Computer Code Documentation

To facilitate solution of problems with the described methods, some computer source code, dynamic link libraries and executables are provided. These codes have been run under Windows with MinGW/G95, Excel 2010, and g++, gfortran and Octave using Cygwin. Older versions were run with Lahey Fortran 95 and MS Visual C++. Native code has been written in Fortran 90 and C++ for calculating the quadrature base points (collocation points), quadrature weights and various operators for derivatives, etc. The calculations and methods are described in Appendix A. The Fortran 90 code was used to create a DLL which interfaced with Visual Basic to support Excel. The C++ code was interfaced with Octave to create a MEX (Matlab EXecutable) file to support Matlab and Octave. These codes have not been tested under Matlab, but are supposed to be compatible. I would appreciate hearing from someone that tries them on Matlab.

This document describes only the code to calculate the quadrature points and weights, and the various operators described in Appendix A. This description is given for all four programming systems: C++, Fortran 90, Matlab/Octave and Excel. A simple test problem which demonstrates the call syntax is also described here for all languages. The code for the specific example problems and various utility routines are described in four other documents, one for each programming system.

The following descriptions use the following common terminology:

n - number interior points, total points are n + 1 (symmetric) or n + 2 (nonsymmetric)
symm – symmetry 0/1 no/yes
typ – type of points 0 Gauss, 1 Lobatto, Chebyshev (2nd kind), etc.
geom – geometry 0/1/2 cartesian, cylindrical, spherical grid
Xc – value of interior points
Wc – quadrature weights
Ac – first derivative matrix
An – first derivative of odd quantity (symmetric problem)
Bc – 2nd derivative Laplacian matrix
Cc – 2nd derivative Laplacian, symmetric stiffness matrix
Dc – mass matrix
Lc – Lagrange interpolating polynomial

Matlab/Octave (see Appendix A)

OCtest.m – program to demonstrate calling the following functions
OCCdefs.m – contains numeric and string designations for typ, geom and symm

The following are native Matlab code:

[xc,wc,Ac] = OCnonsym(n,typ) – nonsymmetric points, weights and derivative operator given n and typ 1 - 5 (Gauss, Lobatto, Chebyshev, Radau right/left)
[xc,wc,Ac,An] = OCsym(n,typ,geom) – symmetric base points, weights and derivative operators (even and odd) given n, typ (1 Gauss, 2 Lobatto, 3 Chebyshev) and geom
[Bc,Cc] = OCBCcoef(wc,Ac,An,symm) – 2nd derivative Laplacian and stiffness matrix given weights, first derivative matrices and symmetry (0/1 no/yes)

The following or compiled .MEX executables:

OCset(symm,typ,geom) – set symmetry (0/1 no/yes), type (1 Gauss, 2 Lobatto) and geometry (0/1/2 planar/cylindrical/spherical)
[xc,wc] = OCquad(n) – base (collocation) points and weights given the number of interior points
[Ac] = OCAcoef(xc) – first derivatives operator, given the points
[An] = OCAnonsym(Ac,xc) – first derivative operator for odd functions
[Bc] = OCBcoef(Ac,xc) – second derivative operator (Laplacian)
[Cc] = OCCcoef(Ac,xc,wc) – second derivative symmetric form (stiffness matrix)
[Dc] = OCDcoef(xc,wc) – mass matrix
[Lc] = OCLcoef(xc,x) – returns Lagrange interpolating polynomials evaluated at x, result is size(x) x size(xc), x may be a scalar or an array

Fortran 90/95

Test.f90 – a simple program which calls most of the functions below
OCC.f90 – module OrthogonalColloc contains:
Definitions: Gauss, Lobatto, Planar, Cylindrical, Spherical, Symmetric, Nonsymmetric
CollocSet(s,t,g) - set symmetry (true/false), type (1 Gauss, 2 Lobatto) & geometry (0,1,2)
Xc = Xpoints(n) - returns quadrature points, given number interior points
Wc = Weights(Xc) - returns quadrature weights
Ac = Acoeff(Xc) - returns first derivative matrix
Bc = Bcoeff(Xc,[Ac]) - returns second derivative matrix, Ac is optional
Cc = Ccoeff(Xc) - returns $\mathbf{A^tWA}$ for Lobatto, **-wB** for Gauss
Dc = MassMat(Xc) - Galerkin or Moments mass matrix
An = AnonSym(Xc,[Ac]) - first derivative of odd function, Ac is optional
L = Lcoef(x,Xc) – returns the values at x of Langrange polynomials through Xc, result is an array of size(Xc)
Ln = LcoefN(x,Xc) – Langrange polynomial through Xc evaluated at points x, result is a m x n array, where m = size(x) and n = size(Xc)
y = Interp(x,Xc,Yc) – interpolates values of Yc, Xc, to return y at x (arrays)
S = CollocSym() - returns the current setting for the symmetry
CollocCoef(Xc,Wc,A,[B],[C]) – calls the various functions to get 5 arrays with one call, B and C are optional. All arrays are size(Xc)

C++ Code

occtest.cpp – code which sets up an Orthcc class and calls most of the class functions
occ.h, occ.cpp – class Orthcc contains public functions. Definitions: Gauss, Lobatto, Planar, Cylindrical, Spherical, Symmetric, Nonsymmetric - const values useful for occ_set
occ_set(s,t,g) – set symmetry (0/1 no/yes), type (1 Gauss, 2 Lobatto) and geometry(0/1/2 planar/cylindrical/spherical), can be done in constructor also
quadrature(x,w,n) – calculate quadrature points and weights for n interior points
Acoeff(**A**,x,n) – calculate first derivative operator given x and n
Anonsym(**An,A**,x) – calculate first derivative operator (odd) given A and x
Bcoeff(**B,A**,x) – calculate second derivative operator given A and x
Ccoeff(**C,A**,x,w) – symmetric second derivative, given A, x and w
MassMat(**D**,x,w,n) – mass matrix given x, w and n

Lcoeff(Li,x,xi,nt) – Li are the Lagrange interpolating polynomials through x evaluated at xi, nt size(x), xi is scalar.

LcoeffN(**L**,x,xi,nt,ni) – L is a 2D array of the Lagrange interpolating polynomials through x evaluated at array of points xi, nt is size(x) and ni is size(xi)

Array.h, Array.cpp – All arrays in bold are type Array2D. A simple 2D array class. It also contains linear solver routines

Array() – default constructor, zero sized array

Array(n,m) – constructs n x m array

~Array() – destructor

void ArraySet(n,m) – resets to n x m array

void ArrayChk(const char *Aname) – for checking allocation status

int LUSolveA(double *b) - solves linear equations with rhs b

int LUFactr() – factors matrix into LU form

int LUSubst(double *b) – solves linear equations with factored matrix and rhs b

Note: see note above (Fortran) stating the shape of the various arrays.

Excel Dynamic Link Library (OCCdll.dll)

Test.xls – simple spreadsheet which calls the various functions

OCC_Setup(s,t,g) – set symmetry, type and geometry

OCC_Points(nt) – returns points given total number

OCC_Weights(Xc) – returns quadrature weights given points

OCC_Acoef(Xc) – first derivative operator

OCC_AcoefN(Xc) – first derivative operator for odd function

OCC_Bcoef(Xc) – 2^{nd} derivative operator

OCC_Ccoef(Xc) – 2^{nd} derivative operator symmetric form

OCC_Dcoef(Xc) – mass matrix

OCC_Linterp(x,Xc) – Lagrange interpolating polynomials evaluated at x

OCCdll.bas – visual basic code to interface with OCCdll.dll

This Visual basic code is needed to interface with the dynamic link library. This code is embedded in the Excel spreadsheets. It is already installed in test.xls. For a new spreadsheet, started from scratch, you will have to add this visual basic code.

Note: Excel has trouble keeping the spreadsheet up to date. Press ctrl-alt-F9 to force an update.