Assignment 1 - ME 562 - Brandon Lampe

Constitutive Equations and Associated Numerical Algorithms

In [224]:
```
from __future__ import division
from IPython.display import display_latex
from IPython.core.display import Math
import sys
sys.path.append('/Users/Lampe/PyScripts')
# print sys.path
from scipy import linalg as LA
import math
import numpy as np
import blfunc as bl
from sympy import *
init_printing()
np.set_printoptions(precision=3)
```

Problem 1

Function for creating orthonormal basis is contained in "blfunc", which is shown seperately.

In [225]:
```
# transformation matrix
a_eE = bl.orth_basis()
a_Ee = np.transpose(a_eE)
print a_eE
```

```
[[ 0.132  0.778  0.614]
 [ 0.898  0.168 -0.406]
 [-0.419  0.605 -0.677]]
```

In [226]:
```
# verify the transformation matrix is orthonormal
i = np.dot(a_Ee, a_eE)
print i
```

```
[[  1.000e+00   5.551e-17   1.110e-16]
 [  5.551e-17   1.000e+00   0.000e+00]
 [  1.110e-16   0.000e+00   1.000e+00]]
```

In [227]:
```
# verify the transformation matrix is nonsingular
print LA.det(a_Ee)
```

```
1.0
```

Problem 2

2.(i)

In [228]:
```
T_EE = np.array([[7, 2, 11],
                 [2, 4, 5],
                 [11, 5, 6]])
print T_EE
```

```
[[ 7  2 11]
 [ 2  4  5]
 [11  5  6]]
```

In [229]:
```
# Transform T to the e-e basis
T_ee = a_eE.dot(T_EE).dot(a_Ee)
print T_ee
```

```
[[ 11.773   5.714  -6.079]
 [  5.714  -1.348  -6.251]
 [ -6.079  -6.251   6.575]]
```

In [230]:
```
# Transform T back to the E-E basis
T_EE_ck = a_Ee.dot(T_ee).dot(a_eE)
print T_EE_ck
```

```
[[  7.   2.  11.]
 [  2.   4.   5.]
 [ 11.   5.   6.]]
```

My algorithm appears correct because I am able to transfrom T from the E-E basis to the e-e basis then back to the E-E basis and obtain my original components in the E-basis.

2(ii)

In [231]:
```
# spherical components of T in the E-E basis
T_EE_sp = np.multiply(np.trace(T_EE) / 3, np.eye(3, 3))
print T_EE_sp
```

```
[[ 5.667  0.     0.   ]
 [ 0.     5.667  0.   ]
 [ 0.     0.     5.667]]
```

In [232]:
```
# deviatoric components of T in the E-E basis
T_EE_dev = T_EE - T_EE_sp
print T_EE_dev
```

```
[[  1.333   2.     11.   ]
 [  2.     -1.667   5.   ]
 [ 11.      5.      0.333]]
```

Problem 3

3(i)

In [233]:
```
eigval, eigvec = LA.eig(T_EE)

l1_EE, l2_EE, l3_EE = eigval
print 'Eigenvalue 1 = %G + %Gi' % (l1_EE.real, l1_EE.imag)
print 'Eigenvalue 2 = %G + %Gi' % (l2_EE.real, l2_EE.imag)
print 'Eigenvalue 3 = %G + %Gi' % (l3_EE.real, l3_EE.imag)
```

```
Eigenvalue 1 = 19.1218 + 0i
Eigenvalue 2 = -5.12943 + 0i
Eigenvalue 3 = 3.00763 + 0i
```

In [234]:
```
P1_E, P2_E, P3_E = eigvec[:, 0], eigvec[:, 1], eigvec[:, 2]

print 'Principal Vector 1 in E basis => %G, %G, %G' % (P1_E[0], P1_E[1], P1_E[2])
print 'Principal Vector 2 in E basis => %G, %G, %G' % (P2_E[0], P2_E[1], P2_E[2])
print 'Principal Vector 3 in E basis => %G, %G, %G' % (P3_E[0], P3_E[1], P3_E[2])
```

```
Principal Vector 1 in E basis => -0.66617, -0.312077, -0.677366
Principal Vector 2 in E basis => -0.623059, -0.266296, 0.735449
Principal Vector 3 in E basis => 0.409896, -0.911973, 0.0170443
```

3(ii)

In [235]:
```
# orthonormal Principal basis wrt E-E:
P_EE = eigvec
print P_EE
```

```
[[-0.666 -0.623  0.41 ]
 [-0.312 -0.266 -0.912]
 [-0.677  0.735  0.017]]
```

In [236]:
```
# check that P basis is orthonormal (transpose = inverse)
P_EE.dot(np.transpose(P_EE))
```

Out[236]:
```
array([[  1.000e+00,   2.220e-16,   1.266e-16],
       [  2.220e-16,   1.000e+00,  -2.134e-16],
       [  1.266e-16,  -2.134e-16,   1.000e+00]])
```

3(iii)

In [237]:
```
# transformation matrix between the E-P basis(a_EP):
a_EP = P_EE
print a_EP
```

```
[[-0.666 -0.623  0.41 ]
 [-0.312 -0.266 -0.912]
 [-0.677  0.735  0.017]]
```

In [238]:
```
# transpose of a_EP
a_PE = np.transpose(a_EP)
print a_PE
```

```
[[-0.666 -0.312 -0.677]
 [-0.623 -0.266  0.735]
 [ 0.41  -0.912  0.017]]
```

In [239]:
```
# components of T in the Principal basis
T_PP = a_PE.dot(T_EE).dot(a_EP)
print T_PP
```

```
[[  1.912e+01   2.665e-15  -1.776e-15]
 [  2.220e-15  -5.129e+00   6.661e-16]
 [ -1.332e-15   2.220e-16   3.008e+00]]
```

Yes, the diagonal components should be the eigenvalues with all other components equal zero.

3(iv)

In [240]:
```
T_PP_inv = LA.inv(T_PP)
print T_PP_inv
```

```
[[  5.230e-02   2.717e-17   3.089e-17]
 [  2.264e-17  -1.950e-01   4.318e-17]
 [  2.317e-17   1.439e-17   3.325e-01]]
```

In [241]:
```
T_EE_inv = a_EP.dot(T_PP_inv).dot(a_PE)
print T_EE_inv
```

```
[[ 0.003 -0.146  0.115]
 [-0.146  0.268  0.044]
 [ 0.115  0.044 -0.081]]
```

In [242]:
```
#check if inverse is correct -> should get Identity matrix"
print np.dot(T_EE, T_EE_inv)
```

```
[[  1.000e+00   5.551e-17  -1.110e-16]
 [  0.000e+00   1.000e+00  -8.327e-17]
 [  0.000e+00   0.000e+00   1.000e+00]]
```

Yes, my result is correct.

3(v)

In [243]:
```
eigval, eigvec = LA.eig(T_ee)

l1_ee, l2_ee, l3_ee = eigval
print 'Eigenvalue 1 = %G + %Gi' % (l1_ee.real, l1_ee.imag)
print 'Eigenvalue 2 = %G + %Gi' % (l2_ee.real, l2_ee.imag)
print 'Eigenvalue 3 = %G + %Gi' % (l3_ee.real, l3_ee.imag)
```

```
Eigenvalue 1 = 19.1218 + 0i
Eigenvalue 2 = 3.00763 + 0i
Eigenvalue 3 = -5.12943 + 0i
```

In [244]:
```
P1_e, P2_e, P3_e = eigvec[:, 0], eigvec[:, 1], eigvec[:, 2]

print 'Principal Vector 1 in e basis => %G, %G, %G' % (P1_e[0], P1_e[1], P1_e[2])
print 'Principal Vector 2 in e basis => %G, %G, %G' % (P2_e[0], P2_e[1], P2_e[2])
print 'Principal Vector 3 in e basis => %G, %G, %G' % (P3_e[0], P3_e[1], P3_e[2])
P_ee = eigvec
```

```
Principal Vector 1 in e basis => 0.746479, 0.376013, -0.548983
Principal Vector 2 in e basis => 0.645372, -0.208187, 0.734951
Principal Vector 3 in e basis => -0.16206, 0.902924, 0.398076
```

The eigenvectors vary as they should. P_ee is the principal basis wrt the e-e basis, while P_EE is the principal basis wrt the E-E basis.

Problem 4

In [245]:
```
# bl.tran_3x3_vm -> located in "blfunc"
T_E_vm = bl.tran_3x3_vm(T_EE)
print T_E_vm
```

```
[[  7.   ]
 [  4.   ]
 [  6.   ]
 [  2.828]
 [  7.071]
 [ 15.556]]
```

In [246]:
```
T_e_vm = bl.tran_3x3_vm(T_ee)
print T_e_vm
```

```
[[ 11.773]
 [ -1.348]
 [  6.575]
 [  8.08 ]
 [ -8.84 ]
 [ -8.597]]
```

Problem 5

5(i) bl.tran_a_A() -> located in "blfunc"

5(ii)

In [247]:
```
A_eE = bl.tran_a_A(a_eE)
A_Ee = np.transpose(A_eE)

T_e_vm_ck = A_eE.dot(T_E_vm)
T_E_vm_ck = A_Ee.dot(T_e_vm)

# verify T_e is correct
print T_e_vm_ck
```

```
[[ 11.773]
 [ -1.348]
 [  6.575]
 [  8.08 ]
 [ -8.84 ]
 [ -8.597]]
```

In [248]:
```
# T_E_vm in V-m notation (check)
print T_E_vm_ck
```

```
[[  7.   ]
 [  4.   ]
 [  6.   ]
 [  2.828]
 [  7.071]
 [ 15.556]]
```

In [249]:
```
# Check that A_eE (6x6) is orthogonal
orthogonal_ck = A_eE.dot(A_Ee)
print orthogonal_ck
```

```
[[  1.000e+00  -4.550e-18   8.990e-18   1.252e-16   6.939e-17   6.939e-18]
 [ -4.550e-18   1.000e+00  -4.322e-17   1.003e-16   8.327e-17   2.776e-17]
 [  8.990e-18  -4.322e-17   1.000e+00  -3.823e-17   0.000e+00  -1.110e-16]
 [  1.252e-16   1.003e-16  -3.823e-17   1.000e+00  -5.551e-17  -8.327e-17]
 [  6.939e-17   8.327e-17   0.000e+00  -5.551e-17   1.000e+00   2.776e-17]
 [  6.939e-18   2.776e-17  -1.110e-16  -8.327e-17   2.776e-17   1.000e+00]]
```

Problem 6

functions for calculating the deviatoric and spherical projects are shown in "bl.func"

In [250]:
```
# P spherical:  componenets of the V-M version of the fourth -order spherical projection
P_dv = P_dv()
P_sp = P_sp()
```

In [251]:
```
# T_dev in VM notation
T_E_dv = P_dv.dot(T_E_vm)
print T_E_dv
```

```
[[  1.333]
 [ -1.667]
 [  0.333]
 [  2.828]
 [  7.071]
 [ 15.556]]
```

In [252]:
```
# T_dev in 3x3 form
bl.tran_vm_3x3(T_E_dv)
```

Out[252]:
```
array([[  1.333,   2.   ,  11.   ],
       [  2.   ,  -1.667,   5.   ],
       [ 11.   ,   5.   ,   0.333]])
```

In [219]:
```
T_E_sp = P_sp.dot(T_E_vm)
bl.tran_vm_3x3(T_E_sp)
```

Out[219]:
```
array([[ 5.667,  0.   ,  0.   ],
       [ 0.   ,  5.667,  0.   ],
       [ 0.   ,  0.   ,  5.667]])
```

Problem 7

The deviatoric and spherical projectors are isotropic; therefore, the components of the two tensors are the same in any basis.

In [223]:
```
# deviatoric projector components in the e-e basis (VM)
P_dv_ck = A_eE.dot(P_dv).dot(A_Ee)
print P_dv_ck
```

```
[[  6.667e-01  -3.333e-01  -3.333e-01   1.110e-16   2.776e-17   2.776e-17]
 [ -3.333e-01   6.667e-01  -3.333e-01   8.327e-17   5.551e-17   2.776e-17]
 [ -3.333e-01  -3.333e-01   6.667e-01  -1.110e-16   0.000e+00  -8.327e-17]
 [  1.110e-16   6.939e-17  -8.327e-17   1.000e+00  -1.110e-16  -5.551e-17]
 [  4.163e-17   0.000e+00   0.000e+00  -8.327e-17   1.000e+00  -5.551e-17]
 [  0.000e+00   0.000e+00  -1.110e-16  -1.110e-16   2.776e-17   1.000e+00]]
```

In [221]:
```
# spherical projector componenets in the e-e basis (VM)
P_sp_ck = A_eE.dot(P_sp).dot(A_Ee)
print P_sp_ck
```

```
[[  3.333e-01   3.333e-01   3.333e-01   2.082e-17   3.469e-17   0.000e+00]
 [  3.333e-01   3.333e-01   3.333e-01   2.776e-17   1.388e-17   0.000e+00]
 [  3.333e-01   3.333e-01   3.333e-01   2.082e-17   3.469e-17   0.000e+00]
 [  2.776e-17   2.776e-17   2.776e-17   2.311e-33   1.541e-33   0.000e+00]
 [  2.776e-17   2.776e-17   2.776e-17   2.311e-33   1.541e-33   0.000e+00]
 [  0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00]]
```