16.930 Advanced Topics in Numerical Methods for PDEs
Massachusetts Institute of Technology – Spring 2015

Version 1.0

Project 2 – The Master Element

Handed Out: March 3, 2015 *Due: March 20, 2015 (submitted on-line by 12:00PM)*

The objective of this project is to learn how to pre-compute the finite element shape functions and their derivatives for triangular elements and to exercise the process of numerical integration. In addition, you will become familiar with the `master` data structure in 2DG and the generation of meshes using Distmesh. A detailed description of the `master` structure can be found in the file 2DG-Structures.pdf. In this project, you will develop some tools for creating it.

2DG Directory structure

When you unpack the file 2DG2.zip, it will create a directory called **2DG.2** which contains the subdirectories **app**, **master**, **mesh** and **util**.

Getting Distmesh

You should visit **http://persson.berkeley.edu/distmesh/** and download a copy of the Delaunay mesh generator Distmesh. Place the distmesh directory as a subdirectory in the **2DG.2** directory.

Setting up 2DG and distmesh

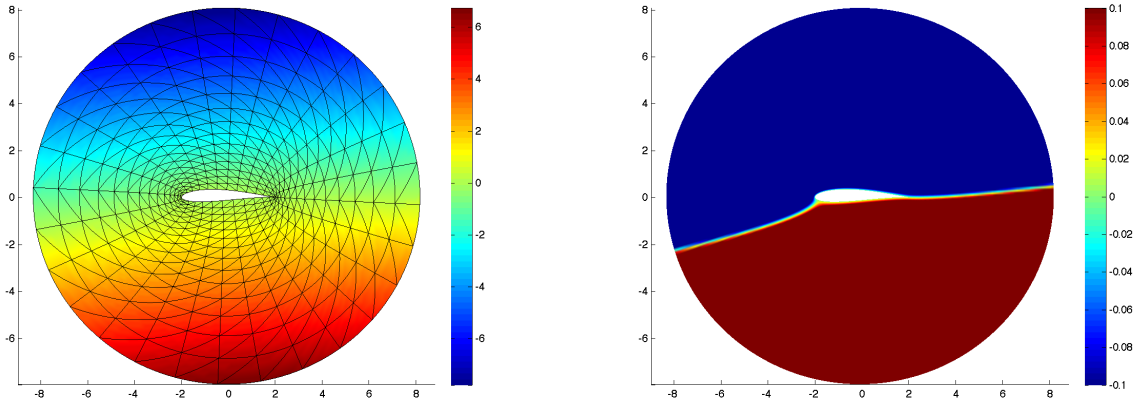
From within Matlab go to **2DG.2** and execute `setup.m`. This will add the paths of the **2DG.2** subdirectories to your current Matlab path.

Generating meshes

You will need to generate meshes for a circular domain and for an Karman-Trefftz airfoil. The **mesh** directory contains the scripts `mkmesh_circle.m` and `mkmesh_trefftz.m` which do exactly that. You can visualize the meshes using the scripts `meshplot.m` (only plots straight lines connecting the vertices) and `meshplot_curved.m` (if curved, this correctly plots the curved mesh but will be slower) in the **util** directory.

Your Tasks

- Since we want to be able to visualize solutions, you will need to write a simple utility to do that. In the subdirectory **util** you will find the prototype for the function `scaplot.m`. Once completed, this function will allow you to visualize a scalar field and set up a threshold to display the variable in a certain prescribed range. The program should also have an option to superimpose the mesh. Typical plots produced with `scaplot.m` will look like this:
- You are to implement the nodal basis for triangles as described by J.S. Hesthaven and T. Warburton, "Nodal High-Order Methods on Unstructured Grids", *Journal Computational Physics*, 181, 186-221 (2002). Two comments:
  - The master elements used in **2DG** are from $0 < \xi < 1$ for the 1D element, and for the 2D triangular master element the nodes are located at $(0, 0)$, $(1, 0)$, and $(0, 1)$.
  - The default node locations in the elements use equi-distribution, which is definitely not a good idea for high p. The location of the nodes in the master element are stored in



mesh.plocal. The script that sets there values is `uniformlocalpnts.m` in the **mesh** subdirectory. If you want, you could write your own function that uses spectral node locations and replaces **mesh.plocal**, and possibly **mesh.tlocal** if you change the ordering of the points in some way. For the problems we will look at in this course ($p \leq 3$), there won't be any issues with uniform points, but beyond that there can be issues. This change is not required. Only if you want to!

To complete the implementation, you will need to create the missing functions in the directory **master**. The functions `gaussquad1d.m` and `gaussquad2d.m` are provided and give you the integration points and weights for numerical integration in 1D and 2D, respectively. Note that the input to these functions is the parameter `pgauss` which is the degree of the complete polynomial that you want to integrate exactly. The function `jacobi.m` is also provided and returns the Jacobi polynomial $P_n^{(\alpha,\beta)}(x)$ which is calculated using the following recurrence relation

$$\begin{aligned} 2(n+1)(n+\alpha+\beta+1)(2n+\alpha+\beta)P_{n+1}^{(\alpha,\beta)}(x) \\ = \; [(2n+\alpha+\beta+1)(\alpha^2-\beta^2) + (2n+\alpha+\beta)_3 x]P_n^{(\alpha,\beta)}(x) \\ - 2(n+\alpha)(n+\beta)(2n+\alpha+\beta+2)P_{n-1}^{(\alpha,\beta)}(x) \end{aligned}$$

with the definition $(m)_3 = m(m+1)(m+2)$. The function `jacobi` is needed to form the Vandermonde matrices for the hierarchical functions in 1D and 2D. This is done in the functions `koornwinder1d.m`, `koornwinder2d.m`. In 1D the hierarchical functions are directly the Legendre polynomials which are precisely $P_n^{(0,0)}(x)$. In 2D, we use the Koornwinder functions which are obtained as tensor products of Jacobi polynomials. In particular, to calculate the n-th function, we proceed as follows. First determine the (i,j) indices corresponding to n in the Pascal triangle. e.g. $n=1$ corresponds to $(0,0)$, $n=2$ to $(1,0)$, $n=3$ to $(0,1)$, $n=4$ to $(2,0)$ etc.. The function $\boldsymbol{\psi}_n(\xi,\eta)$ for $\{(\xi,\eta) \mid \xi \geq 0, \eta \geq 0, 1-\xi\eta \geq 0\}$ is given by

$$\boldsymbol{\psi}_n(\xi,\eta) = P_i^{(0,0)}(r)\left(\frac{1-s}{2}\right)^i P_j^{(2i+1,0)}(s) \; ,$$

where

$$r = \frac{2\xi}{1-\eta} - 1, \qquad s = 2\eta - 1 \; .$$

Note that `koornwinder.m` is simply a driver for `koornwinder1d.m` and `koornwinder2d.m` (and all of these Koornwinder functions have also been provided).

Your task is to complete the functions `shape1d.m`, `shape2d.m` and `mkmaster.m`. Note that `mkmaster.m` is almost complete but some lines need to be filled in. Specifically,

- For `shape1d.m`, you need to implement the ability to evaluate Lagrangian (nodal) basis functions ϕ_i and their derivatives $\partial\phi_i/\partial\xi$ at the points given in the 1D array `PTS` where the Lagrangian basis has polynomials of order `PORDER` and the node locations given in the `PLOCAL`. For this 1D capability, you can choose to implement the function using either the barycentric 1D Lagrangian interpolation formula as described in lecture (see also the paper by Berrut & Trefethen, 2004), or utilize the Vandermonde matrix approach (though in 1D) as described by Hesthaven and Warburton. Note: the function must be written for arbitrary node locations (e.g. do not assume the nodes in `PLOCAL` are equi-distributed).
- For `shape2d.m`, you need to implement the ability to evaluate Lagrangian (nodal) basis functions ϕ_i and their derivatives $\partial\phi_i/\partial\xi_1$ and $\partial\phi_i/\partial\xi_2$ at the points given in the 2D array `PTS` where the Lagrangian basis has polynomials of order `PORDER` and the node locations given in the `PLOCAL`. For this 2D capability, you must utilize the Vandermonde matrix approach as described by Hesthaven and Warburton.
- For `mkmaster.m`, you need to implement the ability to evaluate the master element's mass matrix $\hat{M}_{ij} = \int_{\hat{\kappa}} \phi_i\phi_j \, d\xi$ and "convection" matrices $\hat{C}_{ijk} = \int_{\hat{\kappa}} \frac{\partial\phi_i}{\partial\xi_k}\phi_j \, d\xi$ for $k = 1$ and 2. These integrals should be calculated using quadrature in the master element at the points and weights stored in `master.gpts` and `master.gwgh`. Further, you need to evaluate the mass matrix for the 1D master element: $\hat{m}_{ij} = \int_0^1 \phi_i\phi_j \, d\xi$. These integrals should be calculated using quadrature in the master element at the points and weights stored in `master.gp1d` and `master.gw1d`.

  The results of these calculations are stored as follows:

  `master.ma(i,j)` $= \hat{M}_{ij}$ `master.conv(i,j,k)` $= \hat{C}_{ijk}$ `master.ma1d(i,j)` $= \hat{m}_{ij}$

  Comment: as mentioned in class, the master element mass and convection matrices will not be needed in the rest of this project. I will check them by executing your Matlab scripts and seeing that they return correct results. Also, as mentioned in class, $hatM$, $\hat{C}$, and $\hat{m}$ can only be used in elements which have constant Jacobians so that the corresponding matrices in the actual element will only differ by a constant multiplier (given by the actual elements constant Jacobian).

- In the directory **app** you will find a file called `areacircle.m`. The objective of the function is to calculate the area and perimeter of the unit circle, C, by first meshing it into a triangular mesh and then integrating the area and length of the circumference. The first output of the code is `area1` which is the area obtained by integrating over the elements

$$\texttt{area1} = \int_C dA \ .$$

The second output is the area calculated as a line integral and using the divergence theorem. If we consider the vector function $P = (x/2, y/2)$, then $\nabla \cdot P = 1$ and we have

$$\texttt{area2} = \int_C \nabla \cdot P \, dA = \int_{\partial C} P \cdot \hat{\boldsymbol{n}} \, ds \ .$$

The final output is the perimeter which is obtained as a line integral

$$\texttt{perim} = \int_{\partial C} ds \ .$$

Calculate your results for various discretizations sizes and polynomial orders. Compare to the exact answers.

- Also in the directory **app** you will find the file `potential_trefftz.m` which calculates the potential flow solution about a Trefftz airfoil. The input to `potential_trefftz.m` are the points where you want to know the solution, the magnitude of the free stream velocity, u_∞ (usually normalized to unity), and the angle of attack. The outputs are the streamfunction ψ, the velocity $\boldsymbol{v}$ and the circulation Γ. The velocity and circulation are the analytical results and only provided for comparison. Note that circulation and lift are related by the Kutta-Joukowski theorem $L' = -\rho V_\infty \Gamma$, in which $L' = 1/2\, \rho V_\infty^2 c\, C_L$ is the lift force per unit span for an airfoil of chord c. Thus lift coefficient and circulation are related by

$$C_L = -\frac{2\Gamma}{V_\infty c} \ .$$

Your task is to use the function `potential_trefftz.m` to evaluate the streamfunction ψ at the DG nodes. Once you know ψ you can calculate the velocity by differentiating your approximation: $v_x = \partial\psi/\partial y$, and $v_y = -\partial\psi/\partial x$. You should calculate the value of the velocity at the nodes. Note that if you originally interpolated ψ using a polynomial of order p, v_x and v_y will be now polynomials of order $p-1$ (also note that any polynomial of order $p-1$ can be represented within an approximating space of order p). Once you know the value of the velocity you should be able to compute the pressure using Bernoulli's formula and obtain a pressure coefficient which is simply $C_p = 1-(v_x^2+v_y^2)/V_\infty^2$ (note that our free-stream velocity V_∞ has unit magnitude). By integrating the C_p over the airfoil surface you can then compute the total force coefficient

$$\boldsymbol{C}_F = \frac{1}{c}\int_S C_p\, \hat{\boldsymbol{n}}\ ds$$

where S is the surface of the airfoil and c is the chord. The force coefficient can then be decomposed into lift and drag coefficients C_L and C_D by projecting along the free stream direction and in the direction perpendicular to it, respectively.

Project deliverables

- You should submit your **2DG** directory on the stellar website before coming to class on the due date.
- You should turn in class one table showing the area and perimeters computed for the unit circle using mesh sizes `siz = [0.4, 0.2, 0.1]` and polynomial orders $p = 1, 2$ and 3.
- You should consider the Trefftz airfoil with the default parameters `param = [0.1,0.05,1.98]` and write a function that calculates C_L and C_D versus the angle of attack α for $\alpha = -3 : 0.5 : 10$ degrees. Plot your results. You should consider the default values set for the Trefftz mesh (`m=15, n=30, porder=3`) but you are encouraged to try finer meshes. Plot also the C_p distribution for $\alpha = 10$ degrees using the `scaplot.m` function.