Matrix Analysis Using Python

1. **Sketch the vectors a1 = (1, 1, 1) T , a2 = (0, 1, 2) T , b = (6, 0, 0) T in the 3-D plane.**

```
import numpy as np
import matplotlib.pyplot as plt
V = np.array([[1,1,1],[0,1,2],[6,0,0]])
origin = [0], [0] # origin point
plt.quiver(*origin, V[:,0], V[:,1], color=['r','b','g'], scale=21)
plt.show()
```

2. **Solve the matrix equation Ax = b where A = [a 1 a 2] using row reduction.**

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import inv
from numpy import linalg as LA
'''
a1 = [1, 1, 1]
a2 = [0, 1, 2]
b = [6, 0, 0]
A=np.array([a1,a2])
'''
a=np.array([[3,1],[1,2]])
b=np.array([9,8])
x=np.linalg.solve(a,b)
print(x)
'''np.allclose(np.dot(a,x),b)'''
```

3. **Least Squares**

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import inv
from numpy import linalg as LA
# Creating matrix
A=np.matrix( '1 0; 1 1; 1 2 ' )
# Creating vector
b=np.matrix( ' 6; 0; 0 ' )
#P = inv((A'A)A'
```

```
P=np.dot(inv(np.dot(np.transpose(A),A)),np.transpose(A))
#x_ls=Pb
x_ls=np . dot (P , b )
x=np.matrix ( ' 5; -5 ' )
# ||b-Ax_ls||
exact_ls_metric=(LA.norm(b-np.dot(A,x_ls)))**2
# ||b-Ax||
random_ls_metric= (LA.norm(b-np.dot(A, x )))**2
print ( x_ls )
print (x)
print ( exact_ls_metric )
print ( random_ls_metric )
```

4. **Moore Penrose**

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import inv
from numpy import linalg as LA
A=np.matrix('1 0 ;1 1; 1 2')
b=np.matrix('6;0;0')
U,s,V=LA.svd(A)
mn=A.shape
S=np.zeros(mn)
print(S)
Sinv=S.T
print(Sinv)
S[:2,:2]=np.diag(s)
print('S',S)
print('s',s)
print(U.dot(S).dot(V))
sinv=1./s
print('s',s)
print('sinv',sinv)
Sinv[:2,:2]=np.diag(sinv)
print(Sinv)
Aplus=V.T.dot(Sinv).dot(U.T)
x_ls=Aplus.dot(b)
print(x_ls)
```

5. **I) Obtain the eigenvalues and eigenvectors of G. G=[[1 1;-2 4]]**

   II) Find f(G). This is known as the Cayley-Hamilton Theo− rem.
   III) Stack the eigenvalues of G in a diagonal matrix ? and the corresponding eigenvectors in a matrix F. Find F?F^{-1}

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import det
from numpy.linalg import inv
from numpy import linalg as LA
#eigen value and vector
g=np.matrix('1 1;-2 4')
x,y=LA.eig(g)
print('Eigen Values',x)
print('Eigen Vectors',y)
#cayley Hamilton theorem
o=g.dot(np.identity(2))-g
f=det(o)
print(f,"This proves Cayley Hamilton")
#Eigen value decomposition
z=np.diag(x)
F=y
k=inv(F)
h=F.dot(z.dot(k))
print(h)
```

6. **P is eigen vector matrix as IT has to satisfy C=PDP^{-1} where C = [[37 9 ; 9 13]] and also find PP^T and P^{T}P. P is known as an orthogonal matrix.**

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import inv
from numpy import linalg as LA
#eigen value and vector
c=np.matrix('37 9;13 9')
x,y=LA.eig(c)
print('Eigen Values',x)
print('Eigen Vectors',y)
#Eigen value decomposition
```

```
d=np.diag(x)
P=y
k=inv(P)
h=P.dot(d.dot(k))
print(h)
print("Hence Proved")
#14 orthogonal matrix
prod=P.dot(P.T)
print(P)
prod1=P.T.dot(P)
print(prod1)
```

7. **B=[[4 11 14; 8 7 -2]]**

   Find B^TB and BB^T Note that C = 1/9(BB^T). Obtain the eigenvalues and eigenvectors of B^TB. Verify eigenvalue decomposition and Cayley-Hamilton theorem for B^TB.

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import inv
from numpy import linalg as LA
B=np.matrix('4 11 14; 8 7 -3')
prod=B.T.dot(B)
print(prod)
prod1=B.dot(B.T)
print(prod1)
print(prod1/9)
#eigen val and vect
x,y=LA.eig(prod1)
#eig val decomposition
d=np.diag(x)
f=y
f_=inv(f)
z=f.dot(d.dot(f_))
print(z)
#cayley hamilton theorem
o=prod.dot(np.identity(3))-prod
f=det(o)
print(f,"This proves Cayley Hamilton")
```

8. **Singular Value decomposition**

```
import numpy as np
import math as m
B=np.matrix('4 11 14;8 7 -2')
b=np.dot(B.T,B)
print(b)
len=3
x=np.zeros(3)
for i in range(len):
    sq=0
    for j in range(len):
    sq=sq+b[i,j]**2
    x[i]=m.sqrt(sq)
sq=np.zeros((3,3))
print('square values')
for i in range(len):
    sq[i]=b[i]/x[i]
print(sq)
```

or

```
import numpy as np
from numpy import linalg as LA
B=np.matrix('4 11 14;8 7 -2')
b=np.dot(B.T,B)
lam,V=LA.eig(b)
print(lam)
```

9. **Orgthogonal Diagonalisation**

```
import numpy as np
import math as m
from numpy import linalg as LA
B=np.matrix('4 11 14;8 7 -2')
b=np.dot(B.T,B)
lam,v=LA.eig(b)
print(v)
print(np.dot(v,v.T))
```

10. **Find the singular values of B^TB. The singular values are obtained by taking the square roots of its eigenvalues.**

```
import numpy as np
import math as m
from numpy import linalg as LA
B=np.matrix('4 11 14;8 7 -2')
b=np.dot(B.T,B)
lam,V=LA.eig(b)
sing_val=np.zeros(3)
for i in range(3):
sing_val[i]=m.sqrt(int(lam[i]))
print('singular values')
print(sing_val)

E=np.zeros((3,3))
E=np.diag(sing_val)
print('E matrix')
print(E)
```

11. **I) Stack the singular values of B^TB diagonally to obtain a matrix E.**

II) Obtain the matrix BV. Verify if the columns of this matrix are orthogonal.
III) Extend the columns of BV if necessary, to obtain an or− thogonal matrix U.
IV) Find U−VT. Comment.

```
import numpy as np
import math as m
from numpy import linalg as LA
B=np.matrix('4 11 14;8 7 -2')
b=np.dot(B.T,B)
lam,v=LA.eig(b)
print(v)
#print(np.dot(v,v.T))
sing_val=np.zeros(3)
for i in range(3):
sing_val[i]=m.sqrt(int(lam[i]))
```

```
print('singular values')
print(sing_val)
E=np.zeros((3,3))
E=np.diag(sing_val)
print('E matrix')
print(E)
#prob24
bv=B.dot(v)
#both bv1 and bv2 are orthogonal
bv1=bv.dot(bv.T)
print(bv1)
bv2=bv.T.dot(bv)
print(bv2)
#consider any of the matrix as U
#prob 26
U=bv2
z=U.dot(E.dot(v.T))
print(z)
```