```python
# -*- coding: utf-8 -*-
from __future__ import division

__author__ = 'Lampe'


# Check Path for Loading Modules
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

########################
# print '1'
print '*****************************************'
########################
# Create transformation matrix between E and arbitrary orthonormal basis
a_eE = bl.orth_basis()
a_Ee = np.transpose(a_eE)
print 'Prob. 1: transformation matrix a_eE'
print a_eE

i = np.dot(a_Ee, a_eE)
print 'i'
print i

########################
# Prob. 2
########################
print '*****************************************'
print 'Prob. 2.i'
########################
# arbitrary symmetric tensor
T_EE = np.array([[7, 2, 11],
                 [2, 4, 5],
                 [11, 5, 6]])
print 'T_EE '
print T_EE
T_ee = a_eE.dot(T_EE).dot(a_Ee)
print'T_ee'
```

```
46 print T_ee
47
48 # check that transformations are done correctly
49 T_EE_ck = a_Ee.dot(T_ee).dot(a_eE)
50 print 'T_EE_ck -> verify back transformation'
51 print T_EE_ck
52
53 print '2.ii'
54 ######################
55 # spherical components in the E-E basis
56 T_EE_sp = np.multiply(np.trace(T_EE) / 3, np.eye(3, 3))
57 print 'T_EE_sp'
58 print T_EE_sp
59
60 # deviatoric components in the E-E basis
61 T_EE_dev = T_EE - T_EE_sp
62 print 'T_EE_dev'
63 print T_EE_dev
64
65 ######################
66 # Prob. 3
67 ######################
68 print '*****************************************'
69 print '3.i'
70 ######################
71 eigval, eigvec = LA.eig(T_EE)
72
73 l1_EE, l2_EE, l3_EE = eigval
74 print 'Eigenvalue 1 = %G + %Gi' % (l1_EE.real, l1_EE.imag)
75 print 'Eigenvalue 2 = %G + %Gi' % (l2_EE.real, l2_EE.imag)
76 print 'Eigenvalue 3 = %G + %Gi' % (l3_EE.real, l3_EE.imag)
77
78 P1_E, P2_E, P3_E = eigvec[:, 0], eigvec[:, 1], eigvec[:, 2]
79
80 print 'Principal Vector 1 in E basis => %G, %G, %G' % (P1_E[0], P1_E[1],
 … P1_E[2])
81 print 'Principal Vector 2 in E basis => %G, %G, %G' % (P2_E[0], P2_E[1],
 … P2_E[2])
82 print 'Principal Vector 3 in E basis => %G, %G, %G' % (P3_E[0], P3_E[1],
 … P3_E[2])
83
84 print '3.ii'
85 ######################
86 P_EE = eigvec
87
```

```
88  print 'orthonormal Principal basis wrt E-E:'
89  print P_EE
90
91  print '3.iii'
92  ######################
93  a_EP = P_EE
94  print 'transformation matrix between the E-P basis(a_EP):'
95  print a_EP
96
97  a_PE = np.transpose(a_EP)
98  print 'a_PE'
99  print a_PE
100
101 T_PP = a_PE.dot(T_EE).dot(a_EP)
102 print 'components of T in the Principal basis:'
103 print T_PP
104 print 'yes, the diagonal components should be the eigenvalues with all
    … other components = zero'
105
106 print '3.iv'
107 ######################
108 T_PP_inv = LA.inv(T_PP)
109 # T_EE_inv = np.linalg.multi_dot(a_EP, T_PP_inv, a_PE)
110 T_EE_inv = a_EP.dot(T_PP_inv).dot(a_PE)
111
112 print "check if inverse is correct -> should get Identity matrix"
113 print np.dot(T_EE, T_EE_inv)
114
115 print '3.v'
116 ######################
117 eigval, eigvec = LA.eig(T_ee)
118
119 l1_ee, l2_ee, l3_ee = eigval
120 print 'Eigenvalue 1 = %G + %Gi' % (l1_ee.real, l1_ee.imag)
121 print 'Eigenvalue 2 = %G + %Gi' % (l2_ee.real, l2_ee.imag)
122 print 'Eigenvalue 3 = %G + %Gi' % (l3_ee.real, l3_ee.imag)
123
124 P1_e, P2_e, P3_e = eigvec[:, 0], eigvec[:, 1], eigvec[:, 2]
125
126 print 'Principal Vector 1 in e basis => %G, %G, %G' % (P1_e[0], P1_e[1],
    … P1_e[2])
127 print 'Principal Vector 2 in e basis => %G, %G, %G' % (P2_e[0], P2_e[1],
    … P2_e[2])
128 print 'Principal Vector 3 in e basis => %G, %G, %G' % (P3_e[0], P3_e[1],
    … P3_e[2])
```

```
129 P_ee = eigvec
130
131 print 'Eigenvectors off by a const.'
132
133 ######################
134 # Prob. 4
135 ######################
136 print '*****************************************'
137 print '4'
138 ######################
139 T_E_vm = bl.tran_3x3_vm(T_EE)
140 print "T_EE in v-m notation"
141 print T_E_vm
142
143 ######################
144 # Prob. 5
145 ######################
146 print '*****************************************'
147 print '5'
148 ######################
149 A_eE = bl.tran_a_A(a_eE)
150 A_Ee = np.transpose(A_eE)
151
152 T_e_vm = A_eE.dot(T_E_vm)
153 T_E_vm = A_Ee.dot(T_e_vm)
154
155 print "T_e_vm in v-m notation"
156 print T_e_vm
157 print "T_E_vm in V-m notation"
158 print T_E_vm
159
160 orth_ck = A_eE.dot(A_Ee)
161 print "Check that A_eE is orthogonal"
162 print orth_ck
163
164
```