University of California, Berkeley
CEE C133/ME C180, Engineering Analysis Using the Finite Element Method
Spring 2009
Instructor: S. Govindjee
GSI: N. Hodge

Lab 7

Your assignment this week will be a 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, and because you have a midterm this week, the due date for this lab will be Wednesday, 18-Mar-2009, at 5PM.

The following description is split into several tasks that I strongly suggest you implement *one at a time*, and ensure that each is working before going to the next. Also, I have described the items in a very particular order, and that is the order that I would suggest you do them in. It is (nothing more than) my opinion that, if you implement them in the order presented, your life will be a bit easier. 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. By "input", I mean all of the data for the PDE (i.e., $A(x)$, $E(x)$, $b(x)$, 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.

  The etymology of the work "deck" is quite interesting:

  http://en.wikipedia.org/wiki/Punched_card

  http://en.wikipedia.org/wiki/Computer_programming_in_the_punch_card_era

  Many people called these "decks of cards", or just "decks". I have heard stories from people about spending their whole week getting a deck ready to run, and then tripping right in front of the computer, only to have their cards go spilling all over the floor . . . I have also seen a box containing a deck, once.

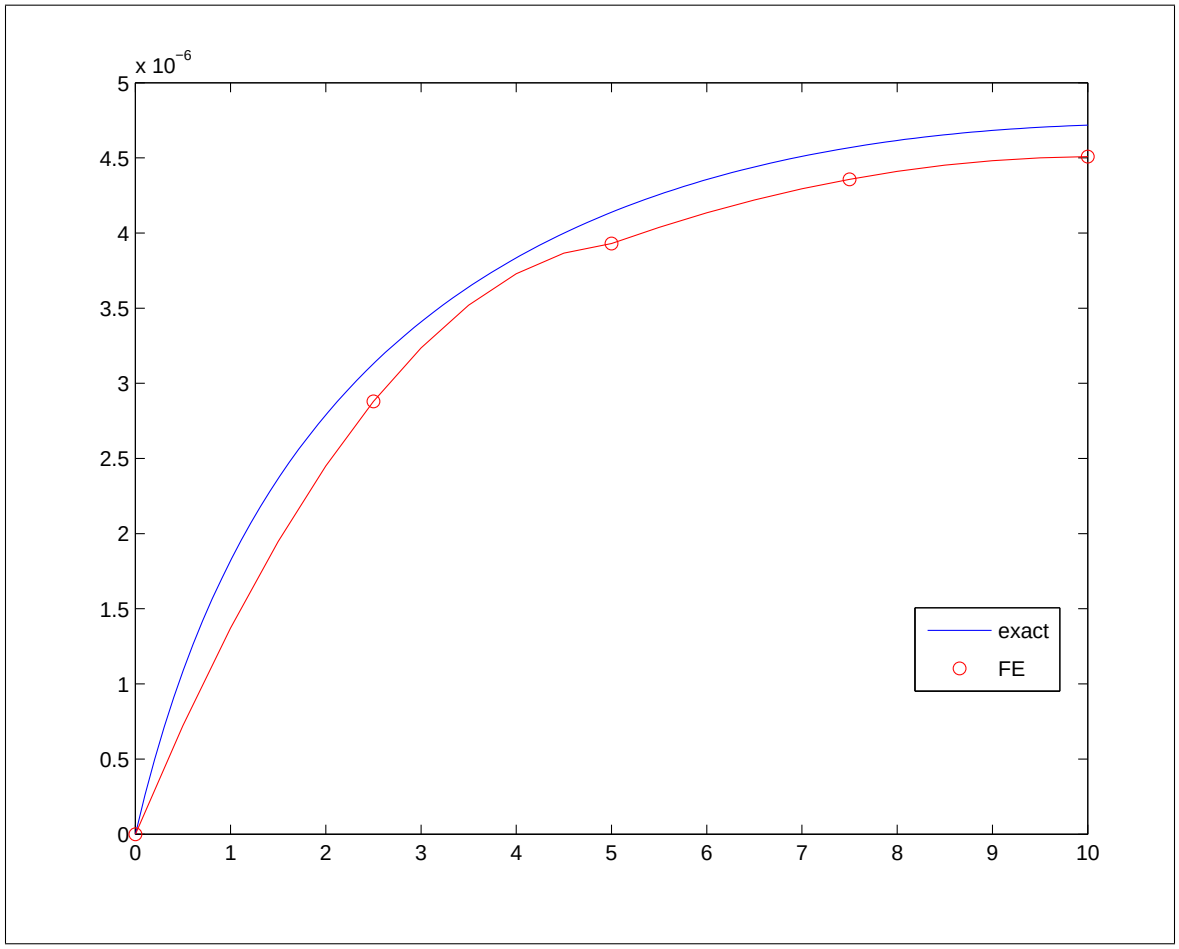
  Per my usage, you can rightfully infer that in some parts of the finite element community, the term "input deck" persists.

- Implement Guassian quadrature. Recall that I have posted a matlab file with Gauss points and weights for various orders on bspace. Also, note that this will require the implementation of an additional loop (for most of you) inside the element loop, to perform the summation over the Gauss points.

- Move your implementation of the interpolation functions from the physical space to the parametric space.

- Implement quadratic elements. Your code should be able to swtich between linear and quadratic elements by changing the value "eltype" in the input deck.

  Note that the curvature of higher order elements means that when you are plotting them, you can't just plot straight lines between the nodal values of the solution. The number of plot points in each element should be controlled by the parameter "nelpp" in the input deck. To illustrate what I mean, consider the following plot, of two quadratic elements.



You need to turn the following in to me:

- Convergence plots of quadratic elements versus linear elements. As a convergence measure, use the absolute value of the FE solution versus the exact solution at $x = L$, for the problem from Lab06, the exact solution of which is what is in my sample input deck on bspace. Plot this measure versus $\frac{1}{h^e}$ (i.e., the reciprocal of the element length) for $nel \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20\}$ for each element type, and comment.

  Please give this 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.