University of California, Berkeley
CEE C133/ME C180, Engineering Analysis Using the Finite Element Method
Spring 2010
Instructor: S. Govindjee
GSI: N. Hodge

Lab 7: 2D Finite Elements

Your assignment this week will be another very non-trivial generalization of your code. I will try to explain what you are to do as clearly as possible, and give you some hints, as appropriate.

Because this will be a fair bit of work (probably even more than Lab 6), the due date for this lab will be after spring break, either 29 or 30 April, 2010.

Your assignment is as follows:

- To make it easier for me to run your codes, you are to take all of your input from my input deck, just like the previous lab. By "input", I mean all of the data for the PDE (i.e., $\mathbf{k}(\mathbf{x})$, $r(\mathbf{x})$, BCs), as well as the FE parameters (number of elements, element type, etc.). I have posted it on bspace. It is fairly well documented, but please ask if anything is unclear.

  The deck is just a matlab function that you can (i.e. *will*) call, and get back (hopefully) all of the data needed to run your code. Please, do not have your code take any command line input; get everything from the input deck.

- Solve a steady state heat conduction problem in 2D. Given the strong form

$$\begin{aligned} \operatorname{div}(\mathbf{k}\,\operatorname{grad} T) + r &= 0, \text{ in } \Omega \\ T &= \bar{T}, \text{ on } \Gamma_u, \\ \mathbf{q}\cdot\mathbf{n} &= \bar{q}, \text{ on } \Gamma_q, \end{aligned}$$

  the exact weak form follows:

$$0 = \int_\Omega \left[\operatorname{div}(\mathbf{k}(\mathbf{x})\,\operatorname{grad}(T)w) - \mathbf{k}(\mathbf{x})\,\operatorname{grad}(T)\cdot\operatorname{grad}(w) + wr\right] dv,$$

$$\Rightarrow$$

$$\begin{aligned} \int_\Omega -\mathbf{k}(\mathbf{x})\,\operatorname{grad}(T)\cdot\operatorname{grad}(w)\,dv &= -\int_\Omega wr\,dv - \int_\Omega \operatorname{div}(\mathbf{k}(\mathbf{x})\,\operatorname{grad}(T)w)\,dv, \\ \int_\Omega \operatorname{grad}(w)\cdot\mathbf{k}(\mathbf{x})\,\operatorname{grad}(T)\,dv &= \int_\Omega wr\,dv - \int_{\partial\Omega} w\mathbf{q}\cdot\mathbf{n}\,da, \\ &= \int_\Omega wr\,dv - \int_{\Gamma_q} w\bar{q}\,da. \end{aligned}$$

Note that, in the previous expressions, $\mathbf{k}$ can be both isotropic and a function of position. Also, recall its approximation:

$$\int_\Omega (\mathbf{B}\hat{\mathbf{w}}) \cdot \mathbf{k}(\mathbf{x}) \left(\mathbf{B}\hat{\mathbf{T}}\right) \, dv = \int_\Omega (\mathbf{N}\hat{\mathbf{w}}) \, r \, dv - \int_{\Gamma_q} \bar{q} \left(\mathbf{N}\hat{\mathbf{w}}\right) \, da,$$

$$\hat{\mathbf{w}}^T \left(\left[\int_\Omega \mathbf{B}^T \mathbf{k}(\mathbf{x}) \mathbf{B} \, dv \right] \hat{\mathbf{T}} \right) = \hat{\mathbf{w}}^T \left(\int_\Omega \mathbf{N}^T r \, dv - \int_{\Gamma_q} \bar{q} \mathbf{N}^T \, da \right),$$

where

$$\mathbf{k} = \begin{bmatrix} k_{1,1}(\mathbf{x}) & k_{1,2}(\mathbf{x}) \\ k_{2,1}(\mathbf{x}) & k_{2,2}(\mathbf{x}) \end{bmatrix}.$$

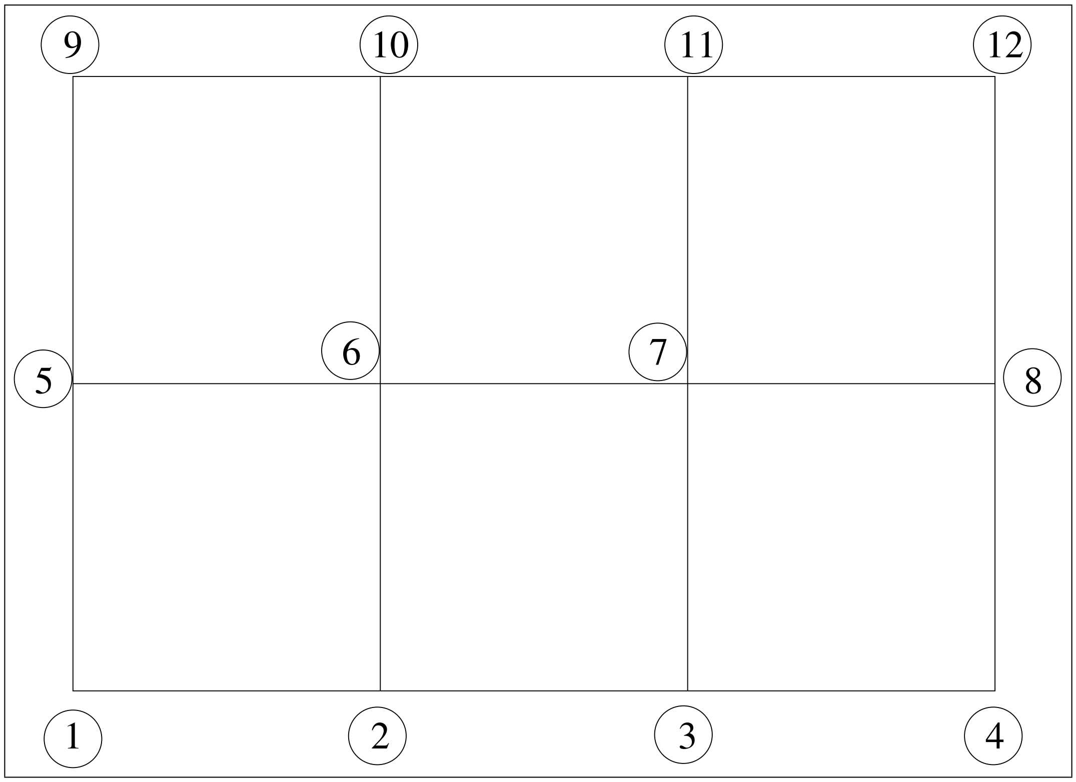
Thus,

$$\left[\int_\Omega \mathbf{B}^T \mathbf{k}(\mathbf{x}) \mathbf{B} \, dv \right] \hat{\mathbf{T}} = \int_\Omega \mathbf{N}^T r \, dv - \int_{\Gamma_q} \bar{q} \mathbf{N}^T \, da,$$

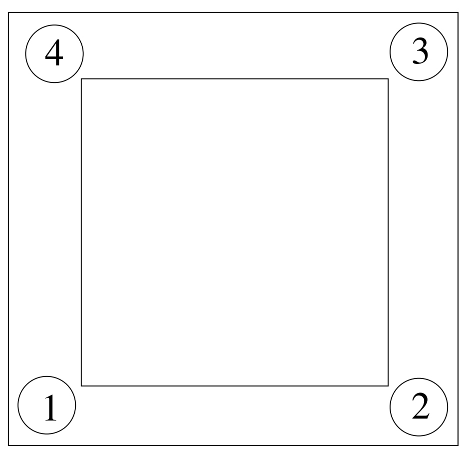
or

$$\mathbf{K}\hat{\mathbf{T}} = \mathbf{F}.$$

- You can assume that the domain is always a rectangle in $\mathbb{R}^2$, such that $x_1 \in \{x : 0 \leq x \leq a\}$, and $x_2 \in \{x : 0 \leq x \leq b\}$.
- You can assume that the global node numbering will always be monotonically increasing, *first* in the x_1 direction, and then in the x_2 direction. An illustration follows, for the case of $nel_1 = 3$ and $nel_2 = 2$.



Note that the *local* node numbering for 4 node quads is typically given as follows:



- Your elements only need to be linear quadrilaterals, formulated in isoparametric space.
- In the 1D case, many of you used the chain rule to claim that

$$\begin{aligned}\frac{\partial N_A}{\partial x}\frac{\partial N_B}{\partial x}\frac{\partial x}{\partial \xi} &= \left(\frac{\partial N_A}{\partial \xi}\frac{\partial \xi}{\partial x}\right)\left(\frac{\partial N_B}{\partial \xi}\frac{\partial \xi}{\partial x}\right)\frac{\partial x}{\partial \xi},\\ &= \frac{\partial N_A}{\partial \xi}\frac{\partial N_B}{\partial \xi}\frac{\partial \xi}{\partial x},\end{aligned}$$

which is ok for the 1D case, but won't work for 2D or 3D. So, we have to calculate $\frac{\partial N}{\partial \mathbf{x}}$ properly. First, note that we represent the gradient of a scalar T in 2D as follows:

$$\text{grad}(T) \approx \mathbf{B}\hat{\mathbf{T}},$$

where

$$\mathbf{B} = \begin{bmatrix} N_{1,1} & N_{2,1} & \dots & N_{nen,1} \\ N_{1,2} & N_{2,2} & \dots & N_{nen,2} \end{bmatrix}.$$

So, if we can calculate $\frac{\partial N_A}{\partial x}$ and $\frac{\partial N_A}{\partial y}$ for all A, then we are good. So, how to do this? We don't have the shape functions written in physical space anymore, so we can't take those derivatives directly, but we also know that there is some relation between points on the master element, and points in the physical element, and we exploit that fact via a 2D chain rule:

$$\begin{aligned} \begin{bmatrix} \dfrac{\partial N_A}{\partial \xi} \\ \dfrac{\partial N_A}{\partial \eta} \end{bmatrix} &= \begin{bmatrix} \dfrac{\partial N_A}{\partial x}\dfrac{\partial x}{\partial \xi} + \dfrac{\partial N_A}{\partial y}\dfrac{\partial y}{\partial \xi} \\ \dfrac{\partial N_A}{\partial x}\dfrac{\partial x}{\partial \eta} + \dfrac{\partial N_A}{\partial y}\dfrac{\partial y}{\partial \eta} \end{bmatrix}, \\ &= \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} \end{bmatrix} \begin{bmatrix} \dfrac{\partial N_A}{\partial x} \\ \dfrac{\partial N_A}{\partial y} \end{bmatrix}. \end{aligned} \tag{1}$$

So, the term on the left hand side is easy to calculate, since the shape functions are formulated directly in isoparametric coordinates. The matrix on the right hand side is also easy to calculate, for the same reason. The vector on the right hand side is what we want to solve for, so, solve the system of equations.

- Your code should perform its integration using Gaussian quadrature. Note that the matrix on the right hand side of (1) above is what we call the "Jacobian" matrix for the element, and its determinant, $J = \det([J])$, is the scale factor we use to map the integrals to the parent space. Thus, the quadrature (of the stiffness, for example) can be calculated as follows:

$$\begin{aligned} \int_{\Omega^e} \mathbf{B}^T \mathbf{k}(\mathbf{x})\mathbf{B}\, dv &= \int_{-1}^{1}\int_{-1}^{1} \mathbf{B}(\xi,\eta)^T \mathbf{k}(\mathbf{x}(\boldsymbol{\xi}))\mathbf{B}(\xi,\eta) J(\xi,\eta)\, d\xi\, d\eta, \\ &= \int_{-1}^{1}\int_{-1}^{1} g(\xi,\eta)J(\xi,\eta)\, d\xi\, d\eta, \\ &\approx \int_{-1}^{1} \left(\sum_{i=1}^{nip} g(\xi_i,\eta)J(\xi_i,\eta)w_i \right) d\eta, \\ &\approx \sum_{j=1}^{nip} w_j \left(\sum_{i=1}^{nip} g(\xi_i,\eta_j)J(\xi_i,\eta_j)w_i \right), \\ &= \sum_{i=1}^{nip}\sum_{j=1}^{nip} g(\xi_i,\eta_j)J(\xi_i,\eta_j)w_i w_j. \end{aligned}$$

- You only need to implement Dirichlet boundary conditions. Implementation of Neumann boundary conditions would be a bit complicated, and I figure that, ultimately, you already know how to do integrations in 1D . . .

  Per the input deck, you can assume that each Dirichlet BC covers a whole edge, and the edges are numbered, as follows:

edge number	description
1	$x = 0$
2	$x = a$
3	$y = 0$
4	$y = b$

- I have posted another template file for this assignment, with a few code fragments, including code to deal with the Dirichlet BCs for this problem, and some data structures to help you think about how to organize the calculation of the interpolation functions. Feel free to use them verbatim, hack them as desired, or make up your own from scratch.

You need to turn the following in to me:

- A convergence plot of your code versus the exact solution, in the l^2 norm, for the following problem data:

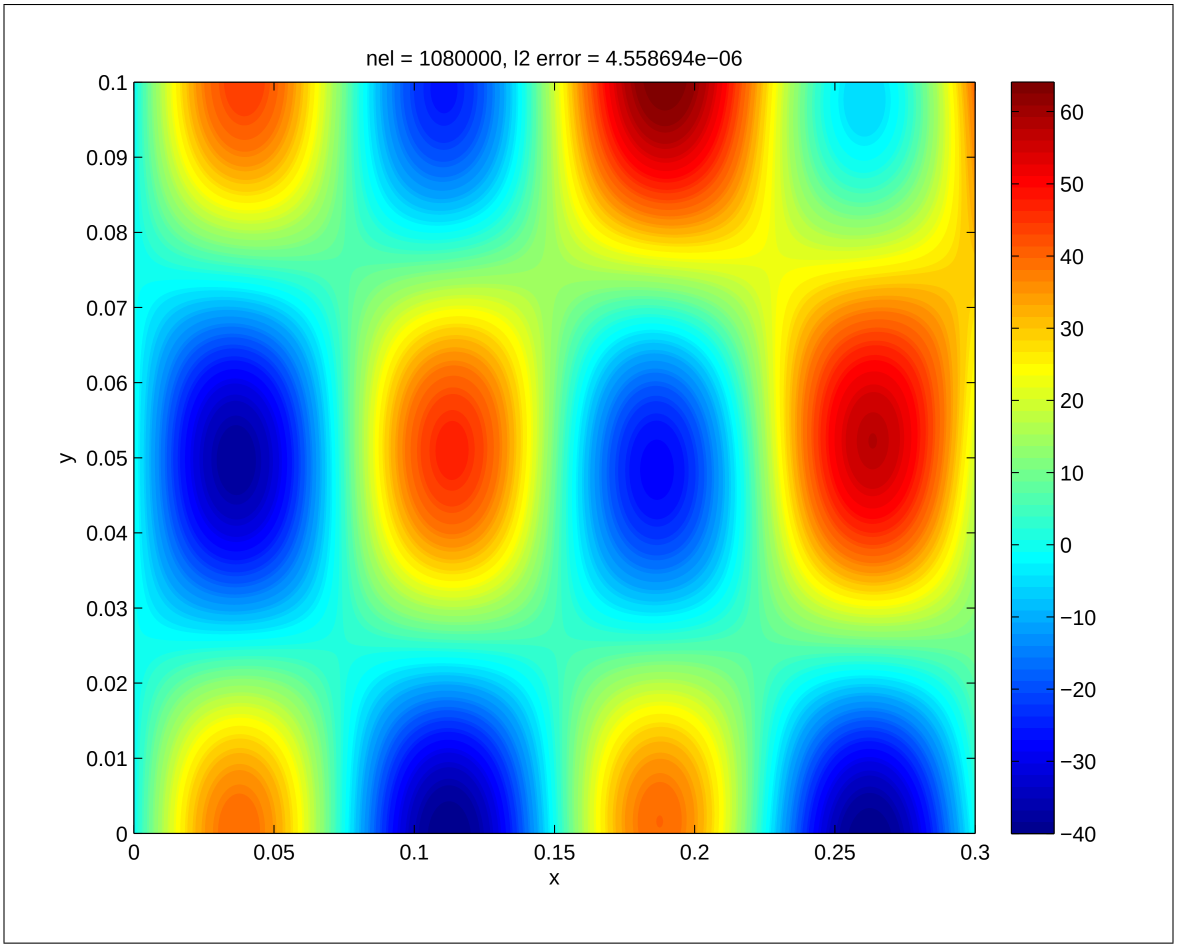
$$\begin{aligned}
\mathbf{k} &= k\mathbf{I}, \\
r(x,y) &= kT_1 4\pi^2 \frac{4b^2 + a^2}{a^2 b^2} \sin\left(4\pi\frac{x}{a}\right)\cos\left(2\pi\frac{y}{b}\right), \\
T(0,y) &= 0, \\
T(a,y) &= T_2 \frac{y}{b}, \\
T(x,0) &= T_1 \sin\left(4\pi\frac{x}{a}\right), \\
T(x,b) &= T_1 \sin\left(4\pi\frac{x}{a}\right) + T_2\frac{x}{a},
\end{aligned}$$

  where $\mathbf{I}$ is the 2D identity matrix, $a = 30cm$, $b = 10cm$, $k = 109\frac{W}{m-K}$,$T_1 = 40C$, and $T_2 = 40C$. The exact solution for this problem is

$$T(x,y) = T_1 \sin\left(4\pi\frac{x}{a}\right)\cos\left(2\pi\frac{y}{b}\right) + T_2\frac{xy}{ab}.$$

  Plot ten different pairs of (nel_1, nel_2), ensuring that your elements are always square, and such that your last run has an l^2 error less than 1×10^{-5}. Plot your error versus $\frac{1}{h^e}$. Comment on how many linear 4-node quads your code took to meet the stated tolerance.

- A plot of the output from your final run. It should look something like the following:



- A plot of the error $|u_{exact} - u_{FE}|$ as a function of x and y.

  Please give these to me on paper (not via email).

- Email me your code, which I will run with an input deck that is different from the one I posted.