I. M-file Heat2d.m (MAIN)

II. M-file Include_flags.m

A. Includes all global variables

1. **global** ndof nnp nel nen nsd neq ngp nee neq
2. **global** nd e_bc s P D
3. **global** LM ID IEN flags n_bc
4. **global** x y nbe
5. **global** compute_flux plot_mesh plot_temp plot_flux plot_nod

III. function [K,f,d] = preprocessor

A. M-file include_flags;

B. M-file input_file_16ele

1. Material Properties
   a) *Thermal conductivity, conductivity matrix*
2. Mesh Specifications
   a) *# of dimensions,# of nodes,# of elements,# element nodes,# of dof per node,# of Gauss points in each direction*
3. Essential Boundary Conditions
   a) *# of node on essential BC*
4. Natural Boundary Conditions (on edges of the NBCs)
   a) *Node 1 & flux at node 1*
   b) *Node 2 & flux at node 2*
   c) *# of edges on the NBCs*
5. Plot flags set to "YES" or "NO"
   a) *Flux, Mesh, NOD, temperature*
6. function **mesh2d**
   a) *define x coordinates*
   b) *define y coordinates*

c) generate connectivity array IEN

*d) **function** plotmesh*

(1) plot mesh and natural boundary

(2) OUTPUT: Mesh Parameters, # of Elements, # of Nodes, # of Equations

C. function d = setup_ID_LM(d)

1. IF: node on essential BC

a) Increment

b) Place the essential BCs nodes first

c) Store Essential BC in the reordered form (d_bar)

2. ELSE: Continue to next node

3. Create the LM matrix

IV. function [ke, fe] = heat2Delem(e)

A. M-file include_flags

B. Initialize element conductance matrix

C. Initialize element nodal source vector

D. Get coordinates of element nodes

1. function [w,gp] = gauss(ngp)

a) Obtain the gauss points in the parent element domain [-1, 1]

b) Obtain the gauss points in the parent element domain corresponding weights

E. function N = NmatHeat2D(eta,psi)

1. Shape function

F. function [B, detJ] = BmatHeat2D(eta,psi,C)

1. Calculate the grad of N

2. Obtain the Jocobian Matrix

3. Compute the B matrix

G. compute element conductance matrix (ke)

H. compute nodal source vector (fe)

V. **function** [K,f] = assembly(K,f,e,ke,fe)

- A. **M-file** include_flags
- B. Assemble nodal force vector (f)
- C. Assemble stiffness matrix (K)

VI. **function** f = src_and_flux(f)

- A. Assign point sources to the global flux vector
- B. Compute the nodal boundary flux vector
  1. Get the Gauss points and weights
  2. integrate along the edge
  3. 1D shape functions in the parent domain
  4. interpolate flux using shape functions
  5. nodal flux
  6. assemble the nodal flux vector

VII. **function** [d,f_E] = solvedr(K,f,d)

- A. Parition and solve the system of equations
- B. M-file include_flags
- C. partition the matrix K
  1. Extract K_E matrix
  2. Extract K_F matrix
  3. Extract K_EF matrix
- D. partition the vector f
  1. Extract f_F vector
- E. partition the d
  1. Extract d_E vector
- F. solve for d_F
- G. Reconstruct the global displacement vector, $\mathbf{d = [d\_E; d\_F]^T}$
- H. DISPLAY: d,f_E

VIII. function postprocessor(d)

A. plot temperature and flux

B. M-file include_flags

C. plot the temperature field(If plot_temp == 'yes')

D. For loop

1. compute temperature distribution at Gauss points

E. Compute flux at gauss points (If compute_flux == 'yes')

1. DISPLAY: the heat flux at the Gauss points