Department of Mechanical Engineering
University of Bath

ME40064: System Modelling & Simulation
ME50344: Engineering Systems Simulation

Extra information: implementing quadratic basis functions

Part A: Representing solution variable using QBF

Implementing quadratic basis functions to represent the solution variable is the easiest way to get started with them. The main changes you need to make to your code to accommodate this are:

1. Increase size of global vector to 2*(Number of elements) + 1 and global matrix to (2*(Number of elements) + 1, 2*(Number of elements) + 1)
2. Local element matrix functions must now return a 3 x 3 element matrix.
3. Ensure overlapping node structure is represented correctly in your new global assembly code
4. Use an appropriate N for your Gaussian quadrature scheme otherwise there will be an error in your integrals. Note for transient problems and/or reaction terms this will be N >= 3.
5. You will need additional functions to evaluate the quadratic basis functions and their gradients at a specific Gauss point.

Part B: Representing material parameters using QBF

To use QBF to represent a material parameter or source term on the quadratic basis function requires some extra work dealing with the mesh structure in order to represent these additional nodes and information. Similar ideas are relevant for using the discontinuous constant basis function for this purpose too. Here are some pointers:

1. Note that the underlying mesh has not changed. It has the same structure & number of elements as before. Therefore the Jacobian for each element is also the same as before. Effectively, we are still discretising our space/coordinate variable *x* using the linear basis functions.
2. Within a given element it is possible to have multiple variables represented, each on a different type of basis function. The distinction here between the nodes that construct the element,

and the nodes that represent the solution variable and material parameters is now more important, but also clearer to see.

3. This distinction means that you can keep the same mesh.elem(i) variable as before in the data structure and simply attach additional member variables to the element.

   For example, a variable such as mesh.elem(i).nq(1:3) could store the global node IDs for the quadratic basis nodes. Similar variables could be used to store the nodal coefficients for the material parameter in each element. The size of the array can be used to determine which set of basis functions to use when interpolating the variable.

4. An alternative approach is to define a vector containing the x coordinate of the quadratic nodes, and then compute a vector that stores the global nodal values for the parameter of interest. These values can be set as a function of x and the resulting vector can be attached directly to the mesh variable, i.e. mesh.quadDc.

   This is same idea presented in Lecture 10 when setting: mesh.Dcvec = 1 + mesh.nvec . Therefore to access the local nodal values for a particular element using this approach, extract the appropriate global node IDs from the global node mapping array described above (and as shown in Tutorial 5: Part B Example Solutions) and use these values as your indices into the material parameter vector.