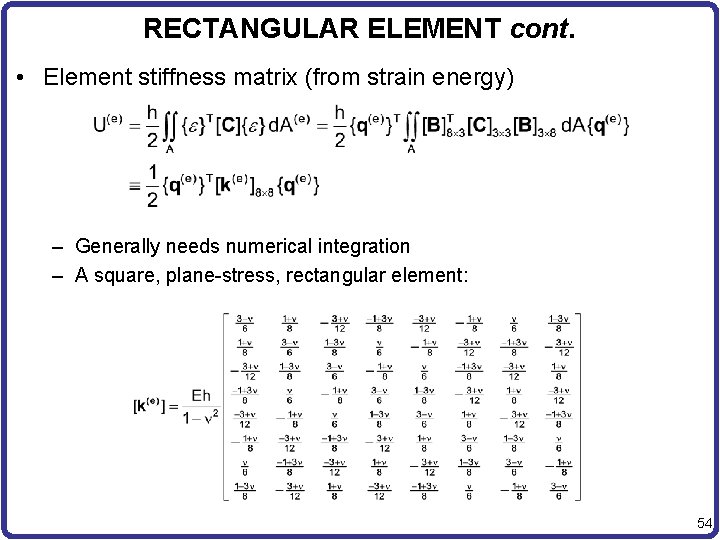
top88.m

|  |  |  |
| --- | --- | --- |
| Lines | Source Code | Description |
| 2 | function top88(nelx,nely,volfrac,penal,rmin,ft) | Main function call,  Inputs: nelx – no. of elements in x direction  nely – no. of elements in y direction  volfrac – volume fraction  penal – penalisation factor  rmin – minimum filter radius  ft – filtering mode (1: Sensitivity filtering, 2: Density Filtering)  Notes/Annotations:  **[**a **x** b**]** – denoting dimension of array/matrix  dofs – degrees of freedom  ndim – no. of dimensions  nlocnod – no. of local nodes on a single element  nlocdofs – no. of local dofs = ndim\*nlocnod |
| 4-6 | E0 = 1;  Emin = 1e-9;  nu = 0.3; | Setting material properties:  E0 – Nominal Stiffness  Emin – Stiffness Lower Bound  nu – Poisson’s ratio |
| 8-11 | A11 = [12 3 -6 -3; 3 12 3 0; -6 3 12 -3; -3 0 -3 12];  A12 = [-6 -3 0 3; -3 -6 -3 -6; 0 -3 -6 3; 3 -6 3 -6];  B11 = [-4 3 -2 9; 3 -4 -9 4; -2 -9 -4 -3; 9 4 -3 -4];  B12 = [ 2 -3 4 -9; -3 2 9 -2; 4 9 2 3; -9 -2 3 2]; | Constructing components for local stiffness matrix for 2D 4-noded bilinear element for plane stress |
| 12 | KE = 1/(1-nu^2)/24\*([A11 A12;A12' A11]+nu\*[B11 B12;B12' B11]); | Assembly of local stiffness matrix  **[**nlocdofs **x** nlocdofs**]** |
| 13 | nodenrs = reshape(1:(1+nelx)\*(1+nely),1+nely,1+nelx); | Create global node numbers matrix (mapping the Cartesian mesh as a 2d array of global node numberings), convention: top down, starting from leftmost column  **[**nely+1 **x** nelx+1] |
| 14 | edofVec = reshape(2\*nodenrs(1:end-1,1:end-1)+1,nelx\*nely,1); | Create 1D array of leading dofs for each element in edofMat (x-direction dof for the bottom left node for each element)  **[**nelx\*nely **x** 1**]** |
| 15 | edofMat = repmat(edofVec,1,8)+repmat([0 1 2\*nely+[2 3 0 1] -2 -1],nelx\*nely,1); | Create element dof matrix, where each row notates the global dof number for each element with convention: starting from the bottom left x-direction dof, then y-direction dof, going anticlockwise for each element, (steering matrix)  **[**nelx\*nely **x** ndim\*nlocdofs**]** |
| 16-17 | iK = reshape(kron(edofMat,ones(8,1))',64\*nelx\*nely,1);  jK = reshape(kron(edofMat,ones(1,8))',64\*nelx\*nely,1); | Preparing index vectors for sparse functions |
| 19 | F = sparse(2,1,-1,2\*(nely+1)\*(nelx+1),1); | Applying load (traction) of -1 at global dof 2 (y-direction for top left node of elemement 1) |
| 20 | U = zeros(2\*(nely+1)\*(nelx+1),1); | Initial Displacement 1D Array (filled with 0s) |
| 21 | fixeddofs = union([1:2:2\*(nely+1)],[2\*(nelx+1)\*(nely+1)]); | 1D array of fixed dofs, (For MBB beam, fixed x-displacements for left side nodes and fixed y-displacement for bottom right corner node) |
| 22 | alldofs = [1:2\*(nely+1)\*(nelx+1)]; | 1D array of all global dofs |
| 23 | freedofs = setdiff(alldofs,fixeddofs); | 1D array of remaining free dofs |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |



<https://slidetodoc.com/chap-6-finite-elements-for-plane-solids-finite-2/>