15 - finite element method

finite element method - volume growth - implementation

THE FOUR STAGES OF DATA LOSS

1. START TO THINK
2. START TO THINK
3. END TO THINK
4. END TO THINK

15 - finite element method

finite element method - volume growth - implementation

integration point based

- loop over all time steps
  - global newton iteration
    - loop over all elements
      - loop over all quadrature points
        - local newton iteration to determine $\theta_{n+1}$
        - determine element residual & partial derivative
        - determine global residual and iterational matrix
        - determine $\varphi_{n+1}$
        - determine state of biological equilibrium

staggered solution

finite element method

integration point based

- discrete residual
  - check in matlab!

\[ R_j^f = A_{\text{el}} \int_{B_j^f} \nabla u_j^f \cdot P_{n+1} \, dV \]

- residual of mechanical equilibrium/balance of momentum

staggered solution

finite element method
integration point based

- stiffness matrix / iteration matrix
  \[ \mathbf{K}^{ij} = \frac{\partial R_i}{\partial x_j} = \mathbf{A}^{ij} \int_N \nabla N_i \cdot \mathbf{D} \cdot \nabla N_j \, dV \]

- linearization of residual w.r.t. nodal dofs
  \( \mathbf{K}^{ij} \mathbf{u}^{(n)} = \mathbf{F}^{(n)} - \mathbf{K}^{ij} \mathbf{u}^{(n-1)} \)

finite element method

tetra_3.m

- constitutive equations - given \( \mathbf{F} \) calculate \( \mathbf{P} \)
  \[ \mathbf{P}^e (\mathbf{F}^e) = \mu_0 \mathbf{F}^e + \lambda_0 \ln(\det(\mathbf{F}^e)) - \mu_0 \mathbf{F}^{-1} \]

- stress calculation @ integration point level
  \( \mathbf{P} \rightarrow \sigma \)

check in matlab!

finite element method

function [Ke,Re,Ie] = tetra_3d(e_mat,e_spa,i_var,mat)
% stiffness matrix Ke and right hand side Re for linear tet's
[nmat,ngw] = size(mat); nod=4; ndim=3; delta=eye(ndim);
Ie = i_var; Ke = zeros(12,1); Re = zeros(12,12);
index = [1:4,7:10];
ex_mat = e_mat(indx);
ey_mat = e_mat(indx);
ex_spa = e_spa(indx);
ey_spa = e_spa(indx);
index = [3:6,9,12];
ex_mat = e_mat(indx);
ey_mat = e_mat(indx);
ex_spa = e_spa(indx);
ey_spa = e_spa(indx);
gw(1)=ex_mat(1)+ex_mat(2)+ex_mat(3)+ex_mat(4))/4.0;
gw(2)=ey_mat(1)+ey_mat(2)+ey_mat(3)+ey_mat(4))/4.0;
gw(3)=ex_mat(1)+ex_mat(2)+ex_mat(3)+ex_mat(4))/4.0;
% shape functions in isoparametric space
N(1)=1-gw(1)-gw(2)-gw(3);
N(2)=+gw(1);
N(3)=+gw(2);
N(4)=+gw(3);
% partial derivatives of shape functions w.r.t. r and s and t
dnr(1,1)=1; dnr(1,2)=1; dnr(1,3)=0; dnr(1,4)=0;
dnr(2,1)=0; dnr(2,2)=1; dnr(2,3)=0; dnr(2,4)=0;
dnr(3,1)=0; dnr(3,2)=0; dnr(3,3)=1; dnr(3,4)=0;
J=nr(1)*nr(2)*nr(3);
Ke(nr(1)*nr(2)*nr(3))=J;
% element stiffness matrix Ke, residual Re, internal variables Ie

finite element method
finite element method

integration point based

- tangent operator / constitutive moduli

check in matlab!

\[ A = D_{f}P_{e} = \partial_{f}P_{e} + \partial_{f}P_{e} \otimes \partial_{f} \theta \]

- linearization of stress wrt deformation gradient

check in matlab!

constitutive equations

finite element method

integration point based

- discrete volume update

check in matlab!

\[ R_{n+1}^{2} = \frac{1}{2\Delta t}[\vec{v}_{n+1} - \vec{v}_{n}] - k_{v}(\vec{v}_{n+1})\text{tr}(C_{e} \cdot S_{e}) \leq 0 \]

- residual of biological equilibrium / volume growth

check in matlab!

constitutive equations

finite element method

updt_vol.m

% % local newton-raphson iteration
while abs(res) > tol
    Pe = f*det(Fe); A = dFm(Fe); AeFe = A*Fe; AeFe = AeFe / det(Fe); % calculate tangent
    k = k + 1; % newton-raphson iteration
    res = - k*tr_\text{Me} - \text{dth}_e - \text{tr}_{\text{Me}} - \text{tr}_{\text{th}_k} + \text{tr}_{\text{th}_k}\text{Me} + \text{C}_{\text{Me}}; % calculate residual
    end
% % update internal volume
    vol = vol + \Delta t; % update volume
end

finite element method
```
% finite element method
% ex_beams.m

function [q0, edof, emat, bc, F_ext, mat, ndim, node, ndof, nip, nlod] = ex_beams
% input data for frame example %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

emann = 100; nue = 0.3; kt = 2.0; kc = 2.0;
mt = 2.0; mc = 2.0; tt = 2.0; tc = 0.5; dt = 1.0;
emat = [emann, nue, kt, kc, mt, mc, tt, tc, dt];
xbox(1) = 0.0; xbox(2) = 8.0; nx = 32;
ybox(1) = 0.0; ybox(2) = 1.0; ny = 4;
[q0, edof] = mesh_sqr(xbox, ybox, nx, ny);

% dirichlet boundary conditions
bc(1,1) = 6; bc(1,2) = 0;
b(2,1) = 159; bc(2,2) = 0;
b(3,1) = 326; bc(3,2) = 0;

% neumann boundary conditions
F_ext = zeros(ndof, 1);
F_ext(10) = Fp/2;
F_ext(20:10:320) = Fp;
F_ext(330) = Fp/2;
% input data for beams example %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```