8. MATLAB code demonstration

(1) *main* program

The main program developed in a script mainly deals with

① Specifying input parameters:

a. S_0 and e_0 are specified in the 10-th and 11-th numbers in the `mate` array.

b. `nho`is the hardening exponent expressed as n in the hypoelastic material constitutive law and is specified in the 12-th number in the `mate` array.

c. `nstpho`, and `nprtho` are the number of load increments, and the number of increment steps for one *.vtk file and are specified in the 13-th and 14-th numbers in the `mate` array.

d. The function `GlobStif` has to be a function of current global displacment vector denoted as `w`.

② Modifying the global stiffness matrix

③ Calculating the global residual vector

```
.
.
.
[ndime,nnode,nelem,nelnd,npres,ntrac,mate,coor,conn,pres,trac] = ReadInput(infile);
rglob = GlobTrac(ndime,nnode,nelem,nelnd,ntrac,coor,conn,trac);
nstpho = mate(13);
tol = 1e-4;
maxit = 100;
wglob = zeros(nnode*ndime,1);
for i = 2:nstpho+1
  faci = (i-1)/nstpho;
  erri = tol*100;
  niti = 0;
  disp(['Step: ' num2str(i) ', factor: ' num2str(faci)]);
  while(erri > tol && niti < maxit)
    niti = niti+1;
    kglob = GlobStif(ndime,nnode,nelem,nelnd,mate,coor,conn,,wglob);
    fglob = GlobResi(ndime,nnode,nelem,nelnd,coor,mate,wglob);
    bglob = faci*rglob-fglob;
    for j = 1:npres
      ir = ndime*(pres(1,j)-1)+pres(2,j);
      for ic = 1:ndime*nnode
        kglob(ir,ic) = 0;
      end
      kglob(ir,ir) = 1;
      bglob(ir,ir) = -wglob(ir) + faci*pres(3,j);
    end
    dwglob = kglob\bglob;
    dwglobsq = dwglob.'*dwglob;
    wglob = wglob + dwglob;
    wglobsq = wglob.'*wglob;
    erri = sqrt(dwglobsq/wglobsq);
    disp(['Iter: ' num2str(niti) ', err: ' num2str(erri)]);
  end
end
.
.
.
```

(2) *GlobStif* function

```
function kglob = GlobStif(ndime,nnode,nelem,nelnd,mate,coor,conn,wglob)
  kglob = zeros(ndim*nnode,ndim*nnode);
  for j = 1:nelem
    kel = ElemStif(j,ndime,nelnd,coor,conn,mate,wglob);
    for a = 1:nelnd
      for i = 1:ndime
        for b = 1:nelnd
          for k = 1:ndime
            ir = ndime*(conn(a,j)-1)+i;
            ic = ndime*(conn(b,j)-1)+k;
            kglob(ir,ic) = kglob(ir,ic)+kel(ndime*(a-1)+i,ndime*(b-1)+k);
          end
        end
      end
    end
  end
end
```

(3) *ElemStif* function

```
function kel = ElemStif(iel,ndime,nelnd,coor,conn,mate,wglob)
  kel = zeros(ndime*nelnd,ndime*nelnd);
  coorie = zeros(ndime,nelnd);
  wie = zeros(ndime,nelnd);
  xii = zeros(ndime,1);
  epsi = zeros(ndime,ndime);
  dxdxi = zeros(ndime,ndime);
  dNdx = zeros(nelnd,ndime);
  M = numIntegPt(ndime,nelnd);
  xi = IntegPt(ndime,nelnd,M);
  w = IntegWt(ndime,nelnd,M);
  for a = 1:nelnd
    for i = 1:ndime
      coorie(i,a) = coor(i,conn(a,iel));
      wie(i,a) = wglob(ndime*(conn(a,iel)-1)+i);
    end
  end
  for im = 1:M
    for i = 1:ndime
      xii(i) = xi(i,im);
    end
    dNdxi = ShpFuncDeri(nelnd,ndime,xii);
    dxdxi(:) = 0;
    for i = 1:ndime
      for j = 1:ndime
        for a = 1:nelnd
          dxdxi(i,j) = dxdxi(i,j)+coorie(i,a)*dNdxi(a,j);
        end
      end
    end
    dxidx = inv(dxdxi);
    jcb = det(dxdxi);
    dNdx(:) = 0;
```

```
    for a = 1:nelnd
      for i = 1:ndime
        for j = 1:ndime
          dNdx(a,i) = dNdx(a,i)+dNdxi(a,j)*dxidx(j,i);
        end
      end
    end
    epsi(:) = 0;
    for i = 1:ndime
      for j = 1:ndime
        for a = 1:nelnd
          epsi(i,j) = epsi(i,j)+0.5*(wie(i,a)*dNdx(a,j)+wie(j,a)*dNdx(a,i));
        end
      end
    end
    dsde = MatStif(ndime,mate,epsi);
    for a = 1:nelnd
      for i = 1:ndime
        for b = 1:nelnd
          for k = 1:ndime
            ir = ndime*(a-1)+i;
            ic = ndime*(b-1)+k;
            for j = 1:ndime
              for l = 1:ndime
                kel(ir,ic)=kel(ir,ic)+dsde(i,j,k,l)*dNdx(b,l)*dNdx(a,j)*w(im)*jcb;
              end
            end
          end
        end
      end
    end
  end
end
```

(4) *MatStif* function

```
function dsde = MatStif(ndime,mate,epsi)
    emod = mate(10)/mate(11)*mate(12);
    nu = mate(3);
    dsde = zeros(ndime,ndime,ndime,ndime);
    dlt = [ [1,0,0]; [0,1,0]; [0,0,1] ];
    if (ndime == 2)
      evol = epsi(1,1)+epsi(2,2);
    else
      evol = epsi(1,1)+epsi(2,2)+epsi(3,3);
    end
    ee = 0;
    eij = zeros(ndime,ndime);
    for i = 1:ndime
      for j = 1:ndime
        eij(i,j) = epsi(i,j)-dlt(i,j)*evol/3;
        ee = ee + eij(i,j)*eij(i,j);
      end
    end
    ee = sqrt(2.*ee/3.);
```

```
    se = EffcStrs(mate,ee);
    dsedee = HdnSlp(mate,ee);
    elascoef = emod/9/(1-2*nu);
    for i = 1:ndime
      for j = 1:ndime
        for k = 1:ndime
          for l = 1:ndime
            if(ee > 0)
              dsde(i,j,k,l) = 2/3*se/ee*(dlt(i,k)*dlt(j,l)-dlt(i,j)*dlt(k,l)/3)+...
                4/9*(dsedee-se/ee)*eij(i,j)*eij(k,l)/ee/ee+...
                elascoef*dlt(i,j)*dlt(k,l);
            else
              dsde(i,j,k,l) = 2/3*dsedee*(dlt(i,k)*dlt(j,l)-dlt(i,j)*dlt(k,l)/3)+...
                elascoef*dlt(i,j)*dlt(k,l);
            end
          end
        end
      end
    end
end
```

(5) *EffcStrs* function

```
function se = EffcStrs(mate,ee)
    s0 = materialprops(10);
    e0 = materialprops(11);
    nho = materialprops(12);
    if(ee <= e0)
      if (abs(nho-1) < 1e-12)
        se = s0*ee/e0;
      else
        se = s0*(sqrt( (1+nho^2)/(nho-1)^2 - (nho/(nho-1)-ee/e0)^2 )-1/(nho-1));
      end
    else
      se = s0*( (ee/e0)^(1/nho) );
    end
end
```

(6) *HdnSlp* function

```
function dsedee = HdnSlp(mate,ee)
    s0 = materialprops(10);
    e0 = materialprops(11);
    nho = materialprops(12);
    if(ee <= e0)
      if (abs(nho-1) < 1e-12)
        dsedee = s0/e0;
      else
        dsedee = s0*(nho/(nho-1)-ee/e0)/e0/sqrt((1+nho^2)/(nho-1)^2-(nho/(nho-1)-ee/e0)^2);
      end
    else
      dsedee = s0/nho/ee*(ee/e0)^(1/nho);
    end
end
```

(7) *GlobResi* function

```
function fglob = GlobResi(ndime,nnode,nelem,nelnd,mate,coor,conn,wglob)
  fglob = zeros(ndime*nnode,1);
  for iel = 1:nelem
    fel = ElemResi(iel,ndime,nelnd,coor,conn,mate,wglob);
    for a = 1:nelnd
      for i = 1:ndime
        ir = ndime*(conn(a,ie)-1)+i;
        fglob(ir) = fglob(ir)+fel(ndime*(a-1)+i);
      end
    end
  end
end
```

(8) *ElemResi* function

```
function fel = ElemResi(iel,ndime,nelnd,coor,conn,mate,wglob)
  fel = zeros(ndime*nelnd,1);
  coorie = zeros(ndime,nelnd);
  wie = zeros(ndime,nelnd);
  xii = zeros(ndime,1);
  epsi = zeros(ndime,ndime);
  dxdxi = zeros(ndime,ndime);
  dNdx = zeros(nelnd,ndime);
  M = numIntegPt(ndime,nelnd);
  xi = IntegPt(ndime,nelnd,M);
  w = IntegWt(ndime,nelnd,M);
  for a = 1:nelnd
    for i = 1:ndime
      coorie(i,a) = coor(i,conn(a,iel));
      wie(i,a) = wglob(ndime*(conn(a,iel)-1)+i);
    end
  end
  for im = 1:M
    for i = 1:ndime
      xii(i) = xi(i,im);
    end
    dNdxi = ShpFuncDeri(nelnd,ndime,xii);
    dxdxi(:) = 0;
    for i = 1:ndime
      for j = 1:ndime
        for a = 1:nelnd
          dxdxi(i,j) = dxdxi(i,j)+coorie(i,a)*dNdxi(a,j);
        end
      end
    end
    dxidx = inv(dxdxi);
    jcb = det(dxdxi);
    dNdx(:) = 0;
    for a = 1:nelnd
      for i = 1:ndime
        for j = 1:ndime
          dNdx(a,i) = dNdx(a,i)+dNdxi(a,j)*dxidx(j,i);
        end
      end
    end
```

```
    epsi(:) = 0;
    for i = 1:ndime
      for j = 1:ndime
        for a = 1:nelnd
          epsi(i,j) = epsi(i,j)+0.5*(wie(i,a)*dNdx(a,j)+wie(j,a)*dNdx(a,i));
        end
      end
    end
    sigm = MatStrs(ndime,mate,epsi);
    for a = 1:nelnd
      for i = 1:ndime
        ir = ndime*(a-1)+i;
          for j = 1:ndime
            fel(ir) = fel(ir)+sigm(i,j)*dNdx(a,j)*w(im)*jcb;
          end
        end
      end
    end
  end
end
```

(9) *MatStrs* function

```
function sigm = MatStrs(ndime,mate,epsi)
  sigm = zeros(ndime,ndime);
  emod = mate(10)/mate(11)*mate(12);
  nu = mate(3);
  dlt = [ [1,0,0]; [0,1,0]; [0,0,1] ];
  if (ndime == 2)
    evol = epsi(1,1)+epsi(2,2);
  else
    evol = epsi(1,1)+epsi(2,2)+epsi(3,3);
  end
  ee = 0;
  eij = zeros(ndime,ndime);
  for i = 1:ndime
    for j = 1:ndime
      eij(i,j) = epsi(i,j)-dlt(i,j)*evol/3;
      ee = ee + eij(i,j)*eij(i,j);
    end
  end
  ee = sqrt(2.*ee/3.);
  se = EffcStrs(mate,ee);
  elascoef = emod/9/(1-2*nu);
  for i = 1:ndime
    for j = 1:ndime
      if(ee > 0)
        sigm(i,j) = elascoef*dlt(i,j)*evol+2/3*se*eij(i,j)/ee;
      else
        sigm(i,j) = 0;
      end
    end
  end
end
```

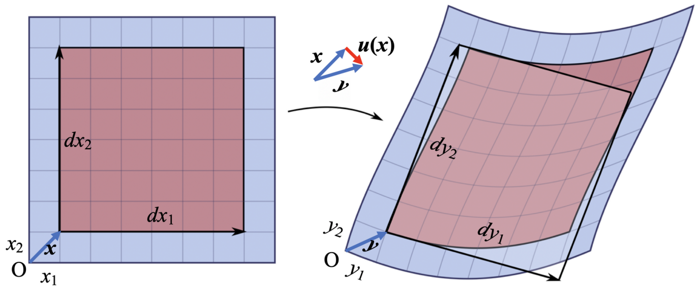
5.3 FEM for hyperelastic materials

1. General concepts

   In this section, FEM is going to be used for solving problems involving *large shape changes* of solids. As usual, we will study how to do this through theoretical derivation and related code demonstration for a solid made from a hyperelastic material.

2. Hyperelastic constitutive law

   (1) For *large shape changes* of such nearly incompressible solids described by the hyperelastic constitutive law, the following deformation measures in finite elasticity are needed:



① Deformed coordinate

$$y_i = x_i + u_i(x_j) \tag{5.49}$$

② Deformation gradient

$$F_{ij} = \frac{\partial y_i}{\partial x_j} = \frac{x_i + u_i(x_k)}{x_j} = \delta_{ij} + \frac{\partial u_i}{\partial x_j} \tag{5.50}$$

③ Deformation Jacobian

$$\hat{J} = \det(\boldsymbol{F}) = \begin{vmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \end{vmatrix} \tag{5.51}$$

such as $dy_1\, dy_2 = \hat{J}\, dx_1\, dx_2$ in 2D

$$\hat{J} = \det(\boldsymbol{F}) = \begin{vmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_2}{\partial x_3} \\ \frac{\partial y_3}{\partial x_1} & \frac{\partial y_3}{\partial x_2} & \frac{\partial y_3}{\partial x_3} \end{vmatrix} \tag{5.52}$$

such as $dy_1\, dy_2\, dy_3 = \hat{J}\, dx_1\, dx_2\, dx_3$ in 3D

④ Left Cauchy-Green deformation tensor

$$\boldsymbol{B} = \boldsymbol{F}\,\boldsymbol{F}^T \quad (\text{i.e. } B_{ij} = F_{ik}\, F_{jk}) \tag{5.53}$$

⑤ Invariants of left Cauchy-Green deformation tensor

$$I_1 = \text{trace}(\boldsymbol{B}) = B_{kk} \tag{5.54}$$

$$I_2 = \frac{1}{2}\left[I_1^2 - \text{trace}(\boldsymbol{B}\,\boldsymbol{B})\right] = \frac{1}{2}\left(I_1^2 - B_{ij}B_{ji}\right) \tag{5.55}$$

$$I_3 = \det(\boldsymbol{B}) = J^2 \tag{5.56}$$

$$\overline{I}_1 = \frac{I_1}{J^{2/3}} = \frac{B_{kk}}{J^{2/3}} \tag{5.57}$$

$$\overline{I}_2 = \frac{I_2}{J^{4/3}} = \frac{1}{2}\left(\overline{I}_1^2 - \frac{B_{ij}B_{ji}}{J^{4/3}}\right) \tag{5.58}$$

$$\hat{J} = \sqrt{\det(\boldsymbol{B})} \tag{5.59}$$

(2) In hyperelastic constitutive law, the calculated displacements, strains, and stresses of such nonlinear materials from FEM need to satisfy the equations following the *neo-Hookean constitutive law* as a hyperelastic constitutive law, which relates stress to finite deformation measures:

① Strain energy density

$$\overline{U} = \frac{\mu_1}{2}(\overline{I}_1 - 3) + \frac{K_1}{2}(\hat{J} - 1)^2 \tag{5.60}$$

where μ_1 and K_1 are the shear modulus and bulk modulus of the solid respectively. The shear modulus μ_1 is sometimes denoted by G or μ; the bulk modulus K_1 is sometimes denoted as K. These two material constants of neo-Hookean constitutive law can be expressed by the Young's modulus E and the Poisson's ratio ν as:

$$\begin{aligned} \mu_1 &= \frac{E}{3(1-2\nu)} \\ K_1 &= \frac{E}{2(1+\nu)} \end{aligned} \tag{5.62}$$

② Cauchy stress

$$\begin{aligned} \sigma_{ij} &= \frac{2}{\hat{J}^{5/3}}\left(\frac{\partial \overline{U}}{\partial \overline{I}_1} + \overline{I}_1 \frac{\partial \overline{U}}{\partial \overline{I}_2}\right) B_{ij} - \frac{2}{3\hat{J}}\left(\overline{I}_1 \frac{\partial \overline{U}}{\partial \overline{I}_1} + 2\overline{I}_2 \frac{\partial \overline{U}}{\partial \overline{I}_2}\right)\delta_{ij} - \frac{2}{\hat{J}^{7/3}}\frac{\partial \overline{U}}{\partial \overline{I}_2} B_{ik}B_{kj} + \frac{\partial \overline{U}}{\partial \hat{J}}\delta_{ij} \\ &= \frac{\mu_1}{\hat{J}^{5/3}}\left(B_{ij} - \frac{1}{3}B_{kk}\delta_{ij}\right) + K_1(\hat{J} - 1)\delta_{ij} \end{aligned} \tag{5.63}$$

3. Governing equation

(1) As always, the stress equilibrium equation is replaced by the equivalent principle of virtual work, which now has to be in a form appropriate for finite deformations.

(2) The shape of the solid in its unloaded condition R_0 will be taken as the stress-free reference configuration.

(3) Body force distributions b acting on the solid, and note that the b denote force per unit volume and unit mass.

(4) Tractions are specified as force per unit deformed area in 3D problems (deformed line in 2D problems). We need the tractions t_0 per unit *undeformed area* acting on $\partial_1 R_0$ for convenience.

(5) Mass density of the solid in its reference configuration R_0 is denoted as ρ_0.

(6) Governing equations of a boundary value problem for hyperelastic materials:

① Static equilibrium for stresses

$$\frac{\partial \sigma_{ij}}{\partial y_i} + \rho\, b_j = 0 \tag{5.64}$$

② Boundary conditions on displacement and stress

$u_i = u_i^*$ on $\partial_1 R$

and

$\sigma_{ij} n_i = t_j^*$ on $\partial_2 R$

③ Cauchy stress related to left Cauchy-Green tensor through the neo-Hookean constitutive law

$$\sigma_{ij} = \frac{\mu_1}{\hat{J}^{5/3}}\left(B_{ij} - \frac{1}{3}B_{kk}\delta_{ij}\right) + K_1(\hat{J} - 1)\delta_{ij} \tag{5.65}$$

4. Principle of virtual work

(1) The virtual work equation can be given in terms of various stress and deformation measures. For our purposes, a slightly modified form of the version in terms of *Kirchhoff stress* is the most convenient in the following derivation. The original virtual work equation is:

$$\int_R \sigma_{ij} \frac{\partial \delta v_i}{\partial y_j} dV - \int_{R_0} \rho_0\, b_i\, \delta v_i\, dV_0 - \int_{\partial R_{02}} t_i^*\, \delta v_i\, dA_0 = 0 \tag{5.66}$$

for all admissible virtual velocity fields $\delta v_i(x_j)$ for the body force and traction force subjected to unit undeformed volume and area respectively. Hence, in such problems of large shape changes of solids, it is more convenient to evaluate integrals over reference configuration.

(2) Recall that

$$dV = \hat{J}\, dV_0 \tag{5.67}$$

therefore

$$\int_{R_0} \sigma_{ij} \frac{\partial \delta v_i}{\partial y_j} \hat{J}\, dV_0 - \int_R \rho_0\, b_i\, \delta v_i\, dV_0 - \int_{\partial R_{02}} t_i^*\, \delta v_i\, dA_0 = 0 \tag{5.68}$$

(3) Kirchhoff stress is defined as

$$\tau_{ij} = \hat{J} \sigma_{ij} \tag{5.69}$$

(4) Virtual velocity gradients is defined as

$$\delta L_{ij} = \frac{\partial \delta v_i}{\partial y_j} = \frac{\partial \delta v_i}{\partial x_m}\frac{\partial x_m}{\partial y_j} = \frac{\partial \delta v_i}{\partial x_m} F_{mj}^{-1} \tag{5.70}$$

satisfying $\delta v_i = 0$ on $\partial_1 R$ for all virtual velocity fields $\delta v_i(x_j)$, where F_{mj}^{-1} is the number of the m-th row and j-th column in the inverse deformation gradient matrix.

(5) The virtual work equation is:

$$\int_{R_0} \tau_{ij} \frac{\partial \delta v_i}{\partial x_m} F_{mj}^{-1}\, dV_0 - \int_R \rho_0\, b_i\, \delta v_i\, dV_0 - \int_{\partial R_{02}} t_i^*\, \delta v_i\, dA_0 = 0 \tag{5.71}$$

5. Finite element equation

(1) The finite element solution follows almost exactly the same procedure as before. First, the displacement field is discretized by choosing the displacement distribution at a set of n nodes.

(2) The coordinates of these nodal points in the *reference configuration* are denoted as x_i^a , where the superscript a ranges from 1 to n. The unknown displacement vector at each nodal point will be denoted as u_i^a. The displacement field and virtual velocity field at an arbitrary point within the solid is again specified by interpolating between nodal values in a convenient way:

$$u_i(\boldsymbol{x}) = \sum_{a=1}^{n} N^a(\boldsymbol{x})\, u_i^a \tag{5.72}$$

$$\delta v_i(\boldsymbol{x}) = \sum_{a=1}^{n} N^a(\boldsymbol{x})\, \delta v_i^a \tag{5.73}$$

(3) Note that

① $\boldsymbol{x}$ denotes the coordinates of an arbitrary point in the reference configuration.

② The interpolation gives virtual velocity as a function of position $\boldsymbol{x}$ in the reference configuration, not $\boldsymbol{y}$ in the deformed configuration, so we have to be careful when computing the velocity gradient.

(4) Observe that the deformation gradient corresponding to a given displacement field can be evaluated as:

$$F_{ij} = \delta_{ij} + \frac{\partial u_i}{\partial x_j} = \delta_{ij} + \sum_{a=1}^{n} \frac{\partial N^a}{\partial x_j} u_i^a \tag{5.74}$$

(5) The derivatives of shape functions with respect to reference coordinates are computed exactly the same as for small strain problems before . Let $N_a(\xi_i)$ denote the shape functions in terms of Gaussian element coordinates $\xi_i)$. Then interpolate position within the element as:

$$x_i = \sum_{a=1}^{N_e} N^a(\xi_j)\, x_i^a \tag{5.75}$$

The number of the i-th row and j-th column in the Jacobian matrix is

$$\frac{\partial x_i}{\partial \xi_j}=\sum_{a=1}^{N_e}\frac{\partial N^a}{\partial \xi_j}x_i^a \tag{5.76}$$

Then,

$$\frac{\partial N^a}{\partial x_j}=\frac{\partial N^a}{\partial \xi_k}\frac{\partial \xi_k}{\partial x_j}=\frac{\partial N^a}{\partial \xi_k}\left(\frac{\partial x_k}{\partial \xi_j}\right)^{-1} \tag{5.77}$$

where $\left(\frac{\partial x_k}{\partial \xi_j}\right)^{-1}$ is the number of the k-th row and j-th column in the inverse Jacobian matrix.

(6) The Kirchhoff stress depends on displacements through the deformation gradient, and the functional relationship can be expressed as $\tau_{ij}[F_{k\ell}(u_m^a)]$.

(7) We now are ready to substitute everything back into the virtual work equation as:

$$\left\{\int_{R_0}\tau_{ij}[F_{k\ell}(u_m^a)]\frac{\partial N^a}{\partial x_m}F_{mj}^{-1}\,dV_0-\int_R \rho_0\,b_i\,N^a\,dV_0-\int_{\partial R_{02}}t_i^*\,N^a\,dA_0\right\}\,\delta v_i^a=0$$

Since the above equation must hold for all δv_i^a, we must ensure that

$$\int_{R_0}\tau_{ij}[F_{k\ell}(u_m^a)]\frac{\partial N^a}{\partial x_m}F_{mj}^{-1}\,dV_0-\int_R \rho_0\,b_i\,N^a\,dV_0-\int_{\partial R_{02}}t_i^*\,N^a\,dA_0=0 \tag{5.78}$$

$\forall\{a,i\}:x_i^a$ not on $\partial_1 R$, and

$u_i^a=u_i^*(x_i^a)\quad\forall\{a,i\}:x_i^a$ on $\partial_1 R$

Now, the equations are *nonlinear*, since the Kirchhoff stress is a nonlinear function of the unknown nodal displacements u_i^a, which is very similar to those for hypoelastic problems, except that now we have to deal with all the additional geometric terms associated with finite deformations. The procedure for solving these equations is outlined in the following sections.

6. Newton–Raphson method

As before, we can solve the nonlinear finite element equation through Newton–Raphson iteration, as follows:

(1) Start with initial guess for u_i^a, say w_i^a (we can start with zero displacements). This guess solution in general will not satisfy the governing equation.

(2) Next, we attempt to correct this guess to bring it closer to the proper solution by setting $w_i^a\to w_i^a+dw_i^a$. Ideally, we would want the correction to satisfy:

$$\int_{R_0}\tau_{ij}[F_{pq}(w_k^b+dw_k^b)]\frac{\partial N^a}{\partial x_m}(F_{mj}+dF_{mj})^{-1}\,dV_0-\int_R \rho_0\,b_i\,N^a\,dV_0-\int_{\partial R_{02}}t_i^*\,N^a\,dA_0=0 \tag{5.79}$$

where $(F_{mj}+dF_{mj})^{-1}$ denotes the number of the m-th row and j-th column in the inverse deformation gradient matrix for the updated solution. However, this equation cannot be solved for dw_k^b in the present form.

(3) It is necessary to linearize the above equation in dw_k^b , just as for the hypoelastic problems discussed in the preceding section. The linearization which can yields a system of linear equations is derived in detailed below:

a. Note that

$$F_{ij}=\delta_{ij}+\frac{\partial w_i}{\partial x_j}=\delta_{ij}+\sum_{a=1}^{n}\frac{\partial N^a}{\partial x_j}w_i^a$$

$$\frac{\partial F_{ij}}{\partial w_k^a}=\frac{\partial N^a}{\partial x_j}\delta_{ik}$$

b. Since $\boldsymbol{F}^{-1}\boldsymbol{F}=\boldsymbol{I}$,

$$\frac{\partial \boldsymbol{F}^{-1}}{\partial \boldsymbol{w}}\boldsymbol{F}+\boldsymbol{F}^{-1}\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{w}}=\boldsymbol{0}$$

$$\frac{\partial \boldsymbol{F}^{-1}}{\partial \boldsymbol{w}}=-\boldsymbol{F}^{-1}\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{w}}\boldsymbol{F}^{-1}$$

$$\frac{\partial F_{mj}^{-1}}{\partial w_k^b}=-F_{mp}^{-1}\frac{\partial F_{pq}}{\partial w_k^b}F_{qj}^{-1}$$

So,

$$
\begin{aligned}
(F_{mj}+dF_{mj})^{-1} &\approx F_{mj}^{-1}+\frac{\partial F_{mj}^{-1}}{\partial w_k^b}dw_k^b \\
&\approx F_{mj}^{-1}-F_{mp}^{-1}\frac{\partial F_{pq}}{\partial w_k^b}F_{qj}^{-1}dw_k^b \\
&\approx F_{mj}^{-1}-F_{mp}^{-1}\frac{\partial N^b}{\partial x_q}\delta_{pk}F_{qj}^{-1}dw_k^b \\
&\approx F_{mj}^{-1}-F_{mk}^{-1}\frac{\partial N^b}{\partial x_q}F_{qj}^{-1}dw_k^b
\end{aligned}
$$

c. In addition,

$$
\begin{aligned}
\tau_{ij}[F_{pq}(w_k^b+dw_k^b)] &\approx \tau_{ij}[F_{pq}(w_k^b)]+\frac{\partial \tau_{ij}}{\partial w_k^b}dw_k^b \\
&\approx \tau_{ij}[F_{pq}(w_k^b)]+\frac{\partial \tau_{ij}}{\partial F_{pq}}\frac{\partial F_{pq}}{\partial w_k^b}dw_k^b \\
&\approx \tau_{ij}[F_{pq}(w_k^b)]+\frac{\partial \tau_{ij}}{\partial F_{pq}}\frac{\partial N^b}{\partial x_q}\delta_{pk}\,dw_k^b \\
&\approx \tau_{ij}[F_{pq}(w_k^b)]+\frac{\partial \tau_{ij}}{\partial F_{kq}}\frac{\partial N^b}{\partial x_q}\,dw_k^b
\end{aligned}
$$

d. Subsituting back into Eq. (5.79), we obtain that:

$$
\int_{R_0}\left\{\tau_{ij}[F_{pq}(w_k^b)]+\frac{\partial \tau_{ij}}{\partial F_{kq}}\frac{\partial N^b}{\partial x_q}\,dw_k^b\right\}\frac{\partial N^a}{\partial x_m}\left\{F_{mj}^{-1}-F_{mk}^{-1}\frac{\partial N^b}{\partial x_q}F_{qj}^{-1}dw_k^b\right\}dV_0-\int_R \rho_0\,b_i\,N^a\,dV_0- \int_{\partial R_{02}} t_i^*\,N^a\,dA_0=0
$$

$$
\int_{R_0}\tau_{ij}[F_{pq}(w_k^b)]\frac{\partial N^a}{\partial x_m}F_{mj}^{-1}\,dV_0+\left\{\int_{R_0}\frac{\partial \tau_{ij}}{\partial F_{kq}}\frac{\partial N^b}{\partial x_q}\frac{\partial N^a}{\partial x_m}F_{mj}^{-1}\,dV_0-\int_{R_0}\tau_{ij}[F_{pq}(w_k^b)]\frac{\partial N^a}{\partial x_m}F_{mk}^{-1}\frac{\partial N^b}{\partial x_q}F_{qj}^{-1}\,dV_0\right\}dw_k^b \\
-\int_R \rho_0\,b_i\,N^a\,dV_0-\int_{\partial R_{02}} t_i^*\,N^a\,dA_0=0
$$

(4) This is evidently a system of linear equations for the correction dw_k^b of the form:

$$
K_{aibk}\,dw_k^b-R_i^a+F_i^a=0 \tag{5.80}
$$

where

$$
K_{aibk}=\int_{R_0}\frac{\partial \tau_{ij}}{\partial F_{kq}}\frac{\partial N^b}{\partial x_q}\frac{\partial N^a}{\partial x_m}F_{mj}^{-1}\,dV_0-\int_{R_0}\tau_{ij}[F_{pq}(w_k^b)]\frac{\partial N^a}{\partial x_m}F_{mk}^{-1}\frac{\partial N^b}{\partial x_q}F_{qj}^{-1}\,dV_0 \tag{5.81}
$$

$$
R_i^a=\int_R \rho_0\,b_i\,N^a\,dV_0+\int_{\partial R_{02}} t_i^*\,N^a\,dA_0 \tag{5.82}
$$

$$
F_i^a=\int_{R_0}\tau_{ij}[F_{pq}(w_k^b)]\frac{\partial N^a}{\partial x_m}F_{mj}^{-1}\,dV_0 \tag{5.83}
$$

a. This expression above is nearly identical to the equation we needed to solve for linear elastostatic problems.

b. There are only three differences:

(a) The stiffness integrals contain the deformation-dependent terms called the "geometric stiffness" because they arise as a result of accounting properly for finite geometry changes instead of the elastic constants.

(b) Although the first integral in the stiffness integrals is symmetric, the second and third are not.

(c) We need to compute an extra term in the residual force vector.

Again, for solving dw_k^b, those integrals are divided up into contributions from each element and evaluated numerically using Gaussian quadrature at the corresponding integration points.

(5) To have a simpler set of formulas for the stiffness matrix K_{aibk} and force vector F_i^a, the concise formulae are suggested as:

$$
\frac{\partial N^a}{\partial x_m}F_{mj}^{-1}=\frac{\partial N^a}{\partial x_m}\left(\frac{\partial y_m}{\partial x_j}\right)^{-1}=\frac{\partial N^a}{\partial x_m}\frac{\partial x_m}{\partial y_j}=\frac{\partial N^a}{\partial y_j} \tag{5.84}
$$

$$
\begin{aligned}
K_{aibk} &= \int_{R_0} \frac{\partial \tau_{ij}}{\partial F_{kq}} F_{\ell q} \frac{\partial N^b}{\partial x_q} F_{q\ell}^{-1} \frac{\partial N^a}{\partial x_m} F_{mj}^{-1}\, dV_0 - \int_{R_0} \tau_{ij} [F_{pq}(w_k^b)] \frac{\partial N^a}{\partial x_m} F_{mk}^{-1} \frac{\partial N^b}{\partial x_q} F_{qj}^{-1}\, dV_0 \\
&= \int_{R_0} \frac{\partial \tau_{ij}}{\partial F_{kq}} F_{\ell q} \frac{\partial N^a}{\partial y_j} \frac{\partial N^b}{\partial y_\ell} dV_0 - \int_{R_0} \tau_{ij} [F_{pq}(w_k^b)] \frac{\partial N^a}{\partial y_k} \frac{\partial N^b}{\partial y_j}\, dV_0 \\
&= \int_{R_0} C^e_{ijk\ell} \frac{\partial N^a}{\partial y_j} \frac{\partial N^b}{\partial y_\ell}\, dV_0 - \int_{R_0} \tau_{ij} [F_{pq}(w_k^b)] \frac{\partial N^a}{\partial y_k} \frac{\partial N^b}{\partial y_j}\, dV_0
\end{aligned}
$$

where (5.85)

$$
C^e_{ijk\ell} \equiv \frac{\partial \tau_{ij}}{\partial F_{km}} F_{\ell m} \tag{5.86}
$$

(6) Similar to hypoelastic problems, the solved magnitudes of the correction displacement vector containing dw_b and the displacement vector containing w_b numbers can be used for checking convergence.

7. Tangent moduli for the hyperelastic solids

Similar to hypoelastic materials, nonlinear FEM usually needs the material tangent moduli. For the hyperelastic constitutive law used in this section, through algegraic deliberations, the equivalent material tangent moduli $C^e_{ijk\ell}$ can be derived as follows.

(1) Determinant derivative identity:

Considering a square matrix $\boldsymbol{A}$,

$$
D = \det(\boldsymbol{A}) \rightarrow \frac{\partial \mathrm{D}}{\partial \boldsymbol{A}} = \mathrm{D}\boldsymbol{A}^{-\mathrm{T}} \tag{5.87}
$$

$$
\frac{\partial D}{\partial A_{ij}} = D A_{ji}^{-1} \tag{5.88}
$$

(2) Recall

$$
\begin{aligned}
C^e_{ijk\ell} &\equiv \frac{\partial \tau_{ij}}{\partial F_{km}} F_{\ell m} \\
&= \frac{\partial (\hat{J}\sigma_{ij})}{\partial F_{km}} F_{\ell m} \\
&= \left(\frac{\partial \hat{J}}{\partial F_{km}} \sigma_{ij} + \hat{J} \frac{\partial \sigma_{ij}}{\partial F_{km}} \right) F_{\ell m} \\
&= \sigma_{ij} F_{\ell m} \hat{J} F_{mk}^{-1} + \hat{J} \frac{\partial \sigma_{ij}}{\partial F_{km}} F_{\ell m} \\
&= \hat{J} \sigma_{ij} \delta_{k\ell} + \hat{J} \frac{\partial \sigma_{ij}}{\partial F_{km}} F_{\ell m}
\end{aligned}
\tag{5.89}
$$

where

$$
\frac{\partial \hat{J}}{\partial F_{km}} = \hat{J} F_{mk}^{-1} \tag{5.90}
$$

and Eq. (5.63)

$$
\sigma_{ij} = \frac{\mu_1}{\hat{J}^{5/3}} \left(B_{ij} - \frac{1}{3} B_{kk} \delta_{ij} \right) + K_1 (\hat{J} - 1) \delta_{ij}
$$

(3) Through algegraic deliberations, it is can be shown that:

$$
C^e_{ijk\ell} = \frac{\mu_1}{J^{2/3}} \left[\delta_{ik} B_{j\ell} + B_{i\ell} \delta_{jk} - \frac{2}{3} (B_{ij}\delta_{k\ell} + B_{k\ell}\delta_{ij}) + \frac{2}{3} \frac{B_{qq}}{3} \delta_{ij} \delta_{k\ell} \right] + K_1 (2\hat{J} - 1)\, \hat{J}\, \delta_{ij} \delta_{k\ell} \tag{5.91}
$$