User material subroutines

Introduction

This notebook discusses the implementation of user material subroutine `UMAT` . The particular subroutine corresponds to the classical metal plasticity model in a J-2 - theory formulation. The subroutine is for plane strain idealizations and it considers non-linear combined isotropic/kinematic hardening.

The formulation of the model is detailed in Simo and Hughes (1998). The user subroutine corresponds to the return mapping integration algorithm. An extended version of the model (considering damage and thermal effects) together with the integration algorithm is formulated in Gomez and Basaran (2004).

Subroutine interface (input and output parameters)

The following set of parameters is passed from the main program to the user subroutine:

- `stress:` (ndarray) Stress tensor at the current integration point at the beginning of the increment.
- `strann:` (ndarray) Total strains tensor at the current integration point.
- `dstran:` (ndarray) Incremental strains at the current integration point
- `statev:` (ndarray) Sate variables array at the current integration point at the beginning of the increment.
- `props:` (ndarray) SMaterial properties for the element
- `ntens:` (int) Number of components for the stress and strain tensors..

The following set of parameters is returned by the `UEL` subroutine to the main program:

- `C:` (ndarray) Constitutive tensor.
- `stress:` (ndarray) Updated stress tensor at the current integration point.
- `statev:` (ndarray) Sate variables array at the current integration point at the end of the increment.

In [1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import sympy as sym
from os import sys
sys.path.append("../source/")
from STRUCTURE import Struct_DYN
from postprocesor import *
from uel_solid import *
```

In [2]:
```
def umat_PCLK(stress , strann , dstran , statev , props , ntens):
    """Return mapping integration algorithm for J2-flow theory
       rate independent plasticity under plane strain conditions
       combined isotropic kinematic hardenig (see Simo, Juan C., and
       Thomas JR Hughes. Computational inelasticity. Vol. 7.
       Springer Science & Business Media, 2006).

    statev     : nd array
                 Sate variables array at the current integration point
                 at the beginning of the increment.
                 4 components of the elastic strain tensor
                 4 components of the plastic strain tensor
                 4 components of the back stress tensor
                 1 equivalent plastic strain
                 1 equivalent stress
    """

    toler = 1.0e-7
    istop = 1

    eelas   = np.zeros([4])
    eplas   = np.zeros([4])
    xback   = np.zeros([4])
    flow    = np.zeros([4])
    eqplas = 0.0
    smises = 0.0

    statev , eelas , eplas , xback , eqplas , smises = stv_handl( 0 , statev , eelas ,
plas , xback , eqplas , smises , ntens)

    Emod       = props[0]
    enu        = props[1]
    eg2=Emod/(1.0+enu)
    eg = eg2/2.0
    sig0    = props[2]
    sigsat  = props[3]
    hrdrate = props[4]
    hmod    = props[5]

    C = elas_tensor(enu , Emod)
    stress = stress + np.dot(C , dstran)
    shydro =(stress[0]+stress[1]+stress[2])/3.0
    eelas  = eelas + dstran
    sdev = deviator(stress)
    stsrel = sdev - xback
    smises = vmises(stsrel)

    fbar = np.sqrt(2.0/3.0)*smises
    syiel0 , syieldk , ehardi , ehardk = uhardnlin(sig0 , sigsat , hrdrate , hmod , eq
as)
    syield   = syiel0
    syieldk0 = syieldk

    if fbar > (1.0+toler)*syiel0:
        flow = stsrel/fbar
        gam_par , eqplas , syieldk , istop = local_NR(syield, syieldk0 , ehardi , ehar
, hmod , sig0 , sigsat , hrdrate , eqplas , fbar , eg)
        for k in range(3):
            xback[k]  = xback[k] + np.sqrt(2.0/3.0)*(syieldk-syieldk0)*flow[k]
            eplas[k]  = eplas[k] + gam_par*flow[k]
```

```
            eelas[k]  = eelas[k] - eplas[k]
            stress[k] = flow[k]*syield + xback[k] + shydro
        xback[3] = xback[3] + np.sqrt(2.0/3.0)*(syieldk-syieldk0)*flow[3]
        eplas[3]  = eplas[3] + 2.0*gam_par*flow[3]
        eelas[3]  = eelas[3] - eplas[3]
        stress[3] = flow[3]*syield + xback[3]
        eqplas = eqplas+np.sqrt(2.0/3.0)*gam_par
        C = np.zeros((4 , 4))
        C = plas_tensor(gam_par , fbar , flow , ehardk , ehardi , enu , Emod)
#
# Store updated values of state variables
#
    statev , eelas , eplas , xback , eqplas , smises = stv_handl( 1 , statev , eelas ,
plas , xback , eqplas , smises , ntens)

    if istop == 0:
        print('local plasticty algorithm did not converged')
        print('After' , iter , 'iterations')
        print('Last value of the consistency parameter' , gam_par)

    return C , stress , statev
```

To test and execute the subroutine the following block of code emulates the required entries from the element subroutine to the material subroutine.

In [3]:
```
nstatev = 14
ntens  = 4
statev = np.zeros([nstatev])
stress = np.zeros([ntens])
strann = np.zeros([ntens])
dstran = np.zeros([ntens])
nprops = 7
props   = np.zeros([nprops])

dstran[0] =-0.0001/3.0
dstran[1] = 0.0001

props[0]= 52.0e3
props[1]= 0.33
props[2]= 60.0
props[3]= 37.0
props[4]= 383.3
props[5]= 2040.0
props[6]= 1000.0

C , stress , statev = umat_PCLK(stress , strann , dstran , statev , props , ntens)
```

Results

In [4]:
```
print(C)
print(stress)
print(statev)
```

```
[[77045.55506413 37947.81070323 37947.81070323     0.        ]
 [37947.81070323 77045.55506413 37947.81070323     0.        ]
 [37947.81070323 37947.81070323 77045.55506413     0.        ]
 [    0.             0.             0.         19548.87218045]]
[1.2265959  6.43962848 2.52985405 0.        ]
[-3.33333333e-05  1.00000000e-04  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  4.69896407e+00]
```

References

Simo, Juan C., and Thomas JR Hughes. Computational inelasticity. Vol. 7. Springer Science & Business Media, 1998.

Gomez, J and Basaran, C(2006) Damage mechanics constitutive model for Pb/Sn solder joints incorporating nonlinear kinematic hardening and rate dependent effects using a return mapping integration algorithm. mechanics of Materials. (38), 585-598.

In [5]:
```
from IPython.core.display import HTML
def css_styling():
    styles = open('./nb_style.css', 'r').read()
    return HTML(styles)
css_styling()
```

Out[5]:

In [ ]:
```

```