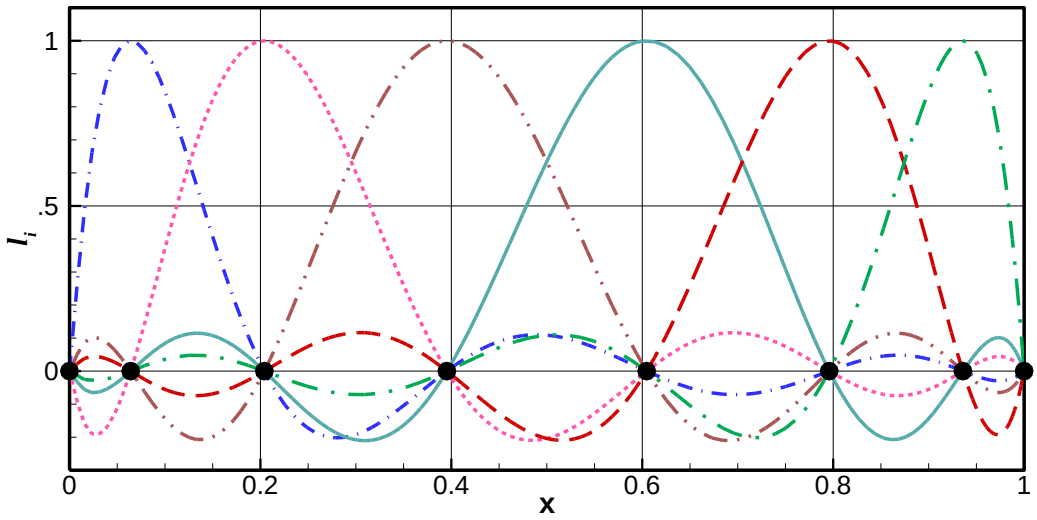


Fig. 1.2 Nodal Lagrange interpolating polynomial trial functions, n = 6

The easiest way to treat multidimensional problems is by using combinations of the one-dimensional trial functions. For example, Eq. (1.3) can be extended by the addition of a second, y, coordinate by:

$$\tilde{u} = \sum_{i=0}^{n_x+1} \sum_{j=0}^{n_y+1} \tilde{u}(x_i, y_j) \ell_i^x(x)\, \ell_j^y(y) \tag{1.4}$$

Trial functions composed of one-dimensional trial functions like this are called *tensor product* trial functions.

Finally, we distinguish between *global* trial functions where a single continuous polynomial approximates the solution, like Eqs. (1.2), (1.3) and (1.4), and piecewise continuous trial functions called *finite element* or *spectral element* approximations. Finite element trial functions are also called *shape functions*. In finite elements, polynomial trial functions are continuous within subdomains called elements but have limited continuity at the interface between elements. When the elements are distorted using a polynomial to describe their boundaries, they are called *isoparametric elements*. Fig. 1.3 shows an example where the trial functions

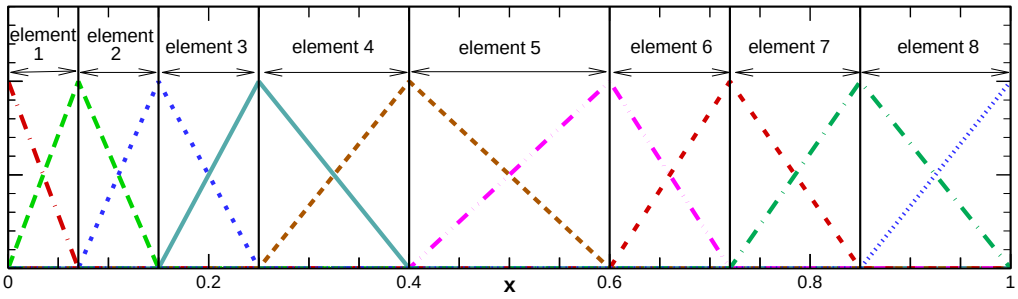


Fig. 1.3 Linear finite element trial functions

are simple linear interpolants of the solution. Finite element trial functions are usually nodal, like these, but modal trial functions have been used also.

Lanczos (1938,1956) was an early adopter of modal approximations. Villadsen and Stewart (1967) were the first to use nodal approximations in this context. Both considered only global trial functions. Courant (1943) was first to use finite element trial functions. In this monograph, we will initially consider global approximations, because they are easier to grasp. Finite elements are more understandable as an extension to global methods.

1.1.1 Orthogonal Polynomials and Quadrature. The distinguishing feature of orthogonal collocation is that the collocation points are chosen to be roots of orthogonal polynomials. The collocation points are usually the roots of Chebyshev polynomials of the 1st or 2nd kind, or the base points of Gaussian, Radau or Lobatto quadrature. Gaussian quadrature base points are the roots of Legendre polynomials. All these choices are the roots of Jacobi polynomials. When defined on the interval [0,1] they possess the orthogonality:

$$\int_0^1 (1-x)^{\alpha} x^{\beta} P_n^{(\alpha,\beta)}(x) P_m^{(\alpha,\beta)}(x) dx = 0 \quad \text{for} \quad n \neq m \tag{1.5}$$

where $\alpha = \beta = -½, 0, +½, +1$ for the Chebyshev 1st kind, Legendre (Gauss), Chebyshev 2nd kind polynomials, and for those whose roots are Lobatto quadrature base points, respectively. For

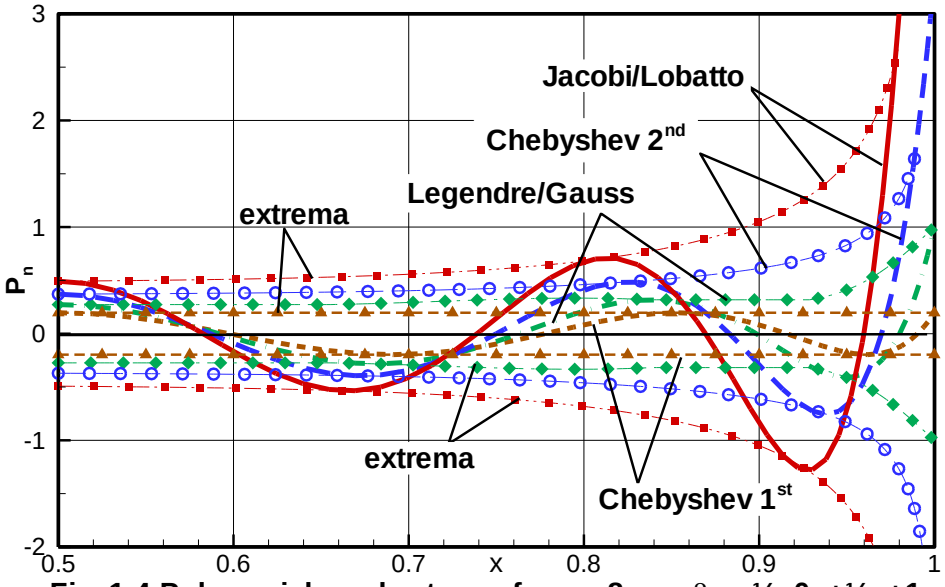


Fig. 1.4 Polynomials and extrema for n = 8, α = β = -½, 0, +½, +1

increasing n, these polynomials are alternating symmetric and antisymmetric about the centerline. Radau points are asymmetric with $\alpha = 0, \beta = 1$ or $\alpha = 1, \beta = 0$. The right half of the symmetric polynomials along with their extrema are compared in Fig. 1.4 for $n = 8$. What should be noted in Fig. 1.4 is that as α increases the roots are shifted away from the boundary and there is a greater variation in the extrema. The endpoint for the Jacobi (1,1) polynomial is off scale at 9. In all cases the points are more tightly clustered near the boundaries, with spacing $O(1/n^2)$. The Chebyshev polynomials of the 1st kind are the favorite ones for interpolation, since in general they minimize the maximum error [Hildebrand (1987) p. 469]. Note that in Fig. 1.4 the Chebyshev polynomials are normalized like other Jacobi polynomials rather than the traditional way.

Orthogonal polynomials are closely tied to the theory of accurate numerical integration methods. For a general reference to orthogonal polynomials and quadrature, the reader is directed to Hildebrand (1987) and Krylov (1962). The selection of these points is related to numerical integration of the form:

$$\int_0^1 f(x)x^\gamma dx \cong W_0 f(0) + \sum_{i=1}^{n} W_i f(x_i) + W_{n+1} f(1) \qquad (1.6)$$

where γ = 0,1 and 2 for planar, cylindrical and spherical geometry. Fig. 1.5 plots an example of quadrature point locations and weights for cases without symmetry, while Fig. 1.6 shows examples for cases with symmetry. For nonsymmetric problems, all but Radau points have points and weights symmetric about the centerline. For Gaussian quadrature, endpoint weights are not used, $W_0 = W_{n+1} = 0$, while the Radau-Left quadrature shown has $W_0 > 0$ and $W_{n+1} = 0$. There is a mirror image Radau-Right formula where the weights and points are switched around. For Lobatto and Chebyshev (of the 2nd kind) points $W_0 = W_{n+1} > 0$. For symmetric problems, the boundary condition $du/dx = 0$ at $x = 0$ is automatically satisfied by symmetry, so a point is not used at the left end. At the right end $W_{n+1} = 0$ for Gauss points and $W_{n+1} > 0$ for Lobatto quadrature. Many claim the approximation of boundary conditions is improved by nonzero weights at the boundaries; however, we will show that the exact opposite is often true.

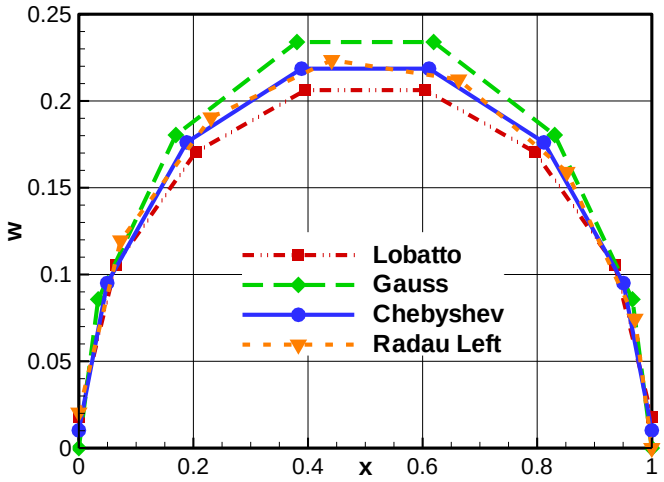


Fig. 1.5 Quadrature points and weights, n = 6

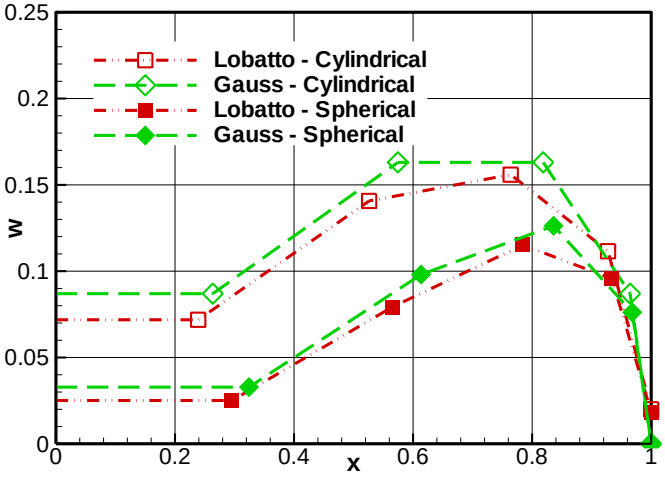


Fig. 1.6 Symmetric quadrature base points and weights

The various quadrature formulas can calculate exact integrals when the integrand of Eq. (1.6) is a polynomial of degree $2n$+1, $2n$, $2n$-1 and n+1 for Lobatto, Radau, Gauss and Chebyshev points, respectively. For problems with symmetry, the quadrature is exact for polynomials in x^2 of degree $2n$ and $2n$-1 for Lobatto and Gauss points, respectively. For Chebyshev points, the Clenshaw-Curtis (1960) quadrature is formally less accurate, which is certainly not obvious from examination of Fig. 1.5. Some claim it is just as accurate in many applications [Trefethen (2008)]. Clenshaw-Curtis quadrature is distinct from Chebyshev-Gauss quadrature which has accuracy O($2n$) when the radical $1/\sqrt{1-x^2}$ is included in the integrand of Eq. (1.6). Note, on the shifted interval [0,1] the radical is $1/\sqrt{x(1-x)}$; however, for purposes of discussion only the unshifted form is given.

1.1.2 A Note on Terminology. One goal of this monograph is to present a unified and consistent description of methods, and to show relationships between methods, including the equivalence of methods which are normally considered to be different. Also, this subject area can be a virtual minefield of conflicting terminology due to different naming conventions and so forth.

It is not surprising the terminology is confusing, since related methods have been developed in several different disciplines and different countries over a period of many years. Barriers exist due to language and lack of familiarity with the many branches of science, engineering and mathematics. A method may be invented or reinvented in one area when much development work has already occurred in a different discipline. It might eventually be discovered that the methods are related, or equivalent or in some cases identical. Even when the methods are shown to be identical, the naming conventions of methods may continue to be different. The proliferation of names is a major problem, since it causes confusion and inhibits communication.

For cases when different names exist for the same method, we have tried to list them all initially but then choose a single name. The choice is based partly on popularity, but with preference accorded to the name given originally. In most cases methods are considered the same, if they give the same answer, i.e. they are equivalent.

For example, some consider a *collocation* method to always use nodal trial functions. If modal trial functions are used it might be called a *pseudospectral* method. Since the names are different, you would think they are different methods. However, the basic method and the results are the same, although the results are in a different form. It makes more sense to call it either a nodal or modal collocation method. There is no need to use two entirely different names. The name collocation is preferred, since it predates the pseudospectral name by several decades. The name *interpolation method* was used even earlier, but outside the Russian literature it is rarely used and not sufficiently descriptive. The method is also called the *method of selected points* and the *matching points method*.

There are also the names *finite element*, *spectral element*, and *differential quadrature element* method. The only distinction is that finite element methods usually employ low order trial

functions, while spectral element methods often employ higher order trial functions. There is no clear line of separation. This difference is relatively minor, so there is little justification for multiple names. Using these different names, is like using entirely different names for second and fourth order finite difference methods. We prefer to use the term finite element for all three, since it is more popular and it predates the other two names by at least two decades. For clarity, a method could be called a low or high order finite element method or the specific order or degree could be used, e.g. Hermite cubic finite element.

To aid the reader, we have endeavored to include the alternate names for various quantities and italicize names when first presented and defined.

Designation of the roots or collocation points is one of the more confusing terminology issues one encounters. For example, the names for the roots considered are:

- **Chebyshev polynomials of the 1st kind** also called: *Chebyshev-Gauss* or abbreviated *CG* points or simply *Chebyshev points* or a *roots grid*
- **Chebyshev polynomials of the 2nd kind** also called: *Chebyshev Extrema* or *Chebyshev-Gauss-Lobatto* or abbreviated *CGL* points or simply *Gauss-Lobatto* or *Chebyshev* points
- **Legendre polynomials** also called: *Gaussian quadrature base points* or *abscissa* or *Legendre-Gauss* or abbreviated *LG* points or simply *Legendre* or *Gauss* points
- **Jacobi Polynomials ($\alpha = 0, \beta = 1$ or $\alpha = 1, \beta = 0$)** also called: *Radau* points, *Radau quadrature base points or abscissa*, *Legendre-Gauss-Radau* or abbreviated *LGR* points
- **Jacobi Polynomials ($\alpha = \beta = 1$)** also called: *Legendre Extrema* points or *Jacobi* points or *Lobatto quadrature base points* or *abscissa* or *Legendre-Gauss-Lobatto* abbreviated *LGL* points or simply *Gauss-Lobatto* or *Lobatto* points

Beware of the terms Chebyshev and Gauss-Lobatto since both are thrown around loosely for two different sets of points. In this monograph, we will not consider collocation at the roots of Chebyshev polynomials of the 1st kind, so *Chebyshev* with no clarification designates Chebyshev polynomials of the 2nd kind. The other points in Figs. 1.5 and 1.6 are called by the names found in most standard numerical analysis texts, i.e. *Gauss*, *Lobatto* or *Radau* (*left* or *right*) points [Hildebrand (1987)].

1.2 Historical Perspective

The *Method of Weighted Residuals* (*MWR*) is a family of methods for the approximate solution of differential equations [Crandall (1956), Ames (1966), Finlayson (1972, 2014)]. Although nonlinear assumed forms are possible, one usually approximates the solution with a linear combination of trial functions like Eqs. (1.1), (1.2), or (1.3). For example, suppose we seek a solution of the equation:

$$u_{xx} + g(x, u, u_x) = 0 \tag{1.7}$$

for $0 < x < 1$, with $u(0) = u_0$ and $u(1) = u_1$. The residual is formed by substituting the approximate solution, Eq. (1.1), into the differential equation:

$$R(x, \boldsymbol{a}) = \sum_{k=1}^{n} a_k \psi_{kxx} + g(x, \tilde{u}, \tilde{u}_x) \tag{1.8}$$

If the residual is zero for all x, an exact solution has been achieved. In general, this will not be possible, so the parameters, $\boldsymbol{a}$, are selected so the residual will be small in some sense. If the residual is small for all x, Eq. (1.1) is a good approximation to the true solution. MWR achieves this goal by forcing the function to zero in a weighted average sense:

$$\int_0^1 w_k(x) R(x, \boldsymbol{a}) dx = 0 \text{ for } k = 1, \ldots, n \tag{1.9}$$

The w_k are called *weight* or *test* functions. Different choices for the n weight functions lead to fundamentally different forms of MWR. Finlayson and Scriven (1966) review the early development of these basic procedures.

We will consider the following methods:

1. $w_k = \psi_k(x)$ or $\partial \tilde{u}/\partial a_k$, Galerkin method
2. $w_k = \partial R/\partial a_k$, least squares
3. $w_k = x^{(k-1)}$ or $P_k(x)$, method of moments
4. $w_k = 1$ for x in Ω_k, subdomain
5. $w_k = \delta(x - x_k)$, collocation method

Several texts include the *tau* method of Lanczos (1938, 1956) as a MWR. The tau method can be equivalent to a number of these methods, but is usually equivalent to the method of moments, see section 1.2.7.

The MWR were created in the early 20th century to generalize the Rayleigh-Ritz variational procedure, which has limited applicability. The name MWR came about from discussions at the First International Congress on Applied Mechanics held in 1924 in Delft, Netherlands. Following a presentation by Biezeno (1924) on the subdomain method, Courant (1924) commented that the subdomain method was an especially simple special case of a method where the basic principal is that "weighted averages of the residuals" should vanish. Crandall (1956, p. 147) was first to illustrate the methods in a unified way, restating Courant's observation. These comments led to the name *Methods of Weighted Residuals* (or *Method of Mean Weighted Residual* or *Weighted Residual Method*). Collatz (1961) called them *error distribution principals.* The principal of error distribution figures prominently in discussions of Lanczos (1956) and Ames (1965). This principal distinguishes the methods from Taylor series which concentrates the accuracy at a single point. The early literature sometimes called them *direct methods*. They are also called *projection methods*, *orthogonalization methods* or *orthogonal projection methods*.

For practical purposes, the popular name *Spectral Method* has become synonymous with MWR. The name seems to have originated in the study of atmospheric modeling. In Silberman (1954) spherical harmonic trial functions were used to describe atmospheric waves. The name *spectral method* was not used, but later papers discussed the "*spectral form of the vorticity equation*" [Platzman (1960)]. The approach eventually became known as a *"spectral method"* to distinguish it from finite differences [Ellsaesser (1966)]. Later, the name was retained for other types of trial functions, such as Fourier series and Chebyshev polynomials [Orzag (1972, 1974)] and for Legendre polynomial and nodal trial functions [Canuto, *et al*. (1988), Trefethen (2000)]. The name is usually associated with global trial functions which are applied with Galerkin, tau and *pseudospectral* methods. The pseudospectral method is just another name for the collocation method and section 1.2.7 shows the tau method is equivalent to the method of moments.

1.2.1 Rayleigh, Ritz, Bubnov and Galerkin Methods. Variational calculus dates from the time of Euler and Lagrange in the eighteenth century, but until the work of Walther Ritz (1908, 1909) it had received little use as a procedure to approximate the solution of differential equations. The variational method associates the differential equation with the Euler equation for a functional. Minimization or maximization of the functional with respect to a trial space gives an approximate solution to the differential equation.

Ritz used this idea to tackle the problem of an elastic plate clamped on both ends for a 1907 competition of the Academy of Science in Paris. He followed it up with solution of the Poisson problem and an eigenanalysis of the problem of the Chladni figures (Gander and Wanner, 2012). He used variational principals to derive elegant solutions to these problems. Ritz contracted tuberculosis in 1900 and had never recovered. He completed his epic papers shortly before his death in 1909 at the age of 31.

Long before the work of Ritz, Lord Rayleigh (John William Strutt) had used variational methods to estimate the principal vibration modes of elastic strings, bars, membranes and plates in his classic book *The Theory of Sound* (Rayleigh, 1877) and other publications. Rayleigh (1911) wrote a paper congratulating Ritz on his work, but claimed it was not a new idea, since he had already used the method long before. Rayleigh's response cast a cloud over Ritz' work.

Since the work of Ritz, the method is sometimes called the Ritz method at other times the Rayleigh-Ritz method. Lately, naming of the method has come under question. After a thorough study of the original publications, Leissa (2005) concluded that Rayleigh's contribution was not significant enough to warrant the inclusion of his name in designation of the method. Leissa writes:

> It concludes that, although Rayleigh did solve a few problems which involved minimization of a frequency, these solutions were not by the straightforward, direct method presented by Ritz and used subsequently by others. Therefore, the present writer concludes that Rayleigh's name should not be attached to the Ritz method; that is, the Rayleigh–Ritz method is an improper designation.

Others offer a different viewpoint [Finlayson [(1972,2014), Ilanko (2008)]. For example, Finlayson writes in the main text and a footnote (pp 223-4):

The variational method is known as the Rayleigh-Ritz method, although some authors call it the Ritz method. Close examination of Rayleigh's works indicates Rayleigh used the variational method to calculate successive approximations to both boundary and eigenvalue problems. For example, Rayleigh (1873) treats eigenvalues which are stationary, although he does not give any approximate method of determining them. He added the comment (Rayleigh, 1896, p. 110) that the eigenvalue is a minimum (the equilibrium is absolutely stable) and gives a method for approximating the eigenvalues. In 1899 Rayleigh clearly applies the variational method to calculate the frequency of vibration of a fluid partially filling a horizontal cylinder, including a second approximation (Rayleigh, 1899). Even earlier Rayleigh (1871) applied the variational method to a boundary value problem. He calculated the approximate solution for flow through a cylindrical hole in a flat plate, including a second approximation, using the variational principle. See also Rayleigh (1896, Section 307 and Appendix A). Rayleigh himself (1911) commented on Ritz's paper (1908), saying he was surprised that Ritz thought his method new. As Courant (1943) remarked, it was probably the tragic circumstances of Ritz's work that caught the general interest. Ritz wrote his papers (1908, 1909) while aware that he was soon to die of tuberculosis. The variational method is also more clearly presented in Ritz's papers.

One convention is to call it the Rayleigh-Ritz method when it is used with a single term, e.g. to estimate a principal eigenvalue, and to call it the Ritz method when multiple terms are used. However, Rayleigh's applications were not restricted to single term approximations. Clearly, Ritz' work demonstrated for the first time, a straightforward application of variational principals to solve differential equations. For many problems, the method converges so fast that only a few terms are required to obtain an accurate solution, so the method is useful even when one is restricted to hand calculations.

For some time, the significance of Ritz' work was not recognized in western Europe and America, but it was used immediately in Russia. The method was first used there by Timoshenko (1910). Then, Ivan Bubnov, a Russian engineer involved in ship building, used the method in designing the hulls of submarines (1913,1914).

Ritz' work was a major breakthrough, but since some problems cannot be based on a variational principal, the method's applicability is limited. The MWR were developed to fill the need for methods which are generally applicable. The Galerkin method was the first such method and is a direct generalization of the Rayleigh-Ritz method. Boris Galerkin was another Russian engineer working with steam locomotives. He weighted the residual by the trial functions. For problems which can be treated with a variational procedure, the Galerkin method is equivalent. It was described by Galerkin (1915) in a study of elastic equilibrium and the stability of rods and plates. The earlier paper by Bubnov (1913) had many similarities, but Galerkin was the first to describe a method without any reliance on a variational principal. Due to these relationships it is called the Bubnov-Galerkin method in Russian literature and often the Rayleigh-Ritz-Galerkin method in western literature. The name or names used to designate the method seem to correlate with the nationality of the author. The Galerkin method is the most heavily used MWR and can be considered the gold standard for these methods.

The Galerkin method is usually implemented in its *weak form*, like the Rayleigh-Ritz method. Substitution of the residual of Eq. (1.7) into Eq. (1.9), weighting by the trial functions and then integrating by parts produces:

$$\int_0^1 \psi_k(x)R(x,\boldsymbol{a})dx = \sum_{j=1}^{n} a_j \int_0^1 \psi_k \frac{d^2\psi_j}{dx^2}dx + \int_0^1 \psi_k g(x,\tilde{u},\tilde{u}_x)dx$$
$$= \psi_k \frac{d\tilde{u}}{dx}\Big|_0^1 - \sum_{j=1}^{n} a_j \int_0^1 \frac{d\psi_k}{dx}\frac{d\psi_j}{dx}dx + \int_0^1 \psi_k g(x,\tilde{u},\tilde{u}_x)dx \quad (1.10)$$

This is called the *weak form* since the trial function continuity requirements are reduced. For the Galerkin method with Dirichlet conditions, the trial solution must obey the boundary conditions, so for the conditions on Eq. (1.7) the boundary terms are zero. The second derivative term is called a *bilinear form*. Of course, integration by parts does not alter the result, but it makes evident the symmetry of the second derivative term. For multidimensional problems, the integration by parts generalizes to the divergence theorem.

A variational or Rayleigh-Ritz method is not normally considered a MWR. However, since the Galerkin method is a generalization of the Rayleigh-Ritz method, variational methods are in effect included. Many authors do not make a distinction and use the names variational method and Galerkin method interchangeably. To be correct, a method should not be called a variational or Rayleigh-Ritz method unless it can be based on variational calculus.

1.2.2 Least Squares. Least squares is a very old idea for approximating functions. Here, it is used to force the residual to approximate zero. In the context of solving differential equations, Finlayson and Scriven (1966) attributed its first use to Picone (1928). However, we note the method was discussed in an earlier paper by Krylov (1926).

The least squares method works well for simple differential equations. However, for equations with nonlinearity and many terms it is more difficult to implement. The method is also problematic for eigenvalue problems, since linear equations become nonlinear eigenvalue problems [Finlayson and Scriven (1966)]. For these reasons, its popularity has waned as more complex problems have been tackled.

1.2.4 Method of Moments. In western literature, the method of moments is usually described as the use of monomial weights in Eq. (1.9) , i.e. $w_k = x^{(k-1)}$, but the weights are sometimes given as $w_k = P_k(x)$, where the P_k are successive members of a complete set [Ames (1966), Villadsen (1970)]. The method is actually more general.

The basic ideas behind the method of moments were first presented by N.M. Krylov at a session of the All Ukraine Academy of Sciences in 1926 [Krylov (1926,1961), Lucka and Lucka (1992)]. The formulation of the method permits a great deal of flexibility. In fact, in a limited way, it includes the least squares and Galerkin methods as special cases. Originally, given some linear differential operator, M, the weight functions are chosen as $w_k = M\psi_k$. The linear operator, M, has the same order, but can be different from that of the equation being solved. The least squares method is produced if linear operator, M, is the same as that of the equation being solved. Weighting by, ψ_k, the eigenfunctions of a linear operator gives the Galerkin method, since $M\psi_k = \lambda\psi_k$.

The method of moments was thoroughly studied by Mykhailo Kravchuk (1926,1932) in fundamental studies published from 1926-1937. The basic results of these studies are summarized by Lucha and Lucha (1992) (including 18 Kravchuk references, mostly in Ukrainian or Russian). Kravchuk spent much of his short and tragic career studying the various conditions on the linear operator and trial functions and resulting convergence properties. Shuleshko (1959) provided some additional elaboration and extensions of the method.

Since Legendre or other Jacobi polynomials are eigenfunctions of a Sturm-Liouville problem, weighting by Jacobi polynomials is a method of moments. Of course, if the residual is orthogonal to Legendre polynomials it is also orthogonal to all the monomials, $x^{(k-1)}$, up to the same degree.

The idea of using monomials is apparently due to Yamada (1947) as described by Fujita (1953) in his study of diffusion in gels. Weighting by monomials can lead to ill conditioned approximations and some have rejected the method of moments for that reason [Boyd (2000) p. 62]. In most western literature today, the method of moments is generally considered to be weighting by monomials, but in the current monograph we include equivalent methods, such as weighting by Legendre polynomials.

An exception to the statement above is the use of the method of moments in electromagnetic field calculations. It has received extensive use in that field of study [Harrington (1968,1990)]. The name is used like MWR, an umbrella term used to cover several methods, including the Rayleigh-Ritz, Galerkin and collocation methods. However, the field has developed its own terminology, e.g. the collocation method is usually called the *point matching method*, while finite element methods are called the *method of subsections*.

A basic idea behind the Galerkin and moments methods are that if the weight functions are members of a complete set, then as $n \longrightarrow \infty$ the residual must converge to zero. Of course, only a finite number of trial functions are used in practice, but the property of completeness is important for insuring convergence of the sequence of approximations.

1.2.3 Subdomain Method. The subdomain method was first described by Biezeno and Koch (1923,1924) and prompted Courant's (1924) observations concerning the common principle of this and other methods, leading to the name MWR. In the subdomain method the weight functions are unity in subdomains of the problem. The differential equation is satisfied on the average in the n subdomains. As n is increased, the differential equation is satisfied in

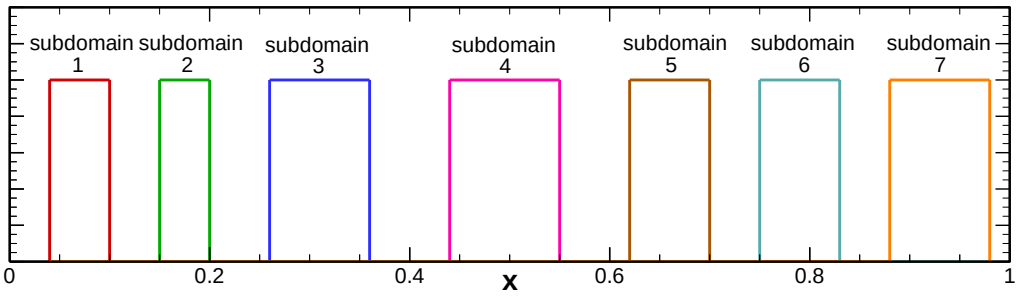


Fig 1.7 Weights used in subdomain method

a larger number of smaller and smaller subdomains. The intersection of the subdomains usually fills the problem domain, but this is not a requirement. An example of subdomain weight functions is illustrated in Fig. 1.7.

1.2.5 Collocation. The collocation method dates from the 1930's. It is similar to the subdomain method, but the subdomains are shrunk to zero width with unit volume. Thus, the weight functions are represented by Dirac delta functions. Integration is not required, the residuals are set to zero at n collocation points. As n is increased the residual is zero at more points. An example with six collocation points is illustrated in Fig. 1.8.

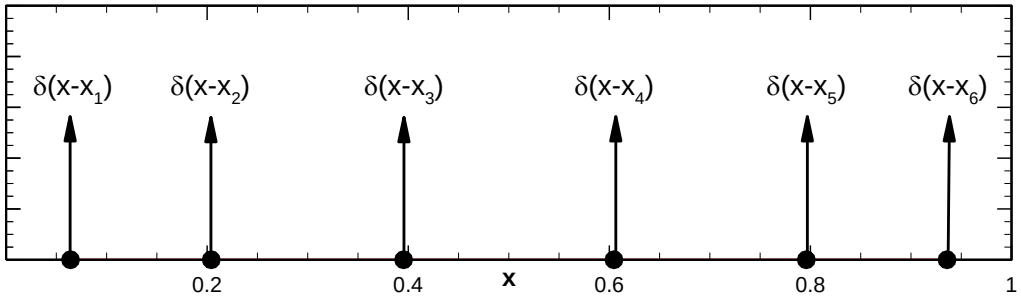


Fig 1.8 Weights used in collocation method

Early work on the method is due to Slater (1934), Kantorovich (1934), Frazer, *et al.* (1937) and Lanczos (1938). Kantorovich called it the *interpolation* method, Lanczos called it the *method of selected points*, while Frazer, *et al.* applied the name *collocation,* citing its definition in the Oxford dictionary. Using all three names we can say the method *interpolates* the residual to zero at *selected collocation points*. Since integration is not required, it is simpler to apply than the other methods which require integration. Simplicity is its most attractive feature.

Slater applied the method to study energy bands in metal. A translation of the Kantorovich article is not available, but it is discussed in other articles. Frazer, *et al.* compared equally spaced collocation to the Galerkin and least squares methods on several problems. Frazer, *et al.* argued that the method would converge whenever the Galerkin or least squares methods converge. However, Karpilovskaya (1963) points out flaws in their argument.

Karpilovskaya (1953,1963) was the first to present proof of convergence of the collocation method. His proof is discussed by Kantorovich and Akilov (1964, p. 581). The proof relies on the assumption that the collocation points are roots of an orthogonal polynomial. Since collocation is basically interpolation of the residual, the Runge (1901) phenomenon can occur, e.g. with equally spaced points (see Fig. 2.36). A good example of problems associated with equally spaced collocation can be found in Bert and Malik (1996). Frazer *et al.* were apparently aware of the Runge phenomenon, but did not observe it for their problems, probably because high order approximations were not calculated.

Lanczos used Chebyshev polynomial trial functions and collocated at the roots of Chebyshev polynomials of the 2nd kind. Clenshaw and Norton (1963) collocated at the same points and discuss several implementation details for nonlinear problems. Wright (1964) collocated at the roots of Chebyshev polynomials of the 1st kind, arguing that this choice would minimize the

maximum deviations of the residual from zero. However, a uniform distribution of the residual does not produce a uniform error in the solution. Villadsen (1970) thoroughly reviews other early Chebyshev applications. The use of Chebyshev roots was an important advance, since the Runge phenomenon does not occur.

Villadsen and Stewart (1967) made a further improvement by selecting the roots of Jacobi polynomials that are the base points of accurate numerical integration schemes, e.g. Gaussian or Lobatto quadrature. They chose the name *Orthogonal Collocation* for their variation on the method. They explained that collocation at Lobatto points was equivalent to numerical integration of the Galerkin method. An accurate (sometimes exact) approximation of the Galerkin method is achieved. They also improved the method by formulating it with nodal values rather than polynomial coefficients. This modification produced a more intuitive, finite-difference-like method and facilitated automation of the method for general applications. The title of their article is – *"Solution of Boundary-Value Problems by Orthogonal Collocation."* Despite the title, they demonstrated the method for not only a boundary value problem, but also the parabolic transient catalyst pellet problem, eigenvalues for the Graetz problem and the elliptic 2-dimensional Poisson equation. The method was demonstrated for all but hyperbolic equations.

In the 1970's, the development of collocation methods occurred in three threads: (1) Orthogonal Collocation (OC), (2) Pseudospectral (PS) and (3) Differential Quadrature (DQ). The OC thread started with the Villadsen and Stewart article, with further development reported in Villadsen (1970), Finlayson (1972) and Villadsen and Michelsen (1978). These references describe collocation at Gauss, Lobatto and Radau points. Collocation at Gauss points was shown to be equivalent to numerical integration of the method of moments. In addition to standard cartesian coordinates, they describe methods for symmetric problems in planar, cylindrical and spherical coordinates. Nodal differentiation matrices are used exclusively. These developments led to a flurry of activity applying the method to different problems [see references in Michelsen and Villadsen (1972, 1981), Finlayson (1974)]. Many of these early papers demonstrated very favorable comparisons of the method with finite differences. A later text describing the method is that of Finlayson (2003).

The Pseudospectral thread began with the work of Orzag (1972), with further developments by Gottlieb and Orzag (1977). The name pseudospectral is normally synonymous with collocation, but occasionally it is used to describe other approximations to exact MWR. Orzag applied collocation to a periodic problem using trigonometric trial functions. He also considered a nonperiodic linear first order hyperbolic problem with collocation at the roots of Chebyshev polynomials of the 2nd kind. For the nonperiodic problem Chebyshev trial functions were used, and collocation gave an accurate approximation of the Galerkin method. Nodal approximations were not considered. An important contribution of this work was the use of fast Fourier transforms (FFT) to perform the calculations. Orzag was apparently unaware of the earlier work of Villadsen and Stewart, as it was not referenced.

FFT was the spark that ignited work in the PS thread. Subsequent work investigated various methods of solution using FFT. A strong mathematical foundation for the method was laid and the solution of periodic problems was highly developed. Most of the early work on nonperiodic problems employed collocation at Chebyshev points with a modal approach using Chebyshev trial functions. Some later applications have used nodal trial functions and collocation at Lobatto (or LGL) points. Some excellent texts describe the method, solution algorithms and applications [Canuto, *et al.* (1988, 2006, 2007), Trefethen (2000), Boyd (2000), Peyret (2002), Shen, *et al.* (2011)].

The Differential Quadrature Method originated with Bellman and Casti (1971), Bellman, *et al.* (1972), and Bellman (1973, p. 244). The first was a short paper advancing the idea of using a nodal differentiation matrix in the solution of differential equations. The second paper developed the method further and demonstrated it for both first and second order equations in one dimension, collocating at Gauss points. However, the paper provided few details regarding boundary condition treatment. Bellman, *et al.* (1972) developed the differentiation matrices for collocation at Gauss points. The use of a nodal differentiation matrix was not a new idea. Nielsen (1956) presents formulas for them with arbitrary nodal locations. By 1972, nodal collocation at Gauss points was well established in the OC literature. Apparently, unlike the OC implementation, the DQ method of Bellman, *et al.* did not use boundary points in order to incorporate boundary conditions [Quan and Chang, 1989)]. Nevertheless, DQ was adopted for many engineering applications. Bert and Malik (1996) and Bellomo (1997) give excellent reviews of developments in DQ. Early development of the method was independent of work in the OC and PS communities. Apparently, it was not recognized as a form of collocation until the late 1980's [Quan and Chang, 1989)]. A reference text devoted to DQ is that of Shu (2000).

Some ideas have been exchanged between the different threads of development, but there have also been many cases of rediscovery and failure to give proper credit to an idea or method. Chebyshev points have historically been popular in PS applications and are used in DQ as well. Gauss points, also called Legendre-Gauss or LG points, are popular in OC applications, but not as popular in the PS and DQ communities. Lobatto points, also called Legendre-Gauss-Lobatto or LGL points, have become popular in the PS literature. Nodal formulations are popular with "the engineers" due to its similarity to finite differences. Nodal approximations are used almost exclusively in OC and DQ applications, while PS applications use a mix of modal and nodal trial functions. Villadsen and Stewart were the first to use a nodal implementation, collocating at Gauss and Lobatto points, but outside the OC community they are not credited for these innovations. Symmetric problems in planar, cylindrical and spherical coordinates are common in OC applications, while problems with periodic solutions are covered almost exclusively in the PS literature.

1.2.6 Other Weight Functions. Some authors take a more general view and refer to virtually all MWR weightings as Galerkin or variational methods. Alternative weighting schemes are often called Petrov-Galerkin methods [Petrov (1940)]. However, the name Petrov-Galerkin is also used to describe a method of upwinding for convection dominated flow problems. Some

authors use the term Petrov-Galerkin for all but the Galerkin and collocation methods. Many of these methods are actually methods of moments.

For Chebyshev methods, the weights in Eq. (1.9) usually include the radical $1/\sqrt{1-x^2}$. This term is necessary to exploit the orthogonality of the polynomials and to achieve accurate integration with Gauss-Chebyshev quadrature. In naming methods, most authors do not distinguish between methods which include the radical. It would be more appropriate to add Chebyshev to the name, e.g. *Chebyshev-Galerkin* method. The naming of methods with alternative weight functions is yet another area where terminology can be confusing. In this monograph, methods which include the radical will not be used.

1.2.7 Tau Method. The *tau* method was originated by Lanczos (1938, 1956) as a method for approximating functions which can be described by differential equations. The basic idea behind the method is that rather than find an approximate solution to the differential equation, why not find an exact solution to an approximate differential equation? For simple problems, he represented the approximate differential equation as the original one with the addition of a small perturbation term of the form $\tau P_m(x)$, where τ is a constant and $P_m(x)$ is a high order orthogonal polynomial, i.e. $m \approx n$. Lanczos discussed using Chebyshev polynomials (2nd kind) and Legendre polynomials, but others could be used. Depending on the complexity of the problem, several terms may be required. These perturbation terms are the residual as defined by Eq. (1.8), so in general Lanczos' idea was to set:

$$R(x, \boldsymbol{a}) = \tau_0 P_m(x) + \tau_1 P_{m+1}(x) + \cdots \tag{1.11}$$

where the number of terms required depends on the nature of the differential equation. Lanczos describes a recursive procedure for determining the values of the τ parameters and the coefficients in the approximation, i.e. a in Eq. (1.1).

The tau method was further developed in a series of articles by Ortiz and coworkers [e.g. Ortiz (1969,1975)]. Several solution methods are described which do not involve integration or orthogonalization as required by most MWR. Later papers [El-Daou, *et al.* (1993), El-Daou and Ortiz (1997)] explore the equivalence between the tau method and other methods. The relationship between the tau method and the integrated MWR is obvious. Since most MWR make the residual orthogonal to the first few members of a complete set, the residual is proportional to the first neglected term and possibly higher terms. If tau methods are defined to include all methods with a residual term like Eq. (1.11), then virtually all MWR (including the Galerkin method) would be tau methods. This idea seems silly. Instead, we view the tau method as an alternate solution procedure for MWR - one which does not require the evaluation of integrals. Indeed, alternative solution algorithms have been described for Galerkin and collocation methods [El-Daou and Ortiz (1994, 2007)]

The lack of a need for integration is the distinguishing feature of the method as presented by Lanczos, Ortiz and coworkers. The tau method as presented in the spectral literature [e.g. Gottlieb and Orzag (1977, p. 11), Canuto, *et al.* (1988, p. 10, 2006, p. 21), Boyd (2000, p. 473)] bears little resemblance to the method as described by Lanczos (1956). Unlike the algorithms

of Lanczos, Ortiz and coworkers, which require no integration, the spectral literature describes a MWR which requires integration. For purposes of discussion, call the two approaches the *Lanczos-tau* method and the *spectral-tau* method. The spectral-Legendre-tau method makes the residual orthogonal to the first few Legendre polynomials, so it is the same as the method of moments. The spectral-Chebyshev-tau method makes the residual orthogonal to Chebyshev polynomials with the additional weight factor of $1/\sqrt{1-x^2}$ [Johnson (1996)], i.e. a Chebyshev-moments method.

The spectral-tau method satisfies boundary conditions by using side conditions, creating a messy matrix structure. Whether the boundary conditions are satisfied by side conditions or whether they are built into the trial functions makes no difference in the final result [Boyd (2000), pp 111-115]. It makes no sense to call it the tau method when side conditions are used and the method of moments when the boundary conditions are built-in. This difference in method of imposing boundary conditions is trivial compared to the principal difference between the Lanczos-tau method and spectral-tau method, i.e. whether integration is required. The messy matrix structure can be avoided. Section 3.1.2 describes a nodal form of the method giving a symmetric matrix, while section 3.1.7 describes a modal formulation with a near upper triangular matrix.

The Lanczos-tau method offers an interesting approach and philosophy. Its view of the residual is revisited in the discussion of the examples in section 3.1.6. However, the spectral-tau method is identical to the method of moments. Since the method of moments is more general and it predates the tau method, we will use the traditional name *method of moments* in this monograph. However, the reader should be aware that the tau method is equivalent.

Since the residual for MWR with polynomial trial functions can be represented by Eq. (1.11), the distribution of residual errors can be shifted by the choice of weighting function, cf. the polynomials in Fig. 1.4. A few studies have investigated the influence of the orthogonal polynomial selected on the accuracy of the method [Lanczos (1972), El-Daou and Ortiz (1992), Namasivayam and Ortiz (1993)]. These error analyses give insight into the differences between MWR weighting methods: Galerkin, moments, etc. They apply to MWR as well as the tau method given the relationship between the two.

1.2.8 Boundary Conditions. Descriptions of MWR distinguish between three different approaches: (1) *interior methods*, (2) *boundary methods* and (3) *mixed methods*. An interior method employs trial functions which obey the boundary conditions, so the parameters are chosen to approximate the differential equation. In boundary methods, the trial functions obey the differential equation and the parameters are chosen to approximate the boundary conditions. In mixed methods, the parameters are adjusted to approximate both the differential equation and boundary conditions. The foregoing sections have primarily addressed approximations to the differential equation, which is usually the biggest challenge.

In order to describe methods for treating boundary conditions, consider the following problem:

$$Lu = f \tag{1.12}$$

In the region Ω, with boundary conditions on $\partial\Omega$:

$$Bu = 0 \tag{1.13}$$

where L and B are linear operators. When the boundary conditions are approximated, the residual of the boundary conditions may be required to meet their own independent weighted residual criteria. Shuleshko (1959) describes a method of this type, which is like:

$$\begin{aligned} &\int_{\Omega} w_k(L\tilde{u} - f)dV = 0 \ \text{ for } \ k = 1, \dots, n_1 \\ &\int_{\partial\Omega} w_k B\tilde{u}\, dS = 0 \ \text{ for } \ k = n_1 + 1, \dots, n \end{aligned} \tag{1.14}$$

Alternatively, the boundary conditions may be enforced with criteria based on a weighted combination of interior and boundary residuals. Kravchuk studied a method of this type [Lucka and Lucka (1992)], which is like:

$$\int_{\Omega} w_k(L\tilde{u} - f)dV + \varrho_n^2 \int_{\partial\Omega} \hat{w}_k B\tilde{u}\, dS = 0 \ \text{ for } \ k = 1, \dots, n \tag{1.15}$$

Where $\varrho_n$is a parameter that depends on n and $\varrho_n \longrightarrow \infty$ as $n \longrightarrow \infty$. With this approach, the boundary condition is satisfied approximately along with the differential equation. Both residuals are driven to zero as $n \longrightarrow \infty$, so in the limit the differential equation and boundary conditions are satisfied. This type of treatment has more recently been called a *penalty method*.

For the moments (or tau) method, boundary conditions are usually satisfied exactly. They may be satisfied through the selection of the trial functions or by using side conditions [Boyd (2000), pp 111-115].

Due to the relationship between the Galerkin method and Rayleigh-Ritz variational method, boundary conditions can be treated in the same way. Boundary conditions can be divided into two types – *essential* boundary conditions and *natural* boundary conditions [Finlayson (1972)]. For second order equations, Dirichlet conditions or other conditions on the value of the solution variable are essential and must be satisfied exactly. Conditions of the 2nd or 3rd type (Neuman or Robin) or others involving derivatives or fluxes are natural boundary conditions. For fourth order problems, conditions on the first derivative are also essential, while those involving the second or third derivatives are natural.

Natural conditions can be handled in one of two ways. They can be satisfied exactly, or they can be approximated. The natural or approximate treatment is also called a *weak* formulation of the boundary conditions. Exact satisfaction of the boundary condition is called a *strong* treatment or *boundary collocation* since, like collocation in the interior, the residual of the boundary condition is zero. Later chapters will show that the natural or weak treatment of boundary conditions is of the combined type, like Eq. (1.15), and as n is increased both interior and boundary residuals converge to zero exponentially.

With the natural treatment the conditions at the boundary are incorporated into the approximation. Consider a Robin or third type boundary condition:

$$\left.\frac{du}{dx}\right|_{x=1} = -h(u(1) - u_1) \tag{1.16}$$

These conditions frequently occur in problems of heat and mass transfer to represent an external resistance, such as heat transfer through a boundary layer, h represents a heat transfer coefficient. The natural treatment provides a means to incorporate the impact of the surroundings on the system of interest. The approach is especially useful when the boundary is governed by another differential equation. The Dirichlet condition, $u(1) = u_1$, is the degenerate or limiting case $h \to \infty$. The Dirichlet condition in the Rayleigh-Ritz or Galerkin method is treated like a degenerate natural boundary condition.

Although not a fundamental requirement, collocation implementations normally use a strong treatment of boundary conditions [e.g. Finlayson (1972), Villadsen and Michelsen (1978), Bert and Malik (1996), Bellomo (1997), Trefethen (2000), Boyd (2000), Peyret (2002)]. Note that these references include the OC, PS and DQ threads of development. Although not widely publicized, early applications of OC with Lobatto or LGL points uncovered inaccuracies with flux boundary conditions [Ferguson and Finlayson (1970), Ferguson (1971), Finlayson (1971,1972), Elnashaie and Cresswell (1973)]. Collocation at Gauss or LG points was the recommended remedy since the problem occurs only when quadrature weights are nonzero at the boundary (see Appendix B). This experience belies the commonly held notion that nonzero boundary weights somehow aid in the approximation of boundary conditions.

Ferguson (1971) described an integral procedure which circumvented the problem with Lobatto points, but it is cumbersome, and its applicability is limited. Young (1977) suggested that a natural boundary condition treatment would be more appropriate. Later, this idea appeared in the spectral literature [Canuto, *et al.* (1988, 2006), Funaro (1992), Shen, *et al.* (2011)], but these texts also cover the strong treatment, and none claim a significant benefit to a weak formulation. Consequently, most applications use a strong treatment or boundary collocation for all boundary conditions. More recently Young (2019) presents compelling evidence supporting a natural or weak treatment. The examples in later chapters provide a close look at this issue.

1.2.9 Finite Elements. As stated above, a MWR is called a finite element method when it is used with piecewise continuous trial functions in subdomains or elements. The beauty of this approach is that the elements can be adapted to all sorts of irregular boundaries, e.g. an airplane wing or an oil well with multiple branches resembling the roots of a tree. The simplest trial functions of this type represent the solution as its linear interpolant, like Fig. 1.3, but elements of higher order can easily be constructed.

Over the years there has been much discussion about the origins of the finite element method and several papers have been written on the subject [Oden (1987), Gupta and Meek (1996), Clough (2004), Gander and Wanner (2012)]. However, before getting into that discussion, we

need to be more explicit about what constitutes a finite element method. It should be defined to include two components: (1) solution of a differential equation using a variational method or MWR and (2) trial functions that are piecewise continuous in subdomains. Some have cited early uses of triangles to calculate areas of irregular objects, but this example does not meet condition 1. Others have cited Ritz' work with global trial functions, which does not meet condition 2.

Many cite a presentation given by Courant in 1941, which was published later with an appendix added [Courant (1943)]. The paper has a nice description of the Rayleigh-Ritz method and one of the better discussions of the role of natural boundary conditions. The appendix presents an example of a plane torsion problem for a hollow square bar, such that an element of symmetry is of trapezoidal shape. Gupta and Meek (1996) give a detailed discussion of the calculations in the appendix. Courant first treats the problem using the Rayleigh-Ritz method with one and two term global trial functions. Then, he checks those results with a "*generalized method of finite difference*" using several small grids of triangles. About the results, he states:

> "…. [the generalized finite difference method] is obviously adaptable to any type of domain, much more so than the Rayleigh-Ritz procedure in which the construction of admissible functions would usually offer decisive obstacles."

Few details are given regarding the generalized finite differences, but this statement seems to indicate it is not based on the Rayleigh-Ritz method. However, closer examination of the article shows that it is based on the Rayleigh-Ritz variational procedure applied to linear triangular approximations. For example, after describing the triangular based trial functions, he states:

> "Our integrals become finite sums, and the minimum condition will be equations for the values of ϕ in the net points [i.e. the solution nodal values]."

This statement shows he is performing integrations and using minimization as part of the Rayleigh-Ritz method. Apparently, Courant considered the method to be different from the Rayleigh-Ritz method, whereas, it is an application of the same method with a different type of trial functions. He chose the misleading name *generalized finite differences*. It is also unfortunate that details of the method were not given. So, although the presentation is obscure and details are absent, it appears the method in Courant's paper meets both conditions to qualify as a finite element method.

Courant clearly promotes triangular grids for domains of irregular shape. About the same time, there were some crude methods developed to perform structural analysis on lattice networks [Oden (1987)].

Despite Courant's stature, his paper had little impact on development of the finite element method. Development began in earnest after digital computers became available in the mid-1950s and by that time Courant's paper was largely forgotten. Work at Boeing and the University of Washington lead to the paper of Turner, Clough, Martin and Topp (1956), which many cite as a "first" finite element paper. The finite element name was first used in a paper by Clough (1960). These early papers apparently based the approximations on heuristic

arguments. It wasn't until the paper by Melosh (1963) that the approximations were shown to be equivalent to the Rayleigh-Ritz method, which helped to establish a better mathematical basis for the method. There were many other contributors to early development of the method which can be found in the references cited. Of particular interest here is the use of numerical integration, which was first described by Bruce Irons (1966).

As the foundation of the method became more firmly established, books began to appear [Zienkiewicz (1971), Strang and Fix (1973)]. However, learning the method from Zienkiewicz' book was not easy for one unfamiliar with structural mechanics. Most of the early applications in structural mechanics were linear steady state problems which could be treated with the Rayleigh-Ritz method. The success of the method in structural mechanics lead to an interest in other application areas. Many of these areas, such as fluid mechanics and flow in porous media are nonlinear and time dependent. The numerical integration methods used with finite elements are calculation intensive. The simplicity of collocation-like methods seemed like an obvious improvement for such problems.

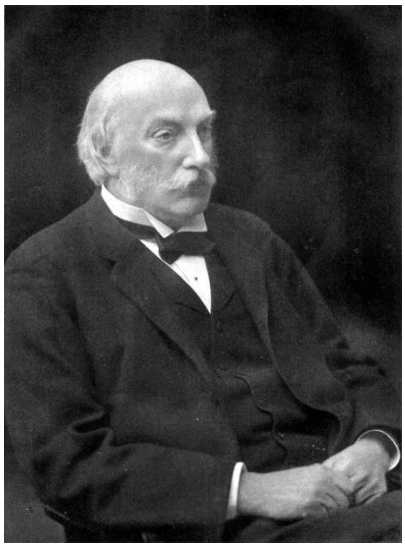
Lanczos (1956) and Villadsen and Stewart (1967) considered only global approximations. However, Villadsen and Stewart's idea of collocating at quadrature points was soon used to extend OC for use with finite elements. First and most popular is collocation at Gauss points with early articles by DeBoor and Swartz (1973), Douglas and Dupont (1973), and Carey and Finlayson (1975). Finite element collocation at Gauss points is a method with continuous derivatives (C^1 continuity) at element interfaces. Several texts are available which describe the method [Davis (1984), Lapidus and Pinder (1999), Finlayson (2003)].

Collocation at Lobatto points extends to a finite element method with simple C^0 continuity. The *Hybrid-Collocation-Galerkin* method was developed first at Rice University by Henry Rachford, Mary Wheeler, Julio Diaz and others [Diaz (1975,1977), Dunn and Wheeler (1976), Wheeler (1977)]. About the same time, the *Lobatto-Galerkin* method was developed independently by three others [Gray (1977), Young (1977,1981), Hennart (1982)]. For many years, these two methods were considered different, but, in fact, they are equivalent [Young (2019)]. Leyk (1986,1997) appears to have developed the same method by reformulating the hybrid-collocation-Galerkin method. Later, the method was developed yet again by Maday and Patera (1989) and popularized as the *Spectral/hp Element* method. Several texts and monographs describe the method [van de Vosse and Minev (1996), Canuto, *et al.* (2007), Karniadakis and Sherwood (2013)]. Apparently, Maday and Patera and the authors of these texts were unaware of all the earlier work as none of it is cited. Maday and Patera are credited with discovery of this popular method even though their work came more than a decade after its initial development. By our count, the method has been independently discovered or rediscovered five times.

The chapter on finite element methods, includes additional discussion and examples of collocation-like finite element methods.

1.2.10 Biographical Sketches. This section is not typical for a monograph like this, but its inclusion here seemed preferable to the use of short footnotes within the text or its placement

in an appendix that no one would read. It gives some brief biographical information on some of the pioneers in this field of study. It is noteworthy that so much of the original work was done at centers in Göttingen, Germany; St. Petersburg, Russia; and Kyiv, Ukraine. There was undoubtedly much exchange between these scholars which is not apparent from reading the sketches. It is remarkable that these people accomplished so much despite the turbulence of the early 20th century. Many received asylum in the United States, benefiting the US and the rest of the World. Unfortunately, Kravchuk is an exception.



John William Strutt, 3rd Baron Rayleigh (1842-1919) was British, born at Langford Grove in Maldon, Essex. He was a theoretical physicist at the University of Cambridge. He made many contributions to physics. His name is attached to many physical phenomena – Rayleigh waves, Rayleigh scattering, Rayleigh number, etc. Among numerous honors, he received the 1904 Nobel Prize in Physics "for his investigations of the densities of the most important gases and for his discovery of argon in connection with these studies." Of interest here is his classic book *The Theory of Sound* (1877). In the book and other articles, he used a rudimentary form of the Rayleigh-Ritz method to estimate the principal eigenvalues or principal vibration modes of elastic strings, bars, membranes and plates.

Boris Grigoryevich Galerkin (1871-1945) (Gal'orkin) was a Russian engineer. He was born in Polotsk, Russian Empire, which is now in northern Belarus, near the border with Latvia. His family was of modest means, which made it difficult for him to obtain an education. At the age of 12 he worked as a calligrapher in the court. He finished secondary school in Polotsk but needed exams from an additional year in order to continue his education. He completed the exams at Minsk in 1893 as a boarding student. He entered the mechanical department at St. Petersburg Technological Institute the same year. Due to his modest means he had to combine his education with work as a draftsman, and later by tutoring. He graduated in 1899.

For three years, Galerkin worked at the Russian Mechanical and Steam-locomotive Union factory in Kharkov, and also taught short courses to other engineers. In 1903 he became an engineer on the construction of the Eastern-Chinese Railway. Later he became active in organizing a union of engineers in St. Petersburg. This led to his arrest in 1906 along with 18 other members of Social-Democratic Party Committee, which later morphed into the

communist party. He was sentenced to prison for one and half years. After this experience, he became less interested in politics and decided to devote his efforts to science and engineering.

He was released from prison in late 1908 and began teaching structural mechanics at the St. Petersburg Polytechnical Institute. There he became acquainted with other faculty members: V.L. Kirpichov, I.G. Bubnov, A.N. Krylov, I.V. Meshcherskiy, and S.P. Timoshenko. He became involved in building construction and along with other faculty members visited other countries to study their building construction methods. He visited Germany, Austria, Switzerland, Belgium and Sweden.

In 1915 he published his famous paper on the Galerkin method for solving boundary value problems. His colleagues Ivan Bubnov and Stepan Timoshenko had published papers using the Rayleigh-Ritz variational procedure. Galerkin's method is an important generalization of the Rayleigh-Ritz method.

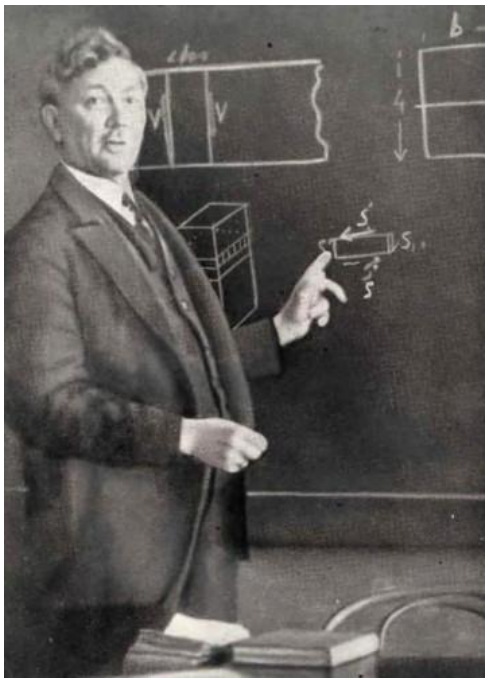
In 1923 Galerkin became dean of the of the Polytechnical Institute civil engineering faculty. In 1924 he made his last trip abroad to participate in the First International Congress on Applied Mechanics in the Netherlands. The conference was attended by most of the prominent academics of the time, and the name Methods of Weighted Residuals was born from Courant's (1924) comments at the meeting.

In later years, Galerkin was involved in all sorts of construction projects from hydro power projects to pulp and paper mills. He also received many honors, including election to headmaster of the USSR Academy of Sciences Institute of Mechanics. When World War II broke out in 1939 he was given the rank of Lieutenant-General, due to his position as head of the structural mechanics department of the military-construction school. The difficult work during the war took a toll on his health and he died in Moscow on July 12, 1945, just two months after VE day, ending the war in Europe.

Ivan Grigoryevich Bubnov (1872-1919) was a Russian born at Nizhny Novgorod, east of Moscow. He was a designer of ships and submarines for the Imperial Russian Navy. He graduated from the Marine Engineering College at Kronstadt in 1891 and the Nikolayev Marine Academy in 1896. In 1900, he was appointed Chief Assistant at the Russian Admiralty. He was involved in the design of the first Russian submarine, the *Delfin*, and also designed many later submarines. From 1904 he taught at St. Petersburg Polytechnical Institute, where he was a colleague of Stepan Timoshenko and Ivan Galerkin. He moved to the Naval Academy in 1910. He published many fundamental works describing for the first time the stresses on the hulls of ships due to water pressure. He adopted the Rayleigh-Ritz variational method for studying these problems [Bubnov, 1913,1914]. He died from typhoid in 1919 in Petrograd.

Walther Ritz (1878-1909) (Walter) was a theoretical physicist. He was born in Sion Switzerland in the Rhone Valley, the son of a well-known landscape artist, Raphael Ritz. He entered the Zurich Polytechnic ETH in 1897 and after three years left for Göttingen where he obtained his PhD in 1902 working on the theory of spectral series under Voldemar Voigt. In the company of his friend Paul Ehrenfest, he went next to Leyden to hear lectures by Lorentz. Ritz moved then to Bonn and then to Paris to work on infra-red spectra in Aimé Cotton's laboratory. This was to be followed by a series of stays in various European physics centers, intertwined with visits in sanatoria where Ritz was trying to fight his worsening tuberculosis. About 1906 he lost hope of recovery, so Ritz decided to return to Germany and devote the remainder of his life to intense research, first in Tübingen and then Göttingen in 1908. His health continued to deteriorate, and he died July 1909 at the age of 31.

Stepan Prokofyevich Timoshenko (1878-1972) (Stephen) was born in the village of Shpotovka which is in the Chernigov Governorate of northern Ukraine near Belarus. It was part of the Czarist Russian empire at that time. He was the son of a serf who had been brought up in the home of a landowner. His father managed to receive an education as a land surveyor and eventually became a landowner. From 1889 to 1896 Stephen received his secondary education at a *realschule*[1] in Romny, Ukraine. His university studies began at the St Petersburg Institute of Engineers Ways of Communication. After graduating in 1901, he stayed on and taught for two more years.



In 1903 he accepted a position as an instructor at the newly organized Saint Petersburg Polytechnical Institute. During the summers he traveled to Munich and Göttingen, Germany where he met many prominent scientists and engineers. He studied under Ludwig Prandtl at Göttingen during the 1904-05 school year. In 1906 he was appointed to the Chair of Strengths of Materials at the Kyiv Polytechnic Institute in his native Ukraine. There he continued his studies of buckling using the Rayleigh-Ritz

[1] Many eastern European countries had different schools for different career paths – in a *realschule* one received a practical education, whereas a *gymnasium* was more theoretical to prepare the student for higher learning

variational procedure. This was probably the first use of the method outside of Germany and England.

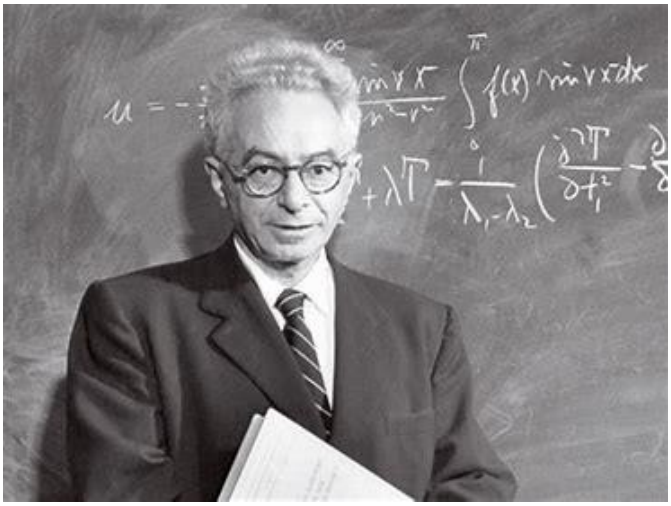
In 1911 in a protest over quotas for Jewish students, Timoshenko and two other department chairs were fired. Several other professors resigned in protest. So, he returned to St. Petersburg. He was awarded the D. I. Zhuravski prize of the St. Petersburg Ways of Communication Institute, which helped him make ends meet during his unemployment. Money from the prize allowed him to travel to Cambridge, England in 1912 to attend a mathematics congress where he met many notable English scholars. Later, he worked as a lecturer and then a Professor in the Electrotechnical Institute and the St Petersburg Institute of the Railways (1911–1917). In 1918 he returned to Kyiv to head the newly established Institute of Mechanics of the Ukrainian Academy of Sciences. These were the years of World War I and the Bolshevik revolution. The Germans, Bolheviks and Russian White Army moved back and forth across Ukraine. To protect himself and his family, they moved around, living in Crimea for a time.

Timoshenko had remained in contact with many former students. In 1920, through those contacts he received a teaching position in the Applied Mathematics Department at the Polytechnic Institute in Zagreb, Croatia. His lectures were first given through a translator, but this destroyed his teaching style. Later, he gave the lectures in Russian with a little Croatian. These lectures were effective. During his two years in Zagreb he maintained contacts in Germany and England. As part of his study of English, he translated some of his papers and got them published through colleagues in England.

His time in Zagreb ended abruptly in 1922 when he accepted an offer at an industrial firm in Philadelphia, Pennsylvania, USA. He later spent four years at Westinghouse Research Laboratory in Pittsburgh, Pennsylvania. However, he was born to teach. In 1927 he was offered a position at the University of Michigan. There his professional and teaching career blossomed. He soon had as many graduate students as he could handle and was involved with seminars and short courses which brought in speakers through his many connections abroad and in industry. In 1936 he moved to Stanford University. He retired in 1944 but remained in California, lecturing at Stanford while continuing his lifelong summer travels to Europe.

In addition to his numerous honors, Timoshenko mentored almost 40 PhD students in the US alone and published more than 20 textbooks in all areas of engineering mechanics. His books are still in widespread use in as many as 36 different languages. He is regarded by many as “The Father of Applied Mechanics.” Stephen Prokofyevitch Timoshenko died in 1972.

Nikolaĭ Mitrofanovich Krylov (1879-1955) (not to be confused with V.I. Krylov or A.N. Krylov) was a Russian mathematical physicist who graduated from the St. Petersburg Institute of Mines in 1902. He was a professor there from 1912 to 1917 followed by a professorship at Crimea University until 1922. He was then appointed head of the mathematical physics department at the All Ukrainian Academy of Sciences in 1922. Krylov published more than 200 articles and books on mathematics and physics. These included an article first describing the method of moments [Krylov (1926)] and the first articles describing error estimates for the Rayleigh-Ritz method [Krylov (1931)]. He also published a book in 1941 with N. Bogoliubov describing practical applications, which is available in English [Krylov and Bogoliubov (1943)].



Richard Courant (1888-1972) was born into a Jewish family in Lublinitz, Germany, which is in Silesia. It is now Lubliniec, Poland in southern Poland near the border with the Czech Republic. The family moved to Glatz when he was three and to Breslau when he was nine. He attended school at the König-Wilhelm Gymnasium. When Richard was fourteen, tragic circumstances caused his father to declare bankruptcy and later move to Berlin. Richard stayed on in Breslau and supported himself by tutoring.

In 1905 Courant began taking courses at the University of Breslau. He started in physics, then switched to mathematics, but did not feel stimulated. Older students Otto Toeplitz and Ernst Hellinger had left Breslau for Göttingen. They influenced him to move there in 1907. He attended classes in mathematics and physics by Hilbert and Minkowski and became Hilbert's assistant in 1908. Hilbert was heavily involved with analysis and Courant took to the subject with ease. He received his doctorate under Hilbert in 1910 and after his compulsory military service did a *habilitation* (a type of postdoc) under Hilbert as well.

He was drafted when World War I broke out. Although he avoided the worst of combat, he was wounded. After the war, he returned to Göttingen. Having divorced his first wife, he married Nerina Runge, daughter of Carl Runge, in 1919. The years following the war were ones of intense research. He founded the Mathematics Institute in 1922 and published a book on function theory and in 1924 published the classic text *Methods of Mathematical Physics* with Hilbert. Also, in 1924 at the First International Congress on Applied Mechanics in Delft,

Netherlands, his comments at the conference led to the name *Methods of Weighted Residuals* [Courant (1924)].

A new building for the Mathematics Institute was dedicated in 1929. However, in 1933 Courant was forced from his position by the Nazis, despite his service in World War I. He landed at New York University. The first few years there were difficult, but he was eventually given the task to build up a mathematics center. Courant was enthusiastic about building a center in his newly adopted country. However, he often encountered resistance due to his background as a foreigner and a Jew. He succeeded in building a center modeled after the one at Göttingen. He also snapped up many excellent scholars, such as Kurt Friedrichs, who were forced out of Germany by the Nazis. Those he could not place at NYU, he helped find other positions in the United States.

In 1943 he published a paper [Courant 1943] of a lecture at the American Mathematics Society. It included an appendix describing a finite element method. However, the paper used the name *generalized finite difference method*, provided no details and was largely overlooked. It had virtually no influence on development of the method, which did not begin until the mid-1950s when computers were first available for calculations. His work was eventually rediscovered, and it is now considered the first description of a finite element method.

His Institute of Mathematical Sciences at NYU was renamed the Courant Institute in 1964. He died following a stroke in 1972.

Robert Alexander Frazer (1891-1959) was an Englishman born in London. His father, Robert Watson Frazer, was principal secretary and librarian at the London Institute in Finsbury Circus. Robert, Jr. was born there where the family lived. Robert, Sr. was retired from the civil service in Madras (Chennai), India and was the author of four books on India and its history. Young Robert was an excellent scholar, winning several scholarships, including one to Pembroke College, Cambridge. He obtained a B.Sc first call honors degree in mathematics at London University in 1911.

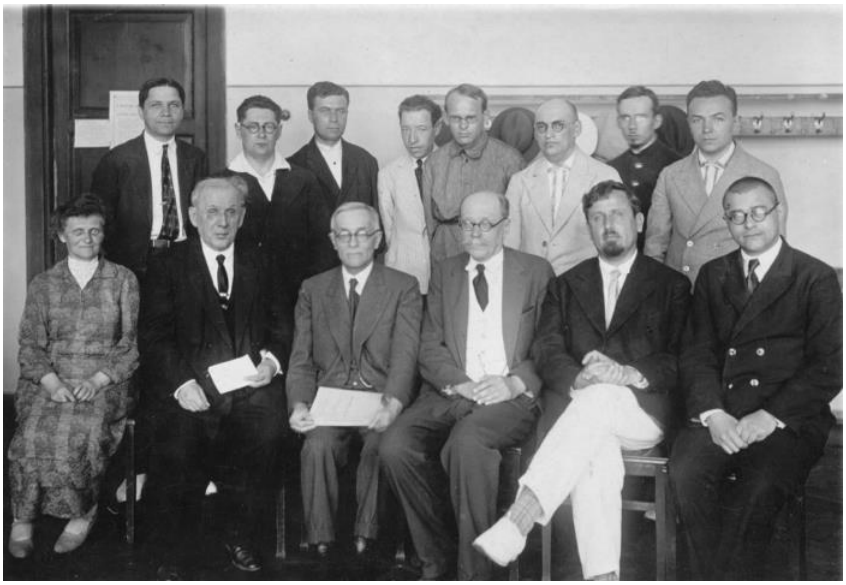
At Pembroke College Frazer was tutored by algebraist J.H. Grace and H.F. Baker, who was interested in the matrizant (systems of initial value equations). In 1930 he received a D.Sc. at University of London. Soon after receiving his B.Sc. degree, he joined the National Physical Laboratory, aeronautics section, Teddington. He spent his entire career there. He is best known for his studies of aircraft flutter and other dynamic issues using both calculations and wind tunnel experiments. He received many awards and honors during his career, including receiving the Rayleigh Prize and the Royal Aeronautical Society's Silver Medal and election as a Fellow of the Royal Society.

Frazer was an early proponent for the use of matrix methods in engineering calculations. Together with W.J. Duncan and A.R. Collar he published the monograph *Elementary Matrices* in 1938 [Frazer, *et al.* (1938)] which was a catalyst for and became a standard text on the use of matrix methods. His paper "*Approximation to Functions and to the Solutions of Differential*

Equations," with W.P. Jones and S.W. Skan coined the name "*collocation*" method and through calculations compared it to the least squares and the Galerkin methods. Although their discovery was three years later than the earliest papers, it was developed independently, more readily available in the west and included thoroughly worked examples with comparisons to other methods.

Frazer died in 1959 at the age of 68.

Mykhailo Pylypovych Kravchuk (1892-1942) (Krawchuk, Krawtchouk) was born in the northwest Volyn Region of Ukraine, bordering Belarus and Poland. It was part of the Czarist Russian empire at that time. His father was a land surveyor. Mykhailo was a proud Ukrainian and probably the foremost Ukrainian mathematician of the 20th century. He entered the University at Kyiv in 1910 studying mathematics and physics, teaching after 1917. He endured many hardships following the Bolshevik revolution and the subsequent loss of Ukrainian independence in 1922, but despite the obstacles, rose to full professor in 1925. Kravchuk was involved in teaching mathematics at secondary and post-secondary levels and in the development of mathematics terminology for the Ukrainian language.



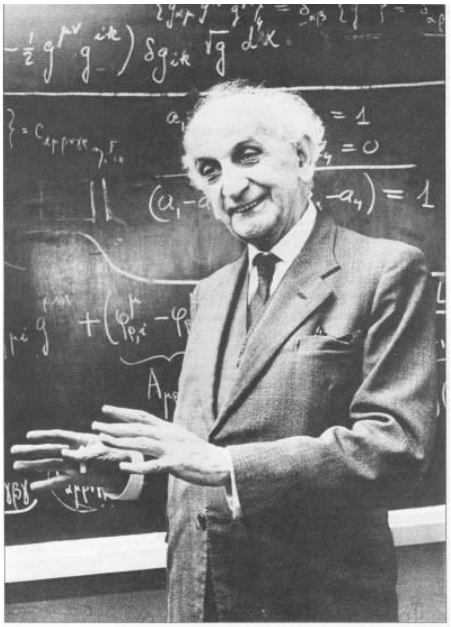
Kravchuk's mother was Polish and through her he became fluent in Ukrainian, Russian, Polish, French and German. He presented several papers at the 1928 International Mathematics Congress held in Bologna, Italy. At the congress and in other travels he developed friendships with other prominent mathematicians such as D. Hilbert, J. Hadamard and F. Tricomi (see photo, Kravchuk seated in white pants).

Kravchuk also corresponded with John V. Atanasoff (1903-1995), a professor of mathematics and physics at Iowa State University. Atanasoff obtained most of Kravchuk's papers and had them translated into English. Unfortunately, these translations were never published in the open literature. They remain in obscurity at the Iowa State University archives. Atanasoff together with graduate student Clifford Berry built the ABC (Atanasoff-Berry-Computer) computer in the years 1937-1942. The ABC computer was declared the first electronic

computer in a famous 1971 patent infringement case between Sperry-Rand and Honeywell. This decision overturned an earlier one that upheld patents associated with the ENIAC computer. There has been much speculation about what influence Kravchuk's work had on the ABC's creation [Katchanovski (2004)]. Atanasoff had used the Rayleigh-Ritz method, a 3-term approximation calculated by hand, in 1930 for his PhD which studied the dielectric constant of Helium. He had interest in better methods for performing linear algebraic calculations [Atanasoff and Brandt (1936)]. Many design features of the ABC were for the purpose of solving linear algebra problems. It is the current author's opinion that Atanasoff was motivated by the desire to perform high order MWR calculations, so Kravchuk's influence on development of the ABC was indirect.

Unfortunately, Kravchuk's friendship with other mathematicians and physicists and his love of Ukraine became his undoing. In 1938, after years of public service as an educator and academician, he fell victim to Stalin's purges. Many of his friends and colleagues turned on him (to save their own skin) and his correspondence with foreign academics was used as "proof" he was a spy. He was convicted and sent to a Gulag death camp in Siberia. He succumbed to the harsh treatment in 1942 at the age of 50.

Cornelius Lanczos (1893-1974) was born in Székesfehérvár, Hungary. His family was of Jewish origin and his father was a lawyer. His original name was Kornél Löwy but when Hungarians reacted to German names it was changed to a more Hungarian form. He attended a Jewish elementary school and then entered the local Catholic secondary school (called a Gymnasium) run by the Cistercians.



LÁNCZOS KORNÉL (1893-1974)
Magyar matematikus és fizikus, az általános relativitáselmélet világhírű kutatója, a kvantum-mechanika egyik megalapozója

He graduated from the Gymnasium in 1910 and entered the University of Budapest in the fall. He was inspired by several professors there, including Eötvös in physics and Fejér in mathematics. After graduating in 1915, Lanczos was appointed an assistant at the Technical University of Budapest, while he worked on his doctorate. He completed his doctorate in 1921, entitled *Relation of Maxwell's Aether Equations to Functional Theory.* He sent a copy of his dissertation to Einstein, who was impressed by his "sound and original thinking."

After graduation, he found the opportunities in Hungary were limited by prejudicial laws against Jews, so he took employment in Germany. First at Freiburg in extreme southwest Germany, then later in Frankfurt and Berlin. He spent the year 1928-29 as an assistant to Einstein in Berlin. He and Einstein became good friends and corresponded for years afterward. He spent 1931 on a sabbatical at Purdue University in the United States and upon his return to Germany found the treatment of Jews had become intolerable. Fortunately, he was offered a

professorship at Purdue starting in 1932. In 1944 he took employment at the Boeing Aircraft Company and in 1949 moved to the Institute for Numerical Analysis at the National Bureau of Standards in Los Angeles. At the Bureau of Standards he adapted many of his numerical methods for the digital computer. In 1952 he accepted a position as head of the Theoretical Physics Department at the Dublin Institute for Advanced Studies in Ireland. In Ireland he finally had a position which offered him the support and opportunity to pursue his first love, the theory of relativity. This period was also the most productive of his career. Of the more than 120 articles and books he published, about half were produced while he was in Ireland.

He was the first to propose collocation at the roots of orthogonal polynomials, section 1.2.5, which he called the *method of selected points*. He also originated the *tau method*, section 1.2.7, which is equivalent to the method of moments. Although he does not normally receive credit, he was first to describe the fast Fourier Transform (FFT), and the basis for the Golub and Welch (1969) eigenvalue method for roots of orthogonal polynomials. These are extraordinary achievements, especially considering his passions were elsewhere.

On a visit to Budapest and 1974, he had a sudden heart attack and died the next day.

Leonid Vital'evich Kantorovich (1912-1986) was born in St. Petersburg, Russia. His father, Vitaliy Moiseevich Kantorovich, was a prominent medical doctor specializing in the treatment of sexually transmitted diseases. One of the earliest events to impact young Leonid was the upheaval accompanying the Bolshevik revolution in 1917-20. For safety, the family fled to Belarus (then known as Byelorussia) for a period.

Kantorovich was truly a child prodigy, developing an interest in science and mathematics at an early age. He entered Leningrad State University at the age of 14. Amongst his classmates were Sergei Lvovich Sobolev, Solomon Grigor'evich Michlin, Dmitrii Konstantinovich Faddeev and Vera Nikolaevna Zamyatin (later known as Vera Nikolaevna Faddeeva), all had a major impact on the field of mathematics. He graduated with the equivalent of a doctorate in 1930 at the age of 18. Afterward he continued at Leningrad State University and held research positions in various departments, including the Research Institute of Mathematics and Mechanics and the Department of Numerical Mathematics. Some of his students refused to believe the 20 year old youngster could be their lecturer.

During this period he began to study variational calculus and in 1933 published his first book *Calculus of Variations* coauthored with Vladimir Ivanovich Krylov and Vladimir Ivanovich Smirnov. He presented two papers at the 1934 Second All-Union Mathematical Congress in Leningrad, including the first paper to describe the collocation method [Kantorovich (1934)], which he called the *interpolation method*. This obscure article was largely unnoticed in the west. In the 1930s functional analysis was heavily studied and Kantorovich made significant

contributions in this area. For our area of interest here, his greatest influence was through two books *Approximate Methods in Higher Analysis* with Vladimir Ivanovich Krylov (1936 1st edition, 1950 Russian, 1958 English) and *Functional Analysis in Normed Spaces* with G.P. Akilov (1959 Russian, 1964 English).

Interestingly, Kantorovich's received his greatest recognition in the field of economics. Although his training was exclusively in mathematics, he seemed to have an excellent common sense understanding of practical problems in economics. He got into the study of economics and operations research almost by accident in 1938 when approached to help determine the most effective allocation of machines in the production plywood. Problems of this type lead to his discovery of linear programming. He was awarded the 1975 Nobel Prize in economics. He contracted cancer and died in 1986.

1.3 A First Example

As a first example, consider diffusion or conduction through a slab or wall, with a temperature dependent thermal conductivity or concentration dependent diffusion coefficient. The problem has been discussed by others [Finlayson (1972), pp. 16-19, Bert and Malik (1996)]. The governing equation for the problem is:

$$\frac{d}{dx}\left(k(u)\frac{du}{dx}\right) = -\frac{dq}{dx} = 0 \tag{1.17}$$

with $k(u) = 1 + u$ and the flux $q = -k(u)du/dx$. Consider either Dirchlet conditions:

$$u(0) = \bar{u}_0, \;\; u(1) = \bar{u}_1 \tag{1.18}$$

or conditions of the third kind also called a Robin or radiation condition:

$$k(u)\frac{du}{dx}\Big|_{x=0} = b_0[u(0) - \bar{u}_0], \quad -k(u)\frac{du}{dx}\Big|_{x=1} = b_1[u(1) - \bar{u}_1] \tag{1.19}$$

We will describe various solutions to this problem with MWR. Only the approximation methods and their results are described. The actual mechanics of solving the resulting nonlinear algebraic problems is not discussed here, but many nonlinear problems are considered in the body of the monograph.

The problem is easily solved analytically. For the case with $\bar{u}_0 = 0$, $\bar{u}_1 = 1$ and $b_0 = b_1 = b$, the exact solution is:

$$q = \frac{-3b}{2b+6} \text{ and}$$
$$u = \sqrt{1 - (2 - q/b)(q/b) - 2qx} \; - \; 1$$

When $b \to \infty$, the Dirichlet conditions result and $u = \sqrt{1+3x} - 1$, while the flux is constant at q = -1.5. MWR will produce the exact solution if it is contained within the trial solution. Since the exact solution is not normally available, a polynomial trial function is used. A suitable monomial type of trial solution is:

$$\tilde{u} = (1-x)\,\tilde{u}(0) + x\,\tilde{u}(1) + x(1-x)\sum_{i=1}^{n} c_i x^{(i-1)} \tag{1.20}$$

For Dirichlet boundary conditions, the boundary values are substituted, $\tilde{u}(0) = \bar{u}_0,\ \tilde{u}(1) = \bar{u}_1$, while for Robin conditions the boundary values are parameters determined during the calculation. Alternatively, a modal approximation can be constructed from shifted Legendre polynomials:

$$\tilde{u} = (P_0 - P_1)\frac{\tilde{u}(0)}{2} + (P_0 + P_1)\frac{\tilde{u}(1)}{2} + \sum_{i=1}^{n} d_i(P_{i-1} - P_{i+1}) \tag{1.21}$$

The shifted polynomials are defined for the interval [0,1] instead of the usual interval of [-1,1]. The properties of Legendre polynomials are detailed in Chapter 2. Eq. (1.21) uses the following properties of the shifted polynomials:

$$P_0 = 1,\ P_1 = 2x - 1,\ P_i(1) = 1 \text{ and } P_i(0) = (-1)^i$$

A nodal trial solution is given by Eq. (1.3), where the $\boldsymbol{x} = \{0, x_1, \ldots x_n, 1\}$. The first and last nodal values are the boundary values and the internal nodes will usually be roots of an orthogonal polynomial. For a given n and method, the same or equivalent results are obtained regardless of the choice of representation, but of course the results are in a different form. Transforms to convert from one form to another are discussed later in Chapter 2.

First, consider a two-term approximation and Dirichlet boundary conditions with $\bar{u}_0 = 0$, $\bar{u}_1 = 1$. For convenience omit the tilde (~) to designate the approximation. Substitution of the trial function, gives the following approximation for the flux:

$$-q = (1+u)\frac{du}{dx} = [1 + x + x(1-x)(c_1 + c_2 x)][1 + (1-2x)c_1 + (2x - 3x^2)c_2]$$

And the residual is:

$$\begin{aligned} R &= (1+u)\frac{d^2u}{dx^2} + \left(\frac{du}{dx}\right)^2 \\ &= -2[c_1 - (1-3x)c_2]\,[1 + x + x(1-x)(c_1 + c_2 x)] + [1 + (1-2x)c_1 + (2x - 3x^2)c_2]^2 \end{aligned}$$

The weighted residuals, Eq. (1.8), are then used to determine the coefficients, c_i:

$$\int_0^1 R(x, \boldsymbol{c})\, w_j = 0 \tag{1.22}$$

for $j = 1, \ldots, n$. The weight functions are as discussed in Section 1.2 above. The first moments and subdomain approximations weight the residual by unity, so the integral of the residual is set to zero. This case is also called the integral method. The Galerkin method weights the residual by the trial functions, while the least squares method minimizes the mean square residual. The collocation method sets the residual to zero at n points.

Table 1.1 Nonlinear Conduction Problem, n = 1

Method	w_1	c_1	$u(½)$	$-q(0)$	$-q(1)$	error $u(½)$	error $q(0)$	error $q(1)$
moments	1	0.33333	0.58333	1.33333	1.33333	0.38%	11.11%	11.11%
Galerkin†	$(1-x)x$	0.32624	0.58156	1.32624	1.34752	0.07%	11.58%	10.17%
Least Squares	$\partial R/\partial c_1$	0.24948	0.56237	1.24948	1.50104	3.23%	16.70%	0.07%
collocation†	$\delta(x - ½)$	0.31662	0.57916	1.31662	1.36675	0.34%	12.23%	8.88%
exact			0.58114	1.5	1.5			

†alternate calculation gives exact boundary flux for Galerkin method and collocation

Table 1.2 Nonlinear Conduction Problem, n = 2

Method	w_1	w_2	c_1	c_2	$u(½)$	$-q(0)$	$-q(1)$	error $u(½)$	error $q(0)$	error $q(1)$
moments	1	x	0.50000	-0.25000	0.59375	1.5	1.5	2.17%	0.00%	0.00%
Galerkin†	$(1-x)x$	$(1-x)x^2$	0.44059	-0.21518	0.58325	1.44059	1.54917	0.36%	3.96%	3.28%
Least Sq.	$\partial R/\partial c_1$	$\partial R/\partial c_2$	0.43134	-0.18727	0.58443	1.43134	1.51186	0.57%	4.58%	0.79%
collocation	$\delta(x - ⅓)$	$\delta(x - ⅔)$	0.40761	-0.19165	0.57795	1.40762	1.56803	0.55%	6.16%	4.54%
Lobatto†	$\delta(x$-0.276)	$\delta(x$-0.724)	0.43526	-0.20622	0.58304	1.43526	1.54193	0.33%	4.32%	2.80%
Chebyshev	$\delta(x - ¼)$	$\delta(x - ¾)$	0.45317	-0.21572	0.58633	1.45317	1.52510	0.89%	3.12%	1.67%
Gauss	$\delta(x$-0.211)	$\delta(x$-0.789)	0.48805	-0.23437	0.59272	1.48805	1.49263	1.99%	0.80%	0.49%
exact					0.58114	1.5	1.5			

†alternate calculation gives exact boundary flux for Galerkin method and Lobatto

Tables 1.1 and 1.2 summarize the results for n = 1 and 2, while Figs. 1.9, 1.10 and 1.11 plot results for n = 2. For the collocation method, one must choose the location of the collocation points. For n = 1, the obvious choice is x = ½. For n = 2, four different choices were used: equally spaced points (⅓, ⅔), Chebyshev points (¼, ¾), and the base points of Gauss, $\left(1 \pm \sqrt{1/3}\right)/2$, and Lobatto, $\left(1 \pm \sqrt{1/5}\right)/2$, quadrature.

Even with n = 1, the error in the solution is at most only a few percent, but the error in the flux is considerably greater. When n = 2, the results are much more accurate. Note that the results for collocation at Gauss points tend to track those for the method of moments. The results for

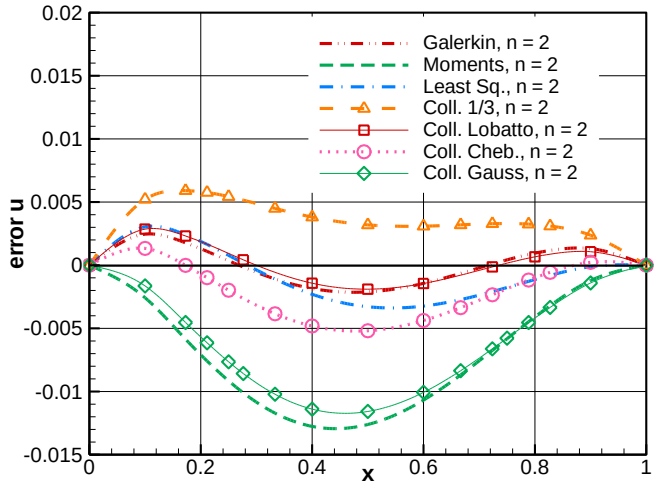


Fig. 1.9 Error in solution, nonlinear conduction problem

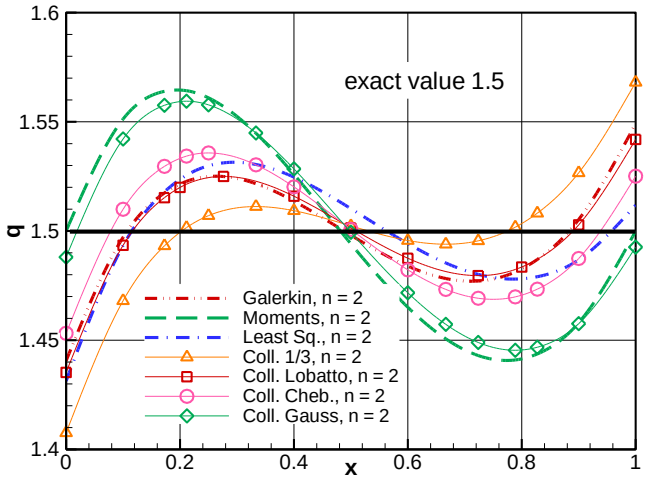


Fig. 1.10 Calculated flux, nonlinear conduction problem

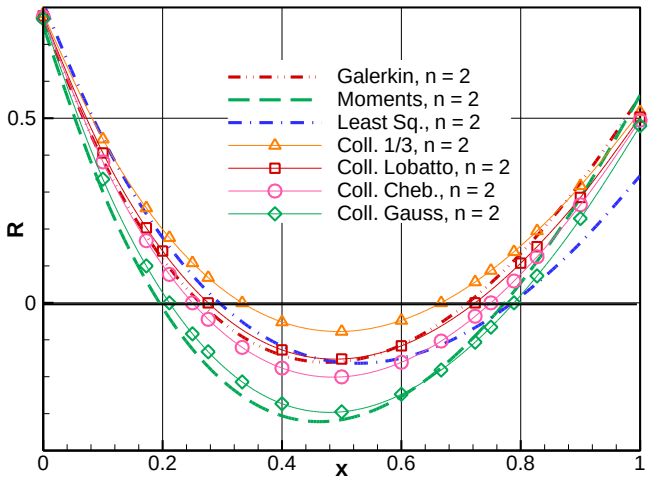


Fig. 1.11 Residual function, nonlinear conduction problem

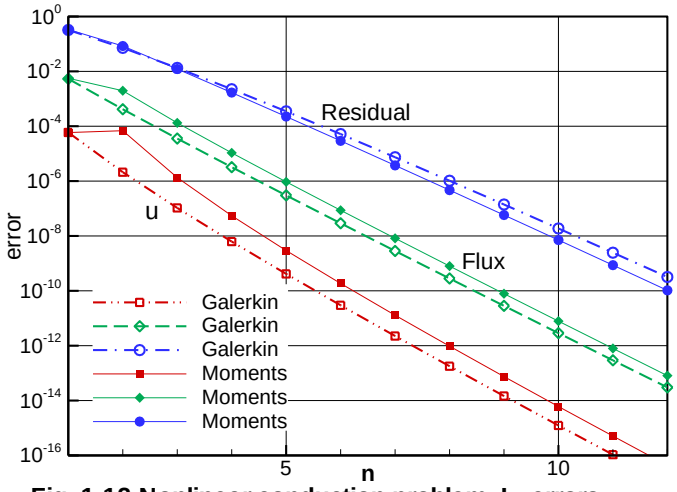


Fig. 1.12 Nonlinear conduction problem, L_2 errors

collocation at Lobatto points are close to those for the Galerkin method. Later in the monograph we will discover why these methods tend to agree. Note also, the method of moments gives the exact endpoint fluxes for $n > 1$, but it has larger errors in the profile of u and q which should be constant at 1.5. Shortly, we describe a more accurate method for calculating endpoint fluxes for the Galerkin method.

Fig. 1.12 shows the L_2 errors versus n for the solution, u, the flux, q, and the residual, R. All these measures converge at an exponential rate, the values are reduced about an order of magnitude for each n until eventually roundoff errors become evident. The L_2 residual errors are smallest with the method of moments, but errors for the solution and flux are somewhat smaller for the Galerkin method. The errors with collocation at Gauss and Lobatto points are not plotted, since they overlay those for the moments and Galerkin methods, respectively.

Now consider a case with boundary conditions of the third kind. This type of condition would be used to simulate heat conduction through a slab surrounded by a fluid. Normally, the flux from the bulk fluid to the surface of the slab would be characterized with a heat transfer coefficient, so $\bar{u}_0$ and $\bar{u}_1$ in Eq. (1.19) represent the bulk fluid temperatures on the two sides of the slab. We will consider the case $\bar{u}_0 = 0$, $\bar{u}_1 = 1$ and $b_0 = b_1 = b = 12$. The analytical solution is:

$$-q = 1.2, \quad \text{and}$$
$$u = \sqrt{1.21 + 2.4x} \; - \; 1$$

The nonlinearity of the boundary conditions complicates the construction of trial functions which meet the boundary conditions. As stated above, the form of the trial functions is somewhat a matter of convenience. To simplify the problem, we use Hermite cubic polynomials as trial functions:

$$\tilde{u} = h_0(x)u_0 + \bar{h}_0(x)u_0' + h_1(x)u_1 + \bar{h}_1(x)u_1' \tag{1.23}$$

where $u_0 = \tilde{u}(0), u_0' = \tilde{u}'(0), u_1 = \tilde{u}(1), u_1' = \tilde{u}'(1)$, represent the values and derivatives of the solution at each end of the interval, giving four parameters to determine a cubic polynomial. The trial functions are the four Hermite cubic functions [Hildebrand (1987), p. 282]:

$$h_0(x) = (1+2x)(1-x)^2$$
$$\bar{h}_0(x) = x(1-x)^2$$
$$h_1(x) = (3-2x)x^2$$
$$\bar{h}_1(x) = (x-1)x^2$$

The boundary conditions reduce the free parameters by two, while weighted residuals are used to provide two additional conditions. The boundary conditions, Eq. (1.19), give the relationships:

$$u_0' = \frac{b\,u_0}{1+u_0}, \quad u_1' = \frac{b(1-u_1)}{1+u_1} \tag{1.24}$$

A trial solution which obeys the boundary conditions is:

$$\tilde{u} = h_0(x)u_0 + \bar{h}_0(x)\left(\frac{b\,u_0}{1+u_0}\right) + h_1(x)u_1 + \bar{h}_1(x)\left(\frac{b(1-u_1)}{1+u_1}\right) \tag{1.25}$$

The trial solution is substituted into the differential equation to construct the residual function and the two remaining free parameters (u_0 and u_1) are determined using two weighted residual conditions, Eq. (1.22).

Table 1.3 lists results for several cubic approximations. The method of moments uses weights of 1 and x or two equivalent linear functions. For the collocation method the residual is set to zero at two points, four choices are listed. For the Galerkin method, the problem is complicated by the nonlinear dependence on the parameters in Eq. (1.25). For a Galerkin method which satisfies the boundary conditions exactly, the correct procedure is to weight the residual by $\partial\tilde{u}/\partial u_0$ and $\partial\tilde{u}/\partial u_1$. This approach is messy, but with the Galerkin method there is a better way.

Table 1.3 Nonlinear Conduction Problem, 3rd kind, b = 12, n = 2

Method	w_1	w_2	$u(0)$	$u(1)$	$u(½)$	$-q(0)$	$-q(1)$	error $u(½)$	error $q(0)$	error $q(1)$
Moments	1	x	0.1	0.9	0.55742	1.2	1.2	0.90%	0.00%	0.00%
Galerkin	$(1-x)x$	$(1-x)x^2$	0.1	0.9	0.55331	1.2	1.2	0.16%	0.00%	0.00%
Lobatto weak	$\delta(x-0.276)$	$\delta(x-0.724)$	0.1	0.9	0.55334	1.2	1.2	0.17%	0.00%	0.00%
collocation	$\delta(x-⅓)$	$\delta(x-⅔)$	0.09666	0.89755	0.54832	1.15992	1.22944	0.74%	3.34%	2.45%
Lobatto	$\delta(x-0.276)$	$\delta(x-0.724)$	0.09778	0.89850	0.55155	1.17338	1.21802	0.16%	2.22%	1.50%
Chebyshev	$\delta(x-¼)$	$\delta(x-¾)$	0.09848	0.89908	0.55354	1.18170	1.21100	0.20%	1.52%	0.92%
Gauss	$\delta(x-0.211)$	$\delta(x-0.789)$	0.09975	0.90015	0.55718	1.19698	1.19815	0.86%	0.25%	0.15%
exact			0.1	0.9	0.55242	1.2	1.2			

As discussed in Section 1.2.8, boundary conditions involving fluxes are called *natural* boundary conditions. The Galerkin method does not require trial functions to satisfy these boundary conditions exactly. In fact, it is better to satisfy them approximately. This claim may seem counterintuitive, but by allowing a small error in the boundary condition, the method has more flexibility to adapt the polynomial to the solution. The tradeoffs can be visualized by

comparing results for the method of moments with those of the Galerkin method. For Dirichlet conditions, Figs. 1.9 and 1.10 and Table 1.2 show that the method of moments gives exact boundary fluxes for the problem, but the errors in the profiles are greater than those for the other methods. Shortly, we show how to calculate exact boundary fluxes for this problem with the Galerkin method, so that it gives the best of both worlds, i.e. accurate profiles and accurate boundary fluxes.

By using a natural treatment, the boundary conditions are incorporated into the approximation and are satisfied approximately, together with the differential equation. To develop the procedure, start again with the weighted residuals, Eq. (1.22), using trial functions which do not satisfy the boundary conditions. For simplicity, the monomial approximation, Eq. (1.20), is used, so the trial functions are: $\boldsymbol{\psi}(x) = \{1-x,\ x,\ (1-x)x,\ (1-x)x^2, \cdots\}$ and the parameters are: $\boldsymbol{a} = \{\tilde{u}(0), \tilde{u}(1), c_1, c_2, \cdots\}$. The weights in Eq. (1.22) are the trial functions, so the weighted residual conditions are:

$$\sum_{k=0}^{n+1} a_k \int_0^1 \frac{d}{dx}\left(k(\tilde{u})\frac{d\psi_k}{dx}\right)\psi_j dx = 0 \tag{1.26}$$

The equation is first integrated by parts to put it into its *weak* form:

$$k(\tilde{u})\frac{d\tilde{u}}{dx}\psi_j\Big|_0^1 - \sum_{k=0}^{n+1} a_k \int_0^1 k(\tilde{u})\frac{d\psi_k}{dx}\frac{d\psi_j}{dx}dx = 0 \tag{1.27}$$

The first term is the flux multiplied by the trial functions evaluated at the two boundaries. It simplifies since only the first trial function is nonzero at x = 0 and only the second one is nonzero at x = 1. The natural boundary condition treatment substitutes the boundary conditions for the fluxes. When the first two equations are written separately the following equations are to be solved:

$$\begin{aligned}
&ba_0 - \sum_{k=0}^{n+1} a_k \int_0^1 k(\tilde{u})\frac{d\psi_k}{dx}dx = 0\\
&b(a_1 - 1) + \sum_{k=0}^{n+1} a_k \int_0^1 k(\tilde{u})\frac{d\psi_k}{dx}dx = 0\\
&\sum_{k=0}^{n+1} a_k \int_0^1 k(\tilde{u})\frac{d\psi_k}{dx}\frac{d\psi_j}{dx}dx = 0 \ \text{ for } j = 2, \ldots, n+1
\end{aligned} \tag{1.28}$$

Results for the Galerkin method are included in Table 1.3. We note that the boundary values of the solution are exact, so exact fluxes result when calculated by: $-q = b\tilde{u}(0) = b[1-\tilde{u}(1)]$ or by Eq. (1.27). The error in u and the flux profiles are displayed in Figs. 1.13 and 1.14 together with results for other methods. We note that in Fig. 1.14 the boundary values of $k(\tilde{u})\, d\tilde{u}/dx$ are not exact, but nevertheless the method provides a means for calculating accurate boundary fluxes.

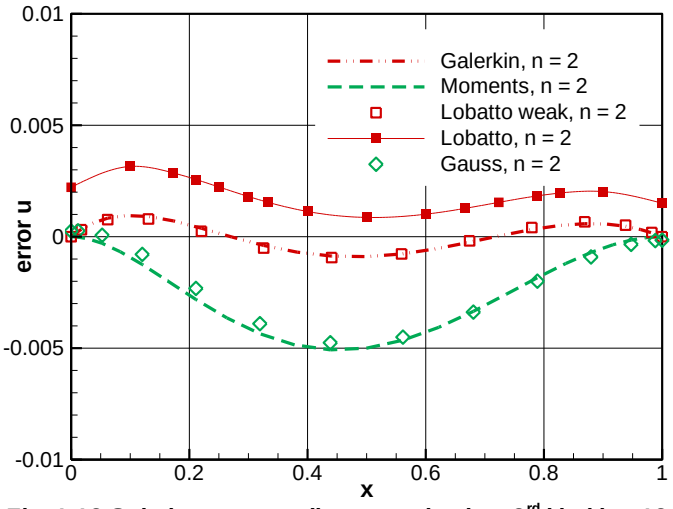


Fig. 1.13 Solution error nonlinear conduction, 3rd kind b = 12

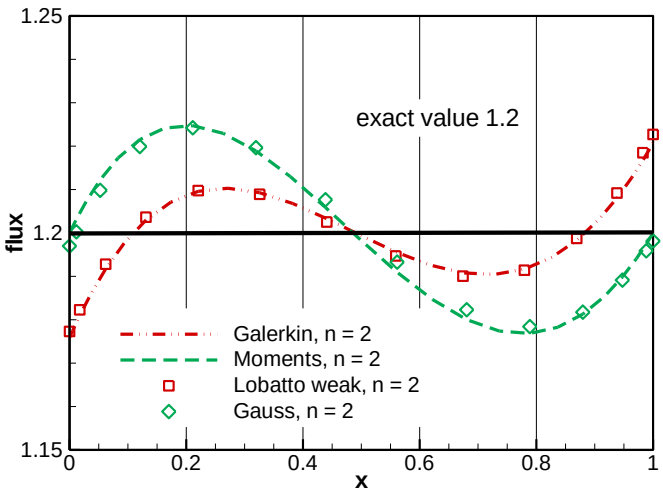


Fig. 1.14 Flux nonlinear conduction, 3rd kind b = 12

To understand this behavior, it is instructive to rewrite the equations in terms of the flux:

$$
\begin{aligned}
& b\,\tilde{u}(0) + \int_0^1 q\,dx = 0 \\
& b(\tilde{u}(1) - 1) - \int_0^1 q\,dx = 0 \\
& \int_0^1 q\,\frac{d\psi_j}{dx}\,dx = 0 \ \text{ for } j = 2, \ldots, n+1
\end{aligned}
\tag{1.29}
$$

The first two conditions are used to approximate the boundary conditions, Eq, (1.19). The boundary values of the flux are replaced by the average flux – an excellent choice since it should be constant. The third condition then weights the residual toward zero, which weights the flux towards a constant value.

Analytically, the exact flux for this problem is given by the equation:

$$
-q = \int_0^1 k(u)\frac{du}{dx}\,dx = \int_{u(0)}^{u(1)} k(u)\,du \tag{1.30}
$$

Since this relationship is imbedded in the approximation, the average flux from the Galerkin method is the analytic value even when u is approximate. The method is approximate, so it cannot give a flux which is constant, but the average value is correct, and the flux approaches the correct constant value as $n \to \infty$, and the boundary conditions also are satisfied in the limit. Like the results in Fig. 1.12, a plot of the L_2 errors for this problem exhibits an exponential convergence rate. For large n, the choice of method makes little difference.

Figs. 1.13 and 1.14 show that the Galerkin method gives more accurate profiles than the method of moments. The method of moments gives the correct values of u and q at the boundaries, fixing all four degrees of freedom for a cubic. Clearly, forcing correct boundary values causes the method of moments to produce larger errors in the internal profiles. The

Galerkin method provides a means for calculating the correct boundary values while producing more accurate internal profiles.

When discussing the Dirichlet results above, we noted that collocation at Lobatto points produces results close to those of the Galerkin method. In the body of the monograph, Section 3.1.3, we show that collocation at Lobatto points is equivalent to a Galerkin method with the integrals approximated by quadrature. For this reason, a weak form of the collocation method can be constructed. Table 1.3 and Fig. 1.13 show results for both the weak form of collocation at Lobatto points and the conventional method, where the boundary conditions are satisfied exactly, i.e. Eq. (1.25). When the boundary conditions are forced to be met, the internal profiles are thrown off and the boundary flux is only approximate. The weak form of collocation at Lobatto points produces results which are nearly identical to those of the Galerkin method, but it is much simpler to implement, especially for nonlinear problems like this one.

As stated in Section 1.2.8, Dirichlet boundary conditions can be thought of as degenerate natural boundary conditions. This idea can be used to calculate more accurate boundary fluxes for the Galerkin method and collocation at Lobatto points. The exact flux is given by: $-q = b\,u(0) = b[1 - u(1)]$. Obviously, this equation cannot be used to calculate the flux as $b \to \infty$, since the equation reduces to the product of zero and infinity. However, Eq. (1.29) can be used and it produces the exact boundary flux, even for n = 1. So, Tables 1.1 and 1.2 should be modified to reflect exact flux calculations for the Galerkin method and collocation at Lobatto points.

Fluxes are important quantities in engineering calculations. This simple example of heat flux through a wall illustrates the various MWR and how accurate approximate solutions, including fluxes, can be calculated. Hopefully, this introduction will motivate a desire for a better understanding of the MWR and how they are implemented. Of special interest are the simpler collocation methods that produce accuracy on par with the more complex Galerkin and moments methods.

1.4 Road Map to this Monograph

The primary purpose of this monograph is to use examples to illustrate how to solve differential equations with MWR, especially collocation-like methods - Orthogonal Collocation, Pseudospectral and Differential Quadrature methods. We start with global approximations, using a single polynomial to approximate the solution. Later, we graduate to finite element methods.

Chapter 2 describes the fundamental relationships and calculation methods which form the building blocks for MWR. The methods can be successfully applied to solve problems without a complete understanding of the material in this chapter. One may choose to initially scan this material and study the examples at the end of the chapter. Further study is warranted for those seeking a deeper understanding. The chapter describes orthogonal polynomials and interpolating polynomials, calculation of the polynomial roots or collocation points, quadrature weights, differentiation matrices for modal and nodal methods and methods for conversion

between the two representations. Computer codes are available to perform these calculations with Matlab/Octave, Python, Excel, Fortran 90+ and C++. Several examples demonstrate some of the basic calculations. Chapter 2 was originally envisioned as an appendix but grew to such proportions that it was elevated to a chapter.

Chapter 3 treats two boundary value problems. The first is a diffusion problem with a source function and the second is a coupled convection dominated chemical reactor model. Several variations of these problems are covered: linear and nonlinear source, various boundary conditions, constant and variable coefficients, asymmetric and symmetric problems, various geometry, nodal and modal solutions. The problems are solved with collocation, Galerkin and moments methods. Flux calculations, the treatment of flux boundary conditions and mass conservation are analyzed. This chapter is fundamental to the later ones, since much of the material learned is applied to other more complex problems.

Chapter 4 considers parabolic problems. The first one considered is the falling liquid film problem studied by Villadsen and Michelsen (1978, ch. 4). This problem is solved with the Galerkin method and with collocation using five different choices of collocation points. The no flux condition used at one boundary, provides an opportunity for further testing the treatment of flux boundary conditions. For this problem the first order derivative term is in the axial spatial coordinate, i.e. z, instead of time which is usually the case. The problem is solved both analytically in z and by various stepping methods, Runge-Kutta, etc. The consideration of both analytical and numerical solutions in z gives valuable insight into the performance of various stepping methods. (Nonlinear example to be added. Graetz problem)

Chapter 5 applies the method to the hyperbolic wave equation to simulate springs in a model of an internal combustion engine valve train.

Chapter 6 finite elements in one dimension – catalyst pellet problem, convective-diffusion problem, Buckley-Leverett and Burger equations.

More to follow ……

2. Fundamental Calculations

The purpose of this chapter is to describe the methods used to calculate the basic quantities needed to implement Methods of Weighted Residuals (MWR), especially orthogonal collocation, which is also called the pseudospectral method and differential quadrature method. Complete knowledge of this material is not needed to apply the methods, since the codes provided can be used as a *Black Box* which provide the quantities needed to apply the methods. The reader may wish to scan this material initially, then use it as a reference and perhaps later study it in more detail. However, the software descriptions and example calculations at the end of the chapter are essential reading for anyone planning to apply these methods.

In a collocation method, the selection of the collocation points is critically important. Once the points are selected, the method is basically specified, since the other parameters in the approximation can be calculated from the points. The orthogonal collocation method differs from an ordinary collocation method by using collocation points that are the roots of orthogonal polynomials. The orthogonal polynomial roots all fall in the interior of the domain, so boundary points are added to facilitate the approximation of boundary conditions.

Why use the roots of orthogonal polynomials? They are selected because they form the basis of highly accurate numerical integration or quadrature methods. The accurate quadrature formulas produce a collocation method which closely approximates the accurate but more cumbersome integrated MWR, e.g Galerkin method or method of moments. With a collocation method, the residual is interpolated to zero. As with the interpolation of other functions, equally spaced and other choices of points can lead to the Runge (1901) phenomenon.

A polynomial trial solution can be represented in several different ways, but in the absence of rounding errors, the solution is the same, regardless of the representation. Some early applications used simple monomials, x^k. Alternatively, a *modal* basis of orthogonal polynomials, usually Chebyshev or Legendre polynomials, can be used, like Eqs. (1.2) or (1.21). To meet boundary conditions, the polynomials may be combined as in Eq. (1.21) or the coefficient can be required to meet side conditions.

Some applications use a modal basis, but as explained in Chapter 1, many prefer a *nodal* basis, like Eq. (1.3), where the unknown coefficients are the values at the collocation points. A nodal basis is more intuitive and finite-difference-like. The trial solution in this case are:

$$u(x) \approx \sum_{i=0}^{n+1} \ell_i(x)\, u(x_i) \tag{2.1}$$

where $\ell_i(x)$ are the Lagrange interpolating polynomials:

$$\ell_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n+1} \frac{(x - x_j)}{(x_i - x_j)} = \frac{\hat{p}_n(x)}{(x - x_i)\hat{p}_n'(x_i)} = \frac{\hat{p}_n(x) W_i^b}{(x - x_i)}$$

and:

$$\hat{p}_n(x) = \prod_{j=0}^{n+1}(x - x_j) \quad \text{and} \quad \hat{p}_n'(x_i) = \left.\frac{d\hat{p}_n(x)}{dx}\right|_{x_i} = \prod_{\substack{j=0 \\ j\neq i}}^{n+1}(x_i - x_j)$$

x_0 and x_{n+1} are the boundary points, usually 0 and 1 or -1 and 1, while the interior points are the roots of orthogonal polynomials. The left definition of $\ell_i(x)$ above is the familiar one and the middle definition is called the fundamental form [Szegö (1975)].

$W_i^b = 1/\hat{p}_n'(x_i)$ are the so called barycentric weights. These quantities figure prominently throughout the fundamental calculations. Not only do they appear in the interpolation formula, but integration and differentiation relationships depend on the same quantities. Once the polynomial roots or collocation points are known and the barycentric weights calculated, the other quantities can be calculated easily.

Many problems have solutions which are symmetric about a central point, x = 0. For a modal approach, one selects orthogonal polynomials which are symmetric, e.g. even numbered Legendre polynomials. With a nodal approach symmetric trial functions are:

$$u(x^2) \approx \sum_{i=1}^{n+1} \ell_i(x^2)\, u(x_i) \tag{2.2}$$

where $\ell_i(x^2)$ are Lagrange interpolating polynomials:

$$\ell_i(x^2) = \prod_{\substack{j=1 \\ j\neq i}}^{n+1} \frac{(x^2 - x_j^2)}{(x_i^2 - x_j^2)}$$

In these problems, the symmetry condition is the boundary condition, $du/dx = 0$ at $x = 0$. Since this condition is satisfied by the trial functions, there is no need for a boundary point at the left end, so that point is dropped, and the remaining points are numbered from 1 to n + 1. The last point, x_{n+1} = 1, is used to satisfy a condition at the right boundary.

Determining the trial functions used can sometimes be confusing. For example, many texts and articles state the trial functions are orthogonal polynomials, but then monomials are used and transformations are applied to produce a nodal formulation. In this monograph, the approximations are developed directly using a nodal basis throughout. Although a nodal basis is used almost exclusively, a few simple examples use a modal approach in order to give a flavor for the differences between nodal and modal trial functions.

It is frequently stated that these methods are successful due to the orthogonality of modal trial functions. However, it is not the trial functions, but the residual weighting criteria, i.e. the w_k in Eq. (1.9), which determines the accuracy and convergence properties. There are simple linear transforms that convert from one representation to another – modal, nodal or monomial. The transforms discussed in Sections 2.9 and 2.10 and examples in later chapters show that for a

given weighting criteria, e.g. Galerkin or moments, the results are identical regardless of the trial function representation.

To apply MWR solution methods, we must be able to calculate various derivatives of the trial functions. For the integrated MWR described in Chapter 1, i.e. Galerkin and moments methods, integration is obviously important. Collocation does not require integration per se and some developments of these methods pay little attention to it. However, integration also plays a key role in efficient collocation methods, so the application of all MWR requires techniques for integrating and differentiating quantities involving the trial functions, whether nodal or modal. The purpose of this chapter is to develop the relationships and methods needed to calculate these quantities.

Since the nodes in Eq. (2.1) consist of the roots of an orthogonal polynomial together with the endpoints, the orthogonal polynomial is related by:

$$\hat{p}_n(x) = (x - x_0)(x - x_{n+1})\, p_n(x) \tag{2.3}$$

where $p_n(x)$ is an orthogonal polynomial. Note that $p_n(x)$, $\ell_i(x)$ and $\hat{p}_n(x)$ are degree n, n + 1 and n + 2, respectively. All are monic polynomials, i.e. with leading coefficient of unity. Here we follow the convention that the monic form of an orthogonal polynomial is denoted by a lowercase p and the conventional form is in uppercase. A hat (^) designates the inclusion of the endpoints.

The values of the barycentric weights or $\hat{p}_n'(x_i)$ can be determined by calculation of the continued products in Eq. (2.1) or directly from the monic orthogonal polynomials, by differentiation of Eq. (2.3):

$$\begin{aligned} 1/W_i^b &= \hat{p}_n'(x_i) = (x_i - x_0)(x_i - x_{n+1})\, p_n'(x_i) \quad \text{for } i = 1, \ldots, n \\ 1/W_0^b &= \hat{p}_n'(x_0) = (x_0 - x_{n+1}) p_n(x_0) \\ 1/W_{n+1}^b &= \hat{p}_n'(x_{n+1}) = (x_{n+1} - x_0) p_n(x_{n+1}) \end{aligned} \tag{2.4}$$

Differentiation of Eq. (2.1) is straight forward, but integration is more of a problem. The integration formulas needed for nonsymmetric problems are of the form:

$$\begin{aligned} &\int_0^1 f(x)dx \cong \sum_{i=0}^{n+1} f(x_i) \int_0^1 \ell_i(x)\, dx = W_0 f(0) + \sum_{i=1}^{n} W_i f(x_i) + W_{n+1} f(1) \quad \text{or} \\ &W_i = \int_0^1 \ell_i(x)\, dx \end{aligned} \tag{2.5}$$

The numerical integration formulas needed for symmetric problems are of the form:

$$\begin{aligned} &\int_0^1 f(x^2) x^\gamma dx = \frac{1}{2}\int_0^1 f(\xi)\xi^\kappa d\xi \cong \frac{1}{2}\sum_{i=1}^{n+1} f(\xi_i) \int_0^1 \ell_i(\xi)\, \xi^\kappa d\xi = \sum_{i=1}^{n+1} W_i f(\xi_i) \quad \text{or} \\ &W_i = \frac{1}{2}\int_0^1 \ell_i(\xi)\xi^\kappa d\xi \end{aligned} \tag{2.6}$$

where $\xi = x^2$, γ = 0,1,2 and $\kappa = (\gamma - 1)/2$, so κ = -½, 0, +½ for planar, cylindrical or spherical geometry. These are called interpolatory quadrature formulas, because the approximate integration can be derived by integration of the interpolant of the integrand.

Kopal (1955) has a particularly clear and understandable introduction to numerical integration methods. Eq. (2.5) has a total of n+2 weights. In an equally spaced Newton-Cotes method, these free parameters can fit the coefficients of an n+1 degree polynomial to achieve exact integration. After one is familiar with Newton-Cotes quadrature, it seems almost unbelievable that it is possible to achieve almost twice the accuracy with a similar formula, i.e. 2n+1 degrees. Take a moment to consider this possibility. Since, by definition, Eq. (2.5) includes the endpoints, there are 2n+2 free parameters, n+2 weights and n base points. We could use brute force and set up a system of equations to solve for the weights and basepoints to produce exact integrals through 2n+1 degrees. We would find that a solution is possible and except for the endpoints, all the base points are within the interval $0 < x_i < 1$. The result would be Lobatto quadrature which is described in Section 2.4.3. Lobatto quadrature is a close cousin of Gaussian quadrature. Gaussian quadrature achieves the highest accuracy, 2n-1 degrees, for a given number of base points, so it minimizes the number of function evaluations, $f(x_i)$. It differs from Lobatto quadrature, because endpoint weights are not utilized, i.e. $W_0 = W_{n+1} = 0$. Radau quadrature is similar and utilized a weight at one endpoint to achieve accuracy of 2n degrees. Clenshaw-Curtis (1960) quadrature uses Chebyshev points as the quadrature base points. The points are not optimally located for these integrals, so like Newton-Cotes only an n+1 degree polynomial can be integrated exactly.

This chapter on basic calculations is divided into several parts. To apply orthogonal collocation, pseudospectral or differential quadrature methods, one first needs the polynomial roots or quadrature base points, $\boldsymbol{x}$. Section 2.1 describes basic properties of the orthogonal polynomials for which the roots are sought, while Section 2.3 discusses methods for calculating their roots. Section 2.2 covers differentiation of the orthogonal polynomials. Derivatives of the polynomials are needed for many purposes: quadrature and barycentric weights, iterative root calculation, nodal differentiation matrices, and MWR solutions with a modal basis. Section 2.4 describes calculation of quadrature and barycentric weights, i.e. $\boldsymbol{W}$, Eqs. (2.5) and (2.6), and $\boldsymbol{W^b}$, Eq. (2.1). For nodal approximations, the differentiation matrices for first and second derivatives are described in Section 2.5. Calculation of the related stiffness and mass matrices are discussed in Sections 2.6 and 2.7. Although this monograph uses nodal formulations, the relationship to other representations is described for completeness. Section 2.9 discusses transformations between modal and nodal bases, i.e. Eq. (1.2) vs (2.1), while Section 2.10 describes transforms from nodal to monomial bases. Section 2.11 describes software to perform the fundamental calculations and outlines the coding style for all software in the project. Several examples of interpolation, integration and differentiation calculations using supplied software and methods of this chapter are described in Section 2.12. These last two sections are essential reading for anyone planning to apply these methods.

Detailed knowledge of how these various quantities are calculated is not essential for one to apply these methods. Formulas for the various quantities could be just written down and the computer codes can be used as a black box. However, background information is included here for those interested in a deeper understanding of the methods. We do not delve into the proofs and development of various formulas which are readily available elsewhere but state the results that are needed for our purposes and provide the applicable reference for details. For general references to the subject of orthogonal polynomials and quadrature the reader is directed to Hildebrand (1987) and Krylov (1962).

2.1 Jacobi Polynomials

Jacobi polynomials are a family of orthogonal polynomials which include all the polynomials of interest for solving nonperiodic problems on a finite interval. Orthogonal polynomials and quadrature formulas are conventionally based on the interval [-1,1], while the normal orthogonal collocation convention uses the more convenient interval [0,1]. On the interval [0,1] the polynomials are called *shifted* polynomials. The interval [-1,1] is used here for this fundamental development. Once the fundamental properties are established, the corresponding properties for the shifted polynomials are given. Any interval can be used with a suitable transformation.

Jacobi polynomials are orthogonal with respect to a weight function with parameters α and β. For endpoints a and b they meet the orthogonality condition:

$$\int_a^b (b-x)^\alpha (x-a)^\beta P_n^{(\alpha,\beta)}(x) P_m^{(\alpha,\beta)}(x) dx = \zeta_n^{(\alpha,\beta)} \delta_{nm} \tag{2.7}$$

where by convention for a = -1 and b = 1:

$$\zeta_n^{(\alpha,\beta)} = \frac{2^{\alpha+\beta+1}\,\Gamma(n+\alpha+1)\,\Gamma(n+\beta+1)}{(2n+\alpha+\beta+1)\,\Gamma(n+\alpha+\beta+1)\,n!}$$

Two cases of interest are $\zeta_n^{(0,0)} = 2/(2n+1)$ for Legendre polynomials and $\zeta_n^{(1,1)} = 8(n+1)/[(2n+3)(n+2)]$. The requirement of orthogonality establishes the polynomials only within a multiplicative constant. Several conventions could be used to complete the specification. If orthonormality is required, the polynomial would be scaled so that ζ_n = 1. Alternatively, monic polynomials could be specified, where the coefficient of the highest order term is unity. The convention above is established from the endpoint condition given below in Eq. (2.11). The Legendre polynomials correspond to $\alpha = \beta = 0$ and Chebyshev polynomials of the 2nd kind to $\alpha = \beta = ½$. For the common case when $\alpha = \beta$, the polynomials are called *ultraspherical* or *Gegenbauer* polynomials. Although Chebyshev polynomials are Jacobi polynomials, they are traditionally scaled differently.

Orthogonal polynomials are closely tied to the theory of accurate quadrature methods. For Eq. (2.5), Gaussian quadrature gives the highest accuracy for a given number of quadrature base points. Consider a more general integration of the form:

$$\int_{-1}^{1} f(x)\omega(x)dx = \sum_{i=1}^{m} W_i^* f(x_i) \tag{2.8}$$

where: $\omega(x) = (1-x)^{\alpha}(1+x)^{\beta}$. The optimal quadrature is Jacobi-Gauss quadrature, where the base points are the roots of the Jacobi polynomials $P_m^{(\alpha,\beta)}$. It can be proven that all the roots lie within the interval, -1 < x < 1. Jacobi-Gauss quadrature will produce exact integrals when $f(x)$ is a polynomial of degree 2m-1.

Eq. (2.8) can be used to develop most of the quadrature formulas of interest. Krylov (1962) shows that if some of the base points are prescribed and the others are determined for optimal accuracy; the weighting function in the orthogonal polynomial must be zero at the specified points. Lobatto quadrature includes both endpoints, so the weight function is made zero at both ends by taking $\alpha = \beta = 1$. Lobatto quadrature interior base points are the roots of the corresponding Jacobi polynomials. As we shall see these points also correspond to the extrema for the Legendre polynomials. For symmetric problems if a weight for Eq. (2.6) is included at the endpoint x = 1 it is correctly called a Radau quadrature; however, these formulas correspond to the right half of Lobatto formulas, so we will call them Lobatto in all cases to simplify the terminology. For symmetric problems β must correspond to κ in Eq. (2.6).

Taking these factors into account for symmetric and nonsymmetric problems of various geometry, Table 2.1 summarizes the specific Jacobi polynomials of interest for quadrature calculations. Nonsymmetric problems in cylindrical and spherical geometry are not considered because such problems make sense only if the other periodic angular coordinates are included and we do not consider periodic problems here.

Table 2.1 Weight Exponents for Jacobi Polynomials

	Planar	Cylindrical	Spherical
Nonsymmetric, Gauss	$\alpha = 0, \beta = 0$	n.a.	n.a.
Nonsymmetric, Lobatto	$\alpha = 1, \beta = 1$	n.a.	n.a.
Symmetric, Gauss	$\alpha = 0, \beta =$ -½	$\alpha = 0, \beta = 0$	$\alpha = 0, \beta =$ ½
Symmetric, Lobatto	$\alpha = 1, \beta =$ -½	$\alpha = 1, \beta = 0$	$\alpha = 1, \beta =$ ½

Note that the Chebyshev weight factors, $\alpha = \beta =$ -½ for the 1st kind and $\alpha = \beta =$ +½ for the 2nd kind, do not appear in Table 2.1. Their roots give optimal quadratures only for integrands involving radicals. If the weight $1/\sqrt{1-x^2}$ is included in the integral, Chebyshev polynomials of first kind produce a Chebyshev-Gauss formula with no endpoint weights, while Chebyshev polynomials of the 2nd kind produce Chebyshev-Gauss-Lobatto quadrature with endpoint weights. One can define MWR which includes radicals in the weighting of the residual function, Eq. (1.9), but such methods are not considered here. The weights for Chebyshev-Gauss and Chebyshev-Gauss-Lobatto quadrature can be found in many texts [Krylov (1962), Canuto, *et al.* (1988)]. Chebyshev polynomials are considered in this monograph primarily because they are a popular choice due to their computational advantages when FFT (Fast Fourier

Transform) calculations can be used. Unlike Chebyshev quadrature, the Clenshaw-Curtis (1960) quadrature described in Section 2.4.6 does not include the radical term. Since the base points are not optimally located for a unit weight, Clenshaw-Curtis quadrature does not achieve the same degree of accuracy as Gaussian-type quadratures. The quadrature formulas are compared for an example problem in Section 2.12.

The Jacobi polynomials and their properties can be determined directly from Eq. (2.7). The first few Jacobi polynomials on [-1,1] are:

$$\begin{aligned} P_0^{(\alpha,\beta)} &= 1, \\ P_1^{(\alpha,\beta)} &= \tfrac{1}{2}(\alpha+\beta+2)x + \tfrac{1}{2}(\alpha-\beta) \\ P_2^{(\alpha,\beta)} &= \tfrac{1}{8}[(3+\alpha+\beta)(4+\alpha+\beta)x^2 + 2(3+\alpha+\beta)(\alpha-\beta)x \\ &\quad + (4+\alpha+\beta) + (\alpha-\beta)^2] \end{aligned} \tag{2.9}$$

The polynomials are designated with superscript (α,β) when needed to avoid ambiguity. For Legendre or generic polynomials or when the meaning is obvious no superscript is used. When $\alpha = \beta$ the ultraspherical polynomials are alternately odd and even or symmetric and antisymmetric about x = 0.

The polynomials possess the following symmetry:

$$P_n^{(\alpha,\beta)}(-x) = (-1)^n P_n^{(\beta,\alpha)}(x) \tag{2.10}$$

The endpoints for the conventional form of the polynomials are:

$$\begin{aligned} P_n^{(\alpha,\beta)}(1) &= \frac{\Gamma(n+\alpha+1)}{n!\,\Gamma(\alpha+1)} \quad \text{and} \\ P_n^{(\alpha,\beta)}(-1) &= (-1)^n \frac{\Gamma(n+\beta+1)}{n!\,\Gamma(\beta+1)} \end{aligned} \tag{2.11}$$

The endpoint values in Eq. (2.11) establish the convention by which the formula for ζ is determined in Eq. (2.7). For Legendre polynomials, the values at x = 1 are unity and are alternating ±1 at x = -1. For the Jacobi $\alpha = \beta$ = 1 polynomials they are n + 1 at x = 1 and alternating ±(n + 1) at x = -1. Chebyshev polynomials of the first kind are traditionally scaled to have endpoint and extrema values of ±1; however, for the Jacobi scaling in Eq. (2.11), the absolute value at the endpoints are $P_n^{\left(\frac{-1}{2},\frac{-1}{2}\right)}(1) = \Gamma(n+\frac{1}{2})/[\sqrt{\pi}\,\Gamma(n+1)]$ = {1,½,⅜,$^5/_{16}$, $^{35}/_{128}$, …}. Chebyshev polynomials of the second kind traditionally have endpoints of ± (n + 1), while with Jacobi scaling $P_n^{\left(\frac{1}{2},\frac{1}{2}\right)}(1) = (2n+1)P_n^{\left(\frac{-1}{2},\frac{-1}{2}\right)}(1)$ = {1,1½,1⅞,2 $^9/_{48}$, …}.

The polynomials can be expanded as linear combination of monomials like Eq. (2.9), but for the higher order polynomials the coefficients become large with alternating signs. If used for calculations in that form roundoff errors soon become important. There is a better way to evaluate the polynomials.

The orthogonality condition, Eq. (2.7), can be used to develop a simple recurrence relationship. Dropping the superscript (α,β) for convenience. The recurrence formula for the monic form is:

$$p_{n+1} = (x - \hat{\alpha}_n)p_n - \hat{\beta}_n p_{n-1} \tag{2.12}$$

First define $p_0 = 1,\ p_{-1} = 0,$ and $\hat{\beta}_0 = \int_{-1}^{1} \omega(x)\,dx$, then the other parameters are:

$$\hat{\alpha}_n = \frac{\int_{-1}^{1} p_n^2\, x\, \omega(x)dx}{\int_{-1}^{1} p_n^2\, \omega(x)dx} = \frac{\beta^2 - \alpha^2}{(2n+\alpha+\beta)(2n+\alpha+\beta+2)} \quad \text{and}$$

$$\hat{\beta}_n = \frac{\int_{-1}^{1} p_n^2\, \omega(x)dx}{\int_{-1}^{1} p_{n-1}^2\, \omega(x)dx} = \frac{4n(n+\alpha)(n+\beta)(n+\alpha+\beta)}{(2n+\alpha+\beta)^2[(2n+\alpha+\beta)^2-1]}$$

Eq. (2.12) can be derived directly from the orthogonality condition, Eq. (2.7), and from the fact that any polynomial can be expressed as a linear combination of orthogonal polynomials.

Eq. (2.12) is the recurrence relationship for the monic polynomials. The leading coefficients for the conventional form in Eq. (2.7) are:

$$P_n = \rho_n p_n \tag{2.13}$$

Where ρ_n can be determined from:

$$\zeta_n = \int_{-1}^{1} \rho_n^2\, p_n^2\, \omega(x)dx = \rho_n^2 \prod_{k=0}^{n} \hat{\beta}_k \tag{2.14}$$

For the convention given in Eq. (2.7), the leading coefficients are:

$$\rho_n = \frac{\Gamma(2n+\alpha+\beta+1)}{2^n n!\,\Gamma(n+\alpha+\beta+1)} \tag{2.15}$$

A recurrence relationship for the conventional form of the Jacobi polynomials can easily be determined from the expressions above:

$$P_{n+1} = [\breve{\gamma}_n x - \breve{\alpha}_n]P_n - \breve{\beta}_n P_{n-1} \tag{2.16}$$

where the three coefficients in the recurrence relation are:

$$\breve{\alpha}_n = \hat{\alpha}_n \frac{\rho_{n+1}}{\rho_n} = \frac{(\beta^2-\alpha^2)(2n+\alpha+\beta+1)}{2(n+1)(n+\alpha+\beta+1)(2n+\alpha+\beta)}$$

$$\breve{\beta}_n = \hat{\beta}_n \frac{\rho_{n+1}}{\rho_{n-1}} = \frac{(n+\alpha)(n+\beta)(2n+\alpha+\beta+2)}{(n+1)(n+\alpha+\beta+1)(2n+\alpha+\beta)}$$

$$\breve{\gamma}_n = \frac{\rho_{n+1}}{\rho_n} = \frac{(2n+\alpha+\beta+1)(2n+\alpha+\beta+2)}{2(n+1)(n+\alpha+\beta+1)}$$

These equations are for polynomials on the interval [-1,1]. They can be converted to the interval [0,1], which is more convenient and has become the standard for orthogonal

collocation applications. These are called *shifted* polynomials. Using a tilde (~) to designate the corresponding values for shifted polynomials, the parameters are:

$$\begin{aligned}\tilde{\zeta}_n^{(\alpha,\beta)} &= \zeta_n^{(\alpha,\beta)}/(2^{\alpha+\beta+1})\\ \tilde{\rho}_n^{(\alpha,\beta)} &= (2^n)\rho_n^{(\alpha,\beta)}\\ \tilde{\alpha}_n^{(\alpha,\beta)} &= \tfrac{1}{2}\left(1+\hat{\alpha}_n^{(\alpha,\beta)}\right)\\ \tilde{\beta}_n^{(\alpha,\beta)} &= \tfrac{1}{4}\hat{\beta}_n^{(\alpha,\beta)}\end{aligned} \qquad (2.17)$$

Fig. 2.1 shows some examples of 5th and 6th order Jacobi polynomials. Note that the polynomials tend to look like sine or cosine curves in the middle of the interval but are compressed near the boundaries. Note also that the endpoints of the polynomials agree with Eq. (2.11) (although the scale cuts off the extreme portions). $P_6^{(1,0)}$ is the only one shown which is not symmetric (or antisymmetric) about the center point. It also follows the Legendre polynomial near x = 0 and the $P_6^{(1,1)}$ polynomial near x = 1. Finally, note that the roots of $P_5^{(1,1)}$ coincide with the extrema of the Legendre polynomial. This last feature will be discussed shortly.

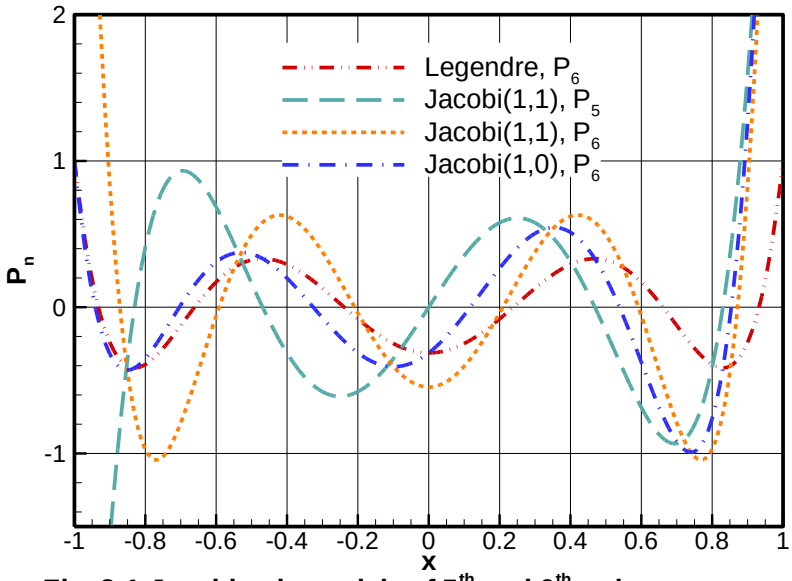


Fig. 2.1 Jacobi polynomials of 5th and 6th order

Eq. (2.10) describes the symmetry/asymmetry characteristics of these polynomials, which is illustrated in Fig. 2.1. Many calculations use the ultraspherical polynomials, $\alpha = \beta$. For some calculations, efficiency can be gained by exploiting the symmetry property. Consider the planar symmetric case in Table 2.1. These polynomials correspond to the right half of the full polynomials that are even, e.g. the Legendre polynomial, P_6, in Fig. 2.1. If, for example, we need all the roots, we could calculate the positive roots and then copy them to get the negative ones, but this would still involve calculations with the full polynomial. It is more efficient to work directly with polynomials in x^2 which have half as many terms.

Table 2.1 hints at an equivalence between the symmetric and nonsymmetric polynomials. For example, the roots and quadrature weights for symmetric planar geometry on the interval [0,1] should correspond to values with $\alpha = \beta$ on the interval [-1,1]. The symmetric polynomials having the same value of α but with β = -½ are designated by S. The equivalence between the polynomials is [Olver et. al (2018), 18.7.13-14]:

$$P_{2n}^{(\alpha,\alpha)}(x) = \tilde{a}_{2n} S_n^{(\alpha,-\frac{1}{2})}(\xi) \tag{2.18}$$

Although the symbol, S, is used to distinguish the symmetric polynomials, they are just ordinary Jacobi polynomials. If both are defined on the interval [-1,1], the independent variables are related by $\xi = 2x^2 - 1$. If we also use the conventional normalization which gives ζ as defined in Eq. (2.7), then the proportionality constant can be determine from the ratio of endpoint values, Eq. (2.11):

$$\tilde{a}_{2n} = \frac{P_{2n}^{(\alpha,\alpha)}(1)}{S_n^{(\alpha,-\frac{1}{2})}(1)} = \frac{n!\,\Gamma(2n+\alpha+1)}{(2n)!\,\Gamma(n+\alpha+1)}$$

For Legendre polynomials the endpoint values are unity, so $\tilde{a}_{2n} = 1$, for the Lobatto case, $\alpha = \beta = 1$, $\tilde{a}_{2n} = (2n+1)/(n+1)$. The proportionality constant for monic polynomials is simply the conversion of the leading coefficient for a shifted polynomial, Eq. (2.17), i.e. 2^{-n}. Given the roots of S, those of the corresponding ultraspherical polynomial are given by $\boldsymbol{x} = \sqrt{(1+\boldsymbol{\xi})/2}$. The roots of S can be determined with fewer calculations, since there are not only half as many roots, but also half as many terms in the polynomial. We will refer to these as the *shortcut* polynomials.

Eq. (2.18) gives the relationship between shortcut polynomials with the even numbered ultraspherical polynomials. Although not as straight forward as for even n, a similar relationship exists for odd n [Krylov (1962) pp 117-121]. An odd ultraspherical polynomial is antisymmetric about the center line, so there are an odd number of roots and one root is at the center, $x = 0$. Division by x creates a symmetric polynomial, which can be treated like the case when n is even. Suppose we define the symmetric polynomial, S, as follows:

$$S_n^{(\alpha,+\frac{1}{2})}(x) = \frac{P_{2n+1}^{(\alpha,\alpha)}(\sqrt{x})}{\sqrt{x}} \quad \text{or} \quad S_n^{(\alpha,+\frac{1}{2})}(x) S_m^{(\alpha,+\frac{1}{2})}(x)\sqrt{x} = \frac{P_{2n+1}^{(\alpha,\alpha)}(\sqrt{x}) P_{2m+1}^{(\alpha,\alpha)}(\sqrt{x})}{\sqrt{x}} \tag{2.19}$$

Dropping the superscripts for convenience, the relationship of their orthogonality is:

$$\begin{aligned} &\int_0^1 S_n(x)\, S_m(x)\sqrt{x}\,(1-x)^\alpha dx \\ &\quad = \int_0^1 P_{2n+1}(\sqrt{x})\, P_{2m+1}(\sqrt{x})(1-x)^\alpha \frac{dx}{\sqrt{x}} \\ &\quad = 2\int_{-1}^1 P_{2n+1}(\eta)\, P_{2m+1}(\eta)\,(1-\eta)^\alpha(1+\eta)^\alpha d\eta = 0 \text{ for } m \neq n \end{aligned} \tag{2.20}$$

where $\eta = \sqrt{x}$. So, the positive roots of P_{2n+1} (on [-1,1]) are the square root of the roots of S_n (on [0,1] with $\beta = +½$). The other roots are the corresponding negative ones and the one at $x = 0$. The relationship above, proves proportionality of the two polynomials. To achieve an exact correspondence, a proportionality constant is determined from the endpoint values, Eq. (2.11).

$$P_{2n+1}^{(\alpha,\alpha)}(x) = \tilde{a}_{2n+1} x\, S_n^{(\alpha,+\frac{1}{2})}(\xi) \tag{2.21}$$

If $\hat{S}_n$ is defined on the interval [-1,1], $\xi = 2x^2 - 1$ and the proportionality constant is:

$$\tilde{a}_{2n+1} = \frac{P_{2n+1}^{(\alpha,\alpha)}(1)}{S_n^{(\alpha,+\frac{1}{2})}(1)} = \frac{n!\,\Gamma(2n+\alpha+2)}{(2n+1)!\,\Gamma(n+\alpha+1)}$$

For Legendre polynomials $\tilde{a}_{2n+1} = 1$, while for the Lobatto case, $\alpha = \beta = 1$, $\tilde{a}_{2n+1} = 2$. If monic polynomials are used, the proportionality constant is 2^{-n}.

These shortcut polynomials are handy for reducing calculations for the ultraspherical polynomials. The degree and the number of coefficients is cut in half, so a polynomial value can be calculated with half the effort. They also provide an analytical expression for values of the even numbered polynomials at $x = 0$:

$$\begin{aligned} P_{2n}^{(\alpha,\alpha)}(0) = \tilde{a}_{2n} S_n^{\left(\alpha,-\frac{1}{2}\right)}(-1) &= P_{2n}^{(\alpha,\alpha)}(1)\frac{S_n^{\left(\alpha,-\frac{1}{2}\right)}(-1)}{S_n^{\left(\alpha,-\frac{1}{2}\right)}(1)} \\ &= (-1)^n \frac{\Gamma\left(n+\frac{1}{2}\right)\Gamma(2n+\alpha+1)}{\sqrt{\pi}\,(2n)!\,\Gamma(n+\alpha+1)} \end{aligned} \tag{2.22}$$

We also note that given this relationship between the polynomials, there are only three unique cases of interest in Table 2.1: (1) Gauss $\alpha = \beta = 0$, (2) Lobatto $\alpha = \beta = 1$ and (3) Radau $\alpha = 1, \beta = 0$. The symmetric cases for planar and spherical geometry correspond to the shortcut representation for ultraspherical polynomials.

From approximation theory, we know that for polynomial interpolation of a function, the error is proportional to the polynomial whose roots are at the interpolation nodes. Chebyshev polynomials of the 1st kind are the best ones for interpolation, because the extrema are uniform and the error is evenly distributed (Lanczos (1956), p.245; Hildebrand (1987) p. 469).

In MWR our goal is to closely approximate the true solution. However, since the solution is not known, we instead make the residual of the differential equation as close to zero as possible. The residual in a collocation method will obey the same principals as in the interpolation of any function. However, a uniformly distributed error in the residual of a differential equation does not produce a uniform error in the solution. Lanczos (1956, p.477) shows for a simple first order differential equation, the dominant errors are shifted by the differential operator. Errors in the solution are less when the residual is proportional to Chebyshev polynomials of the 2nd kind or Legendre polynomials. Larger values of the α and β parameters tend to be better for MWR.

In later chapters we will see that the distribution of the error is influenced by the distribution of the roots and the variation in the amplitude of the polynomial's fluctuations. These characteristics are determined by the polynomial weight function, $\omega(x) = (1-x)^\alpha(1+x)^\beta$, and its value in the center region relative to the boundary area. Larger values of α and β decreases the magnitude of the weight near the boundary relative to the center. Table 2.2 lists the ratio of

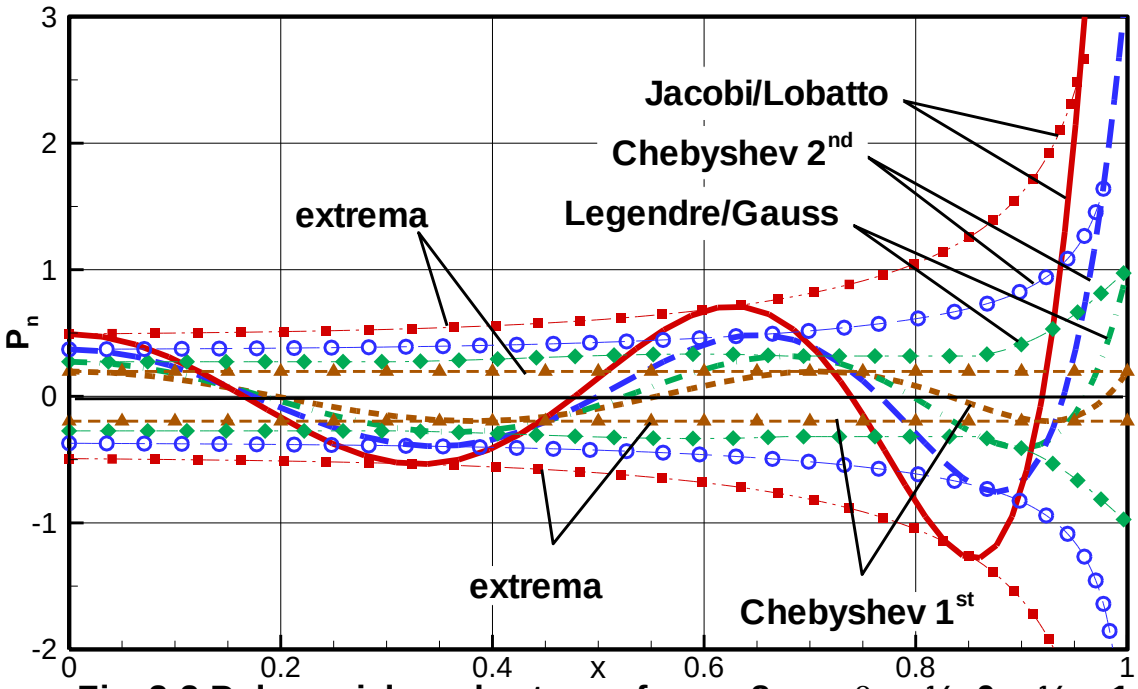


Fig. 2.2 Polynomials and extrema for n = 8, α = β = -½, 0, +½, +1

center to endpoint values, calculated with Eqs (2.11) and (2.22), for four polynomials. These polynomials have increasing values of the weight parameters: (1) Chebyshev 1st kind, $\alpha = \beta$ = -½, (2) Legendre $\alpha = \beta = 0$, (3) Chebyshev 2nd kind $\alpha = \beta$ = +½, and (4) Jacobi with $\alpha = \beta = 1$ for Lobatto quadrature. Fig. 2.2 shows the right half of the same four polynomials for n = 8 together with the locus of their extrema. The increasing α and β values emphasizes the central portion and concentrate the roots in that area. However, the boundary area is deemphasized, since values near the boundary are significantly larger. In Fig. 2.2 the Chebyshev polynomials are scaled like other Jacobi polynomials. What is important is the variation in x not the scaling.

Table 2.2 Relative Boundary to Center Values

n	Chebyshev (-½,-½)	Legendre (0,0)	Chebyshev (+½,+½)	Jacobi (1,1)
2	1	2	3	4
4	1	2.667	5	8
6	1	3.200	7	12.80
8	1	3.657	9	18.29
10	1	4.063	11	24.38

As an example of a Jacobi polynomial, consider the Legendre polynomials, $\alpha = \beta = 0$. The recurrence relation and leading coefficients for the interval [-1,1] are:

$$\hat{\alpha}_n = 0$$
$$\hat{\beta}_n = \frac{n^2}{4n^2 - 1}$$
$$\rho_n = \frac{(2n)!}{2^n (n!)^2} \qquad (2.23)$$
$$\zeta_n = \frac{2}{2n + 1}$$

The recurrence relationship for the monic form of the Legendre polynomials is then:

$$p_{n+1} = x\,p_n - \left(\frac{n^2}{4n^2-1}\right)p_{n-1} \tag{2.24}$$

While the recurrence relationship for the conventional form of the Legendre polynomials is given by:

$$P_{n+1} = \left(\frac{2n+1}{n+1}\right)xP_n - \left(\frac{n}{n+1}\right)P_{n-1} \tag{2.25}$$

The first few Legendre polynomials are:

$$P_0 = 1,\ P_1 = x,\ \ P_2 = \frac{1}{2}(3x^2-1),\ \ P_3 = \frac{1}{2}(5x^3-3x) \tag{2.26}$$

2.2 Differentiation of Jacobi Polynomials

Derivatives of the Jacobi polynomials are needed for many of the fundamental calculations. We have already seen that they appear as the barycentric weights, $W_i^b = 1/\hat{p}_n'(x_i)$, in the interpolation formula, Eq. (2.1). These same quantities appear in the integration formulas and nodal differentiation relationships. Derivatives are also needed when an iterative, e.g. Newton-Raphson, method is used to determine roots as discussed in Section 2.3.

Derivative relationships are needed when *modal* methods, i.e. orthogonal polynomial trial functions, are used with MWR. Although a *nodal* formulation is usually more convenient, occasionally orthogonal polynomial trial functions are used, Eq. (1.2), where the coefficients are analogous to the modes in a Fourier series. Here we describe some orthogonal polynomial relationships that are useful for a *modal* formulation with Jacobi polynomial basis functions.

The most common modal basis functions are Chebyshev polynomials and Legendre polynomials. However, other types of Jacobi polynomials have been used. We will consider the general case but will emphasize the Legendre polynomials. Chebyshev basis functions are heavily covered elsewhere [Canuto, et al. (1988), Boyd (2000), Trefethen (2000), Shen, et al. (2011)]. All of the discussion here is for the interval [-1,1].

One interesting and useful property of Jacobi polynomials is:

$$\frac{dP_n^{(\alpha,\beta)}(x)}{dx} = \tfrac{1}{2}(n+\alpha+\beta+1)\,P_{n-1}^{(\alpha+1,\beta+1)}(x) \tag{2.27}$$

so the roots of the polynomial on the right corresponds to the extrema of the one indicated on the left. This relationship is apparent in Fig. 2.1. It explains why the Lobatto quadrature base points are often called the extrema of the Legendre polynomials. They are also the roots of the Jacobi polynomials with $\alpha = \beta = 1$. By the same token, the extrema of the Chebyshev polynomials of the 1st kind ($\alpha = \beta = -½$) are the roots of the Chebyshev polynomials of the 2nd kind ($\alpha = \beta = +½$).

The derivatives at the boundaries are easily found by using Eq. (2.27) with Eq. (2.11) to give:

$$\left.\frac{dP_n^{(\alpha,\beta)}}{dx}\right|_{x=+1} = \tfrac{1}{2}(n+\alpha+\beta+1)\,\frac{\Gamma(n+\alpha+2)}{n!\,\Gamma(\alpha+2)} \quad \text{and}$$
$$\left.\frac{dP_n^{(\alpha,\beta)}}{dx}\right|_{x=-1} = \tfrac{1}{2}(n+\alpha+\beta+1)\,(-1)^n\frac{\Gamma(n+\beta+2)}{n!\,\Gamma(\beta+2)} \tag{2.28}$$

The derivative at the x = 1 are $n(n{+}1)/2$ for Legendre polynomials and $n(n{+}1)\,(n{+}3)/4$ for Jacobi polynomials with $\alpha = \beta = 1$.

Derivatives of the polynomials can be found by direct differentiation of the recurrence relationship, Eq. (2.16):

$$P'_{n+1} = \check{\gamma}_n P_n + [\check{\gamma}_n x - \check{\alpha}_n]P'_n - \check{\beta}_n P'_{n-1} \tag{2.29}$$

where primes denote first derivatives.

Derivatives of the ultraspherical polynomials, $\alpha = \beta$, also can be calculated using the shortcut relationships given in Eqs. (2.18) and (2.21). Since $\xi = 2x^2 - 1$, differentiation of these relationships for an even numbered polynomial gives:

$$\frac{dP_{2n}^{(\alpha,\alpha)}(x)}{dx} = 2\sqrt{2}\,\tilde{a}_{2n}\sqrt{1+\xi}\;\frac{dS_n^{(\alpha,-\frac{1}{2})}(\xi)}{d\xi} \tag{2.30}$$

The derivative of an odd numbered polynomial is:

$$\frac{dP_{2n+1}^{(\alpha,\alpha)}(x)}{dx} = \tilde{a}_{2n+1}\left(S_n^{(\alpha,+\frac{1}{2})}(\xi) + 2(1+\xi)\frac{dS_n^{(\alpha,+\frac{1}{2})}(\xi)}{d\xi}\right) \tag{2.31}$$

Calculating a derivative of S requires less effort because the polynomials have half as many terms.

Jacobi polynomials are the eigenfunctions of a singular Sturm-Liouville equation:

$$\begin{aligned}
&\frac{d}{dx}\left[(1-x)^{\alpha+1}(1+x)^{\beta+1}\frac{dP_k^{(\alpha,\beta)}}{dx}\right] + \acute{c}_k^{(\alpha,\beta)}(1-x)^{\alpha}(1+x)^{\beta}P_k^{(\alpha,\beta)} = 0, \;\text{ or}\\
&(1-x^2)P_k^{(\alpha,\beta)''} - [\alpha-\beta+(\alpha+\beta+2)x]P_k^{(\alpha,\beta)'} + \acute{c}_k^{(\alpha,\beta)}P_k^{(\alpha,\beta)} = 0, \;\text{ or}\\
&(1-x^2)P_k^{(\alpha,\beta)''} = [\acute{a}^{(\alpha,\beta)}\,x + \acute{b}^{(\alpha,\beta)}]P_k^{(\alpha,\beta)'} - \acute{c}_k^{(\alpha,\beta)}P_k^{(\alpha,\beta)}
\end{aligned} \tag{2.32}$$

where $\acute{c}_k^{(\alpha,\beta)} = k(k+\alpha+\beta+1)$ are the eigenvalues. This equation is useful for calculating second and higher derivatives. Repeated differentiation of Eq. (2.32) produces the following recurrence relationship for higher derivatives:

$$(1-x^2)P_k^{(i)} = (\acute{a}_i\,x + \acute{b})P_k^{(i-1)} - \acute{c}_{ki}P_k^{(i-2)} \tag{2.33}$$

where the (α,β) superscript has been omitted for convenience and replaced with (i) indicating the i^{th} derivative. Starting with the coefficients in Eq. (2.32), those for the higher derivatives are:

$$\acute{a}_i = \acute{a}_{i-1} + 2$$
$$\acute{c}_{ki} = \acute{c}_{k,i-1} - \acute{a}_{i-1}$$

A simple relationship for determining first derivatives is:

$$(1-x^2)P_k^{(\alpha,\beta)\prime} = \bar{c}_k^{(\alpha,\beta)} P_{k-1}^{(\alpha,\beta)} - \bar{b}_k^{(\alpha,\beta)} P_k^{(\alpha,\beta)} - \bar{a}_k^{(\alpha,\beta)} P_{k+1}^{(\alpha,\beta)} \tag{2.34}$$

where the coefficients are: :

$$\bar{a}_k^{(\alpha,\beta)} = \frac{2k(k+1)(k+\alpha+\beta+1)}{(2k+\alpha+\beta+1)(2k+\alpha+\beta+2)}$$
$$\bar{b}_k^{(\alpha,\beta)} = \frac{2k(\beta-\alpha)(k+\alpha+\beta+1)}{(2k+\alpha+\beta)(2k+\alpha+\beta+2)}$$
$$\bar{c}_k^{(\alpha,\beta)} = \frac{2(k+\alpha)(k+\beta)(k+\alpha+\beta+1)}{(2k+\alpha+\beta)(2k+\alpha+\beta+1)}$$

The recurrence relationship, Eq. (2.16), can be substituted for P_{k+1} to give the expression:

$$\begin{aligned}(1-x^2)P_k^{(\alpha,\beta)\prime} &= \left(\bar{a}_k^{(\alpha,\beta)}\check{\beta}_k + \bar{c}_k^{(\alpha,\beta)}\right) P_{k-1}^{(\alpha,\beta)} - \left(\bar{a}_k^{(\alpha,\beta)}[\check{\gamma}_k x - \check{\alpha}_k] + \bar{b}_k^{(\alpha,\beta)}\right) P_k^{(\alpha,\beta)} \\ &= \check{c}_k^{(\alpha,\beta)} P_{k-1}^{(\alpha,\beta)} - \left(\check{a}_k^{(\alpha,\beta)} x + \check{b}_k^{(\alpha,\beta)}\right) P_k^{(\alpha,\beta)}\end{aligned} \tag{2.35}$$

Substituting values gives:

$$\check{a}_k^{(\alpha,\beta)} = \bar{a}_k^{(\alpha,\beta)}\check{\gamma}_k^{(\alpha,\beta)} = k$$
$$\check{b}_k^{(\alpha,\beta)} = \bar{b}_k^{(\alpha,\beta)} - \bar{a}_k^{(\alpha,\beta)}\check{\alpha}_k^{(\alpha,\beta)} = \frac{k(\beta-\alpha)}{(2k+\alpha+\beta)}$$
$$\check{c}_k^{(\alpha,\beta)} = \bar{c}_k^{(\alpha,\beta)} + \bar{a}_k^{(\alpha,\beta)}\check{\beta}_k^{(\alpha,\beta)} = \frac{2(k+\alpha)(k+\beta)}{(2k+\alpha+\beta)}$$

This expression is especially simple for calculating derivatives at the roots, since P_k drops out, but it may not be the best procedure.

By virtue of Eq. (2.27), the derivatives of the polynomials also form an orthogonal set, with a recurrence relationship like Eq. (2.16). The recurrence relationship for the derivatives can be combined with Eq. (2.29) to yield the following:

$$P_k^{(\alpha,\beta)} = \bar{\bar{a}}_k^{(\alpha,\beta)} P_{k+1}^{(\alpha,\beta)\prime} - \bar{\bar{b}}_k^{(\alpha,\beta)} P_k^{(\alpha,\beta)\prime} - \bar{\bar{c}}_k^{(\alpha,\beta)} P_{k-1}^{(\alpha,\beta)\prime} \tag{2.36}$$

where the primes again denote the first derivatives and:

$$\bar{\bar{a}}_k^{(\alpha,\beta)} = \frac{2(k+\alpha+\beta+1)}{(2k+\alpha+\beta+1)(2k+\alpha+\beta+2)}$$
$$\bar{\bar{b}}_k^{(\alpha,\beta)} = \frac{2(\beta-\alpha)}{(2k+\alpha+\beta)(2k+\alpha+\beta+2)}$$

$$\bar{\bar{c}}_k^{(\alpha,\beta)} = \frac{2(k+\alpha)(k+\beta)}{(k+\alpha+\beta)(2k+\alpha+\beta)(2k+\alpha+\beta+1)}$$

Eq. (2.36) can be solved for $P_{k+1}^{(\alpha,\beta)\prime}$ to give:

$$P_{k+1}^{(\alpha,\beta)\prime} = \bar{\alpha}_k^{(\alpha,\beta)} P_k^{(\alpha,\beta)\prime} + \bar{\beta}_k^{(\alpha,\beta)} P_{k-1}^{(\alpha,\beta)\prime} + \bar{\gamma}_k^{(\alpha,\beta)} P_k^{(\alpha,\beta)} \tag{2.37}$$

where the coefficients are conveniently expressed in terms of the recurrence coefficients of Eq. (2.16):

$$\bar{\alpha}_k^{(\alpha,\beta)} = \frac{2(k+1)}{(\alpha+\beta)} \breve{\alpha}_k^{(\alpha,\beta)}$$

$$\bar{\beta}_k^{(\alpha,\beta)} = \frac{(k+1)}{(k+\alpha+\beta)} \breve{\beta}_k^{(\alpha,\beta)}$$

$$\bar{\gamma}_k^{(\alpha,\beta)} = (k+1)\, \breve{\gamma}_k^{(\alpha,\beta)}$$

Eq. (2.37) and the expressions for the coefficients apply equally to the monic polynomials if the monic recurrence coefficients, $\hat{\alpha}, \hat{\beta}$ defined in Eq. (2.12), are substituted above and if $\breve{\gamma}_k$ is replaced by unity. This expression simplifies for the ultraspherical polynomials ($\alpha = \beta$), since $\bar{\alpha}_k^{(\alpha,\beta)} = 0$.

Eq. (2.37) can be used to determine a form which is useful for modal MWR applications. The right-hand-side terms are substituted recursively to calculate the coefficients of:

$$P_n^{(\alpha,\beta)\prime} = \sum_{k=0}^{n-1} d_{nk}^{(\alpha,\beta)} P_k^{(\alpha,\beta)} \tag{2.38}$$

This form is useful for modal MWR solutions, since derivatives are expressed in terms of the basic undifferentiated polynomials. For ultraspherical cases, $\alpha = \beta$, alternate values of d are zero.

The relationships above are most often used for Legendre polynomials. For the Legendre case the Sturm-Liouville relationship, Eq.(2.32), reduces to:

$$(1-x^2)P_k'' - 2xP_k' + k(k+1)P_k = 0 \tag{2.39}$$

The first derivative relationship, Eq. (2.34), reduces to:

$$(1-x^2)P_k'(x) = \frac{k(k+1)}{2k+1}(P_{k-1}(x) - P_{k+1}(x)) \tag{2.40}$$

which can be combined with the recurrence relationship to give the special case of, Eq. (2.35):

$$\frac{(1-x^2)}{k} P_k'(x) = P_{k-1}(x) - xP_k(x) \tag{2.41}$$

Eqs. (2.27) and (2.40) can be used to produce the relationship:

$$(1-x^2)P_k^{(1,1)}(x) = -\frac{2k+2}{2k+3}\left(P_{k+2}(x) - P_k(x)\right) \tag{2.42}$$

In order to use a modal expansion in Legendre polynomials to solve problems with MWR, expressions for the derivatives of Eq. (1.2) are required. For Legendre polynomials Eq. (2.36) reduces to:

$$(2k+1)P_k = P'_{k+1} - P'_{k-1} \tag{2.43}$$

This expression can be applied repeatedly so Eq. (2.38) for the Legendre case is:

$$P'_m = \sum_{\substack{k=0 \\ k+m\ odd}}^{m-1} (2k+1)P_k(x) \tag{2.44}$$

An expression for the second derivative is:

$$P''_m = \sum_{\substack{k=0 \\ k+m\ even}}^{m-2} \frac{(2k+1)}{2}[m(m+1) - k(k+1)]P_k(x) \tag{2.45}$$

Due to the alternating odd/even nature of the polynomials, only alternate values appear in the derivative expressions. The notation $k+m\ odd$ or $even$ indicates that only these values appear in the summation. For example:

$$P'_5 = P_0 + 5P_2 + 9P_4$$

and

$$P''_5 = 42P_1 + 63P_3$$

2.3 Orthogonal Polynomial Roots

One advantage of using Chebyshev points is that their roots can be directly calculated. The interior roots of Chebyshev polynomials on the interval [-1,1] are given by the general formula:

$$x_k = \cos\left(\frac{2k+\alpha-\frac{1}{2}}{2n+\alpha+\beta+1}\,\pi\right) \tag{2.46}$$

for k = 1,…,n. It is easiest to think of these roots as shown in Fig. 2.3. They are equally spaced in the angular, θ, coordinate. Where $\alpha = \beta$ = -½ for Chebyshev polynomials of the 1st kind and $\alpha = \beta$ = +½ for Chebyshev polynomials of the 2nd kind. The first kind correspond to midpoints of equal intervals in θ, while the second kind are at the endpoints of equal intervals. These roots are the base points of Chebyshev-Gauss and Chebyshev-Gauss-Lobatto quadrature. The roots with α = +½, β = -½ or α = -½, β = +½ are the base points of Chebyshev-Gauss-Radau quadrature. These quadrature formulas provide high accuracy when the integrand contains $1/\sqrt{1-x^2}$. The quadrature weights can be directly calculated [Krylov (1962), Canuto, *et al.* (1988)].

For other Jacobi polynomials, the roots cannot be found as easily. There are two basic methods for determining the roots of the other Jacobi polynomials. The most popular method reformulates the problem as an eigenvalue problem for which the roots are the eigenvalues. The other is an iterative method such as the Newton-Raphson method. We will consider the Newton-Raphson method as well as higher order iterative methods.

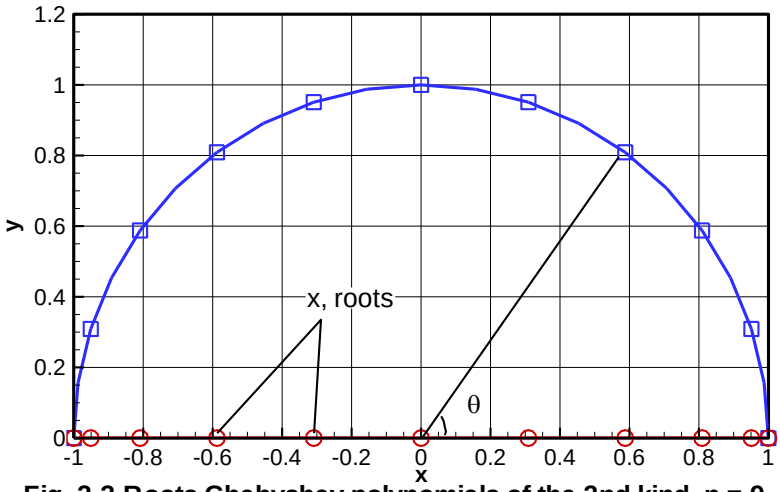


Fig. 2.3 Roots Chebyshev polynomials of the 2nd kind, n = 9

For ultraspherical polynomials, $\alpha = \beta$, both methods benefit from use of the equivalent shortcut polynomials, related by Eqs. (2.18) and (2.21). Since these polynomials have half as many terms and half as many roots, their roots can be determined with one fourth the effort. We also note that if the shortcut method is used for the ultraspherical cases, there are only three fundamentally different Jacobi polynomial roots in Table 2.1. The three cases are: (1) Gauss/Legendre $\alpha = \beta = 0$, (2) Lobatto $\alpha = \beta = 1$ and (3) Radau $\alpha = 1$, $\beta = 0$. The symmetric cases in planar and spherical geometry are the same as shortcut cases for planar geometry.

2.3.1 Eigenvalue Method

Lanczos (1956, p. 376) notes that the recurrence relations for orthogonal polynomials can be cast in the form of a tridiagonal eigenvalue problem, where the eigenvalues are the roots to the polynomial. Using this idea, Golub and Welch (1969) developed a procedure for determining the roots and quadrature weights. They cast the problem in the form of a tridiagonal matrix, called the Jacobi matrix. The matrix is constructed from the monic recurrence coefficients as follows:

$$\begin{bmatrix} \hat{\alpha}_0 & \sqrt{\hat{\beta}_1} & & & 0 \\ \sqrt{\hat{\beta}_1} & \hat{\alpha}_1 & \sqrt{\hat{\beta}_2} & & \\ & \sqrt{\hat{\beta}_2} & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots \\ 0 & & & \ddots & \ddots \end{bmatrix} \tag{2.47}$$

The recurrence relationship arranged in this manner gives the polynomials in *orthonormal* form. The orthogonal polynomial is the characteristic polynomial of this matrix, so its eigenvalues are the roots of the polynomial. Given the recurrence coefficients and a routine for calculating eigenvalues, the roots can be found with only a few lines of code (see code box below). The quadrature weights can also be found from the eigenproblem since they bear a

Matlab Function for Quadrature and Differentiation

```
function [x,w,A] = OCnonsymGLR(n,meth)
   % code for nonsymmetric orthogonal collocation applications 0 < x < 1
   % n - interior points
   % meth = 1,2,3,4 for Gauss, Lobatto, Radau (right), Radau (left)
   % x - collocation points
   % w - quadrature weights
   % A - 1st derivative
   na = [1 0 0 1];  nb = [1 0 1 0];  nt = n + 2;
   a = 1.0 - na(meth);  b = 1.0 - nb(meth);
   ab = r_jacobi(n,a,b);  ab(2:n,2) = sqrt(ab(2:n,2));
   T = diag(ab(2:n,2),-1) + diag(ab(:,1)) + diag(ab(2:n,2),+1);
   x = eig(T);  x=sort(x);  x=0.5*(x+1.0);  x = [0.0;x;1.0];
   xdif = x-x'+eye(nt);     dpx = prod(xdif,2);
   w = (x.^nb(meth)).*((1.0 .- x).^na(meth))./(dpx.*dpx); w = w/sum(w);
   A = dpx./(dpx'.*xdif);  A(1:nt+1:nt*nt) = 1.0 - sum(A,2);
 end
```

simple relationship to the eigenvectors of the Jacobi matrix. However, the code below calculates the weights using a method described in Section 2.4. It also calculates the differentiation matrix using the same parameters as described in Section 2.5. The code relies on Gautschi's (2005) OPQ `r_jacobi` function to obtain the recurrence coefficients.

2.3.2 Newton-Raphson Iterative Method

The other method for determining the roots is a standard iterative method, such as the Newton-Raphson method. For the iterative solution of a problem like this, one must make certain all roots are found and they are in order. Iterative algorithms often use a coarse scan and isolate step before switching to a second order Newton-Raphson method for refinement to the final value. The methods described here can start with a second or higher order method and will converge rapidly and without fail. Most collocation solutions converge with a small number of points. However, there are problems reported where thousands of points have been used. Our original goal here was to develop a method which is reasonably efficient and accurate for up to a few hundred points or so. However, after emersion in this compelling problem, we have taken the problem much further.

If m indicates the iteration number, each iteration of a normal Newton-Raphson method requires calculation of the polynomial value and its derivative. These quantities are used to find an improved estimate of the k^{th} root as follows:

$$x_k^{m+1} = x_k^m - p_n(x_k^m)/p_n'(x_k^m) \tag{2.48}$$

where p_{n} is usually calculated with the recurrence relationship, Eq. (2.12). The derivative of the polynomial can be calculated using one of Eqs. (2.29), (2.35) or (2.38). Eq. (2.35) requires fewer calculations than the other choices.

How can one insure the Newton-Raphson method doesn't find the same root over and over again? This problem can be avoided by using a procedure called deflation, which suppresses the roots which have been found [Villadsen and Michelsen (1978), Karniadakis and Sherwin

(2013)]. Suppose that roots 0 through k - 1 have been found and we want to calculate root k with an algorithm modified to suppress the roots already found. The lower order polynomial which includes only the remaining roots is not explicitly calculated, but the benefits can be obtained nevertheless. The polynomial with suppressed roots is:

$$r_k(x) = \frac{p_n(x)}{s_k(x)} \tag{2.49}$$

where $s_k(x) = \prod_{i=0}^{k-1}(x - x_i)$ with $s_0 = 1$ and its derivative is $s_k'(x) = s_k(x)\sum_{i=0}^{k-1} 1/(x - x_i)$. Eq. (2.48) is then modified to:

$$x_k^{m+1} = x_k^m - \frac{r_k(x_k^m)}{r_k'(x_k^m)} = x_k^m - \frac{p_n}{p_n'\left[1 - \frac{p_n s_k'}{p_n' s_k}\right]} \tag{2.50}$$

Fig. 2.4 illustrates this method for finding the roots of the monic Jacobi polynomial, $p_5^{(1,1)}$. The figure depicts an iteration seeking the 3rd root after the first two have been found. The curves are the full polynomial and the one with the roots suppressed, i.e. $r_2(x)$ in Eq. (2.49). The straight line is the Newton-Raphson linear estimate for the next root. This method has two advantages. First, the iterations should converge faster, because as the iterations proceed, the degree of the polynomial is continually reduced until the last root is on a straight line, requiring no iteration. Second, the radius of convergence is larger and more predictable as shown by the heavy line in Fig. 2.4, so convergence can be assured. Any point to the left of the next root is within the radius of convergence. The method will converge without fail by starting near the left boundary and by always selecting an initial guess which is less than the value of the next root, i.e. $x_k^0 < x_k$. Alternatively, we could start at the right boundary and work back to the left.

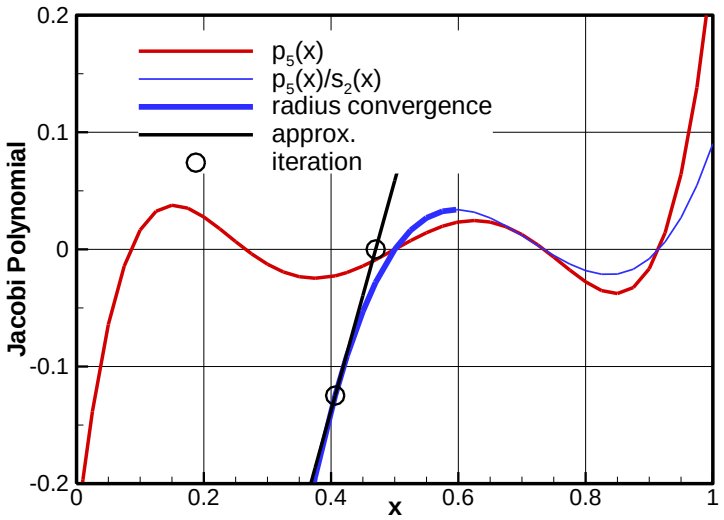


Fig. 2.4 5th Jacobi(1,1) polynomial root finding, first two roots suppressed

2.3.3 Root Estimation Methods

A simple way to achieve greater efficiency is to use better initial estimates. Using crude initial estimates, the code of Villadsen and Michelsen (1978) typically requires 7 to 9 iterations. Others have reported a requirement of about 6 iterations [Shen, *et al.* (2011)]. Eq. (2.46) applies strictly for Chebyshev polynomials, but it can be used to approximate the roots of a general Jacobi polynomials, where α and β are viewed as interpolating parameters. Use of Eq. (2.46) cuts the number of iterations required to about 3, a saving of more than a factor of 2.

There are other benefits to accurate initial estimates. An accurate initial estimate assures the iterations will converge to the root of interest. Also, higher order iterative methods can be used without fear of divergence. The idea of root suppression is a good one, but with accurate initial estimates it is not needed. When the estimate is close to convergence, the bracketed term in the denominator of Eq. (2.50) goes to unity, so root suppression has no effect. With accurate initial estimates the efficiency is improved because fewer iterations are needed, convergence is assured, root suppression is not necessary and higher order methods are effective.

There are much better approximations than Eq. (2.46) for estimating the roots of Jacobi polynomials. For a review of available root estimation methods, see Gautschi and Giordano (2008) and Hale and Townsend (2013). As mentioned above and shown in Figs. 2.1 and 2.2, Jacobi polynomials tend to behave like Bessel functions near the boundaries but are more trig-like in the interior. Due to this behavior, the methods for estimating roots tend to fall in two categories: those more accurate near the boundaries and those more accurate in the interior. We will refer to them as either *interior* methods or *boundary* methods. The best interior method, valid for general Jacobi polynomials, is due to Gatteschi and Pittaluga (1985):

$$x_k = \cos(\bar{\theta}_k + \delta\bar{\theta}_k) + O(n^{-4}) \tag{2.51}$$

where

$$\bar{\theta}_k = \pi\left(2k + \alpha - \tfrac{1}{2}\right)/\sigma_n$$

$$\delta\bar{\theta}_k = \left[\left(\tfrac{1}{4} - \alpha^2\right)\cot\left(\frac{\bar{\theta}_k}{2}\right) - \left(\tfrac{1}{4} - \beta^2\right)\tan\left(\frac{\bar{\theta}_k}{2}\right)\right]/\sigma_n^2$$

$$\sigma_n = 2n + \alpha + \beta + 1$$

and the roots are numbered from largest to smallest or right to left rather than left to right. We note that this equation is simply Eq. (2.46) with the correction, $\delta\bar{\theta}_k$, added. It is exact for all combinations of Chebyshev points, i.e. α = ±½ and β = ±½ in Eq. (2.46), since the correction is zero. The estimated roots are usually accurate to at least four or five digits even for relatively small values of n. The accuracy is better for $\alpha = \beta = 1$ (Lobatto) than for $\alpha = \beta = 0$ (Gauss-Legendre), even though the Lobatto case is outside the stated range of applicability, which is $|\alpha| \le ½$ and $|\beta| \le ½$.

An interior method which is better for the roots of Legendre polynomials or Gauss points is due to Tricomi (1950):

$$x_k = \left[1 - \frac{n-1}{8n^3} - \frac{1}{384n^4}\left(39 - \frac{28}{\sin^2(\bar{\theta}_k)}\right)\right]\cos(\bar{\theta}_k) + O(n^{-5}) \tag{2.52}$$

Where $\bar{\theta}_k$ is the same as above. However, this method does not generalize to other Jacobi polynomials. Both of these interior methods have poor accuracy near the boundaries.

A more accurate approximation for roots near the boundary, x = 1, is also due to Gatteschi (1985). The approximation has a form like Eq. (2.51):

$$x_k = \cos\left(\tilde{\theta}_k + \delta\tilde{\theta}_k\right) + \tilde{\theta}_k^5\, O(n^{-2}) \tag{2.53}$$

where:

$$\tilde{\theta}_k = \frac{2J_{\alpha,k}}{\nu_n}$$

$$\delta\tilde{\theta}_k = -\tilde{\theta}_k\left[\frac{4-\alpha^2-15\beta^2}{45\nu_n^4}\left(\frac{J_{\alpha,k}^2}{2}+\alpha^2-1\right)\right]$$

$$\nu_n = \sqrt{\sigma^2+(1-\alpha^2-3\beta^2)/3}$$

Another boundary method is due to Olver (1974):

$$x_k = \cos\left(\hat{\theta}_k + \delta\hat{\theta}_k\right) + \hat{\theta}_k^2 O(n^{-3}) \tag{2.54}$$

where:

$$\hat{\theta}_k = 2\frac{J_{\alpha,k}}{\sigma_n}$$

$$\delta\hat{\theta}_k = \frac{1}{\sigma_n^2}\left[2\left(\alpha^2-\tfrac{1}{4}\right)\frac{1-\hat{\theta}_k\cot(\hat{\theta}_k)}{\hat{\theta}_k} - (\alpha^2-\beta^2)\tan(\hat{\theta}_k/2)\right]$$

and σ_n is defined with Eq. (2.51). Both of these boundary methods rely on the roots of the Bessel function of kind α, where $J_{\alpha,k}$ is the k^{th} root. The error terms indicate the accuracy deteriorates rapidly as the roots move away from the boundary. Due to the symmetry shown in Eq. (2.10), the roots near x = -1 can be estimated using Eqs. (2.53) and (2.54) with α and β reversed. Bogaert (2014) has improved on Eq. (2.54) for the special case of Legendre polynomials. His method is accurate enough to give the roots to 16 digits for n > 25.

The Bessel roots can be stored for small ones while larger ones can be accurately approximated using the first few terms of McMahon's (1894) (see Olver, et al. (2018), 10.21) expansion:

$$J_{\alpha,k} = \lambda - \frac{\mu-1}{8\lambda} - \frac{4(\mu-1)(7\mu-31)}{3(8\lambda)^3} - \ldots \tag{2.55}$$

where: $\mu = 4\alpha^2$, $\lambda = \left(k+\frac{1}{2}\alpha-\frac{1}{4}\right)\pi$. For α = 0 or 1, truncating the expansion after three terms shown gives roots that are accurate to about 1x10^{-6} for k > 3 and to 2x10^{-9} for k > 12. Using 5 terms the approximation is accurate to 1x10^{-16} for k > 16. We have implemented these methods by storing the first 16 roots and calculating others using a 5 term expansion. If less accurate roots are used, the error will flatten out and become constant at large n.

Estimation methods, Eqs. (2.51), (2.53) and (2.54), produce identical estimates when either the shortcut or full polynomial roots are estimated. We will briefly outline the proofs. We need to compare the roots for the following two cases:

- even case: compare n with β = -½ vs. $2n$ and $\beta = \alpha$
- odd case: compare n with β = +½ vs. $2n$+1 and $\beta = \alpha$

If the estimates for the shortcut cases are $\boldsymbol{\xi}$ and those for the full method are $\boldsymbol{x}$, we ask - is $\boldsymbol{\xi} = 2\boldsymbol{x}^2 - 1$? Given the trigonometric identity $\cos(2\theta) = 2\cos^2(\theta) - 1$, we need only show the angles for the shortcut procedure are twice those with the full polynomial.

For $\bar{\theta}_k$ in Eq. (2.51), the numerators are the same and σ_n is twice as large for the full method, so the angle values have the correct relationship. By using the double angle identity for tangents it can be proven that the expression for $\delta\bar{\theta}_k$ also has the correct relationship. For Eq. (2.53) it is first apparent that like σ, the expressions for ν are twice as large for the full polynomials and the rest of the proof is simple algebraic manipulation. The proof for Eq. (2.54) again requires use of the double angle formula for tangents.

We have performed calculations with the four error estimation methods to evaluate their utility for initiating an iterative solution. Ideally, we would like to know the range of n and x when each method is best for the cases in Table 2.1. The error terms included with the root estimation methods, Eqs. (2.51) to (2.54), give some indication of their accuracy. All the estimation methods give a dependence of the error on n. While the error terms for the interior methods do not provide a dependence on x, the boundary methods include a dependence on θ which is easily translated to x. Unfortunately, we have also found some of the error terms are incorrect, since they do not always agree with calculations.

For the three cases of interest in Table 2.1, Figs. 2.5, 2.6 and 2.7 illustrate calculations for n = 14. By picking the best estimates, the maximum errors are about 10^{-7} for all three cases. In each case a different one of the four estimation methods is best. It is obvious from these graphs that the accuracy of the estimates varies considerably with x. As expected, the interior methods are more accurate in the interior and the boundary methods are best near the boundaries. Table 2.3 lists parameters for the following equation which approximates errors in the estimates:

$$\epsilon = \epsilon_{10}\left(\frac{n}{10}\right)^a\left(\frac{\theta}{\pi/4}\right)^b \tag{2.56}$$

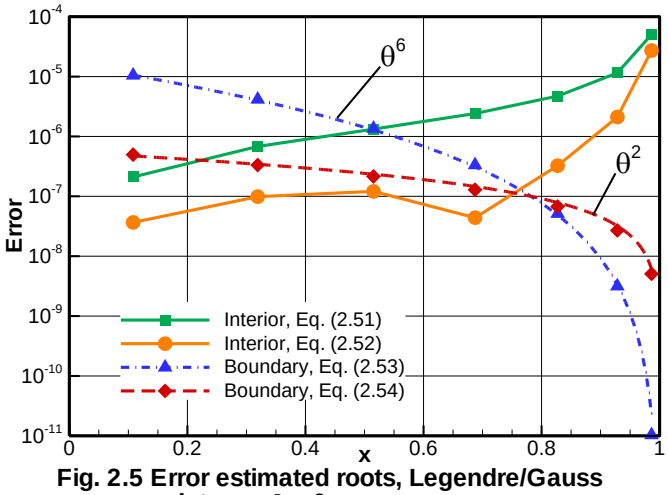


Fig. 2.5 Error estimated roots, Legendre/Gauss points $\alpha = \beta = 0$

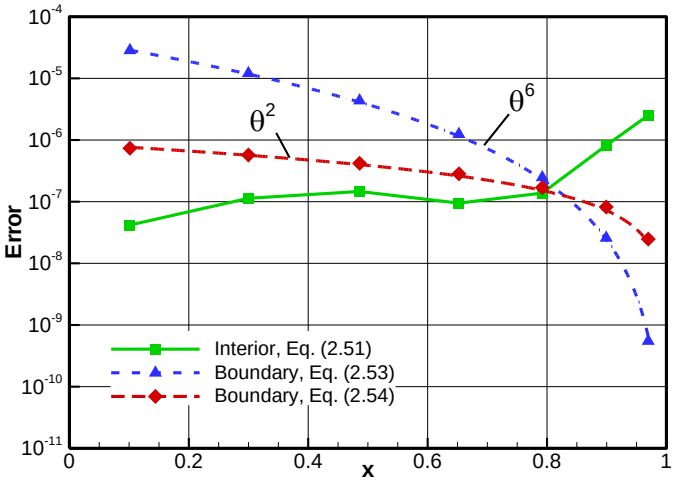


Fig. 2.6 Error estimated roots, Lobatto points $\alpha = \beta = 1$

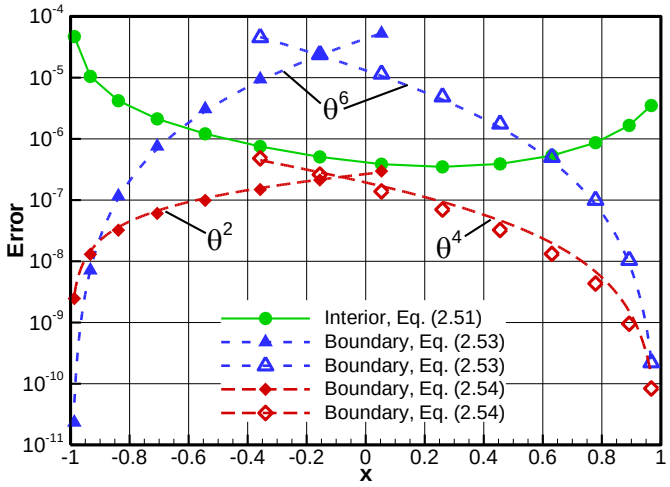


Fig. 2.7 Error estimated roots, Radau points, α = 1, β = 0

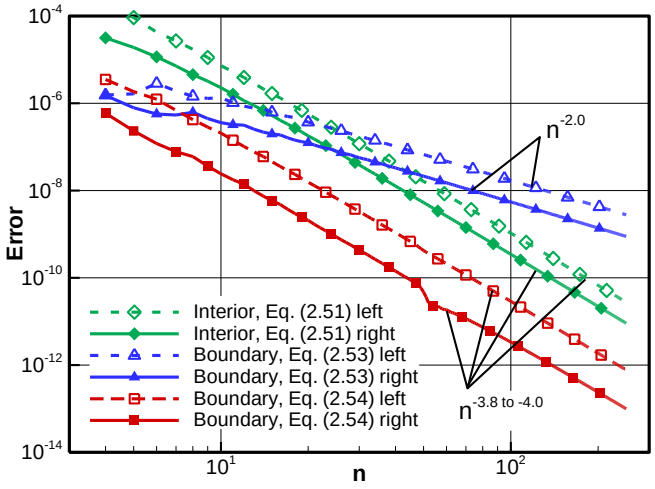


Fig. 2.8 Estimated root errors at x = 0.707, Radau α = 1, β = 0

Table 2.3 Error of Estimated Roots, Eq. (2.56)

	α	β	ϵ_{10}	a	b
Eq. (2.52)	0	0	9.47E-07	-3.87	0
Eq. (2.51)	0	0	1.11E-05	-3.98	0
"	1	1	6.52E-07	-3.95	0
"	1	0	3.17E-06	-3.96	0
"	0	1	9.70E-06	-3.97	0
Eq. (2.53)	0	0	5.70E-07	-1.99	6
"	1	1	1.61E-06	-1.97	6
"	1	0	5.22E-07	-1.98	6
"	0	1	1.63E-06	-1.98	6
Eq. (2.54)	0	0	5.27E-07	-4.00	2
"	1	1	1.12E-06	-3.94	2
"	1	0	1.52E-08	-3.69	2
"	0	1	2.44E-07	-3.93	4

where, of course, $\theta = arccos(x)$. The values above are normalized so that ϵ_{10} approximates the error at x = 0.707 for n = 10. The methods, Eqs. (2.51) to (2.54), are claimed to have exponent a of -2 to -5, while exponent b is indicated to be 5 for Eq. (2.53) and 2 for Eq. (2.54). We will use an exponent of 0 in other cases to indicate no correlation. To better establish the errors, the correlations were compared to the exact roots for n to 250. The errors were interpolated at $\theta = \pi/3, \pi/4$ and $\pi/6$ or $x = 0.500, 0.707$ and 0.866. The errors are listed for only positive x. Values for negative x are given by reversing α and β (see Eq. (2.10)). The b exponents were determined by curve fits like those shown in Figs. 2.5, 2.6 and 2.7. Values for parameter a were determined by fitting the values from n = 25 – 250 as shown in Fig. 2.8, where "left" designates errors at x = -0.707. The a exponent is normally insensitive to x, but some methods had a "sweet spot" with exceptional accuracy at some locations. For example, Eq. (2.51) predicts the Lobatto points near 0.707 with greater accuracy (see Fig. 2.9). The error interpolated at 0.707 produces an exceptionally large exponent of -5, so more typical values at 0.50 are listed in Table 2.3.

In several instances, the values in Table 2.3 disagree with the error estimates listed in Eqs. (2.51) to (2.54). The a exponent for Eq. (2.52) is clearly closer to -4 than -5 and that for Eq. (2.54) is closer to -4 than -3. All but Eq. (2.53) appear to have an a exponent of approximately -4, so their relative accuracy is essentially independent of n. There are also some discrepancies in the b exponent. For Eq. (2.53) the appropriate exponent is clearly 6 not 5. Eq.

(2.54) has the curious behavior that the exponent is 4 for the Radau case, $\alpha = 1$, $\beta = 0$, and 2 in other cases, see Figs. 2.5, 2.6 and 2.7.

From these calculations, we have devised some simple rules to pick the best correlation for accurate initial estimates. With the solution method described in Section 2.3.4, we are primarily interested in selecting estimates to give the smallest maximum error, since the maximum error will control the number of iterations required. Eq. (2.46) is used to give approximate values for the roots. The correlation chosen is based on the value of these rough estimates relative to a simple cutoff value, x^* as follows:

Gauss-Legendre – $x^* = 0.5 + 0.019n$ for $n < 19$ and $x^* = \max(0.48, 0.57 - 0.001n)$ for $n \geq 19$. use Eq. (2.52) for $x_k < x^*$, otherwise use Eq. (2.53) for $n < 19$ and Eq. (2.54) for larger n.

Lobatto-Jacobi(1,1) – $x^* = \max(0.78, 0.813 - 0.00075n)$, use Eq. (2.51) for $x_k < x^*$ otherwise use Eq. (2.53) if $n < 20$ and Eq. (2.54) for larger n.

Radau-Jacobi(1,0) – $x^* = 0.1(\beta - \alpha)$, use only Eq. (2.54), use right end values for $x_k > x^*$, otherwise use left end values.

These rules are compatible with the observations from Figs. 2.5 to 2.8 and Table 2.3. For all points, the best boundary method is Eq. (2.53) for small n, but due to the higher convergence rate, Eq. (2.54) becomes better for large n. Eq. (2.53) is always better for the points closest to the boundary, but we are more interested in reducing the maximum error. For the Gauss-Legendre case, Eq. (2.52) is always the best interior method, while for $n = 18$ to 30, there is little difference between the boundary and interior methods for x from 0.55 to 0.85. The cutoff values for large n approximates the 0.50 value recommended by Hale and Townsend (2013). For the Lobatto case, the cutoff is almost constant because the boundary and interior methods have almost identical convergence rates. For Radau points, the boundary method, Eq. (2.54), is always better than the interior method, but the cutoff governs from which boundary the estimates are chosen, left or right. This boundary method is one of the best methods for general Jacobi polynomials, i.e. excluding Eq. (2.52) which is not general. Even so, it beats Eq. (2.52) for half the domain at larger n and for all or most of the domain it beats the general interior method, Eq. (2.51), for Legendre and Radau points. Bogaert (2014) has improved on it for the Gauss-Legendre case.

Overall, the simple rules above do an excellent job of picking accurate estimates. Unfortunately, they are specialized for the three cases of interest here, see Table 2.1. Although some of the results in Table 2.3 indicate a correlation, e.g. for Eq. (2.53) the results correlate with β. There is no obvious generalized correlation, but one would not be far wrong by using Eq. (2.54) for all but the points nearest zero. Perhaps the information in Table 2.3 will be of benefit for developing a generalized estimation method.

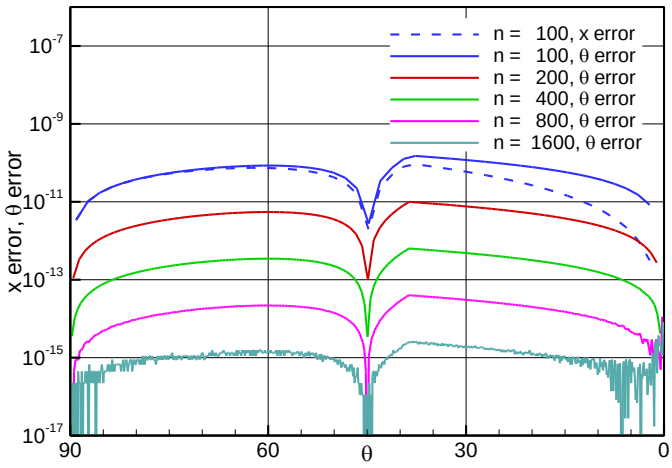


Fig. 2.9 Estimated error in Lobatto roots for large n

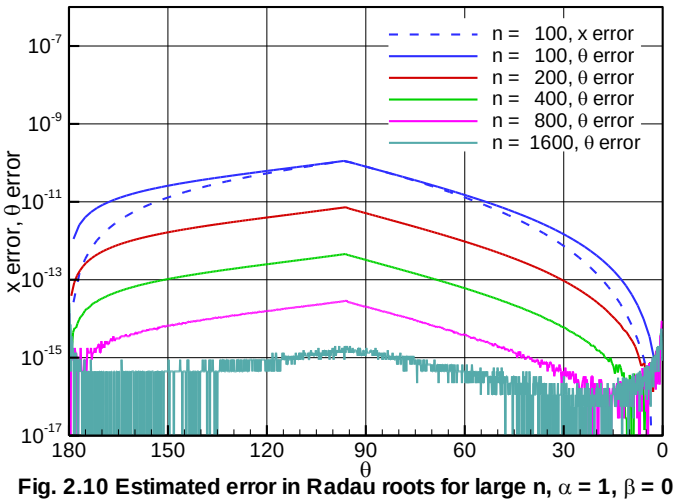


Fig. 2.10 Estimated error in Radau roots for large n, α = 1, β = 0

Figs. 2.9 and 2.10 show the behavior of the estimation methods for Radau and Lobatto points and large n. Earlier we mentioned that Eq. (2.51) has a sweet spot at 45 degrees or x = 0.707. It is clearly visible in Fig. 2.9. Also, the results are consistent with the 4th order rate of convergence in Table 2.3.

Fig. 2.11 shows calculated results using the root estimation methods to initiate a Newton-Raphson iteration. The decline in the maximum initial error is consistent with Table 2.3. For n < 30 the error is driven to roundoff conditions after two iterations, while for larger n only one iteration is required. With accurate initial estimates the efficiency of root determination is dramatically improved relative to simpler initial estimates.

We also note that Fig. 2.11 shows that the maximum error in the Radau roots is constant, while those for Gauss and Lobatto points show a modest linear increase with n. To gain a better understanding, Fig. 2.12 shows the distribution of the error. Since the roots tend to cluster about the endpoints, they are plotted versus $\theta = arccos(x)$ to more clearly show the roots near the boundary. The figure shows the errors are similar except near x = 0. The increasing error is due to use of the shortcut procedure for the Gauss and Lobatto points. The shortcut procedure requires conversion of the roots by $\boldsymbol{x} = \sqrt{(1+\boldsymbol{\xi})/2}$. The roots ξ show no

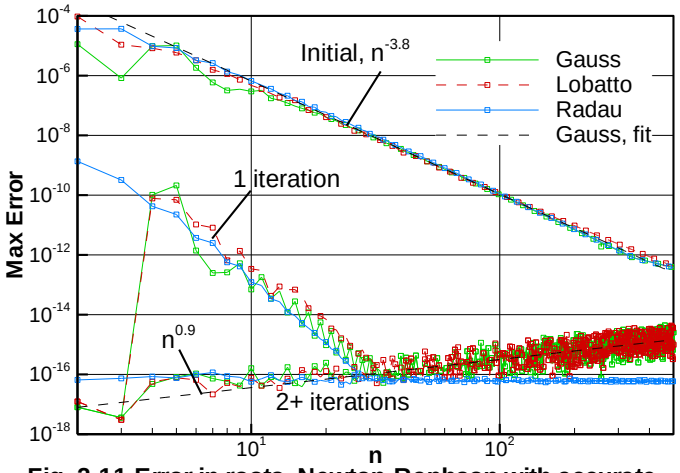


Fig. 2.11 Error in roots, Newton-Raphson with accurate initial estimates

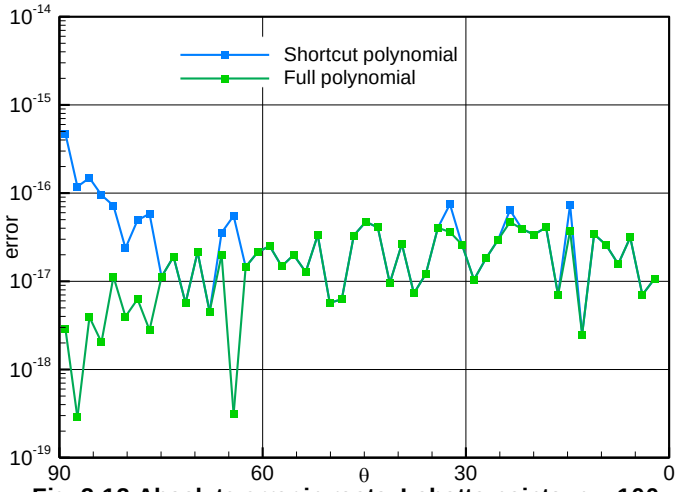


Fig. 2.12 Absolute error in roots, Lobatto points, n = 100

growth in the error, like the Radau roots. However, due to the conversion, the error terms are related by $\boldsymbol{\epsilon_x = \epsilon_\xi / 4x}$. The error in x is greater for roots near zero, $x < ¼$ or $\theta > 75^o$. This area is relatively less important. The usual difficulty is near $x = 1$, where the roots are closely space.

2.3.4 Vectorized Higher Order Iterative Method

As mentioned above, the root suppression algorithm is of no benefit when accurate initial estimates are used. In addition to the extra complication, another disadvantage of the algorithm is the roots are found one at a time. Newer computer hardware has again made vectorization of code important. Writing code for vectorization is especially important for interpreted languages like Matlab or Python even when vector hardware is not available. With an interpreted language, the vector calculations can be carried out with library calls rather than slow looping operations. For these reasons, we have elected not to use root suppression and instead solve for all roots together. The code should vectorize with appropriate hardware and compilers.

The text box below shows vector code for a Newton-Raphson iteration. This code is in Matlab, but similar code can be written in other languages which support array operations. The code relies on three external functions. Like the eigenvalue code, it relies on the x_jacobi function which is an extended version of Gautschi's (2005) r_jacobi function. It returns the coefficients in the conventional form, Eq. (2.16). The $Jacobi\_Deriv$ function returns the three coefficients of Eq. (2.35) and $RootEstimate$ implements the method described in the previous section for estimating roots. The iteration loop requires three statements, one to get the change in the value according to Eq. (2.48), one to update the values, and a test for convergence. Each statement in the iteration loop, including those in $Pcalc$, operates on all roots at once. If the

Matlab Function for Jacobi Roots

```
function [x] = Jacobi_Xroot(n,a,b)
   MaxNR = 4;   xtol = 1000.0*eps;
   ab = x_jacobi(n,a,b);          % recursion coefficients
   c = Jacobi_Deriv(n,a,b);       % derivative relationship
   x(:,1) = RootEstimate(n,a,b); % root estimates
   for i=1:MaxNR
      dp = Pcalc(x,n,ab,c);
      x = x .- dp;
      if(max(abs(dp)) < xtol)break end
   end
end
function dp = Pcalc(x,n,ab,c)  % calculate p, p', return p/p'
   nx = size(x);   p(1:nx,1:n+1) = 0.0;
   p(:,1)  = 1.0;   p(:,2)  = ab(1,3)*x(:,1) - ab(1,1);
   for k = 2:n % calculate p
      p(:,k+1) = (ab(k,3)*x(:,1) - ab(k,1)).*p(:,k) - ab(k,2)*p(:,k-1);
   end
   dp = ((c(1)*x+c(2)).*p(:,n+1) + c(3)*p(:,n));
   dp = p(:,n+1).*(1.0-x.*x)./dp;   % p/p'
end
```

computer hardware has vector processing capabilities and the compiler can generate code for it, substantial speedups are possible.

For an ultraspherical polynomial, only $n/2$ roots need be determined. Their symmetry can be exploited in three different ways. First, the code can be used as is, where n, nx, a and b represent n, $n/2$, α and β, respectively. Alternatively, the code can be used as written by passing $n/2$ for n and ±½ (+ for odd n, - for even) for parameter β or b. The roots ξ are determined, so $x = \pm\sqrt{(1+\xi)/2}$. Thirdly, the code for $Pcalc$ could be modified to calculate the polynomial and its derivative with Eqs. (2.18), (2.21), (2.30) and (2.31). This is a relatively minor modification, since the shortcut polynomials obey the same type of recurrence relationship. In all of these cases $RootEstimate$ must be organized to provide roots in the required form. The first alternative saves roughly half the calculations, since only half as many roots are solved for. The second and third alternatives reduce the calculations by a factor of 4 since there are not only half as many roots, but also the loop in $Pcalc$ is half as long.

There are two problems with the convergence test used in this code. First, near convergence the change in the value is a good estimate of the error at the beginning of the iteration, so convergence is not detected until after it has occurred. This approach usually requires an extra unnecessary iteration. The second problem with the convergence tolerance is that it is based on the maximum error over all the points. If all points have converged except one, then an additional iteration is performed for all the points to reduce the error for the one.

The derivatives of the orthogonal polynomials are needed to calculate the barycentric weights, in Eq. (2.4), quadrature weights, and differentiation matrices (discussed in Sections 2.4 and 2.5). If the derivatives are captured for these purposes, the extra iteration is not wasted. Also, a larger tolerance can be used to reduce unnecessary iterations. The tolerance in the code above is three orders of magnitude greater than the machine epsilon. It could be much larger still without affecting the roots calculated.

The previous section describes how to determine initial estimates which minimize the maximum error to prevent or minimize wasted iterations due to nonuniform convergence. Also, given the results shown in Figs. 2.11, the code in the box above can be altered by removing the test for convergence and by setting $MaxNR = 2$ for $n < 30$ and $MaxNR = 1$ for larger n. This is the ultimate solution to the deficiencies of the convergence test shown in the code above. In this case, there is no need to save dp, so the iteration loop can be reduced to a single statement. These changes give a dramatic improvement in efficiency relative to simpler initial estimates requiring 6 to 9 iterations, one being wasted due to the convergence test.

Fig. 2.11 shows that with accurate initial estimates only one or two Newton-Raphson iterations are required to reach round off conditions. Perhaps even greater efficiency can be achieved with higher order iterative methods. Lether (1978) and Yakimiw (1996) considered several techniques for creating higher order iterative functions. One method expands the solution about a root using Taylor series and then uses reversion of the series [Abramowitz and Stegun (1972), p. 16] to produce a higher order extension to Eq. (2.48):

$$x - x_0 = \delta(x_0) = -\sum_{j=1}^{M} \frac{d_j}{j!} q(x_0)^j \tag{2.57}$$

where x_0 and x are the initial and improved root estimates, respectively, and $q(x_0) = P(x_0)/P'(x_0)$ and the series is carried out to the degree desired. The subscripts and superscripts of P, i.e. α, β and n, have been dropped for convenience. The coefficient $d_1 = 1$, so the Newton-Raphson method results when only one term is used. The expressions for $j > 1$ involve higher derivatives of the polynomial and become increasingly complicated for large j. The higher derivatives are easily calculated from Eq. (2.33).

The procedure for determining the coefficients, $\boldsymbol{d}$, can be simplified. The Sturm-Liouville relationship for the derivatives, Eq. (2.32), are first rewritten:

$$\begin{aligned} P'' &= \frac{\alpha - \beta + (\alpha + \beta + 2)x}{1 - x^2} P' - \frac{n(n + \alpha + \beta + 1)}{1 - x^2} P \\ &= \frac{\acute{a}x + \acute{b}}{1 - x^2} P' - \frac{\acute{c}}{1 - x^2} P \\ &= b_1(x)P' + b_0(x)P \end{aligned} \tag{2.58}$$

Following Traub (1964) and Yakimiw this relationship is used to derive simpler expressions for the coefficients, d. The coefficients depend on x and b_1 and b_0, but not explicitly on the higher derivatives of P. The coefficients are developed from the expansion by forcing $d\delta/dx = -1$ near the root. The differentiation of Eq. (2.57) gives:

$$-\frac{d\delta}{dx} = 1 = \left[d_1' q + \tfrac{1}{2} d_2' q^2 + \tfrac{1}{6} d_3' q^3 + \cdots\right] + \left[d_1 + d_2 q + \tfrac{1}{2} d_3 q^2 + \cdots\right] q' \tag{2.59}$$

Now, using Eq. (2.58), substitute $q' = 1 - b_1 q - b_0 q^2$ and collect terms with like powers of q:

$$-\frac{d\delta}{dx} = 1 = d_1 + [d_1' + d_2 - d_1 b_1] q + \left[\tfrac{1}{2} d_2' + \tfrac{1}{2} d_3 - d_2 b_1 - d_1 b_0\right] q^2 + \cdots \tag{2.60}$$

Clearly, $d_1 = 1$ and each of the terms in brackets must be zero, so the coefficients can be expressed by the recurrence relationship:

$$d_{j+1} = -d_j' + j d_j b_1 + j(j-1) d_{j-1} b_0 \tag{2.61}$$

It is straight forward to work out the coefficients with this relationship. We have derived them for a general Jacobi polynomial through $M = 5$. They may be found in the supplied code. However, we find that convergence is achieved with one iteration when fewer terms are included. After working through the algebra and simplifying, the expansion can be represented as:

$$\delta(x_0) = -(1 - x^2) \sum_{j=1}^{M} \frac{\hat{d}_j}{j!} \hat{q}(x_0)^j \tag{2.62}$$

where: $\hat{q}(x) = P(x)/[(1-x^2)P'(x)]$. Using the parameters of Eq. (2.32) explicitly, the terms through M = 3 are:

$$\begin{aligned} \hat{d}_1 &= 1 \\ \hat{d}_2 &= \acute{b} + \acute{a}x \\ \hat{d}_3 &= 2(\acute{b}^2 - \acute{c}) - \acute{a} + 2\acute{b}(2\acute{a} - 1)x + (2\acute{a}^2 - \acute{a} + 2\acute{c})x^2 \end{aligned} \tag{2.63}$$

where the parameters are defined in Eq. (2.58). The Newton-Raphson gives a quadratic convergence rate. Each additional term in the expansion increases the rate of convergence by one, so the rate of convergence is then: $\epsilon_1 = \epsilon_0^{(M+1)}$, where ϵ_0 is the error at the beginning of an iteration and M is the number of terms in the expansion

Fig. 2.13 shows error reduction with one iteration, i.e. ϵ_1 vs. ϵ_0. The results are illustrated for the first Lobatto point with n = 14. The first point, closest to the boundary, converges at a slightly slower rate due to larger higher derivatives. The convergence behavior is stable and by fitting the slopes of the lines, the convergence rates are consistent with the expected order. A Newton-Raphson iteration drops the error from 10^{-6} to about 10^{-10} in one iteration, while the 4th order method (including terms through d_3) drops it from 10^{-5} to 10^{-15} and the 6th order iteration from 10^{-4} to 10^{-16}. Fig. 2.6 shows the estimated roots for this case are in error by 10^{-7} or less, so only one iteration is required for all of the methods except for the Newton-Raphson.

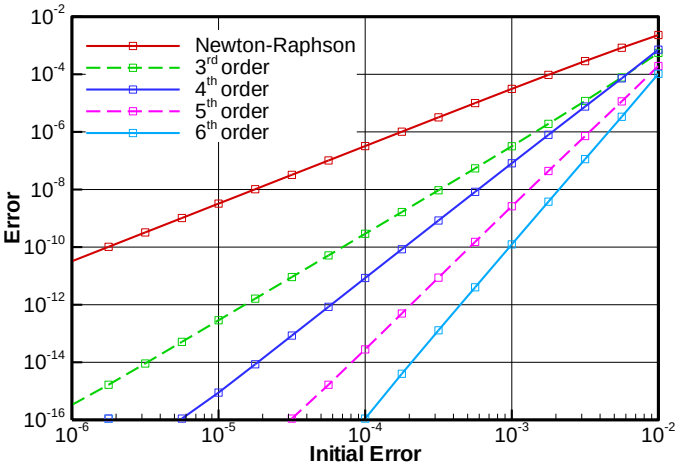


Fig. 2.13 Convergence of higher order iterations, first Lobatto point, n = 14

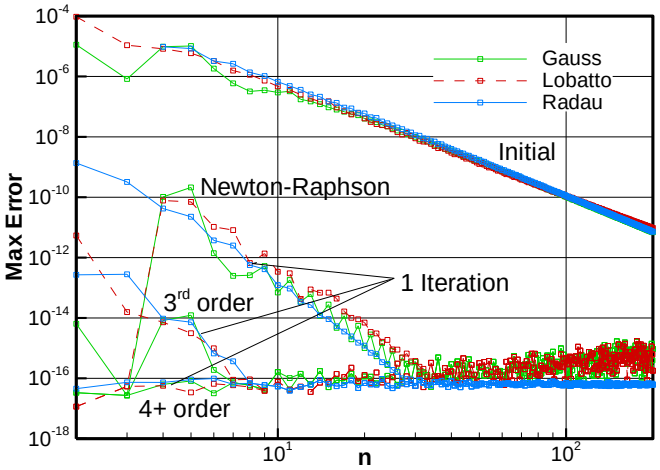


Fig. 2.14 Convergence of root calculation, higher order iterative methods, accurate initial estimates

Fig. 2.14 is like Fig. 2.11 but with the higher order methods added and only one iteration considered. With the accurate initial estimates, the 3rd order method (M = 2) converges in one iteration for n > 7. The higher order methods require only one iteration for all n. The shortcut polynomials were again used for Gauss and Lobatto points. For this reason, the equations are linear for n < 4, so a single Newton-Raphson iteration is exact, while the 3rd order method is not. The 4th and higher order methods are also not exact, but reach roundoff conditions after one iteration for all n.

Before leaving this section, we mention the clever method of Bogaert (2014) which addressed the case of Gauss points. He calls his method *iteration free*, but it is iteration free only for $n \gtrsim$

25. The results for small n are stored rather than calculated. Fig. 2.14 shows that Yakimiw's method is also iteration free when combined with accurate initial estimates. However, instead of using the two steps described here, i.e. root estimation followed by high order iteration using reversion of series, Bogaert combined the two analytically to produce an extremely accurate root estimation method, accurate to 10^{-16} for $n > 25$. He used simple root estimates, $\hat{\theta}_k = 2J_{0,k}/\sigma$ i.e. Eq. (2.54) truncated to the first term. The simplicity of the initial estimates combined with symbolic algebraic software allowed him to combine the two steps into one. In principal, the same technique could be used for Lobatto quadrature by expansion about the estimated roots $\hat{\theta}_k = 2J_{1,k}/\sigma$. Eq. (2.54) is an excellent one to expand upon since it gives good accuracy in both the boundary and interior regions. Unfortunately, Bogaert's method has not yet been extended to Lobatto and Radau quadrature. However, we use some of his work for large n.

In summary, the accurate initial estimates together with higher order iterative methods alleviates the need to iterate at all. As a result, the Jacobi polynomial needs to be evaluated only once for each unique root. For the most common ultraspherical case, polynomial evaluation with the shortcut recurrence relations requires $O(5n^2/4)$ floating point operations. Calculation of the polynomial derivatives and other calculations needed for a single iteration are approximately $5n$, $8n$ and $12n$ for 2nd (Newton-Raphson), 3rd and 4th order methods, respectively. The use of higher order iterations is obviously worthwhile since it avoids a second $O(5n^2/4)$ calculation of the polynomial. Higher order iterations are not needed for $n > 30$, and a third order method is adequate for $n = 8$ to 30.

2.4 Quadrature and Barycentric Weights

Here we discuss the calculation of the quadrature weights and barycentric weights. Calculation of the quadrature weights is discussed in many texts, with many different procedures given. In this development, the traditional interval of [-1,1] is used for most of the calculations. Conversion of the results to the shifted interval [0,1] is described since it is more convenient for solving problems with MWR.

Here we are concerned with the efficiency and simplicity of the calculations, computability of the numbers and rounding errors. We are primarily interested in small to intermediate values of n, say $n < 200$. However, MWR atmospheric modeling calculations with n of several thousand are mentioned by Yakimiw (1996). The quadrature weights are usually calculated from the eigenvectors found during the eigenvalue analysis described in Section 2.3.1. Alternatively, they may be calculated directly from the roots with various formulas. We will use the later approach here, together with the highly efficient root finding method discussed in Section 2.3.4.

The weights can be calculated directly from the roots using many formulas involving the polynomials and/or their derivatives evaluated at the roots. In the literature, one is frequently presented with one formula (and often one computer program) for Gaussian quadrature,

another for Radau quadrature, a third for Lobatto quadrature and so forth. Here we are interested not only in the three fundamental cases, Gauss, Lobatto and Radau, but also symmetric problems in planar, cylindrical and spherical geometry. We will develop all from a single formula. We show how the basic formula and the polynomial relationships of Sections 2.1 and 2.2 can produce other formulas found in the literature. The derivatives of the polynomials can be calculated in many ways, e.g. Eqs. (2.29), (2.35) and (2.38). To accurately and explicitly specify the calculation procedure one must state how the polynomial derivatives are determined. Some previous works have not provided this information, making the results difficult to interpret.

Formulas for the weights can be developed from the single formula for the Jacobi-Gauss weights [Krylov (1962, p 113) and Hildebrand (1987, p. 401)]. The general formula for the Jacobi-Gauss weights in Eq. (2.8) for the interval [-1,1] is:

$$W_i^* = \frac{2^{\alpha+\beta+1}\Gamma(n+\alpha+1)\Gamma(n+\beta+1)}{\Gamma(n+\alpha+\beta+1)\,n!\,(1-x_i^2)}\left[P_n^{(\alpha,\beta)\prime}(x_i)\right]^{-2}$$
$$= (2n+\alpha+\beta+1)\zeta_n^{(\alpha,\beta)}\left[\sqrt{1-x_i^2}\,P_n^{(\alpha,\beta)\prime}(x_i)\right]^{-2} \tag{2.64}$$

The asterisk denotes the weights for Eq. (2.8), where the integrand contains the weight function $\omega(x) = (1-x)^\alpha(1+x)^\beta$. Gaussian quadrature is the Jacobi-Gauss quadrature for the special case where $\alpha = \beta = 0$. The weights for Lobatto and Radau quadrature methods and weights for symmetric problems can be calculated from the Jacobi-Gauss weights.

Consider Lobatto quadrature and recall the definitions in Eqs. (2.5) and (2.8). The Jacobi-Gauss weights for $\alpha = \beta = 1$ are given by Eq. (2.64) and also:

$$W_i^* = \int_{-1}^{1} \ell_i^*(x)(1-x)(1+x)\,dx = \int_{-1}^{1} \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{(x-x_j)}{(x_i-x_j)}(1-x^2)dx$$

where ℓ^* interpolates through the interior points only. The Lobatto quadrature weights can be expressed in terms of the interpolant through the interior and end points, so they are given by:

$$W_i = \int_{-1}^{1} \ell_i(x)\,dx = \int_{-1}^{1} \left[\frac{(1-x^2)}{(1-x_i^2)} \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{(x-x_j)}{(x_i-x_j)}\right] dx = \frac{W_i^*}{(1-x_i^2)} \tag{2.65}$$

The boundary weights are easily calculated from $\sum W_i = 2$ or as described below. The weights for other cases can be calculated in a similar way using the appropriate values of α and β from Table 2.1. Throughout this section, the values of α and β from Table 2.1 are implied when the polynomials are shown without the (α,β) superscript designation.

The quadrature weights can also be expressed in terms of the barycentric weights, $W_i^b = 1/\hat{p}_n'(x_i)$ in Eq. (2.1), so they can be determined directly from the roots using the continued

products. Expressing the quadrature weights in terms of the barycentric weights is especially attractive, since the barycentric weights are needed for interpolation and differentiation, discussed in Section 2.5. The simplicity and economy of expressing quadrature weights, derivatives and interpolants all in terms of the barycentric weights is appealing. For this reason, we first consider calculation of the barycentric weights.

If continued products are used in the calculation of the weights, e.g. Eq. (2.1), it can be tedious to calculate $\tilde{\rho}_n$ and other parameters in the proportionality constants in Eq. (2.64). A somewhat simpler alternative is to ignore $\tilde{\rho}_n$ and the other constant terms and determine the proportionality constant from the integral of unity. For example, on [0,1], $\sum W_i = 1/(\gamma+1)$, where $\gamma = 1, 2$ and 3 for planar, cylindrical and spherical coordinates, respectively.

2.4.1 Barycentric Weights

The barycentric weights can be calculated directly using the product relationship in Eq. (2.1) or (2.2) or they can be calculated from the derivatives of the monic polynomials as shown in Eq. (2.4). Simplifications can be made for the ultraspherical cases, $\alpha = \beta$, regardless of the calculation procedure.

The direct use of the weights is problematic for large n. Eq. (2.4) relates the weights to the monic polynomial and its derivatives. Normal 64 bit double precision arithmetic can represent numbers in the range of $2.2\text{x}10^{-308}$ to $1.8\text{x}10^{+308}$. The maximum value of a monic Legendre polynomial is $1/\rho_n$, Eqs. (2.15) and (2.23), which is approximately 10^{-300} at $n = 1000$ for calculations on [-1,1] and at $n = 500$ on [0,1], Eq. (2.17). Furthermore, calculation of quadrature weights requires that these numbers be squared which reduces the maximum n by another factor of two. To avoid this limitation, we will usually work with weights which are normalized by the leading coefficient, ρ_n, for the polynomials under consideration:

$$\hat{W}_i^b = \frac{W_i^b}{\rho_n} = \frac{1}{\rho_n \hat{p}_n'(x_i)} \tag{2.66}$$

Working with normalized weights, designated by the hat (^), causes little difficulty and it insures the values remain in a range which is valid for computations.

For the ultraspherical cases, $\alpha = \beta$, on the interval [-1,1] the roots are symmetrically located about zero, so only half of the weights need to be calculated. For an odd number of roots, the center root is zero. Consider an odd number of interior points, the continued product representation from Eq. (2.1), simplifies as follows:

$$\begin{aligned}\hat{p}_n'(x_i) &= \prod_{\substack{j=0\\ j\neq i}}^{n+1}(x_i - x_j)\\ &= (x_i - x_0)(x_i - x_1)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_n)(x_i - x_{n+1})\\ &= (x_i - x_0)(x_i - x_1)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i + x_1)(x_i + x_0)\\ &= (x_i^2 - x_0^2)(x_i^2 - x_1^2)\ldots(x_i^2 - x_{i-1}^2)(x_i^2 - x_{i+1}^2)\ldots(x_i)(2x_i)\\ &= 2x_i^2\prod_{\substack{j=0\\ j\neq i}}^{n/2}(x_i^2 - x_j^2)\end{aligned} \tag{2.67}$$

for $i = 1,\ldots,n/2$. In our notation $n/2$ is truncated to an integer, i.e. rounded toward zero. The points at the right end can be paired up with those on the left end. The only roots which cannot be paired are the zero root and the one at -x_i (yielding the $2x_i$ term) The derivative for the center zero root is given by:

$$\hat{p}_n'(x_{n/2+1}) = \prod_{j=0}^{n/2}(-x_j^2) \tag{2.68}$$

The derivatives and weights alternate in sign. For an odd polynomial the weights are symmetric about the center weight, so $\hat{p}_n'(x_{n+1-i}) = \hat{p}_n'(x_i)$. From Eq. (2.68) we see the weight at the center point is negative when $n/2$ is even and positive when it is odd. Consequently, the endpoint values are always positive.

For an even number of points there is no center point, so the derivatives are given by:

$$\hat{p}_n'(x_i) = 2x_i\prod_{\substack{j=0\\ j\neq i}}^{n/2}(x_i^2 - x_j^2) \tag{2.69}$$

for $i = 1,\ldots,n/2$. The values are antisymmetric about the center, so $\hat{p}_n'(x_{n+1-i}) = -\hat{p}_n'(x_i)$. The values alternate sign with the first value always negative and the last value always positive. The easiest way to remember the signs for the barycentric weights is to remember the last one is always positive and that they alternate in sign.

If the roots are calculated using the shortcut procedure, then $\boldsymbol{x}^2 = (1 + \boldsymbol{\xi})/2$. If calculated directly before conversion, the expressions above simplify to:

$$\hat{p}_n'(x_i) = \left(\frac{1}{2}\right)^{\frac{n}{2}} (1+\xi_i) \prod_{\substack{j=0 \\ j\neq i}}^{n/2} (\xi_i - \xi_j) \quad \text{and}$$

$$\hat{p}_n'(x_{n/2+1}) = \left(\frac{-1}{2}\right)^{(\frac{n}{2}+1)} \prod_{j=0}^{n/2} (1+\xi_j) \quad \text{for } n \text{ odd, and} \tag{2.70}$$

$$\hat{p}_n'(x_i) = \left(\frac{1}{2}\right)^{(\frac{n}{2}-1)} \sqrt{(1+\xi_i)/2} \prod_{\substack{j=0 \\ j\neq i}}^{n/2} (\xi_i - \xi_j) \quad \text{for } n \text{ even}$$

The alternative method of calculation is based on Eq. (2.4). The normalized barycentric weights on [-1,1] are:

$$\begin{aligned} \widehat{W}_i^b &= [\rho_n \hat{p}_n'(x_i)]^{-1} = -[(1-x_i^2)\, P_n'(x_i)]^{-1} \quad \text{for } i = 1, \ldots, n \\ \widehat{W}_0^b &= [\rho_n \hat{p}_n'(x_0)]^{-1} = -[2P_n(-1)]^{-1} \\ \widehat{W}_{n+1}^b &= [\rho_n \hat{p}_n'(x_{n+1})]^{-1} = [2P_n(1)]^{-1} \end{aligned} \tag{2.71}$$

The endpoint values are easily calculated using the expressions for the endpoints of the polynomials, Eq. (2.11).

The derivatives for the interior weights in Eq. (2.71) can be simplified for the ultraspherical polynomials, $\alpha = \beta$. The shortcut polynomials use the roots of polynomials with the given α and $\beta = \pm\frac{1}{2}$. With the roots ξ_i the positive roots are related by $x_{n-n/2+i} = \sqrt{(1+\xi_i)/2}$. Substituting the derivatives from Eqs. (2.30) and (2.31) into Eq. (2.71) gives the following expressions for the interior barycentric weights for the ultraspherical polynomials:

$$\widehat{W}_j^b = -\left[\sqrt{2}\,\tilde{a}_n \sqrt{(1-\xi_i)(1-\xi_i^2)}\; P_{n/2}^{\left(\alpha,-\frac{1}{2}\right)'}(\xi_i)\right]^{-1} \quad \text{for } n \text{ even}$$

$$\widehat{W}_j^b = -\left[\tilde{a}_n (1-\xi_i^2)\; P_{n/2}^{\left(\alpha,+\frac{1}{2}\right)'}(\xi_i)\right]^{-1} \quad \text{and} \tag{2.72}$$

$$\widehat{W}_{n/2+1}^b = -\left[\tilde{a}_n P_{n/2}^{\left(\alpha,+\frac{1}{2}\right)}(-1)\right]^{-1} \quad \text{for } n \text{ odd}$$

where again $i = 1,\ldots,n/2, j = n - n/2 + i$, with $n/2$ rounded down to an integer. Eqs. (2.18) and (2.21) define the normalizing constants in terms of endpoint values, Eq. (2.11). The values of primary interest are: $\tilde{a}_n = 1$ for the Gauss case, $\alpha = \beta = 0$, while for the Lobatto case, $\alpha = \beta = 1$, $\tilde{a}_n = (n+1)/(n/2+1)$, which is 2 for odd n since $n/2$ is rounded down.

Since these weights correspond to the positive roots x_j, those for the negative roots must account for the alternating signs of the weights discussed above. They are given by:

$$\widehat{W}_i^b = (-1)^{n+1}\, \widehat{W}_{n+1-i}^b \tag{2.73}$$

for i = 1,…,n/2. The weights for odd n are symmetric about the center, while for even n they are antisymmetric. Also, the center weight alternates in sign with n/2 as discussed below Eq. (2.68).

Most texts do not give the analytical expression for the center weight for odd n. After substitution for the endpoint expressions, the weights for the two endpoints and for the center point for odd n is:

$$\begin{aligned}
\widehat{W}_0^b &= -[2P_n(-1)]^{-1} = (-1)^{(n+1)}\frac{n!\,\Gamma(\beta+1)}{2\,\Gamma(n+\beta+1)} \\
\widehat{W}_{n+1}^b &= [2P_n(1)]^{-1} = \frac{n!\,\Gamma(\alpha+1)}{2\,\Gamma(n+\alpha+1)} \\
\widehat{W}_{n/2+1}^b &= -\frac{P_{n/2}^{\left(\alpha,+\frac{1}{2}\right)}(1)}{P_{n/2}^{\left(\alpha,+\frac{1}{2}\right)}(-1)}\left[\frac{1}{P_n^{(\alpha,\alpha)}(1)}\right] = (-1)^{\left(\frac{n}{2}+1\right)}\frac{\sqrt{\pi}\,n!\,\Gamma\left(\frac{n}{2}+\alpha+1\right)}{2\,\Gamma(n+\alpha+1)\Gamma\left(\frac{n}{2}+1\frac{1}{2}\right)}
\end{aligned} \tag{2.74}$$

The first two formulas above apply for all n and the last only for odd n. In our notation n/2 is always rounded down to an integer.

The relationships above are for the weights on the interval [-1,1]. Using Eq. (2.17), they may be converted to the corresponding weights on [0,1] by:

$$\begin{aligned}
\widetilde{W}_i^b &= 2^{(n+1)}W_i^b \\
\widehat{\widetilde{W}}_i^b &= \frac{\widetilde{W}_i^b}{\tilde{\rho}_n} = \frac{2^{(n+1)}W_i^b}{\tilde{\rho}_n} = \frac{2^{(n+1)}\rho^n\widehat{W}_i^b}{\tilde{\rho}_n} = 2\,\widehat{W}_i^b
\end{aligned} \tag{2.75}$$

As before, the tilde (~) designates values on [0,1] and the hat (^) designates normalized values. The normalized weights on [0,1] use both decorations. The relationship between the normalized leading coefficients is given in Eq. (2.17).

For symmetric problems, the barycentric weights appear in Eq. (2.2). When solving differential equations with MWR, the symmetry about x = 0 forms the boundary condition, so no point is needed there. The points are numbered starting with 1, while n is the last interior point as before. The boundary point, x_{n+1} = 1, is used to satisfy a condition at the external boundary. The symmetric cases only make sense on the interval [0,1], because radial polar and spherical coordinates are nonnegative. Nevertheless, the calculations are performed for [-1,1] and then converted.

The points for symmetric problems are the roots of the Jacobi polynomials with the values of α and β in Table 2.1, i.e. β = -½, 0, ½ for planar, cylindrical and spherical coordinates and α = 0 or 1 for Gauss or Lobatto quadrature, respectively. The symmetric cases are closely related to the shortcut cases discussed above. The roots on the interval [-1,1] are converted by $\boldsymbol{x}^2 = (1+\boldsymbol{\xi})/2$. The weights can be calculated with a continued product:

$$\widetilde{W}_i^b = \left[\prod_{\substack{j=1\\ j\neq i}}^{n+1}\left(x_i^2 - x_j^2\right)\right]^{-1} = (2)^n \left[\prod_{\substack{j=1\\ j\neq i}}^{n+1}\left(\xi_i - \xi_j\right)\right]^{-1} = 2^n W_i^b \tag{2.76}$$

for $i = 1, \ldots, n+1$. The weights are normalized with the leading coefficient of the Jacobi polynomial as in Eq. (2.66). Since there is one less point, the normalized weights are the same on [0,1] and [-1,1], i.e. $\widehat{\widetilde{W}}_i^b = \widehat{W}_i^b$, which differs from the nonsymmetric case, Eq. (2.75).

The barycentric weights for symmetric problems bear the following relationships to the Jacobi polynomials:

$$\begin{aligned} &\widehat{W}_i^b = -[(1-\xi_i)P_n'(\xi_i)]^{-1} \text{ for } i = 1,..,n, \text{and} \\ &\widehat{W}_{n+1}^b = 1/P_n(1) \end{aligned} \tag{2.77}$$

2.4.2 Gaussian Quadrature Weights

Gaussian quadrature has been studied more heavily than any of the others. It uses the roots of Legendre polynomials, $\alpha = \beta = 0$. Starting with Eq. (2.64) several different formulas can be developed using the relationships described in Sections 2.1 and 2.2. The following are valid formulas for calculating the Gaussian quadrature weights:

$$\begin{aligned} W_i &= \frac{2}{(1-x_i^2)[P_n'(x_i)]^2} \\ &= \frac{2(1-x_i^2)}{[n\,P_{n-1}(x_i)]^2} \\ &= \frac{2(1-x_i^2)}{[(1-x_i^2)P_n'(x_i) - x_i P_n(x_i)]^2} \\ &= \frac{2(1-x_i^2)}{[n\,P_{n-1}(x_i) - (n+1)x_i P_n(x_i)]^2} \\ &= \frac{2}{\sum_{k=0}^{n-1}(2k+1)\left(P_k(x_i)\right)^2} \\ &= 2(1-x_i^2)\left(\widehat{W}_i^b\right)^2 \end{aligned} \tag{2.78}$$

The first expression is directly from Eq. (2.64) with $\alpha = \beta = 0$. The second results by substitution of Eq. (2.41) with $P_n(x_i) = 0$. The third is due to Yakimiw (1996) and is claimed to produce more accurate results. The fourth is derived from the third expression by substitution of Eq. (2.41) with the $P_n(x_i)$ term retained. The fifth, was used by Lether (1978), who claimed it was more accurate. A general Jacobi-Gauss formula of this type is given by Shen, *et al.* (2011) and others. The last formula, in terms of barycentric weights, is found using Eq. (2.71).

Yakimiw developed the third expression by forcing $dW(x)/dx = 0$ at the root to reduce the sensitivity of the weights to errors in the roots. It can also be derived by linearizing the first expression and using a Newton-Raphson update to correct for small errors in x. Yakimiw apparently calculated first derivatives with Eq. (2.41), if so, the fourth formula was actually

used. By comparing the first to the third expression and the second to the fourth, it is apparent that a small correction term has been added. Yakimiw also set higher derivatives of $W(x)$ to zero to derive expressions which are even less sensitive to errors in the roots. However, he states that only the first correction term is needed if the root calculations are iterated to roundoff conditions. These claims are investigated below in Appendix A.1.

We note the boundary weights are zero, $W_0 = W_{n+1} = 0$. Combining the last expression with Eq. (2.74) gives the following analytical expression for the center weight with an odd number of points:

$$W_{n/2+1} = \frac{\pi}{2}\left[\frac{\left(\frac{n}{2}\right)!}{\Gamma\left(\frac{n}{2}+1\frac{1}{2}\right)}\right]^2 \tag{2.79}$$

Expressions in terms of the shortcut polynomials can be found by substitution of Eq. (2.72) into the last expression of Eq. (2.78):

$$\begin{aligned} W_j &= \frac{1}{2(1-\xi_i^2)[P_{n/2}^{(0,-\frac{1}{2})\prime}(\xi_i)]^2} \text{ for } n \text{ even, and} \\ &= \frac{1}{(1+\xi_i)(1-\xi_i^2)[P_{n/2}^{(0,+\frac{1}{2})\prime}(\xi_i)]^2} \text{ for } n \text{ odd} \end{aligned} \tag{2.80}$$

for $i = 1,\ldots,n/2, j = n - n/2 + i$ where $n/2$ is rounded down to an integer. Since the values are symmetric about zero, the weights are copied, $W_i = W_{n+1-i}$. The weights are divided by 2 to obtain weights for the interval [0,1].

2.4.3 Lobatto Quadrature Weights

Lobatto quadrature includes weights at both endpoints. As shown by Krylov (1962) the polynomial weighting function must be zero at both points. Since the weight function for Jacobi polynomials is $\omega(x) = (1-x)^\alpha(1+x)^\beta$, Jacobi-Gauss quadrature with $\alpha = \beta = 1$ can be adapted. The more common developments of Lobatto quadrature are based on Legendre polynomials, by finding either the roots of $P_n^{(0,0)}(x) - P_{n+2}^{(0,0)}(x)$ or $(1-x^2)P_{n+1}^{(0,0)\prime}(x)$. From Section 2.2, these three approaches are related by Eqs. (2.40) and (2.42), so all will produce the same roots.

We start again with Eq. (2.64), substitute $\alpha = \beta = 1$, and divide the Jacobi-Gauss weight by $(1-x_i^2)$ as in Eq. (2.65), which results in the first expression below. From that starting point, several other representations are developed:

$$
\begin{aligned}
W_i &= \frac{8(n+1)}{(n+2)[(1-x_i^2)P_n^{(1,1)\prime}(x_i)]^2} \\
&= \frac{8(n+1)}{(n+2)[(1-x_i^2)P_n^{(1,1)\prime}(x_i) - 2x_iP_n^{(1,1)}(x_i)]^2} \\
&= \frac{2(n+2)(n+1)}{[(1-x_i^2)P_{n+1}^{(0,0)\prime\prime}(x_i) - 2x_iP_{n+1}^{(0,0)\prime}(x_i)]^2} \\
&= \frac{2}{(n+2)(n+1)[P_{n+1}^{(0,0)}(x_i)]^2} \\
&= \frac{8(n+1)}{(n+2)}\left(\widehat{W}_i^b\right)^2
\end{aligned}
\tag{2.81}
$$

for i = 1,…,n. The second applies a correction to the first found by setting $dW(x)/dx = 0$ at the root. The third formula is given by Yakimiw. It is equivalent to the second when Eq. (2.27) is substituted. The third expression is commonly reported with the correction term removed (the second denominator term). The fourth expression is another commonly reported formula using Legendre polynomials. It is derived from the third by substitution of Eq. (2.39). Yakimiw shows that no correction term is required to produce accuracy equivalent to the second and third formulas. The last results from substituting the relationships for the barycentric weights, Eq. (2.71).

Combining the last expression with Eq. (2.74) gives the following analytical expression for the endpoint weights and the center weight with an odd number of points:

$$
\begin{aligned}
W_0 = W_{n+1} &= \frac{2}{(n+2)(n+1)} \\
W_{n/2+1} &= \frac{2\pi}{(n+2)(n+1)}\left[\frac{\left(\frac{n}{2}+1\right)!}{\Gamma\left(\frac{n}{2}+1\frac{1}{2}\right)}\right]^2
\end{aligned}
\tag{2.82}
$$

where n/2 is rounded down to an integer. Expressions using the shortcut polynomials can be found by substitution of Eq. (2.72) into the last expression of Eq. (2.81). Since the relationship is so simple, the shortcut weights are not reproduced here. The weights are divided by 2 to obtain weights for the interval [0,1].

2.4.4 Radau Quadrature Weights

Radau quadrature is similar to Lobatto quadrature but includes a weight at only one endpoint. We call it Radau-right if the weight is at x = 1 and Radau-left if the weight is a x = -1. The development for the two cases are analogous, so only the Radau-right case is described in any detail.

Krylov (1962) states that the polynomial weight function, $\omega(x)$, must be zero at the point where the quadrature weight is desired. Since the Jacobi weighting function is $\omega(x) = (1-x)^\alpha(1+x)^\beta$, the weight will be zero at the right end when α = 1 and β = 0. Following the

same type of approach as used for Lobatto quadrature, substitute the values for α and β into Eq. (2.64), and divide the Jacobi-Gauss weight by 1- x_i. This approach differs from the normal development of Radau quadrature which uses the zeros of the combination of Legendre polynomials, $P_n^{(0.0)}$- $P_{n+1}^{(0.0)}$. Since the Legendre polynomials are unity at the right end, this combination will be zero there. It can also be shown that [Shen, et al. (2011), p. 75]:

$$(1-x)P_n^{(1,0)} = P_n^{(0,0)} - P_{n+1}^{(0,0)} \tag{2.83}$$

so the two approaches are related. Several formulas for Radau-right quadrature weights are:

$$\begin{aligned} W_i &= \frac{4}{(1-x_i^2)(1-x_i)\left[P_n^{(1,0)\prime}(x_i)\right]^2} \\ &= \frac{1+x_i}{\left[(n+1)\,P_n^{(0,0)}(x_i)\right]^2} \\ &= \frac{1}{(1+x_i)\left[P_n^{(0,0)\prime}(x_i)\right]^2} \\ &= \frac{4(1+x_i)}{\left[(1-x_i^2)P_n^{(1,0)\prime}(x_i) - \frac{1}{2}(1+3x_i)P_n^{(1,0)}(x_i)\right]^2} \\ &= 4(1+x_i)\left(\widehat{W}_i^b\right)^2 \end{aligned} \tag{2.84}$$

The first formula follows directly from Eq. (2.64), while the second and third are commonly given in other texts. The fourth applies a correction to the first to reduce sensitivity to errors in the roots, see Section 2.4.7. The last results from substituting the relationships for the barycentric weights, Eq. (2.71). The endpoint weight at the right end is found by substitution of Eq. (2.74) into the last expression:

$$W_{n+1} = \frac{2}{(n+1)^2} \tag{2.85}$$

Equivalence of the first and third expressions of Eq. (2.84) can be shown by first using Eq. (2.35) with α = 1 and β = 0, then Eq. (2.83) and finally Eq. (2.40) together with $P_n^{(1,0)}(x_i) = P_n^{(0,0)}(x_i) - P_{n+1}^{(0,0)}(x_i) = 0$:

$$\begin{aligned} (1-x_i^2)P_n^{(1,0)\prime}(x_i) &= \frac{2n(n+1)}{2n+1}\,P_{n-1}^{(1,0)}(x_i) \\ &= \frac{2n(n+1)}{(2n+1)(1-x_i)}\left(P_{n-1}^{(0,0)}(x_i) - P_n^{(0,0)}(x_i)\right) \\ &= \frac{2(1-x_i^2)}{(1-x_i)}P_n^{(0,0)\prime}(x_i), \quad \text{or} \\ (1-x_i)P_n^{(1,0)\prime}(x_i) &= 2\,P_n^{(0,0)\prime}(x_i) \end{aligned} \tag{2.86}$$

The last expression clearly shows equivalence of the first and third expressions in Eq. (2.84). Equivalence of the second and third expressions of Eq. (2.84) can be shown using Eq. (2.41) together with the recurrence relation, Eq. (2.25), to give:

$$\begin{aligned}(1-x_i^2)P_n^{(0,0)\prime}(x_i) &= (n+1)\left(x_i\,P_n^{(0,0)}(x_i) - P_{n+1}^{(0,0)}(x_i)\right)\\ &= -(n+1)(1-x_i)\,P_n^{(0,0)}(x_i), \quad \text{or}\\ (1+x_i)P_n^{(1,0)\prime}(x_i) &= -(n+1)\,P_n^{(0,0)}(x_i)\end{aligned} \tag{2.87}$$

The points and weights of the Radau-left quadrature are the mirror image of those for the Radau-right quadrature. The Radau-left quadrature uses the roots of the Jacobi polynomial with α = 0 and β = 1, or the following sum of Legendre polynomials which are equivalent due to the identity:

$$(1+x)P_n^{(0,1)} = P_n^{(0,0)} + P_{n+1}^{(0,0)} \tag{2.88}$$

The resulting expressions for the weights are like Eq. (2.84), but with $1 + x_i$ replaced by $1 - x_i$ and vice versa and with the alternate Jacobi polynomial substituted:

$$\begin{aligned}W_i &= \frac{4}{(1-x_i^2)(1+x_i)\left[P_n^{(0,1)\prime}(x_i)\right]^2}\\ &= \frac{1-x_i}{\left[(n+1)\,P_n^{(0,0)}(x_i)\right]^2}\\ &= \frac{1}{(1-x_i)\left[P_n^{(0,0)\prime}(x_i)\right]^2}\\ &= \frac{4(1-x_i)}{\left[(1-x_i^2)P_n^{(0,1)\prime}(x_i) + \frac{1}{2}(1-3x_i)P_n^{(0,1)}(x_i)\right]^2}\\ &= 4(1-x_i)\left(\widehat{W}_i^b\right)^2\end{aligned} \tag{2.89}$$

The weight at the left endpoint, W_0, is the same as for the Radau right quadrature, Eq. (2.85). The Radau points are easy to calculate but are of limited usefulness for nonsymmetric problems.

2.4.5 Symmetric Quadrature Weights

Eq. (2.6) shows the type of quadrature formulas needed for symmetric problems. The formulas of interest can be developed from the general Jacobi-Gauss formula, Eq. (2.64), with the values of α and β in Table 2.1, i.e. β = -½, 0, +½ for planar, cylindrical and spherical coordinates and α = 0 or 1 for Gaussian or Lobatto quadrature, respectively. When α = 0 the quadrature is more precisely Jacobi-Gauss quadrature for planar and spherical cases since the term ξ^κ in the integrand of Eq. (2.6) is retained. When α = 1 the quadrature for planar and spherical cases is more precisely Jacobi-Gauss-Radau quadrature since a weight is employed at only one endpoint. However, for planar geometry the symmetric weights are identical to the right half of those for normal Gaussian or Lobatto quadrature. As discussed below, the other

geometries bear a simple relationship to nonsymmetric cases. For simplicity, we call it Gaussian quadrature when α = 0 and there is no weight on the boundary and Lobatto quadrature when α = 1 giving a weight on the boundary at x = 1.

For symmetric problems the quadrature formulas of interest are like Eq. (2.6), where the abscissa are values of $\boldsymbol{x}^2$ on the interval [0,1]. If the roots are calculated on [-1,1] in the normal way, the values determined are called $\boldsymbol{\xi}$. The values of $\boldsymbol{x}$ are the square roots of the translated roots, i.e. $\boldsymbol{x} = \sqrt{(1+\boldsymbol{\xi})/2}$.

Starting with the roots of the Jacobi polynomials with α = 0 and β = -½, 0 or +½, the Gaussian quadrature weights are determined from Eq. (2.64). Since we are interested in the weights on [0,1], the constant $2^{\alpha+\beta+1}$ can be dropped, but the factor ½ appearing in Eq. (2.6) is required. The Gaussian quadrature weights are:

$$\begin{aligned} W_i &= \frac{1}{2(1-\xi_i^2)\left[P_n^{(0,\beta)\prime}(\xi_i)\right]^2} \\ &= \frac{1-\xi_i}{2(1+\xi_i)}\left(\widehat{W}_i^b\right)^2 \end{aligned} \tag{2.90}$$

For calculation of the Lobatto weights on the interval [0,1], start with the roots of the Jacobi polynomials with α = 1 and β = -½, 0 or +½. The values are substituted into Eq. (2.64) with the constant $2^{\alpha+\beta+1}$ again replaced with ½ and the result divided by $x_i = (1-\xi_i)/2$ as was done for Lobatto and Radau quadrature. The quadrature weights are:

$$\begin{aligned} W_i &= \frac{n+1}{(n+\beta+1)(1-\xi_i)(1-\xi_i^2)\left[P_n^{(1,\beta)\prime}(\xi_i)\right]^2} \\ &= \frac{n+1}{(n+\beta+1)(1+\xi_i)}\left(\widehat{W}_i^b\right)^2 \end{aligned} \tag{2.91}$$

Table 2.1 shows the relationships between the weights for symmetric and nonsymmetric cases. The weights for planar and spherical geometry are related to the nonsymmetric cases for $2n$ and $2n$+1, respectively. The cylindrical Gauss and Lobatto weights are related to the nonsymmetric Gauss and Radau right cases, respectively, with the same value of n. As one would expect, there is a simple relationship between the weights for the symmetric and the corresponding nonsymmetric problems. For planar and spherical geometry, the relationship between the barycentric weights is found by comparing Eq. (2.77) to (2.72). For cylindrical geometry, the relationship between the barycentric weights is found by comparing Eq. (2.77) to (2.71). The relationship between the Gaussian quadrature weights is found by comparing Eq. (2.90) to Eqs. (2.78) and (2.80). The relationship between the Lobatto quadrature weights for planar and spherical geometry is found by comparing Eq. (2.91) to Eq. (2.81) with the substitution of Eqs. (2.30) and (2.31).

Table 2.4 shows the ratio of the weights for symmetric and corresponding nonsymmetric problems. Since planar and spherical cases are directly related to shortcut formulas, the corresponding weights are to the right half of the full set. For the cylindrical cases, all weights are related. Keep in mind that the formulas for symmetric weights are on the interval [0,1], while for nonsymmetric cases, the formulas are for the interval [-1,1]. By integrating unity, the nonsymmetric quadrature weights must sum to 2, while the symmetric weights must sum to $1/(\gamma+1)$, where $\gamma = 0,1,2$ for planar, cylindrical and spherical geometry. Table 2.4 shows the planar quadrature weights are identical to the right half of nonsymmetric weights. For cylindrical geometry, the quadrature weights are ¼ of a full set of nonsymmetric weights. For other cases the relationship depends on the values of the roots. All the symmetric weights bear a simple relationship to nonsymmetric weights.

Table 2.4
Ratio Symmetric to Nonsymmetric Weights

	Relative to	α, β	Quadrature	Barycentric
Planar Gauss	Gauss, $2n$	0, -½	1	$2x_i$
Planar Lobatto	Lobatto, $2n$	1, -½	1	$2\left[\frac{2n+1}{n+1}\right]x_i$
Cylindrical Gauss	Gauss, n	0, 0	$1/4$	$2x_i^2$
Cylindrical Lobatto	Radau Right, n	1, 0	$1/4$	$2x_i^2$
Spherical Gauss	Gauss, $2n$+1	0, +½	x_i^2	$2x_i^2$
Spherical Lobatto	Lobatto, $2n$+1	1, +½	x_i^2	$4x_i^2$

2.4.6 Clenshaw-Curtis Quadrature Weights

The quadrature associated with the Chebyshev points is due to Clenshaw and Curtis (1960). As discussed above this quadrature differs from Chebyshev quadrature, because the integration weighting function is unity rather than $1/\sqrt{1-x^2}$. Without the weighting function these roots give a quadrature which is exact for polynomials of degree n + 1. An example in Section 2.12 compares the the accuracy of the quadrature formulas. Clenshaw-Curtis is an interpolatory quadrature, so Eqs. (2.5) and (2.6) apply. For Cartesian coordinates, formulas for the weights are readily available [Funaro (1992), Trefethen (2000)]. They can be calculated by either direct integration or by using a fast Fourier transform.

Although the fast Fourier transform is more efficient for large n, the direct method is sufficient for our purposes. The formulas are slightly different for odd and even n. The formula for even n is:

$$W_0 = W_{n+1} = \frac{1}{(n+1)^2}$$
$$W_i = \left[1 - \sum_{k=1}^{n/2} \frac{2\cos(2k\theta_i)}{4k^2-1}\right]\frac{2}{n+1} \qquad (2.92)$$

For odd n the formula is:

$$W_0 = W_{n+1} = \frac{1}{(n+1)^2 - 1}$$

$$W_i = \left[1 - (-1)^i W_0 - \sum_{k=1}^{n/2} \frac{2\cos(2k\theta_i)}{4k^2 - 1} \right] \frac{2}{n+1}$$

Where $\theta_i = i\,\pi/(n+1)$ are the roots of the Chebyshev polynomials of the second kind, Eq. (2.46). The roots and weights are symmetric, so only half the weights need to be calculated.

For symmetric planar geometry, it is obvious to use the roots of the even polynomials on (-1,1), i.e. the right half of those used for nonsymmetric problems. Since the use of Chebyshev polynomials is not tied to the accuracy of integration, it is not obvious how to use them for symmetric problems in cylindrical and spherical geometry. It appears that the points are normally not altered for cylindrical and spherical geometry, so we simply use the same points as for planar geometry. The points are not shifted toward the boundary to improve the accuracy of the quadrature as they are for Gauss and Lobatto points, see Figs. 1.5, 1.6, 2.32, and 2.33. We calculate the quadrature weights for cylindrical and spherical geometry by the direct integration of Eq. (2.6).

2.4.7 Higher Order Weight Calculations

Many articles discuss the merits of the various formulas in Section 2.4 for calculating the weights [e.g. Lether (1978), Yakimiw (1996), Swarztrauber (2003)]. Most references consider only quadrature weights and often only Gaussian quadrature. For the weights of interest, there are some fundamental differences between the calculations and resultant errors with the various weight formulas.

Yakimiw (1996) claims errors are mostly due to the sensitivity of the weights to the values of the roots. He reduces the sensitivity by setting $dW(x)/dx = 0$ at the root, where the weight expressions are treated as continuous functions. He also expands the weight function in a series and sets higher derivatives of W to zero, which further *flattens* $W(x)$ near the root, reducing the sensitivity further. When the root calculations are iterated to roundoff, he states that zeroing higher derivatives is not needed to achieve the slowest rate of error growth; a first approximation is adequate. Additional analysis of formula accuracy and efficiency is given in Appendix A.1.

Although there are several representations for the weights, all can be expressed in terms of the derivative of the polynomial for which the roots are determined. This form is convenient since the derivatives are also used in a Newton-Raphson or higher order iterative root calculation. Furthermore, the calculations in Appendix A.1 show these formulas are as accurate as any other. The interior barycentric weights can all be expressed by:

$$\widehat{W}_i^b = C_n^b[(1+x_i)^\mu (1-x_i)^\nu\, P_n'(x_i)]^{-1} = C_n^b[r(x_i)P_n'(x_i)]^{-1} \tag{2.93}$$

All the quadrature weight expressions are of the form:

Table 2.5 Parameters in Weights, Eqs. (2.93) and (2.94)			
Barycentric Weights	C_n or C_n^b	μ	ν
Full	-1	1	1
Shortcut Even	$-1/(a_n\sqrt{2})$	½	1
Shortcut Odd	$-1/a_n$	1	1
Symmetric	-1	0	1
Quadrature Weights			
Gauss Full	2	½	½
Gauss Shortcut Even	½	½	½
Gauss Shortcut Odd	1	1	½
Lobatto Full	$8(n+1)/(n+2)$	1	1
Radau Right	4	½	1
Radau Left	4	1	½
Symmetric Gauss	½	½	½
Symmetric Lobatto	$(n+1)/(n+\beta+1)$	½	1

$$W_i = C_n[(1+x_i)^\mu(1-x_i)^\nu\, P_n'(x_i)]^{-2} = C_n[r(x_i)P_n'(x_i)]^{-2} \tag{2.94}$$

where $P_n'(x_i)$ are the derivatives of the Jacobi polynomials listed in Table 2.1 or their shortcut equivalents evaluated at the roots of the polynomial. The values of the parameters in Eqs. (2.93) and (2.94) are given in Table 2.5. The constants are given on the interval [-1,1] for nonsymmetric cases. As stated above the normalized barycentric weights should be doubled and quadrature weights halved for the interval [0,1]. For the symmetric cases, the constants are given on the interval [0,1] even though the roots are computed on [-1,1]. The normalizing factors, ρ_n and $\tilde{\rho}_n = 2^n\rho_n$, for the barycentric weights are related by Eq. (2.17), see Eq. (2.75). As explained in Section 2.4.5, the nonsymmetric quadrature weights sum to 2, while the symmetric weights must sum to $1/(\gamma+1)$, where γ = 0,1,2 for planar, cylindrical and spherical geometry.

To illustrate the basic idea behind the method, we develop a first approximation using a one term Taylor series approximation to Eqs. (2.93) and (2.94). The approximation is:

$$r(x)P'(x) = r(x_0)P'(x_0) + [r'(x_0)P'(x_0) + r(x_0)P''(x_0)](x-x_0) \tag{2.95}$$

where $r(x) = (1+x)^\mu(1-x)^\nu$ and x_0 is the approximate root. The subscript on P has been omitted. After differentiating $r(x)$ and substituting the Sturm-Liouville relationship, Eq. (2.32) or (2.58), for the second derivative, the equation simplifies to:

$$r(x)P'(x) = r(x_0)P'(x_0)\left[1 - (c_{10} + c_{11}x_0)\,\frac{(x-x_0)}{1-x_0^2}\right] \tag{2.96}$$

Where $c_{10} = \nu - \mu - \acute{b} = \nu - \mu - (\alpha - \beta)$ and $c_{11} = \nu + \mu - \acute{a} = \nu + \mu\ - (\alpha + \beta + 2)$. Substituting a simple Newton-Raphson linearization, like Eq. (2.48), produces the following more accurate expression when the roots are subject to small errors:

$$r(x)P'(x) = r(x_0)P'(x_0)\left[1 + (c_{10} + c_{11}x_0)\,\frac{P(x_0)}{(1 - x_0^2)P'(x_0)}\right] \tag{2.97}$$

This equation contains the basic value together with a correction. If the root is accurate, $P(x_0) = 0$, so the correction term should be small. When the parameters for Gaussian quadrature are substituted, the third expression of Eq. (2.78) results. For Lobatto quadrature the second expression of Eq. (2.81) results, while for Radau quadrature the fourth expression of Eq. (2.84) results.

We find that these formulas with the correction produce error growth rates ranging $n^{1.4}$ to $n^{1.9}$, which is not substantially better than some of the other formulas. The growth rate is n^2 even when continued products are used to calculate the weights, Eqs. (2.67) to (2.70). This result would seem to conflict with the contentions of Yakimiw (1996). The utility of Eq. (2.97) depends on the accuracy of the coefficients and the roots and on the magnitude of the second term within the brackets. If the equation is applied with quad (128 bit) precision, the weight errors are ~10^{-25} or less, even if the roots have only double precision accuracy, i.e. ~10^{-16}. If the roots have double precision accuracy and one uses the same double precision recurrence calculations to determine the coefficients of Eq. (2.97), the improvement is relatively small.

Yakimiw determined roots using an iterative calculation with recurrence relations used to evaluate the polynomials. However, the accuracy for the weights near the boundary was improved by using Eq. (2.97) together with a slower but more accurate Fourier formula for the polynomials. Swartztrauber (2003) points out the true utility of the higher order weight calculations is that accurate weights can be calculated with less accurate roots. He found that with a Newton-Raphson root calculation, one polynomial evaluation could normally be eliminated (see discussion in Section 2.3.4). Appendix A.1 contains additional discussion and error analysis.

To produce even higher order weight expressions, the terms in brackets of Eqs. (2.93) and (2.94) can be expanded in much the same way as the root expansion, Eq. (2.57). We will use:

$$r(x)P'(x) = P'(x_0)\sum_{k=0}^{M}\frac{d_j(x_0)}{k!}q(x_0)^k = P'\left[d_0 + d_1 q + \frac{d_2}{2}q^2 + \frac{d_3}{3!}q^3 + \cdots\right] \tag{2.98}$$

Where P is the polynomial of interest, x_0 is a root subject to errors and $q(x) = P(x)/P'(x)$ is the same quantity used in the root expansion, Eq. (2.57). Obviously, $d_0 = r(x_0)$, while the other coefficients in the expansion can be determined using the Sturm-Liouville relationship, Eq. (2.32) or in generic form by Eq. (2.58).

The higher terms are found by setting the derivatives to zero in order to reduce the sensitivity of the function to errors in x:

$$0 = P''\left[d_0 + d_1 q + \frac{d_2}{2}q^2 + \frac{d_3}{3!}q^3 + \cdots\right] + P'\left[d_0' + d_1' q + \frac{d_2'}{2}q^2 + \frac{d_3'}{3!}q^3 + \cdots\right] + P'\left[d_1 + d_2 q + \frac{d_3}{2}q^2 + \frac{d_4}{3!}q^3 + \cdots\right]q' \tag{2.99}$$

This equation can be simplified by substituting the right hand side of Eq. (2.58) for P'' and $q' = 1 - b_1(x)q - b_0(x)q^2$, where b_0 and b_1 are from the same equation. After removing the common factor P' the equation simplifies to:

$$0 = \left[d_0 + d_1 q + \frac{d_2}{2}q^2 + \frac{d_3}{3!}q^3 + \cdots\right](b_1 + b_0 q) + \left[d_0' + d_1' q + \frac{d_2'}{2}q^2 + \frac{d_3'}{3!}q^3 + \cdots\right] + \left[d_1 + d_2 q + \frac{d_3}{2}q^2 + \frac{d_4}{3!}q^3 + \cdots\right](1 - b_1 q - b_0 q^2) \tag{2.100}$$

The collection of terms multiplying each power of q is set to zero, producing the following recursive formula for the higher coefficients:

$$d_{k+1} = -d_k' + (k-1)b_1 d_k + k(k-2)b_0 d_{k-1} \tag{2.101}$$

After working through the algebra and simplifying, the expansion can be expressed by:

$$r(x)P'(x) = r(x_0)P'(x_0)\sum_{k=0}^{M}\frac{\hat{d}_k(x_0)}{k!}\hat{q}(x_0)^k \tag{2.102}$$

where: $\hat{q}(x) = P(x)/[(1-x^2)P'(x)]$ and $\hat{d}_k(x) = c_{k0} + c_{k1}x + \cdots + c_{kk}x^k$. The terms through M = 3 are:

$$c_{00} = 1$$
$$c_{10} = \nu - \mu - \acute{b}$$
$$c_{11} = \nu + \mu - \acute{a}$$
$$c_{20} = (\nu - \mu)c_{10} - c_{11} + \acute{c}$$
$$c_{21} = (\nu - \mu)c_{11} + (\nu + \mu - 2)c_{10}$$
$$c_{22} = (\nu + \mu - 1)c_{11} - \acute{c}$$
$$c_{30} = (c_{20} + \acute{b}c_{10})(\nu - \mu) + (\acute{c} - c_{11})\acute{b} - c_{21}$$
$$c_{31} = (\acute{a}c_{10} + \acute{b}c_{11} + c_{21})(\nu - \mu) + (\acute{b}c_{10} + c_{20})(\mu + \nu) + (\acute{c} - c_{11})\acute{a} - 2(\acute{b}c_{10} + c_{22}) - 4c_{20}$$
$$c_{32} = (\acute{a}c_{11} + c_{22})(\nu - \mu) + (\acute{a}c_{10}' + bc_{11} + c_{21})(\mu + \nu) - 2\acute{a}c_{10} - (\acute{c} + c_{11})\acute{b} - 3c_{21}$$
$$c_{33} = (\acute{a}c_{11} + c_{22})(\nu + \mu) - (\acute{c} + c_{11})\acute{a} - 2c_{22}$$

Eq. (2.58) gives the values of $\acute{a}, \acute{b}$ and $\acute{c}$.

Figs. 2.15 and 2.16 show examples of the sensitivity of weight errors to errors in the roots when the weights are calculated with Eq. (2.102) substituted into Eqs. (2.93) and (2.94). Errors in the Lobatto quadrature weights are shown in Fig. 2.15 and errors in the barycentric weights for Radau points are shown in Fig. 2.16. In each case n = 14 and the error is shown for the weight nearest the boundary. In most cases, the rate of convergence is linear with no correction and the convergence rate increases with each increment of M, i.e. usually $\epsilon_w = (\epsilon_x)^{(M+1)}$. Some formulas display second order convergence, equivalent to M = 1, without an

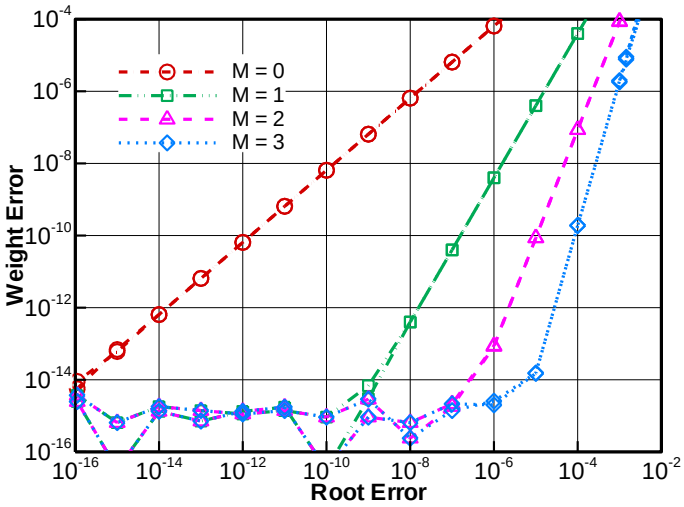


Fig. 2.15 Lobatto quadrature weight error, first point, n = 14

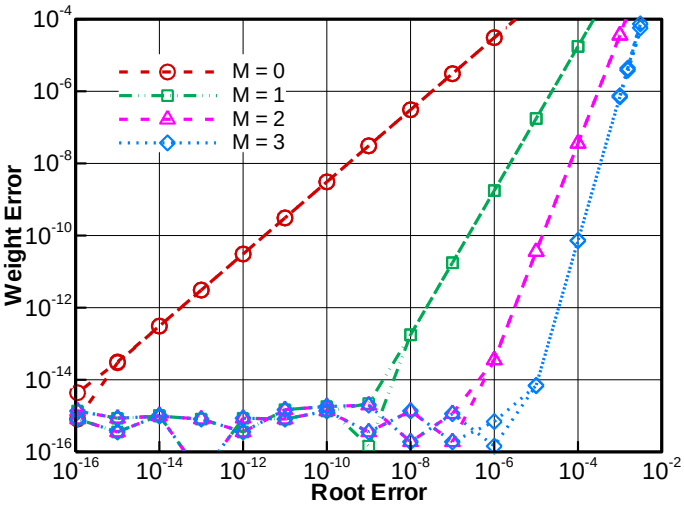


Fig. 2.16 Radau points, barycentric weight error, first point, n = 14

explicit correction term added, but these cases are exceptional (see Fig. 2.18 below for an example).

In Section 2.3.3, the root estimation methods were found to produce maximum errors of approximately 10^{-7} when n = 14. The results in Figs. 2.15 and 2.16 suggest the estimated roots should produce weights with maximum accuracy when M = 2. Figs. 2.17, 2.18 and 2.19 show the weight errors as a function of n and M when calculated with the estimated roots. Quadrature and barycentric weight errors are shown in Figs. 2.17 and 2.18 for Gauss, Lobatto and Radau points. Lobatto quadrature weight errors for symmetric problems are displayed in Fig. 2.19. For nonsymmetric problems, weights of maximum accuracy are obtained for n > 44, 7 and 2 for M = 1, 2 and 3, respectively. For symmetric problems with cylindrical coordinates, the problems are like those for Gauss and Radau points, see Tables 2.1 and 2.4, so the accuracy is like that for nonsymmetric problems. For symmetric planar and spherical geometry, the problems are like nonsymmetric problems with shortcut calculations, so accuracy like the other cases is achieved with roughly half the value of n or n > 23, 3 and 1 for M = 1, 2 and 3, respectively. Except for $n \le 2$, accurate weights can be calculated directly from the estimated roots. Note that the barycentric weights for Gauss points, Fig. 2.18, do not require a first correction to achieve second order convergence.

In section 2.3.4 the combination of accurate root estimates and higher order iteration was found to give accurate roots without iteration. The calculations presented here in Figs. 2.17, 2.18, and 2.19 show that accurate weights can be calculated directly from the estimated roots. The only exceptions are $n \le 2$, which is easily rectified by storing the exact roots for these few cases and

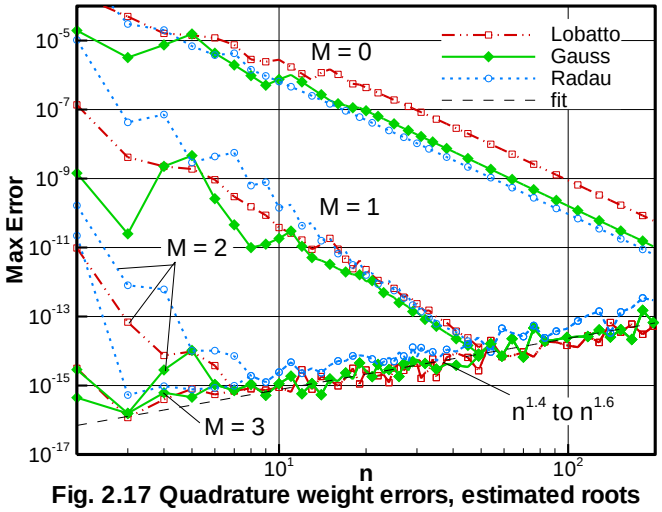


Fig. 2.17 Quadrature weight errors, estimated roots

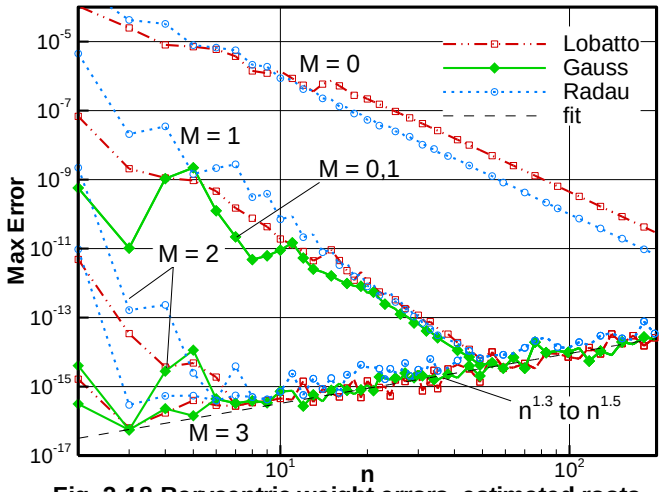


Fig. 2.18 Barycentric weight errors, estimated roots

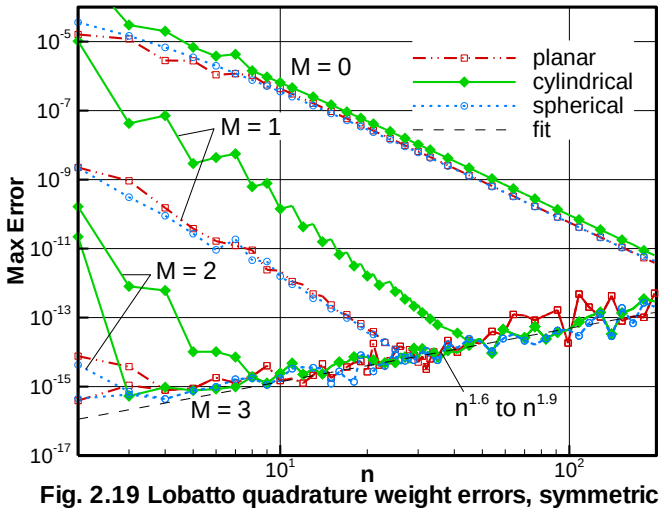


Fig. 2.19 Lobatto quadrature weight errors, symmetric

substituting them for the estimates. As a results, accurate roots and weights can be calculated directly from the estimated roots. No iterations are required and the polynomial and its first derivative are calculated only once. When shortcut polynomial recurrence calculations are used for the most common ultraspherical cases, the polynomial calculation requires O($5n^2/4$) floating point operations. All other operations are O(n).

The principal advantage of the higher order weight calculation described here is that only a single calculation of the orthogonal polynomial is required for each root. A simple alternative is the to use Taylor series. Approximate values for polynomial and derivative are calculated during the root calculation, while Eq. (2.57) gives the correction to the roots. A simple two or three term Taylor series is all that is required to produce accurate values at the roots. This approach appears to produce a modest loss of accuracy, so was not used for that reason.

2.4.8 Accuracy and Efficiency of Weight Calculations

This section discusses the accuracy and efficiency of the barycentric and quadrature weight calculations. The accuracy and efficiency of the weight calculations has been the subject of many studies [e.g. Lether (1978), Yakimiw (1996), Swarztrauber (2003), Hale and Townsend (2013), Bogaert (2014)]. The eigenvalue method of Golub and Welsch (1969) is the most common method for calculating roots and weights, see Section 2.3.1. The weights can be determined from the first components of the associated eigenvectors or by using the weight formulas in Sections 2.4.1 to 2.4.5. Previous studies have found the eigenvector method to be more prone to rounding errors with error growth rates of n^2. Recurrence calculations used in conjunction with the formulas are slightly more accurate, but the growth rates are still $n^{1.5}$ to n^2. Many applications of MWR produce accurate solutions with small n, $n \lesssim 10$, so for those problems accuracy and efficiency is not an issue. However, some problems require larger n. Some global atmospheric models have used n of several thousand [Yakimiw (1996)].

If the calculations do not produce sufficient accuracy, there are potentially two solutions. Either the calculations can be performed with greater floating-point precision or possibly the calculation procedure can be improved. As of this writing, many computing systems provide

three levels of floating-point precision. If coded intelligently, most programming languages make it easy to change the precision. Often, 32 bit, 64 bit and 128 bit calculations are provided using the IEEE-754 specification. The precision is best described in terms of the machine epsilon, which is the smallest number which is significant relative to unity. The epsilon values are approximately $1x10^{-7}$, $2x10^{-16}$ and $2x10^{-34}$ for 32 bit, 64 bit and 128 bit calculations, respectively. This can be thought of as 7, 16 or 34 digits of precision. These alternatives will also be called single, double and quad precision. The code written in this project makes precision changes easy. However, there is a significant loss of efficiency when using higher precision calculations, and precision changes are difficult or impossibe to implement with some systems, so it is best to use calculation procedures which are as accurate as possible. For these reasons, the accuracy of various procedures was investigated. The details of this investigation are provided in Appendix A.1 and A.2, while the results and algorithms are summarized here. All of the results shown here use accurate initial estimates described in Section 2.3.3. For n < 4, more accurate roots are stored and used in place of the estimates. Higher order root and weight calculations are used so that iteration is not required and only one polynomial evaluation is needed.

First, we compare results using recurrent calculation for the full polynomial with those using shortcut calculations. Fig. 2.20 compares calculations for Gauss points. For the results reported here, the errors are maximum fractional errors, also called maximum relative errors defined by $\max_i(|\epsilon_i/W_i|)$, where ϵ are the errors and W are the "correct" values. The only exception is that errors in the roots are simply the maximum error. The reported growth rates are determined from the results for $n > 40$. Here the shortcut calculations are implemented using the third approach discussed in Section 2.3.4, i.e. the shortcut polynomials are calculated and the values converted to the corresponding full polynomial values using Eqs. (2.18), (2.21), (2.30) and (2.31). The shortcut calculations were performed on both the full interval [-1,1] and the shifted interval [0,1]. The figure shows that full calculations produce more accurate roots as discussed at the end of Section 2.3.3 (see Figs. 2.11 and 2.12). However, if shifted shortcut calculations are used, the error growth rate is reduced from $n^{0.9}$ to a modest $n^{0.4}$. For this case full calculations give somewhat less accurate weights than the shifted shortcut calculations ($n^{1.8}$ versus $n^{1.4}$). For other quantities (see Appendix A.1) the observed differences between full calculations and shifted shortcut calculations are not large and overall neither method is clearly more accurate. The shortcut method is preferred since it is more efficient.

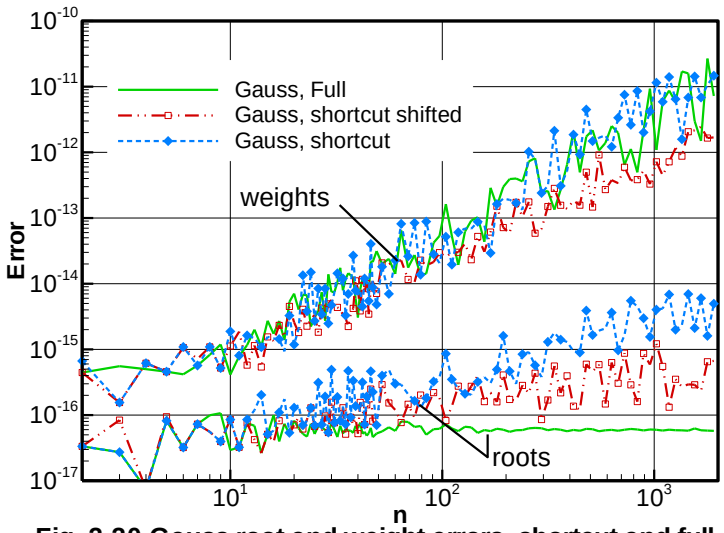


Fig. 2.20 Gauss root and weight errors, shortcut and full

Figure 2.21 shows quadrature weight and root errors for Gauss, Lobatto and Radau quadrature when calculated with the noniterative higher order methods. The error growth rate for the roots is small, while for the quadrature weights it is $n^{1.4\text{-}1.7}$. The accuracy is somewhat better than others have reported with either the Golub-Welch method or recurrence calculations [see Fig. 2.1, Hale and Townsend (2013)]. Based on the results of others and the analysis in Appendix A.1, we have concluded that the results in Fig. 2.21 are the best that can be achieved when the recurrence relationships are used to calculate the polynomials.

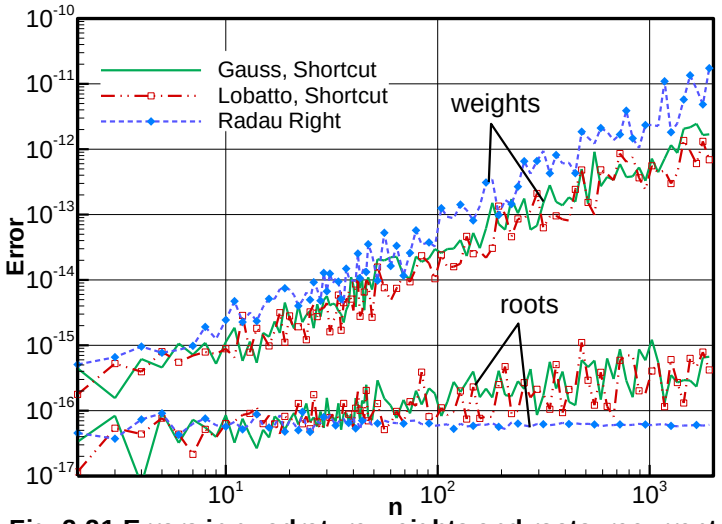


Fig. 2.21 Errors in quadrature weights and roots, recurrent

The primary difficulty is that for large n, the accuracy of the recurrence calculations deteriorates near the boundaries. Yakimiw circumvented this inaccuracy problem by using an alternate Fourier method for calculating the polynomial values and derivatives near the boundary. His use of the Fourier calculation is the primary reason for his improved results. Swartztrauber's Fourier method is somewhat similar. They both use the Fourier procedure to calculate the critical weights near the boundary. A key difference is that Yakimiw calculated roots using recurrence calculations, whereas Swartztrauber used Fourier calculations. Another difference is that Swartztrauber does all the calculations in terms of $\theta = \arccos(x)$ rather than x. The Fourier method for polynomial calculation requires computational effort of O(n^2) involving transcendental functions, which are computationally intensive.

Rather than use an inefficient Fourier method for polynomial calculation, Hale and Townsend (2013) used asymptotic approximations for the polynomials. They address Gaussian and Jacobi-Gauss quadrature strictly for $n > 50$. Appendix A.2 describes asymptotic calculations which have the advantage of both greater accuracy and greater efficiency. If n is large enough the efficiency is greater because there are no calculations of O(n^2) which is the case for using the recurrence relationships or a Fourier method. As was the case with root estimation, asymptotic approximations for the Jacobi polynomials subdivide into those accurate near the boundary and those accurate away from the boundary. We will again refer to these as *boundary* and *interior* methods. The asymptotic approximations also involve trigonometric and Bessel functions, so they are computationally intensive even though the number of terms is much less than the with Fourier method.

Appendix A.2 describes the construction of a composite scheme using recurrence relations together with boundary and interior asymptotic approximations. The composite algorithm uses cutoff values of n and error analysis to determine the best procedure to use; recurrent, boundary or interior. The procedure used varies for each point and for the interior asymptotic

method, the order of the approximation (number of terms) for each root is also determined. Although the scheme could be further optimized, the following procedure is simple and provides a good balance between accuracy and efficiency:

- $n \leq n_{\text{Asmp}}$: the recurrence relationships are used exclusively
- $n > n_{\text{Asmp}}$: recurrence, interior and boundary calculations are used

n_{Asmp} = 40 was used in the results reported. Fig 2.22 shows computation times for calculation of the roots, quadrature weights and barycentric weights for Gauss, Lobatto and Radau points. Gauss and Lobatto quadrature required virtually the same times. The computing times were recorded on a typical 2019 laptop with an Intel Core i7-7600u, 2.8 GHz (2 cores and 4 logical processors). The GNU Fortran (gcc ver. 5.4.0, Cygwin) compiler was used. The calculations were repeated enough times to obtain reproducible timings. In order to avoid artificially good results due to cache reuse, etc., each calculation used separate areas of memory for the roots and weights.

At very large n the computation is O(n^2) with recurrent calculations and O(n) with composite calculations, as expected. Also at large n, the computing time for Lobatto or Gauss recurrent shortcut calculations, full Lobatto/Gauss calculations and Radau calculations are in the ratio of roughly 1:2:4. Also, the asymptotic Lobatto/Gauss to Radau calculations are about 1:2. The Radau calculations require more effort because all roots are unique. However, at small n the time appears to be dominated by overhead effects, e.g. function call overhead, so there is little difference between shortcut and full recurrent calculations. The breakeven point between recurrent and composite asymptotic calculations is n = 500 to 1000. Gauss or Lobatto points and weights with n = 100 can be calculated in only 25 or 60 μsec with recurrent and composite calculations, respectively, while a million points and weights can be calculated in less than 0.3 sec.

With n_{Asmp} of 40 the switch from recurrent to composite method causes a factor of 3 to 4 increase in computation time, but the overall time is so small it hardly seems relevant, so the small improvement in accuracy seems justified. For Gauss and Lobatto points and n = 44, the algorithm uses the boundary method for only one point, while recurrent calculations are used for 40% of the points. By n = 110, recurrent calculations are no longer used, while the boundary method is used for 9 points. At n = 2500, less than one percent (12 points) use the boundary method and of those using the interior method 86% use only 4 or

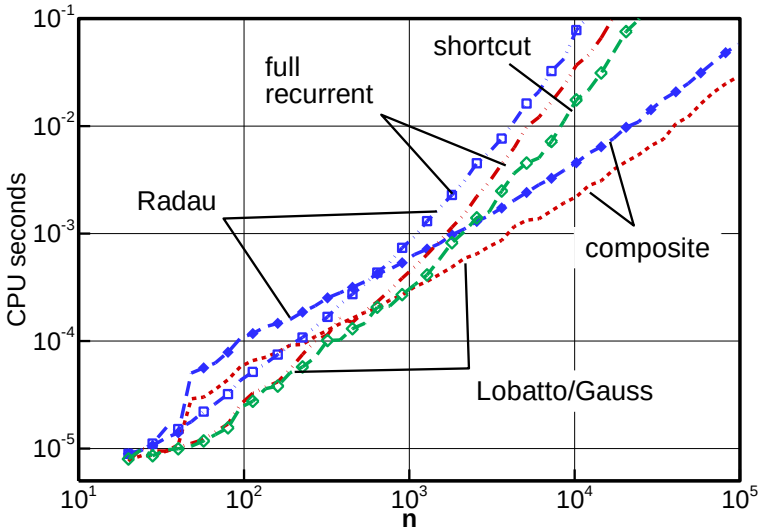


Fig. 2.22 CPU time for calculation of roots and weights [97]

5 terms in the approximation. Additional details are given in Appendix A.2.

The processor and compiler should produce code to utilize vector processing hardware. We were not able to produce vector calculations even with code that vectorizes on other systems. Based on published benchmarks, other compilers would likely be more efficient. Nevertheless, the calculations should provide a good indication of the tradeoffs between different calculation procedures.

Figs. 2.23 through 2.27 show the absolute errors for the roots and relative or fractional errors for the weights when the composite formulas are used to calculate the polynomials. The figures include all the cases likely to be needed for MWR calculations on a finite domain. The barycentric weights are usually as accurate or slightly more accurate than the quadrature weights. Note that with only a few exceptions the error rises initially due to use of the recurrence calculations, but

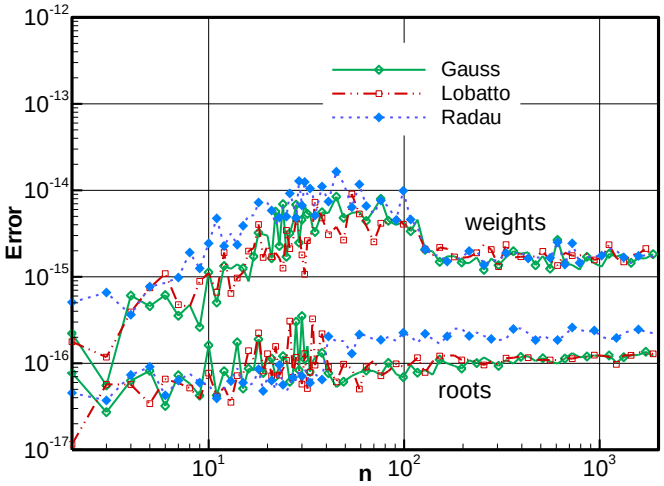


Fig. 2.23 Errors in quadrature weights and roots, composite

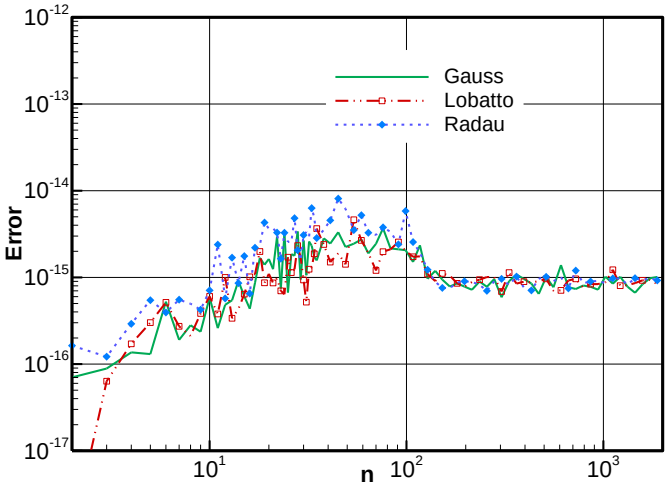


Fig. 2.24 Errors in barycentric weights, composite

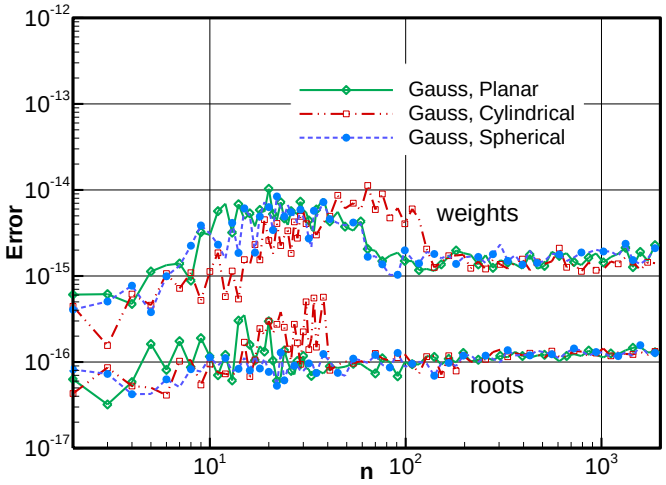


Fig 2.25 Error in quadrature weights and roots, composite

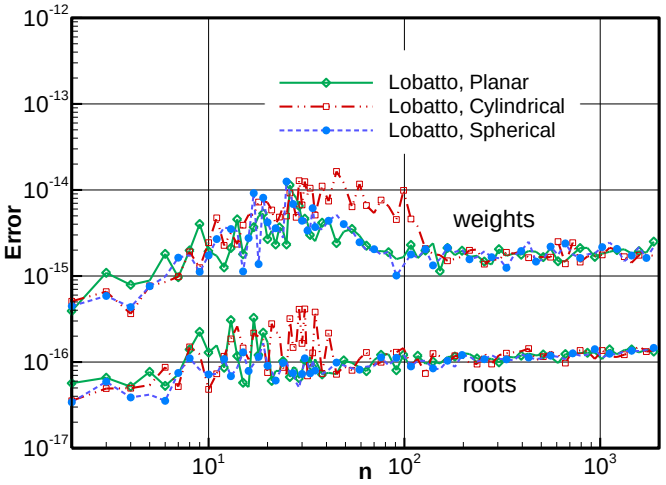


Fig 2.26 Error in quadrature weights and roots, composite

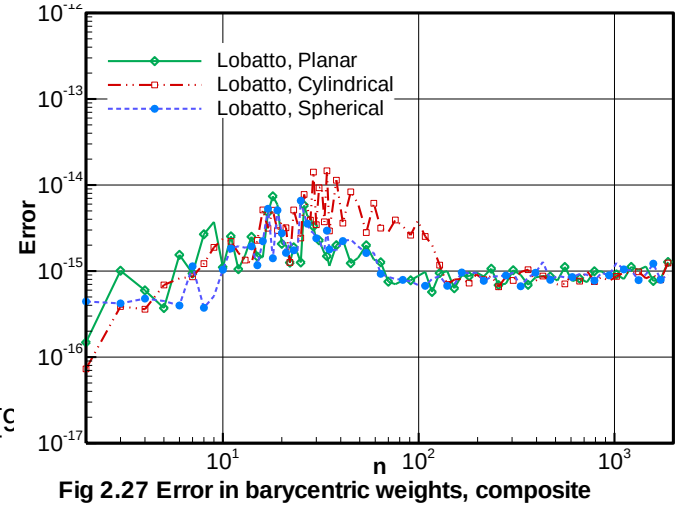


Fig 2.27 Error in barycentric weights, composite

ultimately the error is essentially constant. Only a few cases produced weight errors greater than 10^{-14}.

2.5 Nodal Differentiation Matrices

To develop nodal approximations, derivatives of the basis functions in Eq. (2.1) are required. The first derivative of the Lagrange interpolating polynomials is straight forward. The only trick involved is to take the derivative of the logarithm of Eq. (2.1), thereby converting the continued product to a summation. For the nonsymmetric case, this approach leads to:

$$\begin{aligned} A_{ij} = \left.\frac{d\ell_j(x)}{dx}\right|_{x_i} &= \frac{\hat{p}_n'(x_i)}{(x_i - x_j)\hat{p}_n'(x_j)} = \frac{W_j^b}{(x_i - x_j)W_i^b} \quad \text{for } i \neq j, \text{ and} \\ &= \sum_{\substack{k=0 \\ k\neq i}}^{n+1} \frac{1}{x_i - x_k} \qquad \text{for } i = j \end{aligned} \tag{2.103}$$

and for the symmetric case:

$$\begin{aligned} A_{ij} &= \frac{2x_i\hat{p}_n'(x_i^2)}{(x_i^2 - x_j^2)\hat{p}_n'(x_j^2)} = \frac{2x_iW_j^b}{(x_i^2 - x_j^2)W_i^b} \quad \text{for } i \neq j, \text{ and} \\ &= \sum_{\substack{k=1 \\ k\neq i}}^{n+1} \frac{2x_i}{x_i^2 - x_k^2} \qquad \text{for } i = j \end{aligned} \tag{2.104}$$

These expressions contain the barycentric weights, $W_i^b = 1/\hat{p}_n'(x_i)$, which are common to the relationships for the interpolating polynomials, $\boldsymbol{\ell}$, and quadrature weights, $\boldsymbol{W}$, so they have multiple uses. As an alternative to the equations above, the diagonal elements can be calculated so the row sums are zero. With Lobatto points, $A_{ii} = 0$ for i = 1,…n for nonsymmetric problems. These expressions are general for any set of points, but simplifications can be made in specific cases. Many texts develop one formula for Gauss points, another for Lobatto points, a third for Chebyshev points and so forth. Also, many times the formulas fail to include endpoints. There is no need to make distinctions for the different points, since the formulas above are valid for any choice of points, even equally spaced points. These formulas are especially useful since the barycentric weights are needed for other purposes, so they reduce redundancy in the calculations. The code box in section 2.3.1 shows concise code implementing this calculation along with the calculation of points and quadrature weights.

Many texts and articles have claimed the original development of Eq. (2.103), some as late as almost 1990. I first found a version of it in Ferguson (1971) and programmed it in about 1972. I tested it by solving a problem with n = 45, which was thought to be enormous at the time. It is given by Michelsen and Villadsen (1972) without reference. The earliest reference I have found is Nielsen (1956, pp. 150-4). It is likely this formula is quite ancient.

Early descriptions of the method used a far less accurate procedure for calculating the differentiation matrices. Following Hamming (1962), the differentiation matrices and quadrature weights were determined by inverting the Vandermonde matrix [Villadsen (1970), Finlayson (1972)]. Due to the notoriously ill conditioned Vandermonde matrix, the accuracy of this approach is poor for $n \gtrsim 10$. This problem held back development of the method for large problems. Despite its limitations the approach was still in use beyond 1990.

For the nonsymmetric case, the second derivative can easily be calculated from the first derivative approximation by a matrix multiply, i.e. $\boldsymbol{B} = \boldsymbol{AA}$, or:

$$B_{ij} = \sum_{k=0}^{n+1} A_{ik} A_{kj} \tag{2.105}$$

For the symmetric cases, the Laplacian operator is not as simple as Eq. (2.105). Since the interpolating polynomial is even, its first derivative is odd. The first derivative operator from Eq. (2.104) is only valid if it operates on a symmetric quantity. Instead, a similar operator is needed for odd functions. The Lagrange interpolating polynomial for an odd function is related to the even function interpolating polynomial by:

$$\hat{\ell}_i = \frac{x}{x_i} \prod_{\substack{j=1 \\ j \neq i}}^{n+1} \frac{x^2 - x_j^2}{x_i^2 - x_j^2} = \frac{x}{x_i} \ell_i(x^2) \tag{2.106}$$

The odd function interpolating polynomial and derivative operator are denoted with a hat (^). The direct differentiation of Eq. (2.106) gives:

$$\hat{A}_{ij} = \frac{1}{x^\gamma} \frac{d}{dx} \left[x^\gamma \hat{\ell}_j \right]_{x_i} = \frac{\gamma + 1}{x_i} \delta_{ij} + \frac{x_i}{x_j} A_{ij} \tag{2.107}$$

where γ = 0,1,2 for planar, cylindrical and spherical coordinates. The Laplacian operator for a symmetric problem can be calculated by $\boldsymbol{B} = \hat{\boldsymbol{A}}\boldsymbol{A}$.

The Laplacian operator, $\boldsymbol{B}$, can also be determined by direct differentiation of Eq. (2.1). For the nonsymmetric case, the values from direct differentiation are:

$$\begin{aligned} B_{ij} &= \frac{2\hat{p}_n'(x_i)}{(x_i - x_j)\hat{p}_n'(x_j)} \left[\sum_{\substack{k=0 \\ k \neq i \\ k \neq j}}^{n+1} \frac{1}{x_i - x_k} \right] = 2A_{ij} \left(A_{ii} - \frac{1}{x_i - x_j} \right) \text{ for } i \neq j, \text{ and} \\ B_{ii} &= \sum_{\substack{j=0 \\ j \neq i}}^{n+1} \left[\frac{1}{x_i - x_j} \sum_{\substack{k=0 \\ k \neq i \\ k \neq j}}^{n+1} \frac{1}{x_i - x_k} \right] = A_{ii}^2 - \sum_{\substack{k=0 \\ k \neq i}}^{n+1} \frac{1}{(x_i - x_k)^2} \end{aligned} \tag{2.108}$$

The corresponding expression for the symmetric case is:

$$B_{ij} = \frac{2\hat{p}_n'(x_i^2)}{(x_i^2 - x_j^2)\hat{p}_n'(x_j^2)}\left[\gamma + 1 + \sum_{\substack{k=1 \\ k\neq i \\ k\neq j}}^{n+1} \frac{4x_i^2}{x_i^2 - x_k^2}\right] = 2A_{ij}\left(A_{ii} + \frac{\gamma+1}{2x_i} - \frac{2x_i}{x_i^2 - x_j^2}\right) \text{ for } i \neq j$$

$$B_{ii} = A_{ii}^2 + \frac{\gamma+1}{x_i}A_{ii} - \sum_{\substack{k=1 \\ k\neq i}}^{n+1} \frac{4x_i}{(x_i^2 - x_k^2)^2} \tag{2.109}$$

As an alternative to the equations above, the diagonal elements can be calculated so the row sums are zero.

The nodal differentiation matrices are subject to the same types of rounding errors discussed in Section 2.4.8. Since the barycentric weights are calculated accurately, the primary issue is the calculation of the differences in nodal values, x_i - x_j. If these terms are calculated directly, the error growth rate is O(n^2). If the roots are calculated in θ coordinates, where $x = \cos(\theta)$, then differences can be calculated more accurately using the trig identity:

$$x_i - x_j = \cos\theta_i - \cos\theta_j = -2\sin\left(\frac{\theta_i + \theta_j}{2}\right)\sin\left(\frac{\theta_i - \theta_j}{2}\right) \tag{2.110}$$

The utility of Eq. (2.110) was tested by calculating some matrix error norms of the form $\| A - \bar{A}\|/\| \bar{A}\|$, where the overbar indicates an accurate matrix (quad precision). Several types of norms were considered, but they all give the same general picture. An example of these calculations is shown in Fig. 2.28 for the first derivative matrix and the stiffness matrices discussed in the next section. The results using Eq. (2.110) are labeled with θ, while the direct calculation is labeled with x. As we would expect, the error growth rates are O(n) when Eq. (2.110) is used and O(n^2) when the quantities are directly calculated. The differences are small when recurrence calculations are used, n < 40. By n = 100, Eq. (2.110) is more accurate by an order of magnitude.

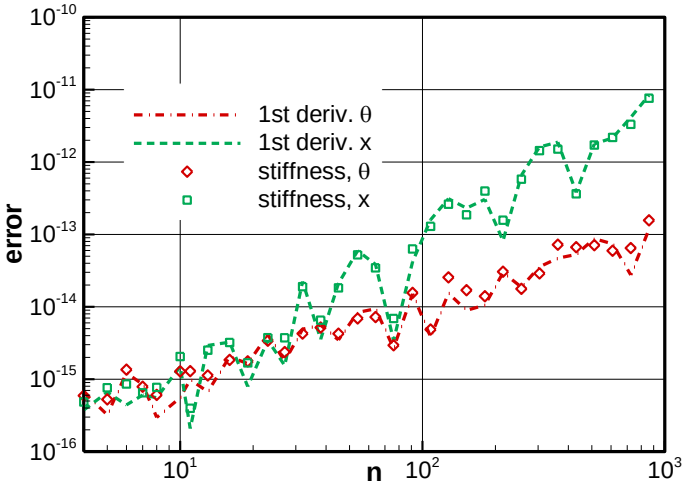


Fig. 2.28 L_1 error norms, Lobatto points, first derivative and stiffness matrices

2.6 Nodal Stiffness Matrices

In finite element methods an alternate approximation is used for the Laplacian. It is called the *stiffness matrix*, which reveals the origins of these methods in structural mechanics. However, it is a general alternative representation of the Laplacian in *weak* form. We generalize it to some extent for use with collocation. It is easily calculated from the other arrays. For nonsymmetric problems:

$$C_{ij} = \delta_{i,n+1}A_{n+1,j} - \delta_{i,0}A_{0,j} - W_iB_{ij}$$
$$= \sum_{k=0}^{n+1} W_k A_{ki} A_{kj} = \int_0^1 \frac{d\ell_i(x)}{dx}\frac{d\ell_j(x)}{dx}dx \tag{2.111}$$

For symmetric problems there is no boundary point at i = 0, so that term is omitted and the $A_{0,j}$ term does not appear in Eq. (2.111). For further details see the discussion of the Galerkin method and method of moments in sections 3.1.2 and 3.1.3 of Chapter 3. As explained there, the top expression amounts to a simple rearrangement of the basic equations, so it is valid for any set of points.

The validity of the second set of equalities depends on the accuracy of the quadrature. The integrand is a polynomial of degree $2n$, so Lobatto and Radau quadrature are valid. It is also valid for Gauss points in symmetric problems. For nonsymmetric problems with Gauss points and for Chebyshev points, the top expression must be used to calculate the stiffness matrix.

2.6.1 Matrix Symmetry with Gauss Points

It is obvious that the stiffness matrix is symmetric whenever the second equality holds for Eq. (2.111), i.e. always for Radau and Lobatto points and for symmetric problems with Gauss points. From calculations, the stiffness matrix is known to be symmetric for nonsymmetric problems with Gauss points also. For this case, symmetry can be shown theoretically as well. The proof is presented, since it is instructive. The interval [-1,1] is used for convenience.

Due to the symmetry of the points about the midpoint of the interval, the terms in the first derivative matrix are related by:

$$A_{ij} = -A_{n+1-i,n+1-j} \tag{2.112}$$

So the corner elements are symmetric, i.e. $A_{0,n+1} = -A_{n+1,0}$. Next, we show that the interior elements are symmetric using:

$$W_iB_{ij} = \int_{-1}^{1} \ell_i^*(x)\ell_j''(x)dx \tag{2.113}$$

for i = 1,..,n, j = 1,..,n and where:

$$\ell_i^*(x) = \frac{p_n(x)}{(x-x_i)p_n'(x_i)} = \ell_i(x)\frac{1-x_i^2}{1-x^2}$$

ℓ_i^* is a reduced polynomial which interpolates through only the interior points, while ℓ_i interpolates through the interior and boundary points. The interpolating polynomials are written in terms of monic Legendre polynomials, but the formula is unchanged when written with Legendre polynomials of conventional form (scaling factors cancel). The Gaussian quadrature is exact for Eq. (2.113) because the integrand is a polynomial of degree $2n$ – 2. The quadrature reduces to a single term because $\ell_i^*(x_j) = \delta_{ij}$. For additional background, see sections 3.1.2 describing the method of moments.

First, following Eq. (2.1) write the interpolating polynomials in terms of the Legendre polynomials:

$$W_i B_{ij} = \int_{-1}^{1} \frac{q_i(x)}{p_n'(x_i)(1-x_j^2)p_n'(x_j)} \frac{d^2}{dx^2}\left[(1-x^2)q_j(x)\right]dx \tag{2.114}$$

where $q_i(x) = p_n(x)/(x-x_i)$. Next, rearrange and define an array $\boldsymbol{V}$:

$$V_{ij} = (1-x_i^2)p_n'(x_i)(1-x_j^2)p_n'(x_j)W_i B_{ij} = \int_{-1}^{1} (1-x_i^2)q_i(x) \frac{d^2}{dx^2}\left[(1-x^2)q_j(x)\right]dx \tag{2.115}$$

If $\boldsymbol{V}$ is symmetric then $\boldsymbol{WB}$ must be symmetric. Make the following substitution:

$$(1-x_i^2) = (1-x^2) + (x-x_i)(x+x_i)$$

After substitution $\boldsymbol{V}$ is:

$$V_{ij} = \int_{-1}^{1} \hat{q}_i(x) \frac{d^2\hat{q}_j(x)}{dx^2}dx + \int_{-1}^{1} (x+x_i)p_n(x) \frac{d^2\hat{q}_j(x)}{dx^2}dx \tag{2.116}$$

where $\hat{q}_i(x) = (1-x^2)q_i(x)$ is an n + 1 degree polynomial with a leading coefficient of -1, so its second derivative is a n – 1 degree polynomial with a leading coefficient of -n(n+1). If the first term is integrated by parts and the terms in the second one are represented as a polynomial series, the following results:

$$\begin{aligned} V_{ij} &= -\int_{-1}^{1} \hat{q}_i'(x)\hat{q}_j'(x)dx - n(n+1)\int_{-1}^{1} p_n(x)(p_n(x) + \ \ldots)dx \\ &= -\int_{-1}^{1} \hat{q}_i'(x)\hat{q}_j'(x)dx - n(n+1)\frac{\zeta_n^{(0.0)}}{\left(\rho_n^{(0,0)}\right)^2} \end{aligned} \tag{2.117}$$

where the ellipses indicate lower order terms which are zero due to orthogonality. This form of the relationship shows that $\boldsymbol{V}$ is clearly symmetric and therefore $\boldsymbol{WB}$ is symmetric for i = 1,..,n, j = 1,..,n.

To complete the proof we must also show the boundary or border elements are symmetric:

$$\begin{aligned} &W_i B_{i,n+1} = -A_{n+1,i} \quad \text{and} \\ &W_i B_{i,0} = A_{0,i} \end{aligned} \tag{2.118}$$

Only the first case is considered, since the two are equivalent due to the symmetry of the points about the centerline. Expressions for the reduced polynomials are given above. The derivative of the reduced polynomial and the value and derivative of the endpoint interpolating polynomial are:

$$\frac{d\ell_i^*}{dx} = \frac{(x-x_i)P_n' - P_n}{(x-x_i)^2\, P_n'(x_i)}$$

$$\ell_{n+1} = \tfrac{1}{2}(1+x)P_n(x)$$

$$\frac{d\ell_{n+1}}{dx} = \tfrac{1}{2}[(1+x)P_n' + P_n]$$

Eq. (2.113) is used with $j = n + 1$ and is integrated by parts twice. Substitution into the integral term results in:

$$W_i B_{i,n+1} = \left[\ell_i^* \frac{d\ell_{n+1}}{dx} - \ell_{n+1} \frac{d\ell_i^*}{dx}\right]_{-1}^{+1} + \frac{1}{2}\int_{-1}^{1} (1+x) P_n(x) \frac{d^2\ell_i^*(x)}{dx^2} dx \tag{2.119}$$

The terms in the integrand multiplying the Legendre polynomial form a polynomial of degree n – 2, so the integral is zero due to orthogonality. Eqs.(2.11) and (2.27) are used to evaluate the remaining terms in the equation:

$$\frac{d\ell_{n+1}(1)}{dx} = P_n'(1) + \tfrac{1}{2}P_n(1) = \tfrac{1}{2}[n(n+1)+1]$$
$$\frac{d\ell_{n+1}(-1)}{dx} = \tfrac{1}{2}P_n(-1) = \tfrac{1}{2}(-1)^n$$
$$\ell_i^*(1) = \frac{1}{(1-x_i)\,P_n'(x_i)}$$
$$\ell_i^*(-1) = \frac{(-1)^{n+1}}{(1+x_i)\,P_n'(x_i)}$$
$$\frac{d\ell_i^*(1)}{dx} = \frac{(1-x_i)n(n+1)-2}{2(1-x_i)^2\,P_n'(x_i)}$$

The following result is obtained after substituting these values:

$$\begin{aligned} W_i B_{i,n+1} &= \left[\frac{1}{1-x_i} + \frac{1}{1+x_i} + \frac{2}{(1-x_i)^2}\right]\frac{1}{2P_n'(x_i)} \\ &= \frac{2}{(1-x_i)(1-x_i^2)\,P_n'(x_i)}, \qquad \text{while} \\ -A_{n+1.i} &= \frac{-W_i^b}{(1-x_i)W_{n+1}^b} = \frac{2}{(1-x_i)(1-x_i^2)\,P_n'(x_i)} \end{aligned} \tag{2.120}$$

The last equation is the relationship for the first derivative matrix is from Eq. (2.103), which is evaluated with Eq. (2.71). This completes the proof.

It is clear from this exercise that the first relationship for the stiffness matrix in Eq. (2.111) does not generally produce a symmetric stiffness matrix. The case with Gauss points is unique and the resulting symmetry is dependent on the orthogonality of the Legendre polynomials. For other choices of points, e.g. Chebyshev or equally spaced points, the stiffness matrix is not symmetric.

2.7 Nodal Mass Matrices

The *mass matrix* is another useful matrix with a name from structural mechanics. Of course, for many applications it has nothing to do with mass. Its use is explained in sections 3.1.2 and 3.1.3 which describe the method of moments and the Galerkin method. Here we are concerned with calculation of the mass matrix. The nodal mass matrix is:

$$M_{ij} = \int_{-1}^{1} \ell_i^*(x)\ell_j(x)f(x)dx \tag{2.121}$$

where $\boldsymbol{\ell}$ are the Lagrange interpolating polynomial trial functions defined by Eq. (2.1) and $\boldsymbol{\ell}^*$ are the weight functions. For the Galekin method the weight functions equal the trial functions and for the method of moments the weight functions are the reduced polynomials which interpolate through only the interior points, see Eqs. (2.113) and (3.16). Note that $\boldsymbol{M}$ is symmetric for the Galerkin method and nonsymmetric for moments.

For collocation methods, the mass matrix is approximated by the diagonal matrix:

$$M_{ij} \approx \delta_{ij} W_i f(x_i) \tag{2.122}$$

For problems with symmetry and $f(x) = 1$, the quadrature produces an exact result. For other problems, a more accurate $\boldsymbol{M}$ is a full matrix and Eq. (2.122) is approximate. For a general function, a more accurate result can be found by using a number of extra quadrature points. If the function is a polynomial the number of quadrature points needed for an exact result is easily determined given the known accuracy of the quadrature.

2.7.1 Galerkin and Moments Mass Matrices

If $f(x) = 1$ and the problem is nonsymmetric, the associated quadrature (Gaussian for moments, Lobatto for Galerkin) is one degree shy of that needed for exact integration of Eq. (2.121). For this case, analytical expressions for the mass matrix are developed below.

The interpolating polynomials can be represented in terms of monic Jacobi polynomials, see Eq. (2.1):

$$\begin{aligned} \ell_i(x) &= \frac{(1-x^2)p_n^{(\alpha,\alpha)}(x)}{(1-x_i^2)(x-x_i)p_n^{(\alpha,\alpha)\prime}(x_i)} \\ \ell_i^*(x) &= \frac{(1-x^2)^\alpha p_n^{(\alpha,\alpha)}(x)}{(1-x_i^2)^\alpha (x-x_i)p_n^{(\alpha,\alpha)\prime}(x_i)} \end{aligned} \tag{2.123}$$

where α = 0,1 for moments and Galerkin methods, respectively and $\boldsymbol{x}$ are roots of the respective Jacobi polynomials. Define $q_i(x) = p_n^{(\alpha,\alpha)}(x)/(x-x_i)$ and

$$U_{ij} = p_n^{(\alpha,\alpha)\prime}(x_i)\, p_n^{(\alpha,\alpha)\prime}(x_j)\left(1-x_j^2\right)\left(1-x_i^2\right)^\alpha M_{ij} = \int_{-1}^{1} q_i(x)q_j(x)(1-x^2)^{\alpha+1}dx \tag{2.124}$$

First consider the case $i \neq j$ and for convenience omit the superscripts on p_n:

$$
\begin{aligned}
U_{ij} &= \int_{-1}^{1} p_n(x)\left[\prod_{\substack{k\neq i\\ k\neq j}}^{n}(x-x_k)\right](1-x^2)^{\alpha+1}dx \\
&= -\int_{-1}^{+1} p_n(x)(p_n(x)+\cdots)(1-x^2)^{\alpha}dx \\
&= \frac{-\zeta_n^{(\alpha,\alpha)}}{\left(\rho_n^{(\alpha,\alpha)}\right)^2}
\end{aligned}
\tag{2.125}
$$

The product of $(1 - x^2)$ and the continued product above is an n^{th} degree monic polynomial, so it can be represented as a series of orthogonal polynomials. The remaining term, $(1 - x^2)^{\alpha}$, is the weight function for the respective Jacobi polynomials. The ellipses indicate the lower order terms which do not contribute due to orthogonality. The resulting expression for the mass matrix is:

$$
M_{ij} = \frac{-\zeta_n^{(\alpha,\alpha)}}{P_n'(x_i)P_n'(x_j)\left(1-x_j^2\right)\left(1-x_i^2\right)^{\alpha}} = -\zeta_n^{(\alpha,\alpha)}\left(1-x_i^2\right)^{1-\alpha}\widehat{W}_i^b\widehat{W}_j^b \quad \text{for } i\neq j \tag{2.126}
$$

where $\widehat{\boldsymbol{W}}^b$ are the normalized barycentric weights defined by Eq. (2.66).

The diagonal terms are:

$$
U_{ii} = \int_{-1}^{1}[q_i(x)]^2(1-x^2)^{\alpha+1}dx \tag{2.127}
$$

First, integrate by parts using $d(x - x_i) = dx$:

$$
U_{ii} = (x-x_i)[q_i(x)]^2(1-x^2)^{\alpha+1}\big|_{-1}^{+1} + 2\int_{-1}^{1}[(x^2-1)q_i' + (\alpha+1)xq_i]p_n(x)(1-x^2)^{\alpha}dx \tag{2.128}
$$

The first term above is zero. q is a $(n$ - 1)$^{\text{th}}$ degree monic polynomial, so its derivative is an (n-2)$^{\text{th}}$ degree polynomial with a leading coefficient of n - 1, so the term in brackets can be represented by a polynomial series and the equation reduces to:

$$
U_{ii} = 2\int_{-1}^{1}[(n+\alpha)p_n+\cdots]p_n(x)(1-x^2)^{\alpha}dx = 2(n+\alpha)\frac{\zeta_n^{(\alpha,\alpha)}}{\left(\rho_n^{(\alpha,\alpha)}\right)^2} \tag{2.129}
$$

Where the ellipses again represent lower order terms which are zero due to orthogonality. The diagonal terms of the mass matrix are:

$$
M_{ii} = \frac{2(n+\alpha)\zeta_n^{(\alpha,\alpha)}}{[P_n'(x_i)]^2\left(1-x_i^2\right)^{\alpha+1}} = 2(n+\alpha)\,\zeta_n^{(\alpha,\alpha)}\left(1-x_i^2\right)^{1-\alpha}\left[\widehat{W}_i^b\right]^2 \tag{2.130}
$$

Evaluating $\zeta^{(\alpha,\alpha)}$ with Eq. (2.7) for the specific cases, the mass matrix in terms of barycentric weights for the moments method is:

$$M_{ij} = -\frac{2}{2n+1}\left(1-x_i^2\right)\widehat{W}_i^b\widehat{W}_j^b \quad \text{for } i \neq j$$
$$M_{ii} = \frac{4n}{2n+1}\left(1-x_i^2\right)\left(\widehat{W}_i^b\right)^2 \tag{2.131}$$

For the Galerkin method it is:

$$M_{ij} = -\frac{8(n+1)}{(2n+3)(n+2)}\widehat{W}_i^b\widehat{W}_j^b \quad \text{for } i \neq j$$
$$M_{ii} = \frac{16(n+1)^2}{(2n+3)(n+2)}\left(\widehat{W}_i^b\right)^2 \tag{2.132}$$

Due to the alternating signs of the barycentric weights, the off diagonal terms alternate in sign such that $M_{i,i+k}$ is positive when k is odd and negative when k is even, so the diagonal and the two terms adjacent to the diagonal are positive.

Since the barycentric weights and Gaussian quadrature weighs are related through Eq. (2.78), the coefficients for the method of moments are:

$$M_{ij} = \frac{(-1)^{(i+j+1)}}{2n+1}\sqrt{\frac{\left(1-x_i^2\right)W_iW_j}{\left(1-x_j^2\right)}} \quad \text{for } i \neq j, j = 1, \ldots, n$$
$$M_{ij} = \frac{(-1)^{(i+j+1)}}{2n+1}\sqrt{\tfrac{1}{2}\left(1-x_i^2\right)W_i} \quad \text{for } i \neq j, j = 0, n+1 \tag{2.133}$$
$$M_{ii} = \frac{2n}{2n+1}W_i\,\delta_{ij}$$

The barycentric weights cannot be expressed in terms of quadrature weights for the first and last columns. The value of the endpoint weights are ½ from Eq. (2.74), so these values are substituted to obtain the intermediate expression above.

For the Galerkin method Eq. (2.81) relates the barycentric weights and Lobatto quadrature weights. Substituting these expressions gives the following expression for the mass matrix:

$$M_{ij} = \frac{(-1)^{(i+j+1)}}{2n+3}\sqrt{W_iW_j} \quad \text{for } i \neq j$$
$$M_{ii} = \frac{2n+2}{2n+3}W_i \tag{2.134}$$

Note that for the diagonal terms in each case the coefficient of the quadrature weight is positive and less than unity, but approaches unity for large n, i.e. $0 < M_{ii} \lesssim W_i$. The diagonal and adjacent terms are positive. The row sums of the mass matrices are W_i and the column sums are W_j. The ratio of the diagonal term relative to the sum of off-diagonal terms is greater than $2n$, i.e. $M_{ii}/\sum_{j\neq i} M_{ij} > 2n$. The matrix is strongly diagonally dominant and Eq. (2.122) is a good approximation to the mass matrix.

2.8 Interpolation Using Nodal Derivatives

In some cases it is more convenient to use nodal trial functions which incorporate derivatives at the endpoints. We have already used them in the First Example, section 1.3. There we used Hermite cubic interpolation [Hildebrand (1987), p. 282]. These can be extended to higher order using both values and derivatives at several points. However, for our purposes we need interpolation using values and derivatives at the boundaries but interpolating with derivatives is usually not beneficial at interior nodes. We would prefer a formula of the form:

$$\tilde{u} = \sum_{i=1}^{n} h_i(x)\tilde{u}(x_i) + \bar{h}_0(x)\tilde{u}'(0) + \bar{h}_1(x)\tilde{u}'(1) \tag{2.135}$$

where $x_1 = 0$ and $x_n = 1$. The degree of the polynomials are $n + 1$. The functions obey the conditions: $h_i(x_j) = \delta_{ij}, h_i'(1) = h_i'(0) = 0, \text{and } \bar{h}_i(x_j) = 0,\ \bar{h}_0'(0) = \bar{h}_1'(1) = 1, \bar{h}_0'(1) = \bar{h}_1'(0) = 0$. The interior nodes are usually located at quadrature points to reduce the need for interpolating values. These functions can be represented by:

$$h_i(x) = r_i(x)\ell_i(x); \quad \bar{h}_j(x) = s_j(x)p(x) \tag{2.136}$$

for $i = 1,\ldots,n$ and $j = 0,1$. Where $p(x) = \prod(x - x_j)$ and $\ell_i(x)$ are the Lagrange interpolating polynomials. The $s_j(x)$ are linear functions, while the $r_i(x)$ are quadratic. These functions are determined by applying the conditions above. The first derivatives are:

$$\begin{aligned} h_i'(x) &= r_i'(x)\ell_i(x) + r_i(x)\ell_i'(x) \\ \bar{h}_j'(x) &= s_j'(x)p(x) + s_j(x)p'(x) \end{aligned} \tag{2.137}$$

The conditions on $r(x)$ are:

$$\begin{aligned} &r_i(x_j) = \delta_{ij} \\ &h_i'(0) = r_i'(0)\ell_i(0) + r_i(0)\ell_i'(0) = 0 \\ &h_i'(1) = r_i'(1)\ell_i(1) + r_i(1)\ell_i'(1) = 0 \end{aligned}$$

The following functions satisfy these conditions:

$$\begin{aligned} &r_i(x) = \frac{x(1-x)}{x_i(1-x_i)} \quad \text{for } i = 2, \ldots, n-1 \\ &r_1(x) = (1-x)[1 + (1-\ell_1'(0))x] \\ &r_n(x) = x[1 + (1-x)(1+\ell_n'(1))] \end{aligned}$$

The other conditions lead to:

$$\begin{aligned} &s_0(x) = (1-x)/p'(0) \\ &s_1(x) = x/p'(1) \end{aligned}$$

These functions give the following derivatives:

$$\begin{aligned} &r_i'(x) = \frac{1-2x}{x_i(1-x_i)} \quad \text{for } i = 2, \ldots, n-1 \\ &r_1'(x) = -1 + (1-2x)(1-\ell_1'(0)) \\ &r_n'(x) = 1 + (1-2x)(1+\ell_n'(1)) \\ &s_0'(x) = -1/p'(0) \\ &s_1'(x) = 1/p'(1) \end{aligned}$$

$\ell_1'(0)$ and $\ell_n'(1)$ are elements of the differentiation matrix, section 2.5, while $1/p'(0)$ and $1/p'(1)$ are barycentric weights, section 2.4.1. These relationships can be used to create a differentiation matrix in terms of the nodal values and endpoint derivatives.

Second derivatives are given by:

$$\begin{aligned} h_i''(x) &= r_i''(x)\ell_i(x) + 2r_i'(x)\ell_i'(x) + r_i(x)\ell_i''(x) \\ \bar{h}_j''(x) &= 2s_j'(x)p'(x) + s_j(x)p''(x) \end{aligned} \tag{2.138}$$

A second derivative matrix is easily calculated from these relationships.

2.9 Discrete Jacobi Transforms

As stated in Chapter 1, there are simple relationships between the nodal, modal, and monomial representations. Modal and nodal approximations are given by Eqs. (1.2) and (2.1), respectively, while monomial representations are given in Eq. (1.20) and Eq. (2.152) in the next section. Apart from possible differences in machine roundoff, there is no difference in the computed solution when any of the three representations is used for the trial solution. Although this monograph uses a nodal representation, transforms with the other representations are described for completeness. Here we examine the relationship between nodal and modal representations, while below in Section 2.10 transforms between nodal and monomial representations are described.

If one is given the modal coefficients, $\boldsymbol{a}$, in Eq. (1.2), it is straight forward to compute the values of the polynomials at the nodes and then sum the series to determine the nodal values, $u(x_i)$. However, suppose the nodal values, $\boldsymbol{u}$, are known and we wish to determine the modal coefficients which interpolate $u(x)$. This problem is not as simple. One approach would be to treat the problem as a linear algebraic system to solve for the modal coefficients. The algebraic approach is particularly problematic with monomials, since the resulting Vandermonde matrix is notoriously ill conditioned. However, there is an easier way, which is the subject here.

First, consider a discrete transform for a general Jacobi polynomial. It is analogous to a Fourier transform, so to express a function by a polynomial series of the type:

$$f(x) = \sum_{k=0}^{\infty} a_k P_k^{(\alpha,\beta)} \tag{2.139}$$

The coefficients are determined by:

$$a_k = \frac{1}{\zeta_k^{(\alpha,\beta)}} \int_{-1}^{1} f(x) P_k^{(\alpha,\beta)}(x)(1-x)^{\alpha}(1+x)^{\beta} dx \tag{2.140}$$

where $\zeta_k^{(\alpha,\beta)}$ are from Eq. (2.7). For Legendre polynomials $\zeta_k^{(0,0)} = 2/(2k+1)$.

Table 2.6 shows several examples of Jacobi transforms for $f(x) = 1 - x^2(3+2x^2)/5$. The examples include most of the polynomials considered in Table 2.1. The last four columns are associated with cylindrical and spherical geometry. For cylindrical problems, the

Table 2.6 Jacobi Transform Coefficients, $f(x) = 1 - x^2(3 + 2x^2)/5$

n	$P_n^{(\frac{-1}{2},\frac{-1}{2})}$	$P_n^{(0,0)}$	$P_n^{(\frac{1}{2},\frac{1}{2})}$	$P_n^{(1,1)}$	$P_n^{(1,0)}$	$P_n^{(0,1)}$	$P_n^{(0,0)}(x^2)$†	$P_n^{(1,0)}(x^2)$†	$P_n^{(0,0)}/x$	$P_n^{(1,1)}/x$
0	0.55000	0.72000	0.80000	0.84571	0.72000	0.72000	0.56667	0.73333	0	0
1	0	0	0	0	0.25143	-0.25143	-0.50000	-0.30667	0.46857	0.32381
2	-1.33333	-0.62857	-0.36000	-0.23111	-0.37714	-0.37714	-0.06667	-0.04000	0	0
3	0	0	0	0	0.04063	-0.04063	0	0	-0.41778	-0.13766
4	-0.18286	-0.09143	-0.05079	-0.03048	-0.05079	-0.05079	0	0	0	0
5	0	0	0	0	0	0	0	0	-0.05079	-0.01616

† shifted polynomials

Legendre/Gauss and Radau polynomials are used, but are expanded in terms of x^2 and are shifted to [0,1]. For spherical polynomials, shortcut type polynomials are used, but they are related to the full polynomials through Eq. (2.21). The coefficients in the table have been converted to corresponding full polynomials. The odd polynomials are made symmetric with the division by x. Some of these polynomials may seem strange, but they are the ones that give the highest accuracy quadratures for the various geometries, cf. Table 2.1.

We want to emphasis here, that we will not normally expand the solution in terms of these polynomials, but they are the ones associated with the Lagrange trial functions, i.e. those appearing in Eq. (2.4). These are also the polynomials that the residual is made orthogonal to in MWR as discussed in section 3.1.6.

A Jacobi transform can be used for interpolation by inserting the interpolant for $f(x)$. If an m^{th} degree interpolating polynomial is used, then:

$$\begin{aligned} a_k &= \frac{1}{\zeta_k^{(\alpha,\beta)}} \sum_{i=0}^{m} f(x_i) \int_{-1}^{1} \ell_i(x) P_k^{(\alpha,\beta)}(x)(1-x)^{\alpha}(1+x)^{\beta} dx \\ &\approx \frac{1}{\hat{\zeta}_k^{(\alpha,\beta)}} \sum_{i=0}^{m} f(x_i)\, W_i^* P_k^{(\alpha,\beta)}(x_i) = \sum_{i=0}^{m} Q_{ki} f(x_i) \end{aligned} \tag{2.141}$$

In the second equation, the interpolation is through quadrature points and quadrature is used to approximate the integrations. Use $\boldsymbol{x}$ and $\boldsymbol{W}^*$, base points and weights, for Jacobi-Gauss quadrature defined by Eqs. (2.8) and (2.64). The same formula can be uses for Jacobi-Gauss-Radau and Jacobi-Gauss-Lobatto quadrature which are generalizations (i.e. including both endpoints and a weight within the integrand) of the formulas described in Sections 2.4.3 and 2.4.4. The number of correct coefficients, $\boldsymbol{a}$, depends on the quadrature accuracy. For m+1 nonzero quadrature weights, the accuracy is degree $2m$+1, $2m$ and $2m$-1 for Gauss, Radau and Lobatto quadrature respectively. Since for $k \le m$, the integrand is up to degree $2m$, the coefficients are exact provided a modification is used for the Lobatto case [Shen, *et al*. (2011)]:

$$\hat{\zeta}_k^{(\alpha,\beta)} = \begin{cases} \zeta_m^{(\alpha,\beta)}\,(2m+\alpha+\beta+1)/m & \text{for } k = m \text{ Lobatto} \\ \zeta_k^{(\alpha,\beta)} & \text{otherwise} \end{cases} \tag{2.142}$$

A nodal basis with MWR interpolates through the interior and endpoints, even for Gauss points for which endpoint quadrature weights are zero. Lobatto quadrature utilizes the endpoints, so the equation is exactly what is needed to produce Legendre coefficients from values at Lobatto points, where $m = n + 1$ and $\alpha = \beta = 0$. The transformation matrix is:

$$Q_{ki} = \frac{W_i}{\hat{\zeta}_k^{(0,0)}} P_k^{(0,0)}(x_i) \tag{2.143}$$

Where $\boldsymbol{W}$ and $\boldsymbol{x}$ are the Lobatto quadrature weights and base points from Eq. (2.81).

For Gauss points Eq. (2.141) does not include the endpoints because the endpoint weights are zero. With MWR and collocation, both endpoints are always included in the interpolant. Many texts [e.g. Canuto, et al. (1988), Boyd (2000)] discuss the use of Eq. (2.143), but they fail to consider the endpoints, which are required to meet the boundary conditions. Using the endpoints and n interior points, Eq. (2.141) requires integration through degree $2n + 2$, whereas Gaussian quadrature is exact only through $2n - 1$, so the approach cannot be made to work unless more accurate integration is used.

If the problem has homogenous Dirichlet boundary conditions it is possible to expand the solution in Jacobi polynomials which can then be converted to a Legendre series using Eq. (2.42) as follows:

$$f(x) = (1 - x^2) \sum_{k=0}^{n-1} b_k P_k^{(1,1)}(x) = \sum_{k=0}^{n-1} \hat{b}_k \left(P_k(x) - P_{k+2}(x) \right) \tag{2.144}$$

where $\hat{b}_k = b_k \, 2(k+1)/(2k+3)$ and the coefficients are:

$$b_k = \frac{1}{\zeta_k^{(1,1)}} \sum_{i=1}^{n} f(x_i) \int_{-1}^{1} \ell_i(x) P_k^{(1,1)}(x) dx = \sum_{i=1}^{n} Q_{ki}^J f(x_i) \tag{2.145}$$

and from Eq.(2.7), $\zeta_k^{(1,1)} = 8(k+1)/[(2k+3)(k+2)]$. The Jacobi transform for $\alpha = \beta = 1$ requires the integrand to contain the weight function, $(1 - x^2)$. Eq. (2.145) appears to violate this requirement, but in fact the term is contained within the interpolating polynomial.

If we use the quadrature associated with the points selected, $\boldsymbol{Q}^J$ is given by:

$$Q_{ki}^J = \frac{W_i}{\hat{\zeta}_k^{(1,1)}} P_k^{(1,1)}(x_i) \tag{2.146}$$

$k = 0,\ldots,n - 1$ and $i = 1,\ldots,n$. With this approach, the integrand in Eq. (2.145) is two degrees less than the one in Eq. (2.141), a total of up to $2n$ degrees. Lobatto quadrature is exact, while Gaussian quadrature misses exact integration by one degree. However, an exact result is obtained by using the following for Gauss points:

$$\hat{\zeta}_k^{(1,1)} = \begin{cases} \zeta_k^{(1,1)} & \text{for } k < n-1 \\ \zeta_k^{(1,1)} \left(\dfrac{2k+3}{k+2}\right) & \text{for } k = n-1 \end{cases} \tag{2.147}$$

The modification above is not needed for Lobatto points, since they produce an exact result.

For convenience, let $f_i = f(x_i)$. The development using Eqs. (2.144) and (2.145) assumes homogenous boundary values, $f_0 = f_{n+1} = 0$; however, it is easy to generalize to arbitrary endpoint values by using an expansion of the form:

$$\begin{aligned} f(x) &= \frac{(1-x)}{2} f_0 + \frac{(1+x)}{2} f_{n+1} + (1-x^2) \sum_{k=0}^{n-1} b_k P_k^{(1,1)}(x) \\ &= (f_0 + f_{n+1}) \frac{P_0}{2} - (f_0 - f_{n+1}) \frac{P_1}{2} + \sum_{k=0}^{n-1} c_k (P_k(x) - P_{k+2}(x)) \end{aligned} \tag{2.148}$$

The values of the coefficients, $\boldsymbol{b}$, are determined using Eq. (2.145), with $f(x_i) - (1-x_i)f_0/2 - (1+x_i)f_{n+1}/2$ substituted for $y(x_i)$. Next, define an intermediate transform matrix for the coefficients $\boldsymbol{c}$:

$$\begin{aligned} \tilde{Q}_{k,0} &= -\left(\frac{k+1}{2k+3}\right) \sum_{i=1}^{n} Q_{ki}^J (1-x_i) \\ \tilde{Q}_{kj} &= 2\left(\frac{k+1}{2k+3}\right) Q_{kj}^J \\ \tilde{Q}_{k,n+1} &= -\left(\frac{k+1}{2k+3}\right) \sum_{i=1}^{n} Q_{ki}^J (1+x_i) \end{aligned} \tag{2.149}$$

for k = 0,…,n - 1 with $\tilde{Q}_{kj} = 0$ for $k = -2, -1, n, n+1$. The intermediate transform in Eq. (2.149) can be used to calculate the coefficients of Eq. (2.148) by summing over the boundary and interior points:

$$c_k = \sum_{i=0}^{n+1} \tilde{Q}_{ki} f(x_i) \tag{2.150}$$

The Legendre transformation matrix in Eq. (2.141) can then be completed by collecting like terms:

$$Q_{kj} = \tilde{Q}_{k-2,j} - \tilde{Q}_{kj} + \tfrac{1}{2}(\delta_{k,0} - \delta_{k,1})\delta_{j,0} + \tfrac{1}{2}(\delta_{k,0} + \delta_{k,1})\delta_{j,n+1} \tag{2.151}$$

for k = 0,…,n + 1. The transform matrix defined by Eq. (2.151) is identical to that defined by the Legendre transform in Eqs. (2.141) and (2.143).

In summary, the Legendre transform matrix, $\boldsymbol{Q}$, can be calculated quite simply for either Gauss or Lobatto interior points and the boundary points. A simple matrix multiply, Eq. (2.141), gives the coefficients of the truncated Legendre series, Eq. (1.2), which interpolates the nodal values. Transforming in the other direction is easier, since given the modal coefficients, the

nodal values are calculated by summing the coefficients times the polynomials evaluated at the nodes, i.e. evaluation of Eq. (1.2) at the nodes.

As explained by Boyd (2000), the transform matrix, $\boldsymbol{Q}$, can be substituted into any of the approximations to convert from one form to the other, i.e. modal to nodal or vice versa. For example, suppose a matrix $\boldsymbol{B}$ acts on modal coefficients. It can be converted to one for nodal values by: $\boldsymbol{B}\hat{\boldsymbol{a}} = \boldsymbol{BQy}$. As an example we convert the modal differentiation matrix given by Eq. (2.44) to a nodal matrix as follows:

$$\sum_{k=0}^{n+1} \left.\frac{dP_k(x)}{dx}\right|_{x_i} Q_{kj} = A_{ij} = \left.\frac{d\ell_j(x)}{dx}\right|_{x_i}$$

where $\boldsymbol{A}$ is the first derivative matrix given by Eq. (2.103). As an example, consider Gauss points with n = 2. Eq. (2.44) is evaluated at the endpoints ±1 and the two interior points $\pm\sqrt{1/3}$ for the first four polynomials. Post multiplication by the transformation matrix gives:

$$\begin{bmatrix} 0 & 1 & -3.00000 & 6 \\ 0 & 1 & -1.73205 & 1 \\ 0 & 1 & 1.73205 & 1 \\ 0 & 1 & 3.00000 & 6 \end{bmatrix} \begin{bmatrix} 0.00 & 0.50000 & 0.50000 & 0.00 \\ -0.20 & -0.51962 & 0.51962 & 0.20 \\ 0.50 & -0.50000 & -0.50000 & 0.50 \\ -0.30 & 0.51962 & -0.51962 & 0.30 \end{bmatrix} = \begin{bmatrix} -3.50000 & 4.09808 & -1.09808 & 0.50000 \\ -1.36603 & 0.86603 & 0.86603 & -0.36603 \\ 0.36603 & -0.86603 & -0.86603 & 1.36603 \\ -0.50000 & 1.09808 & -4.09808 & 3.50000 \end{bmatrix}$$

After multiplying by 2 to rescale from [-1,1] to [0,1], the values for $\boldsymbol{A}$ are identical to those from Eq. (2.103) which is implemented in supplied code. This example is also listed in Finlayson (1972), Table 5.5. Some additional conversions of this type are illustrated in the examples.

Some algorithms [Canuto, et al. (1988) p. 86, Boyd (2000), p. 107] require switching back and forth between the modal and nodal representations while solving some nonlinear problems. In such cases, the efficiency of the transformation can become important. In most cases the transformation matrix has special form which can be exploited to speed up the calculation. For Chebyshev trial functions, the transformation can be performed using fast Fourier transforms, which is the most efficient method for very large n, i.e. n > 20 – 100. Our preference is to use a nodal formulation which is much simpler for nonlinear problems and if large n is required then switch to a nodal finite element based method. If a problem truly benefits from use of a high order method, one could always use say 6 interior nodes per element to achieve an 8th order convergence rate!

Many texts suggest or at least imply the form of the trial functions has a major effect on the results, the example calculations illustrate that, with regard to the calculated solution, it makes absolutely no difference whether the approximation is nodal, modal or monomials. The results are always equivalent and can easily be converted from one form to another. Rounding errors are the only potential source of differences.

2.10 Monomial Transforms

Early developments of orthogonal collocation did not calculate the nodal differentiation matrices as described in Section 2.5. In the early descriptions [Villadsen (1970), Finlayson

(1972)], orthogonal polynomials were stated to be the trial functions. Despite these statements to the contrary, the approximations were developed using monomial trial functions:

$$y(x) \approx \sum_{k=0}^{n+1} \breve{a}_k x^k \tag{2.152}$$

The monomials were differentiated to develop expressions for derivatives and a transformation matrix converted the results to nodal approximations. For example, the first derivative matrix was calculated with an expression like:

$$\left.\frac{dy}{dx}\right|_{x_i} \approx \sum_{k=0}^{n+1} k\, x_i^{k-1}\, \breve{a}_k = \sum_{j=0}^{n+1} \left(\sum_{k=0}^{n+1} k\, x_i^{k-1} \breve{Q}_{jk} \right) y(x_j) = \sum_{j=0}^{n} A_{ij}\, y(x_j) \tag{2.153}$$

The other differentiation matrices and the quadrature points were calculated with a similar procedure, i.e. by differentiating or integrating the monomials followed by transformation. Following Hamming (1962), the transformation matrix, $\breve{\boldsymbol{Q}}$, was calculated by inverting the Vandermonde matrix, which is notoriously ill conditioned, so the approach is only valid for $n \lesssim 10$. We will develop a more direct procedure for calculating the transformation matrix, but one which is still subject to rounding errors for large n.

The previous section discussed the transformation between nodal and modal representations of the solution. Here we consider the analogous transformation between nodal and monomial representations. Although this monograph uses nodal approximations, the transforms to other formulations are described for completeness. If the coefficients of Eq. (2.152) are known, it is easy to determine the nodal values by direct substitution. It is more difficult to determine the coefficients from the nodal values. We need a transformation analogous to Eq. (2.141):

$$\breve{a}_\ell = \sum_{k=0}^{n+1} \breve{Q}_{k\ell} y(x_k) \tag{2.154}$$

The coefficients are those that interpolate through the nodal values, so what is needed is an expansion of the interpolating polynomial in Eqs. (2.1). Combining Eqs. (2.1) and (2.154) gives the following relationship:

$$\ell_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n+1} \frac{(x - x_j)}{(x_i - x_j)} = \sum_{k=0}^{n+1} \breve{Q}_{ik} x^k \tag{2.155}$$

The values of $\breve{\boldsymbol{Q}}$ can be calculated by expanding the continued product. The denominator is a simple calculation. The expansion of a continued product can be accomplished as follows:

$$\prod_{i=0}^{n+1}(x-x_i)=(x-x_0)\prod_{i=1}^{n+1}(x-x_i)=[x^2+(-x_0-x_1)x+x_0x_1]\prod_{i=2}^{n+1}(x-x_i)$$
$$=\left(x^k+\sum_{j=0}^{k-1}b_{jk}x^j\right)\prod_{i=k}^{n+1}(x-x_i) \qquad (2.156)$$

The polynomial in the large parenthesis is built up recursively. The equalities on the top row shows values for k = 1 and 2. When k is incremented, the coefficients are updated by $b_{i,k+1} = b_{i-1,k} + (-x_k)b_{ik}$ with $b_{kk} = 1$.

Perhaps it is easier to understand the procedure by examining some Matlab code. Referring to

Monomial Transform Calculation

```
xj = repmat(x(2:end)',n);
for i=1:n
   xj(i,1:i-1) = x(1:i-1)';
   q(i,1) = ((1.0)./prod(x(i)-xj(i,:)));
end
for k=2:n
   q(:,k) = q(:,k-1);
   for kk=k-1:-1:2
      q(:,kk) = q(:,kk-1) .- xj(:,k-1).*q(:,kk);
   end
   q(:,1) = -xj(:,k-1).*q(:,1);
end
```

the text box, `x` are the points and `n` is the total number of points. The first set of statements creates an (`n`)x(`n-1`) array, `xj`, which excludes the point corresponding to the row number and calculates the continued product for the denominator of Eq. (2.155). The second set of nested loops recursively calculates the values for each coefficient by expanding the continued product in the numerator.

We demonstrate the transform by using Eq. (2.153) to calculate the same first derivative matrix considered in the previous section, i.e. for 2 interior Gauss points. The first matrix holds the values of the derivatives $j\,x_i^{j-1}$ at the two endpoints and two interior points $(1 \pm \sqrt{1/3})/2$. The second matrix is the transpose of $\breve{\boldsymbol{Q}}$, and the result on the right-hand-side is the first derivative matrix, $\boldsymbol{A}$. The values agree with those in the furnished software and are twice those in the previous section, since those are on the interval [-1,1] and these are on [0,1].

$$\begin{bmatrix}0 & 1 & 0.0000 & 0.000\\ 0 & 1 & 0.4226 & 0.134\\ 0 & 1 & 1.5774 & 1.866\\ 0 & 1 & 2.0000 & 3.000\end{bmatrix}\begin{bmatrix}10 & 0.000 & 0.000 & 0\\ -7 & 8.196 & -2.196 & 1\\ 12 & -18.588 & 12.588 & -6\\ -6 & 10.392 & -10.392 & 6\end{bmatrix}=\begin{bmatrix}-7.000 & 8.196 & -2.196 & 1.000\\ -2.732 & 1.732 & 1.732 & -0.732\\ 0.732 & -1.732 & -1.732 & 2.732\\ -1.000 & 2.196 & -8.196 & 7.000\end{bmatrix}$$

2.11 Software

Stroud and Secrest (1966) have tabulated the roots and quadrature weights to high accuracy for numerous cases through large n. Rather than using tabular values from a book, it is more convenient to have a computer code which will produce the desired quantities when requested. The computer code could store the results internally, calculate the roots and all of the coefficients, or some combination of the two approaches.

There is some debate in the literature regarding the efficiency of fundamental calculations, especially from Chebyshev proponents. “Back in the day”, we calculated the quantities of interest and punched the results out on cards, which were read by our application codes. Some codes based on asymptotic methods, use a modernized version of this approach, i.e. values for small n are stored within the code rather than calculated. Fig. 2.22 shows the calculation of roots and weights normally requires less than a millisecond, and the other calculations are trivial. It doesn’t make sense to store coefficients in order to save computation time. Normally, the fundamental calculations make up a small portion of the total calculations required to solve a problem.

Our aim is to provide code for the fundamental calculations of this chapter in all of the target computing environments: Python, Matlab/Octave, Excel, Fortran and C++. The calculations of interest are:

1. Base points or roots, $\boldsymbol{x}$
2. Quadrature and barycentric weights, $\boldsymbol{W}$ and $\boldsymbol{W^b}$
3. Lagrange interpolating polynomials, $\boldsymbol{\ell}(x)$
4. First derivative operators, $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$
5. Laplacian operator, $\boldsymbol{B}$
6. Stiffness matrix, $\boldsymbol{C}$
7. Mass matrix for Galerkin or moments methods, $\boldsymbol{M}$
8. Jacobi and Legendre transforms, $\boldsymbol{Q}$
9. Monomial transform, $\breve{\boldsymbol{Q}}$
10. Orthogonal polynomials calculations $P_n^{(\alpha,\beta)}$
11. Polynomial differentiation, Eqs. (2.32), (2.35) and (2.38)

The codes were used to create the results described in preceding sections, while more examples are presented in the next one. As of this writing, the Fortran code is the most complete, accurate and efficient. It can calculate all the quantities above for Gauss, Lobatto, Radau and Chebyshev points. Even equal spaced or some other supplied roots can be used for experimentation. The calculations can be performed for both nonsymmetric problems and symmetric ones in planar, cylindrical and spherical geometry. We plan to make the same calculations available in the other languages. However, only recurrence calculations will be implemented directly. For large problems, the codes will link to the Fortran modules. The online documentation should be checked to determine the status for each language.

The current Fortran code relies on three modules in three files, *OrthPoly.f90*, *OrthCheb.f90* and, *OCC.f90*. The points and weights and other fundamental calculations are carried out by *OrthPoly* and *OrthCheb*, while the *OCC* code calculates most of the other quantities, differentiation matrices etc. A similar approach is taken with the other languages. To solve problems, one will normally interface with the *OCC* code. The *OrthPoly* code is available for more fundamental calculations, e.g. Jacobi Transforms of sections 2.9. The Jacobi roots and the weights are determined using the noniterative high order methods described in Sections 2.3 and 2.4. For small problems, $n < 40$, recurrence calculations are used, while asymptotic calculations are incorporated for larger n. The code has been tested for up to 10^6 points. The accurate and efficient methods used are not available in software from elsewhere, except perhaps for limited cases.

The code is written in an objected oriented style. To use it you first initialize or instantiate a *collocation object* by specifying the value of n, the type of points and the geometry. Once initialized the quantities desired are requested using array valued functions. The boxes below give examples for Python and Fortran. They are remarkably similar. The first lines indicate the modules to use. The "only" option is not mandatory for Fortran, but it is a good practice to explicitly state which functions you want to use. Both languages support renaming or aliasing of function names. Explicit importation is also possible with Python (second line), but it is not necessary because the *obj.function* syntax makes the source of the function more obvious. After initialization, the two examples get the points, quadrature weights, and first derivative, stiffness and mass matrices. Of course, with Fortran there are additional statements declaring the array dimensions. For each language there are test codes provided which illustrates use of the most common features.

The various codes provide a toolkit for solving problems with not only collocation but also other MWR in a wide range of languages and calculation systems. Some of the examples demonstrate use of these codes with Galerkin and moments methods. A spreadsheet is not an ideal platform for solving differential equations, but it is useful for checking the calculations for correctness, comparing results, etc. The example codes create tab delimited text files for easy spreadsheet importation.

Every effort was made to demonstrate good programming style for the code in this project. However, I will admit to occasionally doing what is expedient rather than

Python Example

```
import occ
from occ import OrthColloc
oc = OrthColloc(n,ptyp,geom)
x = oc.Xpoints()
w = oc.WeightQ()
A = oc.Deriv1()
C = oc.Stiffness()
D = oc.MassMatrix()
```

Fortran Example

```
Use OrthogonalColloc, Only :  &
  ColDataType, Initialize, Xpoints, &
  WeightQ, Deriv1, Stiffness, MassMatrix
Type(ColDataType) :: OC
call Initialize(OC,n,typ,geom)
x = Xpoints(OC)
w = WeightQ(OC)
A = Deriv1(OC)
C = Stiffness(OC)
D = MassMatrix(OC)
```

what is correct. My preferences are influenced by my experiences and do not always agree with others. Also, since I have coded in several languages, my coding style tends to be a composite, the Python code may not look "*pythonic*" and the C++ code may look something like Fortran. I prefer code which is concise, compact, and without a lot of extraneous white space. I see nothing wrong with multiple short statements on the same line. Matlab programmers tend to like a compact style, whereas some Fortran programmers like lots of white space. The variable names used in calculations tend to be short, so the code looks more like the equations in this monograph. Longer names are fine for procedures and flow control variables, etc. Unlike most Fortran programmers, I like to use functions, often array valued, rather than subroutines (in Fortran) or void functions (in C++). This seems to be the trend. Unfortunately, only one return is possible with Fortran and C++.

I see nothing wrong with small *include* files, although the feature is absent or obscure in some languages and frowned upon in others. Where would C++ be without include files? Modules are usually recommended for Fortran, and they are used extensively for calculations. The modules use the object-oriented programming concept of *encapsulation*, data and functions are packaged together. They are usually *private* by default with *public* access only for specified items. However, Fortran modules are compiled, which is an unnecessary complication when a small include file suffices. Anyone who has worked much with Fortran modules has accidently created a project with a circular reference, which goes unnoticed until a complete rebuild is attempted.

I try to avoid a lot of *if* statements. Often functions can be used instead, e.g. *a = max(0,a)* is preferable to *if(a < 0)a = 0*. When applicable, I prefer *case* constructs to long *if-elseif-else* constructs. Indexing can often be used instead of *if* constructs, e.g. if you need a different "value" depending on a "method", you might have something like *value = value_list(method)*, where "value_list" is a parameter list.[2] See the code in section 2.3.1 for an example.

Array syntax is preferred rather than loops, especially for interpreted languages. Loops can be faster for compiles languages, but the code is larger. When a loop is required, I tend to use only one type of looping construct. Some upper limit on the loop count is usually needed to avoid an infinite loop. The ideal basic loop type is *do forever*, which can be modified by an index and count and/or *break* and *continue* options, which can be conditional. If that basic looping construct is available, why are *while* and *until* loops needed? Everyone has a finite memory for recalling the syntax of different constructs.

For languages which support different levels of precision, I have tried to make precision changes easy. For example, with Fortran a single value of *float* is used to define the precision (called the *kind* in Fortran). This parameter is in an include file (see include file discussion above). This feature made it easy to compare the accuracy of different calculations, such as those in Figs. 2.11, 2.12, 2.21 and those in Appendix A.1. I shudder when I see code with hardwired *"kind = 8"* throughout, making precision changes a daunting and error prone task.

[2] I once hired a computer science student to help with our coding. He created an *if-elseif-else* construct that went for several hundred of lines. With indexing the same result was achieved with about ten lines of code.

Also, the *"8"* is implementation dependent and does not necessarily mean 8 bytes. I also like the name *float* or maybe *flt* rather than *sp* or *dp*. *dp* looks like it stands for double precision, which would be confusing if you redefine it to convert the program to single or quad precision. *float* is generic and could be single, double or quad precision, depending on how it is defined. It should be defined something like *float = selected_real_kind(14)*, which specifies at least 14 digits of accuracy if it is available.

The example calculations require additional software to implement solutions, e.g. solution of linear algebraic equations, eigenvalues, etc. Some languages like Python and Matlab have code for these tasks which is more or less built in. They have used wrappers around tried and true public domain software like LAPack. For C++ and Fortran external libraries must be used, but they are easily accessed. For Fortran there is LAPack and for C++ there is CLAPack and the GNU Scientific Library, GSL. However, the interface to LAPack routines is ugly, requiring numerous arguments including workspace, etc. I have built wrappers to simplify and modernize these tasks, so a system of equations is solved by *y = LUSolve(A,b)*. The technique of encapsulation by using a wrapper in a Fortran module or C++ class is a method everyone should know. There is no reason to recode a task when proven code is readily available in the public domain.

Nowadays it is relatively easy to mix languages using interlanguage calling, common libraries, and conversion utilities like *f2c, f2f90, f2matlab* and *f2py*. For Python, *f2py* is recommended by many, but I prefer the simplicity of *ctypes*. With these tools one can incorporate tried and true public domain code using many different languages, or you can design a program using multiple languages. Fortran is good for doing calculations efficiently, C++ or Python is better for user interfaces and higher-level administrative tasks. One can use dynamic link library for Excel and Mex files for Matlab. For calculations with large n, I have used a dynamic link library for Python and C++.

Encapsulation is one of the most important concepts in object-oriented programming. Although the example problems are small, the use of protection for data and internal calculations is demonstrated. The codes also show how to make distinct interfaces between program components for more reliable programs.

2.12 Example Calculations

For each of the language systems implemented one or more simple test programs are provided to demonstrate the call syntax and use of code to calculate the quantities described in this chapter. The code and a simple reference manual can be freely downloaded.

This section contains excerpts from the test codes and their results. The example shows some of the basic calculations and demonstrate their use for interpolation, differentiation and integration. For a more detailed understanding, one should experiment with the test codes. The example code below is in Matlab/Octave. Compact code like this can be produced by Python and Fortran also. For C++ loops are required to produce the same results, so many more statements are needed. The C++ code and test spreadsheet perform these same calculations.

Matlab Example Code

```
[x,w,A,An] = OCsym(n,meth,geom);
[B,C] = OCBCcoef(w,A,An,symm);
D = MassMatrix(x);
Df = MassMatrix(x,@MassFunc,2);
Li = Lcoef(xi',x);
Q = Lpoly(x,symm);
for i=1:nt
   Lp(:,i) = polyval(fliplr(Q(i,:)),xi);
end
[fx,fc,dfc] = Expfunc(x,symm,geom);
[fx,f,df] = Expfunc(xi,symm,geom);
fi = Li*fc;
dfa = A*fc;
fw = w'*fc;
```

In the Matlab/Octave code, the first line defines the size and type of problem and retrieves the points, quadrature weights, and the first derivative matrices for symmetric and antisymmetric quantities - ***x, w, A, An***. The second statement produces the Laplacian and stiffness matrices, ***B*** and ***C***. Two versions of the *MassMatrix* function are demonstrated, one assumes no function in Eq. (2.121), while the other gives the name of the function to use. An array of values, *xi* = (0, 0.05, 0.10….), has been set up for tabulation and plotting values. The *Lcoef* function calculates values of the interpolating polynomials, while *Lpoly* calculates the monomial coefficients of the interpolating polynomials, Eq. (2.155). The routine *ExpFunc* calculates the values, derivative and integral of $exp(-5x^2)$ at the collocation points and *xi*. The last three statements interpolate the function and approximate its first derivative at the collocation points, and then calculates an approximate integral using the quadrature formula. All of these calculations are with native Matlab code.

The code above is for a symmetric problem, and some plots are shown with cylindrical and spherical geometry. First, Lagrange interpolating polynomials, *Li*, are shown for four cylindrical Gauss points in

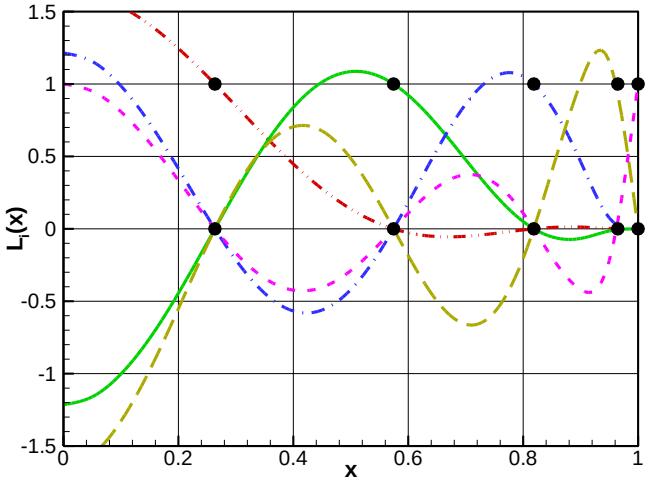


Fig. 2.29 Lagrange polynomials, cylindrical geometry, n = 4

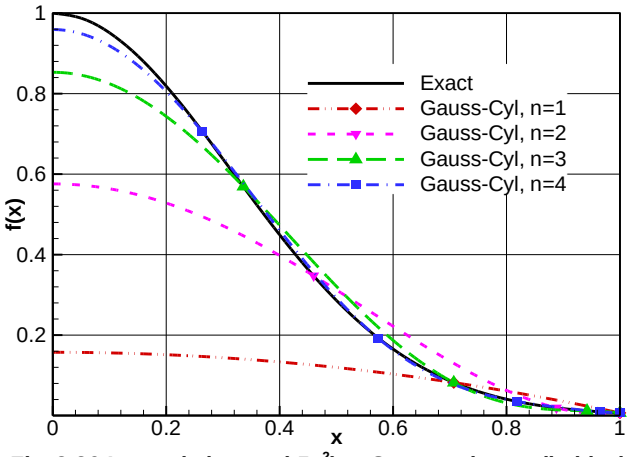


Fig. 2.30 Interpolation exp(-5x²) at Gauss points, cylindrical

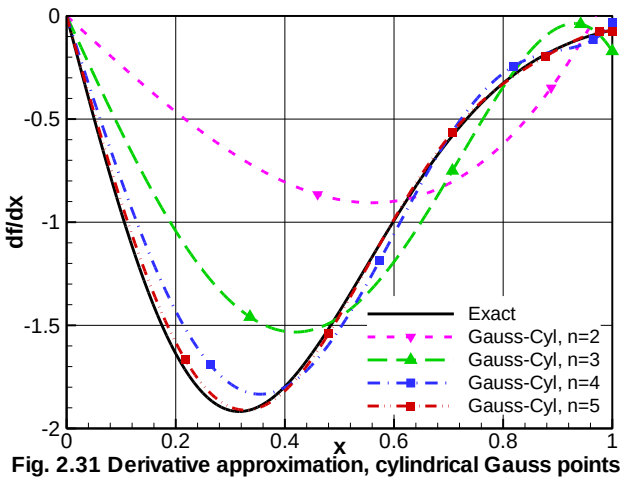


Fig. 2.31 Derivative approximation, cylindrical Gauss points

Fig. 2.29. Then the approximation of the exponential function is in *f* and *fi*, and its first derivative are, *df* and *dfa*, are shown for cylindrical Gauss points in Figs. 2.30 and 2.31. The quadrature weights, *w*, are shown in Figs. 2.32 and 2.33 for four points in cylindrical and spherical geometry, respectively. The error in the integrals from 0 to 1 of $exp(-5x^2)x$ and $exp(-5x^2)x^2$ are shown in Figs. 2.34 and 2.35, i.e. *(fx-fw)/fx* for cylindrical and spherical geometry.

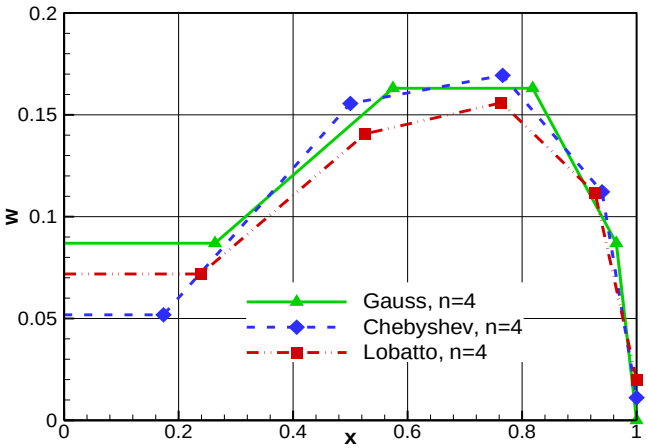


Fig. 2.32 Quadrature points and weights, cylindrical n = 4

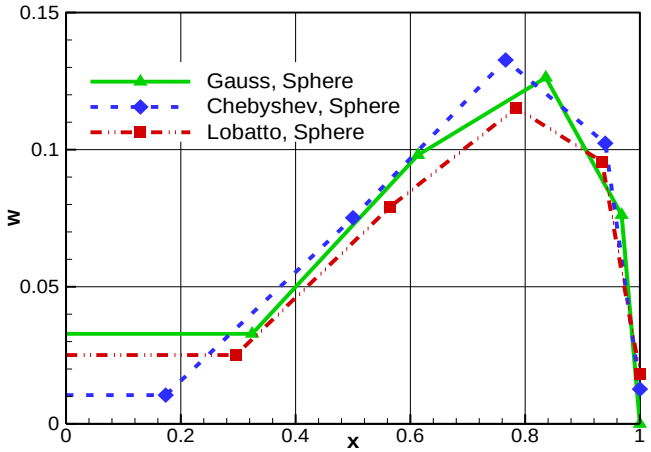


Fig. 2.33 Quadrature points and weights, spherical n = 4

When examining the Figs. 2.32 and 2.33, note that Chebyshev points are not shifted toward x = 1 as they are for the other points (due to β in the polynomial weight, see Eq. (2.7) and Table 2.1). The Chebyshev points are not optimal for integration, which is apparent in Figs. 2.34 and 2.35. The approximate integral converges with $(n)log(n)$, so for large n all give exponential convergence. However, the convergence rate is slower for Chebyshev points by about a factor of 2. Chebyshev points are optimal for interpolation, while Gauss and Lobatto points are optimal for integration.

For a more challenging interpolation problem, consider the Runge function:

$$f(x) = \frac{1}{1 + 25x^2} \tag{2.157}$$

Fig. 2.36 shows the results for interpolation with equally spaced points, while Fig. 2.37 shows results for interpolation at Gauss points. Since the problem is symmetric, only the right half is

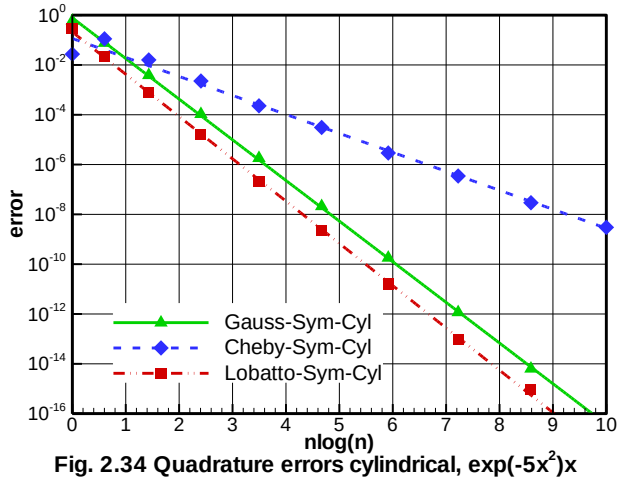


Fig. 2.34 Quadrature errors cylindrical, $\exp(-5x^2)x$

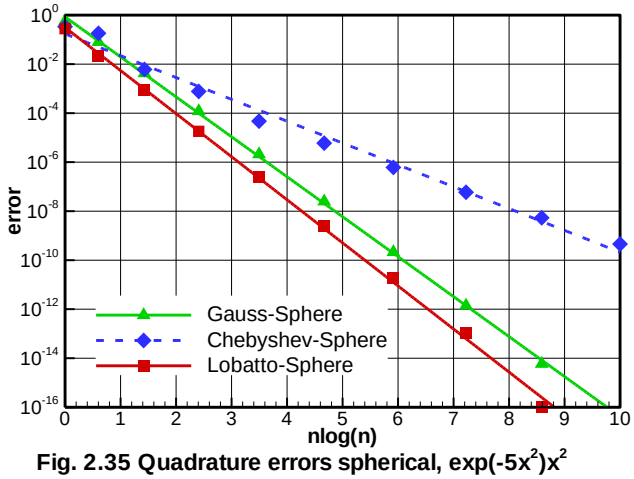


Fig. 2.35 Quadrature errors spherical, $\exp(-5x^2)x^2$

displayed. Fig. 2.36 shows the classic divergence difficulties first considered by Runge (1901). A greater number of points leads to larger excursions near the endpoints. Chebyshev points are normally the best for interpolation; however, the goal here is not interpolation, but the approximation of differential equations. Nevertheless, any of the Jacobi polynomials converge uniformly as shown in Fig. 2.37. Also, divergence like that shown in Fig. 2.36 also occurs with equally spaced collocation, which is nicely illustrated in the review article by Bert and Malik (1996).

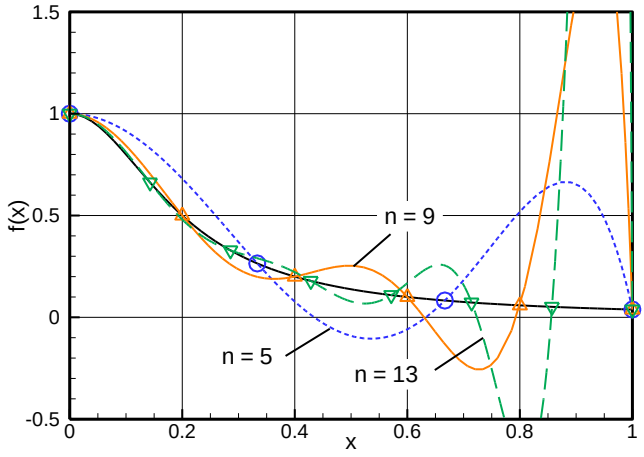


Fig. 2.36 Interpolation of the Runge function, equally spaced

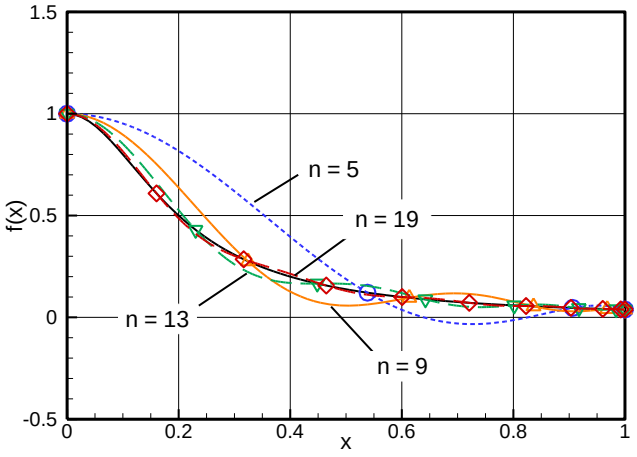


Fig. 2.37 Interpolation of the Runge function, Gauss points

3. Boundary Value Problems

In this chapter boundary value problems are treated with Methods of Weighted Residuals (MWR). The problems are solved with Galerkin, moments and collocation methods. Examination of the Galerkin method and method of moments shows the best ways to formulate the collocation method, i.e. orthogonal collocation (OC), pseudospectral (PS) and differential quadrature (DQ) methods. With a proper formulation, collocation methods produce results almost as good as the Galerkin method and method of moments, but with less complexity, achieving the best of both worlds. The methods are developed around the examples rather than an abstract problem. Each section considers an example problem: 3.1 a diffusion or conduction problem with source (Helmholtz equation) and 3.2 a coupled convective heat and mass transfer problem, a nonisothermal chemical reactor with cooling and axial dispersion.

3.1 Diffusion/Conduction with Source

Consider reaction and diffusion in a porous slab, also called the catalyst particle problem or catalyst pellet problem. This differential equation is one of the most ubiquitous ones in engineering and physics. It is the Helmholtz equation in one dimension and is analogous to heat conduction in a slab with a heat source which is dependent on temperature and position. If the source, r, is constant, the equations describe laminar flow between parallel plates. It also describes the position of an elastic string, where r characterizes the load. The governing equations are:

$$\frac{d^2y}{dx^2} + r(x,y) = 0 \tag{3.1}$$

where $r(x,y) = 4\varphi^2 \hat{r}(x,y)$ and y is the fractional conversion, 0 for no reaction and 1 for complete reaction. φ is called the Thiele modulus. The average value of $\hat{r}(x,0)$ is 1 by definition. The boundary conditions are either first kind, Dirichlet, boundary conditions:

$$y(0) = y(1) = 0 \tag{3.1a}$$

or third kind, Robin, boundary conditions:

$$\left.\frac{dy}{dx}\right|_{x=0} = 2Bi_0\, y(0) \quad \text{and} \quad -\left.\frac{dy}{dx}\right|_{x=1} = 2Bi_1\, y(1) \tag{3.1b}$$

In Eq. (3.1b), the Biot numbers, Bi, account for an external transfer resistance. As Bi tends to infinity, Eq. (3.1b) reduces to Eq. (3.1a). This problem is often symmetric about the centerline, but here we allow for a nonsymmetric profile. Symmetric problems in various geometries can be efficiently treated using polynomials in x^2 as discussed in section 3.1.5. The Thiele modulus and Biot numbers are defined using the half thickness as the characteristic length, so the factors of 4 and 2 are required when the full slab is considered.

This problem has been heavily studied in chemical engineering, including various geometries and highly nonlinear source functions which lead to multiple solutions, see 3.1.5. For a more

thorough discussion see Villadsen and Michelsen (1978), Rawlings and Ekerdt (2015) and references therein.

For a simple k^{th} order reaction with the reactivity independent of position, the source term is:

$$\hat{r}(x,y) = (1-y)^k \tag{3.2}$$

For a first order reaction, k = 1, the analytical solution is:

$$y = 1 - \frac{\cosh\left[\varphi(2x-1)\right]}{\cosh(\varphi) + \varphi\sinh(\varphi)/Bi} \tag{3.3}$$

The quantity of interest from the solution is called the effectiveness factor, η, which gives the overall rate of reaction relative to that with no diffusional resistance:

$$\eta = \frac{\int_0^1 r(x,y)dx}{\int_0^1 r(x,0)dx} = \int_0^1 \hat{r}(x,y)dx \tag{3.4}$$

The effectiveness factor is basically a normalized boundary flux, since the divergence theorem in one dimension gives:

$$-\left.\frac{dy}{dx}\right|_0^1 = 4\varphi^2 \int_0^1 \hat{r}(x,y)dx = 4\varphi^2\eta \tag{3.5}$$

Eq. (3.5) is given the generic name *average energy equation*, since in a heat transfer setting it is an overall energy balance and the right-hand side is the average energy generated. Using the analytical solution, Eq. (3.3), this normalized flux is:

$$\eta = \frac{1}{\varphi[\coth(\varphi) + \varphi/Bi]} \tag{3.6}$$

η is approximately one for φ < ½ and becomes asymptotic when φ > 2. The asymptotic state corresponds to a condition where all the reactants are consumed and the conversion, y, approaches one at the center.

3.1.1 Orthogonal Collocation Method

To solve the problem using a Method of Weighted residuals (MWR), we start with a trial solution composed of either modal, Eq. (1.2) or (1.21), or nodal, Eq. (1.3) or (2.1), or even monomials, Eq. (1.20), trial functions. Modal trial functions, considered in section 3.1.7, are popular with spectral or pseudospectral applications, while nodal formulations are normally used with orthogonal collocation or differential quadrature. Many early developments of orthogonal collocation state that the trial functions are orthogonal polynomials, but then use monomials and by use of a monomial transform the problem is reformulated in terms of nodal values. All of the approximations are equivalent, so the choice of trial function form is primarily a matter of convenience. Transforms, see sections 2.9 and 2.10, can be used to convert from one representation to another. We prefer to dispense with transformations and develop nodal methods directly using the trial solution:

$$y = \sum_{i=0}^{n+1} y(x_i)\ell_i(x) \tag{3.7}$$

The interpolation points include the endpoints, $x_0 = 0$ and $x_{n+1} = 1$, while the interior points are the roots of an orthogonal polynomial, usually either the roots of Chebyshev polynomials (of the 2nd kind), the roots of Legendre polynomials (Gauss points) or the base points of Lobatto quadrature. These roots are often called Chebyshev-Gauss-Lobatto (CGL), Legendre-Gauss (LG) and Legendre-Gauss-Lobatto (LGL) points, respectively. The base points of Radau (LGR) quadrature are used occasionally.

Many authors seem almost paranoid about the boundary points when used with Gauss points in the interior, because the associated quadrature weights are zero. They state that nonzero weights are needed on the boundary to help meet the boundary conditions. There is never any justification for these statements, and, in fact, we will show that nonzero boundary weights can be a detriment rather than an asset for meeting the boundary conditions.

The residual is formed by substitution of the approximate solution into the equation:

$$\sum_{i=0}^{n+1} y(x_i)\frac{d^2\ell_i}{dx^2} + r\left(x, \sum_{i=0}^{n+1} \ell_i(x)y(x_i)\right) = R(x, \boldsymbol{y}) \tag{3.8}$$

Where $\boldsymbol{y}$ is the vector of nodal values $y(x_i)$. With the collocation method the residual is set to zero at the interior collocation points, x_j, $j = 1, \ldots, n$. Since $\ell_i(x_j) = \delta_{ij}$ (the dirac delta function), the resulting equation simplifies to:

$$\sum_{i=0}^{n+1} y(x_i)\left.\frac{d^2\ell_i}{dx^2}\right|_{x_j} + r\left(x_j, y(x_j)\right) = 0 \tag{3.9}$$

By defining $\boldsymbol{B}$ as indicated below and letting $y_i = y(x_i)$ the equation simplifies to:

$$\sum_{i=0}^{n+1} B_{ji}y_i + r(x_j, y_j) = 0 \tag{3.10}$$

The boundary conditions provide two additional conditions, either first kind, Dirichlet, boundary conditions:

$$y_0 = y_{n+1} = 0$$

or third kind, Robin, boundary conditions:

$$\sum_{i=0}^{n+1} A_{0,i}y_i = 2Bi_0\, y_0 \quad \text{and} \quad -\sum_{i=0}^{n+1} A_{n+1,i}y_i = 2Bi_1\, y_{n+1} \tag{3.11}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are differentiation matrices, see section 2.5.:

$$A_{ji} = \left.\frac{d\ell_i}{dx}\right|_{x_j} \quad \text{and} \quad B_{ji} = \left.\frac{d^2\ell_i}{dx^2}\right|_{x_j}$$

For Dirichlet conditions, Eq. (3.1a), the boundary values (whether zero or finite) are simply substituted into Eq. (3.10). For the third kind conditions, Eq. (3.1b), most texts recommend that the boundary condition be satisfied exactly as in Eq. (3.11) [Finlayson (1972), p. 101; Villadsen and Michelsen (1978), p. 137; Bert and Malik (1996); Belomo (1997); Trefethen (2000), p. 137; Boyd (2000), p. 111, Peyret (2002), p. 59]. Note that this list includes popular references in the orthogonal collocation, pseudospectral and differential quadrature literature. When boundary condition is satisfied exactly, it is called *boundary collocation* or a *strong* treatment. The examples in this chapter and the next give a detailed examination of boundary condition treatment. A method superior to Eq. (3.11) is demonstrated.

To solve the problem, we need only the collocation points, $\boldsymbol{x}$, i.e. the roots of the orthogonal polynomial, and the differentiation matrices, $\boldsymbol{A}$ and $\boldsymbol{B}$, the derivatives of the Lagrange interpolating polynomials. We also need the quadrature weights, $\boldsymbol{W}$, for approximating integrals, e.g. Eq. (3.4). As described in chapter 2, all of these quantities can be calculated from the collocation points. Software is supplied to perform the calculations.

Eqs. (3.10) together with the boundary conditions are a set of algebraic equations. After substitution of the boundary values for Dirichlet conditions, a set of n equations are left, which are usually nonlinear. Several versions of this problem are considered, first with constant coefficient, Dirichlet, then Robin boundary conditions. Then, a linear problem with variable coefficient is considered. Finally, nonlinear symmetric problems are considered in section 3.1.5.

The nonlinear reaction terms appear only on the diagonal, which simplifies the calculations. We shall see that for a full moments or Galerkin method, the reaction terms are distributed throughout the matrix.

Linear Source, Constant Coefficients, Dirichlet B.C.

Fig. 3.1 shows solutions of Eqs. (3.1) and (3.1a) for a first order reaction and φ = 5, i.e. Eq. (3.2) with k = 1. Approximate solutions are shown with n = 4 for collocation at Lobatto, Gauss and Chebyshev points. With this relatively high reaction rate most of the reaction occurs near the boundary. The fifth order polynomial can only approximate the sharp profile by oscillating about the exact solution. Actually, since this problem is symmetric about x = 0.5, the coefficient of the fifth order term is zero. Later, we will look at efficient methods to exploit symmetry. Clearly, for this example, Lobatto points produce a more accurate solution followed in order by Chebyshev and Gauss points.

Fig. 3.2 shows the L_2 error norms versus n for the three choices of points. A plot of the L_1 error norm looks very similar. All methods show the typical exponential convergence with Lobatto points giving slightly better results. The error with Chebyshev and Gauss points averages 1.3

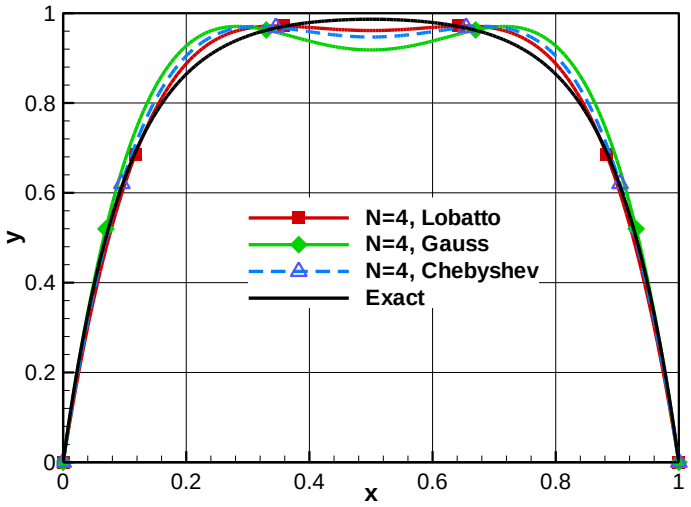


Fig. 3.1 Solutions of Eq. (3.1), Dirichlet, φ = 5, k = 1

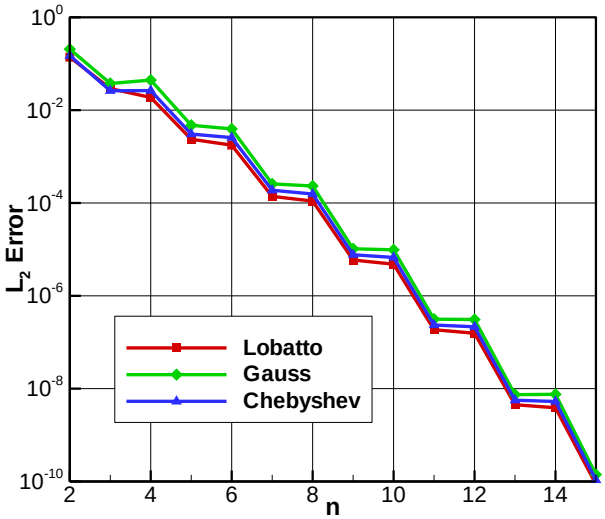


Fig 3.2 L_2 Error Norm, Dirichlet, φ = 5, k = 1

and 1.9 times that with Lobatto points, which is relatively small since the error decreases almost an order of magnitude with each increment of n.

Using Eq. (3.6), the normalized boundary flux, η = 0.199983. Numerical quadrature can be used to calculate this value from the numerical solutions using Eq. (3.4):

$$\eta = \sum_{i=0}^{n+1} W_i \, \hat{r}(x_i, y_i) \tag{3.12}$$

For the cases in Fig. 3.1 with n = 4, the calculated fluxes are in error by 0.8, -3.1 and -4.3 percent for Lobatto, Chebyshev and Gauss points when the flux is calculated by Eq. (3.12). The same quantity can be approximated using Eq. (3.5). The derivatives are given by:

$$\left.\frac{dy}{dx}\right|_{x=0} = \sum_{i=0}^{n+1} A_{0i}\, y_i \quad \text{and} \quad \left.\frac{dy}{dx}\right|_{x=1} = \sum_{i=0}^{n+1} A_{n+1,i}\, y_i \tag{3.13}$$

Using Eq. (3.13), the derivatives of the solution at the boundaries are in error by -14.3, -10.2 and -4.3 percent for Lobatto, Chebyshev and Gauss points. These errors are much larger and the relative accuracy of the methods is a complete reversal of that found with Eq. (3.12). Only Gauss points give the same result with either method of flux calculation. Shortly, we will explain why this is so. For the other two methods, integration gives a far more accurate result.

Fig. 3.3 shows the flux errors for increasing n. Relative to Fig. 3.2, this graph shows a much greater difference between the

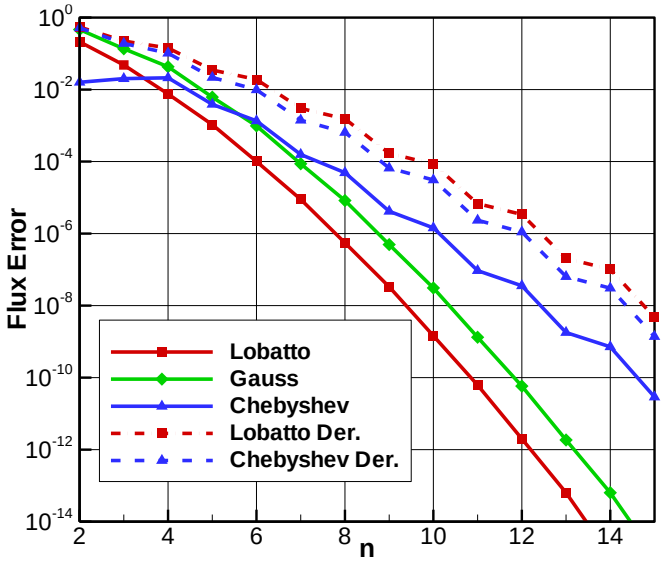


Fig 3.3 Flux Error, Dirichlet φ = 5, k = 1

methods. For n < 4, the error with all three methods is relatively large, while for n > 14, some of the results are affected by rounding errors. Engineering accuracy is obtained with 4 to 8 points depending on the method and accuracy required. In this range, the errors with Gauss and Chebyshev points are similar, while Lobatto points give somewhat greater accuracy. The convergence rate with Gauss points is much greater than with Chebyshev points, so Gauss points prevail for n > 6.

Since Figs. 3.2 and 3.3 are so different let us take a closer look at these measures of the error. Given the exact solution, y^*, the L_p error norm is a measure of the error in the internal profile:

$$\epsilon_p = \left[\int_0^1 |y - y^*|^p dx\right]^{\frac{1}{p}} \tag{3.14}$$

The error in the normalized flux for this linear source function is:

$$\epsilon_\eta = \left|\int_0^1 (y - y^*) dx\right| \tag{3.15}$$

These two error measures look similar, especially when p = 1. However, upon closer examination, it is clearly possible to achieve an exact flux with an imperfect solution which oscillates about the exact solution but in a way that gives the correct solution on average. To gain a better understanding of the nature of the error, Figs. 3.4 - 3.7 show the error, i.e. $y - y^*$, and the residual, R in Eq. (1.3) for several cases.

In Figs. 3.4 and 3.6 the residuals are, of course, zero at the collocation points. We note that the residual functions look very similar to orthogonal polynomials, Figs. 2.1 and 2.2. In fact, for this problem they are. The nature of the residual functions are discussed in section 3.1.6. We notice that Lobatto points produce the largest values of the residual (at the boundaries), but the smallest and most uniform distribution of the profile errors. Gauss points produce the most uniform residual errors, but the largest and least uniform profile errors. This result supports the claim that Chebyshev polynomials of the first kind may be best for interpolation but are not good for MWR since they will likely produce even larger profile errors.

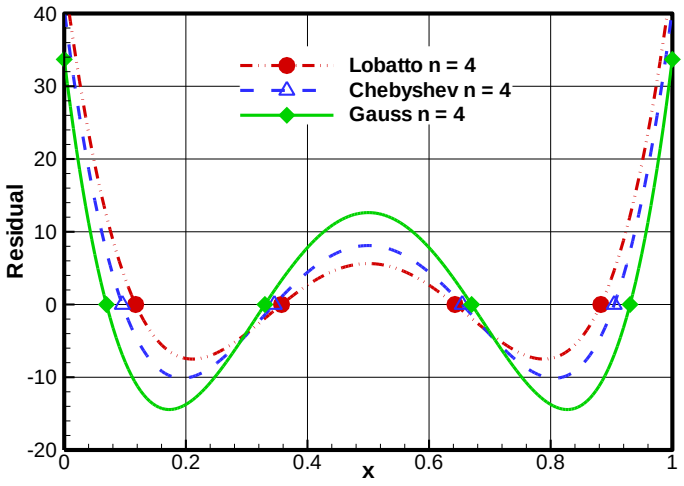


Fig. 3.4 Residual for n = 4, Dirichlet φ = 5, k = 1

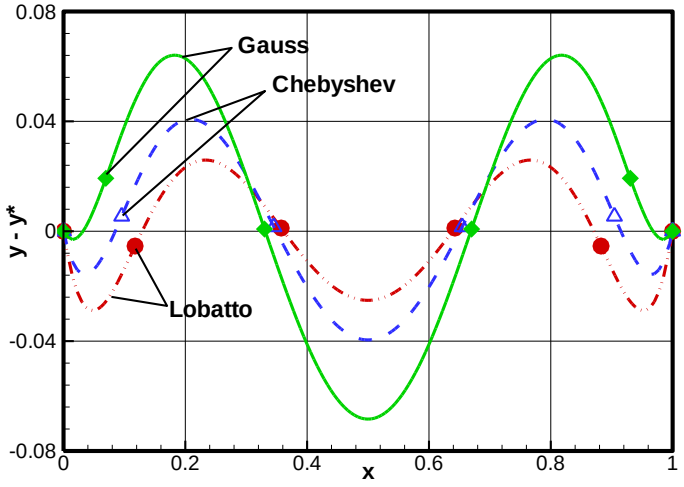


Fig. 3.5 Error in y for n = 4, Dirichlet φ = 5, k = 1

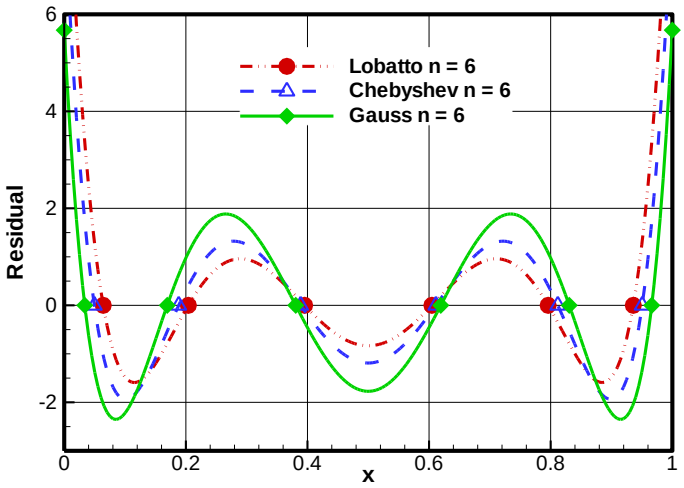


Fig. 3.6 Residual for n = 6, Dirichlet φ = 5, k = 1

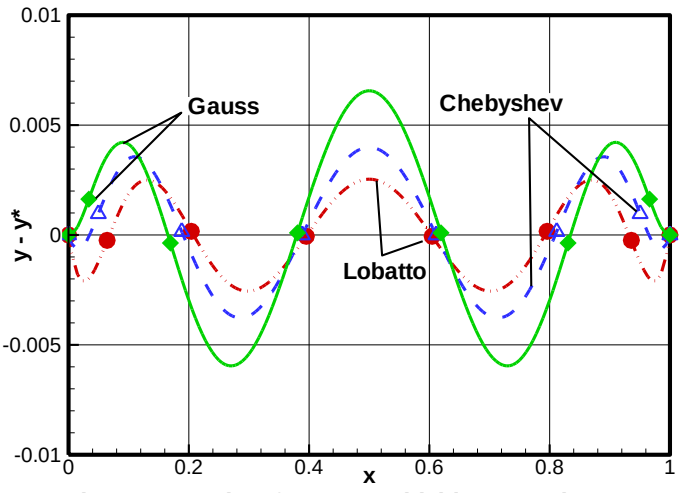


Fig. 3.7 Error in y for n = 6, Dirichlet φ = 5, k = 1

We also note that the errors shown in Figs. 3.5 and 3.7 are close to zero at the collocation points and the positive and negative deviations from zero are nearly balanced, so it appears reasonable that the error calculated by Eq. (3.15) should be small. However, all of the points appear to exhibit this behavior, so why do the solutions converge more slowly with Chebyshev points? A simple experiment sheds some light on this question.

There are two potential sources of error in the flux: (1) errors in the profile (Figs. 3.5 and 3.6) and (2) errors in the numerical integration of the profiles. If the analytical solution, Eq. (3.3), is integrated numerically using the quadrature formulas, Eq. (3.12), the error due exclusively to the numerical integration is obtained. Fig. 3.8 shows the quadrature errors together with the total flux errors from both sources of error. By examination of Fig. 3.8 we see that the errors with Chebyshev points (Clenshaw-Curtis quadrature) do not improve as much as the others. For large n and Gauss or Lobatto points, the overall error from both sources of error are less than the integration errors alone for Chebyshev points. It appears that the greater accuracy of Gauss and Lobatto quadrature is one reason for the difference. The nature of the residual error is discussed more fully in section 3.1.6.

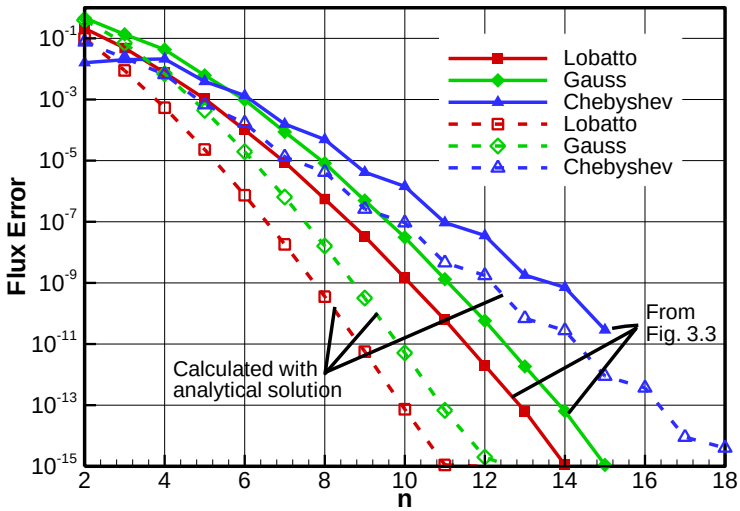


Fig 3.8 Flux error by quadrature - approximate and exact solutions, Dirichlet φ = 5, k = 1

Villadsen and Michelsen (1978) (p. 85) also noted that other methods may produce similar profile errors as we find here (see Fig. 3.2), but the moments and Galerkin methods do an especially good job of balancing the error, so that the flux calculation is very accurate. It is fortunate that Gauss and Lobatto points have this property, because the flux is the most

important result of the calculation. As we shall see, these two methods have equivalence to moments and Galerkin methods, respectively.

There are several disparate studies which discuss convergence properties of these methods. A proof of convergence for collocation was first given by Karpilovskaya (1953,1963) and is discussed by Kantorovich and Akilov (1964). Error bounds for nonlinear problems are given by Ferguson and Finlayson (1972), while an explicit error expression for the linear problem is given in Michelsen and Villadsen (1981). Examples of other early work on convergence analysis are in Gottlieb and Orzag (1977) and Canuto and Quarteroni (1981). More recent work is summarized in many of the reference books cited in chapter 1 [e.g. Canuto, et al. (2006)].

For Lobatto points, Zhang (2005) shows that errors at the interior nodes are smaller than the overall error by one degree and more accurate derivatives can be calculated at the $n + 1$ Gauss points that lie between the Lobatto points. However, the improved accuracy at the interior nodes is not great enough to explain the differences observed in Fig. 3.3, which shows exceptional accuracy at the boundary. A few studies seem to address this issue [Lanczos (1973), El-Daou and Ortiz (1992), Namasivayam and Ortiz (1993)]. These studies, based on the tau method, analyzed Legendre (Gauss) and Chebyshev methods and found greater accuracy with the Legendre case. Although a different problem was considered, a similar approach could be used for the current problem and for the Lobatto case.

The greater observed convergence rate of these global methods is related to the long known *superconvergence* property of some finite element methods [see Křížek and Neittaanmäki (1998)]. The superconvergence phenomenon is one where the solution at certain points converges at a faster rate than the overall solution. Chapter 6 on finite element methods discusses this phenomenon, which also occurs for collocation finite element methods. Since the global methods considered here are but a single element of a finite element procedure, it is likely the exceptional accuracy of the flux calculations in Fig. 3.3 is a related phenomenon.

The error curves in Figs. 3.2 and 3.3 converge at a supergeometric rate, i.e. with $(n)log(n)$, but are plotted versus $log(n)$ as is customary. Lobatto points give the best convergence rate. Gauss points converge at the same rate, but the errors are about 10 to 15 times larger, equivalent to about one increment of n. The convergence rate with Chebyshev points is roughly half that with Gauss or Lobatto points. If derivatives are used to calculate the flux, Eq. (3.13), the results are poor with Lobatto or Chebyshev points. Gauss points produce the same result regardless of calculation method. Since all the methods give exponential convergence and the L_p error norms are similar, one could easily be misled by an incomplete comparison. For this problem, significant differences show up only when fluxes are compared. The problem in section 3.2 shows differences in the internal profile errors as well. One of the advantages of orthogonal collocation or pseudospectral methods is that virtually exact solutions are feasible. The method with Lobatto points is superior for achieving high accuracy for this example, but only if fluxes are accurately calculated.

Linear Source, Constant Coefficients, Third Kind B.C.

Now, consider the same problem as above, but with the third kind or Robin boundary conditions, Eq. (3.1b). It is clear from Eqs. (3.3) and (3.5) that φ/Bi gives the relative importance of internal and external resistance. The shape of the profile is not changed, but the boundary value is scaled up to account for the external resistance. We present calculations here with φ = 5 as above and Bi = 10. For these conditions both are important, but the external resistance is less important than the internal resistance. Other values of Bi are considered to determine its effect on the results.

Most texts recommend that *boundary collocation,* Eq. (3.11), or an equivalent method, be used to approximate the boundary condition. Given that Fig. 3.3 shows poor accuracy of derivatives (Lobatto and Chebyshev) for calculating fluxes, one might question the suitability of this approach for all but Gauss points.

Fig. 3.9 compares the profiles for solutions calculated with n = 4. Note that the boundary value of y is about 0.3, indicating roughly 30% of the resistence is external. The relative accuracy of the methods appears similar to that shown in Fig. 3.1 with Dirichlet conditions. Fig. 3.10 shows the L_2 error norms are almost the same for Chebyshev and Gauss points and averages about 50 percent greater for Lobatto points. However, Fig. 3.11 shows that the disparity in fluxes (calculated using Eq. (3.12)) is significant. Although, the differences are not large for n < 6, the differences in convergence rate are large. As discussed below, similar flux errors persist even when Bi is so large that a Dirichlet condition is

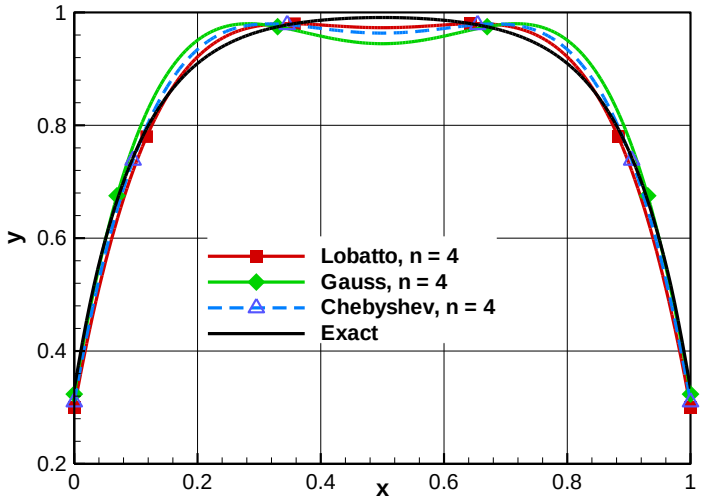


Fig. 3.9 Solutions Eq. (3.1), Bi = 10, φ = 5, k = 1

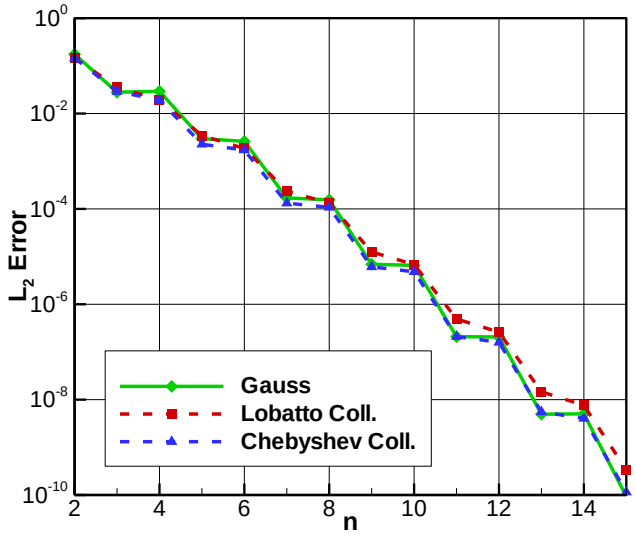


Fig 3.10 L_2 Error Norm, Bi = 10 φ = 5, k = 1

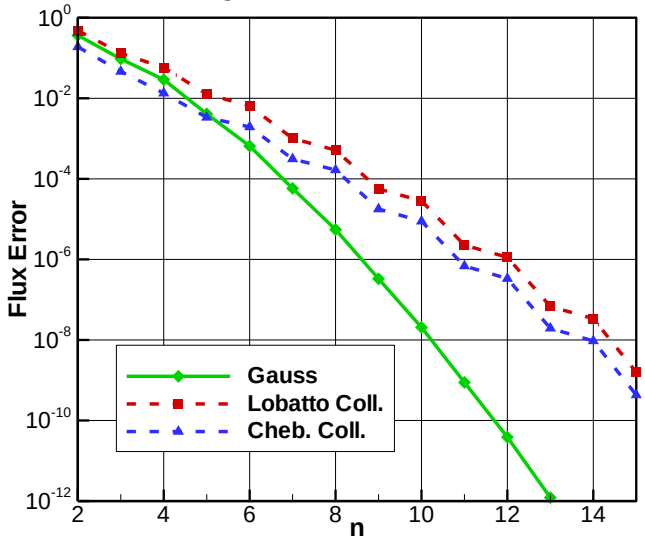


Fig 3.11 Flux Error, Bi = 10 φ = 5, k = 1

approached. Since Eq. (3.3) shows that Bi does not change the shape of the profiles, there can be no reason for this slower convergence rate other than a flaw in the formulation of the method.

This problem with flux boundary conditions (Lobatto points) first came to light in the early 1970s [Ferguson and Finlayson (1970), Ferguson (1971), Finlayson (1971), Elnashaie and Cresswell (1973)]. The problem is clearly evident for two different problems in Finlayson (1972) (see Table 5.7 and Fig. 5.7). The problem is only briefly discussed by Villadsen and Michelsen [(1978), p. 248]. Other articles attributed the problem to the strong treatment of boundary conditions when quadrature weights are nonzero on the boundary [Young and Finlayson (1976), Michelsen and Villadsen (1981)]. Collocation at Gauss points was recommended for problems with flux boundary conditions. Unfortunately, many are not aware of the problem, because later texts perpetuated the use of boundary collocation. A superior alternative method can be found by examining other Methods of Weighted Residuals (MWR) and the foundation of orthogonal collocation.

3.1.2 Method of Moments

To solve Eq. (3.1) by the method of moments, the residual is weighted by x^k for k = 0,…,n - 1; however, weighting by any linearly independent set of n polynomials through degree n - 1 will give identical results. For example, the spectral-tau method often uses the first n Legendre polynomials [see section 1.2.7, Canuto, *et al.* (1988)]. The results with Legendre polynomials would be identical except possibly for rounding errors. Another suitable set of linearly independent polynomials are the Lagrange interpolating polynomials for only the n interior points. These polynomials are related to those in Eq. (1.3) by:

$$\ell_i^*(x) = \ell_i(x)\frac{x_i(1-x_i)}{x(1-x)} \tag{3.16}$$

where the asterisk indicates the reduced polynomial. Using these weight functions in Eq. (1.9) together with the residual function, Eq. (3.8), and integrating numerically, the problem becomes:

$$\sum_{k=1}^{m}\left[\left(\sum_{i=0}^{n+1} y_i \left.\frac{d^2\ell_i}{dx^2}\right|_{x_k}\right) + r\left(x_k, \sum_{i=0}^{n+1}\ell_i(x_k)y_i\right)\right] W_k \ell_j^*(x_k) = 0 \tag{3.17}$$

for j = 1,…,n, where x_k and W_k designate quadrature base points and weights, respectively, which are not yet specified. The method of moments requires that all boundary conditions be satisfied exactly. For Dirichlet conditions, Eq. (3.3), the boundary values are substituted. For third kind conditions, Eq. (3.1b), Eq. (3.11) provides the two extra equations.

For $m > n$ the quadrature base points are naturally different from the nodal interpolation points used to define the trial functions, Eq. (1.3). If $m = n$, and the interpolation points correspond to the quadrature points, some wonderful simplifications occur. For this case, Eq. (3.17) simplifies to:

$$\sum_{i=0}^{n+1} W_j B_{ji} y_i + W_j r(x_j, y_j) = 0 \tag{3.18}$$

Eq. (3.18) is equal to Eq..(3.10) when each row is multiplied by the quadrature weight, W_j, so the equations are equivalent. To make this approach work, we need a quadrature formula that will give an exact or accurate approximate integration of Eq. (3.17) with $m = n$ and $0 < x_i < 1$. The most accurate quadrature of this type is Gaussian quadrature. We now examine the accuracy of Gaussian quadrature for integration of Eq. (3.17).

The trial functions, $\ell_i(x)$, are polynomials of degree $n + 1$, so the second derivative is of degree $n - 1$. The weight functions, $\ell_j^*(x)$ are of degree $n - 1$. Combining the two terms, the diffusion term is of order $2n - 2$ and the first order reaction term is of order $2n$. Since Gaussian quadrature is exact for polynomials through degree $2n - 1$, integration of the diffusion terms is exact, while the source term misses exact integration by one degree. To achieve an exact representation of the method of moments, the following integration must be performed more accurately:

$$\begin{aligned} D_{ji} &= \int_0^1 \ell_j^*(x)\ell_i(x)\,dx \\ &= -\frac{1}{2n+1} x_i(1-x_i)\widehat{\widehat{W}}_i^b \widehat{\widehat{W}}_j^b \quad \text{for } i \neq j \\ &= \frac{2n}{2n+1} W_i \quad \text{for } i = j \\ &\approx \delta_{ji} W_i \end{aligned} \tag{3.19}$$

We call $\boldsymbol{D}$ the *mass matrix*. This name reflects the origins of these methods in structural mechanics. The analytical expression (lines 2 and 3 above) is derived in section 2.7, where $\widehat{\widehat{W}}_i^b$ are the normalized and shifted barycentric weights defined by Eq. (2.66) and W_i are the Gaussian quadrature weights. The collocation method produces the last line above, which is clearly a reasonable approximation of the full matrix. In finite element methods, the reduction of the mass matrix to a diagonal one is called *lumping*. Lumping is achieved automatically here due to the approximate integration. The mass matrix is frequently associated with time derivatives and for this reason it is also called the *capacity matrix*.

To list the complete set of equations, it is convenient to combine the boundary and interior equations by defining:

$$C_{ji} = \delta_{j,n+1} A_{n+1,i} - \delta_{j,0} A_{0,i} - W_j B_{ji} \tag{3.20}$$

With this definition, the complete set of equation for the first order reaction with constant coefficients and third kind boundary conditions is:

$$\delta_{j,0} 2Bi_0 y_0 + \delta_{j,n+1} 2Bi_1 y_{n+1} + \sum_{i=0}^{n+1} \left(C_{ji} + 4\varphi^2 D_{ji} \right) y_i = 4\varphi^2 \sum_{i=0}^{n+1} D_{ji} = 4\varphi^2 W_j \qquad (3.21)$$

$\boldsymbol{C}$ is called the *stiffness matrix*. Another name which comes from structural mechanics. For an arbitrary set of points, $\boldsymbol{C}$ is normally nonsymmetric. However, in section 2.6 we show that due to the specific orthogonality of the Legendre polynomials, it is symmetric for collocation at Gauss points. For the full moments method the complete matrix problem is not symmetric because of $\boldsymbol{D}$, but with its diagonal approximation the complete system of equations is symmetric and positive definite.

One deficiency of orthogonal collocation, pseudospectral or differential quadrature methods is that self-adjoint operators do no lead to symmetric matrix problems. The usual implementation of the equivalent spectral-tau method leads to a horribly messy matrix structure. This development shows that by a simple reorganization of the equations this desirable matrix structure is achieved with Gauss points. A symmetric matrix problem cuts the calculations for solution almost in half (Fadeeva, 1959). This matrix structure has theoretical and computational benefits in other areas, e.g. eigenvalue problems or iterative solution with the conjugate gradient method. For a full moments method, more calculations are required because the mass matrix is not symmetric. Also, for nonlinear reaction terms, the mass matrix must be recalculated every Newton iteration. These extra calculations are rarely worth the effort. More complicated source terms are treated below in section 3.1.5.

Lobatto quadrature with n interior points has sufficient accuracy to integrate all the terms in Eq. (3.17) exactly for a first order reaction. However, it does not reduce to a collocation method because end points terms would appear in Eq. (3.18). These terms are zero with Gauss points, because the quadrature weights are zero on the boundaries. Orthogonal collocation at Lobatto quadrature base points bears no direct relationship to the moments method. With Chebyshev points, the associated Clenshaw-Curtis quadrature also has nonzero quadrature weights at the endpoints. In addition, Clenshaw-Curtis quadrature is not accurate enough to produce a good approximation to the moments method. For these reasons, it also bears no direct relationship to the method of moments. The method of moments and collocation at Gauss points is an important example where quadrature weights on the boundary are undesirable, contrary to the claims of some authors.

The mass and stiffness matrices, $\boldsymbol{D}$ and $\boldsymbol{C}$ in Eqs. (3.19) and (3.20), are available from the computer codes described in Chapter 2.

3.1.3 Galerkin Method

To solve the problem with the Galerkin method, the residual is weighted by the trial functions $\ell_j(x)$. Since the Robin boundary conditions reduce to Dirichlet conditions for large Bi number, we will consider only the more general conditions, Eq. (3.1b). The Galerkin method permits two different methods to treat boundary conditions involving derivatives. The boundary conditions may be satisfied exactly or they may be treated as *natural* boundary conditions. The natural

boundary condition treatment is derived from variational principals and provides a means to integrate the influence of the surroundings on the solution domain [Finlayson (2014)]. Dirichlet conditions can be thought of as degenerate natural conditions [Courant (1943)], e.g. infinite Bi number.

Although it is not a fundamental requirement, the normal procedure with the orthogonal collocation, pseudospectral or differential quadrature method is to satisfy the boundary conditions exactly, i.e. boundary collocation, Eq. (3.11). The short development in Appendix B shows the difficulties with this approach. For this reason, we will use the *natural* boundary condition treatment here. With this approach, the boundary condition is integrated into the solution procedure and is satisfied approximately along with the rest of the differential equation.

To solve the problem with the Galerkin method, the residual Eq. (1.3) is weighted by the trial functions $\ell_j(x)$, for j = 0,…,n + 1. The equations are converted to the *weak formulation* by integrating the second derivative term by parts:

$$\sum_{i=0}^{n+1} \ell_j \frac{d\ell_i}{dx}\bigg|_0^1 y_i - \int_0^1 \left(\sum_{i=0}^{n+1} \frac{d\ell_j}{dx}\frac{d\ell_i}{dx} y_i - \ell_j r(x,y) \right) dx = 0 \qquad (3.22)$$

The first term contains the two boundary derivatives, so we substitute the boundary conditions and integrate the other terms using a suitable quadrature formula:

$$\delta_{j,0} 2Bi_0 y_0 + \delta_{j,n+1} 2Bi_1 y_{n+1} + \sum_{k=1}^{m} W_k \left(\sum_{i=0}^{n+1} \frac{d\ell_j}{dx}\bigg|_{x_k} \frac{d\ell_i}{dx}\bigg|_{x_k} y_i - \ell_j(x_k)\, r(x_k, y(x_k)) \right) = 0 \qquad (3.23)$$

The x_k in Eq. (3.23) designate the quadrature base points, which differ from the nodal interpolation points for $m > n + 2$.

Consider quadrature with n interior points. Since the trial functions are polynomials of degree n + 1, the diffusion term is of degree $2n$. For a first order reaction the source term is of degree $2n$ + 2. An n point Gaussian quadrature is exact through degree $2n$ - 1, so neither term would be integrated exactly. On the other hand, Lobatto quadrature with n interior points gives exact integration through degree $2n$ + 1, so it gives exact integration for the diffusion term, but misses exact integration of the reaction term by one degree. This is the same level of discrepancy found when the moments method is approximated with Gaussian quadrature. Radau quadrature is one degree less accurate than Lobatto quadrature, so it also integrates the diffusion term exactly.

Using quadrature with n interior points, Eq.(3.23) reduces to:

$$\delta_{j,0} 2Bi_0 y_0 + \delta_{j,n+1} 2Bi_1 y_{n+1} + \sum_{i=0}^{n+1} C_{ji}\, y_i - W_j\, r(x_j, y_j) = 0 \qquad (3.24)$$

where:

$$C_{ji} = \sum_{k=0}^{n+1} W_k A_{kj} A_{ki} = \delta_{j,n+1} A_{n+1,i} - \delta_{j,0} A_{0,i} - W_j B_{ji} \tag{3.25}$$

Since Lobatto and Radau quadrature can perform the integration by parts exactly, it follows that the stiffness matrix, $\boldsymbol{C}$, can be calculated by either expression above. The far right expression is identical to Eq. (3.20). Given this relationship, it is clear that at the interior points, Eq. (3.24) is identical to Eq. (3.18) and equivalent to Eq. (3.10), i.e. collocation.

To examine the different treatment of the boundary conditions, compare Eqs. (3.11) and (3.24) at the boundaries, $j = 0$ and $j = n + 1$. Substituting the far right expression of Eq. (3.25) for $\boldsymbol{C}$, the boundary equations are:

$$\begin{aligned} 2Bi_0 y_0 - \sum_{i=0}^{n+1} A_{0i} y_i - W_0 \left(\sum_{i=0}^{n+1} B_{0i} y_i + r(0, y_0) \right) = 0 \\ 2Bi_1 y_{n+1} + \sum_{i=0}^{n+1} A_{n+1,i} y_i - W_{n+1} \left(\sum_{i=0}^{n+1} B_{n+1,i} y_i + r(1, y_{n+1}) \right) = 0 \end{aligned} \tag{3.26}$$

Eq. (3.26) differs from (3.11) by the extra term on the right. It is the boundary quadrature weight multiplying the interior residual, Eq. (3.8), evaluated at the boundaries, $R(0,\boldsymbol{y})$ and $R(1,\boldsymbol{y})$. Rather than setting the boundary condition residual to zero, this procedure sets the weighted combination of the boundary condition residual and the interior residual at the boundary to zero, like Eq. (1.15). Unlike boundary collocation, the boundary condition is not satisfied exactly, but both residuals will converge to zero at an exponential rate, along with the other residuals.

Note also that the right expression of Eq. (3.25) and Eq. (3.20) are identical for the calculation $\boldsymbol{C}$, so Eqs. (3.24) and (3.26) are equally valid for Gauss points, since the quadrature weights are zero at the boundaries. The left expression of Eq. (3.25) is not valid for Gauss or Chebyshev points, because the quadrature is not accurate enough to perform the integration by parts exactly. For an infinite Bi number, Eqs. (3.24) and (3.26) reduce to the Dirichlet conditions, $y_0 = y_{n+1} = 0$.

Lobatto quadrature calculates the second derivative term exactly, but misses exact integration of the source term by one degree. For the first order reaction, the full Galerkin method requires a more accurate calculation of the mass matrix. The exact integration is worked out in section 2.7:

$$
\begin{aligned}
D_{ji} &= \int_0^1 \ell_j(x)\ell_i(x)\,dx \\
&= \frac{(-1)^{(i+j+1)}}{2n+3}\sqrt{W_i W_j} \quad \text{for } i \neq j \\
&= \frac{2(n+1)}{2n+3} W_i \quad \text{for } i = j \\
&\approx \delta_{ji} W_i
\end{aligned}
\tag{3.27}
$$

Collocation at Lobatto points approximates this matrix by a diagonal one composed of the quadrature weights as shown in the last expression above. Substituting a more accurate integration of the source term into the Galerkin method, Eq. (3.23), produces an equation identical in form to Eq. (3.21). For the Galerkin method, both the stiffness and mass matrices are symmetric. The mass matrix is full for the Galerkin method and is diagonal or lumped for collocation at Lobatto points. For more complex rate expressions, a full mass matrix adds complexity and calculations which do not normally improve the accuracy enough to warrant the extra calculations, see section 3.1.5.

In summary, with the development above and in Appendix B, collocation at Lobatto quadrature base points is an accurately approximation of the Galerkin method, but only when a *natural* or *weak* treatment of flux boundary conditions is used. Section 3.1.2 shows collocation at Gauss points is an accurate approximation of the method of moments. Collocation at Radau points is in between and reduces to boundary collocation at one boundary and a natural treatment at the other.

The Clenshaw-Curtis quadrature is not accurate enough to give a good approximation to either the Galerkin or moments method. Collocation at Chebyshev points approximates MWR with the extra radical term, $1/\sqrt{1-x^2}$, in the weight function. Chebyshev points and quadrature weights are between Gauss and Lobatto points and weights (see Figs. 1.4 and 1.5), so can also be justified on that basis. We propose that Chebyshev points also use Eq. (3.24) with $\boldsymbol{C}$ calculated by Eq. (3.20) or the far right expression of Eq. (3.25). Then, it will still be equivalent to collocation at the interior points, but with a *natural* treatment of flux boundary conditions.

Formulations using the stiffness matrix and a natural treatment of flux boundary conditions are referred to as *weak formulations*. Nevertheless, the method is equivalent to collocation at the interior points. Eq. (3.24) or equivalently (3.26) is used at the boundary points. The boundary condition treatment is equivalent to boundary collocation only when the quadrature weight is zero at the boundary. Unlike with Gauss and Lobatto points, the stiffness matrix for Chebyshev points is not symmetric (except for $n < 4$). Chebyshev points have the advantage when fast Fourier transforms (FFT) can be brought to bear, but the disadvantage when symmetric matrix problems are beneficial. Also note the natural boundary condition treatment causes boundary rate terms to appear in the approximation, Eq. (3.26). For a nonlinear rate expression, all $n + 2$ equations are nonlinear, whereas with Gauss points the two boundary equations are linear. These linear equations can be eliminated initially so only n nonlinear equations must be solved iteratively.

The relationship between collocation at Lobatto points and the Galerkin method is what motivated Villadsen and Stewart (1967) to select Lobatto points (they treated the equivalent symmetric problem discussed in section 3.1.5). However, they considered only Dirichlet conditions. The natural boundary condition treatment for the Galerkin method is advocated by Finlayson and Scriven (1966). They point out this treatment sets a combination of boundary and interior residuals to zero, as in Eq. (3.26). The natural treatment is standard in most finite element applications. Its use with collocation at Lobatto points was suggested by Young (1977) and described more fully by Funaro (1988, pp 143, 204). Canuto (1986) discussed an apparently similar procedure for Chebyshev and Lobatto points. The procedure is also described in later work [Canuto, et al. (2006), Shen, et al. (2011)]. Unfortunately, Canuto, et al. chose to rename the method *G-NI* (Galerkin with Numerical Integration), which is likely to cause much confusion, and besides, the Galekin relationship has been known since 1967. Shen, et al. used the more appropriate name *collocation in the weak form*. These texts describe not only the weak or natural treatment, but also the *strong*, boundary collocation treatment. None seem to claim a major benefit to the weak formulation. Consequently, most applications use boundary collocation. The weak treatment should be the standard, correct one, while the strong formulation is incorrect. The examples to follow make a compelling case to support this statement.

Linear Source, Constant Coefficients, Third Kind B.C., Galerkin/moments.

Fig. 3.12 shows the flux errors from Fig. 3.11 updated with results from full Galerkin and moments methods and Lobatto and Chebyshev collocation with a natural treatment of the boundary conditions. These results are labeled *"Nat."* while the boundary collocation results are labeled *"bc"*.

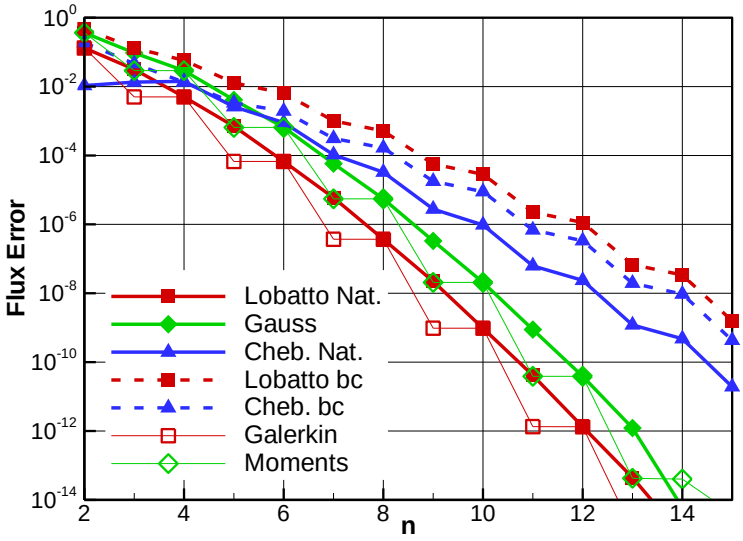


Fig 3.12 Flux Error, Bi = 10 φ = 5, k = 1

The improvement by using natural boundary conditions is impressive, especially with Lobatto points. With the natural boundary condition treatment, the convergence rates for the three choices of point are similar to those in Fig. 3.3. Since the problem reduces to the Dirichlet problem for large Bi, the effect of this parameter was investigated by solving the problem with Bi = 2, 5, 10 and 50 for comparison. With the largest value, the condition approaches a Dirichlet condition. For the Lobatto point cases in Fig. 3.12 with n = 8, the ratio of the error with boundary collocation relative to a natural treatment, is 1.4x10^3. This error ratio at n = 8 varies from 7x10^3 for Bi = 2 to 0.3x10^3 for Bi = 50. The convergence rates are similar, so the disparity grows with n. Clearly, the error with boundary collocation can be several orders of magnitude greater even for quite large

values of Bi. The errors with Chebyshev points are also reduced by using a natural boundary condition treatment, but to a lesser extent, but still almost an order of magnitude at n = 8.

As discussed above, the natural boundary condition treatment, Eq. (3.26), sets a weighted combination of the boundary condition and the boundary value of the interior residual to zero. Neither residual will be identically zero, but they converge to zero at an exponential rate. Fig. 3.13 shows the convergence behavior of these two residuals as a function of n for Bi = 10. Graphs for the other values of Bi are very similar. As shown in Eq. (3.26), the ratio of the two residuals is equal to the boundary quadrature weight, W_0 or W_{n+1} which are O(1/(2n^2)) for Lobatto quadrature and half that for Clenshaw-Curtis quadrature (Chebyshev points).

We also note that in Fig. 3.12 the Galerkin and moments methods improve in a stair step fashion. When n is even, the results with Galerkin and Lobatto points are identical and the results with moments and Gauss points are identical. This result is an artifact caused by the symmetry of the solution about x = 0.5. For example, with n = 4 the trial solution is a 5th order polynomial, but the highest order term has a zero coefficient because of the symmetry. Lobatto points normally miss exact integration of the Galerkin method by one degree, but since the problem symmetry knocks out the highest degree term, the two methods agree. With n = 3 the polynomial is 4th order, so the Galerkin method is just as good as when a 5th order term is included. An analogous situation applies for the moments method and collocation at Gauss points.

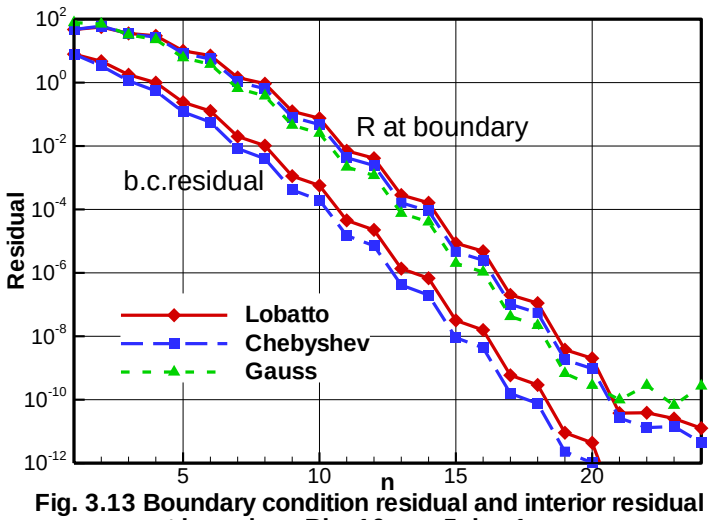


Fig. 3.13 Boundary condition residual and interior residual at boundary, Bi = 10, φ = 5, k = 1

The problem here is one of the few cases when collocation at Lobatto points is identical to a Galerkin method and collocation at Gauss points is identical to the method of moments. Consider the linear equation with variable coefficients:

$$\frac{d}{dx}\left(k(x)\frac{dy}{dx}\right) + p(x)\frac{dy}{dx} + q(x)y + r(x) = 0$$

The trial and weight functions are degree n + 1 and Lobatto quadrature is exact for 2n + 1, so integration is exact if the residual does not exceed degree n. Unless the solution is symmetric, exact integrals are produced if $k(x)$ is linear, $p(x)$ is constant, $q(x)$ = 0 and $r(x)$ is a polynomial of degree n or less. If the solution is symmetric, exact integration is achieved when $q(x)$ is a constant. Collocation at Gauss points is identical to the method of moments under the same conditions, because the weight functions are two degrees less, n – 1, but so is the quadrature accuracy, 2n - 1. Usually, Gauss and Lobatto points give approximations to the moments and

Galerkin methods, but often very good ones. Some texts incorrectly state that collocation at Gauss points replicates the Galerkin method for more general conditions [Boyd (2000), p. 89]. This example problem can be treated more efficiently by exploiting the symmetry with polynomials in x^2, which is discuss below in section 3.1.5.

3.1.4 Mass Conservation and Fluxes

One is usually interested in the flux at the boundaries. For example, if a fluid were flowing on both sides of the slab we would want to know the rate of mass or heat transfer from the slab. For the symmetric problem, the transfer is quantified by a single normalized flux, η given by Eqs. (3.4) and (3.5). For a nonsymmetric problem one would generally want the breakdown of left and right side fluxes, while in multiple dimensions the flux profiles along the boundaries may be important. The *average energy equation* or divergence theorem gives only the total flux. If the sum of the individual fluxes (or integral of the boundary profiles in multiple dimensions) equals the total flux, this gives us greater confidence in the individual values.

The total of the fluxes will give the average rate *provided the method conserves mass*. Eq. (3.5) , the divergence theorem or average energy equation is just an overall balance. It is derived by integrating Eq. (3.1) across the domain. In general, a method will be conservative if the integral of the residual is zero. If the MWR weights ($w_i(x)$ in Eq. (1.9)) contain unity in some combination, the method will be conservative.

The method of moments and collocation at Gauss points are conservative regardless of the formulation. If formulated with monomials or Legendre polynomials the first weight function would be x^0 or $P_0 = 1$. With our formulation from Section 3.1.2, $\sum \ell_i^*(x) = 1$. Since this method is conservative, Eq. (3.5) is obeyed when the boundary fluxes are calculated by differentiation of the approximate solution, Eq. (3.13). This explains why the errors in Fig. 3.3 are the same with either method of calculation for collocation at Gauss points.

For the Galerkin method and collocation at Lobatto points, the sum of all the Lagrange interpolating polynomials is also unity. However, if we consider Dirichlet boundary conditions, the boundary values are directly substituted into Eq. (1.3) so the first and last interpolating polynomials are not used as weight functions. The method appears not to be conservative due to the left over terms on the right side below:

$$\left.\frac{dy}{dx}\right|_0^1 + \int_0^1 r(x,y)\,dx = \int_0^1 [\ell_0(x) + \ell_{n+1}(x)]R(x,\boldsymbol{y})dx \tag{3.28}$$

Using quadrature, Eq. (3.28) is approximated by:

$$\left.\frac{dy}{dx}\right|_0^1 + \int_0^1 r(x,y)\,dx = W_0 R(0,\boldsymbol{y}) + W_{n+1}R(1,\boldsymbol{y}) \tag{3.29}$$

The two terms on the right side are those needed to correct the fluxes in the natural boundary condition treatment, Eq. (3.26), so the correction makes the method conservative. To be consistent with the Galerkin method, the individual fluxes should be approximated by:

$$\left.\frac{dy}{dx}\right|_{x=0} = \sum_{i=0}^{n+1} A_{0i} y_i + W_0 \left(\sum_{i=0}^{n+1} B_{0i} y_i + r(0, y_0) \right) = \sum_{i=0}^{n+1} A_{0i} y_i + W_0 R(0, \boldsymbol{y})$$
$$\left.\frac{dy}{dx}\right|_{x=1} = \sum_{i=0}^{n+1} A_{n+1,i} y_i - W_{n+1} \left(\sum_{i=0}^{n+1} B_{n+1,i} y_i + r(1, y_{n+1}) \right) = \sum_{i=0}^{n+1} A_{n+1,i} y_i - W_{n+1} R(1, \boldsymbol{y}) \tag{3.30}$$

The same equations written with the stiffness matrix are:

$$\left.\frac{dy}{dx}\right|_{x=0} = -\sum_{i=0}^{n+1} C_{0i} y_i + W_0 r(0, y_0)$$
$$\left.\frac{dy}{dx}\right|_{x=1} = \sum_{i=0}^{n+1} C_{n+1,i} y_i - W_{n+1} r(1, y_{n+1}) \tag{3.31}$$

When computed from these equations, the two fluxes will be consistent with Eq. (3.5), the average energy equation or divergence theorem. For the full Galerkin method, the fluxes must be calculated with the equivalent expression using the mass matrix, $\boldsymbol{D}$**.** For the Robin boundary conditions the expressions are consistent with Eq. (3.1b), so it would be far simpler to calculate fluxes by multiplying $2Bi$ by the boundary values of $\boldsymbol{y}$. However, this calculation is subject to roundoff errors when Bi is large.

Note that Eqs. (3.30) and (3.31) are also valid with Gauss points, since the boundary quadrature weights are zero. This equivalence is useful when writing one computer code which will work with either type of points. Using these equations with Chebyshev points will also make that method conservative, which provides some additional justification for using a *natural* treatment with Chebyshev points.

The relationship shown above, between the overall balance and natural boundary condition treatment was noted by Finlayson and Scriven (1966). Ferguson (1971) proposed approximation of flux boundary conditions using the integral rather than the derivative in Eq. (3.5). Ferguson's procedure has some limitations but is equivalent to a natural treatment when it is applicable. The natural treatment is equivalent, more general and easier to implement.

Linear Source, Variable Coefficients, Dirichlet B.C.

To make the problem nonsymmetric and more interesting, consider the case of a first order source, but with a coefficient which varies with position according to:

$$\hat{r}(x, y) = \big(0.2 + 1.6x^2(3 - 2x)\big)(1 - y) = q(x)(1 - y) \tag{3.32}$$

The spatial variation, q(x), goes from 0.2 on the left edge to 1.8 on the right edge, with an average value of 0.5 on the left half and 1.5 on the right half giving an overall average of 1.0. For a given value of φ, the average rate constant is the same as with Eq. (3.2) for k = 1. Fig. 3.14 shows solutions to this problem for φ = 5 with collocation at Gauss, Chebyshev, and Lobatto points. The solution with Lobatto points is again slightly more accurate than with the

other two choices. In Fig. 3.14, the "exact" solution is one computed with large n which is exact for practical purposes.

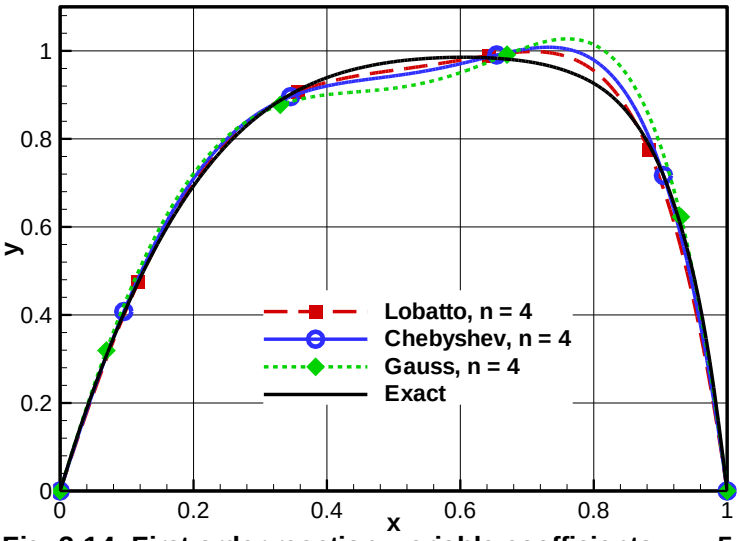


Fig. 3.14 First order reaction, variable coefficients, φ = 5

When the problem is solved with the Galerkin or moments method a more accurate mass matrix must be calculated. For many problems, especially nonlinear ones, an exact calculation would be cumbersome and is not necessary. Approximate calculations with quadrature formulas are most often used. Orthogonal collocation is a special case of approximate quadrature when the interior quadrature points and collocation points coincide. Another common procedure is to interpolate variable or nonlinear terms into the trial space, i.e.:

$$r(x,y) \approx \sum_{i=0}^{n+1} \ell_i(x) r\big(x_i, y(x_i)\big) \tag{3.33}$$

For the method of moments, these various approaches give:

$$\begin{aligned} D_{ji} &= \int_0^1 \ell_j^*(x)\ell_i(x)q(x)\,dx \approx \sum_{k=1}^{m} W_k \ell_j^*(x_k)\ell_i(x_k)q(x_k) \quad \text{or} \\ &\approx q(x_i)\int_0^1 \ell_j^*(x)\ell_i(x)\,dx \quad \text{or} \\ &\approx q(x_i)\,W_i\,\delta_{ij} \end{aligned} \tag{3.34}$$

Where the asterisk denotes the reduced functions defined by Eq. (3.16) and $D_{0i} = D_{n+1,i} = 0$ for the moments method. For the Galerkin method the trial functions, $\ell_j(x)$, are used instead of the reduced functions. The approximation with quadrature is given on the last line, while the approximation for interpolation into the trial space, Eq. (3.33), is given on the second line. Finally, the approximation using collocation is given on the last line. Once calculated, the mass matrix is substituted into Eq. (3.21). The collocation approximation reduces to Eq. (3.24). As $Bi \to \infty$ the equations reduce to specified boundary values, i.e. Dirichlet conditions.

When the mass matrix is calculated with quadrature, one must select the quadrature formula and the number of quadrature base points, m in Eq. (3.34). We select Gaussian quadrature for the moments method and Lobatto quadrature for the Galerkin method. In the discussion above for the constant coefficient problem, when comparing collocation at Gauss points with the method of moments and collocation at Lobatto points with the Galerkin method, we found collocation missed exact integration by one degree. The variable coefficient for this example,

$q(x)$ in Eq. (3.32), is degree 3, so both collocation methods miss exact integration by 4 degrees. With the interpolation approach exact integration is still missed by one degree. The approach using numerical quadrature is exact if n + 2 quadrature points are used.

This example provides a good test of the integration accuracy requirements, because with only one additional quadrature point, the integration is still two degrees shy of exact. So, besides collocation and the Galerkin method there are two additional ways to approximate $\boldsymbol{D}$: (1) interpolation of the source terms, Eq. (3.33) and (2) approximate integration with n + 1 interior quadrature points. Similarly, there are two approximate methods of calculating $\boldsymbol{D}$ for the moments method. All of these possibilities give a large volume of results for this problem. As a matter of notation, the method will be called a *full* Galerkin method or a method of moments, whenever the integrals are more accurate than with the collocation methods.

Fig. 3.14 shows the calculated results with collocation using the three choices of collocation points. Fig. 3.15 compares collocation at Gauss points with the method of moments using

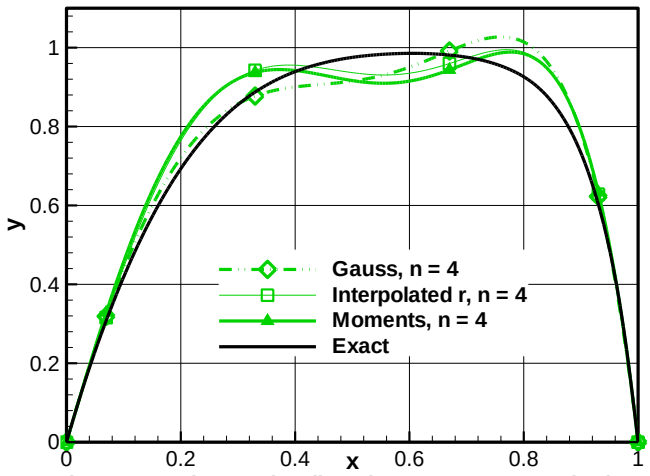


Fig. 3.15 Orthogonal Collocation & Moments methods, first order, variable coefficients, φ = 5

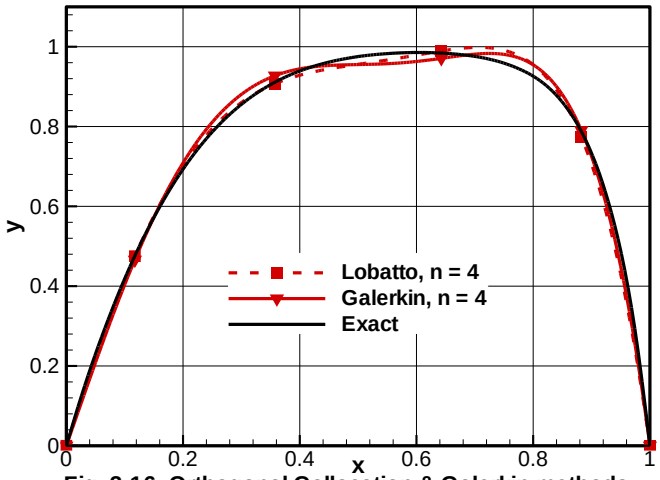


Fig. 3.16 Orthogonal Collocation & Galerkin methods, first order, variable coefficients, φ = 5

Table 3.1 Calculated Fluxes, Variable Coefficients, n = 4

	Flux left	Flux right	Flux Total	Error left	Error right	Error total
Exact	0.05062	0.13368	0.18429			
Gauss Collocation	0.05013	0.12097	0.17110	-0.96%	-9.51%	-7.16%
Chebyshev Collocation	0.05026	0.12910	0.17936	-0.71%	-3.42%	-3.68%
Lobatto Collocation	0.05073	0.13742	0.18814	0.21%	3.80%	3.09%
moments	0.04902	0.12598	0.17500	-3.16%	-5.76%	-5.04%
moments, interpolated r	0.04623	0.12473	0.17097	-8.66%	-6.69%	-7.23%
moments, n+1 Gauss	0.04903	0.12654	0.17557	-3.14%	-5.34%	-4.73%
Galerkin	0.05053	0.13561	0.18614	-0.17%	1.45%	1.00%
Galerkin, interpolated r	0.05070	0.13601	0.18671	0.16%	1.75%	1.31%
Galerkin, n+1 Lobatto	0.05055	0.13583	0.18638	-0.13%	1.61%	1.13%
Chebyshev, derivative	0.04795	0.11116	0.15911	-5.27%	-16.84%	-13.66%
Lobatto, derivative	0.04666	0.10497	0.15163	-7.82%	-21.47%	-17.73%
Galerkin, derivative	0.03953	0.10974	0.14927	-17.21%	-16.93%	-17.00%

both an exact mass matrix and interpolated rates. Fig. 3.16 compares collocation at Lobatto points with the Galerkin method. For the Galerkin method, the other approximations were not distinguishable from those in Fig. 3.16.

Table 3.2 shows results of the flux calculations for this problem when n = 4. For the results in the table, "n+1" indicates the number of interior quadrature points used in the approximation of the mass matrix and "*interpolated r*" indicates those for which Eq. (3.33) was used. In these results Eq. (3.30) or (3.31) was used to calculate the fluxes for Lobatto or Chebyshev points, or the equivalent method for the Galerkin method. The table results labeled, "*derivative*", indicates Eq. (3.13) was used to calculate the fluxes. From Fig. 3.15 it does not appear that the additional complexity of the moments method and a full mass matrix, $\boldsymbol{D}$, adds any substantial accuracy to the results. However, Table 3.2 indicates there is some improvement, but the improvement is spotty. Some flux errors are larger, especially those at x = 0, which is the "easy" side due to the more gentle profile. Fig. 3.16 indicates that relative to Lobatto collocation, the Galerkin method gives a small improvement to the profiles. Table 3.2 shows that with the Galerkin method, the right side and total errors are reduced by about a factor of 3. Also, a Galerkin approximate mass matrix with 5 interior points is very nearly as good as an exact mass matrix.

Fig. 3.17 shows the L_2 error norms versus n. As before, the L_1 and L_2 errors are relatively insensitive to differences between the methods. The ratio of best to worst error averaged over all n from 2 to 16 is only about 2. The ratio of the average error relative to the Galerkin method

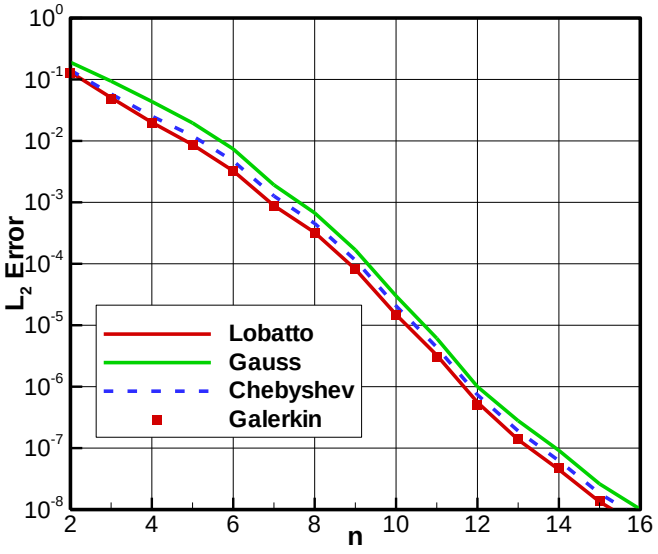


Fig 3.17 L_2 Error, First Order, variable coefficients, φ = 5

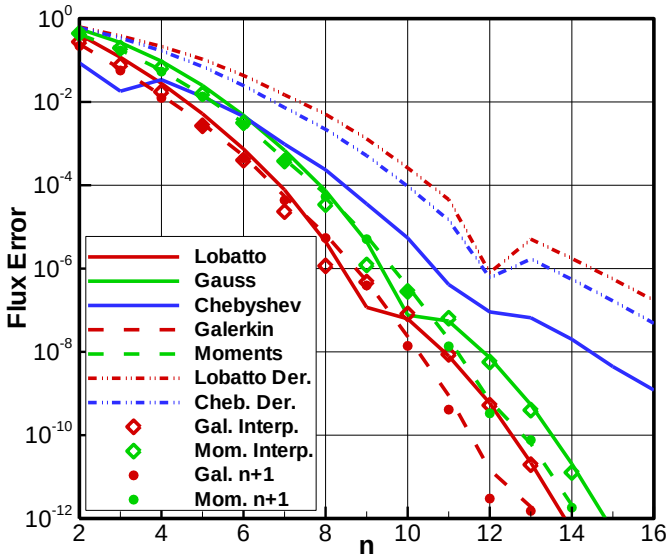


Fig 3.18 Right Side Flux Error, variable coefficients, φ = 5

for Lobatto, Chebyshev and Gauss collocation is 1.02, 1.36 and 2.03, respectively.

Fig. 3.18 shows the error of the flux on the right side, x = 1, for all the methods. The graph is busy due to all the results, but it is worthwhile to examine the effect of approximate and exact integration for the Galerkin and moments methods, since Gauss and Lobatto collocation are also approximations. In this graph, the results labeled *"Interp"* are for the interpolated source

terms, Eq. (3.33) and those labeled *"n+1"* use one additional quadrature point to give a full but approximate mass matrix. *"Der"* is used to indicate fluxes estimated by the first derivative. The other methods calculate fluxes using Eq. (3.30) or (3.31) (or equivalent for Galerkin).

Again, the results calculated from the first derivative are much less accurate except for the Gauss-collocation and moments methods for which they are identical. All the evidence shows that for other methods derivatives, Eq. (3.13), should never be used to estimate fluxes. If the reader takes this one result to heart, writing this monograph will have been worthwhile.

Table 3.3 Average Relative Error of Flux at x = 1, Variable Coefficients, Eq. (3.32)

	Relative Error
Gauss Collocation	17.73
Chebyshev Collocation	193.7
Lobatto Collocation	2.37
moments	8.61
moments, interpolated r	13.84
moments, n + 1 Gauss	7.57
Galerkin	1.00
Galerkin, interpolated r	1.70
Galerkin, n + 1 Lobatto	0.76

Fig. 3.18 shows that the Galerkin method is generally the best method, but frequently the approximate mass matrices give results that are just as good. Exact integration of the mass matrix gives no clear improvement for the moments method either. Due to the large number and variability of the results in Fig. 3.18, they are further summarized in Table 3.3. Table 3.3 lists the ratio of the error for the individual method relative to the Galerkin method geometrically averaged for n = 2 through 16. There could be other ways to summarize these results, but this method yields several clear conclusions. The full Galerkin and moments methods are about twice as accurate as their respective collocation counterparts (Lobatto and Gauss points). Interpolation of the source terms adds only a slight improvement over collocation. One quadrature point greater than collocation is on the average slightly better than an exact Galerkin or moments method. This type of behavior has been observed in finite element methods (Strang and Fix (1973)). Nonlinear examples are considered in the next section. If one considers the computational effort, collocation will invariably win. However, if a full Galerkin or moments method is desired it is wasteful to get carried away with extremely accurate integration.

The comparisons in Figs. 3.18 and Table 3.3 are akin to splitting hairs, since one increment of n normally reduces the error by almost an order of magnitude. For example, the Gauss/moments methods give errors that are about 8 times larger than the Lobatto/Galerkin counterparts, equivalent to about one increment of n. In compensation for this difference, the treatment of flux boundary conditions is simpler and more intuitive, Eq. (3.11) rather than (3.24) or (3.26), and the usually nonlinear source terms appear only at the n interior points. The simplicity of using boundary collocation is one reason for the popularity of Gauss points in the orthogonal collocation literature. The natural boundary condition treatment required to achieve good accuracy with the other choices is usually not considered by others.

Chebyshev points are historically the most popular choice in the pseudospectral and differential quadrature literature. It is the method of choice for problems requiring large n for

which FFT methods can be used. For the problem considered here, relative to the other choices, Chebyshev collocation is not competitive when flux calculations are carried out as normally recommended. With the improved flux calculations, Eq. (3.30) or (3.31), the results are good for small n, but suffer from a lower convergence rate for large n. When compared to the other choices, the case supporting the use of Chebyshev points is:

- $\boldsymbol{x}$ and $\boldsymbol{W}$ can be efficiently calculated
- approximates Chebyshev-Galerkin method with radical, $1/\sqrt{1-x^2}$
- justified by point distribution relative to other methods

On the other side of the argument:

- efficiency of $\boldsymbol{x}$ and $\boldsymbol{W}$ calculation insignificant
- Chevyshev-Galerkin convergence slower than integrated MWR without radical
- less efficient calculations due to nonsymmetric matrices

The argument concerning point distributions is an interesting one. Many view the collocation point selection problem strictly from the standpoint of point distributions. Since Chebyshev points are between Gauss and Lobatto points (see Fig. 1.5), the results should be intermediate between them. If we compare the profiles and error norms that argument holds up, but it falls apart when comparing fluxes. The other characteristic to consider is that Gauss and Lobatto quadrature are both accurate for O($2n$), while Clenshaw-Curtis quadrature is accurate for O(n), see Figs. 2.34, 2.35 and 3.8. It appears that the differences are not so important for small n, but the quadrature accuracy affects the convergence rate for fluxes at large n. Further analysis on this subject is warranted.

Proponents for the use of Chebyshev points cite the greater efficiency for computing the fundamental approximations, $\boldsymbol{x}$ and $\boldsymbol{W}$. This advantage is irrelevant for most problems since those calculations require a fraction of a millisecond for n < 1000. Also, one can make the argument that if a problem requires say n > 10, finite element trial functions will likely be more efficient in most cases. In later chapters, we describe orthogonal collocation finite element methods which can achieve very high order.

Many authors bemoan the lack of symmetry for collocation or pseudospectral approximations for self adjoint operators. When conventionally formulated, all methods suffer from this deficiency. However, a simple rescaling of the equations with the quadrature weights makes the method symmetric for Gauss, Radau and Lobatto points, see Eq. (3.18) and (3.24). The number of arithmetic operations to solve a symmetric matrix problem is roughly half that for a nonsymmetric problem (Fadeeva, 1959). There are theoretical and computational advantages for other problems as well, e.g. eigenvalue problems for parabolic equations in Chapter 4. These computational advantages tend to offset any advantages of Chebyshev points.

3.1.5 Symmetric and Nonlinear Problems

Consider now the diffusion/conduction problem similar to Eq. (3.1), but for which the solution is symmetric about the centerline, x = ½. We renormalize the coordinates to place the centerline

at x = 0, while the outer boundary remains at x = 1. The governing equation for planar, cylindrical and spherical geometry (γ = 0,1,2 respectively) is:

$$\frac{1}{x^\gamma}\frac{d}{dx}\left(x^\gamma \frac{dy}{dx}\right) + r(y) = 0 \tag{3.35}$$

with

$$\left.\frac{dy}{dx}\right|_{x=1} + Bi\, y(1) = 0 \quad \text{and} \quad \left.\frac{dy}{dx}\right|_{x=0} = 0 \tag{3.36}$$

Although, the equations were previously written to allow for a nonlinear source, the examples so far are for a linear first order reaction. The source function used for the examples which follow is of the form:

$$r(y) = \varphi^2 \hat{r}(y) = \varphi^2 \frac{(1-y)^k}{(1-K_a y)^2} \tag{3.37}$$

where we note r̂(0) = 1 as required in the definition of φ. When the denominator term, K_a = 0, the equation is identical to Eq. (3.2). When K_a > 0 and k = 1, the source function exhibits some interesting nonlinear behavior. For large K_a it is said to be *autocatalytic*, i.e. the rate increases as the reaction proceeds and the conversion, y, increases. Fig. 3.19 shows a graph of the nonlinear source functions considered in this section along with the 1st order linear case.

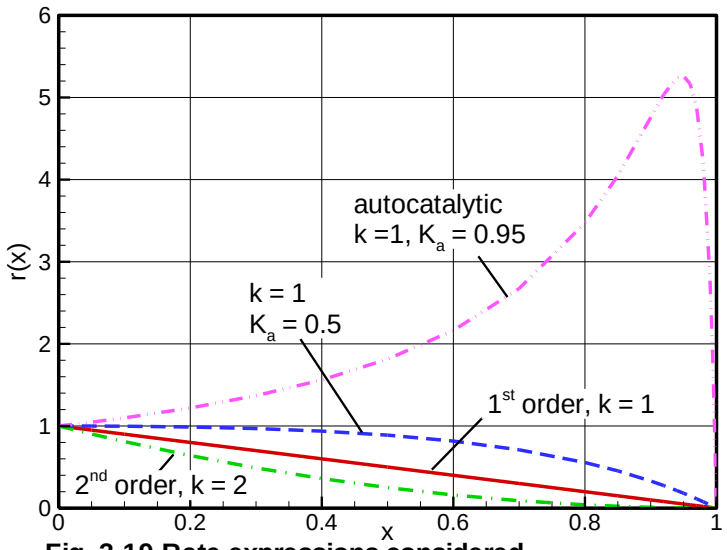


Fig. 3.19 Rate expressions considered

The dimensionless rate constant, the Thiele modulus, can be generalized to account for the geometry and source function. In all cases, the effectiveness factor, η, is unity for small Thiele modulus and becomes asymptotic for high reaction rates. The asymptotic condition corresponds to the case when all reactants are consumed, and the conversion is unity in the center of the particle. The asymptotic condition acts like a semi-infinite domain, which can be treated analytically (Rawlings and Ekerdt, 2015). For the Dirichlet boundary conditions all source functions and geometries have the following asymptotic behavior:

$$\eta = 1/\varphi^* \text{ as } \varphi^* \to \infty \tag{3.38}$$

when correlated using the generalized parameter, φ*, defined by:

$$\varphi^* = \frac{\varphi}{\gamma+1}\left[2\int_0^1 r(\hat{y})dy\right]^{\frac{-1}{2}} \tag{3.39}$$

The geometric term, $\gamma + 1$, is equivalent to using the ratio of volume to surface area as the characteristic length in the definition of φ. For a k^{th} order reaction, Eq. (3.37) with $K_a = 0$, the generalized parameter is:

$$\varphi^* = \varphi \frac{\sqrt{2(k+1)}}{2(\gamma+1)} \tag{3.40}$$

When $k = 1$ in Eq. (3.37) the generalized parameter is:

$$\varphi^* = \frac{\varphi}{\gamma+1} \frac{K_a}{\sqrt{-2[K_a + \ln(1-K_a)]}} \tag{3.41}$$

The specific nonlinear source functions considered in this section use Eq. (3.37) with: (1) 2nd order, $k = 2$, $K_a = 0$; (2) $k = 1$, $K_a = 0.5$; and (3) autocatalytic, $k = 1$, $K_a = 0.95$. These functions lead to the following values from Eq. (3.39): $(\gamma+1)(\varphi^*/\varphi)$ = 1.2247, 0.8045, 0.4697, respectively. The first two cases are mildly nonlinear, while the highly nonlinear third case is called *autocatalytic* because the rate increases initially as reactants are consumed. This type of rate relationship can lead to multiple steady state solutions to the problem.

Orthogonal Collocation

By using symmetric trial functions, the condition at $x = 0$ is automatically satisfied. For planar geometry, $\gamma = 0$, Eq. (3.35) is equivalent to Eq. (3.1) and (3.1b) when the source is not dependent on x. The third kind or Robin condition is used at $x = 1$, since it reduces to a Dirichlet condition when Bi goes to infinity. Here we consider nonlinear source functions in addition to different geometries. Since the solution is symmetric, the trial functions are Lagrange interpolating polynomials in x^2, Eq. (2.2), which is repeated here:

$$y(x) \approx \sum_{i=1}^{n+1} y(x_i)\ell_i(x^2) \tag{3.42}$$

Although the application of a Method of Weighted Residuals (MWR) is fundamentally the same, the geometry and symmetric trial functions cause some subtle differences to occur. Substituting the trial functions, the residual for the problem is:

$$R(x, \boldsymbol{y}) = \sum_{i=1}^{n+1} y_i \frac{1}{x^\gamma} \frac{d}{dx}\left(x^\gamma \frac{d\ell_i(x^2)}{dx}\right) + r(y(x^2)) = 0 \tag{3.43}$$

With the collocation method, the residual is set to zero at the interior collocation points, j = 1,…,n and the boundary condition provides the final equation. The usual recommendation is to apply boundary collocation, but from the previous examples (see Fig. 3.12) we found this to be the best approach only for Gauss points. A natural boundary condition treatment is better in general. Using conventional collocation at interior points and a natural boundary condition treatment the equations are:

$$\sum_{i=1}^{n+1} B_{ji} y_i + r(y_i) = 0 \tag{3.44}$$

at the interior points, i.e. j = 1,…,n and from the boundary condition, following Eq. (3.30):

$$\left(Biy_{n+1} + \sum_{i=1}^{n+1} A_{n+1,i} y_i \right) - W_{n+1} \left(\sum_{i=1}^{n+1} B_{n+1,i} y_i + r(y_{n+1}) \right) = 0 \tag{3.45}$$

Eq. (3.45) reduces to the Dirichlet problem, y_{n+1} = 0, for large Bi. Otherwise, it sets the weighted combination of the residuals to zero. The left term is the boundary condition residual and the right one is the boundary weight multiplying the differential equation residual at the boundary. This treatment also ensures the method is conservative as discussed in Sec. 3.1.4.

The following matrix operators are defined as before:

$$A_{ji} = \left. \frac{d\ell_i(x^2)}{dx} \right|_{x_j} \quad \text{and} \quad B_{ji} = \frac{1}{x^\gamma} \frac{d}{dx} \left[x^\gamma \left. \frac{d\ell_i(x^2)}{dx} \right]_{x_j} \right.$$

Calculation of these quantities is somewhat different for symmetric problems. For a symmetric problem $\boldsymbol{B} \neq \boldsymbol{AA}$. The correct calculation procedures are described in section 2.5.

A Newton-Raphson procedure works well for a nonlinear rate term. The procedure is outline for the Dirichlet problem. The boundary value of y (whether zero or finite) is substituted into Eq. (3.44) leaving n nonlinear equations. Given some estimate, $\boldsymbol{y}^0$, a single iteration to determine an improved estimate solves for the change as follows:

$$\sum_{i=1}^{n} \left(B_{ji} + \delta_{ji} \left. \frac{dr}{dy} \right|_{y_i^0} \right) \Delta y_i = - \sum_{i=1}^{n} B_{ji} y_i^0 - r\left(y_j^0\right) \tag{3.46}$$

After solving for the change, an improved estimate is found by adding the change to the estimate. To reduce roundoff errors, it is usually better to formulate the iterations as shown, i.e. with the residual of the algebraic equation and the change in y. If the reaction is linear, i.e. first order or 0th order, the solution is obtained after one Newton iteration.

$\boldsymbol{B}$ is not symmetric even for the self-adjoint differential operator in Eq. (3.35). However, as shown in Sections 3.1.2 and 3.1.3 a minor restructuring of the equations into its weak form produces a symmetric matrix problem:

$$\delta_{j,n+1} Bi\, y_{n+1} + \sum_{i=1}^{n+1} C_{ji} y_i - W_j r\left(y_j\right) = 0 \tag{3.47}$$

where:

$$C_{ji} = \delta_{j,n+1} A_{n+1,i} - W_j B_{ji} = \sum_{i=1}^{n+1} W_k A_{kj} A_{ki} \tag{3.48}$$

When the stiffness matrix, $\boldsymbol{C}$, is calculated with the left equality in Eq. (3.48) it is clear that Eq. (3.47) is equal to Eqs. (3.44) and (3.45) after each row is multiplied by the quadrature weight, W_j. The equality on the far right of Eq. (3.48) follows from integration of the Laplacian, $\boldsymbol{B}$, by parts as is done to convert a Galerkin method to weak form, e.g. Eq. (3.23). For a symmetric problem, Gaussian and Lobatto quadrature are accurate for degree $2n - 1$ and $2n$ in x^2, respectively. For the integration by parts, the integrand is a polynomial of degree $2n - 1$ in x^2, so the far right expression is valid for both Gaussian and Lobatto quadrature. It is not valid for the Clenshaw-Curtis quadrature used for Chebyshev points. For Chebyshev points, $\boldsymbol{C}$ must be calculated using the left equality and the resulting stiffness matrix is not symmetric for $n > 2$.

Since orthogonal collocation is known to produce good results when it closely approximates the moments or Galerkin methods, we compare it to these methods for this symmetric problem. For the Galerkin method, the weight functions are the same as the trial functions in Eq. (3.42), so the trial functions and the weight functions are both degree n in x^2. The integrand for a simple first order source term, $k = 1$ in Eq. (3.2), is degree $2n$, while the second order term is degree $2n - 1$. Lobatto quadrature is exact for degree $2n$, so both a linear source term and the second order term are integrated exactly. The method is identical to the Galerkin method.

For the method of moments, weighting the residual by the symmetric Lagrange interpolating polynomials through the interior points (like Eq. (3.16)) is equivalent to weighting by monomials in x^2 or the symmetric or even numbered Legendre polynomials thru degree n - 1 in x^2 or thru P_{2n-2}. When combined with a linear source term the integrands are $2n - 1$ and $2n - 2$ degree for source and second order terms, respectively. Gaussian quadrature is exact for degree $2n - 1$, so for the linear problem collocation at Gauss points is identical to the method of moments.

The results for the symmetric problem are like those found for nonsymmetric problems, sections 3.1.2 and 3.1.3, i.e. Gauss points yield a good approximation to the method of moments, while collocation at Lobatto points approximates the Galerkin method accurately. However, for nonsymmetric problems we found that in each case the corresponding quadrature formula to be one degree shy of the accuracy needed for exact integration of the source terms, even for a linear problem, so this result for symmetric problems is slightly better. This analysis provides further explanation for the exact correspondence between moments, Galerkin and collocation methods for the problem whose results are shown in Fig. 3.12 with an even number of points.

In order to achieve the accuracy stated above for Gaussian and Lobatto quadrature, the geometric parameter, γ, must be considered. Chapter 2 explains how the Jacobi polynomial roots and quadrature weights are determined and some examples are shown in Chapters 1 and 2 (see Figs. 1.6, 2.32 and 2.33). Except for Chebyshev points, the collocation points are shifted closer to the boundary for cylindrical and spherical geometry. This result is consistent with intuition, since the point density is shifted in the direction of large incremental volume.

The justification for the use of Chebyshev points relies more on approximation theory than on accurate approximations of integrated MWR. It approximates MWR with the radical, $1/\sqrt{1-x^2}$, included in the weight or test function. It appears that Chebyshev points are not normally altered to better suit the geometry (Trefethan (2000), p. 115; Boyd (2000), p. 380). For our purposes, we simply use the right half of the points used for a nonsymmetric problem. Chapter 2 describes the calculation for the interpolatory quadrature weights for cylindrical and spherical coordinates. The second derivative matrix operator, $\boldsymbol{B}$, is calculated the same way as for other points (see Chapter 2) and the stiffness matrix, $\boldsymbol{C}$, is calculated using the left expression in Eq. (3.48).

Galerkin Method

For a nonlinear reaction term, orthogonal collocation at Gauss and Lobatto points are only approximations of moments and Galerkin methods, respectively. For most nonlinear problems, a moments or Galerkin method with exact integration would be difficult to implement. However, the results from the variable coefficient problem, see Fig. 3.18, suggest exact integration is totally unnecessary. Those results showed that one extra quadrature point was actually better than exact integration. An improved approximation of the Galerkin method is developed by using a greater number of quadrature base points. As a point of nomenclature, we use the names Galerkin and moments, whenever the integrals are more accurately approximated than with collocation. A Galerkin approximation with additional quadrature points requires solution of the nonlinear algebraic equations:

$$\begin{aligned}
&\int_0^1 \left[\sum_{i=1}^{n+1} y_i \frac{1}{x^\gamma} \frac{d}{dx}\left(x^\gamma \frac{d\ell_i(x^2)}{dx} \right) + r\big(y(x^2)\big) \right] \ell_j(x^2) x^\gamma \, dx \\
&\quad = \delta_{j,n+1} \left. \frac{dy}{dx} \right|_{x=1} - \int_0^1 \left[\sum_{i=1}^{n+1} y_i \ell_j' \ell_i' - r\big(y(x^2)\big) \ell_j \right] x^\gamma \, dx \qquad (3.49) \\
&\quad \approx \delta_{j,n+1} Bi \, y_{n+1} + \sum_{i=1}^{n+1} C_{ji} y_i - \sum_{k=1}^{m+1} W_k r\left(y\left(x_k^2\right) \right) \ell_j\left(x_k^2\right) = 0
\end{aligned}$$

where $m > n$. To designate the accuracy of the integration, we use the notation e.g. "*Galerkin +2*" to designate quadrature of $m = n + 2$. There is no need to use more accurate quadrature for the second order term, since it is integrated exactly in Eq. (3.47). Only the source term needs more accurate integration. Since r is nonlinear, an iterative solution is required. Dropping the square on x for convenience, proceed by linearizing the source term about the value from an initial guess or previous iteration, denoted by y^0:

$$\sum_{k=1}^{m+1} W_k r\big(y(x_k)\big)\,\ell_j(x_k) \approx \sum_{k=1}^{m+1} W_k \ell_j(x_k)\left[r\big(y^0(x_k)\big) + \big(y(x_k) - y^0(x_k)\big)\left.\frac{dr}{dy}\right|_{y^0(x_k)}\right]$$
$$= \sum_{k=1}^{m+1} W_k \ell_j(x_k)\, r(y^0(x_k)) + \sum_{i=1}^{n+1} \Delta y_i \sum_{k=1}^{m+1} W_k \ell_j(x_k)\ell_i(x_k)\left.\frac{dr}{dy}\right|_{y^0(x_k)} \qquad (3.50)$$
$$= h_j^0 - \sum_{i=1}^{n+1} D_{ji}^0 \Delta y_i$$

Interpolation is used to calculate values at the interior quadrature points, $y(x_k)$, from the values at the nodes, y_i, e.g. $y(x_k) = \sum_{i=1}^{n+1} \ell_i(x_k) y_i$, k = 1,…,m. $\boldsymbol{D}$ is a mass matrix and in the standard parlance, $\boldsymbol{h}$ is called a *load vector*. The superscript denotes that the values depend on the estimated value y^0, so for a Newton-Raphson iteration these quantities must be recalculated every iteration. The values are updated by solving for the change Δy_i

$$\delta_{j,n+1} Bi\, \Delta y_{n+1} + \sum_{i=1}^{n+1} \big(C_{ji} + D_{ji}^0\big)\Delta y_i = h_j^0 - \delta_{j,n+1} Bi\, y_{n+1}^0 - \sum_{i=1}^{n+1} C_{ji} y_i^0 \qquad (3.51)$$

The change is then added to the previous estimates.

It is obvious that the computational complexity increases substantially when moving from collocation with n interior quadrature points to a greater number of quadrature points, because of the interpolations required. Each iteration, the values of y at the quadrature points are calculated by interpolation of the nodal values. Then, the rate is evaluated at the greater number of quadrature points. Finally, the mass matrix and load vector are calculated. For symmetric problems interpolation of the rate term, Eq. (3.33), is identical to collocation.

Mildly Nonlinear, Dirichlet B.C., Various Geometry:

Here we consider Eq. (3.35) with the two mildly nonlinear source functions, Eq. (3.37) with (1) 2nd order k = 2, K_a = 0 and (2) k = 1, K_a = 0.5. The normalized flux or effectiveness factor, η, defined in Eqs. (3.4) and (3.5) is again the quantity of primary interest from the solution. As described above η is unity for small φ and asymptotic for large φ. This behavior is qualitatively the same for all source functions and geometries. The asymptotic solution for large φ, Eq. (3.39), for these two functions gives: $(\gamma + 1)(\varphi^*/\varphi) = 1.2247, 0.8045$.

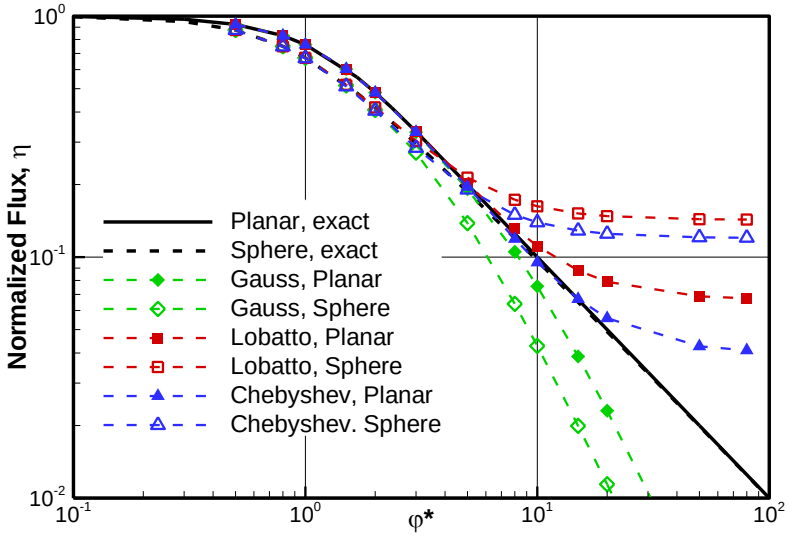


Fig. 3.20 Normalized flux for first order source, k = 1, n = 2

Fig. 3.20 shows the normalized flux for a first order source, k = 1, planar and spherical geometry, together

with numerical solutions for n = 2. The results for an infinite cylinder lie between these two. For planar geometry with φ^* = 5 the results in Fig. 3.20 correspond to those displayed in Figs. 3.1 through 3.3. This symmetric treatment with n = 2 corresponds to the nonsymmetric treatment with n = 4 in Fig. 3.1. This value of φ^* corresponds to the point where the numerical results with n = 2 begin to depart from the exact solution. For smaller values of φ^* the results agree reasonably well with the analytical solutions, while for larger φ^* the asymptotic solution is more accurate. For spherical geometry, $\varphi = 3\varphi^*$, so the problem is correspondingly more difficult and the departures of the numerical results from the analytical solutions occur at lower values of φ^*.

Based on the results in Fig. 3.20 one would conclude that Chebyshev points are better, which seems to conflict with the results in Fig. 3.3. Closer examination reveals the apparent difference is due to the scale of the graphs. For reference, the maximum difference between the analytical results for planar and spherical geometry is about 15%. The numerical errors for planar geometry at φ^* = 5 are not distinguishable from the exact curve in Fig. 3.20. As discussed in Section 3.1.1, for φ^* = 5 the errors are 0.8, -3.1, and -4.3 percent for Lobatto, Chebyshev and Gauss points, respectively. For small n, the Chebyshev results in Fig. 3.3 are the most accurate, but the flux errors are greater than 1 percent and there are significant errors in the profiles. However, for a relatively loose error tolerance Chebyshev points are characteristically more accurate for these problems, in addition to Fig. 3.3 see Figs. 3.12 and 3.18. Lobatto points usually overpredict the exact η, while Gauss points usually underpredict it. Chebyshev points generally underpredict η at small φ^* and overpredict at large φ^*.

Now consider a second order source, k = 2, with K_a = 0. Fig. 3.21 shows calculated profiles for spherical geometry, n = 4 and φ^* = 5 or $\varphi = 15\sqrt{2/3} = 12.247$. The errors near the center of the sphere are significant for Gauss and Lobatto points. This result is likely because the Galerkin and moments methods shift the points away from the center where the volume is small and the solution is relatively less important. Chebyshev points are usually not shifted, so remain the same regardless of the geometry.

Integrating with respect to the differential volume, r^2dr for spheres, is consistent with a variational treatment. The collocation points correspond to the most accurate quadrature formulas over differential volumes. Relative to planar geometry, the points are shifted away from the center with small differential volume. Although there are some excursions between the nodes, the errors at the collocation points are characteristically small. As discussed above, it seems this

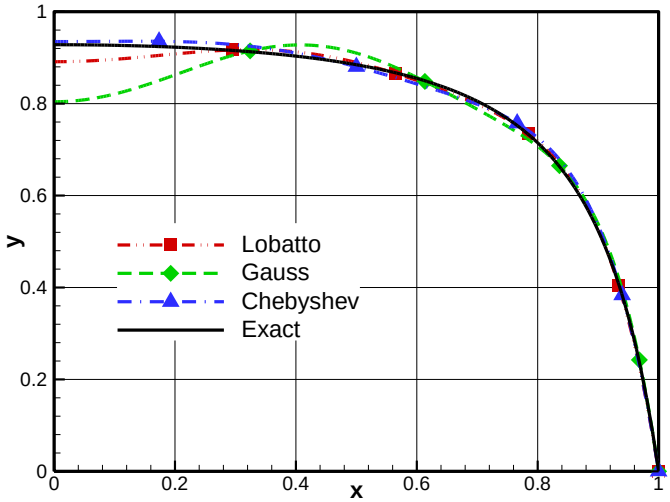


Fig. 3.21 Profiles, spherical, k = 2, K_a = 0, φ^* = 5, n = 4

behavior is likely related to the property of *superconvergence*, exhibited by some finite element methods.

Despite the errors in the profiles in Fig. 3.21, collocation at Chebyshev points give a flux error of 1.5 percent which is virtually the same as the error with Gauss points and twice the error given by Lobatto points. Fig. 3.22 shows the flux error as a function of n for this problem and the corresponding one with planar geometry. The relative performance of the methods is similar to that shown in Figs. 3.3, 3.12 and 3.18. The results suggest that for comparable accuracy spherical problems require about 1.6 to 1.7 times as many points as planar geometry when compared for a given accuracy and value of φ^*. For this nonlinear problem, the errors for a given number of points is not substantially worse than for the linear first order source function. For example, with planar geometry, φ^* = 5 and n = 2, Lobatto, Chebyshev and Gauss points give respectively, flux errors of 0.8, -3.1 and -4.3 percent for k = 1 and 2.0, -2.1 and -6.0 percent for k = 2. However, the convergence rate is somewhat slower, so the disparity grows with n.

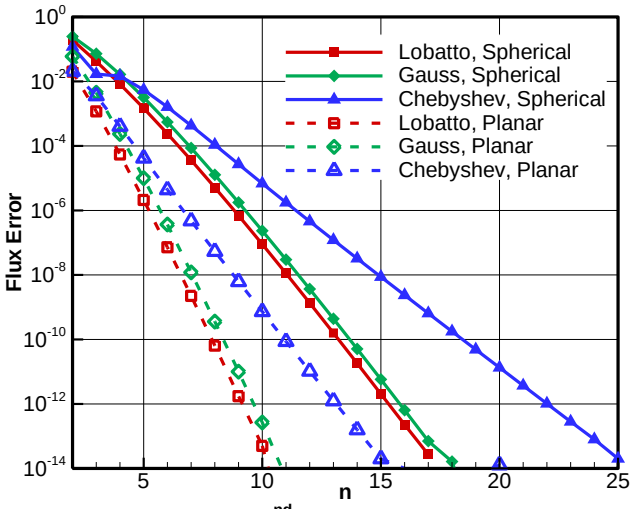


Fig 3.22 Flux error 2nd order source, k = 2, φ* = 5

Fig. 3.23 shows the normalized flux as a function of φ^* for the second order source for planar and spherical geometry, along with numerical results for n = 2. The "exact" solution was again computed with large n. Comparing with Fig. 3.20, the maximum difference between the curves for first and second order source functions is only about five percent, which is one third the difference between the curves for the two geometries. The numerical errors also appear to be qualitatively similar. The numerical methods track the exact solution up to the start of the asymptotic solution. Since these calculations are for a single catalyst pellet, a full scale chemical reactor simulation requires solution of the problem repeatedly at various locations and times in the larger system. Using OC with small n in combination with the asymptotic solution is an economical method for treating the full system [e.g. Young and Finlayson (1976), Finlayson and Young (1979)].

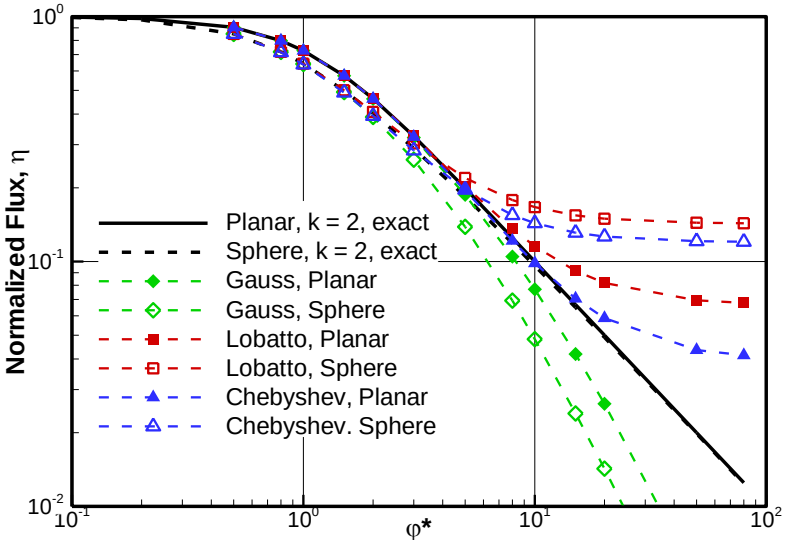


Fig. 3.23 Normalized flux for second order source, k = 2, n = 2

One must be careful about generalizing the results for the 2nd order source to other nonlinear source functions. The second nonlinear source function, k = 1, K_a = 0.5, decreases at a slower rate, so it behaves more like a zeroth order rate expression, which would be constant for all y < 1. A spherical problem with φ^* = 5 corresponds to $\varphi = 18.646$. For spherical geometry, the profiles are shown in Fig. 3.24 for n = 5, while the error in the normalized flux is displayed versus n in Fig. 3.25, and the normalized flux is shown as a function of φ^* in Fig. 3.26. For φ^* = 5 this problem is more difficult than the second order problem, which is apparent from comparison of the profiles in Figs. 3.21 and 3.24. In Fig. 3.24 only one node is present in the steep portion of the curve. The oscillations are greater than for the second order source, even though an additional point is used. However, as with other cases the errors are exceedingly small at the collocation points. This result together with the accuracy of the quadrature produces accurate values for the normalized flux, η. For the case in Fig. 3.24, the errors in η are only 0.1, 1.5 and 3.0 percent for Lobatto, Gauss and Chebyshev points, respectively. The numerical results in Fig. 3.26 were again calculated with n = 2. In Fig. 3.26 the difference due to geometry is as much as 17 percent. Comparing to Figs. 3.20 and 3.23, the values here are as much as 17 percent greater than for a first order source and 22 percent greater than for the second order source.

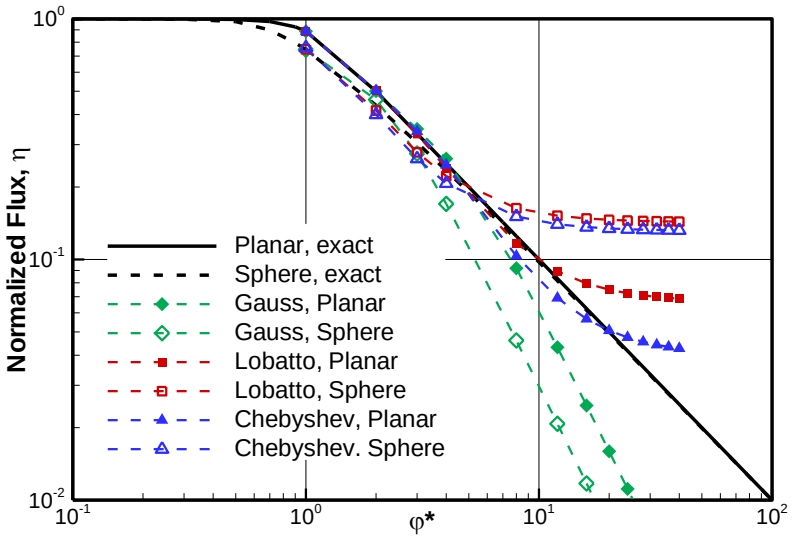


Fig. 3.26 Normalized flux for k = 1, K_a = 0.5 and n = 2

These nonlinear problems have also been solved with the Galerkin method. Based on the results for the variable coefficient problem, see Fig. 3.18, we anticipate that one or two

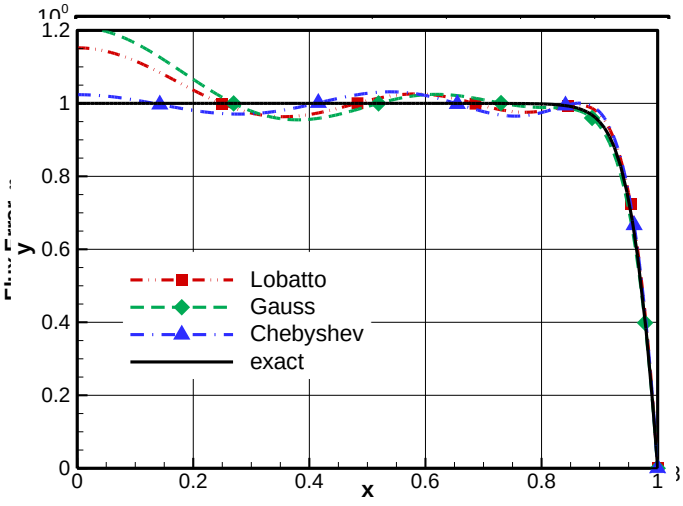


Fig. 3.24 Profiles, spherical k = 1, K_a = 0.5, φ* = 5, n = 5

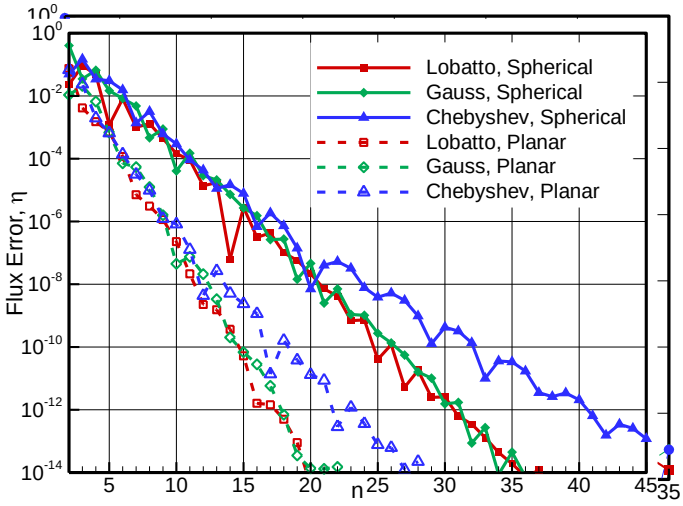


Fig 3.25 Flux error, k = 1, K_a = 0.5, φ* = 5

additional quadrature points may produce some improvement, but more accurate integration would be a waste of computing resources. Fig. 3.27 shows the error vs n for the second order source and Fig. 3.28 shows the errors for k = 1, K_a = 0.5. For the Galerkin method the integrals were evaluated with Lobatto quadrature and $m = n + 1$, $n + 2$ and $n + 3$. The results are compared to collocation at Lobatto points, which is equivalent to a Galerkin method with $m = n$. The Galerkin results are labeled "+1" and "+2" in the graphs. The results with one extra quadrature point was slightly better than collocation. The improvement with two extra points was smaller still. The scatter in the second problem makes the improvement scarcely evident. Overall, the improvement with a Galerkin method and one extra quadrature point is similar to the improvement of collocation at Lobatto points relative to collocation at Gauss points. One additional quadrature point, $m = n + 1$, increases the problem complexity significantly. Since the quadrature points and nodes are no longer the same, interpolations are required to evaluate the rates, and then the mass matrix and load vector must also be calculated, see Eq. (3.50). In the vast majority of cases, collocation will be more efficient than a full Galerkin method, even if one or two extra points are required for comparable accuracy.

Table 3.4 Relative Average Flux Error

	Planar	Spherical	Planar	Spherical
	k = 2, K_a = 0		k = 1, K_a = 0.5	
Gauss	17.23	3.00	2.17	1.50
Chebyshev	169.9	189.8	10.4	10.2
Lobatto	3.75	3.83	1.22	0.91
Galerkin +1	1.00	1.00	1.00	1.00
Galerkin +2	0.97	0.76	0.60	0.73
Galerkin +3	0.96	0.74	0.67	0.58

Table 3.4 summarizes the errors in Figs. 3.22, 3.25, 3.27 and 3.28. Due to the scatter in some of the results, the average errors are shown relative to those of the Galerkin method with $m = n + 1$. For the four columns the values are geometric averages of the relative error for $n \le 9$, 16, 19 and 34, respectively for the four columns. The Galerkin method is generally more accurate than collocation at Lobatto points, but the differences are small, and the number of additional quadrature points is relatively unimportant. The collocation methods in order of accuracy are Lobatto > Gauss > Chebyshev. However, as the problem becomes more difficult, the differences are lessened.

Autocatalytic Source, Third Kind B.C.:

Eq. (3.35) with the autocatalytic source function, Eq. (3.37) with k = 1, K_a = 0.95, is considered here. A third kind boundary conditions is used with Bi = 3, 10 and 100 for comparison. With the largest value of Bi the conditions approach a Dirichlet boundary condition. For a linear source, boundary collocation gave poor results compared to use of a *natural* treatment, Eq. (3.45), see Fig. 3.12. Since the geometry causes no fundamental differences in solution procedure, planar geometry is used. The source creates a rate which is essentially negative one order for small y and is called *autocatalytic*. This type of expression is not uncommon and occurs for the oxidation of carbon monoxide to carbon dioxide in millions of automotive catalytic converters. For this problem, Eq. (3.41) gives $\varphi = 2.129\varphi^*$.

Table 3.5 Autocatalytic Reaction Parameters for n = 1

	c_0	W_1	W_2	a	b	φ_{min}	φ_{max}	Error φ_{min}	Error φ_{max}
Gauss	3.0000	1.0000	0.0000	2.9126	0.0000	0.723	0.901	1.8%	12.6%
Chebyshev	2.3704	0.8889	0.1111	2.6049	0.0029	0.683	0.852	-3.8%	6.4%
Lobatto	2.0833	0.8333	0.1667	2.4490	0.0041	0.663	0.825	-6.7%	3.2%

The nonlinear equations are again solved with a Newton-Raphson method, Eq. (3.46). Using a quadratic profile for an initial guess works reasonably well. However, when two solutions exist convergence problems can occur if the initial estimate is poor. During the calculations, the value of y can exceed the physical limit of 1. In this case it is generally better to use a linear extrapolation of the rate for y > 1 rather than setting r = 0.

An autocatalytic rate expression can lead to multiple steady state solutions. This is easily illustrated by considering the approximation with a single term, n = 1. The governing equations are Eqs. (3.44) and (3.45) or equivalently Eq. (3.47), where for all methods the stiffness matrix is of the form:

$$C = \begin{bmatrix} c_0 & -c_0 \\ -c_0 & c_0 \end{bmatrix} \tag{3.52}$$

The values of c_0 are listed in Table 3.5 along with the quadrature weights. Eq. (3.47) for n = 1 is:

$$\begin{aligned} c_0(y_1 - y_2) - W_1\varphi^2\hat{r}(y_1) &= 0 \\ c_0(y_2 - y_1) - W_2\,\varphi^2\hat{r}(y_2) + Bi\,y_2 &= 0 \end{aligned} \tag{3.53}$$

$\hat{r}(y_2)$ could be approximated by a Maclaurin series, but given the magnitude of Bi = 100, there is little error by using $\hat{r}(y_2)$ = 1. After using the second equation to eliminate y_2, the remaining equation is of the form:

$$a\frac{y_1}{\varphi^2} - b = \hat{r}(y_1) \tag{3.54}$$

Table 3.5 gives the parameters for the different approximations, while Fig. 3.29 is a graph of Eq. (3.54). The graph illustrates that three solutions occur for a range of φ. The intermediate solution is unstable, while the other two are stable. The predicted range of occurrence is calculated from the limiting slopes in Fig. 3.29. With an accurate approximation, multiple solutions occur for 0.71 < φ < 0.80, so the equation with n = 1 are only approximate. However, considering the simplicity of the approximation for this

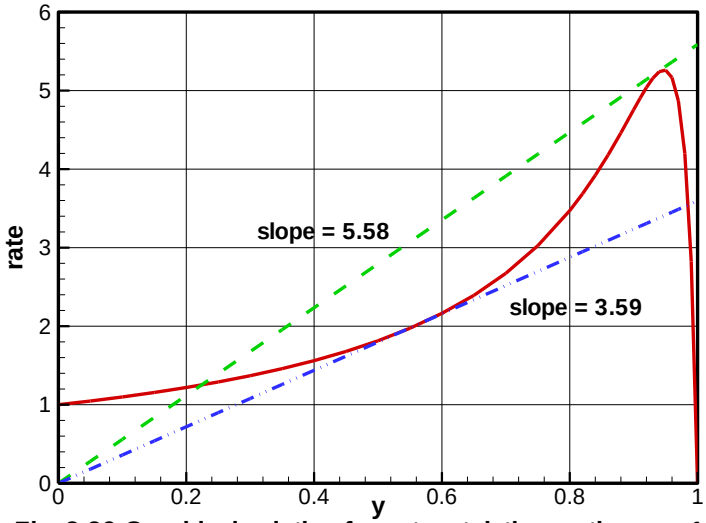


Fig. 3.29 Graphical solution for autocatalytic reaction, n = 1

highly nonlinear problem, the results are good. Also, the graphical solution shown in Fig. 3.29 is helpful for understanding the multiple solution phenomenon.

For φ = 0.75, Figs. 3.30 and 3.31 show calculated conversion profiles for several approximations together with an accurate profile (n = 12). As predicted by the simple n = 1 approximation, there are two stable profiles at this condition. The lower one is accurately approximated with n = 1 or 2, whereas the higher profile requires more terms to achieve an

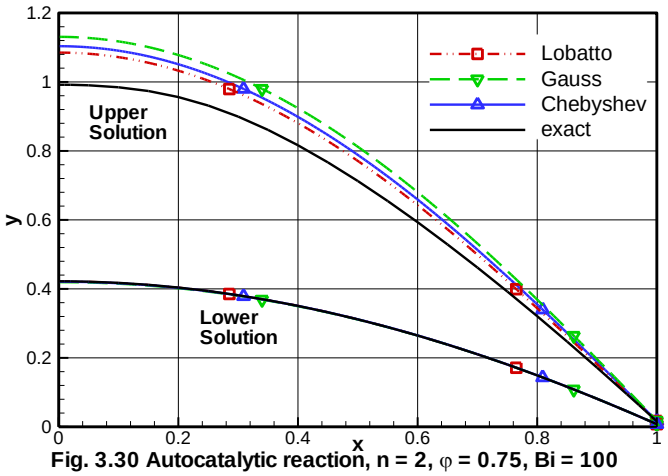


Fig. 3.30 Autocatalytic reaction, n = 2, φ = 0.75, Bi = 100

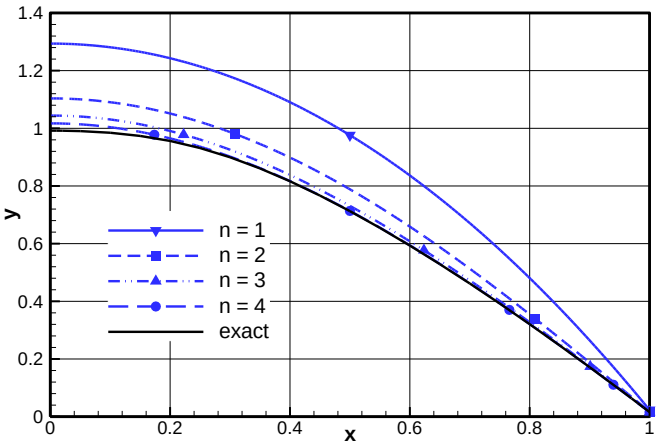


Fig. 3.31 Autocatalytic reaction, Chebyshev points, φ = 0.75, Bi = 100

accurate solution. Even though the profile appears to be relatively smooth, keep in mind that the second derivative must follow the rate expression plotted in Fig. 3.29. The nonlinearity of the rate function makes the problem more difficult for large conversion, y > 0.8. The low order approximations predict nonphysical conversion values greater than unity. However, at the collocation points y < 1.

Fig. 3.32 shows the residual, Eq. (3.43), of the upper solution for several approximations. Since for these conditions the greatest nonlinearity occurs for y > 0.8 or x < 0.4, the residual becomes very large and negative near the center. These errors contribute to the errors observed in Figs. 3.30 and 3.31.

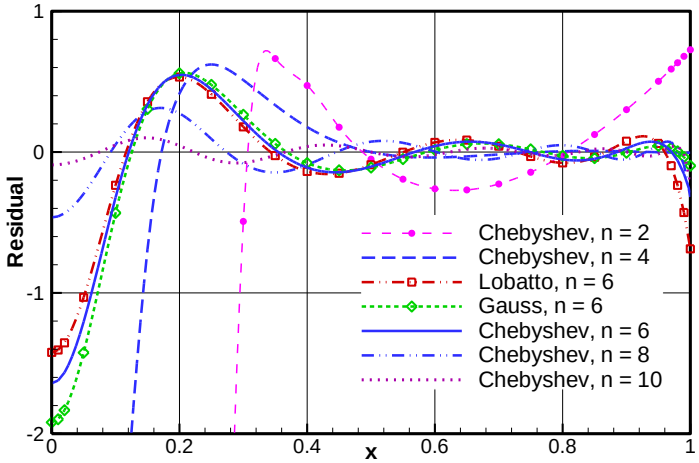


Fig. 3.32 Residual, autocatalytic rate, φ = 0.75, Bi = 100

The normalized boundary flux, η, is shown for a range of φ in Figs. 3.33 and 3.34. The asymptotic solution for $Bi \to \infty$, also plotted in Fig. 3.33 tracks the upper solution for all φ. The "exact" solution, calculated with large n, is accurate to the scale of the graph. It shows the two solutions for 0.71 < φ < 0.80. The lower solution shows convergence to within 0.05% for n = 2, in agreement with Fig. 3.30. However, considerably more points are required to approximate the upper solution and large φ. For n = 3, Chebyshev and Gauss points predict false multiple solutions at large φ = 1.2 – 1.3. The maximum percentage error for

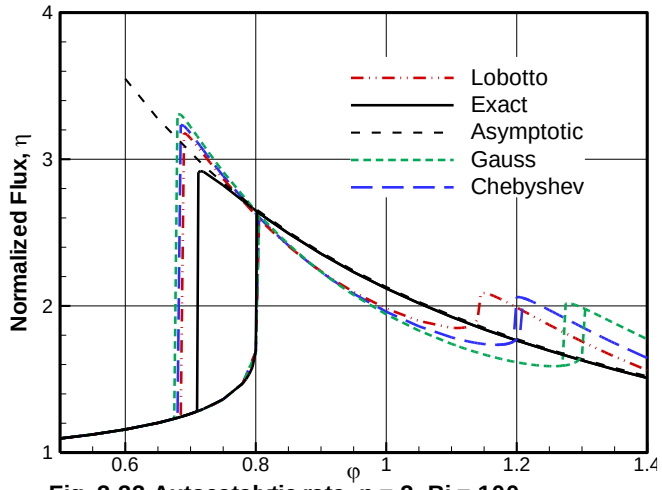


Fig. 3.33 Autocatalytic rate, n = 3, Bi = 100

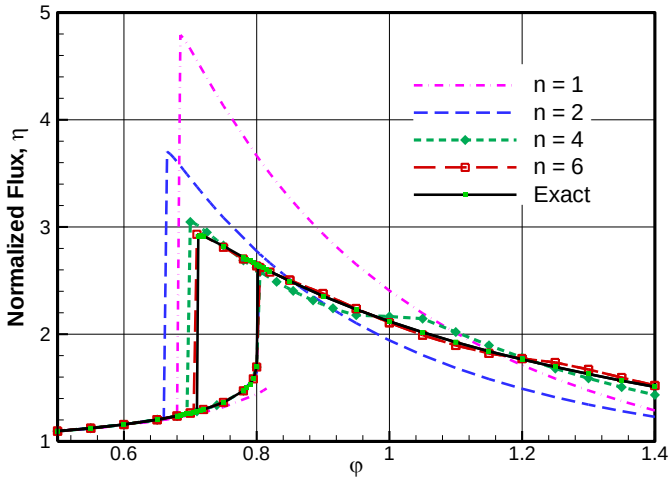


Fig. 3.34 Autocatalytic rate, Chebyshev points, Bi = 100

the upper solution, φ < 1.4 with Gauss, Chebyshev and Lobatto points, respectively, is 7.6, 6.3 and 4.8 when n = 4, and 3.3, 2.6 and 2.2 for n = 6.

Figs. 3.35 and 3.36 show errors in the normalized flux or effectiveness factor as a function of n for φ = 0.75 and Bi values of 3 and 100. There are no major differences in accuracy with the different points, at least for the upper solution when a natural boundary condition treatment is used. Lobatto points produce slightly better results, but the geometric average of the error ratios for n < 35 is only about 2 for the three values of Bi considered. This result is different from the other problems with a linear or mildly nonlinear source where we found a slower convergence rate with Chebyshev points. Compared to the other problems, the number of points required to achieve high accuracy is greater for a given value of φ*, especially when you

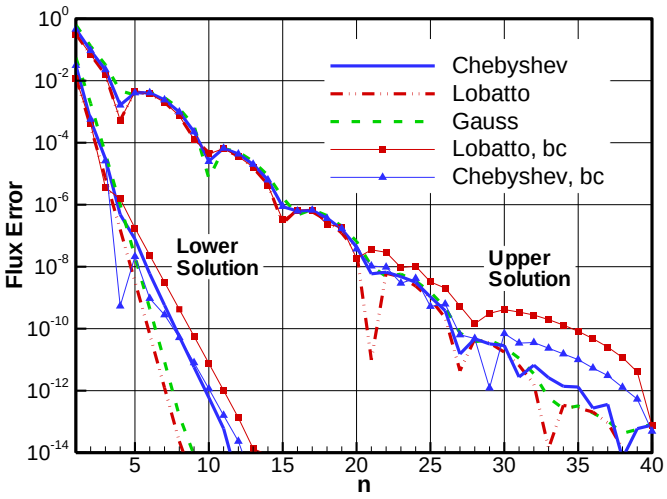


Fig 3.35 Flux Error, Autocatalytic, φ = 0.75, Bi = 100

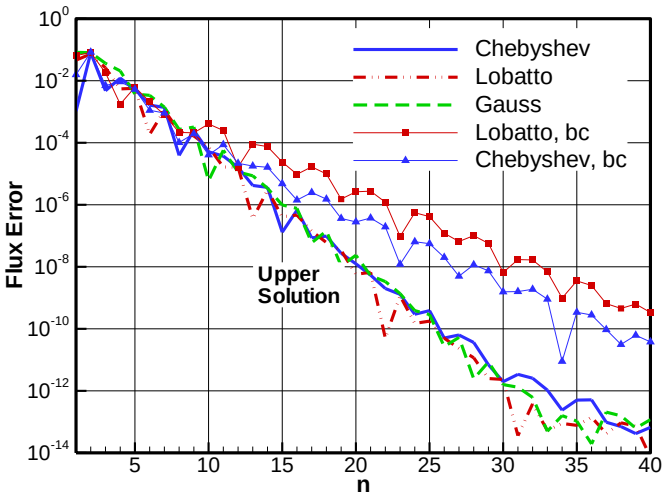


Fig 3.36 Flux Error, Autocatalytic, φ = 0.75, Bi = 3

consider that this symmetric treatment is equivalent to using twice as many points in a nonsymmetric treatment. However, even for this highly nonlinear problem, engineering accuracy is achieved with only 5 to 10 points.

The problem was also solved with the Galerkin method integrated with $m = n + 1$, $n + 2$ and $n + 3$ interior Lobatto quadrature points. These cases are again labeled “+1”, etc., for the flux

errors shown in Fig. 3.37. Since collocation at Lobatto points is equivalent to a Galerkin method with $m = n$, the Lobatto results are shown for comparison. The errors are similar, but on average the results are slightly better with the Galerkin +1 method. Due to the scatter of the results in Fig. 3.37, the geometric mean (n < 35) of the errors relative to the Galerkin +1 are listed in Table 3.6. The table shows that on average, collocation at Lobatto points is about as accurate as a Galerkin method with 2 or 3 additional quadrature base points.

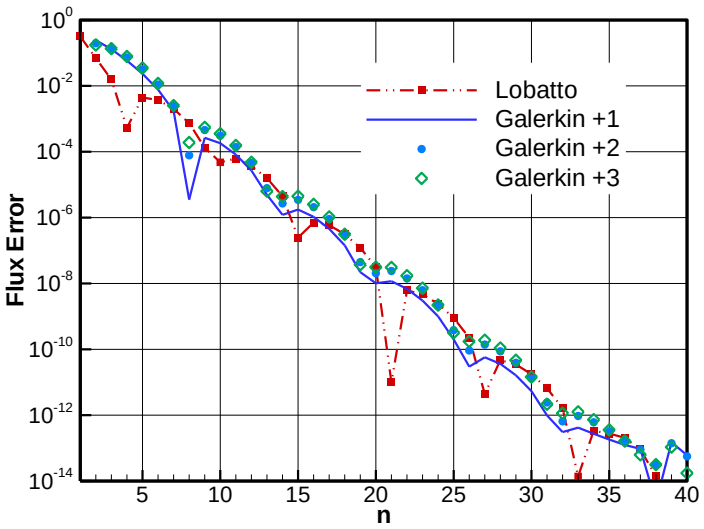


Fig 3.37 Flux Error, Autocatalytic, φ = 0.75, Bi = 100

Table 3.6 Flux Error Relative Average

Gauss coll.	4.1
Chebyshev coll.	3.8
Lobatto coll.	2.0
Galerkin +1	1.0
Galerkin +2	2.0
Galerkin +3	2.3

To put these numbers in perspective, each increment of n reduces the error by a factor of roughly 2.3, so these differences are quite small. Fig 3.35 shows that for the easier lower solution, Chebyshev points give a slower convergence rate than the other two choices, which is the same behavior observed for the other problems tested. However, for the upper solution and for errors greater than 10^{-6}, the figures and table show similar results for all methods.

For comparison, both boundary collocation (labeled *"bc"*) and a natural treatment, Eq. (3.45), are included. The results in Fig. 3.12 show that a natural boundary condition treatment is significantly better for that problem even for small n, a relatively loose error tolerance, and Bi = 2 to 50. However, for this problem Figs. 3.35 and 3.36 show that the differences are not as dramatic. The accuracy with both approaches is similar for errors larger than 10^{-4}, 10^{-6} and 10^{-8} for Bi of 3, 10 and 100, respectively. For the linear problem and Bi = 5, the value $y_{n+1} = 0.5000$, while for this problem and Bi = 3 the value $y_{n+1} = 0.4578$ indicates the relative amount of external resistance is similar in both cases. Most likely, the difference in convergence behavior is due to the extreme nonlinearity of the autocatalytic source function.

To examine the boundary condition treatment, observe that the left parenthesis of Eq. (3.45) contains the residual of the boundary condition, while the right parenthesis contains the interior residual evaluated at the boundary, i.e. $R(1,\boldsymbol{y})$ from Eq. (3.43). This treatment sets the weighted combination of the two residuals to zero, like Eq. (1.15). The ratio of the two residuals is the boundary quadrature weight, W_{n+1}. For Gauss points, the equation reduces to boundary collocation because $W_{n+1} = 0$, while $W_{n+1} \approx 1/(2n^2)$ and $1/(4n^2)$ for Lobatto and Chebyshev points, respectively. The natural boundary condition treatment drives both residuals to zero as the approximation is refined. This result is demonstrated for the linear problem in Fig. 3.13. For this highly nonlinear problem, Figs. 3.38 and 3.39 show the behavior

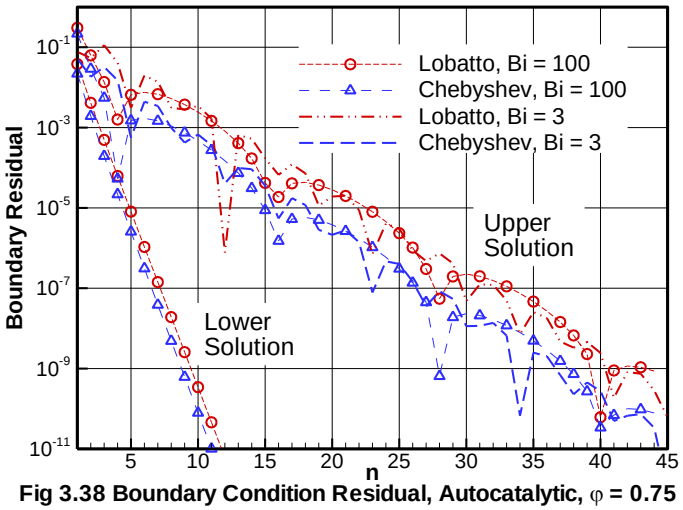


Fig 3.38 Boundary Condition Residual, Autocatalytic, φ = 0.75

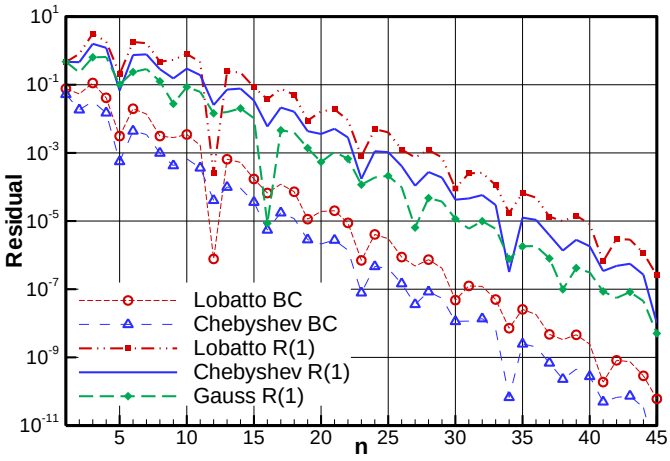


Fig 3.39 Boundary Condition and Interior Residuals, Autocatalytic Rate, φ = 0.75, Bi = 3

of these residuals for increasing n. Fig. 3.38 plots the boundary condition residual for two values of Bi, while Fig. 3.39 shows both the boundary condition residual and $R(1,\boldsymbol{y})$ for Bi = 3. Both residuals go to zero at an exponential rate, but the convergence is somewhat erratic with periodic dips and frequent sign changes. This erratic behavior is evident at x = 1 in Fig. 3.32 but is difficult to see due to the number of curves. The ratio of the two residuals in Fig. 3.39 is the quadrature weight, W_{n+1}. The exponential convergence rate of the residuals overwhelms the O(1/n^2) change in the quadrature weights, so both residuals appear to converge exponentially.

Apparently, for this problem with large Bi and smaller values of n, the errors due to the boundary condition treatment are less important than other errors in the approximation, e.g. the residuals in Fig. 3.32 which are largest near the center. For engineering accuracy, the boundary condition treatment makes little difference for this problem. This result is quite different from the behavior for other problems tested. Even for small n and loose error tolerance, other problems exhibit orders of magnitude improvement using the natural boundary condition approximation. When you consider the natural treatment is easy to apply, often better and never worse, it must be the preferred method.

It may seem that we are belaboring the point regarding the treatment of flux boundary conditions. Texts and articles have continued to recommend boundary collocation [Bert and Malik (1996), Bellomo (1997), Trefethen (2000), p. 137; Boyd (2000), p. 111, Peyret (2002), p. 59], when it was shown to give poor results many years before those writings. Again, this issue does not apply to Gauss points because W_{n+1} = 0. Although many claim benefits for a nonzero boundary weight, it is not an asset when boundary collocation is used. In the OC literature Gauss points have become the preferred choice since it avoids this issue altogether.

Although a natural boundary condition treatment was first suggested more than 40 years ago [Young (1977)] and later in some of the spectral literature [Canuto, *et al*. (1988,2006), Funaro (1992), Shen and Tang (2006)], it appears this alternative has not caught on. A stronger case for this approach was made in more recent work [Young (2019)]. Apparently, the benefits of a natural boundary condition treatment in conjunction with the collocation method are not known

to most practitioners. Most PS (pseudospectral) and DQ (differential quadrature) applications continue to make the mistake of using boundary collocation with Chebyshev and Lobatto points. The problem is easily avoided by using a natural treatment of flux boundary conditions instead. If one is dead set on the use of boundary collocation, then Gauss points should be used. The problems investigated so far in this monograph have shown improvements in flux calculations, but the problem in section 3.2 shows significant improvements in the internal profiles also. Chapter 4 on parabolic problems further emphasizes the pitfalls of using boundary collocation.

3.1.6 Relationship to the Tau Method

Section 1.2.7 of the introduction briefly discusses the relationship between the tau method and MWR. The nature of the residual is analyzed here in greater detail. First, consider the residual, Eq. (3.8). If the problem is linear with constant coefficients, the residual is a polynomial of degree n + 1. For the problem with variable coefficients the cubic variation in Eq. (3.32) causes the residual to be a polynomial of degree n + 4. If a symmetric treatment is used, the residual, Eq. (3.43), is a polynomial of degree n in x^2 for a linear first order source and degree $2n$ in x^2 for a second order source. For other nonlinear problems with $K_a > 0$, the residual is not a polynomial.

The profiles in Fig. 3.1 are symmetric and collocation at Gauss points is equivalent to the method of moments, while collocation at Lobatto points is the same as the Galerkin method. The method of moments makes the residual orthogonal to the first n polynomials, i.e. through degree n – 1, so the residuals for the Gauss case shown in Figs. 3.4 and 3.6 are proportional to Legendre polynomials P_4 and P_6, respectively. The Galerkin method makes the residual orthogonal to the trial functions. The trial functions represent all polynomials through degree n + 1 with the multiplying factor 1 – x^2 to insure they obey essential boundary conditions. The Jacobi polynomials with $\alpha = \beta = 1$ are orthogonal with respect to this same multiplying factor. Since the residual is orthogonal to the first n trial functions (thru degree n – 1), the residuals in Figs. 3.4 and 3.6 for the Lobatto case are proportional to $P_4^{(1,1)}$ and $P_6^{(1,1)}$, respectively. Collocation at Chebyshev points is equivalent to a Chebyshev-Galerkin method, i.e. with the additional weight factor $1/\sqrt{1-x^2}$, so the residuals for the Chebyshev case are proportional to the Chebyshev polynomials $P_n^{(0.5,0.5)}$.

In addition to the τ coefficients, two residual norms are considered: the L_2 norm and a weighted L_ω norm. These norms are defined by:

$$L_2 = \sqrt{\frac{\int R^2 dx}{\int dx}}$$
$$L_\omega = \sqrt{\frac{\int R^2 \omega(x) dx}{\int \omega(x) dx}} \qquad (3.55)$$

where $\omega(x) = (1-x)^{\alpha}(1+x)^{\beta}$ are the weighting functions for the Jacobi polynomials in Eq. (2.7) and $\alpha = \beta$ = 0, ½, and 1 for Gauss or moments, Chebyshev and Lobatto or Galerkin methods, respectively. For cylindrical and spherical geometry, the geometric parameters, x^{γ}, (on [0,1] domain) would be included in the first integral. Table 2.6 gives examples of transforms with the polynomials for symmetric problems. In the second integral, the geometric parameter is accounted for by the β parameter of the weight function, see Eq. (2.6) and Table 2.1.

The residuals can be represented by an equation like Eq. (1.11):

$$R(x) = \sum_{k=n_1}^{n_2} \tau_k P_k^{(\alpha,\beta)}(x) \tag{3.56}$$

Where, as discussed above, $n_1 = n_2 = n$ for the linear constant coefficient problem, residuals in Figs. 3.4 and 3.6. The τ coefficients can be calculated by a Jacobi transform as described in section 2.9. Due to the relationships between polynomials, the method of moments will also make the residual orthogonal to other Jacobi polynomials of lower degree. For example, due Eq. (2.42), the method of moments will make the residual orthogonal to all the $P_k^{(1,1)}$ starting two degrees lower. The residuals will also be orthogonal to the $P_k^{(1,0)}$ and $P_k^{(0,1)}$ starting one degree lower due to Eqs. (2.83) and (2.88).

Using the orthogonality of the polynomials, the weighted norm can be calculated by:

$$L_{\omega} = \sqrt{\frac{1}{\zeta_0^{(\alpha,\beta)}} \sum_{k=n_1}^{n_2} \tau_k^2 \zeta_k^{(\alpha,\beta)}} \tag{3.57}$$

Since the weight function, $\omega(x)$ = 1 for Legendre polynomials, the two norms are the same. Eq. (2.7) shows the polynomial integral terms, $\zeta_n^{(\alpha,\beta)}$, are O(1/n) for large n, so each successive term's contribution to the weighted norm declines at a slightly faster rate than given by the τ coefficient values.

Table 3.7 lists the τ coefficients and the L_2 and L_{ω} residual norms for the first order problem with constant coefficients. For consistency, the results are reported for the domain [-1,1] rather than [0,1]. Also, rather than using the traditional Chebyshev scaling, the Chebyshev polynomials are normalized like the other Jacobi polynomials, Eqs. (2.7) and (2.11). Since the polynomials are different for each case, the τ coefficients and L_{ω} columns cannot be directly

Table 3.7 Residual Properties, 1st Order Constant Coefficients

	Gauss/Moments		Chebyshev			Lobatto/Galerkin		
n	τ	$L_2 = L_{\omega}$	τ	L_2	L_{ω}	τ	L_2	L_{ω}
4	8.4232	2.8077	4.1180	2.7097	2.0199	2.2514	2.9066	1.5179
6	1.4184	0.3934	0.7099	0.4159	0.2953	0.3810	0.5041	0.2254
8	0.1428	0.0346	0.0723	0.0387	0.0265	0.0386	0.0518	0.0206

compared for the different methods. The L_2 norms can be compared. The Galerkin method has the largest values for this norm even though it has the smallest solution errors, see Figs. 3.2 and 3.3.

For the linear variable coefficient problem, Fig. 3.18 shows the flux errors for all of the approximations considered. For the Galerkin and moments approximations, see Figs. 3.15 and 3.16, $n_1 = n$ and $n_2 = n + 4$. For this problem collocation is only an approximation of the Galerkin and moments methods because the quadrature is not accurate enough for an exact correspondence. Two additional quadrature points are required for exact integration. However, with collocation the quadrature is accurate enough to make the residual orthogonal to polynomials of lower degree such that $n_1 = n - 4$. Table 3.3 shows the most accurate results were achieved when one additional quadrature point was used. In that case, $n_1 = n - 2$. The other option considered for this problem was to interpolate the coefficients into the trial space, for which $n_1 = n - 4$. To designate the accuracy of the integration, we use the notation, e.g. *"+1"* to indicate one additional quadrature point, i.e. $m = n + 1$ in Eq. (3.49).

For the variable coefficient problem, Fig 3.40 shows the absolute value of coefficients when $n = 6$ for collocation at Lobatto points, the Galerkin method with exact integration, and other approximate Galerkin methods. A graph like this for Gauss points and moments methods looks remarkably similar. The coefficients for the first two terms are identically zero in all cases as well as those for $n > n_2 = 10$. The coefficients for each degree are similar. Only the 5th or $n - 1$ degree term is significantly different for the collocation method. This term is identically zero for the Galerkin method and small for the other approximations. The n^{th} term is the largest, while the $n + 1$ degree terms are the only others within an order of magnitude of the largest one.

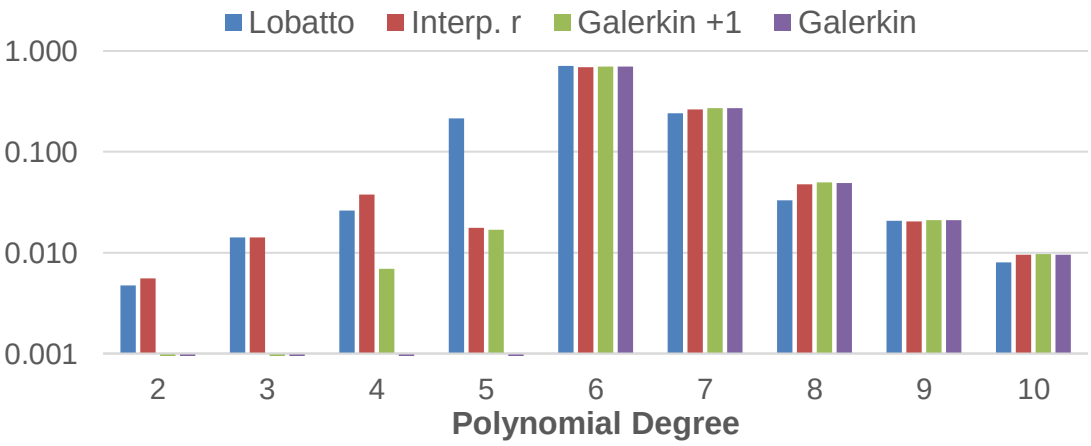


Fig. 3.40 Tau coefficients $n = 6$, linear variable coefficient problem

Flux errors for the nonlinear second order rate expression are shown in Fig. 3.27. The coefficients for it look qualitatively like those in Fig. 3.40. Only the coefficients for n and $n + 1$ terms ($2n$ and $2n + 2$ degree) have significant magnitude when the Galerkin method is approximated with at least one extra quadrature point. The coefficient for the n – 1 term ($2n$ –

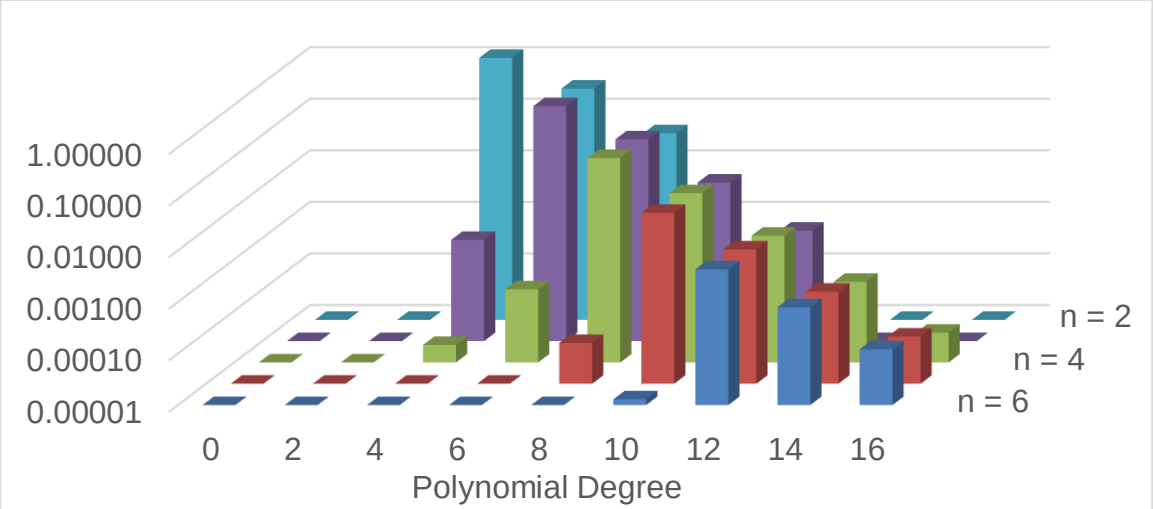


Fig. 3.41 Tau coefficients, Galerkin +1 n = 2 - 6, Second order rate, k = 2, K_a = 0

2 degree) is also significant for the collocation method. Fig. 3.41 shows the evolution of the coefficients for the second order rate expression approximated with Galerkin methods n = 2 to 6 using $m = n + 1$ interior quadrature points. The logarithmic scale shows the coefficients fall off exponentially from a maximum at $2n$. An exact Galerkin method would make the $2n - 2$ term zero. For this approximate integration, those terms are not zero, but they are nearly three orders of magnitude smaller than the $2n$ degree term, so they contribute little to the residual norm.

The other nonlinear rate expression, k = 1, K_a = 0.5, makes for a more difficult problem when φ^* = 5. The flux errors are plotted in Fig. 3.28. In this case, the residual is not a polynomial. Fig. 3.42 shows the τ coefficients for various Galerkin approximations with n = 8. In this case the residual is more highly populated with significant coefficients. The largest coefficients are

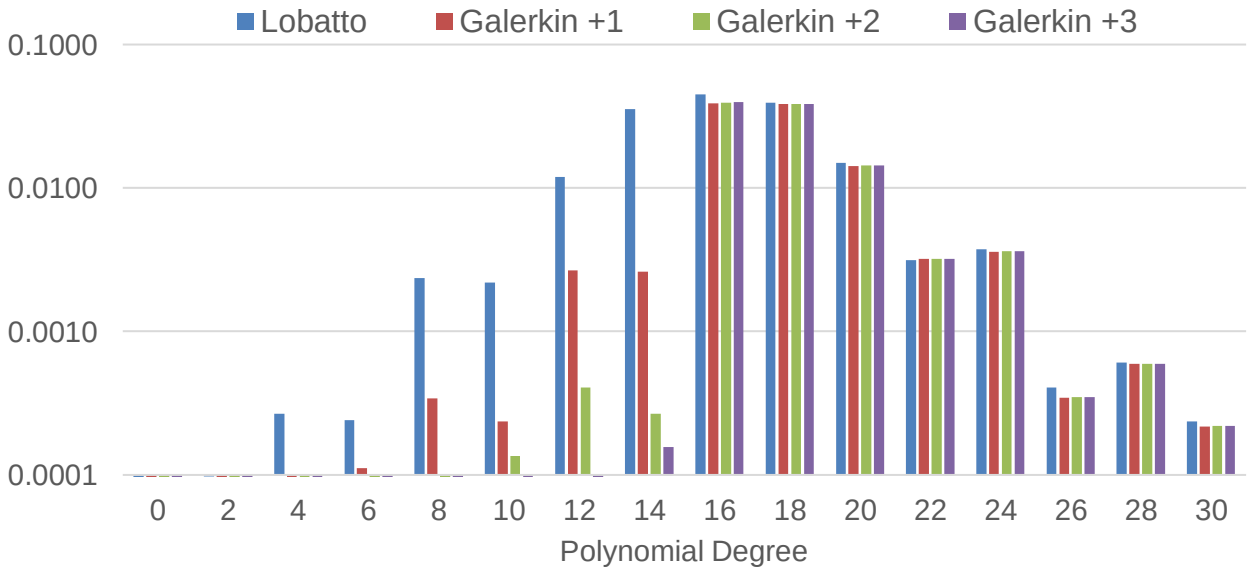


Fig. 3.42 Tau coefficients, Galerkin methods, n = 8, nonlinear rate k = 1, K_a = 0.5

again at $2n$ and $2n$ + 2 (16 and 18 degree). The $2n$ + 4 (20 degree) term is also significant, but the higher terms are smaller than the largest by at least an order of magnitude. In this case the collocation method also has significant values at $2n$ – 2 and $2n$ – 4 (12 and 14). The effect of the additional quadrature points on the coefficients below $2n$ is readily apparent. Results for the autocatalytic reaction, k = 1, K_a = 0.95, are like those in Fig. 3.42, showing a larger array of significant coefficients.

These calculations do not answer the question – why does only one extra quadrature point often produce smaller errors than a more accurate integral approximation with a larger number of quadrature points? This effect has been observed with finite element methods. Strang and Fix (1973) offer the less than satisfactory explanation that exact integration makes the problem "too stiff". The differences due to quadrature accuracy are relatively small, so this question is just a curiosity and not of primary importance. However, keep in mind that the absolute value of the coefficients are plotted in the figures above. The value of the L_ω norm does not depend on the sign of the coefficients. However, we have observed differences between error norms and flux errors. The modal solutions in the next section provide some insight into the nature of flux errors.

It is obvious from these calculations that the MWR methods could be viewed as τ methods, since the residuals can be represented by Eq. (1.11) or (3.56). The Galerkin method, $\alpha = \beta$ = 1, has been emphasized in these calculations, but the same relationships apply to the method of moments, $\alpha = \beta$ = 0. The relationships also apply to Chebyshev versions of the Galerkin method and method of moments with additional weighting by the radical $1/\sqrt{1-x^2}$, so that $\alpha = \beta$ = +½ and -½, respectively. Even orthogonal collocation methods obey the equation, since they are MWR methods with integrals approximated with quadratures. For each method, the residual is orthogonal to a different type of orthogonal polynomial. As a result, the distribution of the residual errors is different as shown in Fig. 2.2. Larger values of α and β reduce the residual away from the boundary at the expense of values near the boundary. The Galerkin method, $\alpha = \beta$ = 1, uses the largest values considered. It deemphasizes the boundary area, so does not produce the smallest residuals, but at least for this problem, it gives the most accurate solutions. At a boundary, the solution is tied to the boundary condition, so a reduced weighting near the boundary makes sense intuitively.

Many texts emphasize the importance of using orthogonal polynomial trial functions, whereas the emphasis should be placed on the weight or test functions, i.e. the type of polynomials the residual is made orthogonal to. It is easy to prove that the form of the trial functions - nodal, modal or monomial - makes no difference, apart from possibly rounding errors. So far, all the examples have been solved using nodal trial functions, while the following sections solves a few problems with modal trial functions. Although the form of the trial functions is not of primary importance, the residual orthogonality is very important. The orthogonality of the residuals is at the heart of the different MWR. This section highlights these differences and illustrates their influence on the residual errors.

3.1.7 Orthogonal Polynomial - Modal Trial Functions

The choice of trial functions is discussed in Section 1.1 and the justification for using Lagrange interpolating polynomials rather than modal methods is given there. For problems of interest, interpolating polynomials rather than orthogonal polynomials are usually the best choice. For completeness, a couple of examples are considered here using orthogonal polynomial trial functions (modal methods) to give the reader a flavor of the differences between the two approaches.

The problem of diffusion with a linear source and Dirichlet boundary conditions, Eqs. (3.1) with (3.1a) and (3.2), is considered with constant and variable coefficients. We do not consider Chebyshev polynomials here, but their use is well documented in the references (e.g. Shen, et al. (2011)). For convenience, the domain [-1,1] is used. The factor of 4 used in the rate function is not required for the same problem with this larger domain.

We wish to use Legendre polynomials as trial functions, so the solution is approximated by Eq. (1.2), duplicated here:

$$\tilde{y} = \sum_{k=0}^{n+1} P_k(x)\,\hat{a}_k \tag{3.58}$$

where n corresponds to the number of interior points in a nodal method. The method is called a modal method, because the unknown coefficients, $\hat{\boldsymbol{a}}$, are equivalent to the modes in a Fourier series.

As discussed in section 2.1, the even numbered polynomials are even functions of x, i.e. symmetric about $x = 0$, while the odd numbered ones are odd. If the problem is symmetric like those considered in section 3.1.5, only even number polynomials would be used in the approximation, Eq. (3.58). The first few Legendre polynomials are given in Eq. (2.26) while the others can be built up using the three term recurrence relations discussed in section 2.1.

These polynomials are orthogonal on the interval [-1,1] with a weight function of unity. They correspond to a Jacobi polynomial with $\alpha = \beta = 0$. There are several ways one could treat the boundary conditions.

$$y(\pm 1) = 0$$

One method is to add constraint equations to explicitly enforce them, while another method is to select combinations of the orthogonal polynomials which will meet the boundary conditions [Boyd (2000)]. Shen, et al. (2011) describe this later approach for a general boundary condition, including 1st, 2nd and 3rd type as special cases. That approach is used here for this simple problem. Legendre polynomials are normalized so the boundary values are:

$$P_k(1) = 1 \quad \text{and} \quad P_k(-1) = (-1)^k$$

Since the polynomials are alternately odd and even about $x = 0$, the simplest method to meet the boundary conditions is to subtract $P_0 = 1$ from the even numbered polynomials in Eq. (3.58)

and subtract $P_1 = x$ from the odd numbered ones. A better matrix structure is achieved if Eq. (3.58) is modified to:

$$\tilde{y} = \sum_{k=0}^{n-1} \big(P_k(x) - P_{k+2}(x)\big)\, a_k = \sum_{k=0}^{n-1} \psi_k(x)\, a_k \tag{3.59}$$

If the problem is symmetric, only the even numbered polynomials are used, so the trial solution is:

$$\tilde{y} = \sum_{k=0}^{n-1} \big(P_{2k}(x) - P_{2k+2}(x)\big)\, a_k = \sum_{k=0}^{n-1} \psi_k(x)\, a_k \tag{3.60}$$

Villadsen and Michelsen (1978) considered this problem for a symmetric solution in cylindrical geometry. They expanded the solution in terms of Jacobi polynomials. A similar approach for this geometry would use Jacobi polynomials with α = β = 1, and a multiplier of (1 – x^2) enforces the boundary conditions. These trial functions are equivalent to those in Eq. (3.59), since Eq. (2.42) is the identity:

$$(1-x^2)P_k^{(1,1)}(x) = \frac{2k+2}{2k+3}\big(P_k(x) - P_{k+2}(x)\big) \tag{3.61}$$

Here we follow the same convention used in Chapter 2. A superscript is used to designate the α and β of a Jacobi polynomial, e.g. (1,1) designates α = β = 1, while no superscript indicates a Legendre polynomial, α = β = 0.

Given the trial functions, $\boldsymbol{\psi}$, the residual of Eq. (3.1) with first order source and constant coefficients is:

$$R(x, \boldsymbol{a}) = \sum_{k=0}^{n-1} [\psi_k''(x) - \varphi^2 \psi_k] a_k + \varphi^2 \tag{3.62}$$

Substituting the trial functions, Eq. (3.59), and then Eq. (2.45) for the second derivative, the residual is:

$$\begin{aligned} R(x, \boldsymbol{a}) &= \sum_{k=0}^{n-1} [P_{2k}'' - P_{2k+2}'' - \varphi^2 (P_{2k} - P_{2k+2})] a_k + \varphi^2 \\ &= -\sum_{k=0}^{n-1} \left[\sum_{\ell=0}^{k} S_{k\ell} P_{2\ell} + \varphi^2 (P_{2k} - P_{2k+2}) \right] a_k + \varphi^2 \end{aligned} \tag{3.63}$$

where $\boldsymbol{S}$ is constructed from Eq. (2.45), $S_{k\ell} = (4k+3)(4\ell+1)$.

With collocation the residual, Eq. (3.63), is set to zero at n interior collocation points. The resulting problem is a full n x n matrix problem. Solution of the matrix problem produces coefficients, a_k, which give exactly the same nodal values as the nodal solutions in Section

3.1.1. The effort required to solve the problem is like that of a nodal formulation. It turns out though, that if an integrated MWR is applied, simplifications sometimes occur.

Method of Moments

Section 1.2.4 briefly described the method of moments and Section 3.1.1 developed a nodal formulation for the problem considered here. Now we wish to use the trial functions in Eq. (3.59) rather than interpolating polynomials. The weight functions with the method of moments are nominally the monomial powers of x. However, as explained in sections 1.2.4 and 3.1.1 any linearly independent set of polynomials which contains the monomials is equivalent. Sections 1.2.7 and 3.1.6 show the equivalence of the method of moments and the Legendre-tau method. The only difference is that here the trial functions have been constructed to meet the boundary conditions, while with the spectral-tau method side conditions are normally used to enforce the boundary conditions. The end result is the same, but the construction here produces a better matrix structure.

For a problem with symmetry, the natural choice of weight functions for a modal method are the even numbered Legendre polynomials:

$$\begin{aligned}&\int_{-1}^{1}\left\{\sum_{j=0}^{n-1}\left[\sum_{\ell=0}^{j}S_{j\ell}P_{2\ell}+\varphi^2\left(P_{2j}-P_{2j+2}\right)\right]a_j-\varphi^2\right\}P_{2i}(x)dx\\&=\sum_{j=0}^{n-1}\left[(4j+3)(4i+1)+\varphi^2\left(\delta_{ij}-\delta_{i,j+1}\right)\right]\zeta_{2i}a_j-\zeta_0\varphi^2\delta_{i0}=0\end{aligned}\tag{3.64}$$

i = 0,…,n - 1. Eq. (3.64) may be written in matrix notation as:

$$[\boldsymbol{C}+\varphi^2\boldsymbol{D}]\boldsymbol{a}=\varphi^2\boldsymbol{h}\tag{3.65}$$

We identify $\boldsymbol{C}$ and $\boldsymbol{D}$ as the stiffness and mass matrices, respectively, and $\boldsymbol{h}$ as the load vector. Substituting the values of ζ from Eq. (2.7), the approximation is:

$$\begin{aligned}C_{ij}&=\zeta_{2i}(4j+3)(4i+1)=\ 2(4j+3)\ \text{ for } j\geq i\\D_{ij}&=\frac{2}{4i+1}\left(\delta_{ij}-\delta_{i,j+1}\right)\\h_i&=2\delta_{0i}\end{aligned}$$

The common factor of 2 can also be eliminated. The stiffness matrix, $\boldsymbol{C}$, is then upper triangular, while the mass matrix, $\boldsymbol{D}$, consists of a diagonal and one subdiagonal. The load vector, $\boldsymbol{h}$, is integrated by substituting P_0 = 1 and only the first row has a nonzero value. The first 6 rows of the matrices are:

$$\boldsymbol{D}=\begin{pmatrix}1&0&0&0&0&0\\-1/5&1/5&0&0&0&0\\0&-1/9&1/9&0&0&0\\0&0&-1/13&1/13&0&0\\0&0&0&-1/17&1/17&0\\0&0&0&0&-1/21&1/21\end{pmatrix}$$

$$\boldsymbol{C} = \begin{pmatrix} 3 & 7 & 11 & 15 & 19 & 23 \\ 0 & 7 & 11 & 15 & 19 & 23 \\ 0 & 0 & 11 & 15 & 19 & 23 \\ 0 & 0 & 0 & 15 & 19 & 23 \\ 0 & 0 & 0 & 0 & 19 & 23 \\ 0 & 0 & 0 & 0 & 0 & 23 \end{pmatrix}$$

When combined the matrix is upper triangular except for the addition of the subdiagonal from the mass matrix. The matrix problem can be solved more efficiently than a full matrix, since only the subdiagonal needs to be eliminated and back substitution is simpler due to the common coefficients. Some sample solutions with φ = 5, are:

$$a = \{0.892857\} \quad \text{for } n = 1$$
$$a = \{0.808625, 0.336927\} \quad \text{for } n = 2$$
$$a = \{0.800214, 0.281414, 0.056737\} \quad \text{for } n = 3$$
$$a = \{0.800020, 0.280131, 0.050260, 0.005711, 0\} \quad \text{for } n = 4$$

When Eq. (3.59) is evaluated at the Gauss points with these coefficients for n = 2, the values are identical to those found in Fig. 3.1 of Section 3.1.1 with an even number of Gauss points (which is equivalent to moments). The rapid decay of the coefficient values and the rapid convergence of the lower order coefficients, indicates the fast convergence of the solution with increasing n. This convergence behavior has important consequences for the accuracy of the boundary flux, Eq. (3.4).

Integration of the approximate solution, Eq. (3.59), gives the normalized flux:

$$\eta = \frac{1}{2}\int_{-1}^{1}(1-y)dx = 1 - \frac{1}{2}\sum_{k=0}^{n-1} a_k \int_{-1}^{1}\big(P_{2k}(x) - P_{2k+2}(x)\big)dx = 1 - a_0 \tag{3.66}$$

The analytical solution, Eq. (3.6), gives the exact value $\eta = \tanh(\varphi)/\varphi$ or a_0 = 0.800018 for φ = 5. Normally, the derivatives of an approximate solution converge slower than the solution itself. However, the boundary flux is a special case, since integration can be used as explained in the discussion of Eq. (3.5). When solved with the moments method the boundary flux converges more quickly than the overall solution. This behavior is clearly demonstrated by comparison of the L_2 error norms and flux errors, see Figs. 3.2 and 3.3.

Galerkin Method

Now consider the solution of this problem with the trial functions, Eq. (3.59), giving the same residual, Eq. (3.62). However, with weighting by the trial functions, the weighted residual is modified to:

$$\int_{-1}^{1} R(x, \boldsymbol{a})(P_{2i} - P_{2i+2})dx = 0 \tag{3.67}$$

By comparing Eqs. (3.64) and (3.67), a simple relationship between the Galerkin and moments approximations is apparent. Both the mass matrix and stiffness matrix are related as follows:

$$C_{ij}^{G} = C_{ij}^{M} - C_{i+1,j}^{M} \tag{3.68}$$

where the superscripts, G and M, indicate the matrices for the Galerkin and moments methods, respectively. The resulting matrices for a six term approximation are:

$$\boldsymbol{D^G} = \begin{pmatrix} 6/5 & -1/5 & 0 & 0 & 0 & 0 \\ -1/5 & 14/45 & -1/9 & 0 & 0 & 0 \\ 0 & -1/9 & 22/117 & -1/11 & 0 & 0 \\ 0 & 0 & -1/11 & 30/221 & -1/15 & 0 \\ 0 & 0 & 0 & -1/15 & 38/357 & -1/19 \\ 0 & 0 & 0 & 0 & -1/19 & 46/525 \end{pmatrix}$$

$$\boldsymbol{C^G} = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 11 & 0 & 0 & 0 \\ 0 & 0 & 0 & 15 & 0 & 0 \\ 0 & 0 & 0 & 0 & 19 & 0 \\ 0 & 0 & 0 & 0 & 0 & 23 \end{pmatrix}$$

Compared to the moments method, these modifications produce a matrix problem which can be solved more efficiently, especially for large n. The stiffness matrix is diagonal, while the mass matrix is tridiagonal.

One nice feature of the modal formulation is that it is easier to calculate the solution for different n in order to monitor the convergence. The matrices can be calculated for the maximum n of interest, which gives the matrices for smaller n as just the upper left submatrix of the larger matrix. For example, the solution for n = 3 can be obtained by using the upper left 3x3 submatrix of the larger matrix. For a nodal formulation, the matrix problem is different for each n, so monitoring convergence requires the calculation of a new matrix for each n of interest.

Some example solutions for φ = 5, are:

$$a = \{0.757576\} \quad \text{for } n = 1$$

$$a = \{0.798511, 0.270173\} \quad \text{for } n = 2$$

$$a = \{0.799998, 0.279987, 0.049535\} \quad \text{for } n = 3$$

$$a = \{0.800018, 0.280119, 0.050201, 0.005249\} \quad \text{for } n = 4$$

When Eq. (3.59) is evaluated at the Lobatto points with these coefficients for n = 2, the values are identical to those found in Fig. 3.1 of section 3.1.1, since collocation at Lobatto points is equivalent to the Galerkin method.

The convergence rate of these coefficients is somewhat better than that for the moments method, so the comments above regarding the accuracy of flux calculations apply equally to the Galerkin method.

The equations for the method of moments and Galerkin method can be solved with φ as a free parameter. For the method of moments, the first four approximations are:

$$\eta = \frac{3}{\varphi^2 + 3} \quad \text{for } n = 1$$

$$\eta = \frac{10\varphi^2 + 105}{\varphi^4 + 45\varphi^2 + 105} \quad \text{for } n = 2$$

$$\eta = \frac{21\varphi^4 + 1260\varphi^2 + 10395}{\varphi^6 + 210\varphi^4 + 4725\varphi^2 + 10395} \quad \text{for } n = 3$$

$$\eta = \frac{36\varphi^6 + 6930\varphi^4 + 270270\varphi^2 + 2027025}{\varphi^8 + 630\varphi^6 + 51975\varphi^4 + 945945\varphi^2 + 2027025} \quad \text{for } n = 4$$

The first four Galerkin approximations are:

$$\eta = \frac{\varphi^2 + 15}{6\varphi^2 + 15} \quad \text{for } n = 1$$

$$\eta = \frac{\varphi^4 + 105\varphi^2 + 945}{15\varphi^4 + 420\varphi^2 + 945} \quad \text{for } n = 2$$

$$\eta = \frac{\varphi^6 + 378\varphi^4 + 17325\varphi^2 + 135135}{28\varphi^6 + 3150\varphi^4 + 62370\varphi^2 + 135135} \quad \text{for } n = 3$$

$$\eta = \frac{\varphi^8 + 990\varphi^6 + 135135\varphi^4 + 4729725\varphi^2 + 34459425}{45\varphi^8 + 13860\varphi^6 + 945945\varphi^4 + 16216200\varphi^2 + 34459425} \quad \text{for } n = 4$$

The relationships for n = 2 reproduce the results plotted in Fig. 3.20. These expressions agree with the behavior in Fig 3.20. i.e. for moments $\lim_{\varphi \to \infty} \eta \propto \varphi^{-2}$, while for the Galerkin method $\lim_{\varphi \to \infty} \eta \propto 1$.

The Padé approximant is the “best” approximation of a function by a rational function of a given order. These solutions are identical to Padé approximations calculated directly from the analytical solution $\eta = \tanh(\varphi)/\varphi$ to the same order in the numerator and denominator. Compared to the method of moments, the solutions for the Galerkin method have one additional term in the numerator. The extra order in the numerator is the reason it produces slightly better results.

These results remind one of Lanczos’ (1956) motivation for developing the tau method and method of selected points (collocation). His interest was in developing approximations to various functions. These calculations could be used as a somewhat roundabout procedure for developing accurate approximations to the hyperbolic tangent function. The Galerkin approximation for n = 4 is accurate to within 0.1% for φ < 11.

Variable Coefficients

The banded matrix structure given by the Galerkin method above and even the near upper triangular matrix produced by the moments method can be solved more efficiently than the full matrix problems produced by the nodal approximations resulting from Lagrange interpolating polynomials. Unfortunately, these efficient matrix structures are lost if the problem is nonlinear

or even if it has variable coefficients. To demonstrate this property, consider the variable coefficient problem with source function in Eq. (3.32), which is linear in y, but cubic in x. The cubic function $q(x)$ varies from 0.2 to 1.8, but with an average value of unity (note the factor of 4 is dropped from the equation because the domain here is [-1,1]). To facilitate solution of the problem, the variable coefficient is expressed as a function of Legendre polynomials. One way to determine these coefficients is through a discrete Legendre transform as described in section 2.9. The result is:

$$q(x) = P_0 + 0.96P_1 - 0.16P_3$$

In this form it is clear from the properties of Legendre polynomials that the average value is unity and the values at the two boundaries are as stated above.

Eq. (3.65) still applies for this problem. The stiffness matrices are the same as before, but since the problem is no longer symmetric, the rows and columns associated with the odd numbered polynomials must be included. So, the stiffness matrix for the method of moments is:

$$\boldsymbol{C} = \begin{pmatrix} 6 & 0 & 14 & 0 & 22 & 0 \\ 0 & 10 & 0 & 18 & 0 & 26 \\ 0 & 0 & 14 & 0 & 22 & 0 \\ 0 & 0 & 0 & 18 & 0 & 26 \\ 0 & 0 & 0 & 0 & 22 & 0 \\ 0 & 0 & 0 & 0 & 0 & 26 \end{pmatrix}$$

The mass matrix and load vector change due to the cubic coefficient. For the method of moments, these quantities are:

$$D_{ij} = \int_{-1}^{1} P_i(x)\,\psi_j(x)\,q(x)\,dx = \int_{-1}^{1} P_i(x)\,\Big(P_j(x) - P_{j+2}(x)\Big)\,q(x)\,dx$$

$$h_i = \int_{-1}^{1} P_i(x)\,q(x)\,dx$$

The load vector can easily be integrated analytically when q is expressed in terms of the Legendre polynomials. Integration of the mass matrix analytically is more complicated. One possibility is to interpolate the product of q and ψ as a discrete Legendre series and then integrate. It is probably simpler just to use numerical quadrature to integrate the expressions as was done when this problem was solved by nodal methods. Sections 3.1.3 and 3.1.4 the degree of the mass matrix integrand. Exact integration for the Galerkin method is obtained using Lobatto quadrature with n + 2 interior points. For the method of moments n + 2 interior points is needed for Gaussian quadrature or n + 1 for Lobatto quadrature.

Exact integration of the Galerkin mass matrix and load vector gives:

$$D^G = \begin{bmatrix} 2.4000 & 0.6095 & -0.4000 & -0.2119 & 0 & 0.0139 \\ 0.6095 & 0.9524 & 0.2101 & -0.2857 & -0.1446 & 0 \\ -0.4000 & 0.2101 & 0.6222 & 0.1432 & -0.2222 & -0.1131 \\ -0.2119 & -0.2857 & 0.1432 & 0.4675 & 0.1090 & -0.1818 \\ 0 & -0.1446 & -0.2222 & 0.1090 & 0.3761 & 0.0884 \\ 0.0139 & 0 & -0.1131 & -0.1818 & 0.0884 & 0.3152 \end{bmatrix} \text{ and } h^G$$
$$= \begin{bmatrix} 2.0000 \\ 0.6857 \\ 0 \\ -0.0457 \\ 0 \\ 0 \end{bmatrix}$$

Except for a couple of zeros, this matrix is full, so the beneficial matrix structure found for the constant coefficient problem is lost. A full matrix also results when the source is nonlinear. Solving a nonlinear problem is further complicated by the indirect dependence of the dependent variable, y, on the unknown coefficients, $\boldsymbol{a}$. Nonlinear problems are much easier to solve with a nodal method, and as found in section 3.1.5, collocation is much simpler than the Galerkin or moments methods.

Some examples of solutions using the method of moments with φ = 5 are:

$$a = \{0.780464, 0.139001, 0.290515, 0.136558\} \quad \text{for } n = 4$$

$$a = \{0.765444, 0.154772, 0.232674, 0.105690, 0.061149, 0.026173\} \quad \text{for } n = 6$$

Some solutions with the Galerkin method are:

$$a = \{0.764431, 0.151352, 0.218663, 0.107983\} \quad \text{for } n = 4$$

$$a = \{0.765254, 0.154036, 0.229734, 0.106528, 0.047540, 0.022546\} \quad \text{for } n = 6$$

These solutions are identical to those found for the nodal moments and Galerkin methods shown in Figs. 3.15 and 3.16 and Table 3.2 when converted to a common basis.

Methods to convert between modal and nodal representations are discussed in section 2.9. To demonstrate the conversion, consider the Galerkin solution of the variable coefficient problem with n = 4. The values at the Lobatto points are illustrated in Fig. 3.16. These values can be compared by using the coefficients above to evaluate Eq. (3.59) at the Lobatto points. The result is:

$$y = \{0, 0.46603, 0.92731, 0.97087, 0.79069, 0\}$$

which is identical to the results found using the nodal formulation.

Suppose we had not solved the problem with the modal formulation but want to know the modal coefficients. In that case, a Legendre transformation matrix, discussed in section 2.9, can be used to calculate the coefficients from the nodal values by $\hat{\boldsymbol{a}} = \boldsymbol{Q}\boldsymbol{y}$. The result of the calculation is the coefficients of Eq. (3.58), which are:

$$\hat{a} = \{0.76443, 0.15135, -0.54577, -0.04337, -0.21866, -0.10798\}$$

These coefficients can be determined from those above (for Eq. (3.59)), by collecting like terms.

If we have a modal approximation, Eq. (3.65), but want to calculate the nodal values directly, we could do so by substitution of the appropriate transform:

$$[\boldsymbol{C} + \varphi^2 \boldsymbol{D}]\boldsymbol{Q}\boldsymbol{y} = \varphi^2 \boldsymbol{h} \tag{3.69}$$

In this case, since the coefficients are for the difference, $P_n - P_{n+2}$, the appropriate transform would be like the intermediate one, Eq. (2.144).

The point of this discussion is that the solution is exactly the same, but there are different ways to represent it. Using one form of trial function or another does not change the solution, though some texts seem to suggest that it does.

3.2 Chemical Reactor with Axial Dispersion

For the next example of a boundary value problem, consider nonisothermal flow through a chemical reactor with axial conduction and dispersion. This is a coupled heat and mass transfer problem. The governing equations for the problem are:

$$\begin{aligned} &\frac{1}{Pe_m}\frac{d^2y}{dz^2} - \frac{dy}{dz} + r_m(y,T) = 0 \quad \text{and} \\ &\frac{1}{Pe_t}\frac{d^2T}{dz^2} - \frac{dT}{dz} - UT + r_t(y,T) = 0 \end{aligned} \tag{3.70}$$

with:

$$\begin{aligned} &\frac{dy}{dz} = Pe_m y \quad \text{and} \quad \frac{dT}{dz} = (Pe_t + U)T \quad \text{at } z = 0 \text{ and} \\ &\frac{dy}{dz} = 0 \quad \text{and} \quad \frac{dT}{dz} + UT = 0 \quad at\ z = 1 \end{aligned}$$

The model allows for cooling or heating at the wall using a lumped parameter approximation with an overall heat transfer coefficient, parameter U. These problems are convection dominated, since the Peclet numbers, Pe_m and Pe_t, are normally large. The solution is nonsymmetric and all the boundary conditions are of the second or third kind. Models with heating and cooling often fail to account for this effect in the boundary conditions. The boundary conditions were correctly treated by Young and Finlayson (1973).

In an industrial reactor model several component balances could be required to represent a system of reactions, so Eq. (3.70) is formulated as the following system of coupled equations:

$$\frac{1}{Pe_k}\frac{d^2y_k}{dz^2} - \frac{dy_k}{dz} - U_k y_k + r_k(\boldsymbol{y}) = 0 \tag{3.71}$$

with

$$\frac{dy_k}{dz} = (Pe_k + U_k)y_k \quad \text{at } z = 0 \quad \text{and} \quad \frac{dy_k}{dz} + U_k y_k = 0 \quad at\ z = 1$$

for k = 0,…,n_m. We assign k = 0 to the energy balance which is combined with as many mass balances as required to represent the reaction system of interest.

After the application of conventional orthogonal collocation, Eq. (3.71) is approximated by:

$$\sum_{i=0}^{n+1}\left(\frac{1}{Pe_k}B_{ji} - A_{ji} - U_k\delta_{ji}\right)y_{ik} + r_k(\boldsymbol{y}_j) = 0 \tag{3.72}$$

for j = 1,…,n, and using boundary collocation as recommended by most authors:

$$\sum_{i=0}^{n+1}\left[A_{0,i}-(Pe_k+U_k)\delta_{o,i}\right]y_{ik}=0 \quad \text{and}$$
$$\sum_{i=0}^{n+1}\left[A_{n+1,i}+U_k\delta_{n+1,i}\right]y_{ik}=0 \tag{3.72a}$$

where $y_{ik}=[T(z_i),y_1(z_i),\ldots,y_{n_m}(z_i)]$ and $\boldsymbol{y}_j$ is the vector of all values evaluated at point z_j. Based on the results for the previous example, we anticipate that boundary collocation, Eq. (3.72a), will work well for Gauss points but not so well for Lobatto or Chebyshev points.

A natural treatment of the boundary conditions works better for Lobatto and Chebyshev points. It is the standard method for treating flux boundary conditions with Galerkin methods and it falls out when the equations are cast in weak form. The weak form of the Galerkin method is derived by integrating the Laplacian by parts and substituting the boundary conditions for the boundary derivatives. This exercise was carried out for the previous problem in Eqs. (3.22) and (3.23). Then quadrature is used to approximate the integrals as in Eq. (3.24). With a slight generalization in the definition of the stiffness matrix, $\boldsymbol{C}$ in Eq. (3.25), the same formulation can be applied for any type of points. For this problem, the resulting weak form is:

$$\delta_{0,j}\left(1+\frac{U_k}{Pe_k}\right)y_{0,k}+\delta_{n+1,j}\frac{U_k}{Pe_k}y_{n+1,k}+\sum_{i=0}^{n+1}\left(\frac{C_{ji}}{Pe_k}+W_jA_{ji}+U_kW_j\delta_{ji}\right)y_{ik}-W_jr_k(\boldsymbol{y}_j)=0 \tag{3.73}$$

From the previous discussions, we know the weak form, Eq. (3.73), is equivalent to the conventional form, Eq. (3.72), at the interior points, but the boundary conditions are treated differently. In order to compare the two methods for treating the boundary conditions, Eq. (3.25) is substituted for $\boldsymbol{C}$, which gives the boundary equations:

$$\begin{aligned}
&\sum_{i=0}^{n+1}\left[\left(1+\frac{U_k}{Pe_k}\right)\delta_{0,i}-\frac{A_{0,i}}{Pe_k}\right]y_{ik}-W_0\left[\sum_{i=0}^{n+1}\left(\frac{B_{o,i}}{Pe_k}-A_{0,i}-U_k\delta_{0,i}\right)y_{ik}+r_k(\boldsymbol{y}_0)\right]=0\\
&\sum_{i=0}^{n+1}\left[\frac{U_k}{Pe_k}\delta_{n+1,i}+\frac{A_{n+1,i}}{Pe_k}\right]y_{ik}-W_{n+1}\left[\sum_{i=0}^{n+1}\left(\frac{B_{n+1,i}}{Pe_k}-A_{n+1,i}-U_k\delta_{n+1,i}\right)y_{ik}+r_k(\boldsymbol{y}_{n+1})\right]=0
\end{aligned} \tag{3.74}$$

The first term is the boundary condition residual and the second is the interior residual evaluated at the boundary and multiplied by the quadrature weight. This relationship is like that found for the previous example, see Eq. (3.26). Rather than setting one or the other residual to zero, this procedure sets a weighted combination of the two to zero, like Eq. (1.15). Although neither residual will be identically zero, they will converge to zero. For Gauss points, the boundary quadrature weights are zero, so Eq. (3.72a) and (3.74) are equivalent.

When Gauss points are used, the method is an accurate approximation to the method of moments and when Lobatto points are used, it is an accurate approximation to the Galerkin method. In each case the dispersion and convection terms are integrated exactly. If the rate terms are interpolated, like Eq. (3.33), integration of both the rate term and heat transfer term

miss exact integration by one degree. The reaction term is nonlinear, so the accuracy of the approximation depends on the severity of the nonlinearity.

Eq. (3.73) can be expressed in the form:

$$\sum_{i=0}^{n+1} \tilde{A}_{ji}^{k}\, y_{ik} - W_j r_k(\boldsymbol{y}_j) = 0 \tag{3.75}$$

where:

$$\tilde{A}_{ji}^{k} = \frac{1}{Pe_k} C_{ji} + W_j A_{ji} + \delta_{ji}\left[U_k W_j + \delta_{0,j}\left(1 + \frac{U_k}{Pe_k}\right) + \delta_{n+1,j}\frac{U_k}{Pe_k}\right]$$

Eq. (3.72) can be expressed in the same form.

Although the problem is formulated for any number of mass balance equations, we consider only one mass balance together with the energy balance. A first order exothermic reaction is used:

$$r_k = Da_k \hat{r}(y,T) = Da_k(1-y)\exp\left(\frac{20T}{T+1}\right) \tag{3.76}$$

Da are the Damkohler numbers for energy and mass. The parameter values considered in the example are: Pe_t = 100, Pe_m = 200, U = 3, Da_t = 0.2, Da_m = 0.5. This problem is highly nonlinear due to the temperature dependence in Eq. (3.76).

The form of Eq. (3.75) is similar to that of Eq. (3.44), so an iterative solution procedure like Eq. (3.46) can be used. The only difference is that here we have a coupled set of energy and mass balance equations. The coupling poses no difficulty other than creating some bookkeeping issues. Due to the coupling a Newton-Raphson method requires the simultaneous solution of all equations. An alternative procedure which solves the equations sequentially is a Picard iteration:

$$\sum_{i=0}^{n+1}\left(\tilde{A}_{ji}^{k} + \delta_{ji}\mu_{ki}\right)\Delta y_{ik} = W_j r_k\left(\boldsymbol{y}_j^0\right) - \sum_{i=0}^{n+1}\tilde{A}_{ji}^{k}\, y_{ik}^0 \tag{3.77}$$

where the Δ indicates the change in the variable over the iteration and superscript 0 indicates the initial guess or the values from the previous iteration. The μ are iteration parameters. One can think of these parameters as an approximation to the rate derivatives. For proper scaling, we take $\mu_{ki} = a_k W_i Da_k$, where an a is supplied for each balance. Larger values of the iteration parameter make the matrix more diagonally dominant and will tend to reduce the size of changes, which can stabilize the iterations, but may slow down convergence. With this procedure, the matrices are modified by the iteration parameters and then factored, requiring $O(\frac{2}{3}(n_m+1)(n+2)^3)$ operations. Since the matrix does not change from one iteration to the next, the factors need to be calculated only once. Once the matrices are factored, the iterations consist of first calculating the rates and then sequentially for each balance, calculate the right side of Eq. (3.77) and then update the values by a forward and back substitution with the factors of the matrices. The calculations per iteration are $O(4(n_m+1)(n+2)^2)$. This method

normally converges slowly, because it is basically successive substitution and does not account for how the rates change with changes in $\boldsymbol{y}$. However, the iterations require minimal calculation.

The rate terms can be linearized about some initial or intermediate estimates $\boldsymbol{y}^0$ to give the approximation:

$$r_k(\boldsymbol{y}_j) \approx r_k(\boldsymbol{y}_j^0) + \sum_{\ell=0}^{n_m} \left.\frac{\partial r_k}{\partial y_\ell}\right|_{y_j^0} \Delta y_{j\ell} = r_{jk} + \sum_{\ell=0}^{n_m} d_{j\ell}^k \, \Delta y_{j\ell} \tag{3.78}$$

where $\boldsymbol{y}$ is the vector which includes temperature, T, as its first member. This linearization is substituted into Eq. (3.75) and a Newton-Raphson iteration, like Eq. (3.46), can be used to solve the system of equations.

The iterations are then:

$$\sum_{i=0}^{n+1} \left(\tilde{A}_{ji}^k - \delta_{ji} W_j d_{jk}^k\right) \Delta y_{ik} - W_j \sum_{\substack{\ell=0 \\ \ell \neq k}}^{n_m} d_{j\ell}^k \, \Delta y_{j\ell} = W_j r_{jk} - \sum_{i=0}^{n+1} \tilde{A}_{ji}^k \, y_{ik}^0 \tag{3.79}$$

It is usually more accurate to formulate the equations to solve for changes, i.e. $\Delta\boldsymbol{y}$, rather than directly for the updated values.

The resulting matrix for the linearized problem, Eq. (3.79), with 3 interior points is of the form:

$$\begin{bmatrix}
x & x & x & x & x & x & & & & & \\
x & x & x & x & x & & x & & & & \\
x & x & x & x & x & & & x & & & \\
x & x & x & x & x & & & & x & & \\
x & x & x & x & x & & & & & x & \\
x & & & & & x & x & x & x & x & \\
 & x & & & & x & x & x & x & x & \\
 & & x & & & x & x & x & x & x & \\
 & & & x & & x & x & x & x & x & \\
 & & & & x & x & x & x & x & x &
\end{bmatrix}
\begin{bmatrix}
\Delta T_0 \\ \Delta T_1 \\ \Delta T_2 \\ \Delta T_3 \\ \Delta T_4 \\ \Delta y_0 \\ \Delta y_1 \\ \Delta y_2 \\ \Delta y_3 \\ \Delta y_4
\end{bmatrix} \tag{3.80}$$

This equation is a block 2x2 matrix (or in general (n_m+1)x(n_m+1)), where each block is an (n+2)x(n+2) matrix. The diagonal blocks are full and the off-diagonal blocks are diagonal.

The coupling between y and T is through the derivatives of the rate terms in Eq. (3.78), which appear on the diagonals of each block. The Picard iteration neglects this coupling. This procedure, Eq. (3.79), requires solving one large matrix, rather than n_m+1 smaller ones. Also, with the Newton-Raphson iteration, the matrix changes each iteration. Overall, each iteration requires O(⅔(n_m+1)^3(n+2)^3) operations, substantially more than for the Picard iteration. We refer to this procedure as a *full* Newton-Rapson, since all the nonlinear equations are solved together.

The rate dependence in the energy equation can always be expressed as a linear combination of the mass balance equations. This relationship can be used to develop an alternate formulation. We will demonstrate the procedure for n_m = 1, but keep in mind the energy equation can always be eliminated with a similar procedure. The energy and mass balance equations can be combined to give in matrix notation:

$$\widetilde{\boldsymbol{A}}^0\boldsymbol{y}_0 = \frac{Da_0}{Da_1}\,\widetilde{\boldsymbol{A}}^1\boldsymbol{y}_1 \; or \; \boldsymbol{y}_0 = \frac{Da_0}{Da_1}\left(\widetilde{\boldsymbol{A}}^0\right)^{-1}\widetilde{\boldsymbol{A}}^1\boldsymbol{y}_1 = \frac{Da_0}{Da_1}\boldsymbol{G}\boldsymbol{y}_1 \tag{3.81}$$

Then the Newton-Raphson iterations simplify to:

$$\sum_{i=0}^{n+1}\left(\tilde{A}^1_{ji} - \delta_{ji}W_j d^1_{j1} - W_j d^0_{j0}G_{ji}\right)\Delta y_{i1} = W_j r_{j1} - \sum_{i=0}^{n+1}\tilde{A}^1_{ji}\,y^s_{i1} \tag{3.82}$$

This formulation requires more calculations to set up the iterations, since the calculation of $\boldsymbol{G}$ requires roughly four times the calculations required to factor $\widetilde{\boldsymbol{A}}^0$ or O(2⅔$(n$+2)^3), but it requires fewer calculations per iteration since (when n_m = 1) the dimension of the matrix is half that for full Newton-Raphson, Eq. (3.79). With this procedure, Eq. (3.82) is calculated and solved for $\Delta\boldsymbol{y}_1$, then Eq. (3.81) is used to calculate $\Delta\boldsymbol{y}_0$. We will call this the *reduced* procedure, since it reduces the size of the problem by eliminating the temperature dependence.

From the discussion above, it is apparent there are tradeoffs between the iterative procedures discussed. The operation counts discussed above are:

Picard: $W_{PI} = \frac{2}{3}(n_m+1)(n+2)^3 + 4(n_m+1)(n+2)^2 N_{PI}$

Full Newton: $W_{FN} = \frac{2}{3}(n_m+1)^3(n+2)^3 N_{NI}$

Reduced Newton: $W_{RN} = 2\frac{2}{3}(n+2)^3 + n_m^2(n+2)^2\left(\frac{2}{3}n_m(n+2)+8\right)N_{NI}$

Where W and N designate the total operations and number of iterations for the different methods.

The Picard iterations is favored for larger systems, since the computational work grows in direct proportion to n_m rather than to some power. It is also favored for large n. From testing we have found that solution of the algebraic problem to about 6 digits of accuracy requires about 8 Newton-Raphson iterations or 20 Picard iterations. The Picard method typically requires 2½ or more times as many iterations as a Newton method depending on the convergence tolerance. In our experiments, we have not found the number of iterations to be very sensitive to the values of the iteration parameters. For the Picard method, it is surprising that the number of iterations tends to decrease with n. If n is too small, none of the methods converge.

Table 3.8 summarizes operation counts (in thousands of floating point operations, Kflops) for the three iterative procedures described above. The full Newton method is not nearly as efficient as the other two. This problem is small enough that for a single solution, efficiency is not a major issue, so an in-depth analysis is overkill.

Table 3.8 Estimated Kflops for n = 18

Method	Picard	Reduced Newton	Full Newton
Setup	10.7	21.3	0.0
Iteration	3.2	8.5	42.7
total*	74.3	89.6	341.3

*8 iterations for Newton-Raphson, 20 for Picard

However, knowing how to analyze algorithms, without solving all the alternatives, is an important skill to have. Also, there are many cases where the problem could be much larger or could require many solutions. For example, the number of reactions and n_m could be quite large. Radial as well as axial dispersion could be important making the problem two dimensional with many more points. Suppose a small reactor was used to measure reaction rates, and the data needs to be analyzed as an integral reactor using nonlinear regression. In that case the problem would be solved hundreds of times. This type of analysis has been done and efficient methods are essential [Young and Finlayson (1973)].

There are some other techniques that could potentially improve the efficiency of the calculations. Eq. (3.80) is a band matrix, so a band solver would be slightly more efficient. When Gauss points are used, the boundary equations for the first and last points do not depend on the rate, see Eqs. (3.72a) and (3.74). Those two equations can be eliminated at the outset to reduce the number of nonlinear equations which must be solved.

In Newton-Raphson methods, the matrix in Eq. (3.79) or Eq. (3.82) is called the *Jacobian* matrix. Frequently, some efficiency can be gained by updating the Jacobian only periodically, e.g. every other iteration. For our problem calculating and factoring the Jacobian consumes about 70% of the calculations for an iteration, so there is a good potential for improvement. Other solution strategies may be discovered by working with the codes for this problem, e.g. start with a few Picard iterations, then switch to a Newton-Raphson. One could also consider iterative solution methods for the linear algebraic problem and a finite element discretization to create a more sparse matrix problem.

Figs. 3.43, 3.44 and 3.45 show some example profiles with Gauss, Chebyshev and Lobatto points, respectively. In all cases, the solid

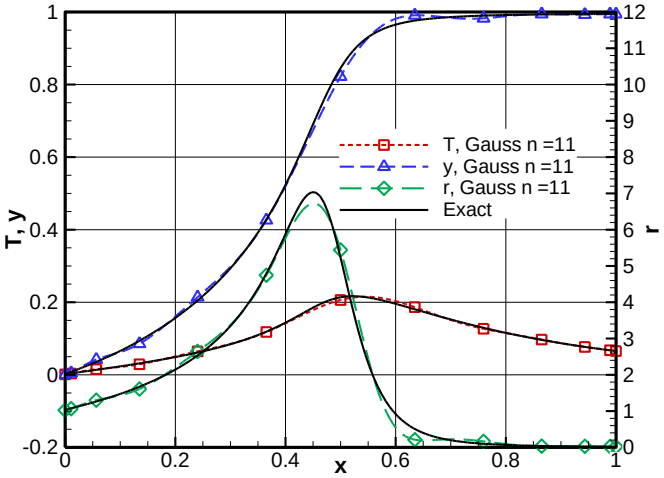


Fig. 3.43 T, y and rate profile, Gauss, n = 11

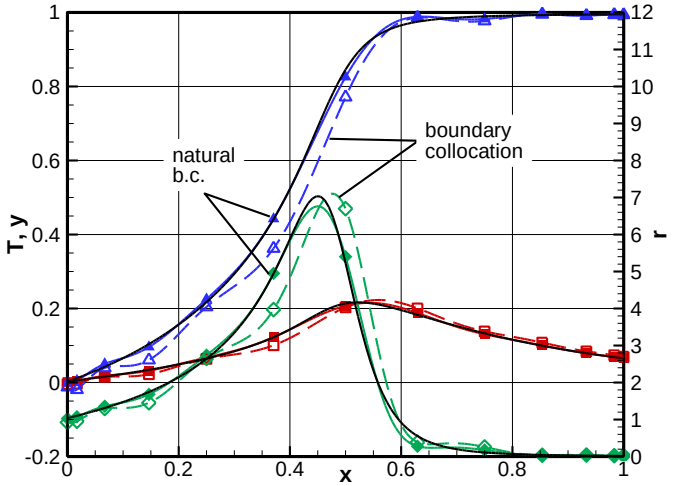


Fig. 3.44 T, y and rate profiles, Chebyshev, n = 11

black line is the solution with n = 51, which is exact for practical purposes. The graphs for Chebyshev and Lobatto points show the solutions for both boundary collocation, Eq. (3.72a), and a natural treatment, Eq. (3.73) or (3.74). In the previous problem, the errors caused by boundary collocation were not obvious when comparing the profiles, see Figs. 3.9 and 3.10, and were only apparent when comparing fluxes. For this problem, the differences are clearly visible in the profiles of Figs. 3.44 and 3.45.

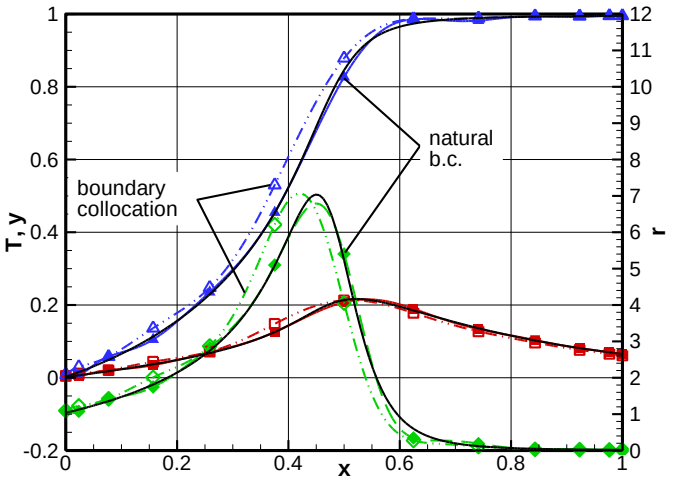


Fig. 3.45 T, y and rate profiles, Lobatto, n = 11

The differences in the profiles are also visible in the L_1 error norms for the conversion, y, shown in Fig. 3.46. There is again little difference in the comparison with L_1 or L_2 error norms. All of the errors for temperature, T, are smaller by roughly a factor of 5, which corresponds to the maximum variation of y relative to T. For this reason, all the reported error results are for conversion, y, and the mass balance equation. Fig. 3.46 shows virtually no difference in the results with the different points, Gauss, Chebyshev or Lobatto, but there are significantly greater errors when boundary collocation, Eq. (3.72a), is used with Chebyshev or Lobatto points instead of a natural treatment, Eq. (3.74). These results were calculated for every n for n = 9 to 31 and only odd points for larger n. The error tends to oscillate based on where the points lie with respect to the more difficult parts of the profile such as where the rate is large.

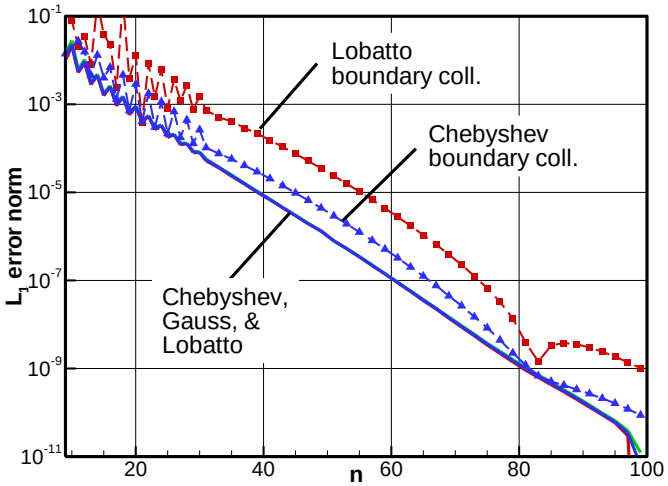


Fig. 3.46 L_1 error norms of conversion y

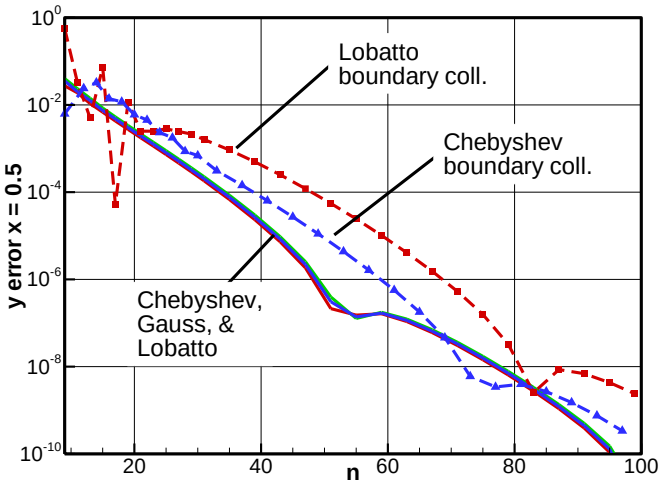


Fig. 3.47 Error of y at x = 0.5

Fig. 3.47 shows the error in conversion at the center. The error is calculated only for an odd number of points, since an even number requires interpolation to determine the value at the center and consequently the error is larger. In agreement with Fig. 3.46, Fig. 3.47 shows significantly larger error with boundary collocation and erratic behavior for n < 20. Although the overall average convergence rate is similar with boundary collocation, the actual error at some n is greater by almost one order of magnitude for Chebyshev points and two orders of

magnitude with Lobatto points. The error is larger with Lobatto points, because the difference between Eqs. (3.72a) and (3.74) is proportional to the boundary quadrature weights, $W_0 = W_{n+1}$, and the weights are approximately twice as large for Lobatto points, so it is more important to treat the boundary condition correctly with Lobatto points.

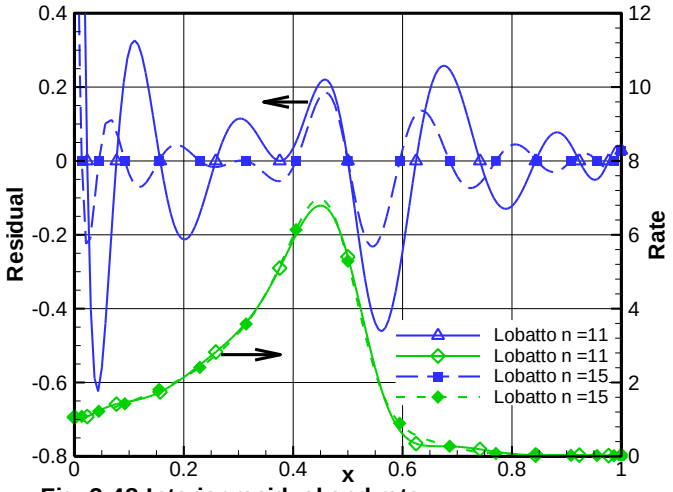


Fig. 3.48 Interior residual and rate

Fig. 3.48 shows the residual of Eq. (3.71) for two cases. The residual errors tend to be largest near the inlet and where the rate declines sharply as complete conversion is approached. All point types display this behavior, but the errors are slightly smaller near the inlet and larger near the center for Chebyshev and Gauss points due to the greater concentration of points near the ends at the expense of their spacing near the center.

Fig. 3.49 shows the residual of the inlet and outlet boundary conditions for the natural treatment of the mass balance equation, i.e. the left term in Eq. (3.74). These are identically zero for Gauss points. The errors are larger at the inlet as expected from Fig. 3.48, but they converge exponentially. Eq. (3.74) relates the residual of the boundary condition to the interior residual of Fig. 3.48 at $x = 0$ and 1. The boundary values of the interior residual are larger by a factor of $1/W_0 = 1/W_{n+1}$. Although the boundary weights decrease in proportion to $1/n^2$, the residuals will decline at the faster exponential rate as shown for the previous examples in Figs. 3.13, 3.38 and 3.39.

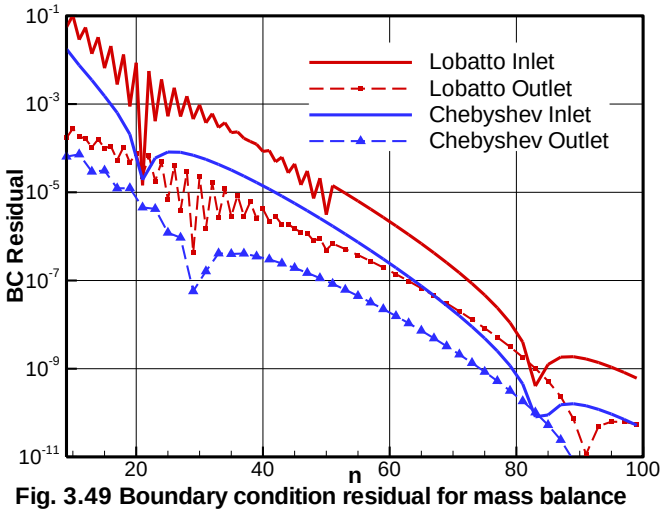


Fig. 3.49 Boundary condition residual for mass balance

The results for Lobatto points in Fig. 3.49 were calculated for every n from $n = 9$ to 51 and at only odd points for larger n. For Lobatto points the odd points give lower errors for $n < 38$ and even numbered points give lower errors at larger n. The oscillations between odd and even points are a function of the distribution of points with respect to the more difficult parts of the profile. Only odd numbered points were used for the Chebyshev calculations, so the results do not oscillate like those for the Lobatto results.

These calculations provide another example of an application to a nonlinear boundary value problem. This example is different from the previous one since it contains the first derivative

convection term which dominates the smaller dispersion term. It shows, once again, that a natural or weak treatment of flux boundary conditions is superior to boundary collocation. In this case, there are significant errors in the internal profiles, not just the fluxes. The sharp profiles in this example require large n to achieve acceptable engineering accuracy. For example, 11 points are required to achieve a one percent average L_1 error and 15 for a one percent error in y at the center. This problem is a good candidate for a finite element approach.

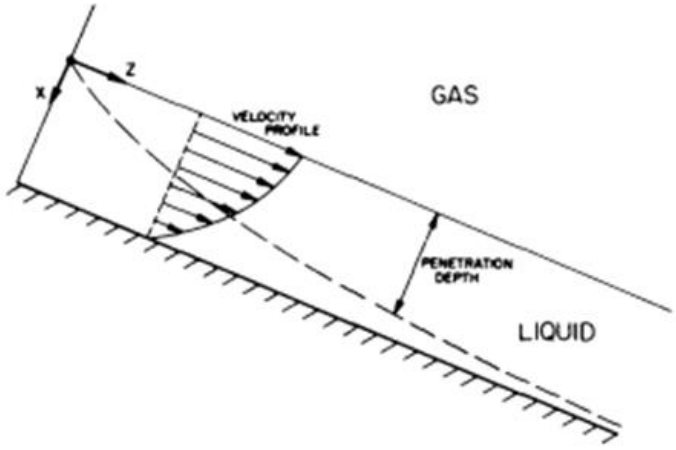
4. Parabolic Partial Differential Equations

Parabolic equations are considered in this chapter. The first problem is a convective heat or mass transfer problems with a variable coefficient due to the velocity profile. This problem is a nonsymmetric one for transfer from a falling liquid film. It has a Dirichlet condition at one boundary and a no flux (Neumann) condition at the other. Bird, et al. (1960, p. 538) describe the problem and present a penetration solution valid near the inlet. This problem was also treated by Finlayson [1972, 2014, pp 41, 58] and by Villadsen and Michelsen (1978, sec. 4.3). These references include many others which address the problem.

Finlayson used trigonometric trial functions while Villadsen and Michelsen used collocation at the base points of Radau quadrature. The reason for using Radau points will become evident. We wish to supplement the results of Villadsen and Michelsen by considering other choices of points. We consider collocation at Gauss, Radau (left and right), Lobatto and Chebyshev points. A Galerkin method is also considered. Due to the no flux boundary condition, this problem also offers the opportunity to further test different methods for treating boundary flux conditions.

When the spatial operators are discretized with a MWR a system of ordinary differential equations results. This system is first solved analytically to give a continuous solution in z. Numerical solution of the system of ordinary differential equations is then considered by several stepping techniques.

4.1 Mass or Heat Transfer from Falling Liquid Film



Consider laminar flow of a liquid film down a solid wall as shown in the illustration. Let the gas-liquid interface be at x = 0 and the solid wall at x = 1. The film is in laminar flow, so the velocity profile is parabolic, 0 at x = 1 and a maximum at x = 0. Like most convective heat and mass transfer problems, the no slip condition at the wall creates a singularity there. Assume the entering liquid is at one composition (temperature) and the gas-liquid interface is at a different composition. Since the wall is solid, assume no flux there. The dimensionless governing equations for the problem are:

$$(1-x^2)\frac{\partial y}{\partial z}=\frac{\partial^2 y}{\partial x^2} \tag{4.1}$$

with:

$$y(0,z)=0,\ \ \frac{\partial y}{\partial x}(1,z)=0 \ \text{ and } \ y(x,0)=1 \tag{4.1a}$$

This model can be compared to a simpler lumped parameter model where the transfer is governed by a mass transfer coefficient:

$$\frac{2}{3}\frac{d\bar{y}}{dz} = -Sh\,\bar{y} \tag{4.2}$$

where Sh is the Sherwood number. For heat transfer, the analogous parameter would be the Nusselt number, a dimensionless heat transfer coefficient. $\bar{y}$ is the mixing cup average composition given by:

$$\bar{y}(z) = \frac{\int_0^1 (1-x^2)\,y\,dx}{\int_0^1 (1-x^2)\,dx} = \frac{3}{2}\int_0^1 (1-x^2)\,y(x,z)\,dx \tag{4.3}$$

Using the average energy equation or divergence theorem, the Sherwood number can be calculated from the solution by two different methods:

$$Sh = \frac{-2}{3\bar{y}}\frac{d\bar{y}}{dz} = \frac{1}{\bar{y}}\frac{\partial y}{\partial x}(0,z) \tag{4.4}$$

The Sherwood number is another normalized flux quantity, similar to the effectiveness factor in the reaction and diffusion problem.

4.1.1 Orthogonal Collocation

The conventional method of solution by orthogonal collocation, pseudospectral, or differential quadrature methods is to use the matrix derivative operators to convert the problem to a coupled set of ordinary differential equations:

$$\left(1-x_j^2\right)\frac{dy_j}{dz} - \sum_{i=0}^{n+1} B_{ji}y_i = 0 \tag{4.5}$$

for $j = 1,\ldots,n,$ with

$$y_j(0) = 1,\ \ y_0 = 0,\ \text{and}\ \sum_{i=0}^{n+1} A_{n+1,i}y_i = 0 \tag{4.5a}$$

As before, $\boldsymbol{A}$ and $\boldsymbol{B}$ give approximations to the first and second derivatives, section 2.5. The matrix, $\boldsymbol{B}$, is not symmetric even though the operator is self adjoint.

Eq. (4.5a) approximates the no flux condition at x = 1 using boundary collocation as suggested in virtually all texts on the orthogonal collocation or pseudospectral method. Based on the results in Chapter 3 for boundary value problems, we can expect the boundary collocation approach to be inferior to a natural treatment, except for Gauss points or a method with a zero quadrature weight, W_{n+1} = 0 (see Appendix B). Villadsen and Michelsen solved the problem using the base points of the Radau quadrature which has a quadrature weight at x = 0, but no quadrature weight at x = 1. We call these points Radau-left, while those with a weight at x = 1

we call Radau-right. The base points and weights for Gauss, Radau-left, Lobatto and Clenshaw-Curtis quadrature (Chebyshev points) are compared in Fig. 4.1. All but the Radau points are symmetric about x = 0.5. The Radau-left points are skewed away from x = 0 and are more densely spaced near x = 1. Using collocation at the Radau-left points is somewhat counterintuitive, since for this problem large gradients occur near x = 0 for small z and the point spacing is less dense in that area. The Radau-right points are the mirror image of the Radau-left points and have a greater density of points near x = 0.

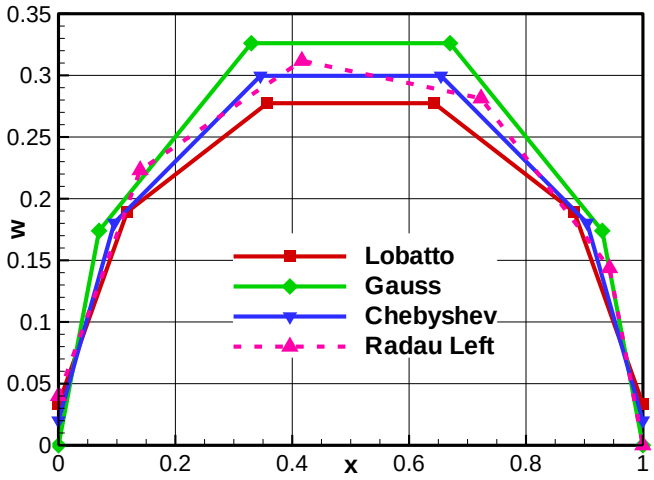


Fig. 4.1 Points and quadrature weights

As discussed in Chapter 3, an alternative weak formulation can be constructed using the stiffness matrix. For Eq. (4.1) the weak formulation is:

$$\left(1-x_j^2\right)W_j\frac{dy_j}{dz}+\sum_{i=0}^{n+1}C_{ji}y_i=0 \tag{4.6}$$

for $j=1,\ldots,n+1$ where $\boldsymbol{C}$ is the stiffness matrix given by Eq. (2.111) or (3.25). By examining the definition of $\boldsymbol{C}$, it is clear that at the interior points Eq. (4.6) is equal to Eq. (4.5) multiplied by the quadrature weights, W_j, so the equations are equivalent. At the boundary point, $n+1$, Eq. (4.6) is equivalent to:

$$\sum_{i=0}^{n+1}A_{n+1,i}y_i-W_{n+1}\sum_{i=0}^{n+1}B_{n+1,i}y_i=0 \tag{4.7}$$

It differs from Eq. (4.5a) by the additional second term. The two terms are a weighted combination of two residuals: the boundary condition residual and the interior residual evaluated at the boundary weighted by W_{n+1}. The equation is like Eq. (1.15). This is the same relationship as before, see Eqs. (3.26), (3.45) and (3.74). With this natural treatment of the boundary condition, we can expect the residual of the boundary condition to converge along with the residual of the differential equation as in Figs. 3.13, 3.38, 3.39.

The stiffness matrix, $\boldsymbol{C}$, is symmetric for Gauss, Radau (left and right) and Lobatto points and for Chebyshev points when n < 4. So, the no flux boundary condition is embedded in Eq. (4.6). The boundary weight is zero for Gauss and Radau-left points, so the two formulations, Eqs. (4.5a) and (4.7), are equivalent.

After eliminating the boundary values, y_0 and y_{n+1}, from the equations, Eq. (4.5) reduces to:

$$\left(1-x_j^2\right)\frac{dy_j}{dz}-\sum_{i=1}^{n}\hat{B}_{ji}y_i=0 \tag{4.8}$$

where:

$$\hat{B}_{ji}=B_{ji}-B_{j,n+1}\frac{A_{n+1,i}}{A_{n+1,n+1}}$$

Similarly, elimination of boundary values from Eq. (4.6) gives:

$$\left(1-x_j^2\right)W_j\frac{dy_j}{dz}+\sum_{i=1}^{n}\hat{C}_{ji}y_i=0 \tag{4.9}$$

where:

$$\hat{C}_{ji}=C_{ji}-C_{j,n+1}\frac{C_{n+1,i}}{C_{n+1,n+1}}$$

Eqs. (4.8) and (4.9) are identical in form.

4.1.2 Galerkin Method

Section 3.1.3 describes the Galerkin method for the boundary value problem. For this problem, the weak form, Eq. (4.6), with Lobatto or Radau points approximates the Galerkin method, which is given by:

$$\sum_{i=0}^{n+1}\left(D_{ji}\frac{dy_i}{dz}+C_{ji}y_i\right)=0 \tag{4.10}$$

where, using quadrature, the mass matrix $\boldsymbol{D}$ is:

$$D_{ji}=\sum_{k=0}^{m+1}W_k(1-x_k^2)\ell_j(x_k)\ell_i(x_k) \tag{4.10a}$$

Lobatto and Radau points give exact integration of the Galerkin stiffness matrix, Eq. (2.111) or (3.25). The mass matrix reduces to the diagonal form in Eq. (4.6) when $m = n$. Lobatto and Radau quadrature are exact for polynomials of degree $2n + 1$ and $2n$ respectively, while the integrand of Eq. (4.10a) is a polynomial of degree $2n + 4$. The other point choices give less accurate integration. As explained in Section 3.1.2, Gauss points give the stiffness matrix for the method of moments. Chebyshev points can also use a weak formulation with a natural boundary condition treatment. For additional discussion, see Sections 3.1.2 and 3.1.3.

If the problem is solved by the full Galerkin method, Eq. (4.10), the coefficients of dy_{n+1}/dz are nonzero, so y_{n+1} cannot be eliminated from the approximation as with the collocation methods. Of course, y_0 can be eliminated, so the Galerkin method requires the simultaneous solution of $n + 1$ rather than n equations.

Regardless of method, the equations are coupled initial values problems. To obtain accurate results, the initial conditions must be determined correctly. Most examples using the Galerkin method evolve in time, whereas this one evolves with z. Time dependent problems frequently have a constant coefficient on the time derivative, so the correct treatment here is not obvious. When the initial conditions are a general function of x, the collocation methods use the procedure given in Eq. (4.5a):

$$y_0 = 0,\ y_{n+1} = \sum_{i=1}^{n} \frac{-C_{n+1,i}}{C_{n+1,n+1}} y_i \text{ and } y_i = y(x_i, 0) \text{ for } i = 1, \ldots, n$$

These are referred to as *collocation* initial conditions. For the Galerkin method, with a constant coefficient, the initial values are selected so the weighted distribution of the initial condition follows:

$$\int_0^1 \ell_j(x) y(x,0) dx = \sum_{i=1}^{n+1} y_i \int_0^1 \ell_j(x) \ell_i(x)\, dx$$

with, y_0 = 0, since it is an essential boundary condition. For reference, this is called the *standard* treatment. However, think in terms of energy, suppose y represents temperature which is a general function of x. The requirement is to ensure the distribution of the initial energy is consistent. The energy flowing at a given x depends on the velocity, so velocity must be taken into consideration. The correct initial condition for the Galerkin method is:

$$\int_0^1 \ell_j(x)(1-x^2) y(x,0) dx = \sum_{i=1}^{n+1} y_i \int_0^1 \ell_j(x) \ell_i(x)(1-x^2) dx = \sum_{i=1}^{n+1} D_{ji} y_i$$

Where $\sum_{i=1}^{n+1} D_{ji} y_i = W_j\left(1 - x_j^2\right) y_j$ except for the case n = 1. This formulation insures not only that the initial mixing cup value is correct, but the weighted distribution is also correct. These are termed *velocity weighted* initial conditions. Both the standard and velocity weighted methods reduce to collocation when the integrals are approximated with the quadrature associated with the collocation method.

4.1.3 Continuous Solutions in z

Eq. (4.8) or (4.9) can be solved analytically by assuming a solution of the form $y = e^{-\lambda z}$ and then computing the eigenvalues and eigenvectors of the generalized eigenproblem:

$$\widehat{\boldsymbol{C}}\boldsymbol{y} = \lambda \widehat{\boldsymbol{W}}\boldsymbol{y} \tag{4.11}$$

In Eq. (4.11), $\widehat{\boldsymbol{C}}$ is replaced by $-\widehat{\boldsymbol{B}}$ for the first formulation, and $\widehat{\boldsymbol{W}}$ is the diagonal matrix multiplying $d\boldsymbol{y}/dz$ in either collocation formulation. For the Galerkin method $\boldsymbol{C}$ is used in place of $\widehat{\boldsymbol{C}}$, and $\boldsymbol{D}$ is substituted for $\widehat{\boldsymbol{W}}$, also i and j = 1,…n + 1. Consequently, there is an additional eigenvalue with the Galerkin method.

If the eigenvalues, λ_k, are all real, then the continuous time solution is [Franklin (1968)]:

$$y_i = \sum_{k=1}^{n}\sum_{j=1}^{n} v_{ik} v_{kj}^{-1} y_j(0)\, e^{-\lambda_k z} = \sum_{k=1}^{n} a_{ik} e^{-\lambda_k z} \tag{4.12}$$

where v_{ik}, are the eigenvectors. For collocation, $a_{n+1,k}$ for the boundary value y_{n+1} are calculated by substitution of Eq. (4.12) into Eq. (4.5a) or (4.6). For a Galerkin method the upper limit of the summation is $n + 1$, y_{n+1} and $a_{n+1,k}$ are directly from Eq. (4.12).

Using Eq. (4.3) the mixing cup composition is:

$$\bar{y} = \sum_{k=1}^{n}\sum_{i=1}^{n} \frac{3}{2} W_i (1 - x_i^2)\, a_{ik}\, e^{-\lambda_k z} = \sum_{k=1}^{n} c_k e^{-\lambda_k z} \tag{4.13}$$

Determination of the Sherwood number requires calculation of the boundary flux. As shown in Eq. (4.4), the boundary flux can be calculated from either $\partial y/\partial x|_{x=0}$ or from $d\bar{y}/dz$. If calculated correctly as discussed in section 3.1.4, it makes no difference which method is used. The coefficients of the boundary flux are conveniently given by the weak form of the approximation:

$$\left.\frac{\partial y}{\partial x}\right|_{x=0} = -\sum_{i=1}^{n+1} C_{0,i} y_i = -\sum_{k=1}^{n}\sum_{i=1}^{n} C_{0,i} a_{ik}\, e^{-\lambda_k z} = \sum_{k=1}^{n} d_k e^{-\lambda_k z} \tag{4.14}$$

It can be shown that $d_k = -\frac{2}{3}\lambda_k c_k$, so Eq. (4.14) gives the same boundary flux as the differentiation of Eq. (4.13). The Sherwood number is then:

$$Sh = \frac{-2\sum_{k=1}^{n} \lambda_k c_k e^{-\lambda_k z}}{3\sum_{k=1}^{n} c_k e^{-\lambda_k z}} = \frac{\sum_{k=1}^{n} d_k e^{-\lambda_k z}}{\sum_{k=1}^{n} c_k e^{-\lambda_k z}} \tag{4.15}$$

The asymptotic Sherwood number applies for large z:

$$Sh_\infty = \lim_{z\to\infty} Sh = \frac{-2\lambda_1}{3} \tag{4.16}$$

The analytical solution to the problem is like Eq. (4.13) except that it is an infinite series. The collocation solution approximates a truncation of the infinite series. In the analytical solution, all of the eigenvalues are not only real and negative but also the coefficients, $\boldsymbol{c}$, are all positive. The solution is a series of different modes each decaying at a different rate.

The statements above apply to the analytical solution, but what about the numerical solutions? Whether an approximation produces real eigenvalues, is one test for the suitability of a method. If $\widehat{\boldsymbol{C}}$ and $\widehat{\boldsymbol{W}}$ in Eq. (4.11) are symmetric, it can be proven that the eigenvalues are all real. The weak formulation, Eqs. (4.6) and (4.9), produces symmetric matrices for all but Chebyshev points with $n > 3$. As explained above, the conventional formulation, Eqs. (4.5) and (4.8), and weak formulation are equivalent when the boundary quadrature weight is zero, which is true for Gauss and Radau left points. Boundary collocation makes the matrix

asymmetric for Lobatto and Radau right points and creates additional asymmetry for Chebyshev points. The consequences of this asymmetry will be discussed shortly.

An eigenvalue problem like this can be solved more efficiently when matrices are symmetric. When the matrices are not symmetric there is no advantage to treating the eigenproblem in generalized form, Eq. (4.11), so one should divide through by diagonal matrix, $\widehat{\boldsymbol{W}}$, to give the identity matrix on the right hand side.

To test for possible complex eigenvalues and negative coefficients, $\boldsymbol{c}$, all the nonsymmetric eigenproblems were solved for $n \le 60$. For the weak formulation, Chebyshev points produce only real eigenvalues and positive coefficients, $\boldsymbol{c}$. For the conventional formulation, Eqs. (4.5) and (4.8), Radau-right points give complex eigenvalues for $n > 4$, Lobatto points give complex values for n = 4-6 and $n > 11$, Chebyshev points give complex values for n = 28, 45, 50 and 58. The coefficients, $\boldsymbol{c}$, are negative for many cases when the eigenvalues are real. The propensity to produce complex eigenvalues and negative coefficients is directly related to the magnitude of the boundary quadrature weights, W_{n+1}, and hence the degree of difference between Eqs. (4.5a) and (4.7). For large n, the Chebyshev (Clenshaw-Curtis) boundary weights are half the Lobatto and Radau boundary weights.

On the surface, these results appear to be different from previous work which has analyzed approximations to the constant coefficient heat equation. Gottlieb and Lustman (1983) found that Chebyshev points produce only real eigenvalues for the heat equation with general boundary conditions that include those in Eq. (4.1a). The analysis of the heat equation by Canuto, et al. (1988, p. 407) proved real eigenvalues for Lobatto and Chebyshev points with Neumann and Dirichlet boundary conditions. However, they consider a weak formulation with a natural boundary condition treatment, similar to that used here. Funaro (1993, pp. 143, 204) also considered eigenvalues with a natural boundary condition treatment. Unfortunately, these previous works are cited to support the suitability of the conventional formulation using boundary collocation. It is obviously incorrect to generalize the results to a different boundary condition treatment, and, in this case, to a problem with variable coefficients.

To reconcile these differences, we did some limited testing of the constant coefficient heat equation with boundary collocation. Since Canuto, et al. (1988) and Funaro (1992) used a natural boundary condition treatment, their results do not apply to the case of boundary collocation. We found that with boundary collocation, using Chebyshev points produces only real negative eigenvalues, but the coefficients, $\boldsymbol{c}$, are frequently negative. For Lobatto points, the eigenvalues are complex for $n > 6$, so with Lobatto points the conventional formulation with boundary collocation is unsuitable even for the constant coefficient problem. Although not obviously wrong, the use of boundary collocation with Chebyshev points is likely to be less accurate as found for all the problems tested here.

The accuracy of our results were double checked with two completely independent calculations: (1) in Fortran using the LAPack code DGEEV [Anderson, *et al.* (1999)] and (2) in Octave using the *eig* function. The results agreed to within the limits of machine precision.

Table 4.1
Conventional Formulation, Eq. (4.5)

Points	Equivalent to Eq. (4.6)	Eigenvalues
Gauss	yes	real
Chebyshev	no	complex
Lobatto	no	complex
Radau Left	yes	real
Radau Right	no	complex

Table 4.2
Weak Formulation, Eq. (4.6)

Points	Symmetric	Eigenvalues
Gauss	yes	real
Chebyshev	no	real
Lobatto	yes	real
Radau Left	yes	real
Radau Right	yes	real

Tables 4.1 and 4.2 summarize this discussion concerning the suitability of the various formulations. Based on these results, Lobatto and Radau-right points are unsuitable using the conventional formulation but are viable using the weak formulation. Some limited calculations using the conventional formulation with Chebyshev points (excluding n = 28, 45, 50 and 58) are included due to its popularity.

Villadsen and Michelsen (1978) gave no clear explanation for their selection of Radau-left points. Since they used the conventional formulation, i.e. boundary collocation, their choice gives the most accurate quadrature with a zero boundary weight at the no flux boundary. Its equivalence to the weak formulation insures the eigenvalues are real. They did not present results with other points nor did they consider a natural or weak formulation of flux boundary conditions.

Figs. 4.2 and 4.3 show the first few eigenvalues calculated by several methods for n = 4 and 8, respectively. Tables 4.3 and 4.4 list the calculated eigenvalues and coefficients, $\boldsymbol{c}$, for n = 6. The notation “bc” (boundary collocation) designates the conventional formulation of the Chebyshev method, i.e. Eq. (4.5) and (4.5a). Due to the equivalence of the Galerkin method

Table 4.3 Eigenvalues for n = 6

	Exact	Lobatto	Gauss	Chebyshev	Cheb. bc	Radau L	Radau R	Galerkin
λ_1	5.1216693	5.1216693	5.1216703	5.1216811	5.1213806	5.1216691	5.1216695	5.1216693
λ_2	39.660839	39.661044	39.650772	39.658315	39.496120	39.674012	39.646485	39.662482
λ_3	106.24923	105.95054	107.19163	105.87924	105.32354	103.91631	108.32712	106.59530
λ_4	204.85606	201.21666	198.03906	197.75929	215.87933	243.94612	180.92125	215.45265
λ_5	335.47320	367.15910	781.54249	502.19930	506.14967	299.49872	663.99515	457.39698
λ_6	498.09708	950.82201	2031.1571	1304.0961	959.73919	2999.4014	1011.6461	1054.5628

Table 4.4 Coefficients for n = 6

	Exact	Lobatto	Gauss	Chebyshev	Cheb. bc	Radau L	Radau R	Galerkin
c_1	0.7897026	0.7897026	0.7897032	0.7897058	0.7896109	0.7897025	0.7897027	0.7897026
c_2	0.0972551	0.0972323	0.0969374	0.0971973	0.0973883	0.0973712	0.0971233	0.0972655
c_3	0.0360936	0.0370370	0.0424662	0.0386532	0.0391686	0.0337807	0.0381707	0.0369531
c_4	0.0186864	0.0084942	0.0060119	0.0066653	0.0037217	0.0170300	0.0171504	0.0183208
c_5	0.0114018	0.0407402	0.0647798	0.0524277	0.0552174	0.0315027	0.0004645	0.0300564
c_6	0.0076760	0.0000080	0.0001015	0.0000446	-0.0004130	0.0000007	0.0573884	0.0047906
$\sum_k c_k$	0.9608154	0.9732143	1.0000000	0.9846939	0.9846939	0.9693878	1.0000000	0.9732143

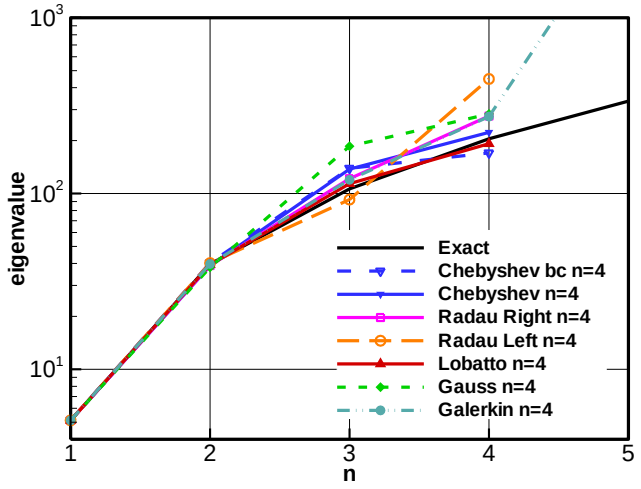


Fig. 4.2 Eigenvalues calculated with n = 4

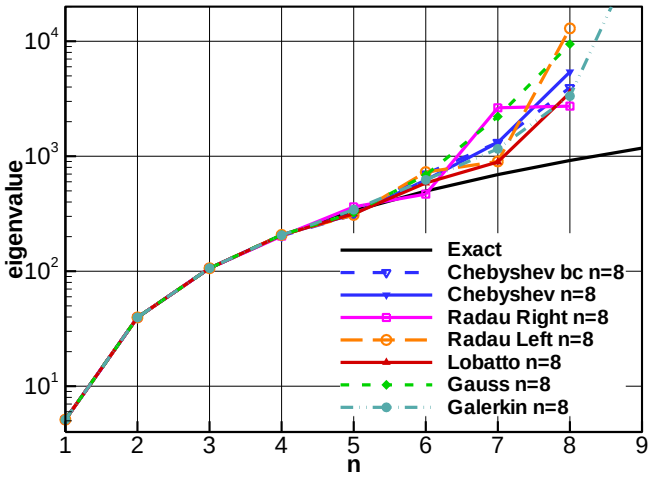


Fig. 4.3 Eigenvalues calculated with n = 8

and Rayleigh-Ritz method for this problem, it can be shown that the eigenvalues are greater than the exact ones [Finlayson (1972, 2014),].

Figs. 4.4 and 4.5 show the convergence of the first and sixth eigenvalues, respectively. When formulated correctly, the convergence of the coefficients mirrors that of the eigenvalues. Fig. 4.6 shows how Galerkin initial conditions influence the convergence of coefficient c_1. For the results labeled “coll. ic” the initial conditions are like those for the collocation method. The result labeled “standard” use the integral formula without velocity weighting, while those labeled “velocity” include the velocity weighting. Clearly, the method of treatment makes a difference and the velocity weighted method is the correct one.

The relative convergence rates using the various points are similar to those for the linear boundary value problems in Chapter 3. However, in this case we have added the Radau points, which tend to have accuracy between that of Lobatto and Gauss points. Chebyshev points with boundary collocation give the worst convergence, which is disconcerting since it is probably the most popular formulation in the pseudospectral literature. Although no formal

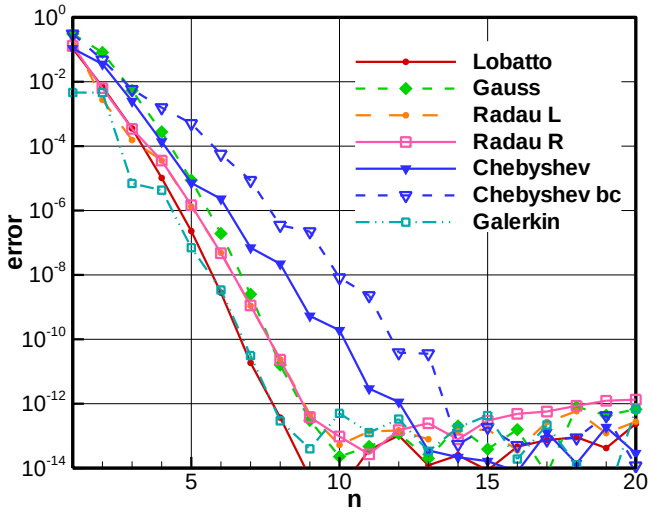


Fig. 4.4 Error in first eigenvalue

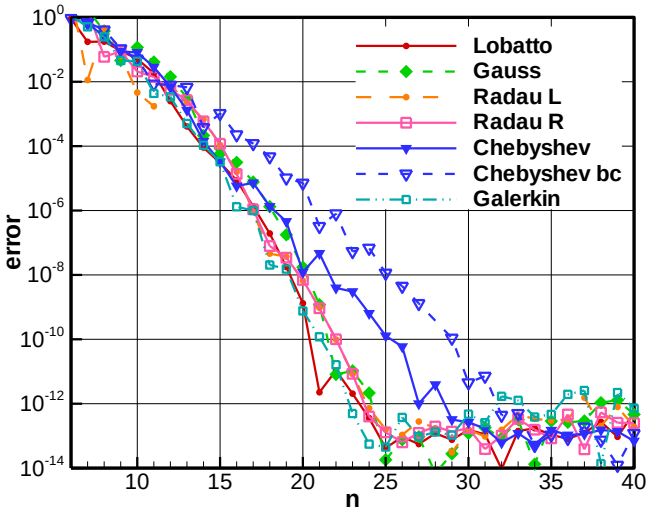


Fig. 4.5 Error in sixth eigenvalue

analysis was performed, these calculations reached the limits of roundoff with about 13 digits of accuracy and then began to creep upward slowly for very large n.

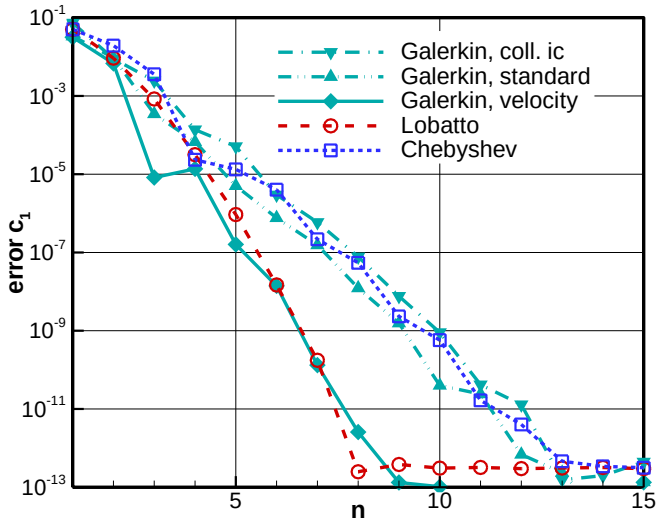


Fig. 4.6 Effective of Galerkin initial conditions on c_1

As others have observed, the first n/2 eigenvalues are reasonably accurate in all cases. For example, Fig. 4.5 shows that most methods give approximately one percent error in the 6th eigenvalue when n = 12. The largest calculated eigenvalues are significantly greater than the exact ones. The differences between the exact and maximum calculated eigenvalues grows rapidly with n. If one thinks about how best to approximate the infinite series, then at least qualitatively, this behavior makes sense. The smaller eigenvalues are the most important, since their effects damp out more slowly. Engineers often perform calculations with the simplified Eq. (4.2) and the asymptotic Sherwood number which is dependent only on the first eigenvalue. It is logical to use exact values for the smaller eigenvalues and perhaps lump several of the larger ones together. We might think of the 3rd eigenvalue in Fig. 4.2 as representing the 3rd and 4th modes and the 4th representing the 5th and larger. Also, note that the last row of Table 4.4 gives $\bar{y}(0) = \sum_k c_k$. The exact value is unity, while the sum of the first six analytical values is 0.9608. In the limit, as an eigenvalue goes to infinity, the value of its term will damp immediately. Truncating the infinite series is equivalent to treating the neglected higher eigenvalues as if they are (negative) infinite. The difference between unity and the sum of the approximate coefficients gives the sum of the coefficients for the terms treated as if they had infinite eigenvalues.

Although this argument may seem plausible, we note in Table 4.4, the large eigenvalues are frequently accompanied by small coefficients, so they contribute little to the solution. These large eigenvalues also have a downside when the problem is integrated numerically in z by a stepping method. Large eigenvalues create issues with numerical stability for explicit stepping methods. Numerical solutions in z will be discussed shortly.

Asymptotic relationships are available for the eigenvalues and coefficients of the analytical solution. From those relationships and the approximate solutions the following asymptotic relationships were determined:

$$\lambda_k \to (4k - 1.677)^2$$
$$c_k \to 3.820/\lambda_k \qquad (4.17)$$

These relationships give values within 0.1% for k = 5 to 53 of the values calculated with 100 Lobatto points.

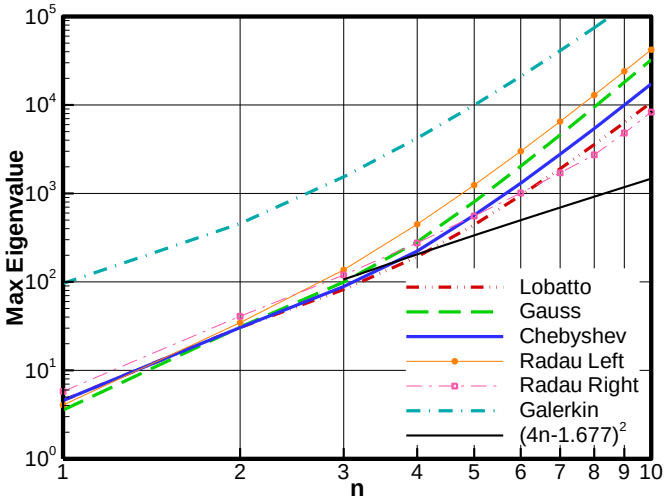


Fig. 4.7 Maximum eigenvalues vs. number of points

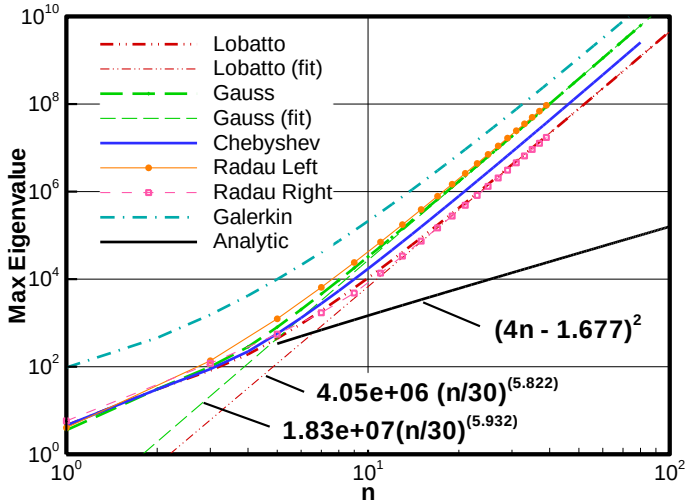


Fig. 4.8 Maximum eigenvalues vs. number of points

Figs. 4.7 and 4.8 show the calculated maximum eigenvalues for the various methods. In all cases, the values of the maximum eigenvalues become asymptotic at large n and can be approximated by:

$$\lambda_{max} \to a\left(\frac{n}{30}\right)^{b} \tag{4.18}$$

Values for Eq. (4.18) were determined from the computed values for n > 30 and are listed in Table 4.5. It appears that if still larger values were fit, the asymptote will have an exponent of 6. Values to Eq. (4.18) were not calculated for the Radau points. For large n the values for Radau left points follow those for Gauss points and results for the Radau right points follow those for the Lobatto points. An exponent of 6 for b is larger than that for a simple constant coefficient transient heat conduction problem. For that problem the exponent is 4 for both Dirichlet and Neumann boundary conditions [Canuto, et al. (1988), p. 98]. For this problem, Eq. (4.1), with mixed Dirichlet/Neumann boundary conditions, the parabolic velocity profile causes the exponent to approach a value of 6. The additional two degrees is to be expected from the matrix norm of $\widehat{\boldsymbol{W}}$. Here again, we find a significant difference between this problem and a simple constant coefficient problem. Obviously, the power of 4 does not generalize to this problem.

Table 4.5 Parameters of Eq. (4.18)

	Lobatto	Gauss	Chebyshev	Galerkin
a	$4.05\text{x}10^{6}$	$1.83\text{x}10^{7}$	$7.98\text{x}10^{6}$	$6.25\text{x}10^{7}$
b	5.822	5.932	5.869	5.690

The asymptotic lines in Fig. 4.8 are approximately parallel, so by comparing the constants, a, the maximum eigenvalues are found to be smallest for Lobatto or Radau-right points. The maximum eigenvalues 4.5 times greater for Gauss and Radau-left points and 2.0 times greater for Chebyshev points. Referring back to Fig. 4.1, it is apparent that the magnitude of the eigenvalues correlates with the grid density near x = 1, where the velocity goes to zero. A fine grid in this area causes larger eigenvalues to be computed. For n = 30 the maximum eigenvalues are roughly 2 to 3 orders of magnitude larger than those given by Canuto, et. al. (1988) for the transient heat equation (depending on point selection and boundary condition).

For finite difference and finite element methods, the eigenvalues grow with n^2 [Strang and Fix (1973)], so those methods would also have much smaller maximum eigenvalues for large n. The implication of these differences will become more apparent when we discuss numerical solutions in z for this problem.

Rather than discuss the accuracy of eigenvalues and coefficients, we should discuss the accuracy of the computed solutions to determine the number of points actually needed. The magnitude of the eigenvalues for n > 20 is of little practical interest if only a few points are required to give the desired accuracy. The real power of these methods is that they frequently

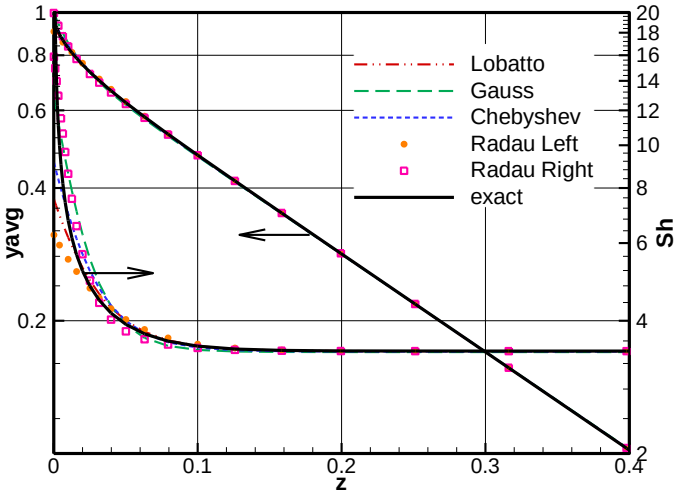


Fig. 4.9 Average composition and Sh vs z, n = 3

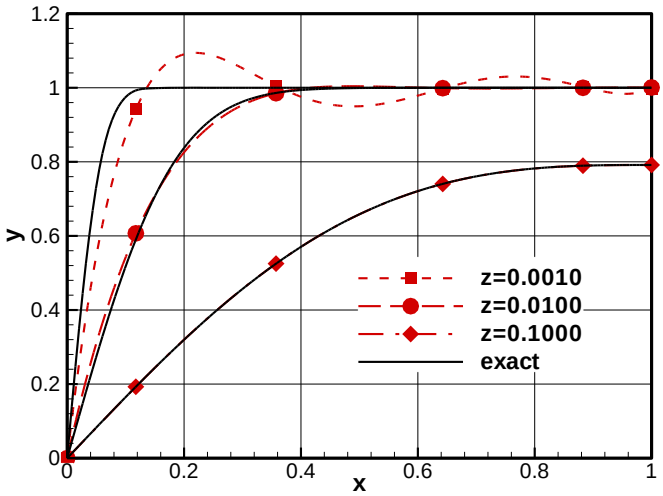


Fig. 4.10 Calculated profiles Lobatto n = 4

produce good accuracy with only a few points. For example, Fig. 4.9 shows the mixing cup average y and Sh as a function of z for the various point choices with n = 3. The higher terms damp out by z = 0.10, so the solution is asymptotic from there on. The slope and intercept of the asymptotic part is determined by the first eigenvalue and coefficient, respectively. Since one would normally be interested in transferring all or most of a component, the figure shows the portion of the curve for approximately 90 percent transfer which is reached at z = 0.4. The average y overlays the exact solution for all methods and errors in Sh are evident only at very small z. If one is interested in a significant transfer, n can be small.

There could be some problems where the results for small z are important. Fig. 4.10 shows the evolution of the profiles with z together with approximate solutions for 4 Lobatto points. Due to the sudden change in boundary composition at z = 0, the solution is not well approximated by a low order polynomial, so the approximate solution oscillates about the actual one between the collocation points, but the values at the nodes are reasonably accurate and show minimal overshoot. All of the point choices display the same behavior to some degree. Fig. 4.11 shows the profiles at z = 0.01 for the different points. Do not attempt to draw too many conclusions from any single figure, because the approximate solutions tend to oscillate about the exact solution. Fig. 4.12 shows the trend of the approximate profiles as the number of Lobatto points is increased. The other points display similar but somewhat less accurate behavior.

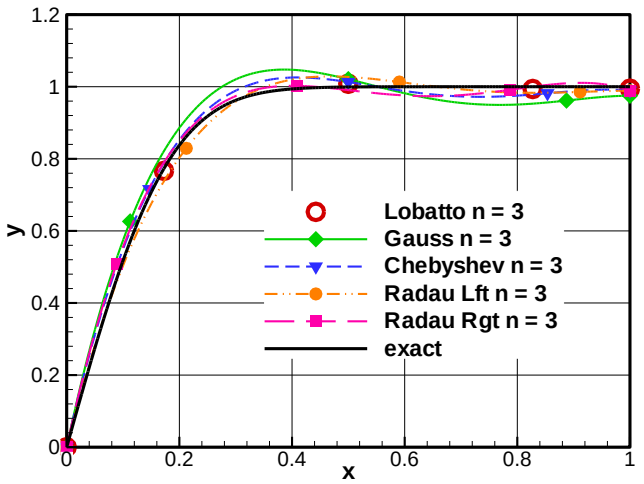


Fig. 4.11 Calculated profiles at z = 0.01, n = 3

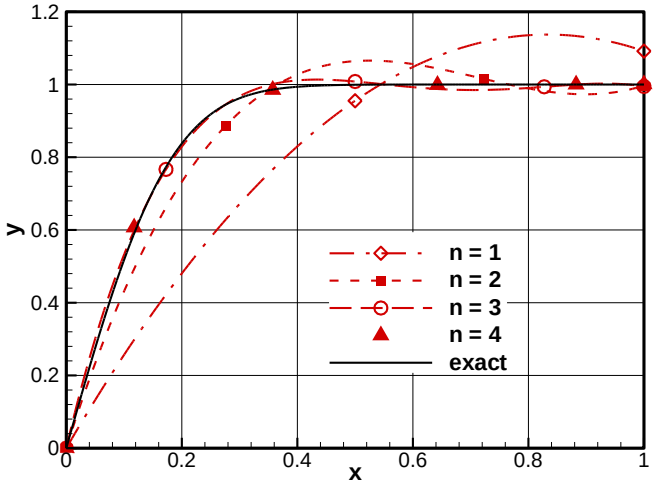


Fig. 4.12 Calculated profiles at z = 0.01, Lobatto points

Figs. 4.13 and 4.14 focus more closely on the Sherwood number or normalized flux near the inlet. A penetration solution is presented by Bird, et al. (1960). Villadsen and Michelsen (1978) extend that approximation with additional terms. Their first perturbation gives:

$$Sh \rightarrow 1/\left(\sqrt{\pi\, z} - 3z\right) \tag{4.19}$$

For small z, the second denominator term is insignificant, so the Sherwood number varies inversely with the square root of z. Including the second term extends the range, so Eq. (4.19) is within one percent of the exact value for z < 0.02. Additional terms extend the valid range to larger z. These approximations are accurate at precisely the same conditions where the global collocation or MWR solutions require large n. Fig. 4.13 shows the Sherwood number calculated for various numbers of Lobatto points. Tracing the values for n = 4, we see that as z increases the calculated value crosses the exact value at about z = 0.006, then reaches a maximum positive deviation of about 4% at z = 0.013 and is within 1% of the exact values for z > 0.030. The curves for all n, follow the same basic behavior, but accurate solutions are shifted to smaller and smaller z. If one is content with a maximum 5% error, the cut off z values for valid solutions are: z = 0.035, 0.0051, 0.0015, 0.00056, 0.00026 with n = 2, 4, 6, 8, and 10

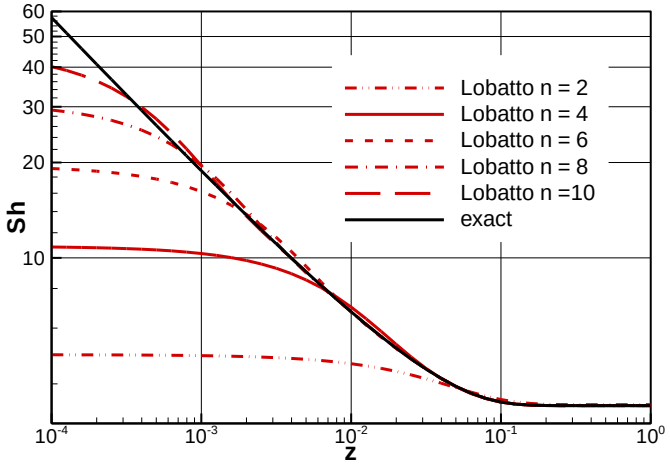


Fig 4.13 Sh vs z, Lobatto points, various n

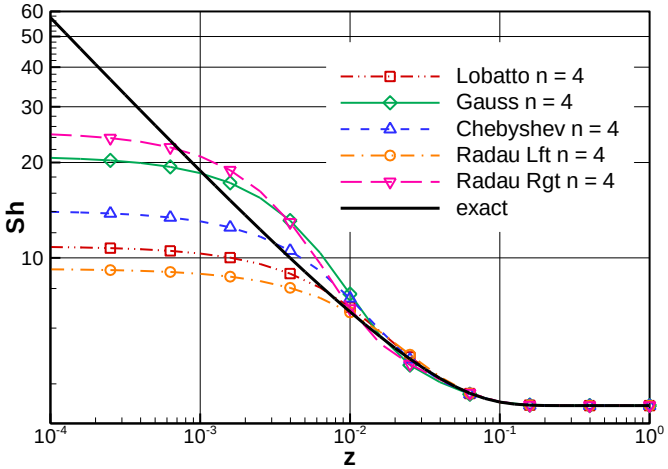


Fig 4.14 Sh vs z, n = 4, various point types

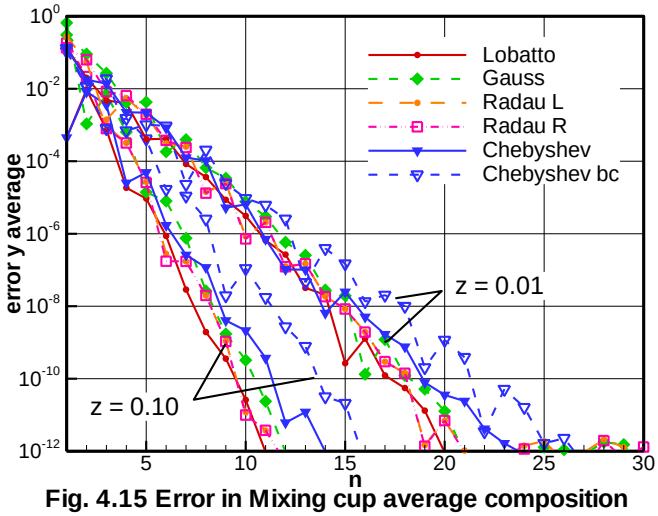


Fig. 4.15 Error in Mixing cup average composition

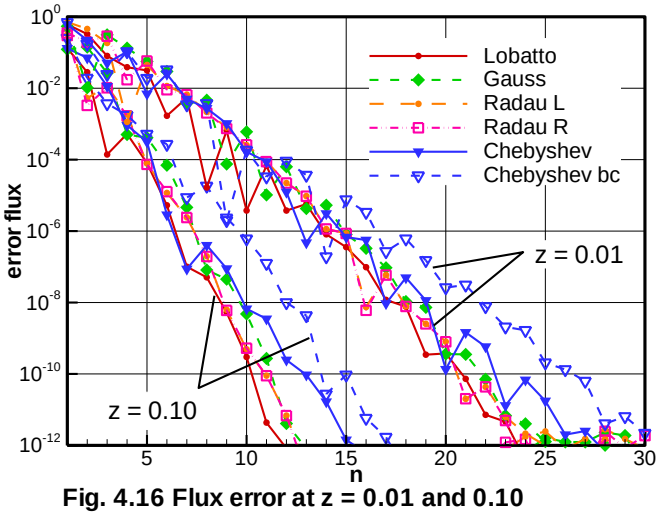


Fig. 4.16 Flux error at z = 0.01 and 0.10

Lobatto points respectively. These values are exceedingly small compared to the 90% transfer point of z = 0.40.

Fig. 4.14 shows that other point choices have the same basic behavior. This is a case where you can almost pick your poison and then pay now or pay later. The point choices that level off at higher values of Sh and cross at lower z have a greater positive deviation at larger values of z. For example for a wide range of n, the maximum positive deviations are 4 to 5 percent for Lobatto and Radau-left points, 25 to 35 percent for Gauss and Radau-right points and 10 to 15 percent for Chebyshev points.

The behavior of the errors in Figs. 4.13 and 4.14 makes it difficult to state which points are best. Figs. 4.15 and 4.16 show the convergence of the average composition, $\bar{y}$ and the boundary flux, $\partial y/\partial x|_{x=0}$, for the various formulations. Here again, the popular Chebyshev method with boundary collocation, Eqs. (4.5) and (4.5a), has been designated as “bc”, and again this choice gives the worst performance of the formulations tested. The limits of roundoff was again at about 10^{-13}, but the clutter was not included in the figures. There are no clearcut winners in these tests, the results tend to follow those in Figs. 4.4 and 4.5. It appears that there is a greater difference between the methods for the easier conditions at z = 0.10 than at z = 0.01. For this problem and those in Chapter 3, it seems to be a general trend that easier problems show a greater difference between the different points. Problems with smooth profiles, easily approximated by low order polynomials, are the ones where these collocation methods shine relative to other methods, such as finite differences or finite elements.

We believe the conventional formulation using boundary collocation, Eq. (4.5), with Chebyshev or Lobatto (any points with a nonzero quadrature weight at the boundary) should be buried and forgotten. There is a good chance that nonphysical complex eigenvalues will be calculated. When the eigenvalues are not complex, it gives the slowest rate of convergence. The weak formulation with natural boundary conditions, Eq. (4.6), is symmetric for all but Chebyshev points. Although the weak formulation with Chebyshev points gives real eigenvalues, less efficient nonsymmetric eigenvalue software must be used. When considering there is no gain in accuracy relative to the other choices, there is not much to recommend it over the other

methods for similar problems. The Lobatto points seem to have a slight edge in terms of accuracy and because their maximum eigenvalues are somewhat smaller and more accurate than those of the other choices.

4.1.4 Numerical Solutions in z

Parabolic equations like, Eq. (4.1) are more often solved numerically with time (or in this case z) stepping methods rather than analytically, because the analytical approach is not applicable for nonlinear problems. Once collocation is applied, the problem is reduced to a set of coupled ordinary differential equations (ODEs), initial value problems, Eq. (4.8) or (4.9). This is sometimes called the *method of lines*, referring to the lines traced out in z by each of the nodes in x.

The numerical solution of initial value problems is a large mature field of study. With Matlab/Octave there are robust built in functions which can do all the work for you. For other languages there are libraries available, which can be freely downloaded. However, here we will do some of our own calculations to gain better insight. We cannot possibly cover the entire field, but will demonstrate some of the terminology, types of methods available, and tradeoffs between different methods when applied to paraboloic collocation problems. The ODE packages are concerned with step size selection and other practicalities. We will use a constant step size to more clearly visualize the characteristics of different methods.

The problem can be expressed as:

$$\begin{aligned} &\widehat{\boldsymbol{W}}\frac{d\boldsymbol{y}}{dz}+\widehat{\boldsymbol{C}}\boldsymbol{y}=0 \text{ or} \\ &\frac{d\boldsymbol{y}}{dz}=-\widehat{\boldsymbol{W}}^{-1}\widehat{\boldsymbol{C}}\boldsymbol{y}\;=\;\boldsymbol{F}(\boldsymbol{y},z) \end{aligned} \tag{4.20}$$

Where $\boldsymbol{y}$ is the vector of values $y(x_i,z)$ at the interior collocation points. $\widehat{\boldsymbol{W}}$ and $\widehat{\boldsymbol{C}}$ are, respectively, diagonal and full n by n matrices defined by either Eq. (4.8) or (4.9). The collocation initial conditions are $\boldsymbol{y}$ = 1. The far right expression is a more general representation of the problem which admits nonlinear and variable coefficient problems.

Basic Methods and Stability

The values are known initially, so the solution proceeds in a stepwise fashion moving in increments of $h=z^{k+1}-z^k$. The simplest approach is the Euler method:

$$\boldsymbol{y}^{k+1}-\boldsymbol{y}^k=h\,\boldsymbol{F}(\boldsymbol{y}^k,z^k) \tag{4.21}$$

which is a one sided first order finite difference approximation to Eq. (4.20). The right hand side is calculated from the known values starting at the initial conditions. Once the right hand side has been calculated Eq. (4.21) gives the conditions at the end of the step. This process is repeated over and over until the desired end point is reached. For our problem, the right hand side is calculated with a matrix-vector multiply, requiring O($2n^2$) floating point operations (*flops*).

Fig. 4.17 shows the mixing cup average composition and flux calculated for Lobatto points with n = 6. The values for the continuous z solution are given in Tables 4.3 and 4.4. This solution gives the exact integration of Eq. (4.20), but is approximate due to spatial discretization errors. Figs. 4.15 and 4.16 show the spatial errors are 0.04% and 0.2% for average y and flux respectively at $z = 0.01$ and the solution is accurate to 6 or 7 digits at $z = 0.10$. The stepping solution converges to this solution when $h \to 0$. The continuous z solution is plotted against the results from stepping with the Euler method. From Fig. 4.17, we see that the numerical solution looks fine with $h =$ 0.00208, but falls apart when $h =$ 0.00227. With this restriction, approximately 200 steps are required to integrate the solution to z = 0.4 and the error in the flux is approximately 0.7% at $z = 0.1$. It would be preferable if a more accurate solution could be achieved with fewer steps or fewer calculations.

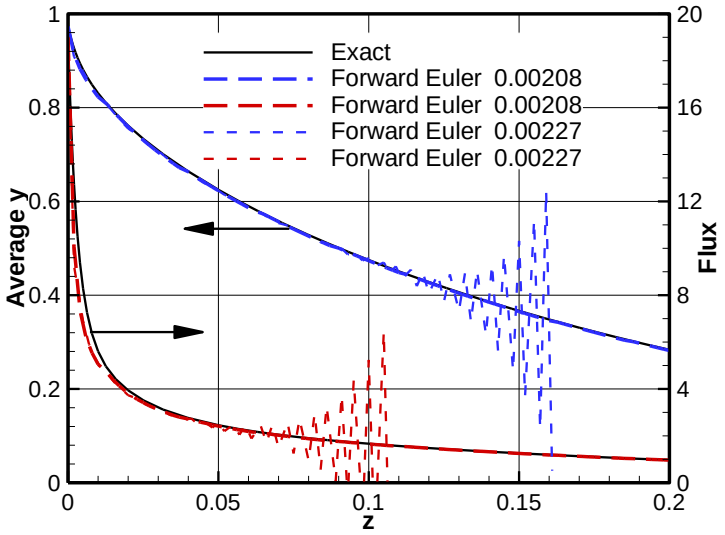


Fig. 4.17 Average y and flux, Lobatto n = 6, Euler method

To gain a better understanding of the problem, look more closely at the simpler problem with n = 1. Eq. (4.21) reduces to the scalar equation:

$$\begin{aligned} y^{k+1} - y^k &= h\lambda\, y^k \text{ or} \\ y^{k+1} &= (1 + h\lambda)\, y^k \text{ or} \\ y^{k+1} &= (1 + h\lambda)^{k+1} \end{aligned} \tag{4.22}$$

λ is the negative real eigenvalue approximating the first eigenvalue of the infinite series analytical solution. Eq. (4.22) is a first term Taylor series approximation of the analytical solution $e^{\lambda z}$. After repeated steps the last line in Eq. (4.22) results. The quantity in parenthesis is call the *amplification factor*. The approximation will fail if $|1 + h\lambda| > 1$ or $h > -2/\lambda$. This failure mode is called *numerical instability* and due to the step size limit, the Euler method is *conditionally stable*. Numerical stability is completely separate and apart from the issue of *truncation error*. The eigenvalue for n = 1 is approximately -5 depending on the points selected, so there is not much of a restriction. However, for larger n, all of the eigenvalues must meet the restriction imposed by Eq. (4.22). For n = 6, the largest eigenvalue for Lobatto points in Table 4.3 gives the step size limit of h < 2/951 = 0.0021, which agrees with the computations shown in Fig. 4.17. Note too that the coefficient of the maximum eigenvalue in Table 4.4 is 8×10^{-6} which explains why it takes about 30 steps before the instability grows large enough to be visible in Fig. 4.17. The flux is more sensitive than the average, so it is the first to show signs of instability.

If we had selected the Radau-left points with n = 6, computation of a stable solution would require four times as many steps (see Table 4.3). If we use n = 8 (see Figs. 4.3 and 4.7), the number of steps required varies from 550 to 2600, which is clearly excessive. It is often assumed that larger steps can be taken when the solution is quiescent. This is not true. If small steps are used up to say $z = 0.3$ and then the step size is increased beyond the stability limit, the solution will become unstable soon after. There is no way around this limitation when using the Euler integration method or any method with conditional stability.

For this example problem the eigenvalues are all real and negative. If the problem were one with complex eigenvalues, the condition would still apply. For example given a complex eigenvalue, $\lambda = \lambda_r + i\,\lambda_i$:

$$|1 + h(\lambda_r + i\,\lambda_i)| < 1 \text{ or}$$
$$\sqrt{(1 + h\lambda_r)^2 + (h\lambda_i)^2} < 1 \qquad (4.23)$$

This is often represented as a circle in the complex plane having radius of unity and a center at (-1,0). Stability maps of other conditionally stable integration methods can be represented in a similar manner and may be found in numerous locations [e.g. Hairer and Wanner (1996), Fornberg (1996), p. 197]

Getting back to the example problem, the issue of stability can be eliminated by using the backward Euler method:

$$\boldsymbol{y}^{k+1} - \boldsymbol{y}^{k} = h\,\boldsymbol{F}(\boldsymbol{y}^{k+1}, z^{k+1}) \text{ or}$$
$$\left(\frac{\widehat{\boldsymbol{W}}}{h} + \widehat{\boldsymbol{C}}\right)\boldsymbol{y}^{\boldsymbol{k+1}} = \frac{\widehat{\boldsymbol{W}}}{h}\boldsymbol{y}^{\boldsymbol{k}} \qquad (4.24)$$

Fig. 4.18 shows calculations with this method using larger steps than those in Fig. 4.17. The same linear stability analysis as was used for the forward Euler method leads to the following expression instead of Eq. (4.22):

$$y^{k+1} = \left(\frac{1}{1 - h\lambda}\right)^{k+1} \qquad (4.25)$$

The absolute value of the amplification factor in parenthesis must again be less than one. The stability map for this method covers all but a circle of radius one centered at (0,1). When a method is stable for all eigenvalues with a negative real part, it is called absolutely stable or *A-stable*, so the backward Euler method is A-stable. One problem with this method is, like the forward Euler method, it is only correct to O(h), so the results with the larger step size in Fig. 4.18 give a flux error of about 7% at z = 0.10.

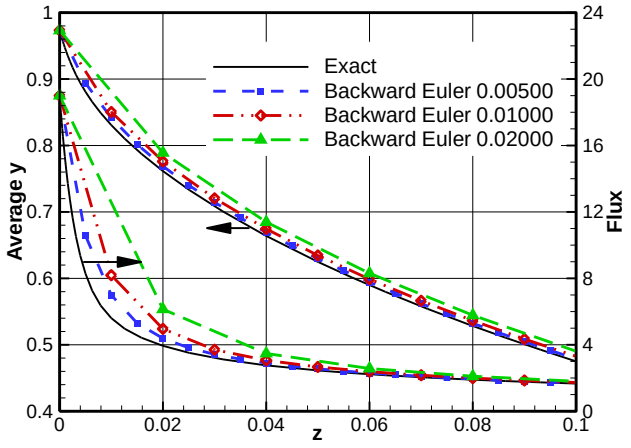


Fig. 4.18 y and flux, Lobatto n = 6, Backward Euler method

The other problem with the backward Euler method is that it requires solving a set of algebraic equations to advance the solution in z. This type of method is called *implicit*. In contrast, the forward Euler method is called *explicit* because there are no unknowns on the right-hand side of Eq. (4.21). This issue is not so important for our simple linear example problem because we can factor the matrix $\widehat{\boldsymbol{W}}/h + \widehat{\boldsymbol{C}}$ initially and when the step size changes. Then at each step, solution of the matrix problem requires only a forward elimination and back substitution of the right-hand side, which requires O($2n^2$) calculations, the same as a matrix multiply.

Chebyshev points can employ fast Fourier transforms (FFT) to perform explicit calculations. With this approach, time steps require calculation of O[(n)log(n)], whereas a matrix-vector multiply requires O(n^2), so this method is especially useful when explicit-like methods can be used for problems requiring large $n \gtrsim 40$. This method is not considered here, but is discussed in several of the references [e.g. Canuto, et al. (1988), Trefethen (2000)).

Although the example is a small linear problem, some of the tradeoffs apply to larger nonlinear ones. Factoring a matrix requires O($2n^3$/3) flops for a general matrix and O(n^3/3) flops for a symmetric one. The weak formulation, Eq. (4.9), gives a symmetric matrix problem for all but Chebyshev points. A forward and back solve is O($2n^2$) flops, the same count as for a matrix-vector multiply required for the explicit forward Euler method. The ratio for a symmetric matrix factorization and solve relative to a solve alone is O((n+8)/6)) or about 3 for n = 10. For efficiency one would want to minimize the calculations associated with factorizations by selecting a method which produces symmetric matrices. If step sizes are changed, the step size selection logic should consider the added calculations of the factorization when the step size is changed. For nonlinear problems solved by a Newton-Raphson iteration, the matrix problem is constructed using the Jacobian from the linearization. In many cases the Jacobian does not need to be updated and factored every step. Depending on the specific problem, other tradeoffs will likely be at play. For a highly nonlinear problem requiring frequent Jacobian updates and several nonlinear iterations per step, an implicit method requires substantially more work for each step. One must weigh the extra calculations per step versus smaller explicit steps.

For greater accuracy, the trapezoidal rule or Crank-Nicolson method. It has accuracy of O(h^2) so it achieve greater accuracy than the Euler methods. With this method each step requires solution of:

$$\begin{aligned}
&\boldsymbol{y}^{k+1} - \boldsymbol{y}^{k} = \frac{h}{2}\left[\boldsymbol{F}(\boldsymbol{y}^{k+1}, z^{k+1}) + \boldsymbol{F}(\boldsymbol{y}^{k}, z^{k})\right] \quad \text{or} \\
&\left(\frac{2\widehat{\boldsymbol{W}}}{h} + \widehat{\boldsymbol{C}}\right)(\boldsymbol{y}^{k+1} - \mathbf{y}^{\mathrm{k}}) = -2\,\widehat{\boldsymbol{C}}\boldsymbol{y}^{\boldsymbol{k}} \quad \text{or} \\
&\left(\frac{2\widehat{\boldsymbol{W}}}{h} + \widehat{\boldsymbol{C}}\right)(\boldsymbol{y}^{k+1} + \mathbf{y}^{\mathrm{k}}) = \frac{4\widehat{\boldsymbol{W}}}{h}\boldsymbol{y}^{\boldsymbol{k}}
\end{aligned} \tag{4.26}$$

Two arrangements are listed in Eq. (4.26). Both require a matrix solve. The first also requires a matrix-vector multiply, but may be less susceptible to roundoff errors. $\widehat{\boldsymbol{W}}$ is diagonal, so the second formulation saves some calculations by not requiring a full matrix-vector multiply. Nevertheless, for a more typical nonlinear problem, the calculations per step are similar for the Trapezoidal rule and backward Euler methods. Fig. 4.19 shows calculated results with this method using the same step sizes as used for the calculations shown in Fig. 4.18. This method gives good results for the smaller steps, but with the larger ones the error at small z is worse than with the backward Euler method.

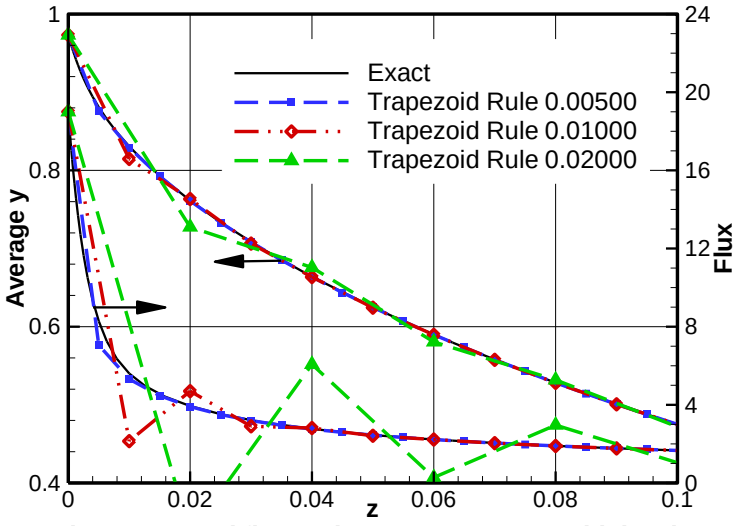


Fig. 4.19 y and flux, Lobatto n = 6, Trapezoidal Rule

To understand what is going on with Fig. 4.19, the same linear stability analysis as above gives the following result for the trapezoidal rule:

$$y^{k+1} = \left(\frac{2+h\lambda}{2-h\lambda}\right)^{k+1} \tag{4.27}$$

The amplification factor has an absolute value less than one for all eigenvalues with a negative real part (like backward Euler), so it is A-stable. However, it has a completely different character for large $h\lambda$. For the backward Euler method the amplification factor goes to zero in the limit, while for the trapezoidal rule it goes to -1. The trapezoidal rule is stable in the sense that instabilities will eventually die out, but it does so in an oscillatory fashion. This behavior is evident in Fig. 4.19. Since the true solution, $e^{\lambda z}$, goes to zero for large z, the backward Euler method gives a better approximation of this behavior.

When the amplification factor goes to zero as $h\lambda \rightarrow -\infty$ the method is called *L-stable*. For the example problem, at the end of the first step of 0.02 in Fig 4.19, the three terms with the largest eigenvalues (see Tables 4.3 and 4.4) contribute only 0.02% to $\bar{y}$ and 0.7% to the flux, so they could easily be approximated by zero with little loss of accuracy.

Stiffness

Collocation methods or other MWR for this problem with large n produce systems of equation that are called *stiff*. There is no single, universally accepted definition of what constitutes a stiff set of equations. One definition is based on the ratio of the largest to the smallest eigenvalue. For the example problem with n = 6, Table 4.3 shows this ratio varies from 185 to almost 600, excluding the Galerkin method for which it is over 4,000. For very large n, the ratio increases in proportion to n^6. Fortunately, n = 6 is good enough to give spatial errors of 0.04% and 0.2%

for $\bar{y}$ and flux, respectively, even at z = 0.01 and 6 or 7 digit accuracy at z = 0.10, so calculations with extremely large n are not normally needed.

Basically, stiff problems are ones with greatly different time (z in our case) scales. Usually, but not always, we are interested in the longest time scale. For this example, there are different time scales within the same problem. For coupled systems of differential equations, the variation can be more extreme. One example is an automotive catalytic converter and many other chemical reactor models. The models are coupled heat and mass transfer problems and both the fluid and solid must be modeled. The transient warm up response is very important since federal test cycles use a cold start. The response is governed almost exclusively by the thermal response of the solid material. Rapid warmup was improved by the development of honeycomb-like monolith substrates. One can set the time derivatives to zero in all equations except the energy balance for the solid (pseudo-steady assumption). A small coefficient of a time derivative term is equivalent to a large $h\lambda$, so an L-stable method, like the backward Euler, yields a pseudo-steady approximation automatically. For example, when $h \to \infty$ Eq. (4.24) reduces to the pseudo-steady approximation, i.e.:

$$\boldsymbol{F}(\boldsymbol{y}^{k+1}, z^{k+1}) = 0 \tag{4.28}$$

Another example of a stiff problem comes from flow in porous media, such as salt water intrusion into coastal aquifers or water influx into oil reservoirs. Due to the low liquid compressibility, pressure and flow field changes usually occur in a matter of hours or days, while composition changes or fluid bank movement requires months or years. Some tests of oil or water wells monitor pressure changes over hours or days, so fluid movement is minimal. Well test problems are not stiff because only the rapidly changing time scale is of interest. These pressure transient problems illustrate that the ratio of largest to smallest eigenvalue is not an adequate way to define stiffness – the time scale of interest also is important.

The point of these examples is that stiff problems abound in real life, especially for problems governed by coupled systems of differential equations.

All three of the methods tried so far have difficulties with either accuracy (truncation error) or stability. There are two families of methods that go beyond these basic methods. These are called (1) multistep (MS) methods and (2) Runge-Kutta (RK) methods. The methods used so far involve conditions only at the beginning and end of one step and one operation is required to advance to the next step. Multistep methods use information from earlier steps. Runge-Kutta methods involve no information at earlier steps, but instead use a series of stage calculations to advance to the end of the step. These methods are described briefly. For a more complete discussion of methods for stiff problems, see Hairer and Wanner (1996)

Runge-Kutta Methods

Runge-Kutta methods are one of the most popular families of methods for solving initial value problems. The most familiar ones are explicit, like the forward Euler method. In fact, the forward Euler may be considered the first member of the Runge-Kutta family. The higher order RK methods use multiple stages for each step. These methods are of the form:

$$\boldsymbol{f}_i = \boldsymbol{F}\left(\boldsymbol{y}^k + h(\textstyle\sum_{j=1}^{n_s} a_{ij}\boldsymbol{f}_j),\ z^k + c_i h\right) \text{ for } i = 1,..,n_s$$
$$\boldsymbol{y}^{k+1} = \boldsymbol{y}^k + h\sum_{i=1}^{n_s} b_i \boldsymbol{f}_i \qquad (4.29)$$

where n_s is the number of stages. There are three basic classes of Runge-Kutta methods:

1. Classic explicit methods, $a_{ij} = 0$ for $j \geq i$
2. Diagonally implicit methods (DIRK), $a_{ij} = 0$ for $j > i$
3. General implicit methods

Further subdivision can be made based on additional characteristics of the parameters, a, b and c. The general implicit methods include recognizable names like Gauss, Radau and Lobatto. These methods are not only implicit but require the solution of several implicit stages simultaneously. Simplifications can be made if a single linearization of the equations can be used throughout the step. The DIRK methods are attractive because they require only one level of implicitness at each stage.

The most familiar Runge-Kutta methods are the explicit ones. There are a range of 2-stage second order explicit methods of the form:

$$\boldsymbol{y}^{k+1} - \boldsymbol{y}^k = h\left[(1-\theta)\boldsymbol{F}(\boldsymbol{y}^k, z^k) + \theta\boldsymbol{F}\left(\boldsymbol{y}^k + \frac{h}{2\theta}\boldsymbol{F}(\boldsymbol{y}^k, z^k), z^k + \frac{h}{2\theta}\right) + \right] \qquad (4.30)$$

Any value of θ will give a second order method. The two most popular choices are θ = 1, the modified Euler method, which approximates the midpoint rule and θ = ½, the improved Euler or Heun's method, which approximates the trapezoidal rule.

A popular explicit third order Runge-Kutta method is:

$$\boldsymbol{f}_1 = \boldsymbol{F}(\boldsymbol{y}^k, z^k)$$
$$\boldsymbol{f}_2 = \boldsymbol{F}\left(\boldsymbol{y}^k + \tfrac{h}{2}\boldsymbol{f}_1,\ z^k + \tfrac{h}{2}\right)$$
$$\boldsymbol{f}_3 = \boldsymbol{F}(\boldsymbol{y}^k - h\boldsymbol{f}_1 + 2h\boldsymbol{f}_2,\ z^k + h) \qquad (4.31)$$
$$\boldsymbol{y}^{k+1} = \boldsymbol{y}^k + h(\boldsymbol{f}_1 + 4\boldsymbol{f}_2 + \boldsymbol{f}_3)/6$$

It has the same weights as Simpson's rule. Simpson's rule is fourth order, but since the results at stage 2 and 3 are only

Table 4.6
Butcher Tableau

c_1	a_{11}	a_{12}	a_{13}	a_{14}
c_2	a_{21}	a_{22}	a_{23}	a_{24}
c_3	a_{31}	a_{32}	a_{33}	a_{34}
c_4	a_{41}	a_{42}	a_{43}	a_{44}
	b_1	b_2	b_3	b_4

Table 4.7
2nd Order Eq. (4.30)

0	0	0
$\frac{1}{2\theta}$	$\frac{1}{2\theta}$	0
	$1-\theta$	θ

Table 4.8
3rd Order Eq. (4.31)

0	0	0	0
½	½	0	0
1	-1	2	0
	1/6	2/3	1/6

approximate, this three stage RK method is third order. The classic fourth order RK method also approximates Simpson's rule, but an additional stage gives a refined estimate of the middle point to achieve the same convergence order as Simpson's rule.

The parameters for these methods are frequently represented using a Butcher tableau. The tableau for a generic four stage method is shown in Table 4.6, while the tableaus are listed for the explicit methods of second order and third order, in Tables 4.7 and 4.8, respectively.

Fig. 4.20 shows calculations using a 2nd order Runge-Kutta method, the improved Euler or Heun's method, which corresponds to $\theta = ½$ in Eq. (4.30) and Table 4.7. This method improves on the accuracy of the Euler method, and in fact is almost as accurate as the trapezoidal rule which it approximates. However, it has exactly the same stable step size restriction as the forward Euler method, i.e. $h < -2/\lambda$. The 3rd and 4th order explicit Runge-Kutta methods have stability limits of $h < -2.51/\lambda$ and $h < -2.79/\lambda$ for real eigenvalues, respectively. There is only a small increase in the stability limit for a substantial increase in calculation effort. Since the calculations required for these methods are proportional to the number of stages, the relative efficiency for stiff problems is given by normalizing the stability limit by the number of stages. The normalized stability limit is 2, 1, 0.84 and 0.70 for 1st through 4th order, respectively. These methods are unsuitable for stiff problems. The first order Euler method is not accurate enough and the others have serious step size restrictions. Greater implicitness is required.

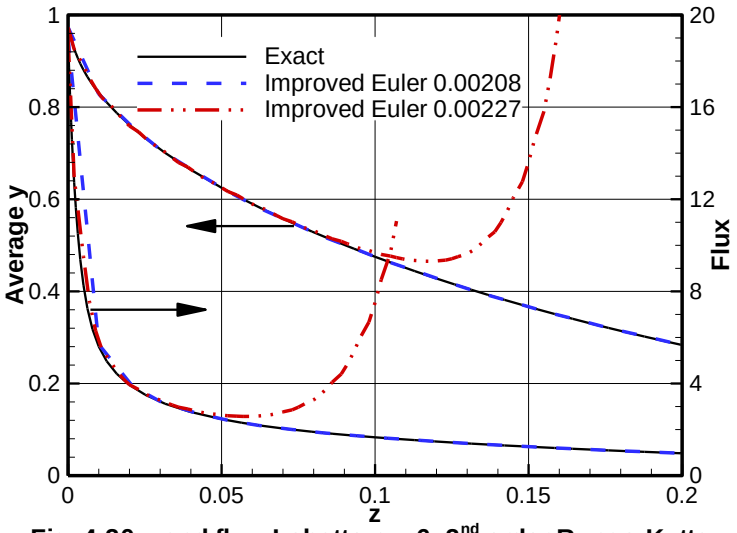


Fig. 4.20 y and flux, Lobatto n = 6, 2nd order Runge-Kutta

For problems like this example and many others with large n, A-stable and L-stable methods are needed. Implicit Runge-Kutta methods are a class of methods to consider. Of these, the diagonal implicit methods (DIRK) are most attractive, since they do not require the implicit solution of several stages simultaneously. There are hundreds of these methods, which are nicely summarized and tested by Kennedy and Carpenter (2016). There are several subcategories of DIRK methods. When all diagonal entries in the tableau are equal, the method is called singly diagonally implicit (SDIRK). If the first stage is explicit, it is called an EDIRK method. If the first stage is explicit and the others have the same diagonal, it is designated an ESDIRK method. A few simple 2nd and 3rd order methods of this type are considered, but the computer codes are written for a general tableau, so other methods can be added easily.

Tables 4.9 and 4.10 give the tableaus for two 2nd order methods, a 2 stage SDIRK method and a 3 stage ESDIRK method. The parameters are given to 5 digits in the tables, but more

accurate values are contained in Kennedy and Carpenter (2016, pp 71-72) and in the example codes. The diagonal in both these methods is $(2-\sqrt{2})/2$. It is possible to achieve 3rd order accuracy and A-stability with the same number of stages. However, to also achieve L-stability with 2 implicit stages the order falls to 2nd order. These two methods are quite similar. Both may be viewed as starting with an initially step of $0.5858h$. The method in Table 4.9 uses a midpoint rule for the first step, while the one in Table 4.10 uses a trapezoidal rule initially. The last step achieves the L-stable property while retaining 2nd order accuracy. For our linear example problem with constant coefficients the trapezoidal rule and midpoint rule are identical, so these two methods produce identical results.

Table 4.9
2nd Order Implicit (SDIRK)

0.29289	0.29289	0
1.00000	0.70711	0.29289
	0.70711	0.29289

Table 4.10
2nd Order Implicit (ESDIRK)

0	0	0	0
0.58579	0.29289	0.29289	0
1.00000	0.35355	0.35355	0.29289
	0.35355	0.35355	0.29289

Implementing a DIRK method for our example problem requires at each implicit stage the solution of:

$$\left(\frac{\widehat{\boldsymbol{W}}}{\tilde{a}_{ii}}+\widehat{\boldsymbol{C}}\right)\left(\boldsymbol{y}_{i-1}+\tilde{a}_{ii}\boldsymbol{f}_i\right)=\left(\frac{\widehat{\boldsymbol{W}}}{\tilde{a}_{ii}}\right)\boldsymbol{y}_{i-1} \tag{4.32}$$

where: $\boldsymbol{y}_{i-1}=\boldsymbol{y}^k+\sum_{j=1}^{i-1}\tilde{a}_{ij}\boldsymbol{f}_j$ and $\tilde{a}_{ij}=ha_{ij}$. Fig. 4.21 shows the calculated results with either of the methods in Tables 4.9 or 4.10. These methods show some overshoot on the first step, but this step is quite large given the large gradients at small z. They quickly recover and show excellent accuracy and overall behavior, far superior to the results shown in Fig. 4.19 calculated with the trapezoidal rule.

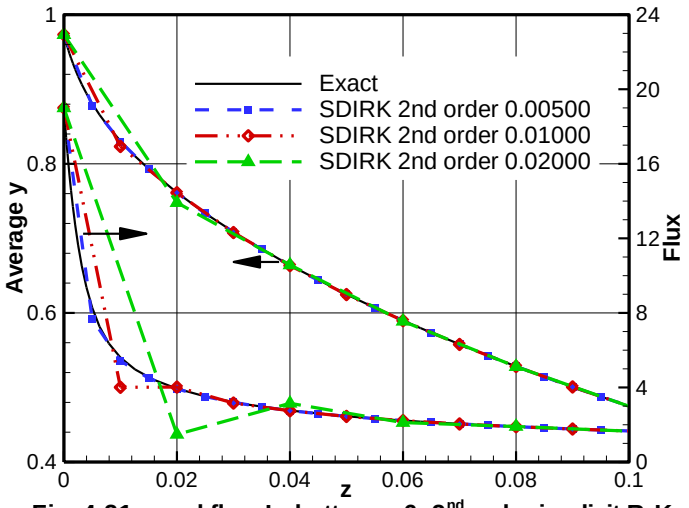


Fig. 4.21 y and flux, Lobatto n = 6, 2nd order implicit R-K

Calculations were also performed using the two third order methods shown in Tables 4.11 and 4.12. These methods are described by Kennedy and Carpenter (2016, pp.77-81). Both methods are L-stable. The first is an SDIRK method with three stages and the second is

Table 4.11
3rd Order Implicit (SDIRK)

0.43587	0.43587	0	0
0.71793	0.28207	0.43587	0
1.00000	1.20850	-0.64436	0.43587
	1.20850	-0.64436	0.43587

Table 4.12
3rd Order Implicit (ESDIRK)

0	0	0	0	0	0
0.45000	0.22500	0.22500	0	0	0
0.76820	0.27160	0.27160	0.22500	0	0
0.60000	0.22374	0.22374	-0.07249	0.22500	0
1.00000	0.17555	0.17555	-0.34686	0.77077	0.22500
	0.17555	0.17555	-0.34686	0.77077	0.22500

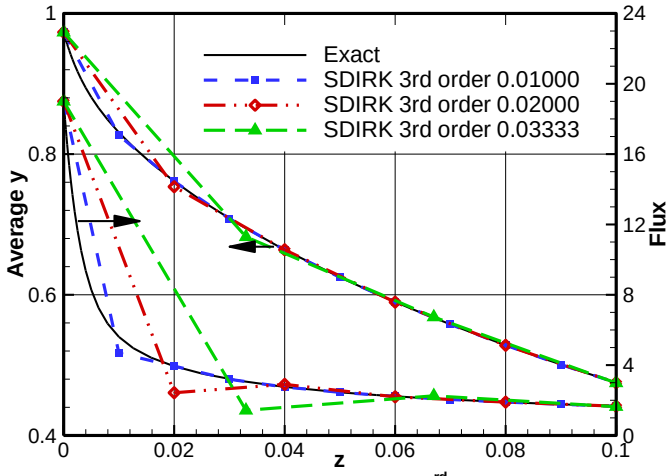


Fig. 4.22 y and flux, Lobatto n = 6, 3rd Order Implicit R-K

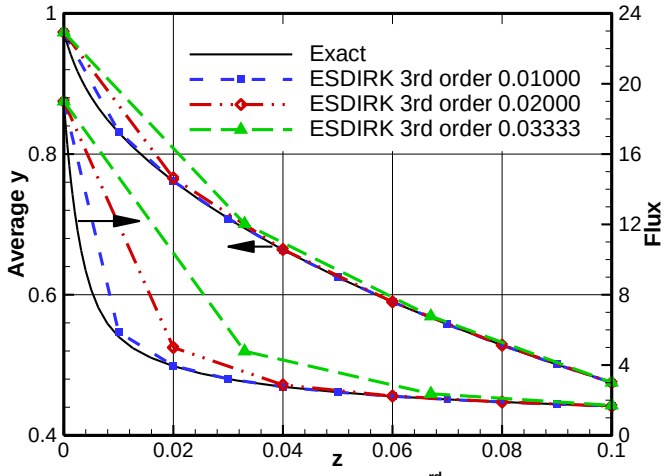


Fig. 4.23 y and flux, Lobatto n = 6, 3rd Order Implicit R-K

an ESDIRK method with an explicit first stage followed by four implicit stages. An L-stable ESDIRK with three implicit stages has also been formulated, and for our test problem it produces identical results to the method in Table 4.11. Figs. 4.22 and 4.23 shows results calculated with these methods. Both methods are well behaved and accurate, but they require substantial calculations per step. They are accurate enough with the large steps that additional values are needed for points within the step rather than using the linear representation depicted in the figures. This is called *dense output* and many of the implicit Runge-Kutta methods give interpolation formulas for calculating intermediate values of the solution.

Multistep methods

Multistep methods are another popular family of methods for solving initial value problems. They are of the form:

$$y^{k+1} = \sum_{j=k-n_b}^{k} \alpha_j \boldsymbol{y}^j + \sum_{j=k-n_b}^{k+1} h\,\beta_j \boldsymbol{F}(\boldsymbol{y}^j, z^j) \qquad (4.33)$$

Where n_b is the number of conditions considered from earlier steps. The methods we will consider are the classic Adams-Bashforth and Adams-Moulton methods, along with higher order backward difference methods. These methods have issues with starting and step size changes. Starting can be accomplished using one of the single step methods, e.g. an implicit RK method. For a constant step size, the parameters of Eq. (4.33) for these methods are given in Table 4.13 together with those for the Euler, backward Euler and trapezoidal rule. For the Adams methods, the parameters are interpolatory quadrature weights, while for the backward difference methods they are differentiation matrix values. Chapter 2 describes methods for calculating such quantities, so the calculations could easily be adapted to calculate the coefficients for multistep methods with a variable step size.

In addition to the parameter values, Table 4.13 lists basic stability properties. For conditionally stable methods, parameter a is listed in the condition $h < -a/\lambda$ for real eigenvalues. A-stable

Table 4.13 Multistep Methods

Method	α values			β values				Stability	L-stable
	k	k-1	k-2	k+1	k	k-1	k-2		
Euler	1	0	0	0	1	0	0	2	no
Backward Euler	1	0	0	1	0	0	0	A-stable	yes
Trapezoidal Rule	1	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0	A-stable	no
2nd Order Adams-Bashforth	1	0	0	0	$\frac{3}{2}$	$\frac{-1}{2}$	0	1	no
3rd Order Adams-Bashforth	1	0	0	0	$\frac{23}{12}$	$\frac{-4}{3}$	$\frac{5}{12}$	0.545	no
3rd Order Adams-Moulton	1	0	0	$\frac{5}{12}$	$\frac{2}{3}$	$\frac{-1}{12}$	0	6	no
4th Order Adams-Moulton	1	0	0	$\frac{3}{8}$	$\frac{19}{24}$	$\frac{-5}{24}$	$\frac{1}{24}$	3	no
2nd Order Backward Difference	$\frac{4}{3}$	$\frac{-1}{3}$	0	$\frac{2}{3}$	0	0	0	A-stable	yes
3rd Order Backward Difference	$\frac{18}{11}$	$\frac{-9}{11}$	$\frac{2}{11}$	$\frac{6}{11}$	0	0	0	A(α)-stable	yes*

methods are so designated. The 3rd order backward difference method is listed as A(α)-stable. This is yet another method for describing stability and indicates the method is stable for all eigenvalues that fall within an angle $\pm\alpha$ of the negative real axis. For the 3rd order backward difference method α = 86.03º, so it is A-stable for practical purposes. Since it is not quite A-stable, it does not meet the formal definition for L-stability. However, a yes with an asterisk is used since it reduces to a pseudo-steady approximation, Eq. (4.28), when the steps are large relative to the coefficients of the z derivatives. The only methods with attractive stability properties are the backward difference methods. The only other method which comes close to meeting our stability requirements is the 3rd order Adams-Moulton. Although the stability limit is three times that of Euler method, it is still a limiting factor and the method is not L-stable, so it does not make the cut.

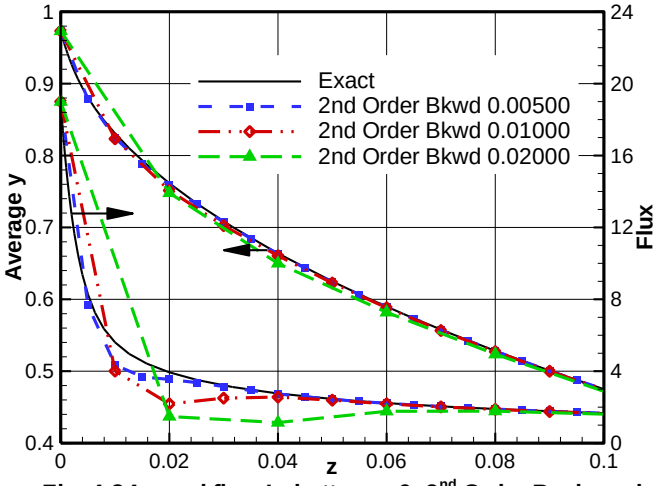


Fig. 4.24 y and flux, Lobatto n = 6, 2nd Order Backward

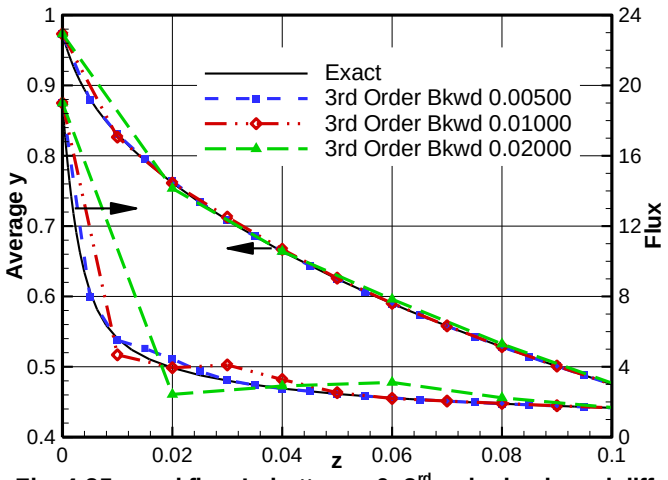


Fig. 4.25 y and flux, Lobatto n = 6, 3rd order backward diff.

The only multistep methods tested were the 2nd and 3rd order backward difference methods, which were started by the SDIRK methods in Tables 4.9 and 4.11, respectively. The calculations with these methods are shown in Figs. 4.24 and 4.25. The overall behavior and accuracy of these methods is very good, especially considering they require essentially the same calculation effort as the backward Euler method. However, the most difficult part of the example problem are the first one or two steps which were computed by the more accurate and calculation intensive SDIRK methods. We will have to reserve judgement on these methods until further testing is done.

Comparison of Methods

So far, we have only made general qualitative comments about the accuracy of the various methods. Fig. 4.26 plots the error in the flux at z = 0.10 vs the step size on a log-log scale. For small h, the first order (red), 2nd order (blue) and 3rd order (green) methods settle at a slope equal to their order. The slope for the forward Euler and improved Euler methods go vertical at h = 0.021 where they become unstable. The other methods deviate from their asymptotic behavior at larger step sizes. Kennedy and Carpenter describe a "superconvergence-like" behavior of the 3rd order ESDIRK method, which is evident from h = 0.010 to 0.033 where the error drops precipitously, by almost 5 orders of magnitude.

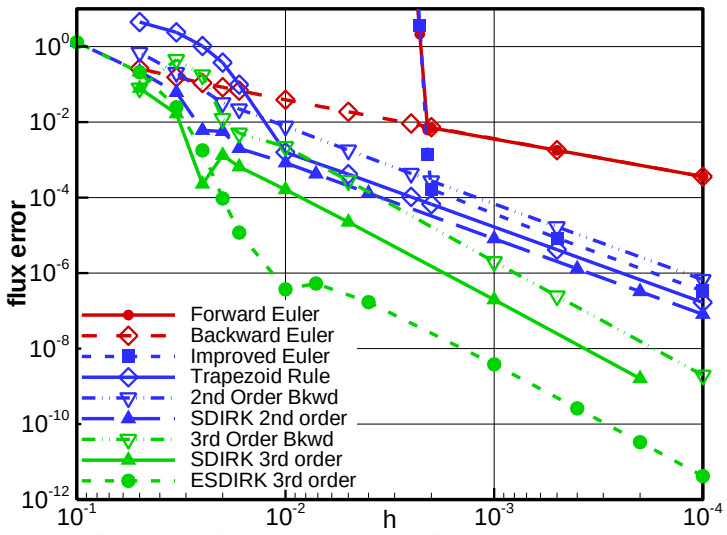


Fig. 4.26 Flux error vs step size at z = 0.10

It is difficult to compare the methods based on Fig. 4.26 alone. For example, the trapezoidal rule (see Fig. 4.19) exhibits unacceptable behavior for the first few steps when h > 0.005. However, the oscillations damp out, so the unacceptable behavior does not show up in Fig. 4.26. Also, some of the methods, e.g. backward difference methods, require little work per step while others require considerably more effort for each step, e.g. the ESDIRK in Table 4.12.

Table 4.14 Flux Error at z = 0.1 and Work Summary

	Step Size for Error			Work per step	Work for Error		
	1.00%	0.10%	0.01%		1.00%	0.10%	0.01%
Forward Euler	**	0.0003	0.00003	1	**	1434	14415
Backward Euler	0.003	0.0003	0.00003	1	147	1443	14361
Improved Euler	**	**	0.00159	1	**	**	252
Trapezoidal Rule	0.013	0.0079	0.00245	1	32	51	163
2nd Order Backward	0.011	0.0038	0.00121	1	35	107	331
2nd Order SDIRK	0.027	0.0111	0.00349	2	30	72	229
3rd Order Backward	0.019	0.0076	0.00358	1	21	52	112
3rd Order SDIRK	0.032	0.0186	0.00843	3	37	64	142
3rd Order ESDIRK	0.030	0.0239	0.02008	4	53	67	80

This simple linear problem is not an ideal one for comparing methods, but in an effort to gain as much insight as possible, the results are summarized in Table 4.14. The table gives the step size needed for a given accuracy, and then combines that with a work per step estimate. For a more typical nonlinear problem, the work per step for an implicit method will be many times that for an explicit one. Since most of the methods are implicit, perhaps this issue does not skew the comparison. One can always modify the values in the work per step column if other numbers are preferred.

Table 4.14 seems to adequately isolate some of the better methods for further consideration. The results show the usual trend that a greater required accuracy favors higher order methods and there is a large improvement of 2nd order methods over 1st order ones. The 3rd order backward difference method comes out surprisingly good, because it works reasonably well and requires little work per step. However, keep in mind that it is started with the 3rd order SDIRK method, so for h = 0.02 over 70% of the implicit solves are during the two startup steps. Nevertheless, it shows to be a competitive method even when the accuracy requirements are greater and startup has less influence. The trapezoidal rule appears to be surprisingly competitive. Its truncation error coefficient is half that of the 2nd order backward difference method, so the backward difference method will be better only for problems more stiff than the example. The 3rd order DIRK methods do quite well considering the amount of work required per step. They generate accurate solutions with only 4 or 5 steps to reach z = 0.10. All of these results were generated using a constant step size, while the example would benefit from small steps initially followed by an increasing step size.

This problem shows how to accurately apply the orthogonal collocation or pseudo spectral method to a parabolic problem and demonstrates how to solve the resulting coupled set of ordinary differential equations. The coupled ODEs are solved both analytically and by various numerical methods. Although the example is a simple linear problem, it clearly shows some of the issues and tradeoffs that are characteristics of these problems. We hope to follow this simple linear example by more complicated nonlinear ones. The investigation of other numerical stepping methods would also be of interest.

5. Hyperbolic Problems – The Wave Equation

In this chapter hyperbolic problems are treated with global methods.

Specifically, we examine the damped wave equation in the context of a model of an automotive valve train.

6. Finite Elements in One Dimension

Previous chapters have covered all the basic types of differential equations with global trial functions, this chapter extends the methods to trial functions that are piecewise continuous. This approach is better known as a *finite element method*, but the names *spectral element method* and *differential quadrature element method* are used also. The extension to finite elements is conceptually simple. However, one must keep track of points within elements which requires more complex bookkeeping.

Fig. 1.3 shows an example of a one dimensional finite element grid composed of simple piecewise linear trial functions. A grid is constructed by first defining the element boundary nodes:

$$x_0 < x_1 < \cdots < x_k < \cdots < x_{n_e} \tag{6.1}$$

Then the size of each element is defined by $\Delta x_k = x_k - x_{k-1}$, for $k = 1,\ldots,n_e$. Within each element a local coordinate is used, $\xi = (x - x_{k-1})/\Delta x_k$. Internal nodes are usually at polynomial roots or quadrature points, ξ_i, so a node i in element k is given by $x_{ik} = x_{k-1} + \Delta x_k \xi_i$ for $i = 0,\ldots,n+1$. The double and single subscript notation are related by $x_k = x_{n+1,k} = x_{0,k+1}$. This subscript notation will be used throughout this chapter, i.e. a single subscript on x or y denotes the number of an element interface while a double subscript denotes a point within an element.

The element interfaces are like internal boundaries. For a second order differential equation, the conditions at the element interfaces are:

$$\begin{aligned} y(x_{n+1,k}) &= y(x_{0,k+1}) \\ \left.\frac{dy}{dx}\right|_{x_{n+1,k}} &= \left.\frac{dy}{dx}\right|_{x_{0,k+1}} \end{aligned} \tag{6.2}$$

If one thinks in terms of heat transfer, the second condition requires the flux to be continuous. The following shows how finite element methods follow naturally from the global methods studied in the previous chapters.

6.1 Finite Element Trial and Weight Functions

Several types of finite element trial functions are possible. In finite element literature they are traditionally called *shape functions*. If one is solving for displacements of a structure, the shape functions literally dictate the contour of the solution. In one dimension, trial functions differ by the degree of the polynomial and the continuity imposed at the element interface nodes. Regarding continuity, the first requirement is that the weighted residual must be integrable. If applied in strong form, like Eq. (1.9), first derivatives must be continuous or C^1, but weight functions can be discontinuous or C^{-1}. If the Galerkin method is applied in weak form, like Eq. (1.10) or (3.22), then only simple C^0 continuity is required for both the trial function and weight function. For the Galerkin method, the continuity of the derivatives can be weakly imposed by treating them as natural boundary conditions.

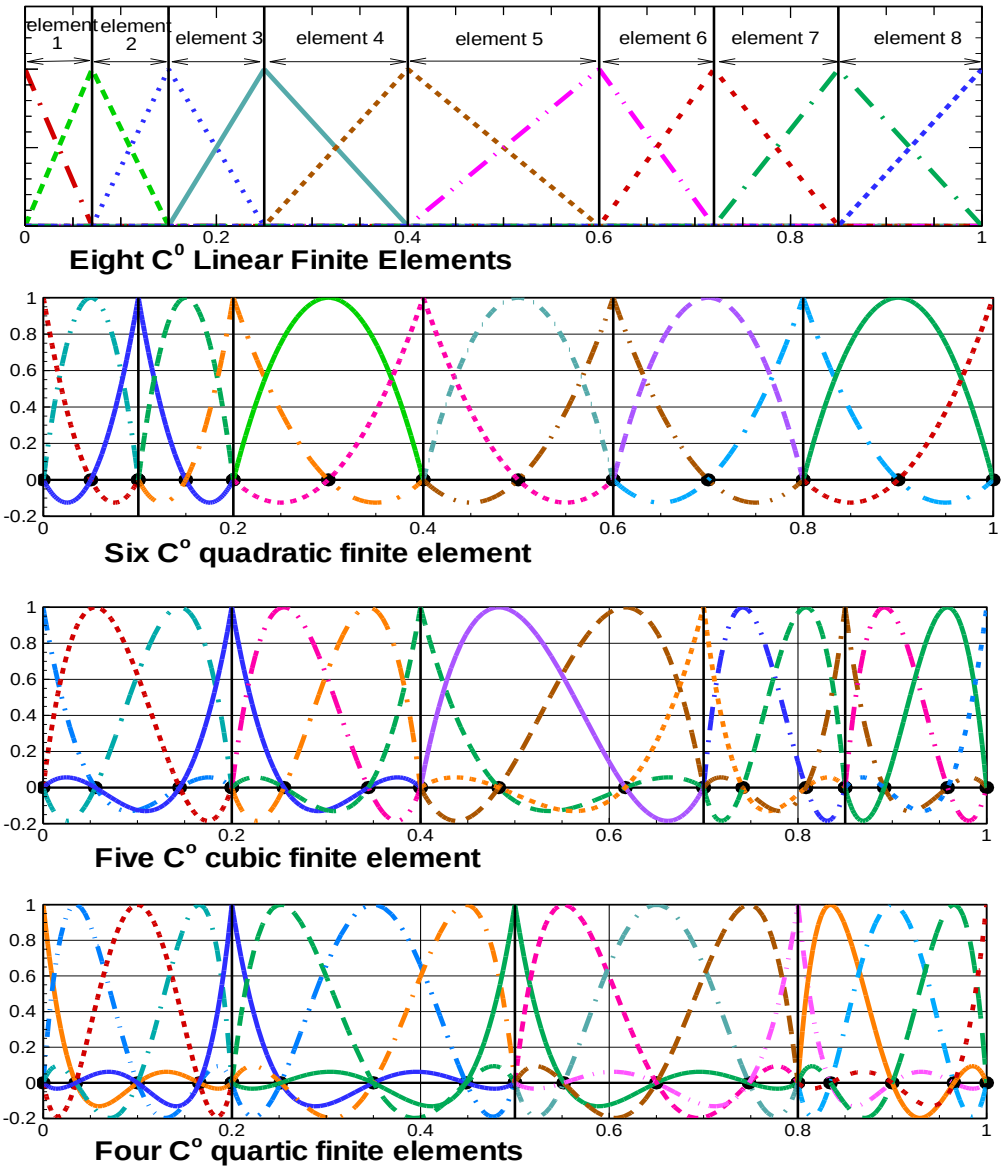


Fig. 6.1 Finite element trial or shape functions, C^0 continuity

The simplest C^0 elements are the piecewise linear functions shown in Fig. 1.3. These are the most common shape functions. Fig. 6.1 shows the linear functions together with higher order C^0 functions. The functions within each element are Lagrange polynomials and the nodes are located at quadrature points. In this example, the nodes are at Lobatto points. The functions at the element interfaces span two elements, while those associated with internal nodes are contained within one element.

Other degrees of continuity have been used ranging from spline functions to discontinuous, C^{-1}, functions. We will consider the C^0 trial functions in Fig. 6.1 and functions which are C^1. The strong enforcement of the derivative condition of Eq. (6.2) can be achieved using the C^0 shape functions together with side conditions or automatically by using C^1 functions with built-in continuity.

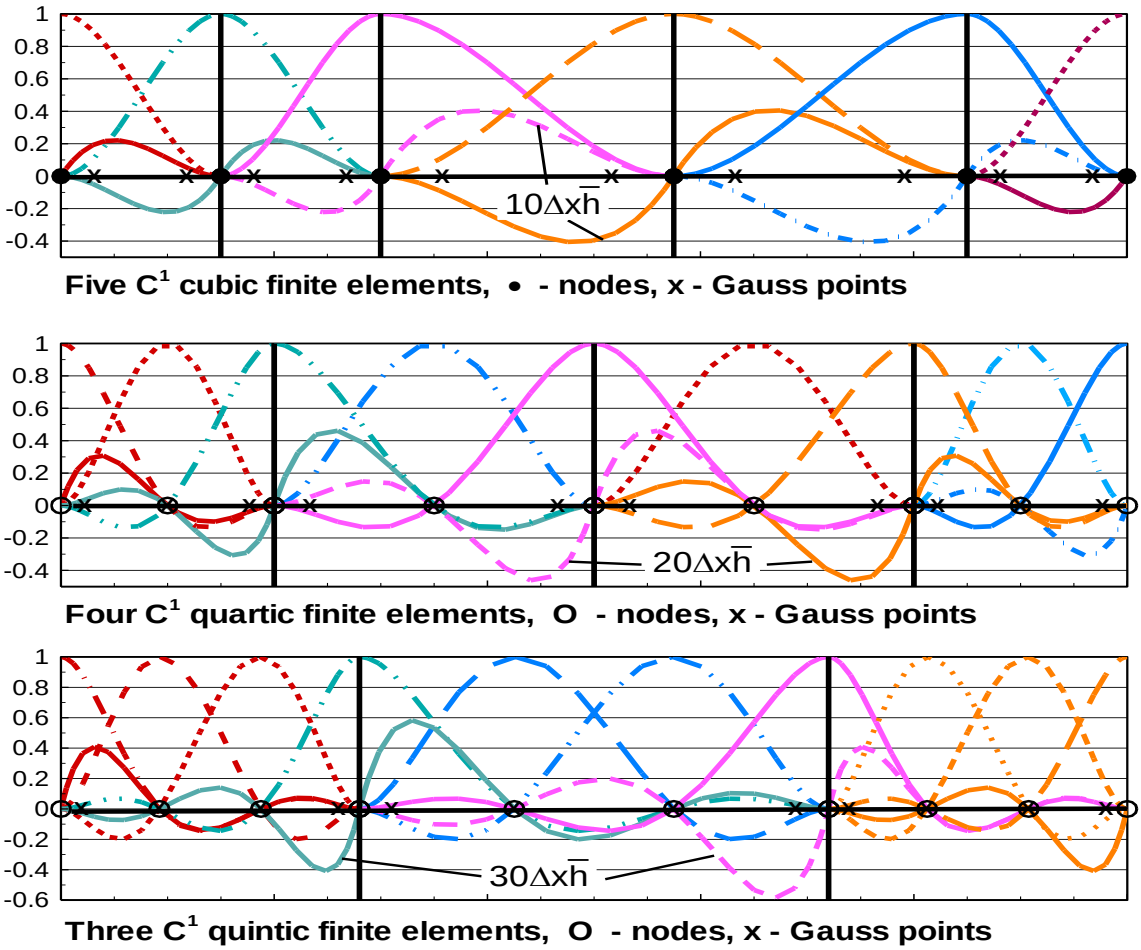


Fig. 6.2 Element trial or shape functions with C^1 continuity

The simplest trial functions with built-in C^1 continuity are the cubic Hermite functions used in the first example problem of section 1.3, Eq. (1.23) [Hildebrand (1987), p. 282]. Section 2.8 describes the extension of this approach to higher degree functions by interpolating the function and its derivative at the end nodes together with function values at internal nodes. With this approach, the solution is represented in local coordinates by:

$$\tilde{y} = \sum_{i=1}^{n} h_i(\xi)\tilde{y}(\xi_i) + \bar{h}_0(\xi)\tilde{y}'(0) + \bar{h}_1(\xi)\tilde{y}'(1) \tag{6.3}$$

Where $\xi_1 = 0$ and $\xi_n = 1$. The degree of the polynomial is $n + 1$. The functions obey the conditions: $h_i(\xi_j) = \delta_{ij}, h_i'(1) = h_i'(0) = 0, \text{and } \bar{h}_i(\xi_j) = 0,\ \bar{h}_0'(0) = \bar{h}_1'(1) = 1, \bar{h}_0'(1) = \bar{h}_1'(0) = 0$. The interior nodes are usually located at quadrature points to reduce interpolating requirements. Some examples of higher order C^1 functions are displayed in Fig. 6.2. In Eq. (6.3) the derivative are with respect to ξ, but it is the derivatives with respect to x which are continuous. The functions must be scaled to account for variations in grid size. If there is a change in material properties, such as thermal conductivity, the scaling must take this into consideration also. In the figure the $\Delta x\bar{h}$ functions are scaled up as indicated. In these examples, the functions are constructed to ease the use Gaussian quadrature, so the internal

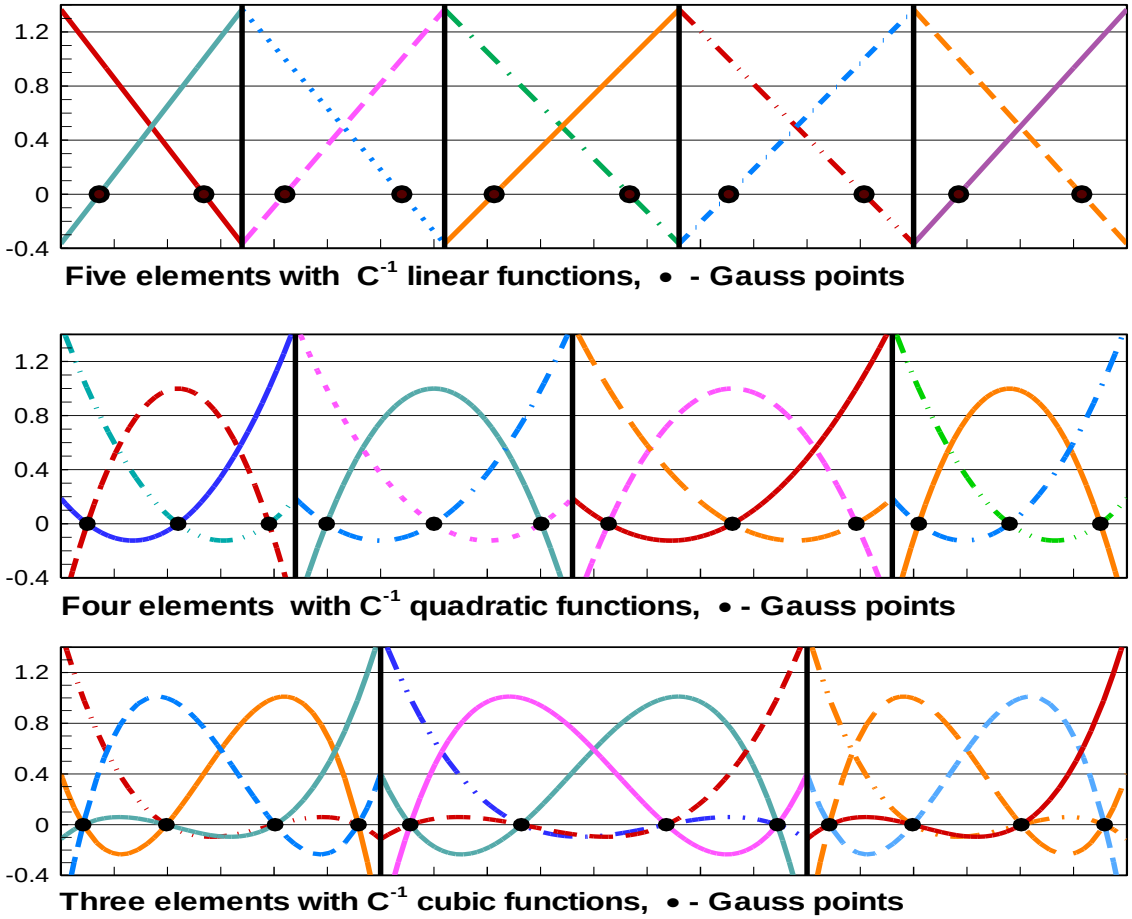


Fig. 6.3 Discontinuous C^{-1} weight functions

nodes are located at Gauss points. With the Lagrange functions of Fig. 6.1, one would have nodes at all of the Gauss points. In these examples, the derivative conditions are substituted for the nodes nearest the boundary. When the derivatives are strongly enforced these trial functions reduce the size of the algebraic problem by $n_e - 1$; however, some additional calculations are often needed for interpolation of some terms.

If a strong treatment is used and the trial functions are C^1, the weight functions can be discontinuous or C^{-1}. Fig. 6.3 shows examples of discontinuous weight functions. This example uses Lagrange polynomials through Gauss points, so that simplification occur when Gaussian quadrature is used. Note, these functions meet at the element boundaries, but they are separate functions as indicated by the different line types in the figure.

6.2 C^1 Collocation and Method of Moments

For global methods, we found that collocation works best when it approximates one of the integral-type MWR like the Galerkin method or method of moments. Here we demonstrate the extension to finite elements for the problem of section 3.1. The method of moments was treated in section 3.1.2. It uses a strong treatment of boundary conditions, including the element interface conditions. The strong treatment can be achieved using either the C^0

Lagrange functions of Fig. 6.1 together with side conditions or by using the functions of Eq. (6.3) with built-in C^1 continuity like those in Fig. 6.2. In either case, the weight functions are discontinuous functions, like those in Fig. 6.3. The Lagrange approach is considered first.

The method of moments is equivalent to the H^1 Galerkin method [Douglas, et al. (1974)]. The H^1 Galerkin method is not a Galerkin method in the classic sense, because it does not weight the residual by the trial functions. Instead, it uses trial functions which are C^1 and weights the residual by the second derivative of the trial functions. The second derivatives of the trial functions are C^{-1} functions. The resulting approximation is equivalent to that produced by the weight functions in Fig. 6.3. However, the approach used here will produce a more compact approximation because the weights span only one element. The H^1 Galerkin method is a classic method of moments as studied extensively by Kravchuk (see section 1.2.4 and Kravchuk (1926,1932), Lucka and Lucka (1992)).

In the 1970s, the success of the FEM in structural mechanics lead to interest in other disciplines. When first applied to nonlinear and time dependent problems, conventional formulations were found to be computationally intensive due in part to the numerical quadratures. Collocation-like methods seemed like an obvious way to reduce computations. The first collocation FEM was collocation at Gauss points with early articles by DeBoor and Swartz (1973), Douglas and Dupont (1973), and Cary and Finlayson (1975). This method is an almost trivial extension of the global collocation methods described in earlier chapters. As we shall see, the method of moments or H^1 Galerkin method is approximated by collocation at Gauss points.

The residual weighting within each element is by the reduced Lagrange polynomials of Eq. (3.16) and Fig. 6.3, designated as ℓ^*. These polynomials interpolate through interior quadrature points, Gauss points in this case. For our test problem, the method of moments gives:

$$\frac{1}{\Delta x_k}\sum_{i=0}^{n+1} y_{ik}\int_0^1 \ell_j^*\ell_i''\,d\xi + \Delta x_k\int_0^1 \ell_j^* r_k(x,\tilde{y})\,d\xi = \frac{W_j}{\Delta x_k}\sum_{i=0}^{n+1} B_{ji}\,y_{ik} + \Delta x_k\int_0^1 \ell_j^* r_k\,d\xi = 0 \qquad (6.4)$$

for j = 1,…,n and k = 1,…,n_e, which is like Eq. (3.17) but scaled for an element k. The subscript on r indicates the integration is for values in element k. The differentiation matrix, $\boldsymbol{B}$, is the same as before and $\boldsymbol{W}$ are the Gaussian quadrature weights. These equations at the interior points are combined with a strong treatment of the interface conditions, Eq. (6.2):

$$\frac{1}{\Delta x_k}\sum_{i=0}^{n+1} A_{n+1,i}y_{ik} - \frac{1}{\Delta x_{k+1}}\sum_{i=0}^{n+1} A_{0,i}y_{i,k+1} = 0 \qquad (6.5)$$

A strong treatment of the external boundary conditions is used also:

$$\frac{1}{\Delta x_1}\sum_{i=0}^{n+1} A_{0,i}y_{i1} = 2Bi_0\,y_{01} \quad \text{and} \quad -\frac{1}{\Delta x_{n_e}}\sum_{i=0}^{n+1} A_{n+1,i}y_{i,n_e} = 2Bi_1\,y_{n+1,n_e} \qquad (6.6)$$

For a linear rate term, the full mass matrix produced from exact integration is given in section 2.7 and used in section 3.1.2. Accurate integration of a nonlinear rate term is described in the next section, see Eq. (6.25). However, if the rate term is approximated with n point Gaussian quadrature, the equations simplify to:

$$\frac{W_j}{\Delta x_k}\sum_{i=0}^{n+1} B_{ji}\, y_{ik} + \Delta x_k W_j\, r(x_{jk}, y_{jk}) = \Delta x_k W_j R_{jk} = 0 \tag{6.7}$$

for j = 1,…,n and k = 1,…,n_e. $\boldsymbol{R}$ is the residual. This form reveals the method to be a collocation method since it is equivalent to setting the residual to zero. This equation is Eq. (3.10) or Eq. (3.18) after scaling for element size. This is like the form of the equations treated by Carey and Finlayson (1975); however, the scaling here gives a symmetric matrix problem. By substitution of the symmetric stiffness matrix $\boldsymbol{C}$ for $\boldsymbol{A}$ and $\boldsymbol{WB}$, see Eq. (2.111), the matrix symmetry is evident.

There are $(n)n_e$ equations at interior nodes, n_e - 1 element interface conditions and 2 boundary conditions for a total of $(n+1)n_e+1$ equations. This total does not count the trivial condition $y(x_{n+1,k}) = y(x_{0,k+1})$ which is handled by the labeling or numbering of the unknowns. The structure of the algebraic problem is shown in Fig. 6.4. The n = 0 case is not applicable for the

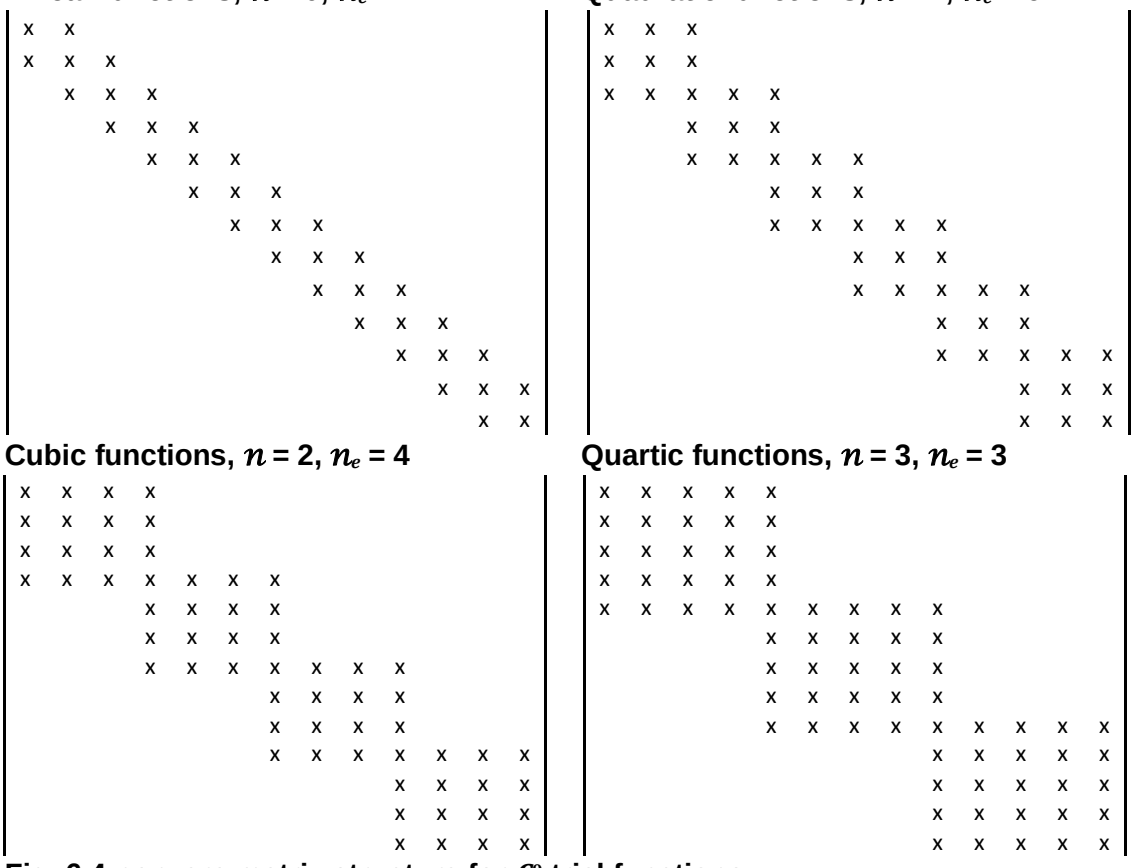


Fig. 6.4 nonzero matrix structure for C^0 trial functions

method of moments but is included in the figure for subsequent discussion. The equations can be viewed as either banded or block tridiagonal matrices. The band width is $2n$+3. The maximum bandwidth occurs for the linear element interface condition.

These equations would have identical form for collocation at Lobatto or Chebyshev points; however, from earlier chapters, we know the strong enforcement of the derivative conditions is good only for collocation at Gauss points or the method of moments. For other cases, a weak or natural boundary condition treatment should be used.

Rather than using side conditions, trial functions with built-in C^1 continuity, Eq. (6.3), can be used. To simplify the terminology, the trial functions are numbered from 0 to n+1 by $\boldsymbol{H_k} = \{h_1, \Delta x_k \bar{h}_0, h_2, \ldots, h_n, \Delta x_k \bar{h}_1\}$ and the parameters are correspondingly numbered $\boldsymbol{a_k} = \{\tilde{y}(x_{k-1}), \tilde{y}'(x_{k-1}), \tilde{y}(x_{2,k}), \ldots, \tilde{y}(x_{n-1,k}), \tilde{y}(x_k), \tilde{y}'(x_k)\}$. In $\boldsymbol{a_k}$, the derivatives are with respect to x not ξ, since it is those derivatives which are continuous. Some additional scaling of the derivatives may be desirable in some cases. Also, if there is a material change at the interface, such as diffusivity or thermal conductivity, the definition should incorporate the change to achieve continuity of flux. Using our notation, the singly subscripted parameters are the interface nodes and the internal nodes are at quadrature points, $x_{ik} = x_{k-1} + \xi_i \Delta x_k$. The quadrature points are either Gauss or Lobatto points. Like in Fig. 6.2, the two closest to the element boundary, ξ_1 and ξ_n, are not included as nodal points in the free parameters, $\boldsymbol{a_k}$.

Application of the method of moments gives:

$$\frac{1}{\Delta x_k} \sum_{i=0}^{n+1} a_{ik} \int_0^1 \ell_j^* H_{ki}'' d\xi + \Delta x_k \int_0^1 \ell_j^* r_k d\xi = 0 \tag{6.8}$$

for j = 1,…,n and k = 1,…n_e. The integrand of the first term is degree $2n$-2 and a linear rate term is degree $2n$. If the integration is performed using quadrature, then:

$$\sum_{\ell=0}^{m+1} W_\ell \ell_j^*(\xi_\ell) \left[\left(\sum_{i=0}^{n+1} H_{ki}''(\xi_\ell) \frac{a_{ik}}{\Delta x_k} \right) + \Delta x_k r_k\big(x(\xi_\ell), \tilde{y}(\xi_\ell)\big) \right] = 0 \tag{6.9}$$

Where the term in brackets is the residual evaluated at the quadrature points. The trial functions have n nodes with two on the element boundaries. None of the quadrature formulas give exact integration of the second derivative using only the values at the nodes. It should be possible to devise a quadrature scheme using the nodal values and the endpoint derivatives, but this would be cumbersome. We will consider Gauss and Lobatto quadrature with n interior points, which give accuracy of $2n$-1 and $2n$+1, respectively. Both are accurate enough to maintain the maximum convergence rate. For weight functions, we choose the Lagrange polynomials which interpolate through the interior quadrature points (see Fig. 6.3), while for the trial function nodal points we choose all but the two quadrature points nearest the boundary (see Fig. 6.2). With this integration scheme, Eq. (6.9) in terms of residuals is:

$$\sum_{\ell=0}^{m+1} W_\ell \ell_j^*(\xi_\ell) R(x_{\ell k}) = W_0 \ell_j^*(0) R(x_{k-1}^+) + W_{n+1} \ell_j^*(1) R(x_k^-) + W_j R\big(x_{jk}\big) = 0 \tag{6.10}$$

for j = 1,…,n. For the Gauss case, the first two terms are zero, so the method reverts to a collocation method. The residuals are not continuous, so superscript "-" and "+" are used to designate the side of the discontinuity. Note, including the endpoints for Lobatto quadrature causes little difficulty and gives more accurate integrals, but in that case, the method cannot be viewed as a collocation method. Substituting the expression for the residual gives the following for element k:

$$\sum_{i=0}^{n+1} C_{jik} \frac{a_{ik}}{\Delta x_k} - \Delta x_k \sum_{\ell=0}^{m+1} W_\ell \ell_j^*(\xi_\ell) r(x_{\ell k}, \tilde{y}(x_{\ell k})) = 0 \tag{6.11}$$

Where a somewhat obscure stiffness matrix is formed by:

$$C_{jik} = -\sum_{\ell=0}^{n+1} W_\ell \ell_j^*(\xi_\ell) H_{ki}''(\xi_\ell) = W_0 \ell_j^*(0) H_{ki}''(0) + W_{n+1} \ell_j^*(1) H_{ki}''(1) + W_j H_{ki}''\big(\xi_j\big) \tag{6.12}$$

for j = 1,…,n and i = 0,…,n+1. For a nonlinear rate term, the values are linearized about an initial guess or a result from a previous iteration:

$$r(\tilde{y}(x_{\ell k})) \approx r_k\big(\tilde{y}^0(x_{\ell k})\big) + [\tilde{y}(x_{\ell k}) - \tilde{y}^0(x_{\ell k})] \left.\frac{dr}{dy}\right|_{\tilde{y}^0(x_{\ell k})} = r_{\ell k}^0 + r_{\ell k}' \tilde{y}(x_{\ell k}) \tag{6.13}$$

which will be referenced using the shorthand notation at the far right. The x dependence is omitted for convenience. It is usually better to formulate the problem in terms of the change, $\Delta y_{\ell k} = \tilde{y}(x_{\ell k}) - \tilde{y}^0(x_{\ell k})$, but the derivation is simpler using $\tilde{y}(x_{\ell k})$ as unknowns. The rate term is:

$$\begin{aligned} -\sum_{\ell=0}^{m+1} W_\ell \ell_j^*(\xi_\ell) r(\tilde{y}(x_{\ell k})) &= -\sum_{\ell=0}^{m+1} W_\ell \ell_j^*(\xi_\ell) \big(r_{\ell k}^0 + r_{\ell k}' \tilde{y}(x_{\ell k})\big) \\ &= \sum_{i=0}^{n+1} M_{jik} a_{ik} - G_{jk} \end{aligned} \tag{6.14}$$

where $\boldsymbol{M}$ and $\boldsymbol{G}$ are basically a mass matrix and load vector, respectively. The load vector is:

$$G_{jk} = W_0 \ell_j^*(0) r_{0,k}^0 + W_{n+1} \ell_j^*(1) r_{n+1,k}^0 + W_j r_{jk}^0$$

The mass matrix is a little more complicated because some of the $\tilde{y}(x_{\ell k})$ terms must be interpolated. With our labeling of the unknowns $\tilde{y}(x_{0k}) = a_{0,k}, \tilde{y}\big(x_{n+1,k}\big) = a_{n,k}$, and $\tilde{y}\big(x_{jk}\big) = a_{jk}$ for $j = 2, \dots, n-1$. $\tilde{y}\big(x_{1,k}\big)$ and $\tilde{y}\big(x_{n,k}\big)$ are interpolated with Eq. (6.3), so they depend on all coefficients of the element. The mass matrix can be constructed from:

$$
\begin{aligned}
&\sum_{\ell=0}^{m+1} W_\ell \ell_j^*(\xi_\ell) r'_{k\ell} \tilde{y}(x_{k\ell}) \\
&\quad = W_0 \ell_j^*(0) r'_{0,k} a_{0,k} + W_{n+1} \ell_j^*(1) r'_{n+1,k} a_{n,k} + W_j r'_{jk} a_{jk} \quad \text{for } j = 2, \ldots, n-1 \\
&\quad = W_0 \ell_j^*(0) r'_{0,k} a_{0,k} + W_{n+1} \ell_j^*(1) r'_{n+1,k} a_{n,k} + W_j r'_{jk} \sum_{i=0}^{n+1} H_{ki}(\xi_j) a_{ik} \quad \text{for } j = 1, n \\
&\quad = -\sum_{i=0}^{n+1} M_{jik} a_{ik}
\end{aligned}
\tag{6.15}
$$

Unlike other cases, the mass matrix is not diagonal due the required interpolations, but it is sparse. The equations simplify further for the Gauss or collocation case since $W_0 = W_{n+1} = 0$, However, the Lobatto or moments case is only slightly more complicated and requires little additional calculation. With these definitions, Eq. (6.11) reduces to:

$$
\sum_{i=0}^{n+1} \left(\frac{C_{jik}}{\Delta x_k} + \Delta x_k M_{jik} \right) a_{ik} = \Delta x_k G_{jk} \tag{6.16}
$$

where $j = 1,\ldots n$ in each element $k = 1,\ldots,n_e$.

The conditions at the external boundaries are:

$$
\begin{aligned}
&\tilde{y}'(0) = 2Bi_0\, \tilde{y}(0) \quad \text{and} \quad -\tilde{y}'(1) = 2Bi_1 \tilde{y}(1) \quad \text{or} \\
&\quad a_{11} = 2Bi_0\, a_{01} \quad \text{and} \quad -a_{n+1,n_e} = 2Bi_1 a_{n,n_e}
\end{aligned}
\tag{6.17}
$$

One of the boundary parameters is easily eliminated, so that only a single boundary unknown is required. At interface nodes, of course, the values and derivatives are continuous:

$$
a_{nk} = a_{0,k+1} \quad \text{and} \quad a_{n+1,k} = a_{1,k+1} \tag{6.18}
$$

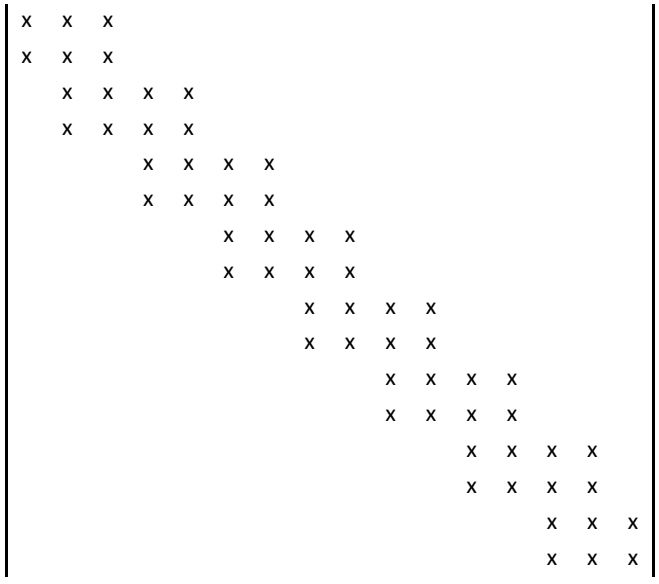


Fig. 6.5 Nonzero matrix structure for moments or collocation, C^1 trial functions

The equal values are treated as single unknowns. With one unknown eliminated at each external boundary, we are left with $(n)n_e$ equations and unknowns. The nonzero matrix structure for these equations is illustrated in Fig.6.5. This matrix may also be viewed as a band matrix of width $2n$ + 1. Compared to the Lagrange approach, this formulation with C^1 trial functions not only reduces the number of unknowns by approximately n_e, but also reduces the band width by 2. In multidimensional problems the differences are more dramatic. Note also, the inclusion of the endpoint weights for Lobatto quadrature adds little complexity to the formulation.

6.3 C^0 Collocation and Galerkin Method

The most common finite element procedures are based on a Galerkin method with C^0 trial functions, like those of Fig. 6.1. Quadrature formulas are normally used to approximate the integrals. Early applications of FEM for nonlinear time dependent problems found the quadrature calculations computationally intensive. Collocation-like methods were an obvious way to reduce the computations. Several equivalent C^0 collocation FEM were developed independently and given different names. First was the *Hybrid-Collocation-Galerkin* method [Diaz (1975,1977), Dunn and Wheeler (1976), Wheeler (1977)]. An alternative method was developed independently by three others [Gray (1977), Young (1977,1981), Hennart (1982)] and called the *Lobatto-Galerkin* method. These methods organize the equations differently. For years, they were thought to be different, but in fact they are equivalent [Young (2019)]. This later formulation was also developed by reformulating the Hybrid-Collocation-Galerkin method [Leyk (1986,1997)]. Still later, the method was rediscovered and popularized as the *Spectral/hp Element* method [Maday and Patera (1989), Karniadakis and Sherwin (2013)]. None of the earlier developments are cited in these or other spectral sources. It has also been called *G-NI* and *SEM-NI* (Galerkin or Spectral Element Method with Numerical Integration) [Canuto, et al. (2007)]. These last names are truly awful since the vast majority of FEM implementations use a Galerkin method with numerical integration. Rightfully, the method should be called the C^0 *Orthogonal Collocation Finite Element Method.*

With C^0 trial functions the approximate solution is continuous, but its derivatives are discontinuous at interface nodes. The derivatives are continuous only in the limit of a fine grid, i.e. large n and/or n_e. Integration of the method must consider the limited continuity of the trial functions. There are two approaches possible: (1) consider the interfaces as internal boundaries and treat the derivatives as natural boundary conditions, or (2) integrate the equations in their weak form, like Eq. (1.10) or (3.22). Both lead to the same result.

If the first approach is applied to the problem of section 3.1. Eq. (3.22) applies directly after scaling for element size, so for an element k:

$$\sum_{i=0}^{n+1} \ell_j \frac{d\ell_i}{d\xi}\bigg|_0^1 \frac{y_{ik}}{\Delta x_k} - \int_0^1 \left(\sum_{i=0}^{n+1} \frac{d\ell_j}{d\xi}\frac{d\ell_i}{d\xi}\frac{y_{ik}}{(\Delta x_k)^2} - \ell_j r_k(x,\tilde{y}) \right) \Delta x_k d\xi = 0 \qquad (6.19)$$

where the subscript on r indicates the integration is performed for values in element k. The first term is the flux at the boundaries. It simplifies and the second derivative term can be replaced by the same stiffness matrix used before, Eq. (2.111):

$$\delta_{j,0}\left.\frac{d\tilde{y}}{dx}\right|_{x_{0,k}} - \delta_{j,n+1}\left.\frac{d\tilde{y}}{dx}\right|_{x_{n+1,k}} + \sum_{i=0}^{n+1}\frac{C_{ji}}{\Delta x_k}y_{ik} - \Delta x_k\int_0^1 \ell_j r_k d\xi = 0 \tag{6.20}$$

The first two terms are zero at internal nodes. At an external boundary, the treatment is like before, so for the first and last nodes:

$$\begin{aligned} &2Bi_0 y_{01} + \sum_{i=0}^{n+1}\frac{C_{0i}}{\Delta x_1}y_{i1} - \Delta x_1\int_0^1 \ell_0 r_1 d\xi = 0 \quad \text{and} \\ &2Bi_1 y_{n+1,n_e} + \sum_{i=0}^{n+1}\frac{C_{n+1,i}}{\Delta x_{n_e}}y_{i,n_e} - \Delta x_{n_e}\int_0^1 \ell_{n+1} r_{n_e} d\xi = 0 \end{aligned} \tag{6.21}$$

Which reduce to $y_{01} = y_{n+1,n_e} = 0$ when the Bi numbers are large.

Eq. (6.2) applies at the interface of two elements. The condition $y_{n+1,k} = y_{0,k+1}$ is implemented by treating the two as a single unknown. Adding Eq. (6.20) for the nodes on each side of the interface produces:

$$\begin{aligned} &\left(\left.\frac{d\tilde{y}}{dx}\right|_{x_{0,k+1}} - \left.\frac{d\tilde{y}}{dx}\right|_{x_{n+1,k}}\right) + \sum_{i=0}^{n+1}\left(\frac{C_{n+1,i}}{\Delta x_k}y_{ik} + \frac{C_{0i}}{\Delta x_{k+1}}y_{i,k+1}\right) \\ &\qquad -\Delta x_k\int_0^1 \ell_{n+1} r_k d\xi - \Delta x_{k+1}\int_0^1 \ell_0 r_{k+1} d\xi = 0 \end{aligned} \tag{6.22}$$

Where the first term in large parenthesis is the interface derivative condition. Using a natural treatment of the derivative conditions, this term is set to zero. The derivatives are not equal, but by setting the term to zero the procedure will force it to approximately zero. This procedure is no different from that used for the treatment of the conditions at external boundaries, see Section 3.1.3. With this natural treatment of the derivative condition, the approximation is:

$$\sum_{i=0}^{n+1}\left(\frac{C_{n+1,i}}{\Delta x_k}y_{ik} + \frac{C_{0i}}{\Delta x_{k+1}}y_{i,k+1}\right) - \Delta x_k\int_0^1 \ell_{n+1} r_k d\xi - \Delta x_{k+1}\int_0^1 \ell_0 r_{k+1} d\xi = 0 \tag{6.23}$$

As was the case for global trial functions, this approach treats the interface derivative condition as part of the approximation which is satisfied in the limit.

If the second approach is used, i.e. direct application of the method in weak form (like Eq. (1.10)), the same result is produced. For interior points there is no contribution from neighboring elements, so Eq. (6.20) remains applicable. For the interface nodes, the weight functions span two elements and the integration is performed by adding the contribution from each element, so Eq. (6.23) results.

The rate term can be treated by the same procedures used for global methods: (1) approximated with Lobatto quadrature and n interior points, (2) approximation using quadrature with more than n quadrature points, or (3) interpolation into the trial space.

The first approach is the simplest, since the rate term is approximated by $\int_0^1 \ell_j r_k d\xi \approx W_j r\left(x_{jk}, y_{jk}\right)$, so these terms appear only on the diagonal of the matrix. This approach and substitution of Eq. (2.111) or (3.20) into Eq. (6.20) reveals the approximation is equivalent to Eq. (6.7) at interior points, so the method is clearly a collocation method at internal nodes. The same substitution for the interface shows that Eq. (6.23) is equivalent to:

$$\left(\frac{1}{\Delta x_k}\sum_{i=0}^{n+1} A_{n+1,i}y_{ik} - \frac{1}{\Delta x_{k+1}}\sum_{i=0}^{n+1} A_{0,i}y_{i,k+1}\right) - \Delta x_k W_{n+1}R_{n+1,k} + \Delta x_{k+1}W_0 R_{0,k+1} = 0 \tag{6.24}$$

The first term, in large parenthesis, is the residual of the interface derivative or flux condition, Eq. (6.5), while R is the residual of the differential equation defined in Eq. (6.7). At the interface nodes this collocation method sets a combination of interior and boundary residuals to zero. All residuals converge to zero as the grid is refined. This relationship is like that for external boundaries with global methods, i.e. Eq. (3.30). In this form, the treatment can be viewed as a *penalty method*, like Eq. (1.15).

If the rate term is nonlinear and a more accurate integration is desired, a procedure like that in section 3.1.5 can be used, i.e. approximate integration with $m > n$ interior quadrature points. Results in Chapter 3 show that one or two additional quadrature points usually produces the smallest error. The error with $m \gg n$ is sometimes marginally less than with $m = n + 1$, but at other times the error can be larger.

Given an initial guess or the result of a previous iteration, the rate term is approximated at the quadrature points by Eq. (6.13). Interpolations are required for all but the interface nodes, i.e. $\tilde{y}(x_{\ell k}) = \sum_{i=0}^{n+1} \ell_i(\xi_\ell) y_{ik}$. Substitution of the linearized rate expression gives:

$$\begin{aligned}
\Delta x_k \int_0^1 \ell_j r_k d\xi &\approx \Delta x_k \sum_{\ell=0}^{m+1} W_\ell \ell_j(\xi_\ell)\left[r^0_{\ell k} + r'_{\ell k}\tilde{y}(x_{\ell k})\right] \\
&= \Delta x_k \sum_{\ell=0}^{m+1} W_\ell \ell_j(\xi_\ell) r^0_{\ell k} - \Delta x_k \sum_{i=0}^{n+1} y_{ik} \sum_{\ell=0}^{m+1} W_\ell \ell_j(\xi_\ell)\ell_i(\xi_\ell) r'_{\ell k} \\
&= \Delta x_k G_{jk} - \Delta x_k \sum_{i=0}^{n+1} M_{jik} y_{ik}
\end{aligned} \tag{6.25}$$

Rounding errors are usually reduced by solving for the change over an iteration, but this derivation is simpler. For the linear rate term, $r = \varphi^2(1 - \tilde{y})$, $\boldsymbol{M}$ is the same for all elements. For this case, analytical integrals are derived in section 2.7:

$$
\begin{aligned}
M_{ji} &= \varphi^2 \int_0^1 \ell_j \ell_i d\xi = \varphi^2 \widehat{M}_{ji} \\
&= \varphi^2 \frac{(-1)^{(i+j+1)}}{2n+3} \sqrt{W_i W_j} \quad \text{for } i \neq j \\
&= \varphi^2 \frac{2(n+1)}{2n+3} W_i \quad \text{for } i = j \\
&\approx \delta_{ji} \varphi^2 W_i
\end{aligned}
\tag{6.26}
$$

Where the last equation is the approximation with $m = n$.

Interpolation of the rate into the trial space was used in Chapter 3 for the variable coefficient problem, Eq. (3.32). With this approach, $r_k \approx \sum_{i=0}^{n+1} \ell_i r(y_{ik})$, so:

$$
\Delta x_k \int_0^1 \ell_j r_k d\xi \approx \Delta x_k \sum_{i=0}^{n+1} r(y_{ik}) \int_0^1 \ell_j \ell_i d\xi = \Delta x_k \sum_{i=0}^{n+1} r(y_{ik}) \, \widehat{M}_{ji} \tag{6.27}
$$

Where $\widehat{\boldsymbol{M}}$ is the expression given in Eq. (6.26) for a linear rate function.

This approach eliminates the interpolations but produces a nonsymmetric mass matrix. Regardless how the rate term is approximated, the method produces an algebraic problem with $(n+1)n_e+1$ equations and unknowns. The matrix structure is identical to that for collocation at Gauss points or moments when implemented with C^0 trial functions, Fig. 6.4.

The case with linear functions, $n = 0$, is of special interest. The integrals are calculated with the trapezoidal rule. In this case a residual cannot even be calculated, but the method produces the following consistent approximation:

$$
\begin{aligned}
&\frac{y_k - y_{k-1}}{\Delta x_k} + \frac{y_k - y_{k+1}}{\Delta x_{k+1}} - \tfrac{1}{2}(\Delta x_k + \Delta x_{k+1}) r_k = 0 \ \text{ for } k = 1, \ldots, n_e - 1 \text{ and} \\
&2Bi_0 y_0 + \frac{y_0 - y_1}{\Delta x_1} - \frac{\Delta x_1}{2} r_0 = 0 \\
&2Bi_1 y_{n_e} + \frac{y_{n_e} - y_{n_e-1}}{\Delta x_{n_e}} - \frac{\Delta x_{n_e}}{2} r_{n_e} = 0
\end{aligned}
\tag{6.28}
$$

This approximation is finite differences derived from a MWR. When finite differences are based on Taylor series it is often unclear how to accurately account for a variable grid or variable coefficients. Also, this second order approximation for the boundary conditions is not obvious from Taylor series analysis. Since this approximation is based on the Galerkin method, it is consistent and accurate and is the best way to derive difference approximations. Interestingly, in the first article describing a finite element method, Courant (1943) called it a *generalized finite difference method*.

If the rate term for the $n = 0$ case is integrated more accurately rather than with the trapezoidal rule, a different approximation is obtained:

$$\frac{y_k - y_{k-1}}{\Delta x_k} + \frac{y_k - y_{k+1}}{\Delta x_{k+1}} - \frac{\Delta x_k}{6}\left(r_k + 2r_{k-\frac{1}{2}}\right) - \frac{\Delta x_{k+1}}{6}\left(r_k + 2r_{k+\frac{1}{2}}\right) = 0 \quad \text{and}$$
$$2Bi_0 y_0 + \frac{y_0 - y_1}{\Delta x_1} - \frac{\Delta x_1}{6}\left(r_0 + 2r_{\frac{1}{2}}\right) = 0 \tag{6.29}$$
$$2Bi_1 y_{n_e} + \frac{y_{n_e} - y_{n_e-1}}{\Delta x_{n_e}} - \frac{\Delta x_{n_e}}{6}\left(r_{n_e} + 2r_{n_e-\frac{1}{2}}\right) = 0$$

Where $r_{k+\frac{1}{2}} = r((y_k + y_{k+1})/2)$ with Simpson's rule and $r_{k+\frac{1}{2}} = (r(y_k) + r(y_{k+1}))/2$ if the rate term is interpolated, Eq. (6.27). With this approximation, the rate term or "load" is distributed rather than concentrated at the center in the difference approximation, Eq. (6.28).

The quadratic case, n = 1, with Lobatto quadrature or Simpson's rule produces a method which also somewhat finite difference like. The approximation is:

$$\frac{8}{3}\left[\frac{y_j - y_{j-1}}{\Delta x_k} + \frac{y_j - y_{j+1}}{\Delta x_k}\right] - \frac{2\Delta x_k}{3} r_j = 0 \quad \text{at center node } x_{1,k}$$
$$-\frac{y_j - y_{j-2}}{3\Delta x_k} + \frac{8}{3}\left(\frac{y_j - y_{j-1}}{\Delta x_k} + \frac{y_j - y_{j+1}}{\Delta x_{k+1}}\right) - \frac{y_j - y_{j+2}}{3\Delta x_{k+1}} - \frac{\Delta x_k + \Delta x_{k+1}}{6} r_j = 0 \text{ at } x_{2,k} = x_{0,k+1} \tag{6.30}$$
$$2Bi_0 y_0 + \frac{8(y_0 - y_1) - (y_0 - y_2)}{3\Delta x_1} - \frac{\Delta x_1}{6} r_0 = 0 \text{ at } x_{0,1}$$

Where, in order to better reveal the finite difference character of the equations, the dual subscript notation has been dropped (the central node of the approximation is designated by j). The equation at the right boundary is analogous to that shown. The equation at the center node is the central difference approximation. A method of higher order is produced when it is combined with the five-point formula for the interface nodes. The coefficients may look strange, but they give a symmetric matrix problem.

The Galerkin method can be used with C^1 trial functions, but it gives a matrix problem with greater band width than with the method of moments. With the method of moments, the weight functions are local to each element, whereas with the Galerkin method the weight functions are the trial functions and $h_1, h_n, \bar{h}_0$ and $\bar{h}_1$ are linked to functions in adjacent elements. For Hermite cubic functions there are 4 matrix entries per row for moments (Fig. 6.5), while for the Galerkin method there are 6. For multidimensional problems, the differences are more dramatic.

6.4 Quadrature Requirements and Convergence Rates

The MWR integrals in the finite element methods are usually approximated using the quadrature formulas of section 2.4. Irons (1966) was an early proponent for the use of quadrature calculations. To perform element by element integration, the quadrature is used directly with appropriate scaling. To perform numerical integration on the entire grid, the quadrature weights are scaled to the grid and combined at the element interface nodes, which gives:

$$W_{ik} = \Delta x_k W_i \quad \text{for } i = 1, \ldots, n \text{ and}$$
$$W_{0,k+1} = W_{n+1,k} = \Delta x_k W_{n+1} + \Delta x_{k+1} W_0 \tag{6.31}$$

Note that this procedure produces the well-known trapezoidal rule and Simpson's rule for the first two Lobatto cases in Fig. 6.1.

For the global methods we found that collocation at Gauss points approximates the method of moments and collocation at Lobatto points approximates a Galerkin method. For the diffusion problem with linear source and constant coefficients, section 3.1, the collocation methods missed exact integration of moments and Galerkin methods by one degree in each case. Even so, for the global methods the convergence rates are similar to methods with more accurate integration. In sections 6.2 and 6.3 we find the same one degree discrepancy for the finite element collocation formulations. We ask – Is this level of approximation adequate in a finite element context? Several studies address this question for the Galerkin method.

Studies usually concentrate on integration accuracy for the stiffness matrix, since it is most important for convergence. The stiffness matrix always involves polynomials of lower degree than the mass matrix and in multiple dimensions the degree varies with coordinate direction. For simplicity, this discussion considers a general mass matrix, like Eqs. (2.121), (3.19) or (3.27) for global methods or Eqs. (6.15) or (6.26) for the FEM:

$$M_{kj} = \int_0^1 w_k(x)\psi_j(x)dx \tag{6.32}$$

where the w_k are the weight functions and the ψ_j are the trial functions. We say the method employs *full* integration if the quadrature is accurate enough to integrate $\boldsymbol{M}$ exactly. With this definition, the integration may still be approximate since the equation could have nonlinear terms or variable coefficients. We call it *reduced* integration if the integration is not accurate enough to integrate $\boldsymbol{M}$ exactly. Douglas and Dupont (1975) show that the convergence rate is not affected by interpolation of nonlinear coefficients, like Eq. (6.27), so this definition of full integration is sufficient.

Many studies have investigated the question of integration requirements for Galerkin finite element methods [Ciarlet and Raviart (1972), Fix (1972), Strang and Fix (1973), Raviart (1973), Fried (1974), Ciarlet (1978)]. Although there are some side issues related to problem smoothness, these studies show that for most problems the maximum rate of convergence can be achieved with less than full integration. Exact integration of all but the highest degree terms in Eq. (6.32) is sufficient to maintain the maximum convergence rate. For example, cubic elements can produce a 4^{th} order convergence rate. For a Galerkin method, full integration would require an exact integration of all terms through 6^{th} degree, since both terms in Eq. (6.32) are cubic. However, the 4^{th} order convergence rate is maintained when only terms through 5^{th} degree are integrated exactly. 5^{th} degree integration accuracy is achieved with 3 Gauss points or 4 Lobatto points.

Although the theoretical analyses cited above apply only for Galerkin methods, it is obvious that the same rule applies to the method of moments. A moments method with cubic trial functions would use discontinuous linear weight functions, so full integration would require

exact integration through 4th degree. To achieve a 4th order convergence rate only terms through 3rd degree must be integrated exactly, requiring 2 Gauss points or 3 Lobatto points.

These illustrations are for cubics, but for higher order trial functions the quadrature keeps pace with the required integration accuracy. If n is incremented by one to increase the degree of the trial function, the integrand in Eq. (6.32) and the quadrature accuracy both increase by two, so the integration is one degree less than full integration for any n.

C^0 collocation at Lobatto points has the desirable feature that the mass matrix is diagonal or lumped, so explicit time stepping is simplified. The work of Fried and Malcus (1975) lends additional support for this procedure. They found that the mass matrix from C^0 collocation at Lobatto points resulted in no loss of convergence rate. Unfortunately, they did not consider Lobatto quadrature for integration of the stiffness matrix.

The studies of integration requirements are supported by theoretical studies of convergence rates for FEM, including both integrated MWR, i.e. Galerkin and moments method, with full integration and the collocation methods with reduced integration. When discussing convergence rates, we say the rate is *optimal* if it is the same as for interpolation of a function. Linear interpolation is $O(\Delta x^2)$ and each additional degree adds to the exponent. The trial function here are polynomials of degree n+1, so optimal convergence is $O(\Delta x^{n+2})$. Derivatives of the approximate solution usually converge more slowly. An $O(\Delta x^{n+1})$ rate for the first derivative is optimal. Most methods achieve optimal rates of convergence overall. However, certain points in the domain converge at a rate which is faster than optimal. When these points can be determined the points are said to be *superconvergent* [reviewed by Křížek and Neittaanmäki (1998)].

Dupont (1976) summarizes the results for the C^0 Galerkin method and the H^1 Galerkin method or equivalently the method of moments. The convergence and superconvergence rates for the orthogonal collocation methods are the same as for the finite element methods they approximate. The rates for C^1 collocation at Gauss points are the same as for the H^1 Galerkin method or moments, while those for C^0 collocation at Lobatto points are the same as for a C^0 Galerkin method. The overall convergence rate is optimal, e.g. $O(\Delta x^{n+2})$ for L_2 or L_∞ norm. However, they also exhibit some superconvergence properties.

The method of moments (or H^1 Galerkin method) and C^1 collocation at Gauss points converge with $O(\Delta x^{2n})$ for both the solution and its first derivative, i.e. fluxes, at element interface nodes [DeBoor and Swartz (1973), Douglas and Dupont (1973), Dupont (1976)]. So, at the interface nodes, the solution is superconvergent for $n > 2$ and derivatives are superconvergent for $n \geq 2$, $n = 2$ corresponds to cubic functions.

C^0 orthogonal collocation at Lobatto points converges with $O(\Delta x^{2n+2})$ for both the solution and its first derivative at element interface nodes [Wheeler (1977)], the same as for the Galerkin method [Dupont (1976)]. It should be no surprise that to achieve this rate the first derivative or flux must be calculated by the method described in section 3.1.4, i.e. using Eq. (6.20) evaluated at the element boundary. It has also been shown that for the C^0 method the solution

at the Lobatto points within the element is O(Δx^{n+3}) for both Galerkin [Chen (1979), Bakker (1981)] and collocation [Nakao (1984)] methods. First derivatives converge with O(Δx^{n+2}) at the n+1 Gauss points which lie between the Lobatto points [Chen (1979), Lesaint and Zlámal (1979)]. Fig. 6.6 illustrates the special superconvergent points for a single element with n = 4. In summary, the solution is superconvergent, O(Δx^{n+3}), at the Lobatto points, the derivatives are superconvergent, O(Δx^{n+2}), at the Gauss points and both are superconvergent, O(Δx^{2n+2}), at the end or interface nodes.

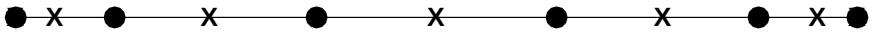


Fig. 6.6 Superconvergent points: • - Lobatto, x - Gauss

These estimates are based on certain assumptions on solution smoothness which will not apply in all cases, but they indicate the potential accuracy of these methods. The convergence and superconvergence results have been demonstrated in several studies [e.g. DeBoor and Swartz (1973), Carey, *et al.* (1981)].

These convergence rates are in terms of element size, Δx. In *hp finite element* terminology, this refinement is called *h refinement*. The other alternative is to increase n or the degree of the polynomial trial function, i.e. *p refinement*. Maday and Patera (1989) analyzed C^0 collocation at Lobatto points and confirmed exponential convergence is achieved. Actually, an exponential convergence rate is implied by the O(Δx^{n+2}), since for increasing n, the rate is not proportional to any fixed power of Δx. The superconvergence properties of this method have been largely overlooked in the spectral literature. However, Zhang (2005) analyses the method for both *h* and *p* refinement, producing results consistent with those of Nakao (1984) and others cited above. Like most articles in the spectral literature, these do not cite earlier work outside the spectral literature.

6.5 Diffusion/Conduction with Source

As an example, the problem described by Eq. (3.1) in Chapter 3 is solved by the finite element approximations described in sections 6.2 and 6.3. We treat the nonlinear, symmetric problem of Section 3.1.5, Eq. (3.35). The trial functions are either the C^0 functions illustrated in Fig. 6.1 or, for moments or collocation at Gauss points, the C^1 functions illustrated in Fig. 6.2. In either case, the trial functions do not take any special advantage of the symmetry. For the symmetric problem, the domain is half as large, but the approximations still apply by setting Bi_0 = 0 to account for the symmetry and by substitution of φ^2 for $4\varphi^2$ and Bi_1 for $2Bi_1$ to account for the smaller domain. Since these methods can converge quite fast, many of the calculations were performed in quad (128 bit) precision so the convergence trend could be observed without the influence of rounding errors.

From the preceding sections of this chapter, it is obvious the extension from global to finite element approximations is conceptually simple. However, implementation of FEM is considerably more complicated. One must keep track of internal nodes and interface nodes and where they belong in the overall grid and the final matrix problem. This bookkeeping

problem becomes even more complicated in multiple dimensions. Usually, the approximations are first calculated on an element-by-element basis. Once these local element approximations are determined, they are loaded into an overall matrix problem, a procedure called *element assembly*. The easiest way to visualize this process is to work through a simple example with a spreadsheet (one is supplied). The spreadsheet also provides an excellent way to check a computer language implementation.

The supplied implementations in computer code utilize readily available band matrix solvers for the most part. However, some use purpose-built solvers. Band solvers store the matrix in a rectangular two-dimensional array. The shape is either the number of bands by the number of unknowns or vice versa. Whether bands are stored as rows or columns is dependent on the language and implementation. The storage scheme should be documented for the solver. Since the matrix structures in Figs. 6.4 and 6.5 are not truly band matrices, zeros must be used to fill in. This approach is not the most efficient since these zeros will be eliminated by the solver unnecessarily. A purpose-built solver would be more efficient, but the band solver is adequate for our purposes.

Since the grid can be refined by increasing either n (*p* refinement) or n_e (h refinement), it is important to understand how the calculations are affected by the refinement method. For purposes of comparison, a multiply or divide is called an *operation*. Additions and subtractions are not counted since they tend to be coupled with multiplies and are often completed in the same machine cycle. Only calculations required for each iteration are counted. To solve the problem as a band matrix, the number of these operations normalized by the number of equations is approximately $(3n_b^2 + 4n_b + 1)/8$, where n_b is the band width. For our problem, the band width is $2n + 3$ with C^0 functions and $2n + 1$ with C^1 functions. For the C^0 case, the operations per node is $(3n^2 + 11n + 10)/2$ using a band solver. If a purpose-built solver is constructed for the C^0 trial functions illustrated in Fig. 6.4, the operations number roughly $(n^2 + 8n + 15)/3$ per node. A purpose-built solver for the C^1 matrix problem of Fig. 6.5 requires $(2n^2 + 9n + 19)/6$ calculations per unknown. These calculations could be reduced for the symmetric C^0 matrix problems, but this complication is not considered.

Although the calculations for all are O(n^2), the proportionality constants or leading coefficients differ and there are significant differences for small to intermediate values of n. For moments or collocation at Gauss points, either C^0 or C^1 functions can be used. The constants are the same for the two approaches using purpose-built solvers, so there is no difference for large n. However, the use of C^1 functions reduces the operations by more than a factor of 2 for $n < 4$ and 1½ for $n < 8$. The constant is 4½ times greater for the band solver, but the difference is only about a factor of 2 for cubics. Often only the order of the calculations is listed, but these other factors can make a significant difference.

For the C^0 method, the number of nodes or unknowns is $(n+1)n_e+1$, while for the C^1 methods there are $(n)n_e$; however, the formulas above show the computations per unknown are similar. The effort is comparable when n for the C^1 method is one greater than for the C^0 method. e.g.

C^1 cubics require slightly less effort than C^0 quadratics. In either case if the grid is refined by adding elements (h refinement), the calculations increase linearly; whereas if it is refined by increasing n (p refinement), the calculations increase with n^3. The exponential convergence rates achieved with *p* refinement are often used to justify its use. With numerical methods, one seldom gets something for nothing. The computational cost of the two alternatives is significantly different. It is even greater in multidimensional problems. One must keep this difference in mind when evaluating the different modes of refinement.

For a full C^0 Galerkin method with $m > n$, the calculation of $\boldsymbol{M}$ and $\boldsymbol{G}$ in Eq. (6.25) is far from insignificant. If the symmetry of $\boldsymbol{M}$ is taken into account, these calculations require roughly $m(n+2)(n+5)$ operations per element, where $m > n$. These calculations are O(n^2) whereas the matrix solution requires O($n^2/3$). The calculations are more than twice the matrix solve when n and m are small. There is a large step change in calculation effort when m is increased from n to $n + 1$.

A method of moments or collocation at Gauss points with C^1 trial functions requires the calculation of $\boldsymbol{M}$ and $\boldsymbol{G}$ in Eq. (6.14). The extra computations are not as significant because the trial functions conform to the quadrature points to some extent. Calculation of these quantities requires roughly $4n$ and $8n$ per element for collocation and the method of moments, respectively. Moments also requires an additional rate calculation at the interface nodes. These calculations are relatively small. One could consider a similar strategy for the full Galerkin method with $m > n$, i.e. select trial function nodes which are a subset of the quadrature points.

The C^0 Galerkin or collocation approximation with the Lagrange functions of Fig. 6.1 is given by Eqs. (6.20), (6.21) and (6.23). These same equations apply for moments or collocation at Gauss points, since Eq. (6.23) reduces to the side condition, Eq. (6.5). The convergence and superconvergence rates have been tested in several ways. First, a grid construction scheme was devised so that grids with different refinement levels could have a point in common. Superconvergence at a given point was tested by observing the convergence at that point in the domain. This refinement scheme is strictly for the purpose of testing convergence rates. In practice one would selectively refine the grid where needed, for example see Carey and Finlayson (1975).

A fine grid with large n and n_e is used to approximate the exact solution. Various norms are calculated. The L_2 norm is calculated by interpolating the solution onto the finer grid and then integrating the square of the difference on the fine grid. The L_∞ or maximum norm is also calculated. Some approximate norms are also calculated. For the C^0 Galerkin method the error at the Lobatto points is measured by the norm:

$$L_2^{LGL} = \sqrt{\sum_{i,k} W_{ik}(y_{ik} - \tilde{y}_{ik})^2} \tag{6.33}$$

where $\boldsymbol{W}$ are the overall Lobatto weights from Eq. (6.31), y and $\tilde{y}$ are exact (fine grid) and approximate solutions. A similar norm at the Gauss points is designated with superscript LG.

These approximate norms indicate the convergence rates at Gauss and Lobatto points within the approximating grid. These points are illustrated in Fig. 6.6. For example, if the C^0 method converges with an overall rate of O(Δx^{n+2}), but with O(Δx^{n+3}) at Lobatto points, then the L_2 norm will show the normal rate, while the L_2^{LGL} norm will show the higher rate. Many interpolations are required to calculate these norms. Some of the codes provided with the project are dominated by interpolations and norm calculations.

The various FEMs have been tested for different n, n_e, Bi and various rate expressions. As a representative problem a nonlinear one from section 3.1.5 is considered. It uses the rate expression of Eq. (3.37) with $k = 1$, $K_a = 0.5$, and $\varphi^* = 5$. Profiles for spherical geometry are shown in Fig. 3.24, however, here we consider planar geometry. First, a simple grid of three C^0 quadratic elements is used with $\Delta x = 0.2, 0.4, 0.4$. With quadratics, collocation is applied at the one interior Lobatto point and integration is by Simpson's rule. Fig 6.7 shows the profiles of $\tilde{y}$, while Fig. 6.8 shows profiles of $d\tilde{y}/dx$. As explained in section 3.1.5, the boundary derivative approaches an asymptotic value of $-\varphi^2/\varphi^* = -7.726$ for this problem. The derivatives are normalized by this factor. This grid is too coarse to resolve the profile with three simple quadratic functions. Problems are apparent in the element nearest the boundary where changes in the solution are most severe. As expected, the largest errors occur for the derivative, including the important boundary value which governs flux to the surroundings.

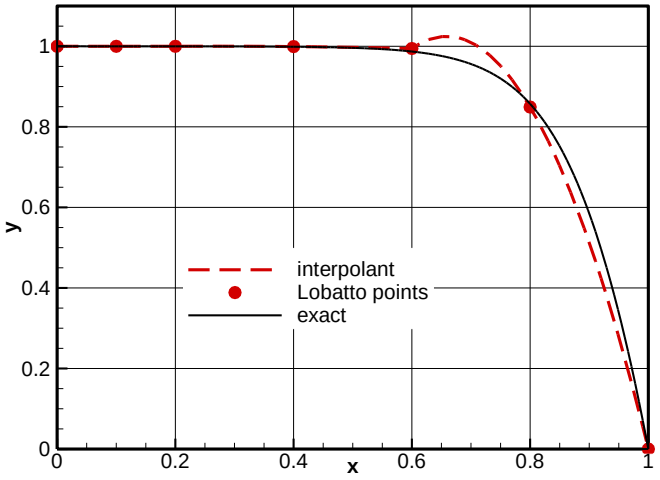


Fig 6.7 profiles 3 quadratic elements, k = 1, K_a = 0.5, φ* = 5

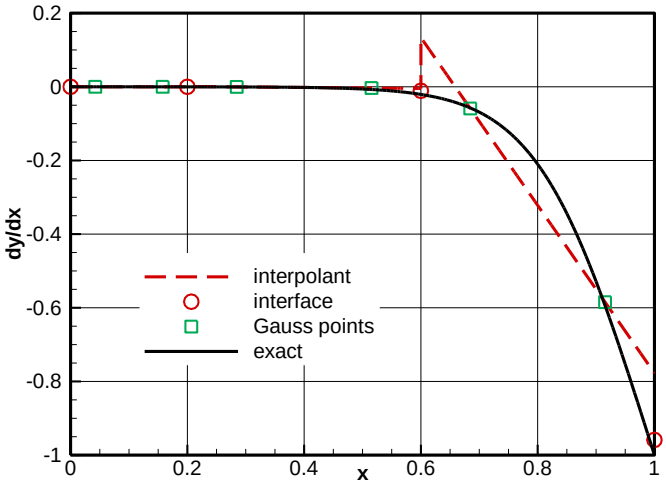


Fig. 6.8 derivative 3 quadratic elements, k = 1, K_a = 0.5, φ* = 5

For quadratics, since $n = 1$, the method should exhibit an optimal overall convergence rate O(Δx^3) for the solution and O(Δx^2) for the derivative. However, the solution should convergence with O(Δx^4) at both the interior and interface nodes. The derivatives should converge with O(Δx^4) at the interface nodes and with O(Δx^3) at the two interior Gauss points. The results shown in the figures seem to indicate greater accuracy at these special points relative to the interpolated solution. To confirm these convergence rates, the change of the error with grid refinement must be observed.

Figs. 6.9 and 6.10 display various measures of the errors in $\tilde{y}$ and $d\tilde{y}/dx$, respectively. The calculations start with the grid used in Fig. 6.7, then it is refined, but all grids contain an interface node $x = 0.60$. As expected, the solution converges with third order, but at the

interface nodes and Lobatto points the convergence is fourth order. The overall convergence rate for the derivatives is second order, but convergence is fourth order at interface nodes and third order at the Gauss points. These results are all consistent with analysis theory.

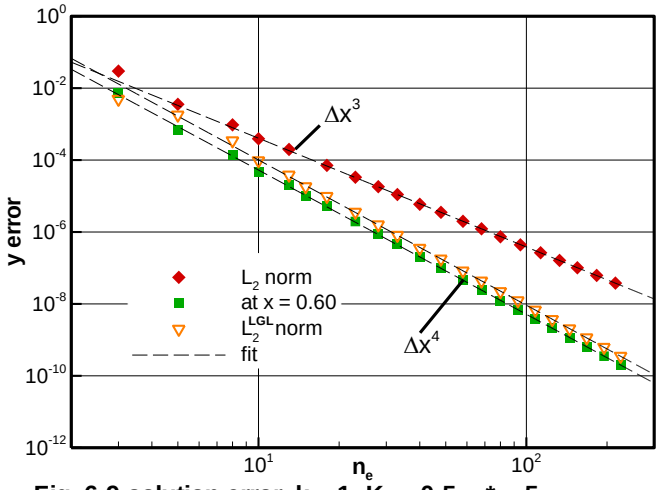


Fig. 6.9 solution error, k = 1, K_a = 0.5, φ* = 5

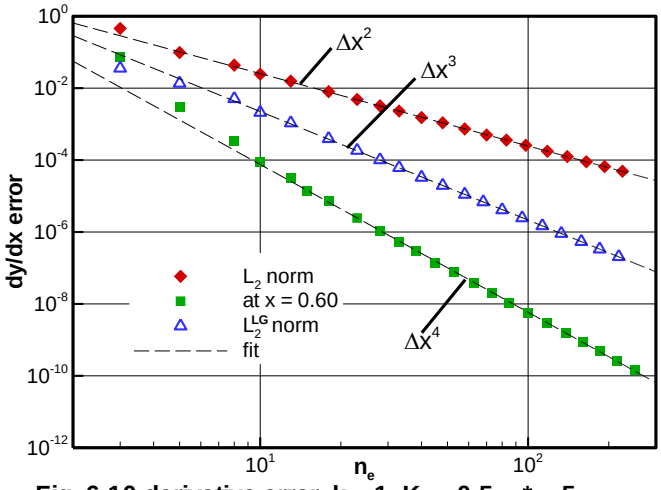


Fig. 6.10 derivative error, k = 1, K_a = 0.5, φ* = 5

Solutions which are more accurate globally can be constructed from the superconvergent results. For example, one could use the solution and derivatives at interface nodes together with higher order Hermite interpolation [Hildebrand (1987), p. 282]. This approach would require interpolation over several elements, so it would work poorly when the grid is relatively coarse. A procedure which is local in nature would be more desirable. The C^1 interpolating functions of Eq. (6.3), described in section 2.8 and illustrated in Fig. 6.2, are a good candidate. These functions can be used to interpolate through the solution at the nodes and the derivatives at the interface. Since all of these points are superconvergent, the interpolant will be an $n+3$ degree or quartic C^1 polynomial with improved accuracy at all points. Figs. 6.11 and 6.12 show the results when this approach is applied to those in Figs. 6.7 and 6.8. This *enhanced* interpolation procedure produces a wonderful improvement for this example.

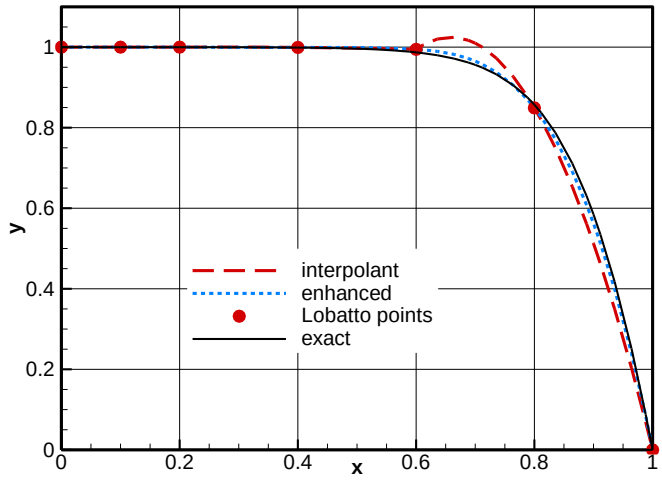


Fig 6.11 profiles 3 quadratic elements, k = 1, K_a = 0.5, φ* = 5

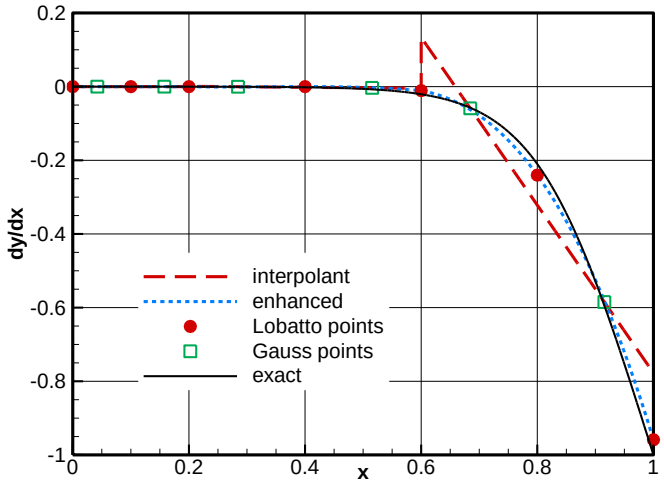


Fig. 6.12 derivative 3 quadratic elements, k = 1, K_a = 0.5, φ* = 5

Figs. 6.13 and 6.14 show that the enhanced interpolation increases the overall rate of convergence by one order for both the solution and the derivative. However, Fig. 6.14 also shows the unexpected result that the derivatives at Lobatto points are two orders more

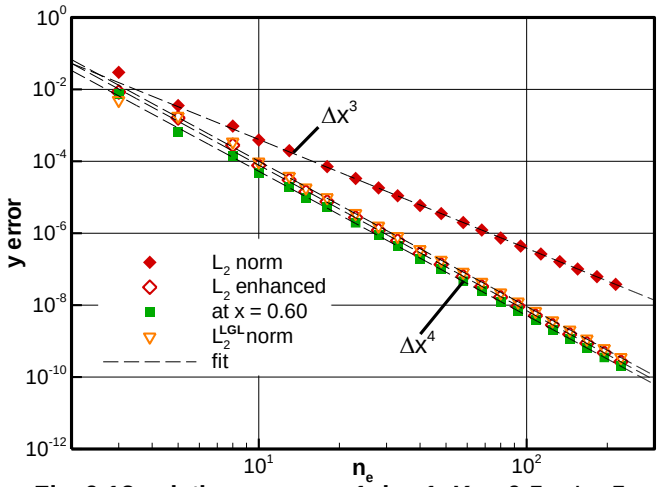


Fig. 6.13 solution error, n = 1, k = 1, K_a = 0.5, φ^* = 5

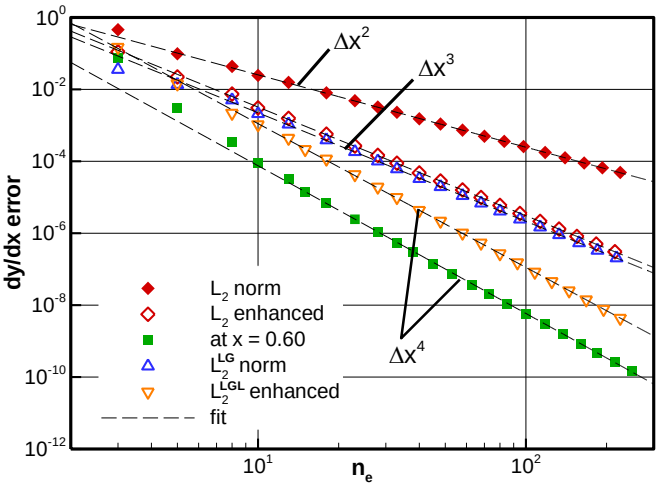


Fig. 6.14 derivative error, n = 1, k = 1, K_a = 0.5, φ^* = 5

accurate with enhanced interpolation. This quadratic method with fourth order convergence rate is quite remarkable. Although it is likely, someone has thought of this interpolation procedure, results with it have not been reported previously to the authors' knowledge. The enhanced interpolation procedure could be used with global approximations and would improve those results.

For $n = 0$, linear trial functions or finite differences, there are no interior Lobatto points, only interface nodes and the interface nodes are not superconvergent. As a result, the enhanced, Hermite cubic, interpolation does not improve the convergence rate. The $O(\Delta x^{n+3})$ observed for quadratic and higher order methods is reduced by one order to second order. For $n = 0$, the derivatives at the interface nodes and from enhanced interpolation are second order and are superconvergent, since first order is optimal. Also for $n = 0$, the single Gauss point is at the center and the derivative there is superconvergent, since it is the well-known second order central difference approximation.

Figs. 6.15 and 6.16 show profiles for the finite difference case with n_e = 8. Although the accuracy at the nodes appears quite good, the error with linear interpolation is significant between nodes. The trial functions represent the derivatives as piecewise constants, so they

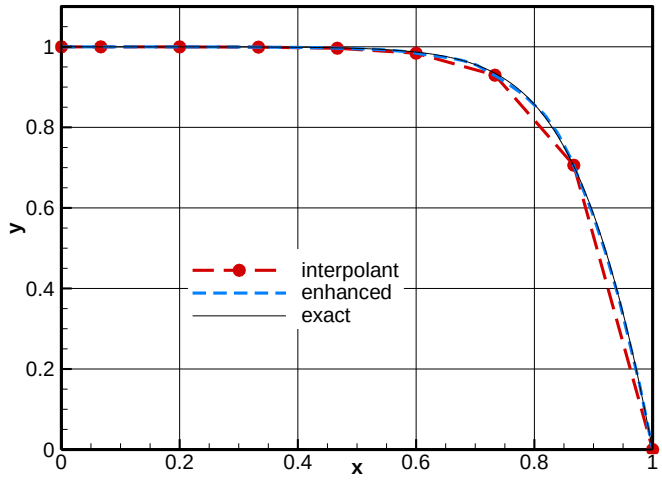


Fig 6.15 finite difference profiles, k = 1, K_a = 0.5, φ^* = 5

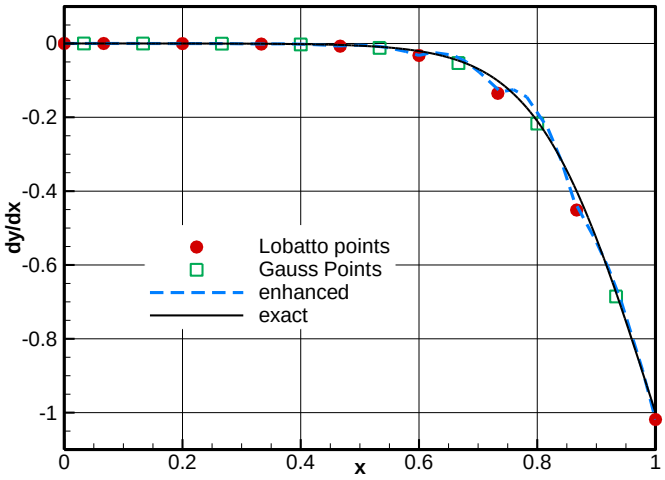


Fig 6.16 finite difference derivative, k = 1, K_a = 0.5, φ^* = 5

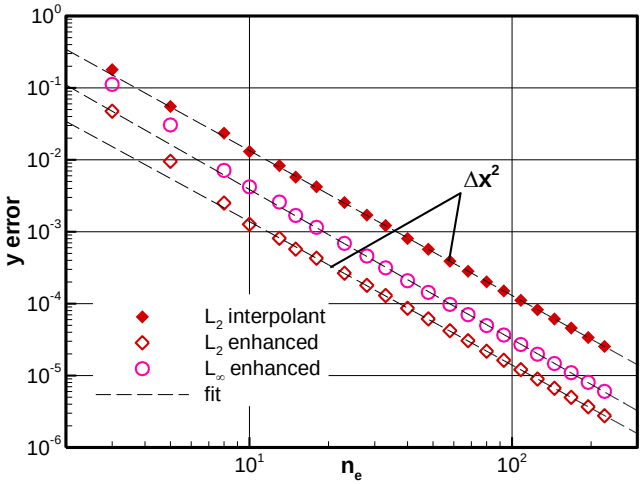


Fig. 6.17 finite difference error, k = 1, K_a = 0.5, φ* = 5

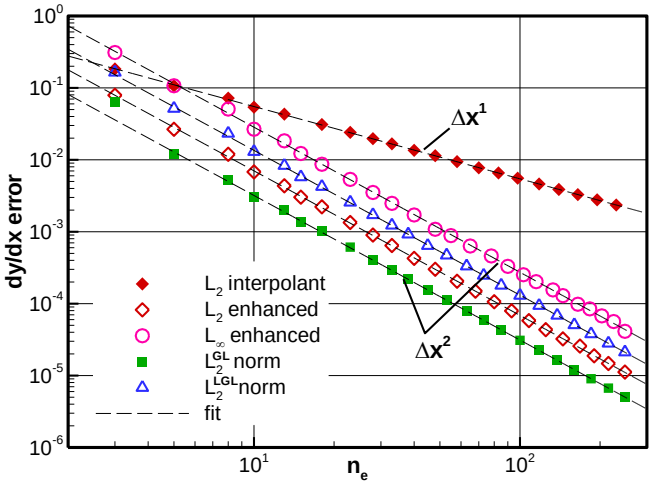


Fig. 6.18 finite difference derivative error, k = 1, K_a = 0.5, φ* = 5

give even greater error. Even though enhanced interpolation does not improve the convergence order for the solution, it improves the profiles considerably compared to direct use of the trial functions. The convergence rates shown in Figs. 6.17 and 6.18 support the observations in Figs. 6.15 and 6.16. The L_2 rates are shown with linear and enhanced interpolation and the maximum, L_∞, error is shown with enhanced interpolation. Here the approximate L_2^{LGL} and L_2^{GL} norms are calculated with the trapezoidal and midpoint rules, respectively. The convergence rates are as indicated above. Although enhanced interpolation does not improve the convergence rate, it nevertheless reduces the error by about an order of magnitude, which is a huge factor – equivalent to using more than 3 times as many nodes. The derivatives at the center or Gauss points and at the nodes are both second order, provided the nodal values are calculated with Eq. (6.20) evaluated at the element interface or:

$$\left.\frac{dy}{dx}\right|_{x_k} \approx \frac{y_{k+1}-y_k}{\Delta x_{k+1}} + \frac{1}{2}\Delta x_{k+1} r_k$$
$$\left.\frac{dy}{dx}\right|_{x_k} \approx \frac{y_k-y_{k-1}}{\Delta x_k} - \frac{1}{2}\Delta x_k r_k \qquad (6.34)$$

Either expression may be used as they are equal by virtue of Eq. (6.28). However, the derivative values at Gauss points are more accurate than those at the nodes. The results suggest an improved enhanced interpolation method could be constructed using the derivatives at the Gauss points.

All the results above use C^0 collocation at Lobatto points, which is equivalent to a reduced Lobatto integration procedure for the Galerkin integrals. Now we ask the question – How much accuracy is gained with an improved approximation of the integrals? Consider the full C^0 Galerkin method from section 6.3, Eq. (6.25) with $m > n$, whereas $m = n$ gives the collocation procedure. The overall convergence rates with $m > n$ are the same with the one exception noted below. Even so, the results for the global calculations in section 3.1.5 suggest relative errors are problem dependent, but reductions can be a factor of four. In some cases little or no

improvement is observed when $m > n + 1$, but there is usually some improvement with one extra quadrature point, i.e. $m = n + 1$. That is the case used here.

Figs. 6.19 and 6.20 compare the maximum error in y or L_∞ error norms and boundary derivative errors for the Galerkin method, with $m = n + 1$, and collocation at Lobatto points, $m = n$. Both methods use the enhanced method to interpolate between nodes. The $n = 0$ case produces finite difference approximations; the collocation approach gives the approximation in Eq. (6.28), while the Galerkin method with Simpson rule integration gives Eq. (6.29). For this problem, the simpler finite difference case produces results which are more accurate by more than a factor of 2.

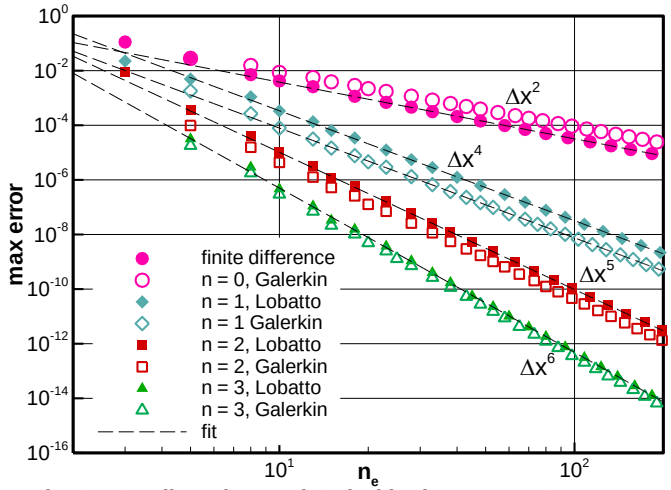


Fig. 6.19 collocation and Galerkin, k = 1, K_a = 0.5, φ^* = 5

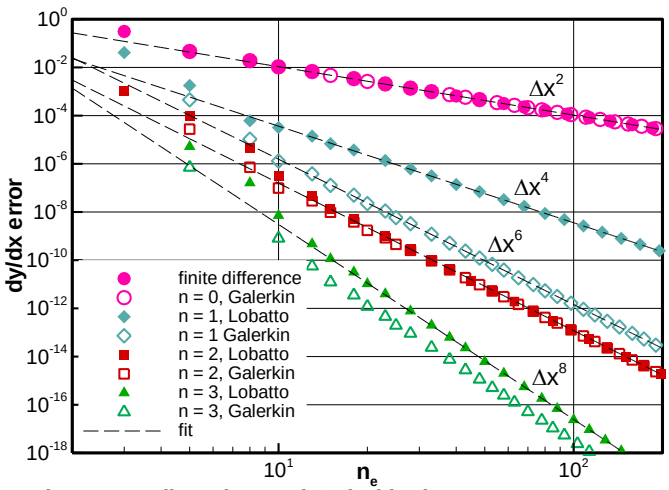


Fig. 6.20 collocation and Galerkin, k = 1, K_a = 0.5, φ^* = 5

The relative accuracy for the quadratic case, $n = 1$, is totally different. The maximum error in y converges at the same rate in both cases, but the error is reduced more than a factor of 4 with the full Galerkin method. Differences for the boundary derivative are more dramatic. The collocation method produces the expected O(Δx^4) superconvergent boundary derivative error, but the full Galerkin method produces an anomalous convergence rate which is better by 2 orders. The rate of O(Δx^6) is quite astonishing since the optimal rate for quadratics is O(Δx^2). This improved accuracy occurs only at the boundary, not at interior interface nodes. We have no explanation for this behavior.

For the higher order cases, the convergence order is the same for both a full Galerkin method and collocation at Lobatto points. The differences are smaller than in the quadratic case, especially for even values of n. In all cases, the small accuracy improvements with a full Galerkin method do not offset the additional calculations required.

Table 6.1 summarizes the results of calculations with different boundary conditions, source expressions and different values of n. The results apply for $n > 0$ with both collocation at Lobatto points and a full C^0 Galerkin method with $m > n$. The convergence rates at the interface nodes, including the important boundary node, are extraordinary for large n.

Next, consider the C^1 methods, both collocation at Gauss points and the method of moments, described in section 6.2. The profiles with cubics, $n = 2$, with 3 elements are very similar to

Table 6.1 Convergence Order C^0 Collocation and Galerkin

Error	Solution		Derivative	
	Normal	Enhanced	Normal	Enhanced
L_2 Norm	$n+2$	$n+3^\dagger$	$n+1$	$n+2$
Interface Nodes	$2n+2$	-	$n+1^*$	$2n+2$
Lobatto Points	$n+3^\dagger$	-	$n+1$	$n+3^\dagger$
Gauss Points	$n+2$	$n+3^\dagger$	$n+2$	$n+2$

$^*n+2$ when n is even, $^\dagger n+2 = 2$ when $n = 0$

those in Figs. 6.11 and 6.12 with C^0 quadratics and enhanced interpolation. As discussed above, the effort using C^1 cubic functions is also similar to that needed for C^0 quadratic collocation at Lobatto points. Since these results are so close to the exact solution, the error profiles are used for comparison in Figs. 6.21 and 6.22. The maximum solution error is about 2% for the collocation methods and 1% for the method of moments. The derivative errors are also more accurate with the method of moments. Here the integrals for the method of moments are approximated with the Lobatto quadrature method, described in section 6.2, which is two degrees more accurate than the integration of collocation at Gauss points. Errors with the method of moments are similar when more accurate integration is used.

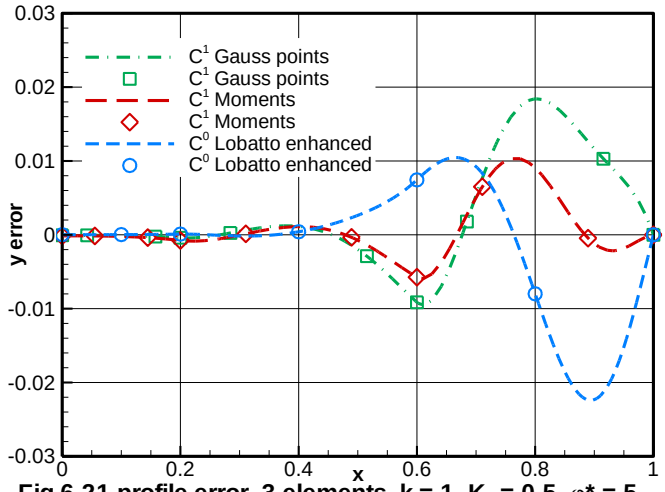


Fig 6.21 profile error, 3 elements, k = 1, K_a = 0.5, φ^* = 5

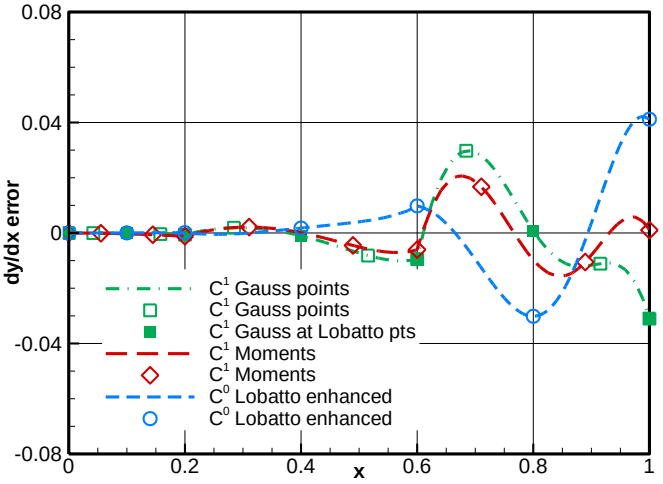


Fig. 6.22 derivative error 3 elements, k = 1, K_a = 0.5, φ^* = 5

Error norms for the solution and derivatives with n = 3, quartic polynomials, are plotted in Figs. 6.23 and 6.24. In agreement with the convergence studies, the overall error is $O(\Delta x^{n+2})$ and $O(\Delta x^{n+1})$ for the first derivative. The error is $O(\Delta x^{2n})$ for the solution and derivative at interface nodes, so the solution is superconvergent for $n > 2$. However, we discovered that at the n - 1 interior Lobatto points, the derivatives are superconvergent $O(\Delta x^{n+2})$. This result appears to be new. These interior derivatives and the interface solution and derivatives could be used to construct a solution which is globally more accurate. This more accurate interpolant would be a C^1 function of degree n + 2. This approach has not been implemented.

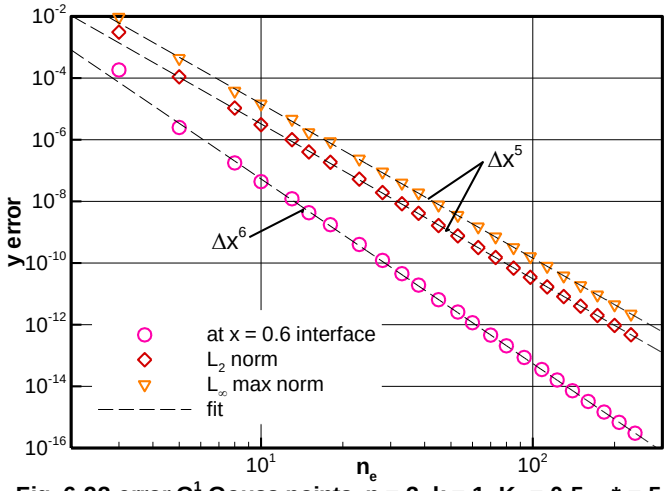


Fig. 6.23 error C^1 Gauss points, n = 3, k = 1, K_a = 0.5, φ* = 5

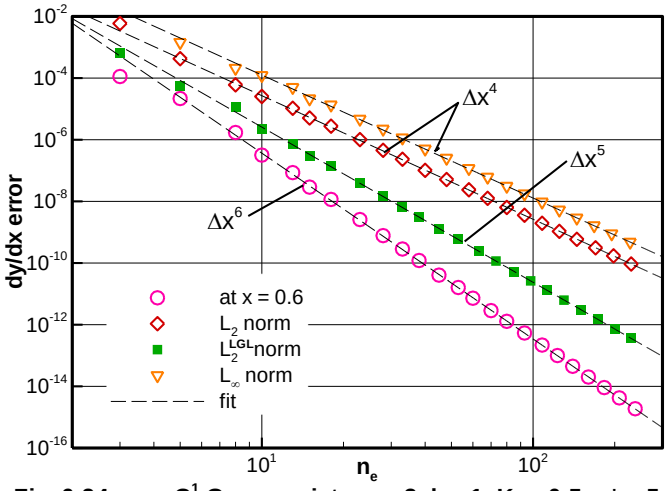


Fig. 6.24 error C^1 Gauss points, n = 3, k = 1, K_a = 0.5, φ* = 5

Collocation at Gauss points is a method of moments with integrals approximated with n point Gaussian quadrature. Now consider the question – How much accuracy is gained with a more accurate approximation of the moment method integrals? Although the results are problem dependent, results for this problem with collocation at Gauss points are compared to those with the method of moments. The integrals in the method of moments were approximated using one extra Gauss point. One extra Gauss point produces the same degree of accuracy and similar results to the Lobatto quadrature method described in section 6.2. For trial functions of various degree, Fig. 6.25 shows the maximum error in the solution, while Fig. 6.26 shows the error in the derivative or flux at x = 1. Errors in the solution are similar with the two methods, while larger differences are evident in the boundary derivative. The order of convergence of the boundary derivative is anomalous for the cubic moments case, i.e. n = 2. The rate should be O(Δx^4) according analytical studies. That rate is observed for internal interface nodes, but at the boundary the rate is two orders higher. A similar anomaly was observed the n = 1 C^0 Galerkin method. We have no explanation for the anomaly. What is striking about these results is the overall accuracy for this relatively difficult problem. Even without selective refinement of the grid an accuracy of 5 or 6 digits is obtained with only a few elements.

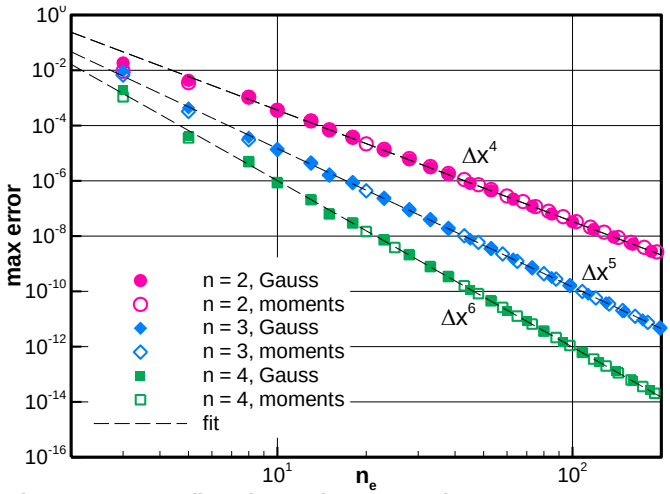


Fig. 6.25 error, collocation and moments, k = 1, K_a = 0.5, φ* = 5

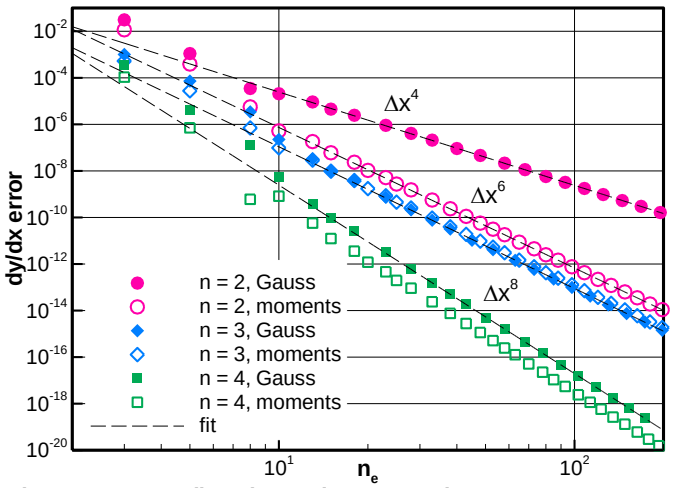


Fig. 6.26 error, collocation and moments, k = 1, K_a = 0.5, φ* = 5

The convergence rates for C^1 collocation at Gauss points and the method of moments are summarized in Table 6.2. The results apply for $n > 1$. For $n = 1$, i.e. quadratic trial functions, the convergence rate for the solution drops by one order, so the convergence order is $n + 1 = 2$ for both the solution and the derivative. Derivatives at interface nodes are superconvergent for $n \geq 2$ and the solution at interface nodes are superconvergent for $n > 2$. Derivatives at the $n - 1$ interior Lobatto points are superconvergent by one order.

Table 6.2 Convergence Order C^1 Collocation or Moments

Error	Solution	Derivative
L_2 Norm	$n + 2$	$n + 1$
Interface Nodes	$2n$	$2n$
Gauss Points	$n + 2$	$n + 1$
Lobatto Points	$n + 2$	$n + 2$

The usual reference given for origination of the spectral element method is an article by Patera (1984). The article describes the finite element method with collocation at Chebyshev points. Most later spectral element papers cite Maday and Patera (1989) who used collocation at Lobatto points as described in section 6.3. The Chebyshev method is easily implemented using the same computer code. It should be implemented with C^0 functions and a natural boundary condition treatment at the interface nodes. The method is unique only for $n > 1$, since for other cases, it is the same as collocation at Lobatto points. Collocation at Chebyshev points achieves optimal convergence rates. Through testing we have found the method achieves convergence rates for both solution and derivative at interface nodes of $n + 2$ for even n and $n + 3$ when n is odd, when the derivatives are calculated using Eq. (6.20). These rates are superconvergent for derivatives and for the solution when n is odd. The convergence rates are always worse than collocation at Lobatto points and worse than collocation at Gauss points for $n > 3$.

Many results are presented above, but the question now arises – Which method is most efficient? The answer is, of course, problem dependent, and your selection depends on your criteria. It also depends on the strategy used for constructing the grid. Although the results are not applicable for all problems, the exercise to compare methods is instructive since it reveals tradeoffs that are universal.

Table 6.3 compares methods on the basis of operations for a given accuracy, where an operation is defined as a multiply or division. It was constructed as follows. First, from the results described above, the number of elements required for a given accuracy is determined. Next, the operations for the matrix solution is calculated on a per element basis using the formulas given near the beginning of this section. Finally, to that number we add $11(n + 1)$ for collocation at Lobatto points and $13n$ for collocation at Gauss points to give the column “cost factor” in the table. Of these last numbers, 8 accounts for calculation of the rate and its

Table 6.3 Relative Cost of Methods

n	Cost factor	Maximum y error			Error dy/dx at x = 1		
		0.01	0.001	0.0001	0.01	0.001	0.0001
Lobatto Points:							
0	16	115	304	928	266	832	2634
1	38	147	309	532	178	439	936
2	68	202	291	449	212	352	631
3	108	211	307	449	236	368	586
Gauss Points:							
2	41	150	317	563	194	481	1017
3	71	209	308	477	235	388	765
4	110	241	353	493	275	427	694

derivative and the remainder are for construction of the matrix problem. The cost factor is an ideal number if all optimizations are utilized. The code developed in the project performs more operations than those listed, since it is designed for flexibility and testing. The cost factor is exceedingly small for the finite difference case, Lobatto n = 0. Note also, the factor for an n point Gauss method is similar to that for an n - 1 point Lobatto method. These methods which require roughly the same effort also converge the solution and derivatives at the same rate, provided enhanced interpolation is used with Lobatto points.

The other columns in Table 6.3 designate different accuracy criteria, either the maximum error in y or error in the boundary flux. The boundary flux was used extensively in the comparisons of Chapters 3 and 4. Each entry gives the number of elements required multiplied by its cost factor. If one is satisfied with a 1% error in the solution, the finite difference method wins, hands down. Enhanced interpolation gives it a substantial boost for a maximum error criteria, otherwise it is not competitive. If a 1% error is required for the boundary flux, the quadratic Lobatto method is most efficient, but the cubic Gauss method is a close second. All methods require similar effort to achieve a 0.1% error in the solution. A 0.1% error in the boundary flux is achieved most efficiently by n = 2 or 3 Lobatto points, followed closely by n = 3 Gauss points. A somewhat loose error criteria favors low order methods, while a tight error tolerance favors higher order methods. The conditions when one is favored over the others depends on the problem and this comparison method is crude. Nevertheless, the trends always follow those found here.

Most of the supplied project codes follow the usual procedure of calculating the approximation for each element and then assembling them into the overall matrix problem. With this procedure, the element calculations are performed on arrays which are O(n) in length. We will call this the *vector length*. Vector length is important in most computations because there is always some overhead to start up a calculation. For example, with interpreted languages, like Matlab or Python, library functions are normally used for array operations and there is overhead for function calls. Efficiency can be gained by operating on all elements at once, since the vector lengths will normally be greater. This approach is tedious if the code is to work

for any value of n. Efficient code is easier to create if each polynomial degree is coded separately. Separate implementations also permit the exploitation any special features of the basic arrays, e.g. zero terms in the first derivative matrix $\boldsymbol{A}$.

The order of the calculations and the way arrays are laid out in memory also affect efficiency. Computers work most efficiently when calculations involve sequential accesses to memory, called *stride-one* calculations, rather than when the calculations involve skipping around in memory. Different languages have different conventions for storing multi-dimensional arrays, either as *column-major-order* or *row-major-order*. The arrays should be defined to give stride-one calculations as much as possible.

Appendices

Appendix A.1 - Accuracy of Weights Calculated by Recurrence

In Chapter 2 several procedures are described for calculating the Jacobi polynomials, their derivatives, and the quadrature and barycentric weights. This section discusses the accuracy and efficiency of those calculations. If the calculations do not produce sufficient accuracy, there are potentially two solutions. Either the calculations can be performed with greater floating-point precision or possibly the calculation procedure can be improved. Since it is most desirable to use accurate calculation procedures, that approach is investigated here using numerical experiments.

The accuracy and efficiency of the weight calculations has been the subject of many studies [e.g. Lether (1978), Yakimiw (1996), Swarztrauber (2003), Hale and Townsend (2013), Bogaert (2014)]. Most studies have considered only Gaussian quadrature, but a few have also addressed Lobatto, Radau or Jacobi-Gauss quadrature. The eigenvalue method of Golub and Welsch (1969), described in Section 2.3.1, is the most common method for calculating roots and weights. The weights can be determined from the first components of the associated eigenvectors. Previous studies have found it more prone to rounding errors with large n. We prefer to use the formulas discussed in Section 2.4, since that approach is compatible with the efficient root finding method discussed in Section 2.3.4. However, since several formulas are available, which is the most accurate and efficient?

Figs. A.1 and A.2 show the maximum relative error for n = 4 - 4000 for barycentric weight calculations using the formulas in Section 2.4.1. The weight errors are shown for Gauss and

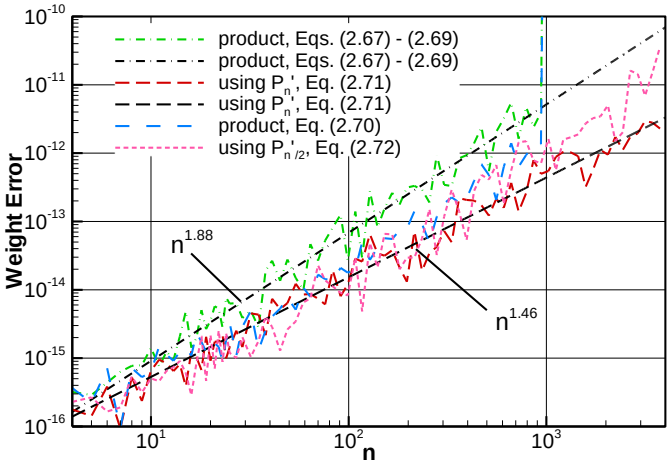


Fig. A.1 Accuracy of formulas for barycentric weights, Gauss points

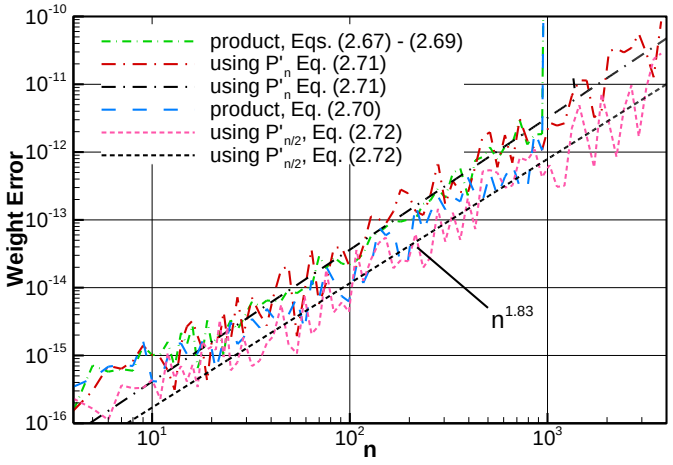


Fig. A.2 Accuracy of formulas for barycentric weights, Lobatto points

Lobatto points from continued products for full, Eqs. (2.67) to (2.69), and shortcut, Eq. (2.70), roots and directly from full and shortcut polynomial derivatives, Eqs. (2.71) and (2.72)., labeled P_n' and $P_{n/2}'$, respectively. The polynomial derivatives were calculated using Eq. (2.35) which reduces to Eq. (2.41) for the Legendre polynomials or Gauss points. The continued product

calculations were afflicted by underflows for n > 1000 even after a few attempts to scale the calculations. The errors shown are the maximum fractional error or the maximum relative error defined by $\max_i(|\epsilon_i/W_i|)$, where ϵ are the errors and W are the "correct" values from 128 bit calculations (quad precision). The error growth rate is indicated by the straight line fit to the results for n > 20. The error growth rate is about n^2 for the product formulas since the point spacing near the boundaries is $1/n^2$. However, in both cases the errors for the shortcut cases are smaller by approximately a factor of four because the square of the polynomial roots is calculated directly using Eq. (2.70). The rate of error growth is similar when the barycentric weights are calculated directly from the derivatives of the polynomial using Eq. (2.71). However, for Gauss points when the derivatives are calculated with Eq. (2.41) the error grows at a slower rate of about $n^{1.5}$.

Figs. A.3 and A.4 show the results of calculations for Gaussian and Lobatto quadrature weights using the formulas in Eqs. (2.78) and (2.81), respectively. The maximum relative error

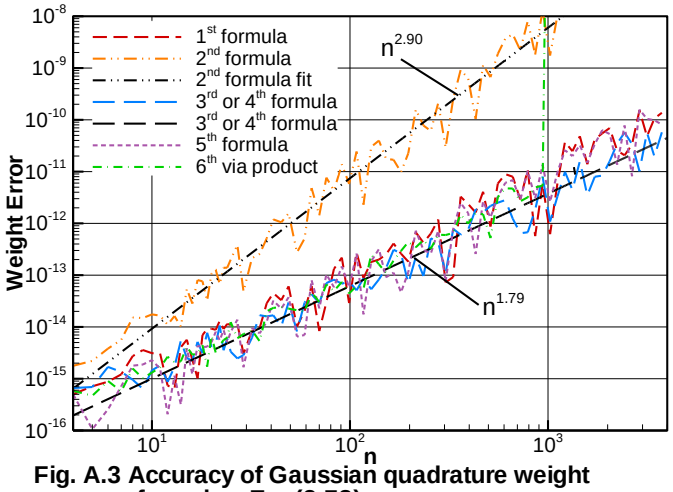


Fig. A.3 Accuracy of Gaussian quadrature weight formulas, Eq. (2.78)

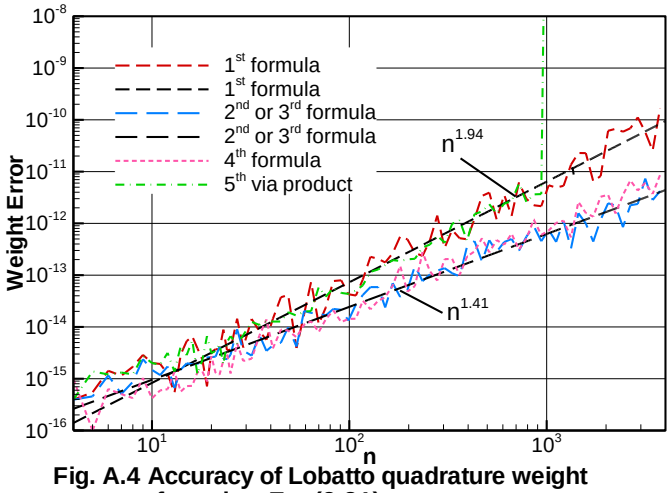


Fig. A.4 Accuracy of Lobatto quadrature weight formulas, Eq. (2.81)

in the Gaussian quadrature weights grow with approximately n^2 in most cases. As many have noted, the second formula is worse than the others. The third and fourth formulas are more accurate than the first by approximately a factor of two on average. For Lobatto quadrature weights, the error grows with approximately $n^{1.4}$ for the second, third and fourth formulas and with n^2 for the others. Since the Lobatto quadrature weights are directly proportional to the square of the barycentric weights, the results in Fig. A.4 suggest more accurate barycentric weights can be calculated by applying the correction in the second formula of Eq. (2.81) or:

$$W_i^b = -\left[(1-x_i^2)P_n^{(1,1)\prime}(x_i) - 2x_iP_n^{(1,1)}(x_i)\right]^{-1} \tag{A.1}$$

Fig. A.5 shows the errors from calculations using Eq.(A.1) compared to the other cases in Fig. A.2. From these results Eq. (A.1) clearly produces a slower error growth and smaller overall errors for n > 100.

The individual quantities in some of the quadrature formulas were calculated to gain a better understanding of the errors shown in Figs. A.1 to A.5. The various formulas; Eqs. (2.71),

(2.72), (2.78), (2.81), etc.; depend on values of the roots, the polynomials and/or their derivatives evaluated at the roots. To obtain accurate weights, we need accurate methods for calculating the individual quantities in the formulas. One issue of importance is that the formulas do not depend explicitly on the roots, but rather on $1 \pm x$. Since the points are clustered near the endpoints, these quantities are proportional to $1/n^2$ near the boundaries. Even if the roots have constant error of say ~10^{-16}, the fractional error in these quantities will increase with n^2 near the boundaries. For example, with n = 400, the first Gauss point is approximately 0.99998, so when $1 - x^2$ is calculated roughly four or five digits of accuracy are lost. Also, there are several methods which can be used to calculate the polynomial derivatives, e.g. Eqs. (2.29), (2.35) and (2.38). Eq. (2.35) requires the fewest calculations, but would one of the other methods be more accurate? Another issue is that Eq. (2.35) gives the product of the derivative and $1 - x^2$. If the derivative is calculated by division with $1 - x^2$, its error will grow with at least n^2 due to the clustering near the end points.

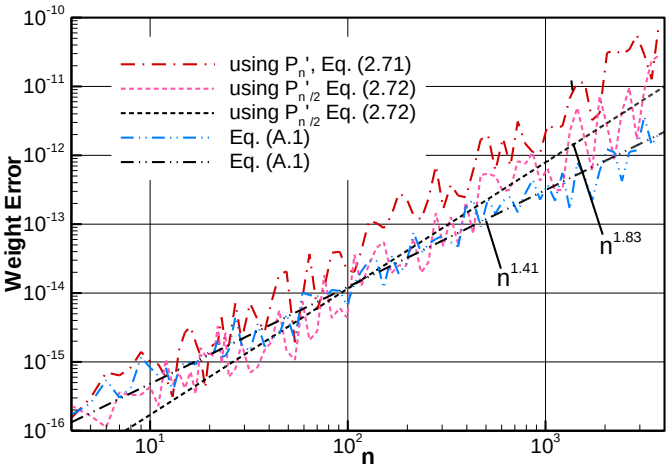


Fig. A.5 Accuracy of improved formula for barycentric weights, Lobatto points

Figs. A.6 and A.7 show some results of the calculations performed to investigate the accuracy of the individual parameters in the formulas for the weights. These calculations were made with roots that were iterated to roundoff conditions with 64 bit (double precision) arithmetic.

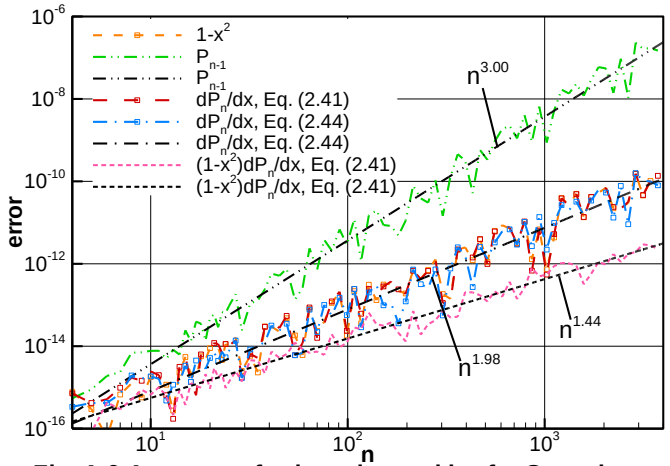


Fig. A.6 Accuracy of selected quantities for Gaussian quadrature weights

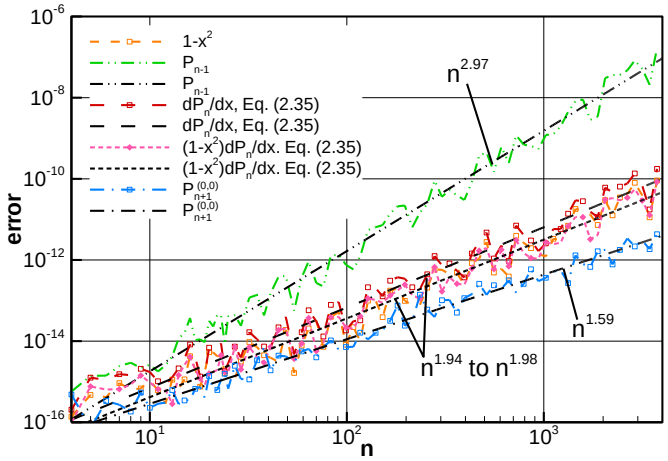


Fig. A.7 Accuracy of selected quantities for Lobatto quadrature weights

Then the accuracy was determined by comparing double precision calculations with quad precision calculations. The quad precision calculations used the same roots to double precision accuracy.

Fig. A.6 shows some of the parameters for the Legendre polynomials in Eq. (2.78) used for calculating the Gaussian quadrature weights. As expected, the maximum relative error in $1 -$

x^2 grows with n^2. For the Legendre polynomials, Eq. (2.35) simplifies to Eq. (2.41) and Eq. (2.38) reduces to Eq. (2.44). Eq. (2.41), $(1-x^2)P_n'(x) = nP_{n-1} - nxP_n$, is of interest since it is the easiest way to calculate the first derivative of the polynomial. It consists of the two terms involving $P_{n\text{-}1}$ and P_n. P_n is zero to within the limits of machine precision, while $P_{n\text{-}1}$ shows a rapid, n^3, rate of error growth. Yet, we find the left-hand side of the equation shows a slower $n^{1.5}$ rate of error growth. Clearly, the P_n term is important and produces greater accuracy even though it is small. Fig. A.3 shows the second formula in Eq. (2.78) is the least accurate, which is consistent with the accuracy of $P_{n\text{-}1}$ in Fig. A.6. The first expression in Eq. (2.78) depends on $\left(\sqrt{1-x^2}P_n'\right)^2$. To calculate the derivative alone with Eq. (2.41) requires division by $1-x^2$, so as expected the error in the derivative grows with n^2. One might hope another method might yield a more accurate calculation of the derivative. Unfortunately, Eqs. (2.29) (results not shown), (2.41) and (2.44) give virtually the same maximum error. Eq. (2.41) is as accurate as any other and requires far less computation.

Fig. A.7 shows calculations relevant for Lobatto quadrature. All but one calculation are with the Jacobi polynomials, $P_n^{(1,1)}$, and its derivatives (superscripts not shown in figure). For these polynomials, Eq. (2.35) reduces to $(1-x^2)P_n^{(1,1)\prime} = (n+1)P_{n-1}^{(1,1)} - nxP_n^{(1,1)}$. Again, the error in $P_{n-1}^{(1,1)}$ grows with n^3. For Lobatto quadrature, Eq. (2.81), the weights depend only on the combination $(1-x^2)P_n^{(1,1)\prime}$ and not the derivatives alone. Unlike the Gauss case, the product is not more accurate than the derivative alone. Although this combination is slightly more accurate, its error grows with n^2 like the error for most of the parameters. Eqs. (2.29), (2.35) and (2.38) produce derivatives of similar accuracy, so there is again no advantage to the longer calculations required by Eqs. (2.29) and (2.38). The error growth for the Legendre polynomials, $P_{n+1}^{(0,0)}$, is somewhat less than the others. This suggests that the fourth formula in Eq. (2.81) will produce greater accuracy, which is observed in Fig. A.4. Calculations like those shown in Figs. A.6 and A.7 have been performed for the other weight formulas and similar results have been found. It appears we can do no better than approximately an $n^{1.5}$ to $n^{1.8}$ rate of error growth for the weights.

Yakimiw (1996) claims that more accurate weights can be calculated by reducing the sensitivity of the weights to small errors in the roots, The sensitivity is reduced by setting $dW(x)/dx = 0$ at the root, where the weight expressions are viewed as continuous functions. Section 2.4.7 describes this approach in greater detail. When the root calculations are iterated to roundoff, the first correction, Eq. (2.97), is claimed to achieve the slowest rate of error growth.

When the parameters for Gaussian quadrature are substituted in Eq. (2.97), the third expression of Eq. (2.78) results. For Lobatto quadrature the second expression of Eq. (2.81) and Eq. (A.1) result. Figs. A.3 and A.4 have shown that these formulas produce error growth rates of $n^{1.4}$ to $n^{1.8}$, which is not substantially better than some of the others. This result would seem to conflict with the contentions of Yakimiw.

Eq. (2.97) is reproduced here:

$$r(x)P'(x) = r(x_0)P'(x_0)\left[1 + (c_{10} + c_{11}x_0)\,\frac{P(x_0)}{(1-x_0^2)P'(x_0)}\right]$$
$$= (1-x_0^2)\,P^{(1,1)\prime}(x_0)\left[1 - 2x_0\,\frac{P^{(1,1)}(x_0)}{(1-x_0^2)P^{(1,1)\prime}(x_0)}\right] \qquad \text{(A.2)}$$

The specific form for Lobatto quadrature is shown in the second equation after substitution of the parameters from Table 2.5. By comparison with Eq. (2.81), the expression can be identified with the second formula, while the first corresponds to neglecting the second term within the brackets. We call the second term a "correction", since with precise calculations, the second term is zero at the root. The accuracy of the overall expression depends on the accuracy of the coefficients and the magnitude of the second term within the brackets. If the equation is applied with quad (128 bit) precision, the weight errors are ~10^{-25} or less, even if the roots have only double precision accuracy, i.e ~10^{-16}. The comparisons in Figs. A.3 and A.4 have already shown that the second term within the brackets is significant relative to unity.

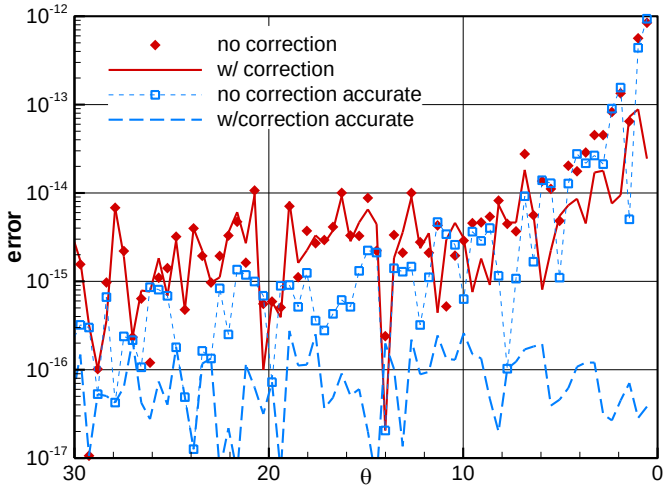


Fig. A.8 Errors in Eq. (2.97) for Lobatto quadrature, n = 400

Fig. A.8 show some results of calculations to investigate the utility of Eq. (2.97). The calculations in the figure are specific to Lobatto quadrature, but similar results apply to the others. The plot is against $\theta = \arccos(x)$ to spread the roots out. Only the points nearest the boundary are shown, $\theta < 30°$ or $x > 0.866$. The figure shows the relative or fractional error in $r(x)P_n'(x)$ for the Jacobi ($\alpha = \beta = 1$) polynomials at Lobatto points. The point nearest the boundary is approximately 0.999954, which accounts for the roughly 4 digit loss of accuracy. The calculation used roots, i.e. x_0, iterated to roundoff conditions in double precision. The results labeled *no correction* are the raw values of $r(x_0)P_n'(x_0)$, while those labeled *w/correction* use the full Eq. (2.97) reproduced above. The first two curves (filled diamonds and solid line) were all calculated in double precision using the recurrence relations to calculate the polynomials and Eq. (2.35) to calculate their derivatives. In this case, Eq. (2.97) gives little improvement. The second pair of results, labeled as *"accurate"*, use the same values of the roots, but quad precision calculations of $r(x_0), P_n(x_0)$ and $P_n'(x_0)$ truncated to double precision and then used in Eq. (2.97). The remainder of the calculation was performed in double precision using double precision values, including the calculation of the correction term. With these three parameters calculated more accurately, we find that without correction the results near the boundary are little better than those calculated in double precision. However, the correction term reduces the error by up to 4 orders of magnitude. From these results, we can conclude that Eq. (2.97) is

useful, but only if the values of the polynomial, its derivative and the $r(x_0)$ term are calculated more accurately.

Yakimiw circumvented this inaccuracy problem by using an alternate Fourier method for calculating the polynomial values and derivatives near the boundary. The Fourier method for polynomial calculation is an O(n^2) process involving transcendental functions, which are computationally intensive.

It appears that our original conclusion is correct. If the recurrence relations are used to calculate the polynomials, the best error growth rate we can hope to achieve for the weights is approximately O($n^{1.5}$).

Appendix A.2 - Asymptotic Jacobi Polynomial Approximations

The problems with accuracy described in the preceding section only occurs for large n. Rather than use an inefficient Fourier method for polynomial calculation, Hale and Townsend (2013) used asymptotic approximations for the polynomials. They address Gaussian and Jacobi-Gauss quadrature strictly for $n > 50$. This approach has the twin advantages of greater accuracy and greater efficiency. If n is large enough the efficiency is greater because there are no calculations of O(n^2) which is the case for using the recurrence relationships or a Fourier method.

As was the case with root estimation, asymptotic approximations for the Jacobi polynomials subdivide into those accurate near the boundary and those accurate away from the boundary. We will again refer to these as *boundary* and *interior* methods.

Szegö, (1975) and Olver, *et al.* (2018) give the following interior asymptotic formula for ultraspherical polynomials:

$$P_n^{(\alpha,\alpha)}(cos\theta) \approx \sum_{k=0}^{M} h_{n,k} \frac{\cos\left(a_{n,k}\right)}{(2\sin\theta)^{k+\alpha+\frac{1}{2}}} \qquad \text{(A.3)}$$

where:

$$a_{n,k} = \left(n + k + \alpha + \tfrac{1}{2}\right)\theta - \left(k + \alpha + \tfrac{1}{2}\right)\tfrac{\pi}{2} \quad \text{and}$$

$$h_{n,k} = \frac{2^{2\alpha+1}\Gamma(n+\alpha+1)\Gamma\left(k+\frac{1}{2}-\alpha\right)\Gamma\left(k+\frac{1}{2}+\alpha\right)}{\sqrt{\pi}\,\Gamma\left(\frac{1}{2}+\alpha\right)\Gamma\left(\frac{1}{2}-\alpha\right)\Gamma\left(n+k+\alpha+1\tfrac{1}{2}\right)k!}, \text{ or}$$

$$h_{n,k} = h_{n,k-1}\frac{\left(k-\frac{1}{2}-\alpha\right)\left(k-\frac{1}{2}+\alpha\right)}{k\left(n+k+\frac{1}{2}+\alpha\right)}, \text{ and}$$

$$h_{n,0} = \frac{2^{2\alpha+1}\Gamma(n+\alpha+1)}{\sqrt{\pi}\;\Gamma\left(n+\alpha+1\tfrac{1}{2}\right)}$$

This relationship is valid for $\frac{1}{6}\pi < \theta < \frac{5}{6}\pi$. It works well even somewhat outside this range but is far better in the interior of the interval and extremely poor for the first few roots nearest the

boundary. There is an extension of this formula for general Jacobi polynomials which requires a double summation.

For Legendre polynomials, Bogaert (2014) developed coefficients for an expansion proposed by Szegö (1934). This boundary approximation is:

$$P_n(cos\theta) \approx s(\theta) \sum_{k=0}^{M} b_k(\theta) J_k\left(\frac{\sigma_n \theta}{2}\right)\left(\frac{2}{\sigma_n}\right)^k \qquad \text{(A.4)}$$

where $\sigma_n = 2n + 1$, the same definition given with Eq. (2.51) and J_k is the Bessell function of the first kind of order k. The expressions for b_k become increasingly complicated, but he worked out the first few using symbolic algebra software:

$$s = \sqrt{\frac{\theta}{sin\theta}}$$
$$b_0 = 1$$
$$b_1 = [\theta \cos\theta - \sin\theta](8\theta \sin\theta)^{-1}$$
$$b_2 = \tfrac{1}{2}[6\theta \cos\theta \sin\theta - 15 \sin^2\theta + \theta^2(9 - \sin^2\theta)](8\theta \sin\theta)^{-2}$$
$$b_3 = \tfrac{5}{2}\{[(\theta^3 + 21\theta) \sin^2\theta + 15\theta^3] \cos\theta - [(3\theta^2 + 63) \sin^2\theta - 27\theta^2] \sin\theta\}(8\theta \sin\theta)^{-3}$$

These asymptotic formulas are incomplete for our purposes, since Eqs. (A.3) is not valid for the Radau case ($\alpha = 1, \beta = 0$) and Eq. (A.4) is valid only for Legendre polynomials or the Gauss case ($\alpha = \beta = 0$). Radau quadrature is relatively less important, but we must address the very important Lobatto case. We also need the derivatives of the polynomials. In the discussion of the equivalence of the different quadrature formulas in Section 2.4, we have seen that all of the polynomials of interest are interrelated through the relationships given in Sections 2.1 and 2.2. The polynomials and derivatives for the Lobatto and Radau cases can be determined from the Legendre polynomials. However, rounding errors must be minimized in the application of these relationships. For large values of n, it makes sense to work with θ coordinates rather than x. The asymptotic relationships are directly dependent on θ and roundoff errors are reduced because the roots are nearly equally spaced in θ rather than clustered about the endpoints.

First, we must work out the derivatives of the asymptotic relationships. Eqs. (2.27) and (2.35) were considered for use with the interior approximation. Each approach requires an additional polynomial calculation, Eq. (2.35) requires the calculation of $P_{n-1}^{(\alpha,\alpha)}$, while Eq. (2.27) utilizes the value of $P_{n-1}^{(\alpha+1,\alpha+1)}$. The use of Eq. (2.27) worked well for most problems, but produced roundoff errors for Radau points. The results with Eq. (2.35) were afflicted by rounding errors for all point types. The best results were found using direct differentiation of the expressions described above.

The derivative of the interior approximation, Eq. (A.3), is:

$$\frac{dP_n^{(\alpha,\alpha)}}{d\theta} \approx -\sum_{k=0}^{M} \frac{h_{n,k}}{(2\sin\theta)^{k+\alpha+\frac{1}{2}}}\left[\left(k + \alpha + \tfrac{1}{2}\right)\cot\theta \cos(a_{n,k}) + \left(n + k + \alpha + \tfrac{1}{2}\right)\sin(a_{n,,k})\right] \qquad \text{(A.5)}$$

The derivative of the boundary approximation, Eq. (A.4), is:

$$\frac{dP_n}{d\theta} \approx \sum_{k=0}^{M} \left[(s' b_k + s\, b_k') J_k + s\, b_k \left(\frac{\sigma_n}{2}\right) J_k' \right] \left(\frac{2}{\sigma_n}\right)^k \tag{A.6}$$

The derivatives of the Bessell functions are straight forward, while the derivatives of the coefficients are:

$$\begin{aligned}
s' &= -4s(\theta) b_1(\theta) \\
b_1' &= 8[\sin^2\theta - \theta^2](8\theta \sin\theta)^{-2} \\
b_2' &= 24[\theta \cos^3\theta - 5\cos^2\theta \sin\theta - (3\theta^2+1)\theta\cos(\theta)\, 15\sin^2\theta \\
&\quad -(\theta^2-5)\sin(\theta)](8\theta\sin\theta)^{-3} \\
b_3' &= 20[\,(3\theta^2+189)\cos^4\theta + 42\theta\cos^3\theta\sin\theta - (29\theta^4 - 42\theta^2 + 378)\cos^2\theta \\
&\quad -(54\theta^2+42)\,\theta\cos\theta\sin\theta - 16\theta^4 - 45\theta^2 + 189](8\theta\sin\theta)^{-4}
\end{aligned}$$

The key to the reduction of rounding errors is to avoid subtracting two numbers of nearly equal magnitude. All coefficients of the expansion are subject to rounding errors. For example, consider the numerator of b_1. A Maclaurin series expansion gives:

$$b_1(8\theta\sin\theta) = \theta\cos\theta - \sin\theta = \left(\theta - \frac{\theta^3}{2!} + \cdots\right) - \left(\theta - \frac{\theta^3}{6!} + \cdots\right) = -\frac{\theta^3}{3} + \cdots$$

The individual terms in the series go to zero with θ but the expression as a whole goes to zero with θ^3. For small values the two terms are of comparable value causing a loss of accuracy. The loss of accuracy can be avoided by using the Maclaurin series expansion for the full expression rather than computing each term and then subtracting. All coefficients b_k in the expansion behave in this manner, but only the lower order coefficients need to be expanded in this way since the higher order terms are less important near $\theta = 0$.

For Lobatto quadrature, the value and derivative of $P_n^{(1,1)}$ are needed. Interior approximations provide these values directly from Eqs. (A.3) and (A.5). For the boundary area however, Eqs. (A.4) and (A.6) are valid only for Legendre polynomials, or $P_n^{(0,0)}$. The value and derivative of $P_n^{(1,1)}$ are easily calculated from the value and derivative of $P_{n+1}^{(0,0)}$.The two polynomials are related through Eq. (2.27), which can be differentiated to give a relationship for the derivative of $P_n^{(1,1)}$ in terms of the second derivative of the Legendre polynomial. Eq. (2.39) relates the second derivative to the value and first derivative. After converting to θ coordinates the resulting relationships are:

$$\begin{aligned}
P_n^{(1,1)} &= \frac{2}{(n+2)\sin\theta}\frac{dP_{n+1}^{(0,0)}}{d\theta} \\
\frac{dP_n^{(1,1)}}{d\theta} &= \frac{2}{\sin\theta}\left[(n+1)P_{n+1}^{(0,0)} - \cos\theta\, P_n^{(1,1)}\right]
\end{aligned} \tag{A.7}$$

For Radau quadrature, the value and derivative of $P_n^{(1,0)}$ or $P_n^{(0,1)}$ can also be expressed in terms of Legendre polynomials through Eqs. (2.83) or (2.88) when it is combined with the recurrence relationship, Eq. (2.25), the Sturm-Liouville relationship, Eq. (2.39), and the

expression for the first derivative, Eq. (2.41). After some algebraic manipulation and conversion to θ coordinates, the following relationships are derived:

$$
\begin{aligned}
P_n^{(1,0)} &= P_n^{(0,0)} - \frac{2\cos^2\left(\frac{\theta}{2}\right)}{(n+1)\sin\theta}\frac{dP_n^{(0,0)}}{d\theta} \\
\frac{dP_n^{(1,0)}}{d\theta} &= \frac{1}{\sin^2\left(\frac{\theta}{2}\right)}\left[\frac{1+n\sin^2\left(\frac{\theta}{2}\right)}{n+1}\frac{dP_n^{(0,0)}}{d\theta} + \frac{n}{2}\sin\theta\, P_n^{(0,0)}\right] \\
P_n^{(0,1)} &= P_n^{(0,0)} + \frac{2\sin^2\left(\frac{\theta}{2}\right)}{(n+1)\sin\theta}\frac{dP_n^{(0,0)}}{d\theta} \\
\frac{dP_n^{(0,1)}}{d\theta} &= \frac{1}{\cos^2\left(\frac{\theta}{2}\right)}\left[\frac{1+n\cos^2\left(\frac{\theta}{2}\right)}{n+1}\frac{dP_n^{(0,0)}}{d\theta} - \frac{n}{2}\sin\theta\, P_n^{(0,0)}\right]
\end{aligned}
\tag{A.8}
$$

Note these formulas are consistent with the symmetry relationship, Eq. (2.10). Eqs. (A.7) and (A.8) permit all the polynomials of interest to be calculated from Legendre or ultraspherical polynomials.

For symmetric problems in planar and spherical geometry, the values and derivatives of the polynomials $P_n^{\left(\alpha,\pm\frac{1}{2}\right)}$ are needed also. These polynomials can be calculated using relationships for the shortcut polynomials. The equations are written in terms of x and by $\xi = 2x^2 - 1$. The angular coordinates are related by the identity $\cos\phi = \cos(2\theta) = 2\cos^2\theta - 1$, where $\xi = \cos\phi$ and $x = \cos\theta$. The formulas for the shortcut polynomials and their derivatives are Eqs. (2.18), (2.21), (2.30) and (2.31). Converting these equations to angular coordinates produces the following relationships:

$$
\begin{aligned}
P_n^{\left(\alpha,-\frac{1}{2}\right)}(\phi) &= P_{2n}^{(\alpha,\alpha)}(\theta)/\tilde{a}_{2n} \\
\frac{dP_n^{\left(\alpha,-\frac{1}{2}\right)}(\phi)}{d\phi} &= \frac{1}{2\,\tilde{a}_{2n}}\frac{dP_{2n}^{(\alpha,\alpha)}(\theta)}{d\theta} \\
P_n^{\left(\alpha,+\frac{1}{2}\right)}(\phi) &= P_{2n+1}^{(\alpha,\alpha)}(\theta)/[\tilde{a}_{2n+1}\cos\theta] \\
\frac{dP_n^{\left(\alpha,+\frac{1}{2}\right)}(\phi)}{d\phi} &= \left(\frac{1}{\tilde{a}_{2n+1}}\frac{dP_{2n+1}^{(\alpha,\alpha)}(\theta)}{d\theta} + \sin\theta\, P_n^{\left(\alpha,+\frac{1}{2}\right)}(\phi)\right)\frac{1}{2\cos\theta}
\end{aligned}
\tag{A.9}
$$

Eqs. (2.18) and (2.21) define the normalizing constants. The values of interest are: $\tilde{a}_{2n} = \tilde{a}_{2n+1} = 1$ for the Gauss case, $\alpha = \beta = 0$, and, $\tilde{a}_{2n} = (2n+1)/(n+1)$ and $\tilde{a}_{2n+1} = 2$ for the Lobatto case, $\alpha = \beta = 1$.

Figs. A.9 and A.10 show the accuracy of the asymptotic relationships for the Jacobi polynomials needed for Lobatto quadrature, i.e. $\alpha = \beta = 1$. The asymptotic calculations all used quad precision (128 bit) calculations with the roots iterated to roundoff conditions. They are compared to the values from the recurrent relationships calculated in double precision, after truncation of the roots. The absolute value of P_n is plotted in Fig. A.9 and the relative error in the derivative is plotted in Fig. A.10. The value of P_n is the error, since zero is the correct value.

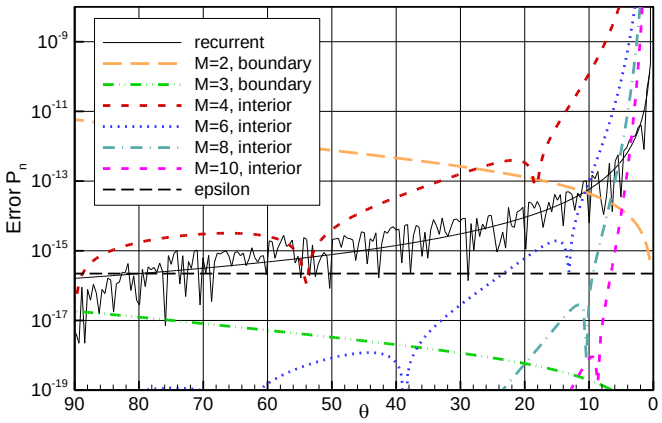


Fig. A.9 Accuracy of P_n asymptotic formulas, $\alpha = \beta = 1$, n = 400

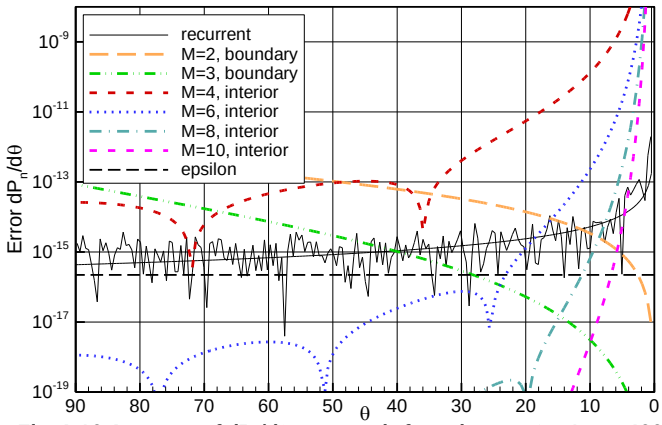


Fig. A.10 Accuracy of $dP_n/d\theta$ asymptotic formulas, $\alpha = \beta = 1$, n = 400

When n = 400, only a few terms are needed in these formulas to surpass the accuracy of calculations in double precision with the recurrent relations. As expected the boundary formula is accurate near the boundary, while the interior formula is more accurate elsewhere.

At the roots of $P_n^{(\alpha,\alpha)}$ the error in the polynomial and the relative error in its derivative using the interior asymptotic relationships, Eqs. (A.3) and (A.5), can be approximated by the equation:

$$\epsilon_{n,M} = \frac{g_0 + g_1 M}{(\theta n^b)^{(M+e)}} \tag{A.10}$$

where the four parameters are listed in Table A.1 for θ in degrees. Figs. A.11 and A.12 show the excellent agreement between the actual errors and those predicted by Eq. (A.10) This equation is useful for determining when to use the interior approximation and how many terms to include in the expansion. The equation is used by solving it for θ and substituting the product of the machine epsilon and a safety factor for $\epsilon_{n,M}$. The resulting value gives the maximum θ for which the interior

Table A.1 Error Parameters, Eq. (A.10)

	b	e	g_0	g_1
$P_n^{(0,0)}$	1.0	1.507	6.945	8.671
$dP_n^{(0,0)}/d\theta$	1.0	0.789	-13.800	10.580
$P_n^{(1,1)}$	0.9	2.527	58.440	2.500
$dP_n^{(1,1)}/d\theta$	1.0	0.900	0.000	9.621

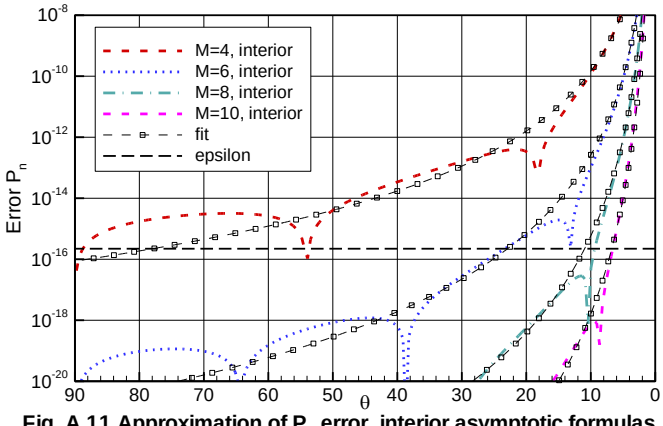


Fig. A.11 Approximation of P_n error, interior asymptotic formulas, $\alpha = \beta = 1$, n = 400

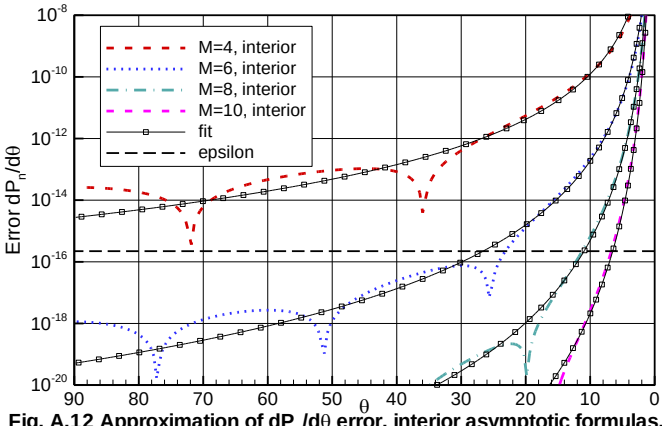


Fig. A.12 Approximation of $dP_n/d\theta$ error, interior asymptotic formulas, $\alpha = \beta = 1$, n = 400

approximation is accurate for a given M. The interior method with M = 3 gives accurate values when $\theta < \theta^* = 2.25 + 0.11n$ (degrees).

For simplicity, the boundary method always uses M = 3 and for the interior method only M = 4, 5, 7, 9 and 12 are considered. The limiting values of θ gives the range where each method is valid. If the valid range for interior and boundary methods do not overlap, recurrent calculations are used in between. Usually, the boundary method provides no improvement over recurrent calculations for $n \lesssim 40$. Recurrent calculations are not needed for n $\gtrsim$ 110.

The composite procedure, just described, can also be tailored to the problem size as follows:

- $\boldsymbol{n \le n_{Asmp}}$**:** the recurrence relationships are used exclusively, with all the calculations in x coordinates
- $\boldsymbol{n_{Asmp} < n \le n_{Intr}}$**:** only recurrent and boundary methods are used.
- $\boldsymbol{n > n_{Intr}}$**:** a combination of recurrent, interior and boundary methods are used.

Generally, we have found that n_{Asmp} = n_{Intr} = 40 works well. Table A.2 shows the number of points calculated with each method for various n. The row designation "M = xx" indicates the number of terms used in the interior method, Eqs. (A.3) and (A.5).

Table A.2 Distribution of Methods Used For Calculating $P_n^{(1,1)}$ and $dP_n^{(1,1)}/d\theta$

n	44	64	80	128	160	320	640	1280	2560	5120
boundary	1	3	4	9	9	9	10	11	12	12
recurrent	9	6	5	0	0	0	0	0	0	0
M = 12	8	9	9	9	9	9	9	9	10	11
M = 9	4	14	21	21	21	21	19	20	21	23
M = 7	0	0	1	25	41	121	139	137	134	131
M = 5	0	0	0	0	0	0	143	358	358	358
M = 4	0	0	0	0	0	0	0	105	745	2025

Several figures have been constructed from calculations using the procedures outlined above. The accuracy of the calculation procedure is illustrated for $P_n^{(1,1)}$ and its derivative evaluated at the Lobatto points. Similar or perhaps slightly better results are achieved for $P_n^{(0,0)}$ at Gauss points. These calculations are exclusively in double precision, but use roots, $\boldsymbol{x}$ and $\boldsymbol{\theta}$, iterated to roundoff conditions in quad precision and then truncated to double precision. The values (or error) of the polynomial and relative error in its derivatives were calculated at the roots.

For an intermediate value, n = 64, Figs. A.13 and A.14 show the errors – absolute error in $P_n^{(1,1)}$ and relative error in $dP_n^{(1,1)}/d\theta$. Results are shown for the recurrence calculations, the highest order asymptotic methods considered and the composite method which used the combination of methods indicated in Table A.2. The boundary method cutoff value from the equation above is θ^* = 9.3°, so the boundary method is used for only the 3 points nearest the boundary. The recurrence method was used for the next 6 points and the remainder of the calculations used the interior method with 12 or 9 terms. For this case, recurrence calculations

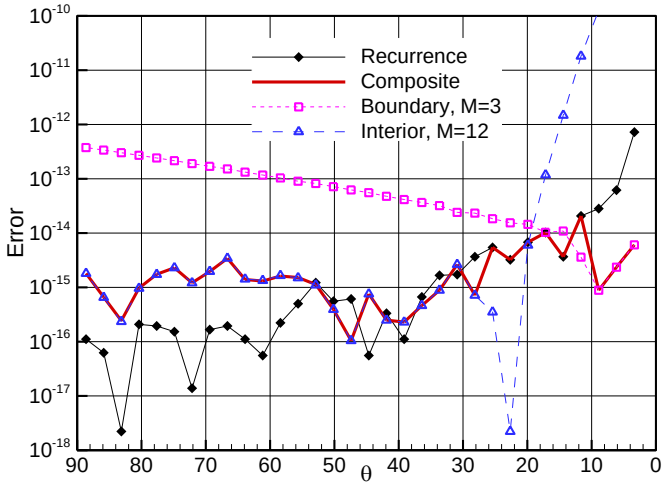


Fig. A.13 Error in P_n at the roots, $\alpha = \beta = 1$, n = 64

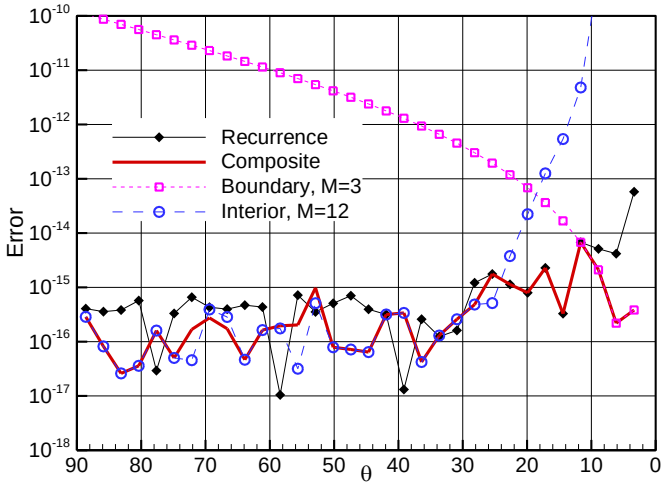


Fig. A.14 Error in $dP_n/d\theta$ at the roots, $\alpha = \beta = 1$, n = 64

would have worked well for all the interior points, but for some of the symmetric cases recurrence calculations produced larger errors near $\theta = 90°$ ($x = 0$). With either method, the error distribution is more uniform, and the maximum error is about an order of magnitude less than with the recurrence relationships alone. Note that for $10° < \theta < 25°$ neither asymptotic approximation gives sufficient accuracy.

Figs. A.15 and A.16 shows error calculations for a larger value, $n = 128$. The distribution of methods used is shown in Table A.2. The composite results used only asymptotic calculations,

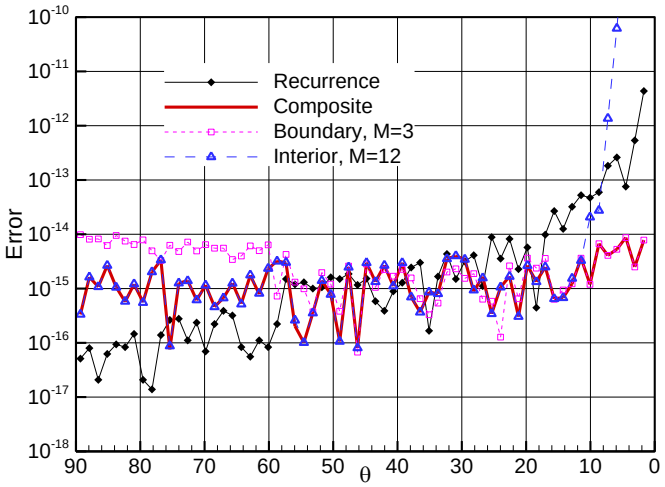


Fig. A.15 Error in P_n at the roots, $\alpha = \beta = 1$, n = 128

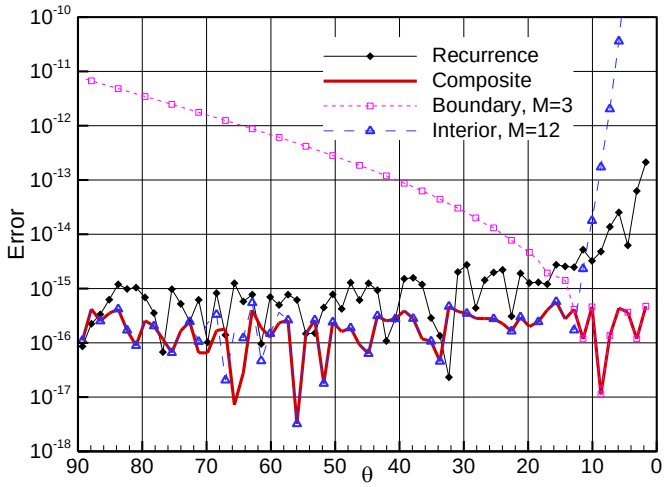


Fig. A.16 Error in $dP_n/d\theta$ at the roots, $\alpha = \beta = 1$, n = 128

but the values from recurrence calculations are included for comparison. The maximum error with the composite calculations is 2 to 3 orders of magnitude less near the boundary and more uniform than the recurrence calculations.

The approach produces consistent accuracy for quite large values of n. Note from Table A.2 that fewer terms are needed for points further from the boundary. The calculations per node decreases, since fewer and fewer terms are needed in the asymptotic expansions. Even smaller values of M could be considered as an additional optimization.

Radau quadrature introduces new opportunities for rounding errors, which others have not discussed. Due to the symmetry of the ultraspherical polynomials, only half the points need to

be considered, i.e. $0 < x < 1$ or $\pi/2 > \theta > 0$. For Radau quadrature, the Jacobi polynomials, $\alpha = 0, \beta = 1$ and $\alpha = 1, \beta = 0$, are asymmetric, so the entire interval must be considered, i.e. $-1 < x < 1$ or $\pi > \theta > 0$. There has been much discussion about the rounding errors introduced by the $O(n^2)$ spacing in x near the endpoints. θ coordinates are the recommended solution for improved accuracy since they give a more uniform spacing of the points. However, the $O(n)$ spacing can still lead to a linear error growth rate unless one is careful. Rounding errors are reduced not only by the spacing of the points, but also because the origin in θ coordinates is at the troublesome endpoint, $x = 1$ or $\theta = 0$.

Fig. A.17 shows the results of calculations with $n = 256$ for the two cases $\alpha = 0, \beta = 1$ and $\alpha = 1, \beta = 0$. The (0,1) case is plotted for 0° to 30° (the right end), while for the (1,0) case the left end from 180° to 150° is plotted in the opposite direction. According to symmetry, Eq. (2.10), the two results should be identical, but the rounding errors are different. The (0,1) polynomial has smaller rounding errors, because the region shown is near the origin. The (1,0) polynomials is more accurate at its origin, i.e. the other end. For larger n, the differences are greater. The cause of the difference is simple. To apply the boundary approximation for $\theta > \pi/2$ the coordinates must be folded, i.e. $\breve{\theta} = \pi - \theta$, utilizing Eq. (2.10) to produce the results. The subtraction introduces a relative error which is proportional to n. The more accurate results in Fig. A.17 are achieved on both ends for both polynomials if the subtraction is carried out in higher precision.

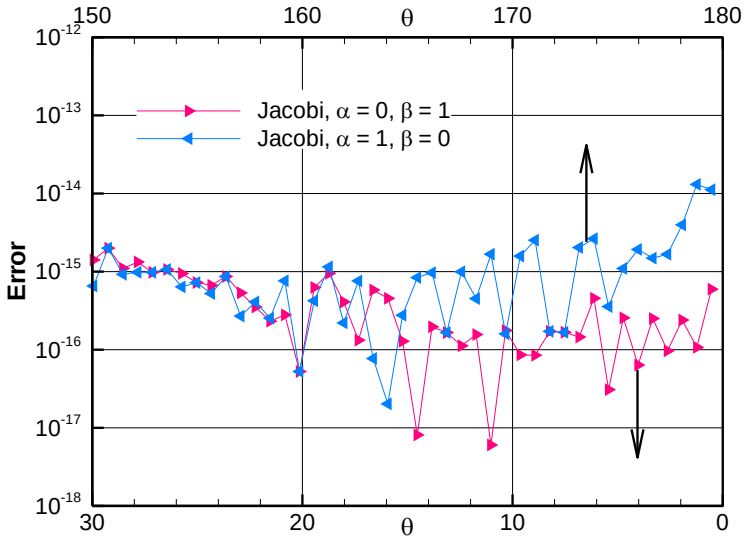


Fig. A.17 Error in $dP_n/d\theta$ for asymmetric polynomials, n = 256

We would like to avoid calculations in higher precision, since it adds complexity and that option is not always available. To achieve higher accuracy for asymmetric polynomials without higher precision, all calculations should be carried out in the folded coordinates, i.e. $\breve{\theta} = \pi - \theta$ for $\theta > \pi/2$. Eq. (2.10) is used to establish the sign. Although it is somewhat cumbersome, this approach has been implemented, so the more accurate results are achieved at both ends. Using the same approach with shortcut polynomials was found to give slightly more accurate roots for ultraspherical polynomials. Eqs. (A.3) and (A.5) give more accurate calculations near $\pi/2$ or $x = 0$ when rewritten in terms of $\pi/2 - \theta$.

Since the results displayed in Fig. A.8 caused us to launch off into the study of asymptotic methods, that problem is revisited here. Figs. A.18 and A.19 show similar results calculated with the composite asymptotic method. The results in Fig. A.18 are like those in Fig. A.8, except, the recurrent calculations are replaced with the composite calculations. The results are not as good as the quad precision calculations, but they are much better than the recurrent

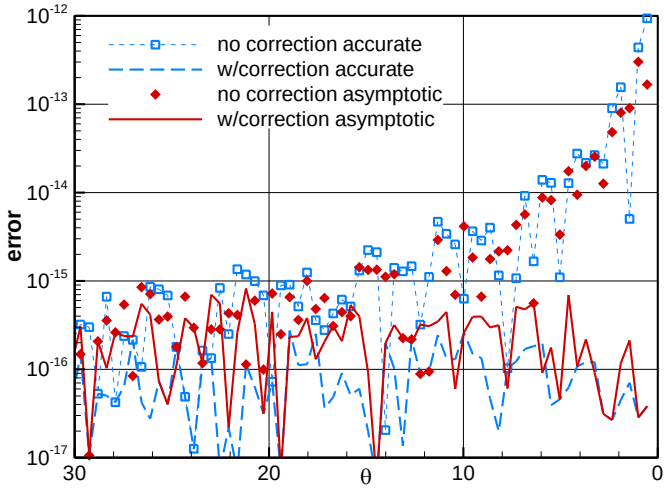


Fig. A.18 Errors in Eq. (2.97) for Lobatto quadrature, n = 400

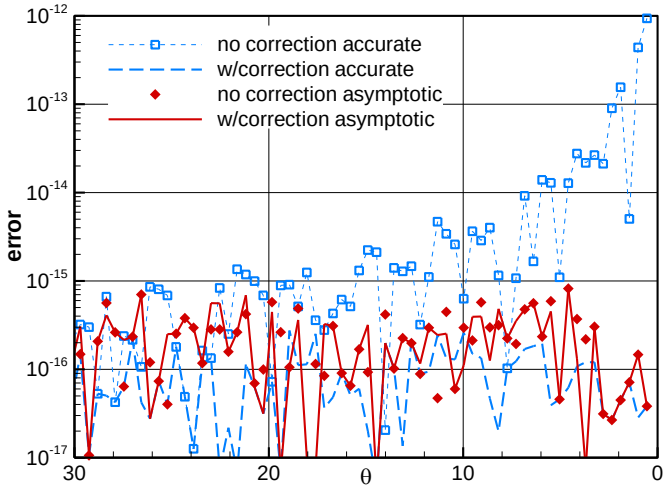


Fig. A.19 Errors in Eq. (2.97) for Lobatto quadrature, n = 400

calculations, reducing the maximum error by over two orders of magnitude. The results in Fig. A.8 and Fig. A.18 are calculated with roots from double precision recurrent calculations. Fig. A.19 shows the results when the composite method is also used in the root calculation. Much of the improvement given by the correction is due to errors in the roots. If the roots are calculated with more accurate asymptotic polynomial calculations, the difference between corrected and uncorrected results is relatively small.

Appendix B - Galerkin Method with Flux Boundary Conditions

Consider a problem approximated by the following trial functions:

$$y(x) = \sum_{i=0}^{n+1} \ell_i(x)\, y_i \qquad \text{(B.1)}$$

where $\ell_i(x)$ are the Lagrange interpolating polynomials. These functions are then substituted into the differential equation to form the residual as in Eq. (1.8). Assume the boundary conditions are:

$$y = 0 \quad at\ x = 0$$
$$\frac{dy}{dx} + hy = 0 \quad at\ x = 1$$

The usual procedure in orthogonal collocation, pseudospectral or differential quadrature methods is to satisfy both of these conditions exactly, i.e. use boundary collocation [Finlayson (1972), p. 101; Villadsen and Michelsen (1978), p. 137; Bert and Malik (1996); Belomo (1997); Trefethen (2000), p. 137; Boyd (2000), p. 111, Peyret (2002), p. 59].

$$y_0 = 0$$
$$\sum_{i=0}^{n+1} A_{n+1,i} y_i + h y_{n+1} = 0$$

where $\boldsymbol{A}$ is the first derivative matrix. If the boundary values are eliminated, Eq. (B.1) becomes:

$$y(x) = \sum_{i=1}^{n} [\ell_i(x) + b_i \ell_{n+1}(x)]\, y_i \qquad \text{(B.2)}$$

where $b_i = -A_{n+1,i}/(A_{n+1,n+1} + h)$. The trial functions are now the terms in the brackets of Eq. (B.2) and the Galerkin method weights the residual, R, by these trial functions. If the integrals are approximated by quadrature using all n + 2 collocation points, the result is:

$$\sum_{k=0}^{n+1} W_k R(x_k, \boldsymbol{y})[\ell_i(x_k) + b_i \ell_{n+1}(x_k)] = W_i R(x_i, \boldsymbol{y}) + W_{n+1} R(1, \boldsymbol{y}) b_i = 0 \qquad \text{(B.3)}$$

Eq. (B.3) is not equivalent to a collocation method because of the second term involving the residual at the boundary. Collocation sets only the first interior residual term to zero, so the nonzero second term seriously violates the Galerkin method. The boundary weight, W_{n+1}, is O(n^2), so this inconsistency creates an error over and above that caused by the approximate quadrature. Numerical experiments suggest the error introduced is significant. Since the residual at the boundary decreases exponentially with n, the method with boundary collocation converges exponentially, but usually at a much slower rate, especially for fluxes. Boundary collocation also causes errors in the overall mass or energy balance (see Section 3.1.4).

Boundary collocation works well with Gauss points or Radau points with W_{n+1} = 0. However, Lobatto points usually produce greater accuracy if boundary conditions are properly treated.

Chebyshev points use Clenshaw-Curtis quadrature, which also has a nonzero boundary weight. Although it is a less accurate quadrature, boundary collocation should not be used with Chebyshev points either. W_{n+1} for Clenshaw-Curtis quadrature is roughly half that for Lobatto quadrature, so the inconsistency is smaller, but still significant.

A far better solution is to treat the boundary condition as a natural condition, so that it becomes part of the approximation. This approach is standard for most Galerkin applications and is described more fully in Section 3.1.3 and 3.1.4. The method is conservative with a natural treatment. The examples demonstrate that it works very well. Others have proposed similar alternatives to boundary collocation [Young (1977), Canuto, *et al*. (2006), Funaro (1992), Shen and Tang (2011)]; however, these methods appear not to have gained much traction. Finally compelling evidence for the correct treatment was presented recently [Young (2019)]. Mathematicians often pay little attention to the accuracy of fluxes, whereas it is one of the most important results for engineers, since it determines the exchange between the objects being modeled.

References

Ames, W. F., *Nonlinear partial differential equations in engineering*, Academic Press (1965)

Abramowitz, M. and Stegun, I. A. (Eds.), *Handbook of Mathematical Functions, with Formulas, Graphs and Mathematical Tables,* 9th printing, Dover, New York, 1972.

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D., LAPACK Users' Guide (Third ed.). Philadelphia, PA: Society for Industrial and Applied Mathematics (1999).

Atanasoff, J.V. and Brandt, A.E. “Application of Punched Card Equipment to the Analysis of Complex Spectra,” *J. Optical Soc. of America*, **26**, 83-88 (1936).

Bakker, M. “A Note on C^0 Galerkin Methods for Two-Point Boundary Value Problems,” *Numer. Math.*, **38** 447-453 (1981).

Bellman, R. and Casti, J.: “Differential Quadrature and Long-Term Integration,” *J. Math. Anal. Appl.*, **134**, 235-238 (1971).

Bellman, R., Kashef B.G., and Casti, J.: “Differential Quadrature: A Technique for the Rapid Solution of Nonlinear Partial Differential Equations,” *J. Comp. Physics* **10**, 40-52 (1972).

Bellman, R., *Methods of Nonlinear Analysis*, **2** Academic Press, New York (1973).

Bellomo, N.: “Nonlinear Models and Problems in Applied Sciences from Differential Quadrature to Generalized Collocation Methods,” *Math. Comput. Modelling*, **26**, 4, 13-34 (1997).

Bert, C.W. and Malik, M.: “Differential Quadrature Method in Computational Mechanics: A Review,” *Appl. Mech. Review*, **49**, 1 (1996).

Biezeno, C.B. and Koch, J.J., “Over een Nieuene Methode ter Berekening van Vlokke Platen met Toepassing op Enkele Voor de Techniek Belangrijke Belastingsgevallen, *Ing. Grav.* **38**, 25-36 (1923).

Biezeno, C.B. “Graphical and Numerical Methods for Solving Stress Problems,” Proc. 1st Intl. Congress Appl. Mech., Delft, 3-17 (1924).

Bird, R.B., Stewart, W.E. and Lightfoot, E.N., *Transport Phenomena*, John Wiley and Sons, (1960).

Bogaert, I., “Iteration-Free Computation of Gauss-Legendre Quadrature Nodes and Weights”, *SIAM J. Sci. Computing*, **36** (3) A1008-A1026 (2014).

Boyd, J.P., *Chebyshev and Fourier Spectral Methods*, 2nd ed., Dover Publications (2000).

Bubnov, I.G., “Report on the works of Professor Timoshenko which were awarded the Zhuranskyi Prize,” Symposium of the Institute of Communication Engineers, No. 81, All Union Special Planning Office (1913). (in Russian)

Bubnov, I.G., *Structural Mechanics of Shipbuilding*, (1914) (in Russian).

Canuto, C. and Quarteroni, A.: “Spectral and Pseudo-spectral Methods for Parabolic Problems with Nonperiodic Problems Boundary Conditions,” *Calcolo*, **18**, 3 197-217 (1981).

Canuto, C., *Boundary Conditions in Chebyshev and Legendre Applications*, NASA Contract Report 172469 (1986)

Canuto, C., Hussaini, M.Y., Quarteroni, A. and Zang, T.A., *Spectral Methods in Fluid Dynamics*, Springer-Verlag, Berlin (1988)

Canuto, C., Hussaini, M., Quarteroni, A. and Zang, T., *Spectral Methods: Fundamentals in Single Domains*, Springer, Berlin (2006)

Canuto, C., Hussaini, M., Quarteroni, A. and Zang, T., *Spectral Methods Evolution to Complex Geometries and Applications to Fluid Dynamics*, Springer, Berlin (2007)

Carey, G.F. and Finlayson, B.A., “Orthogonal Collocation on Finite Elements,” *Chem. Engr. Sci*. **30**, 587-596 (1975).

Chen, C.M., “Superconvergent Points of Galerkin’s Method for Two-Point Boundary Value Problems,” Numer. Math. J. Chinese University **1** 73-79 (1979).

Carey, G., Humphrey, D., and Wheeler, M. F., “Galerkin and Collocation-Galerkin Methods with Superconvergence and Optimal Fluxes,” *Intl. J. Num. Meth. Engr*., 17, 6, 939-950 (1981) 939-950.

Ciarlet, P.G. and Raviat, P.A.: “The Combined Effect of Curved Boundaries and Numerical Integration in Isoparametric Finite Element Methods,” in *The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations*, A.K. Aziz, ed., Academic Press, New York, pp. 409–474 (1972).

Ciarlet, P.G. *The Finite Element Method for Elliptic Problems*, North-Holland, Amsterdam, (1978).

Clenshaw, C.W. and Curtis, A.R., “A Method for Numerical Integration and an Automatic Computer,” *Numerische Mathematik* 2, 197 (1960).

Clenshaw, C. W. and Norton, H. J.: “The Solution of Nonlinear Ordinary Differential Equations in Chebyshev Series," *Comp. J.*, **6**, 88-92 (1963).

Clough, R.W., “The Finite Element Method in Plane Stress Analysis,” *Proc. of the 2nd ASCE Conf. Electronic Computation*, Pittsburg, PA, USA (1960).

Clough, R.W., “Early History of the Finite Element from the Viewpoint of a Pioneer,” *Intl. J. Num. Meth. Eng.*, **60**, 283-287 (2004).

Cohen, G., Joly, P., Roberts, J. E. and Tordjman, N.: “Higher Order Triangular Finite Elements With Mass Lumping for the Wave Equation,” *SIAM J. Numer. Anal.*, **38,** 6, 2047-2078 (2001).

Collatz, L., *The Numerical Treatment of Differential Equations*, Springer, 1960

Courant, R., remarks on “weighted averages of the residual”, in discussion following C.B. Biezeno’s paper at Proc. 1st Intl. Congress Appl. Mech., Delft (1924), p. 17.

Courant, R., “Variational Methods for the Solution of Problems of Equilibrium and Vibrations,” *Bulletin of the American Mathematical Society*, 49, 1-23 (1943).

Crandall, S. H., *Engineering Analysis*, McGraw-Hill, (1956).

Davis, M.E.: *Numerical Methods and Modeling for Chemical Engineers*, John Wiley, New York (1984)

DeBoor, C. and Swartz, B.: “Collocation at Gaussian Points”, *SIAM J. Numer. Anal.*, **10** 582–606 (1973).

Diaz, J., *A Hybrid Collocation-Galerkin Method for Two-point Boundary Value Problems Using Continuous Piecewise Polynomial Spaces*, Ph.D. Thesis, Rice Univ. (1975)

Diaz, J., “A Collocation-Galerkin Method for the Two-point Boundary Value Problem Using Continuous Piecewise Polynomial Spaces,” *SIAM J. Num. Anal.*, **14** (5) 844-858 (1977).

Douglas, J., Jr., Dupont, T., “A Finite Element Collocation Method for Quasilinear Parabolic Equations”, *Math. Comp.*, **27** 17-28 (1973).

Douglas, J., Jr., Dupont, T., The Effect of Interpolating the Coefficients in Nonlinear Parabolic Galerkin Procedures”, *Math. Comp.*, **29** (130) 360-389 (1975).

Douglas, J., Jr., Dupont, T. and Wheeler, M. F., “H^1-Galerkin Methods for the Laplace and Heat Equations”, *Math. Aspects of Finite Elements in Partial Differential Equations*, Carl De Boor, Ed., Academic Press, 383 (1974).

Dubiner, M.: “Spectral Methods on Triangles and Other Domains,” *J. of Sci. Comp.*, **6**, 4, 345–390 (1991).

Dunn, R., and Wheeler, M. F.: “Some Collocation-Galerkin Methods for Two-Point Boundary Value Problems,” *SIAM J. Numer. Anal.*, **13,** 5, 720-733 (1976).

Dupont, T.: “A Unified Theory of Superconvergence for Galerkin Methods for Two-Point Boundary Problems”, *SIAM J. Numer. Anal.*, **13,** 3, 362–368 (1976).

Dyksen, W.R., Houstis, E. N., Lynch, R. E. and Rice, J. R.: “The Performance of the Collocation and Galerkin Methods with Hermite Bi-cubics,” *SIAM J. Numer. Anal.*, **21** 4, 695–715 (1984).

El Daou, M. and Ortiz, E.L., “A Note on Accuracy Estimations of the Local Truncation Error of Polynomial Methods for Differential Equations,” *Appl. Math. Lett.*, **5** 6, 69-72 (1992).

El Daou, M., Ortiz, E.L. and Samara, H.: “A Unified Approach to the Tau Method and Chebyshev Series Expansion Techniques,” *Computers Math. Applic.,* **25**, 3, 73-82 (1993).

El Daou, M. and Ortiz, E.L.: "A Recursive Formulation of Collocation in Terms of Canonical Polynomials," *Computing*, **52,** 177-202 (1994): other references included, “A Recursive Formulation of Galerkin’s Method Based on the Tau Method,” Res. Report Imperial College, London (1992)

El Daou, M. and Ortiz, E.L.: "The Tau Method as an Analytic Tool in the Discussion of Equivalence Results Across Numerical Methods," *Computing*, **60,** 365–376 (1998).

El Daou, M. and Ortiz, E.L.: "The Weighted Subspaces of the Tau Method and Orthogonal Collocation," *J. Math. Anal. Appl.*, **326**, 622-631 (2007).

Ellsaesser, H.W., “Evaluation of Sprectral Versus Grid Methods of Hemispheric Numerical Weather Prediction,” *J. Applied Meteorology*, **5**, 246-262 (1966).

Elnashaie, S.S.E. and Cresswell, D.L., “On Dynamic Modelling of Porous Catalyst Particles,” *Chem. Engr. Sci*. **28**, 1387 (1973).

Faddeeva, V.N., *Computational Methods of Linear Algebra,* Dover Publications, New York (1959).

Ferguson, N.B., “Orthogonal Collocation as a Method of Analysis in Chemical Reaction Engineering,” Ph.D. thesis, University of Washington, Seattle, WA (1971).

Ferguson, N.B. and Finlayson, B.A., “Transient Chemical Reactor Analysis by Orthogonal Collocation,” *Chem. Engr. J*., **1** (4) 327-336 (1970).

Ferguson, N.B. and Finlayson, B.A., “Error Bounds for Approximate Solutions to Nonlinear Ordinary Differential Equations,” *Amer. Inst. Chem. Engr. J*., **18** (5), 1053-1059 (1972).

Finlayson, B.A. and Scriven, L.E., “The Method of Weighted Residuals – A Review,” *Applied Mechanics Review*. **19**, (9) 735-748 (1966).

Finlayson, B.A., “Packed Bed Reactor Analysis with Orthogonal Collocation,” *Chem. Engr. Sci*. **26**, 1081-1091 (1971).

Finlayson, B.A., *The Method of Weighted Residuals and Variational Principles*, Academic Press, New York, NY (1972), SIAM Classics in Applied Mathematics (2014).

Finlayson, B.A., “Orthogonal Collocation in Chemical Reaction Engineering,” *Cat. Rev. Sci-Eng.* **10** 69-138 (1974).

Finlayson, B.A. and Young, L.C., “Mathematical Models of the Monolith Catalytic Converter: Part III. Hysteresis in Carbon Monoxide Reactor,” *Amer. Inst. Chem. Engr. J.*, **25** (1), 192-196 (1979).

Finlayson, B.A.: “Orthogonal Collocation on Finite Elements – Progress and Potential,” *Mathematics and Computers in Simulation*, **22** (1), 11-17 (1980).

Finlayson, B.A., *Nonlinear Analysis in Chemical Engineering*, McGraw-Hill, New York (1980). Ravenna Park Publ., Seattle (2003).

Fix, G.J.: “Effect of Quadrature Errors in the Finite Element Approximation of Steady State, Eigenvalue and Parabolic Problems,” in *The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations*, A.K. Aziz (ed.), Academic Press, pp. 525–556 (1972).

Fornberg, B., *A Practical Guide to Pseudospectral Methods*, Cambridge University Press (1996).

Franklin, J., *Matrix Theory*, Prentice-Hall, Englewood Cliffs, NJ (1968).

Frazer, R.A., Duncan, W.J., and Collar, A.R., Elementary Matrices: and Some Applications to Dynamics and Differential Equations, Cambridge University Press (1938, reprint 2007)

Frazer, R. A, Jones, W. P., and Skan, S. W., "Approximation to Functions and to the Solutions of Differential Equations," *Great Britain Aero. Res. Council*, London, Report and Memo No. 1799, (1937).

Fried, I.: “Numerical Integration in the Finite Element Method,” *Computers & Structures,* **4**, 5, 921–932 (1974).

Fried, I. and Malkus, D.S., “Finite Element Mass Matrix Lumping by Numerical Integration with No Convergence Rate Loss,” Intl. J. Solids and Structures, **11**, 4, 461–466 (1975).

Funaro, D., “A multidomain spectral approximation of elliptic equations” Numer. Methods Partial Differential Equations, vol 2 (1986) 187-205

Funaro, D., “Domain Decomposition for Pseudo Spectral Approximations, Part I. Second Order Equations in One Dimension”, *Num. Math*. Vol. 52, (1988) 329-344.

Funaro, D., *Polynomial Approximation of Differential Equations*, Lecture notes in Physics, m8, Springer, Heidelberg (1992).

Galerkin, B.G.: “Series Solution of Some Problems Related to Elastic Equilibrium of Rods and Plates,” *Vestn. Inzhenerov Tech.*, **19**, 897-908 (1915).

Gander, M.J. and Wanner, G., “From Euler, Ritz, and Galerkin to Modern Computing,” *SIAM Review*, **54**, 4 (2012).

Gatteschi, L. “On the zeros of Jacobi polynomials and Bessel functions,” *International Conference on Special Functions: Theory and Computation* (Turin, 1984), *Rend. Sem. Mat. Univ. Politec.* Torino, Special Issue vol. 1985, pp. 149–177.

Gatteschi, L. and Pittaluga, G., “An asymptotic expansion for the zeros of Jacobi polynomials,” *Mathematical Analysis*. *Teubner-Texte Math.*, vol. 79, pp. 70–86. Teubner, Leipzig (1985).

Gautschi, W., *Orthogonal Polynomials: Computation and Approximation*, Oxford Univ. Press (2004).

Gautschi, W., “Orthogonal Polynomials (in Matlab)”, *J. Computational and Applied Mathematics*, vol. 178 (June 2005) 215-234.

Gautschi, W. and Giordano, C., “Luigi Gatteschi’s work on asymptotics of special functions and their zeroes”, *Numer. Algor.*, vol. 49, no. 1, 11-13 (2008) DOI: 10.1007/s11075-008-9208-5

Golub, G.H. and Welsch, J.H., “Calculation of Gaussian Quadrature Rules”, *Mathematics of Computation,* vol. 23 (April 1969), pp. 221-230.

Gottlieb, D. and Lustman, L., “The Dufort-Frankel Method for Parabolic Initial Boundary Value Problems”, *Computers and Fluids*, vol. 11, no. 2, 107-120 (1983).

Gottlieb, D. and Orzag, S.A.: *Numerical Analysis of Spectral Methods: Theory and Applications,* SIAM, Philadelphia, PA (1977).

Gray, W.G.: “An Efficient Finite Element Scheme for Two-Dimensional Surface Water Computations”, *Finite Elements in Water Resources*, W.G. Gray, G.F. Pinder and C.A. Brebbia (ed.), Pentech Press, London (1977).

Gray, W.G. and Van Genuchten, M.T.: “Economical Alternatives to Guassian Quadrature Over Isoparameteric Quadralaterals,” *Intl. J. Num. Meth. Eng.* 12 1478-1484 (1978).

Gupta, K.K. and Meek, J.L., “A Brief History of the Beginning of the Finite Element Method,” *Intl. J. Num. Meth. Engr.*, 39, 3761-3774 (1996).

Hairer, E., Nörsett, S.P. and Wanner, G., *Solving Ordinary Differential Equations I: Nonstiff Problems,* 2nd ed., Springer Verlag, Berlin (1993).

Hairer, E. and Wanner, G., *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems,* 2nd ed., Springer Verlag, Berlin (1996).

Hale, N. and Townsend, A., “Fast and Accurate Computation of Gauss-Legendre and Gauss-Jacobi Quadrature Nodes and Weights”, *SIAM J. Sci. Computing*, 35 (2) A652-A674 (2013).

Hamming, R.W., *Numerical Methods for Scientists and Engineers*. McGraw-Hill, New York, NY (1962), 2nd edition (1973).

Harrington R., *Field Computations by Moments Methods*, Macmillan (1968), republished by IEEE (1993).

Harrington R., “Origin and Development of the Method of Moments for Field Computations,” *IEEE Antennas and Propagation Soc. Mag*., **32**, (3) 31-36 (1990).

Hennart, J.-P.: “Topics in Finite Element Discretization of Parabolic Evolution Problems,” Lecture Notes in Math. 909, Springer-Verlag, Berlin, Heidelberg, (1982). [Cohen, et al. (2001) list an earlier reference which could not be located: Hennart, J.-P., Sainz, E., and Villegas, M.: “On the Efficient Use of the Finite Element Method in Static Neutron Diffusion Calculations,” *Computational Methods Nuclear Engrg.*, **1**, 3.87 (1979)]

Hildebrand, F.B., *Introduction to Numerical Analysis,* 2nd ed., Dover Publications, New York, NY (1987).

Houstis, E.N., Lynch, R.E., Rice, J.R. and Papatheodorou, T.S.: “Evaluation of Numerical Methods for Elliptic Partial Differential Equations,” *J. Comp. Phys.*, **27**, 3, 323–350 (1978).

Ilanko, Sinniah, “Comments on the Historical Bases of the Rayleigh and Ritz Methods,” *J. Sound and Vibration*, 319, **1-2**, 731-733 (2009).

Irons, B. M., “Engineering Application of Numerical Integration in Stiffness Methods,” *Amer. Inst. Aero. and Astro. J.*, **4**, 2035–2037 (1966).

Johnson, D.: *Chebyshev Polynomials in the Spectral Tau Method and Applications to Eigenvalue Problems*, NASA Contractor Report No. 198451 (1996).

Kantorovich, L.V. and Akilov, G.P., *Functional Analysis in Normed Spaces*, Macmillan, New York (1964).

Kantorovich, L.V.: “On an Approximation Method for the Solution of a Partial Differential Equation”, *Dokl. Akad. Nauk SSSR*, **2**, 532-536 (1934)

Kantorovich, L.V. and Krylov, V.I, *Approximate Methods in Higher Analysis*, Interscience (1958).

Karniadakis, G. and Sherwin, S.: *Spectral/hp Element Methods for Computational Fluid Dynamics*, Oxford Univ. Press (2013).

Karpilovskaya, E.B.: “On the Convergence of an Interpolation Method for Ordinary Differential Equations,” *Uspekhi Matem. Nauk*, **8**, 3 (55), 111-118 (1953).

Karpilovskaya, E.B.: “Convergence of the Collocation Method for Certain Boundary Value Problems of Mathematical Physics,” *Sibirsk. Matem. Zh.*, **4**, 3, 632-640 (1963).

Katchanovski, I., “A Puzzle in the Invention and Patenting of the Electronic Computer in the US,” In *Development of Mathematical Ideas of Mykhailo Kravchuk [Krawtchouk]*, Edited by Nina Virchenko, Ivan Katchanovski, V. Haidey, Roman Andrushkiw and Roman Voronka. Shevchenko Scientific Society (USA), New York, and National Technical University of Ukraine, Kyiv, (2004).

Kennedy, C.A. ad Carpenter, M.H., “Diagonally Implicit Runge-Kutta Methods for Ordinary Differential Equations. A Review”, National Aeronautics and Space Administration, report no. NASA/TM-2016-219173 (Mar. 2016).

Komatitsch, D. and Villote, J.-P.: “The Spectral Element Method: An Efficient Tool to Simulate the Seismic Response of 2D and 3D Geologic Structures,” *Bull. Seismological Soc. America*, **88**, 2, 368-392 (1998)

Kopal, Z., *Numerical Analysis*, Chapman and Hall, London (1955).

Kravchuk, M.P.: “On Krylov’s Method in the Theory of Approximate Integration of Differential Equations,” (in Ukrainian),*Tr. Fiz. – Mat. Viddilu VUAN*, **5**, 2, 12-33 (1926) [reference from Lucka and Lucka (1992)]

Kravchuk, M.P.: “Application of the Method of Moments to the Solution of Linear Differential and Integral Equations,” (in Ukrainian), *Kiev. Soobshch. Akad. Nauk UkSSR,* **1**, 168 (1932); [referenced in Kantorovich and Krylov (1958), Mikhlin (1964), Shuleshko (1959)].

Křížek, M. and Neittaanmäki, P., “Bibliography of Superconvergence”, in *Finite Element Method, Superconvergence, Post-Processing and a Posteriori Estimates*, **196**, 315-348 Marcel Dekker, New York (1998).

Krylov, N.M., “On a Method of Approximate Integration for Which the Ritz Method as Well as the Method of Least Squares are Special Cases,” *C.R. Acad. Sci. Paris*, **182**, 676-678 (1926) [reference from Lucka and Lucka, 1992].

Krylov, N.M., “Methods of Approximate Solution of Problems of Mathematical Physics” *Mémorial Sci. Math., Paris*, **49**, (1931).

Krylov, N.M and Bogoliubov, N., *Introduction to Non-linear Mechanics*, Annals of Mathematical Studies 11, Princeton Univ. Press (1943).

Krylov, N.M.. Collected Works in 3 volumes (in Ukranian) , *Izd. Akad. Nauk. UkrSSR*, Kiev, vol. 1-3, (1961) [reference from Lucka and Lucka, 1992].

Krylov, V.I., *Approximate Calculation of Integrals,* Macmillan, New York, NY (1962).

Lanczos, C., “Trigonometric Interpolation of Empirical and Analytical Functions”, *J. Math. Phys.*, Vol. 17, 123-199 (1938).

Lanczos, C., *Applied Analysis*, p. 504, Prentice-Hall, Englewood Cliffs, NJ (1956).

Lanczos, C.: “Legendre Versus Chebyshev Polynomials,” in Topics in Numerical Analysis, Proceedings of the Royal Irish Academy Conference on Numerical Analysis 1972, J.J.H. Miller, ed., Academic Press, London, pp. 191–201 (1973).

Lapidus, L. and Pinder, G.F.: *Numerical Solution of Partial Differential Equations in Science and Engineering*, John Wiley, New York (1999)

Leissa, A.W., “The Historical Bases of the Rayleigh and Ritz Methods,” *Journal of Sound and Vibration*, **287** 961-978 (2005).

Lesaint, P. and Zlámal, M., “Superconvergence of the Gradient of Finite Element Solutions,” *RAIRO Anal. Numer.* **13** 139-166 (1979).

Lether, F.G., “On the Construction of Gauss-Legendre Quadrature Rules,” *J. Comp. and Applied Mathematics,* **4** 47-52 (1978).

Leyk, Z.: “A C^o-Collocation-Like Method for Boundary Value Problems,” *Numerische Mathametik*, **49**, 39-53 (1986).

Leyk, Z.: “A C^o-Collocation-Like Method for Elliptic Equations on Rectangular Regions,” *J. Austral. Math. Soc. Ser. B* , **38**, 368-387 (1997).

Lucka, T.F. and Lucka, A.Y.: “Development of Direct Methods in Mathematical Physics in the Works of M.P. Kravchuk,” translated from *Ukrain. Math. J.*, **44**, 7, 931-939 (1992).

McMahon, J., “On the Roots of Bessel and Certain Related Functions,” *Annals of Mathematics*, **9**, 23-30 (1894).

Maday, Y. and Patera, A. T., “Spectral Element Methods for the Incompressible Navier-Stokes Equations” In *State-of-the-Art Surveys on Computational Mechanics*, A.K. Noor, editor, ASME, New York (1989).

Melenk, J.M., Gerdes, K. and Schwab, C., “Fully Discrete hp-Finite Elements: Fast Quadrature,” *Comp Meth. in Appl. Mech. and Eng.*, **190**, 32, 4339–4364 (2001).

Melosh, R.J.,”Bases for the Derivation of Matrices for the Direct Stiffness Method,” *Amer. Inst. Aero. and Astro. J.*, **1**, 1631-1637 (1963).

Michelsen M.L. and Villadsen J., “A convenient computation procedure for collocation constants,” *Chem. Eng. J.* **4** 64-68 (1972).

Michelsen M.L. and Villadsen J., “Polynomial Solution of Differential Equations,” in: Foundations of Computer Aided Chemical Process. Proceedings of an International Conference, R.S.H. Mah, W.D. Seider (ed.), 341-368 (1981).

Mikhlin, S. G., *Variational Methods in Mathematical Physics*, Pergamon/Macmillan (1964).

Mulder, W.A.: “Higher-Order Mass-Lumped Finite Elements for the Wave Equation,” *J. Comput. Acoustics*, **9**, 2, 671-680 (2001).

Namasivayam, S. and Ortiz, E.L., “Error Analysis of the Tau Method: Dependence or the Approximation Error on the Choice of Perturbation Term,” *Computers Math. Applic.*, **25**, 89-104 (1993).

Nakao, M.: “Superconvergence Estimates at Jacobi Points of the Collocation-Galerkin Method for Two Point Boundary Value Problems”, *J. Information Processing*, **7** 31-34 (1984).

Nielson, K.L.: *Methods in Numerical Analysis*, MacMillan, NY, pp. 150-4 (1956).

Oden, J.T., “Historical Comments on Finite Elements,” *Proceeding of the ACM Conference on History of Scientific and Numeric Computation*, Princeton, NJ, USA, 125-130, May 13-15 (1987).

Olver, F. W. J., *Asymptotics and Special Functions*, Academic, New York, 1974.

Olver, F. W. J., Boisvert, R.F., Lozier, D.W. and Clark, C.W., *NIST Handbook of Mathematical Functions*, Cambridge University Press (2018). Also online Digital Library of Mathematical Functions - dlmf.nist.gov

Ortiz, E.L., "The Tau Method," *SIAM J. Numer. Anal.*, **6**, (3) 480-492 (1969).

Ortiz, E.L., "Step by Step tau Method – Part I. Piecewise Polynomial Approximations," *Comput. and Math. Applic.*, **1**, 381-392 (1975).

Orzag, S.A., "Comparison of Pseudo Spectral and Spectral Approximation," *Studies in Applied Mathematics* **51**(3) 253-259 (1972).

Orzag, S.A., "Fourier Series on Spheres," *Monthly Weather Review*, **102**, 56-75 (1974).

Orszag, S.A.: "Spectral Methods for Problems in Complex Geometries," *J. of Comp. Physics,* **37** 1, 70–92 (1980).

Patera, A.T., "A Spectral Element Method for Fluid Dynamics – Laminar Flow in a Channel Expansion," *J. Comp. Physics*, **54** (3) 468-488 (1984).

Petrov G.I., "Application of the Galerkin Method to the Problem of the Stability of Viscous Fluid," *J. of Applied Mathematics and Mechanics*, **4**, 1-13, (1940).

Peyret, R.: *Spectral Methods for Incompressible Viscous Flows*, Springer (2002)

Picone, M.: "Sul Metodo Delle Minime Potenze Ponderate e Sul Metodo di Ritz per il Calcolo Approssimato nei Problemi Della Fisica-Matematica," *Rend. Circ. Mat. Palermo*, **52**, 225-253 (1928).

Pinder, G.F., Frind, E.O. and Celia, M A.: "Groundwater Flow Simulation Using Collocation Finite Elements," *Proc. Second Int. Conf. Finite Elements in Water Resources*, (eds Brebbia et al.), 1.171-1.185, Pentech Press, London, (1978).

Platzman, G.W., "The Spectral Form of the Vorticity Equation," *J. Meteorology*, **17**, 635-644 (1960).

Prenter, P.M. and Russell, R.D.: "Orthogonal Collocation for Elliptic Partial Differential Equations", *SIAM J. Numer. Anal.*, **13**, 923 (1976).

Quan, J.R. and Chang, C.T.: "New Insights in Solving Distributed System Equations by the Quadrature Method – I. Analysis," *Comput. Chem. Engng*., **13**, 7, 779-788 (1989).

Raviart, P.A.: "The Use of Numerical Integration in Finite Element Methods for Parabolic Equations," in *Topics in Numerical Analysis, Proceedings of the Royal Irish Academy Conference on Numerical Analysis 1972*, J.J.H. Miller, ed., Academic Press, London, pp. 233–264 (1973).

Rawlings, J.B. and Ekerdt, J.B.: *Chemical Reactor Analysis and Design Fundamentals*, 2nd Ed., Nob Hill Publishing (2015)

Rayleigh, Lord (J.W. Strutt), “On the Theory of Resonance,” *Phil. Trans. Roy. Soc.* (London) A161, 77-118 (1871). Also (1964). *Scientific Papers,* Vol I, 55, 1869-1881, Dover, New York.

Rayleigh, Lord (J.W. Strutt), Some General Theorems Relating to Vibrations, *Proc. London Math. Soc.* **4**, 357-368 (1873). also *Scientific Papers,* Vol I, 521, 1869-1881. Dover, New York (1964).

Rayleigh, Lord (J.W. Strutt), *Theory of Sound*, Macmillan, London (1877 1st Ed., 1896 2nd Ed.), also Dover, New York (1945).

Rayleigh, Lord (J.W. Strutt), “On the Calculation of the Frequency of Vibration of a System in Its Greatest Mode, with an Example from Hydrodynamics,” *Phil. Mag.* 5, **47**, 566-572 (1899).

Rayleigh, Lord (J.W. Strutt), “On the Calculation of Chladni’s Figures for a Square Plate,” *Phil. Mag.* 6, **22**, 128 (1911).

Ritz, W., “Über eine neue Methode zur Lösung gewisser Variationsprobleme der mathematischen”, *Physik, J. Reine Angew. Mathematik* **135** 1-61 (1908), also collected works, Paris (1911).

Ritz, W., “Theorie der Transversalschwingungen einer quadratischen Platte mit freien Rändern,” *Ann. Phys*., **18**, 737–807 (1909).

Runge, M.: “Über Empirische Functionen und die Interpolation Zwischen Aequidistanten Ordinaten,” *Zeitschrift für Math. und Physik* tome XLVI (1901).

Schillinger, D., Evans, J.A., Frischmann, F., Hiemstra, R.R., Hsu, M.-C., and Hughes, T.J.R.: “A Collocated C^0 Finite Element Method: Reduced Quadrature Perspective, Cost Comparison with Standard Finite Elements, and Explicit Structural Dynamics,” *Intl. J. Num. Meth. Engrg.*, **102**, no. 3-4, 576-631 (2015).

Sellars, J.R., Tribus, M. and Klein, J.S., “Heat Transfer to Laminar Flow in a Round Tube or Flat Conduit – The Graetz Problem Extended”, Trans. ASME, vol. 78, 441-448 (1956).

Schillinger, D., Evans, J.A., Frischmann, F., Hiemstra, R.R., Hsu, M.-C., and Hughes, T.J.R.: “A Collocated C^0 Finite Element Method: Reduced Quadrature Perspective, Cost Comparison with Standard Finite Elements, and Explicit Structural Dynamics,” *Intl. J. Num. Meth. Engrg.*, **102**, no. 3-4, 576-631 (2015).

Shen,J. and Tang,T., *Spectral and Higher-Order Methods with Applications,* Science Press, Beijing (2006).

Shen,J., Tang,T. and Wang,L.-L., *Spectral Methods: Algorithms, Analysis and Applications*, Springer-Verlag, Series in Computation Mathematics, vol. 41, Berlin (2011).

Shu, C., *Differential Quadrature and Its Application in Engineering,* Springer, London (2000).

Shuleshko, P.: “A New Method of Solving Boundary-Value Problems of Mathematical Physics, Generalizations of Previous Methods of Solving Boundary-Value Problems of Mathematical Physics,” *Austral. J. Appl. Sci.* **10**, 1-7, 8-16 (1959).

Silberman, I., “Planetary Waves in the Atmosphere,” *J. Meteorology*, **11**, 27-34 (1954).

Slater, J. C.: "Electronic Energy Bands in Metal," *Phys. Rev.*, **45**, 794-801 (1934).

Strang, G. and Fix, G.J., *An Analysis of the Finite Element Method,* Prentice-Hall, Englewood Cliffs, NJ (1973).

Stroud, A.H. and Secrest, D., *Gaussian Quadrature Formulas*, Prentice-Hall, Englewood Cliffs, NJ (1966).

Swarztrauber, P.N., “On Computing the Points and Weights for Gauss-Legendre Quadrature,” *SIAM J. Sci. Computing*, 24 (3) 945-954 (2003).

Szegö, G.: “Uber Einige Asymptotische Entwicklungen der Legendreschen Funktionen”, Proc. London Math. Soc., s2-36, 427–450 (1934).

Szegö, G., *Orthogonal Polynomials*, 4th ed., American Mathematical Soc., Providence, RI (1975).

Timoshenko, S. P. “Sur la stabilit´e des syst`emes ´elastiques,” *Annales des Ponts et Chauss´ees*, **9** 496–566 (1913). originally published in *Isvestija Kievskogo Politechnitscheskogo Instituta*, Kiev (1910) (in Russian).

Traub, J. F., *Iterative Methods for the Solution of Equations*, Prentice Hall, Englewood Cliffs, New Jersey, 1964.

Trefethen, L.N., *Spectral Methods in Matlab,* SIAM, Philadelphia, PA (2000).

Trefethen, L.N. *"Is Gauss Quadrature Better than Clenshaw-Curtis?". SIAM Review 50, 67–87 (2008).*

Tricomi, F.G. “Sugli zeri dei polinomi sferici ed ultrasferici,” Ann. Mat. Pura Appl., 31, 93-97 (1950).

Turner, M.J., Clough, R.W., Martin, H.C. and Topp, L.J., “Stiffness and Deflection Analysis of Complex Structures,” J. Aeronaut. Sci., 25, 805-823 (1956).

Van de Vosse, F.N. and Minev, P.D.: *Spectral elements methods: Theory and applications,* EUT Report 96-W-001, Eidenhoven Univ. Tech., The Netherlands (1996) 96-W-001, Eindhoven University of Technology, 1996.

Varga, R., *Matrix Iterative Analysis*, Springer (2009).

Villadsen J., *Selected Approximation Methods for Chemical Engineering Problems*, Inst. for Kemiteknik Numer. Inst., Danmarks Tekniske Hojskole (1970).

Villadsen, J.V. and Michelsen, M.L., *Solution of Differential Equation Models by Polynomial Approximation*, Prentice-Hall, Englewood Cliffs, NJ (1978).

Villadsen, J.V. and Stewart, W.E., “Solution of Boundary-Value Problems by Orthogonal Collocation,” *Chem. Engr. Sci*. **22**, 1483-1501 (1967).

Wheeler, M.F.: “A C^0-Collocation-Finite Element Method for Two-Point Boundary Value and One Space Dimension Parabolic Problems,” *SIAM J. Numer. Anal.*, **14**, 1, 71-90 (1977).

Wright, K.: “Chebyshev Collocation Methods for Equations," *Comp. J.,* **6**, 358-365 (1964).

Yakimiw, E., “Accurate Computation of Weights in Classical Gauss–Christoffel Quadrature Rules,” *J. Computational Physics*, **129**, 406-430 (1996).

Yamada, H., *Rept. Res. Inst. Fluid Engng.*, Kyushu Univ. **3**, 29 (1947); referenced by H. Fujita (1953). Also: “A method of approximate integration of the laminar boundary-layer equation,” *Rept. Res. Inst. Fluid Engng.*, Kyushu Univ. **6**, 87-98 (1950).

Young, L.C. and Finlayson, B.A., “Axial Dispersion in Nonisothermal Packed Bed Chemical Reactors,” *Ind. and Engr. Chemistry, Fund.*, **12** (4), 412-422, (1973).

Young, L.C., *The Application of Orthogonal Collocation to Laminar Flow Heat and Mass Transfer in Monolith Converters,* Ph.D. thesis, University of Washington, Seattle, WA (1974).

Young, L.C. and Finlayson, B.A., “Mathematical Models of the Monolith Catalytic Converter: Part I. Development of Model and Application of Orthogonal Collocation,” *Amer. Inst. Chem. Engr. J.*, **22** (2), 331-343 (1976).

Young, L.C. and Finlayson, B.A., “Mathematical Models of the Monolith Catalytic Converter: Part II. Application to Automotive Exhaust,” *Amer. Inst. Chem. Engr. J.*, **22** (2), 343-353 (1976).

Young, L.C., “A Preliminary Comparison of Finite Element Methods for Reservoir Simulation,” Advances in Computer Methods for Partial Differential Equations-II, R. Vichnevetsky (ed.), IMACS(AICA), Rutgers U., New Brunswick, N.J., 307-320 (1977)

Young, L.C., “A Finite-Element Method for Reservoir Simulation,” *Soc. Petr. Engrs. J.* **21**(1) 115-128, (Feb. 1981), paper SPE 7413 presented Oct. 1978.

Young, L.C., “A Study of Spatial Approximations for Simulating Fluid Displacements in Petroleum Reservoirs,” *Comp. Methods in Appl. Mech. and Engr.* **47**, 3-46 (1984).

Young, L.C., “Orthogonal Collocation Revisited,” *Comp. Methods in Appl. Mech. and Engr.* **345** (1) 1033-1076 (Mar. 2019), doi.org/10.1016/j.cma.2018.10.019, tildentechnologies.com/Numerics.

Zhang, Z., “Superconvergence of Spectral Collocation and *p*-Version Methods in One Dimensional Problems,” *Math. Comp.* **74** (252), 1621-36 (2005)

Zienkiewicz, O.C., *The Finite Element Method in Engineering Science*, McGraw-Hill, London (1971).