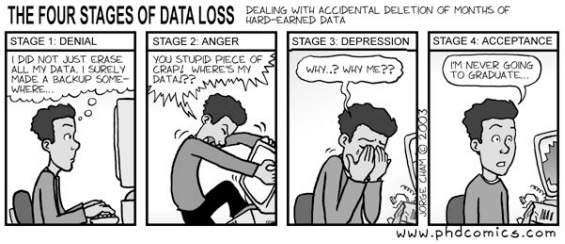
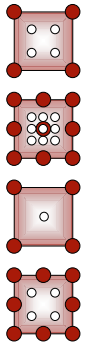
15 - finite element method - volume growth - implementation



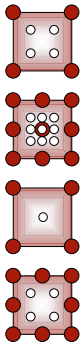
integration point based



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

staggered solution

integration point based



loop over all time steps	`nlin_fem`
global newton iteration	`nlin_fem`
loop over all elements	`tetra_3d`
loop over all quadrature points	`cnst_vol`
local newton iteration $\rho_{0\,n+1}$	`updt_vol`
determine element residual & tangent	`cnst_vol`
determine global residual and tangent	`tetra_3d`
determine φ_{n+1}	`nlin_fem`
determine state of biological equilibrium	`nlin_fem`

staggered solution

integration point based

- discrete residual

check in matlab!

$$\mathbf{R}_J^{\varphi} = \mathbf{A}_{e=1}^{n_{\mathrm{el}}} \int_{\mathcal{B}_0^e} \nabla N_{\varphi}^{j} \cdot \boldsymbol{P}_{n+1} \mathrm{d}V$$

- residual of mechanical equilibrium/balance of momentum

discrete residual

integration point based

- stiffness matrix / iteration matrix

check in matlab!

$$\mathbf{K}^{\varphi\varphi}_{JL} = \frac{\partial \mathbf{R}^{\varphi}_{J}}{\partial \boldsymbol{\varphi}_{L}} = \mathbf{A}_{e=1}^{n_{\mathrm{el}}} \int_{\mathcal{B}^e_0} \nabla N^j_{\varphi} \cdot \mathrm{D}_F \boldsymbol{P} \cdot \nabla N^l_{\varphi} \mathrm{d}V$$

- linearization of residual wrt nodal dofs

linearization

tetra_3d.m

```
function [Ke,Re,Ie] =tetra_3d(e_mat,e_spa,i_var,mat)
%%% stiffness matrix Ke and righthand side Re for linear tets %%%%%%%%%%
[nmat,ngrw]=size(mat);  nod=4;   ndim=3;    delta=eye(ndim);
Ie = i_var;             Re = zeros(12, 1);    Ke = zeros(12,12);
indx  =[1;4;7;10];   ex_mat=e_mat(indx);   ex_spa=e_spa(indx);
indy  =[2;5;8;11];   ey_mat=e_mat(indy);   ey_spa=e_spa(indy);
indz  =[3;6;9;12];   ez_mat=e_mat(indz);   ez_spa=e_spa(indz);
%%% integration points %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
gp(1)=(ex_mat(1)+ex_mat(2)+ex_mat(3)+ex_mat(4)) / 4.0;
gp(2)=(ey_mat(1)+ey_mat(2)+ey_mat(3)+ey_mat(4)) / 4.0;
gp(3)=(ez_mat(1)+ez_mat(2)+ez_mat(3)+ez_mat(4)) / 4.0;     w  = 1/6;
%%% shape functions in isoparametric space %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N(1)=1-gp(1)-gp(2)-gp(3);
N(2)= +gp(1);
N(3)=       +gp(2);
N(4)=             +gp(3);
%%% partial derivatives of shape functions wrt r and s and t %%%%%%%%%%%
dNr(1,1)=-1;  dNr(1,2)=+1;  dNr(1,3)= 0;  dNr(1,4)= 0;
dNr(2,1)=-1;  dNr(2,2)= 0;  dNr(2,3)=+1;  dNr(2,4)= 0;
dNr(3,1)=-1;  dNr(3,2)= 0;  dNr(3,3)= 0;  dNr(3,4)=+1;
JT=dNr*[ex_mat;ey_mat;ez_mat]';
%%% element stiffness matrix Ke, residual Re, internal variables Ie %%%%
```


tetra_3d.m

```
%%% element stiffness matrix Ke, residual Re, internal variables Ie %%%%%%%%%%
detJ=det(JT);
JTinv=inv(JT);
dNx=JTinv*dNr;
F  = zeros(3,3);
for i=1:4
  indx=[3*i-2; 3*i-1; 3*i];
  F=F+e_spa(indx)'*dNx(:,i)';
end

var = i_var(1);
[A,P,var]=cnst_vol(F,var,mat,ndim);
Ie(1) = var;

for i=1:nod
en=(i-1)*3;
Re(en+1)=Re(en+1)+(P(1,1)*dNx(1,i)'+P(1,2)*dNx(2,i)'+P(1,3)*dNx(3,i)')*detJ*w;
Re(en+2)=Re(en+2)+(P(2,1)*dNx(1,i)'+P(2,2)*dNx(2,i)'+P(2,3)*dNx(3,i)')*detJ*w;
Re(en+3)=Re(en+3)+(P(3,1)*dNx(1,i)'+P(3,2)*dNx(2,i)'+P(3,3)*dNx(3,i)')*detJ*w;
end
%%% element stiffness matrix Ke, residual Re, internal variables Ie %%%%%%%%%%
```


integration point based

- constitutive equations - given $\boldsymbol{F}$ calculate $\boldsymbol{P}_{\mathrm{e}}$

check in matlab!

$$\boldsymbol{P}_{\mathrm{e}}(\boldsymbol{F}_{\mathrm{e}}) = \mu_0 \boldsymbol{F}_{\mathrm{e}} + [\lambda_0 \ln(\det(\boldsymbol{F}_{\mathrm{e}})) - \mu_0] \boldsymbol{F}_{\mathrm{e}}^{-\mathrm{t}}$$

- stress calculation @ integration point level

constitutive equations

integration point based

- tangent operator / constitutive moduli

check in matlab!

$$\mathbf{A} = \mathrm{D}_F \boldsymbol{P}_\mathrm{e} = \partial_F \boldsymbol{P}_\mathrm{e} + \partial_\vartheta \boldsymbol{P}_\mathrm{e} \otimes \partial_{\boldsymbol{F}} \vartheta$$

- linearization of stress wrt deformation gradient

constitutive equations

cnst_vol.m

```
function [A,P,var]=cnst_vol(F,var,mat,ndim)
%%% determine tangent, stress and internal variable %%%%%%%%%%%%%%%%%%
emod = mat(1);  nue = mat(2);  kt = mat(3);  kc = mat(4);
mt   = mat(5);  mc  = mat(6);  tt = mat(7);  tc = mat(8); dt = mat(9);
xmu = emod / 2.0 / (1.0+nue);
xlm = emod * nue / (1.0+nue) / ( 1.0-2.0*nue );
%%% update internal variable %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[var,ten1,ten2]=updt_vol(F,var,mat,ndim);
theta  = var(1) + 1;
Fe = F/theta;   Fe_inv=inv(Fe);   Je=det(Fe);    delta =eye(ndim);

%%% first piola kirchhoff stress %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
P = xmu * Fe + (xlm * log(Je) - xmu) * Fe_inv';
%%% tangent %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:ndim;  for j=1:ndim;  for k=1:ndim;  for l=1:ndim
    A(i,j,k,l) =  xlm                  * Fe_inv(j,i)*Fe_inv(l,k) ...
                - (xlm * log(Je) - xmu) * Fe_inv(l,i)*Fe_inv(j,k) ...
                +                  xmu  *  delta(i,k)* delta(j,l) ...
                +                           ten1(i,j)*  ten2(k,l);
end, end, end, end
A = A / theta;
%%% determine tangent, stress and internal variable %%%%%%%%%%%%%%%%%%
```


integration point based

- discrete volume update

check in matlab!

$$\mathrm{R}^{\vartheta}_{n+1} = \frac{1}{\Delta t}[\vartheta_{n+1} - \vartheta_n] - k_\vartheta(\vartheta_{n+1})\mathrm{tr}(\boldsymbol{C}_\mathrm{e} \cdot \boldsymbol{S}_\mathrm{e}) \doteq 0$$

- residual of biological equilibrium / volume growth

constitutive equations

updt_vol.m

```
function [var,ten1,ten2]=updt_vol(F,var,mat,ndim)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
the_k0 = var(1)+1;  the_k1=var(1)+1;  iter=0;   res  = 1;
%%% local newton-raphson iteration %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
while abs(res) > tol
  Fe=F/the_k1;  Ce=Fe'*Fe;   Fe_inv=inv(Fe);  Ce_inv=inv(Ce);  Je=det(Fe);
  Se        = xmu * delta + (xlm * log(Je) - xmu) * Ce_inv;
  tr_Me     = trace(Ce*Se);
  dtrM_dthe = - 1/the_k1 * ( 2*tr_Me + CeLeCe );
  CeLeCe    = ndim * ndim * xlm - 2 * ndim * (xlm * log(Je) - xmu);
  k         = kt*((tt-the_k1)/(tt-1))^mt;
  dk_dthe   = k  /(the_k1-tt)        *mt;
  res  =  k * tr_Me * dt - the_k1 + the_k0;
  dres = (dk_dthe * tr_Me + k * dtrM_dthe)*dt -1;
  dthe = -res/dres;  the_k1 =  the_k1 + dthe;
end
%%% local newton-raphson iteration %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fac1 = -1 / the_k1;        fac2 = -k / dres * dt;
Pe   = xmu * Fe +           (xlm*log(Je) - xmu) * Fe_inv';
AeFe = xmu * Fe + (ndim*xlm - xlm*log(Je) + xmu) * Fe_inv';
ten1 = fac1 *  AeFe;       ten2 = fac2 * (Pe+AeFe);var(1) = the_k1 - 1
%%% update internal variable volume %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

ex_beams.m

```
function [q0,edof,emat,bc,F_ext,mat,ndim,nel,node,ndof,nip,nlod] = ex_beams
%%% input data for frame example %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
emod = 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;
mat = [emod,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);
[nel, sizen]=size(edof);   [ndof,sizen]=size(q0);   node=ndof/2;     nip=4;

%%% dirichlet boundary conditions
bc(1,1) =   6; bc(1,2) = 0;
bc(2,1) = 159; bc(2,2) = 0;
bc(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 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```