7.0 Solving problems with nonlinear materials

Most common FEA coding problem

7.1 Hypoelasticity : small-strain nonlinear elastic material

Preliminaries : Governing equations

- Strains $\varepsilon_{ij} = \text{sym}(\partial u_i / \partial x_j)$
- Equilibrium : weak form

$$\int_R \sigma_{ij}[\varepsilon_{pq}(u_k)] \frac{\partial \eta_i}{\partial x_j} dV - \int_{S_2} t_i^* \eta_i \, dA = 0 \quad \forall \text{admiss } \eta_i$$

$$\sigma_{ij} n_j = t_i^* \text{ on } S_2$$

$$u_i = u_i^* \text{ on } S_1$$

- Stress - strain law:
  1. Reversible ; rate independent
  2. Constant bulk modulus.
  3. Uniaxial $\sigma - \varepsilon$ curve nonlinear

$$\sigma_{ij} = S_{ij} + p\,\delta_{ij} \qquad p = K\,\varepsilon_{kk}$$

Let $$e_{ij} = \varepsilon_{ij} - \frac{\varepsilon_{kk}}{3}\delta_{ij} \qquad \varepsilon_e = \sqrt{\frac{2}{3} e_{ij} e_{ij}}$$

$$S_{ij} = \frac{2}{3}\,\sigma_e(\varepsilon_e)\,\frac{e_{ij}}{\varepsilon_e}$$

We can choose any convenient function for $\sigma_e(\varepsilon_e)$

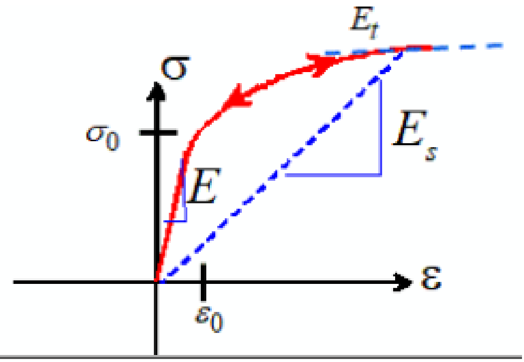
- ABAQUS uses a user-defined table

Example

Material properties:

σ_0, ε_0, $n > 1$

$$\frac{\sigma_e}{\sigma_0} = \begin{cases} \sqrt{\dfrac{1+n^2}{(n-1)^2} - \left(\dfrac{n}{n-1} - \dfrac{\varepsilon_e}{\varepsilon_0}\right)^2} - \dfrac{1}{n-1} & \varepsilon_e \le \varepsilon_0 \\ \left(\dfrac{\varepsilon_e}{\varepsilon_0}\right)^{1/n} & \varepsilon_e \ge \varepsilon_0 \end{cases}$$



Power-law but w/ finite slope @ $\varepsilon_e = 0$

FE equations: use usual FE interpolations

$$u_i = N^a u_i^a \qquad \eta_i = N^a \eta_i^a$$

$\Rightarrow$ PVW:

$$\Big\{ \underbrace{\int_R \sigma_{ij}\left[\varepsilon_{pq}\left[u_k^b\right]\right] \frac{\partial N^a}{\partial x_j}\, dV}_{R_i^a \text{ - internal force on } a\text{th node}} - \underbrace{\int_{S_2} t_i^* N^a}_{f_i^a} \Big\} \eta_i^a = 0 \quad \forall \eta_i^a$$

System of nonlinear equations for U_k^b

$$R_i^a[U_k^b] = f_i^a$$

Solve with Newton - Raphson

(1) Guess solution $U_k^b = W_k^b$ eg $W_k^b = 0$

(2) Correct $R_i^a[W_k^b + dW_k^b] = f_i^a$

(3) Taylor expansion

$$\underbrace{\frac{\partial R_i^a}{\partial U_k^b}}_{\text{Consistent Tangent or Jacobian}} dW_k^b = f_i^a - R_i^a[W_k^b]$$

Kaibk

Newton Iteration Loop

(1) Solve $K_{aibk}\, dW_k^b = f_i^a - R_i^a$

(2) $W_i^a \to W_i^a + dW_i^a$

(3) Check convergence

No

$$\frac{dW_k^b\, dW_k^b}{W_i^a W_i^a} \le tol\,1$$

$$\frac{(R_i^a - f_i^a)(R_i^a - f_i^a)}{R_k^b R_k^b} \le tol\,2$$

Y

(4) Proceed to next time increment

Newton Loop in
EN234 FEA

```
do while (continue_timesteps)

    call assemble_direct_stiffness(fail)

    if (fail) then                 ! Force a timestep cutback if stiffness computation fails
        converged=.false.
        iteration = 0
        call compute_static_time_increment(iteration,converged,continue_timesteps, &
            activatestateprint,activateuserprint,new_time_increment)
        DTIME = new_time_increment
        dof_increment = 0.d0
        cycle
    endif
    call apply_direct_boundaryconditions
    call solve_direct

    converged = .true.

    do iteration = 1,max_newton_iterations
        call assemble_direct_stiffness(fail)

        converged = .false.
        if (fail) exit

        call apply_direct_boundaryconditions
        call convergencecheck(iteration,converged)
        if (converged) exit
        call solve_direct

    end do

    call compute_static_time_increment(iteration,converged,continue_timesteps, &
        activatestateprint,activateuserprint,new_time_increment)

    if (.not.converged) then
        dof_increment = 0.d0
        DTIME = new_time_increment
        cycle
    endif

    if (activatestateprint)  call print_state
    if (activateuserprint) call user_print(current_step_number)

    !        Update solution and continue
    current_step_number = current_step_number + 1
    dof_total = dof_total + dof_increment
    initial_state_variables = updated_state_variables
    TIME = TIME + DTIME
    DTIME = new_time_increment

end do
```

Calculating the Jacobian matrix

$$K_{aibk} = \frac{\partial}{u_k^b} \int_R \sigma_{ij}\left[\varepsilon_{pq}(u_k^b)\right] \frac{\partial N^a}{\partial x_j}\, dV$$

$$= \int_R \frac{\partial \sigma_{ij}}{\partial \varepsilon_{pq}} \frac{\partial \varepsilon_{pq}}{\partial u_k^b} \frac{\partial N^a}{\partial x_j}$$

Recall $\varepsilon_{pq} = \frac{1}{2}\left\{ \frac{\partial N^c}{\partial x_q} u_p^c + \frac{\partial N^c}{\partial x_p} u_q^c \right\}$

$$\Rightarrow \frac{\partial \varepsilon_{pq}}{\partial u_k^b} = \frac{1}{2}\left\{ \frac{\partial N^b}{\partial x_q} \delta_{pk} + \frac{\partial N^b}{\partial x_p} \delta_{qk} \right\}$$

Recall $\frac{\partial \sigma_{ij}}{\partial \varepsilon_{pq}} = \frac{\partial \sigma_{ij}}{\partial \varepsilon_{qp}}$ $\Rightarrow$ can combine the two terms

Hence $K_{aibk} = \int_R \frac{\partial \sigma_{ij}}{\partial \varepsilon_{kl}} \frac{\partial N^b}{\partial x_l} \frac{\partial N^a}{\partial x_j} dV$

Exactly the same as linear elasticity, except σ_{ij} is nonlinear, and C_{ijkl} is now $\partial \sigma_{ij} / \partial \varepsilon_{kl}$

We only need to change calculation for $\underline{\sigma}$ and $[D]$

$$[D] = \begin{bmatrix} \frac{\partial \sigma_{11}}{\partial \varepsilon_{11}} & \frac{\partial \sigma_{11}}{\partial \varepsilon_{22}} & \frac{\partial \sigma_{11}}{\partial \varepsilon_{33}} & \partial \sigma_{11} / \partial \varepsilon_{12} & \cdots \\ \frac{\partial \sigma_{22}}{\partial \varepsilon_{11}} & & & & \\ \frac{\partial \sigma_{33}}{\partial \varepsilon_{11}} & & & & \end{bmatrix}$$

In ABAQUS / EN234FEA we can use the UMAT subroutine to calculate $\underline{\sigma}$ and $[D]$

ABAQUS / EN234FEA will assemble $[k^{el}]$ handle equation solving, Newton iterations etc

By default elements are $\bar{B}$ and in EN234FEA elements are finite strain

```
SUBROUTINE UMAT(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
1 RPL,DDSDDT,DRPLDE,DRPLDT,
2 STRAN,DSTRAN,TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,CMNAME,
3 NDI,NSHR,NTENS,NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,
4 CELENT,DFGRD0,DFGRD1,NOEL,NPT,LAYER,KSPT,KSTEP,KINC)
```

Outputs

Inputs

· STRESS(I) = $\underline{\sigma}$

DDSDDE(I,J) = $[D]$

Other outputs optional

Computing material tangent stiffness $\frac{\partial \sigma_{ij}}{\partial \varepsilon_{kl}}$

Recall $\sigma_{ij} = S_{ij} + p\,\delta_{ij}$ $\qquad p = K\,\varepsilon_{qq}$

$$\frac{\partial p}{\partial \varepsilon_{kl}} = K\,\delta_{qk}\,\delta_{ql} = K\,\delta_{kl}$$

$$\frac{\partial S_{ij}}{\partial \varepsilon_{kl}} = \frac{\partial}{\partial \varepsilon_{kl}}\left(\frac{2}{2}\,\sigma_e(\varepsilon_e)\,\frac{e_{ij}}{\varepsilon_e}\right)$$

$$= \frac{2}{3}\left\{\overset{E_t}{\frac{\partial \sigma_e}{\partial \varepsilon_e}} - \frac{\sigma_e}{\varepsilon_e}\right\}\frac{\partial \varepsilon_e}{\partial \varepsilon_{kl}}\,\frac{e_{ij}}{\varepsilon_e} + \frac{2}{3}\,\overset{E_s}{\frac{\sigma_e}{\varepsilon_e}}\,\frac{\partial e_{ij}}{\partial \varepsilon_{kl}}$$

$$\varepsilon_e = \sqrt{\frac{2}{3}\,e_{ij}\,e_{ij}} \Rightarrow \frac{\partial \varepsilon_e}{\partial e_{pq}} = \frac{2}{3}\,\frac{e_{pq}}{\varepsilon_e}$$

Finally $e_{ij} = \varepsilon_{ij} - \varepsilon_{pp}\,\delta_{ij}/3$

$$\frac{\partial e_{ij}}{\partial \varepsilon_{kl}} = \frac{1}{2}\underbrace{\left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right)}_{\text{Need symmetry in } k,l \text{ and } i,j} - \frac{1}{3}\delta_{kl}\,\delta_{ij}$$

$$\frac{d\sigma_{ij}}{d\varepsilon_{kl}} = \frac{4}{9}(E_t - E_s)\frac{e_{ij}e_{kl}}{\varepsilon_e^2} + \frac{2}{3}E_s\left(\frac{\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}}{2} - \frac{\delta_{ij}\delta_{kl}}{3}\right) + K\delta_{kl}\delta_{ij} \qquad E_s = \frac{\sigma_e}{\varepsilon_e} \qquad E_t = \frac{d\sigma_e}{d\varepsilon_e}$$

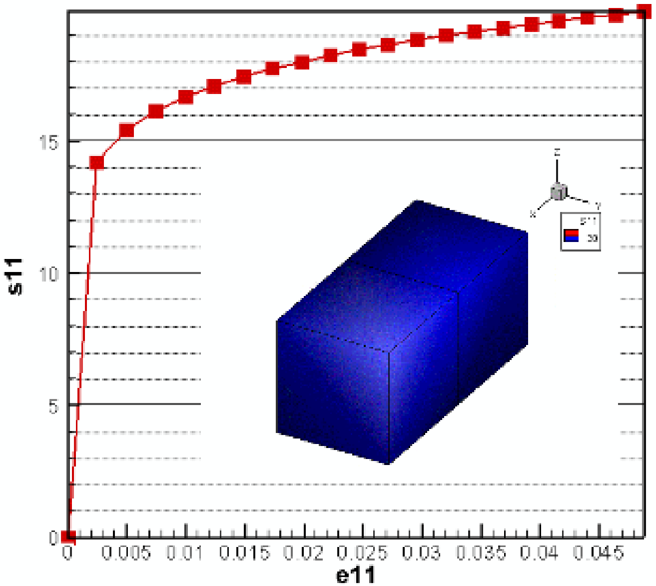
Matrix form: if we define $e = [e_{11}\ e_{22}\ e_{33}\ \underbrace{e_{12}\ e_{13}\ e_{23}}_{\text{NO 2!!}}]$

$$[D] = \frac{4}{9}[E_t - E_s]\,\underline{e} \otimes \underline{e} + \frac{2}{3}E_s \begin{bmatrix} 2 & & & & & \\ & 2 & & & 0 & \\ & & 2 & & & \\ & & & 1 & & \\ & 0 & & & 1 & \\ & & & & & 1 \end{bmatrix} + \left(K - \frac{2}{9}E_s\right) \begin{bmatrix} 1 & 1 & 1 & & & \\ 1 & 1 & 1 & & 0 & \\ 1 & 1 & 1 & & & \\ & & & & & \\ & 0 & & & 0 & \\ & & & & & \end{bmatrix}$$

Example UMAT code

Results

(See code on web)




```
      s0 = PROPS(1)
      e0 = PROPS(2)
      n = PROPS(3)
      K = PROPS(4)

      evol = sum(STRAN(1:3)+DSTRAN(1:3))
      edev(1:3) = STRAN(1:3)+DSTRAN(1:3) - evol/3.d0
      edev(4:6) = 0.5d0*(STRAN(4:6)+DSTRAN(4:6))

      ee = dsqrt(dot_product(edev(1:3),edev(1:3)) +
     1              2.d0*dot_product(edev(4:6),edev(4:6)))/dsqrt(1.5d0)

      DDSDDE(1:6,1:6) = 0.d0
      if (ee<e0) then
         se = s0*( dsqrt( (1.d0+n*n)/((n-1.d0)*(n-1.d0))
     1          - (n/(n-1.d0) - ee/e0)**2.d0 ) - 1.d0/(n-1.d0) )
         dsedee = (s0/e0)*(n/(n-1.d0) - ee/e0)/
     1 dsqrt((1.d0+n*n)/((n-1.d0)*(n-1.d0)) - (n/(n-1.d0)-ee/e0)**2.d0)
         if (ee==0.d0) then
            stress = 0.d0
            Es = n*s0/e0
         else
            Es = se/ee
            Et = dsedee
            stress = 2.d0*se*edev/(3.d0*ee)
            stress(1:3) = stress(1:3) + K*evol
            DDSDDE(1:6,1:6) =
     1         4.d0*(Et-Es)*spread(edev,dim=2,ncopies=6)*
     1                      spread(edev,dim=1,ncopies=6)/(9.d0*ee*ee)
         endif
      else
         se = s0*(ee/e0)**(1.d0/n)
         stress = 2.d0*se*edev/(3.d0*ee)
         stress(1:3) = stress(1:3) + K*evol
         Et = se/(n*ee)
         Es = se/ee
         DDSDDE(1:6,1:6) = 4.d0*(Et-Es)*spread(edev,dim=2,ncopies=6)*
     1                         spread(edev,dim=1,ncopies=6)/(9.d0*ee*ee)
      endif
      forall(j=1:3) DDSDDE(j,j) = DDSDDE(j,j) + 2.d0*Es/3.d0
      forall(j=4:6) DDSDDE(j,j) = DDSDDE(j,j) + Es/3.d0
      DDSDDE(1:3,1:3) =  DDSDDE(1:3,1:3) + (K-2.d0*Es/9.d0)
```