MA471
Lecture 7

Introduction to a basic finite element elliptic solver
Poisson’s Equation

- We wish to solve the following partial differential equation, with homogeneous boundary conditions, for $u$ in the two-dimensional subspace of the plane $\Omega$:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y) \text{ in } \Omega$$

with

$$u(x, y) = 0 \text{ for } (x, y) \in \partial \Omega$$
Sobolev Spaces of Functions

Define the first order Sobolev space as the set of functions which are $L^2$ integrable on $\Omega$ and whose derivatives are $L^2$ integrable on $\Omega$:

$$H^2(\Omega) = \left\{ u: \Omega \rightarrow \mathbb{R} \text{ such that } \int_{\Omega} u^2, \int_{\Omega} \left( \frac{\partial u}{\partial x} \right)^2, \int_{\Omega} \left( \frac{\partial u}{\partial y} \right)^2 < \infty \right\}$$

This will be the space of functions we are going to approximate for the solution of the pde.
Domain of the Laplace Operator

The Laplacian operator \( L = \left( \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \) is a linear unbounded operator in \( L^2(\Omega) \). Supplemented with the homogeneous Dirichlet boundary conditions its domain of definition is the dense subspace of

\[
D_B(L) = \left\{ v \in H^2(\Omega) : v(x, y) = 0 \forall (x, y) \in \partial \Omega \right\}
\]

Define the second order Sobolev space as the set of functions which are \( L_2 \) integrable on \( \Omega \) and whose derivatives are \( L_2 \) integrable on \( \Omega \):

\[
H^2(\Omega) = \left\{ u : \Omega \to \mathbb{R} \text{ such that } \int_\Omega u^2, \int_\Omega \left( \frac{\partial u}{\partial x} \right)^2, \int_\Omega \left( \frac{\partial u}{\partial y} \right)^2, \int_\Omega \left( \frac{\partial^2 u}{\partial x^2} \right)^2, \int_\Omega \left( \frac{\partial^2 u}{\partial x \partial y} \right)^2, \int_\Omega \left( \frac{\partial^2 u}{\partial x^2} \right)^2, < \infty \right\}
\]
Variational Form of Poisson’s Equation
(also known as the weak form)

Find $u \in D_B(\Omega)$ such that:

$$\int_{\Omega} v \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) = \int_{\Omega} v f \text{ for all } v \in H^1(\Omega)$$

and $u(x,y)=0$ for all $(x,y) \in \partial\Omega$

$v$ is known as the test function and $u$ is known as the trial function.
Since $v \in H^1(\Omega)$ and $u \in D_B(\Omega)$ we can integrate by parts to obtain:

Find $u \in D_B(\Omega)$ such that

$$\int_{\Omega} \frac{\partial v}{\partial x} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \frac{\partial u}{\partial y} = -\int_{\Omega} vf \text{ for all } v \in H_1(\Omega)$$

and $u(x,y)=0$ for all $(x,y) \in \partial \Omega$

However, we are not able to solve this for all functions in the infinite set of functions in the Sobolev space.
Approximation Space

• Since we are unable to represent all the functions in the infinite dimensional Sobolev spaces and subspaces we are going to use a subset of these functions.

• We will also restrict our search for $u$ to a subset of $H^1$ which satisfies the homogeneous Dirichlet boundary conditions.

• We will look for continuous solutions, locally represented by linear functions.
Plain-ish Speak

• We first break up Omega into a set of triangles.

• On each triangle we are going to represent H1 with a basis of linear polynomials in the x,y variables

• In fact we are going to think of each triangle as being the map from a reference triangle (also known as the master triangle).

• For instance the physical coordinates (x,y) in terms of the reference coordinates (r,s) are related by:

\[
\begin{pmatrix} x \\ y \end{pmatrix} = - \left( \frac{r + s}{2} \right) \begin{pmatrix} v_1^x \\ v_1^y \end{pmatrix} + \left( \frac{1 + r}{2} \right) \begin{pmatrix} v_2^x \\ v_2^y \end{pmatrix} + \left( \frac{1 + s}{2} \right) \begin{pmatrix} v_3^x \\ v_3^y \end{pmatrix}
\]
Square domain divided into a set of non-overlapping triangles
Reference Triangle

(r, s) are Cartesian coordinates for the reference triangle

\( r = -1 \)
\( s = 1 \)

\( r = 1 \)
\( s = -1 \)
Reference Triangle
Mapped To Physical Triangle

\[
\begin{pmatrix}
  x \\
  y
\end{pmatrix}
= -\left( \frac{r + s}{2} \right)
\begin{pmatrix}
  v_1^x \\
  v_1^y
\end{pmatrix}
+ \left( \frac{1 + r}{2} \right)
\begin{pmatrix}
  v_2^x \\
  v_2^y
\end{pmatrix}
+ \left( \frac{1 + s}{2} \right)
\begin{pmatrix}
  v_3^x \\
  v_3^y
\end{pmatrix}
\]
Basis for Test and Trial Spaces

• We construct a linear polynomial basis with respect to the reference triangle as the following three functions:

\[
\phi_1 = -\left(\frac{r + s}{2}\right), \phi_2 = \left(\frac{1 + r}{2}\right), \phi_3 = \left(\frac{1 + s}{2}\right)
\]

• Functions will be represented in each element k by:

\[
 f \left( x(r, s), y(r, s) \right) = \sum_{i=1}^{3} \phi_i(r, s) f_{ki}
\]

• In fact we will use a collocation representation so the \( f_{k1}, f_{k2}, f_{k3} \), are the values of the approximate field at the three corners of the k’th element.
Recall Variational Formulation

Find $u \in D_B(\Omega)$ such that

$$\int_{\Omega} \frac{\partial v}{\partial x} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \frac{\partial u}{\partial y} = -\int_{\Omega} vf \text{ for all } v \in H_1(\Omega)$$

and $u(x,y)=0$ for all $(x,y) \in \partial \Omega$

We replace this by the following:

Find constants $u^k_j$ such that:

$$\sum_{k=1}^{Ntri} \left( \int_{T^k} \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} + \frac{\partial \phi_i}{\partial y} \frac{\partial \phi_j}{\partial y} \right) u^k_j = -\sum_{k=1}^{Ntri} \left( \int_{T^k} \phi_i \phi_j \right) f^k_j$$
Notes on Differentiation

• For each element we know:

\[
\begin{pmatrix} x \\ y \end{pmatrix} = -\left( \frac{r+s}{2} \right) \begin{pmatrix} v_1^x \\ v_1^y \end{pmatrix} + \left( \frac{1+r}{2} \right) \begin{pmatrix} v_2^x \\ v_2^y \end{pmatrix} + \left( \frac{1+s}{2} \right) \begin{pmatrix} v_3^x \\ v_3^y \end{pmatrix}
\]

• So we can calculate:

\[
\frac{\partial x}{\partial r} = \left( \frac{v_2^x - v_1^x}{2} \right), \quad \frac{\partial x}{\partial s} = \left( \frac{v_3^x - v_1^x}{2} \right)
\]

\[
\frac{\partial y}{\partial r} = \left( \frac{v_2^y - v_1^y}{2} \right), \quad \frac{\partial y}{\partial s} = \left( \frac{v_3^y - v_1^y}{2} \right)
\]
Notes on Differentiation

• From which we can evaluate:

\[
\frac{\partial r}{\partial x} = \frac{1}{J} \left( \frac{v_3^y - v_1^y}{2} \right), \quad \frac{\partial r}{\partial y} = -\frac{1}{J} \left( \frac{v_3^x - v_1^x}{2} \right)
\]

\[
\frac{\partial s}{\partial x} = \frac{-1}{J} \left( \frac{v_2^y - v_1^y}{2} \right), \quad \frac{\partial s}{\partial y} = \left( \frac{v_2^x - v_1^x}{2} \right)
\]

\[
J = -\left( \frac{v_3^x - v_1^x}{2} \right) \left( \frac{v_2^y - v_1^y}{2} \right) + \left( \frac{v_2^x - v_1^x}{2} \right) \left( \frac{v_3^y - v_1^y}{2} \right)
\]

• So we can calculate:

\[
\frac{\partial}{\partial x} = \frac{\partial r}{\partial x} \frac{\partial}{\partial r} + \frac{\partial s}{\partial x} \frac{\partial}{\partial s}
\]

\[
\frac{\partial}{\partial y} = \frac{\partial r}{\partial y} \frac{\partial}{\partial r} + \frac{\partial s}{\partial y} \frac{\partial}{\partial s}
\]
Summary of Differentiation

• Given the physical coordinates of the three vertices of a triangle we can calculate the derivative of a function using the chain rule – whose coefficients are given by the previous formula.
The *Linear* Finite Element Method

- A basic variant of the *linear* finite element method is implemented in Matlab in 4 script files available from the class web site

  - umMESH.m
  - umStartUp.m
  - umMatrix.m
  - umSolve.m
Running The Finite Element Solver Inside Matlab

```matlab
>> umFileName = 'highres.neu';
>> umMESH
>> umStartUp
>> umMatrix
>> umSolve
```

![MATLAB Environment](image)
umMESH.m

- umMESH reads in a set of (umVertX,umVertY) coordinates for a set of nodes in the two-dimensional plane

- It also reads in a list of triples (elmttonode) which specify which three nodes lie in the list of umNel triangles

- It generates an elmttoelmt connectivity array, which represents the intersections of triangle faces.

- … and some other stuff
umStartUp.m

- Sets up the reference element information
- Builds the coordinates of nodes
- Calculates the coefficients used in the chain rule
%---------------------------------------------------------
umNpts = 3;
%---------------------------------------------------------
umR = [-1.0; 1.0; -1.0];
umS = [-1.0; -1.0; 1.0];
umDr = [[-0.5, 0.5, 0.0];[-0.5, 0.5, 0.0];[-0.5, 0.5, 0.0]];
umDs = [[-0.5, 0.0, 0.5];[-0.5, 0.0, 0.5];[-0.5, 0.0, 0.5]];

umMassMatrix = [[1./3.,1./6.,1./6.];[1./6.,1./3.,1./6.];[1./6.,1./6.,1./3.]];
%---------------------------------------------------------
% build coordinates of all the nodes
umX = zeros(numNpts, numNel);
umY = zeros(numNpts, numNel);

for thiselmt=1:numNel
    % note NO change of orientation
    va = elmttonode(thiselmt,1);
    vb = elmttonode(thiselmt,2);
    vc = elmttonode(thiselmt,3);
    umX(:,thiselmt) = (-0.5*(umR+umS)*umVertX(va)+0.5*(1+umR)*umVertX(vb)+0.5*(1+umS)*umVertX(vc));
umY(:,thiselmt) = (-0.5*(umR+umS)*umVertY(va)+0.5*(1+umR)*umVertY(vb)+0.5*(1+umS)*umVertY(vc));

end
%---------------------------------------------------------
% calculate geometric factors
xr = umDr*umX;
xs = umDs*umX;
yr = umDr*umY;
ys = umDs*umY;

jac = -xs(1,:).*yr(1,:)+xr(1,:).*ys(1,:);
rx = ys(1,:)./jac;
sx = -yr(1,:)./jac;
ry = -xs(1,:)./jac;
sy = xr(1,:)./jac;
%---------------------------------------------------------
umMatrix.m

• Assembles the matrix mat which represents the left hand side premultiplier of the unknown u coefficients

Find constants $u_j^k$ such that:

$$\sum_{k=1}^{k=Ntri} \left( \int_{T^k} \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} + \frac{\partial \phi_i}{\partial y} \frac{\partial \phi_j}{\partial y} \right) \phi_i \phi_j f_j^k = - \sum_{k=1}^{k=Ntri} \left( \int_{T^k} \phi_i \phi_j \right) f_j^k$$

• Which will turn up in: mat\*u = rhs
umMatrix.m:

Builds the global bilinear matrix

```matlab
umRR = transpose(umDr)*(umMassMatrix*(umDr));
umRS = transpose(umDr)*(umMassMatrix*(umDs));
umSR = transpose(umDs)*(umMassMatrix*(umDr));
umSS = transpose(umDs)*(umMassMatrix*(umDs));
mat = spalloc(umNnodes,umNnodes, 9*umNel);
for thiselmt=1:umNel
    ljac = jac(thiselmt);
    lrx = rx(thiselmt);
    lsx = sx(thiselmt);
    lry = ry(thiselmt);
    lsy = sy(thiselmt);
    locmat = (lrx*lrx+lry*lry)*umRR;
    locmat = locmat+(lrx*lsx+lry*lsy)*umRS;
    locmat = locmat+(lsx*lrx+lsy*lry)*umSR;
    locmat = locmat+(lsx*lsx+lsy*lsy)*umSS;
    locmat = locmat*ljac;
    nodeid1 = umElmtToGnode(thiselmt,1);
    nodeid2 = umElmtToGnode(thiselmt,2);
    nodeid3 = umElmtToGnode(thiselmt,3);
    mat(nodeid1,nodeid1) = mat(nodeid1,nodeid1)+locmat(1,1);
    mat(nodeid1,nodeid2) = mat(nodeid1,nodeid2)+locmat(1,2);
    mat(nodeid1,nodeid3) = mat(nodeid1,nodeid3)+locmat(1,3);
    mat(nodeid2,nodeid1) = mat(nodeid2,nodeid1)+locmat(2,1);
    mat(nodeid2,nodeid2) = mat(nodeid2,nodeid2)+locmat(2,2);
    mat(nodeid2,nodeid3) = mat(nodeid2,nodeid3)+locmat(2,3);
    mat(nodeid3,nodeid1) = mat(nodeid3,nodeid1)+locmat(3,1);
    mat(nodeid3,nodeid2) = mat(nodeid3,nodeid2)+locmat(3,2);
    mat(nodeid3,nodeid3) = mat(nodeid3,nodeid3)+locmat(3,3);
end
mat = mat(1:umNUknown, 1:umNUknown);
```
umSolve.m

- Builds the right hand side and inverts the linear system of equations.
umSolve.m:

1) Assembles right hand side

2) Solves system

3) Scatters solution back to local elements

4) Plots results
Group Project #2

• A team leader should be nominated to construct a driving routine and create a set of appropriate structs for implementing the project.

• Each member should take responsibility for implementing one of the .m scripts

• There are a couple of routines we need for implementing the code. I have included them in the release in umLINALG.c

• These include matrix multiplication and inverting a matrix