hw1_code

September 16, 2021

[ ]:
```
import numpy as np
import scipy.linalg as linalg
from scipy.spatial.transform import Rotation as Rot
```

1 Problem 1

[ ]:
```
def swaprow(A, i, j):
    temp = np.copy(A[j])
    A[j] = np.copy(A[i])
    A[i] = temp
    return A

def swapele(A, idx1, idx2):
    temp = A[idx2]
    A[idx2] = A[idx1]
    A[idx1] = temp
    return A

def ldu(A):
    #  only row interchanges
    n = A.shape[0]
    L = np.identity(n)
    P = np.identity(n)
    for i in range(n):
        if A[i][i] == 0: # swap row
            # print(A)
            col = A[i+1:, i]
            swap_row_idx = np.nonzero(col)[0][0]+i+1
            A = swaprow(A, i, swap_row_idx)
            P = swaprow(P, i, swap_row_idx)
            L = swaprow(L, i, swap_row_idx)
            L = swapele(L, (i, i), (i, swap_row_idx))
            L = swapele(L, (swap_row_idx, i), (swap_row_idx, swap_row_idx))
        for j in range(i+1, n):
            row = A[j]
            ratio = row[i] / A[i][i]
            A[j] = row - A[i] * ratio
```

```
            L[j, i] += ratio
    D = np.diag(np.diag(A))
    U = (A.T / np.diag(A)).T
    return P, L, D, U
```

1.1 Examples

[ ]:
```
# example 1
A_flat = np.array([1, -2, 1, 1, 2, 2, 2, 3, 4]).astype(float)
# A_flat = np.array([1, 1, 0, 1, 1, 2, 4, 2, 3]).astype(float)

n = np.sqrt(A_flat.shape[0]).astype(int)
A = A_flat.reshape((n, n))

P, L, D, U = ldu(A)

print("A", A)
print("P", P)
print("L", L)
print("D", D)
print("U", U)
```

```
A [[ 1.   -2.    1.  ]
 [ 0.    4.    1.  ]
 [ 0.    0.    0.25]]
P [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
L [[1.   0.   0.  ]
 [1.   1.   0.  ]
 [2.   1.75 1.  ]]
D [[1.   0.   0.  ]
 [0.   4.   0.  ]
 [0.   0.   0.25]]
U [[ 1.   -2.    1.  ]
 [ 0.    1.    0.25]
 [ 0.    0.    1.  ]]
```

[ ]:
```
# example 2
A_flat = np.array([1, 1, 0, 1, 1, 2, 4, 2, 3]).astype(float)

n = np.sqrt(A_flat.shape[0]).astype(int)
A = A_flat.reshape((n, n))

P, L, D, U = ldu(A)

print("A", A)
```

```
print("P", P)
print("L", L)
print("D", D)
print("U", U)
```

```
A [[ 1.  1.  0.]
 [ 0. -2.  3.]
 [ 0.  0.  2.]]
P [[1. 0. 0.]
 [0. 0. 1.]
 [0. 1. 0.]]
L [[1. 0. 0.]
 [4. 1. 0.]
 [1. 0. 1.]]
D [[ 1.  0.  0.]
 [ 0. -2.  0.]
 [ 0.  0.  2.]]
U [[ 1.   1.   0. ]
 [-0.   1.  -1.5]
 [ 0.   0.   1. ]]
```

2 Problem 2

[ ]:
```
def svd(A):
    U, s, VT = linalg.svd(A)

    sigma = np.zeros(A.shape)
    for i, sig in enumerate(s):
        sigma[i, i] = sig

    print("sigma", sigma)
    print("U", U)
    print("VT", VT)
```

2.1 A_1

[ ]:
```
A_flat = np.array([7, 6, 1, 4, 5, 1, 7, 7, 7]).astype(float)

n = np.sqrt(A_flat.shape[0]).astype(int)
A = A_flat.reshape((n, n))

svd(A)
```

```
sigma [[16.05699963  0.          0.        ]
 [ 0.          4.02789209  0.        ]
 [ 0.          0.          0.97408829]]
U [[-0.55358553  0.64368823  0.52840186]
```

```
 [-0.38998772  0.36025094 -0.84742483]
 [-0.73583466 -0.67519235  0.0516008 ]]
VT [[-0.65926963 -0.64908106 -0.37954886]
 [ 0.30300586  0.23263722 -0.92415765]
 [ 0.68815042 -0.7242746   0.04330471]]
```

2.2 A_2

[ ]:
```
A_flat = np.array([12,12,0,0,3,0,-2,0,0,1,-1,0,0,0,0,1,0,0,1,1]).astype(float)

n = np.sqrt(A_flat.shape[0]).astype(int)
A = A_flat.reshape((5, 4))

svd(A)
```

```
sigma [[17.12140667  0.          0.          0.        ]
 [ 0.          3.00170074  0.          0.        ]
 [ 0.          0.          1.60502203  0.        ]
 [ 0.          0.          0.          1.12744436]
 [ 0.          0.          0.          0.        ]]
U [[-9.90941122e-01  1.25878924e-01 -1.32838739e-02  3.11952653e-02
  -3.22580645e-02]
 [-1.27647037e-01 -9.43895873e-01  2.75850620e-01  4.89157065e-03
   1.29032258e-01]
 [-4.17239152e-02 -9.48902857e-02 -5.11590597e-01 -7.60056640e-01
   3.87096774e-01]
 [ 3.49210517e-06  3.59486219e-02  4.35899861e-01 -6.26471002e-01
  -6.45161290e-01]
 [ 1.02019257e-03  2.87955914e-01  6.87019902e-01 -1.69855569e-01
   6.45161290e-01]]
VT [[-7.16894051e-01 -6.96964777e-01  1.74073421e-02  5.97897528e-05]
 [-4.40130662e-01  4.71618234e-01  7.56450474e-01  1.07907005e-01]
 [ 4.16284238e-01 -4.18060981e-01  4.03053197e-01  6.99628880e-01]
 [ 3.45043987e-01 -3.42113076e-01  5.14808493e-01 -7.06311196e-01]]
```

2.3 A_3

[ ]:
```
A_flat = np.array([7,6,4,0,3,3,7,3,1]).astype(float)

n = np.sqrt(A_flat.shape[0]).astype(int)
A = A_flat.reshape((n, n))

svd(A)
```

```
sigma [[1.26839208e+01 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 4.13740891e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 4.69957459e-16]]
U [[-0.78940534 -0.20858061 -0.57735027]
```

```
 [-0.21406656 -0.78793539  0.57735027]
 [-0.57533878  0.57935477  0.57735027]]
VT [[-0.75317475 -0.56013028 -0.34493749]
 [ 0.62730544 -0.45372009 -0.63295021]
 [ 0.19802951 -0.69310328  0.69310328]]
```

3 Problem 5

[ ]:
```python
def findtrans(P, Q):
    # input is 2 matrices with column vector in 3D space
    assert P.shape[0] == 3 and Q.shape[0] == 3
    assert P.shape[1] == Q.shape[1]
    n = P.shape[1]
    p_ = np.mean(P, axis=1).reshape((3, 1))
    q_ = np.mean(Q, axis=1).reshape((3, 1))
    X = P - p_
    Y = Q - q_
    S = X @ Y.T
    U, s, VT = linalg.svd(S)
    V = VT.T
    det = linalg.det(V @ U.T)
    if det > 0:
        sign = 1
    else:
        sign = -1
    M = np.identity(3)
    M[-1, -1] = sign
    R = V @ M @ U.T
    t = q_ - R @ p_
    return t, R
```

[ ]:
```python
n = 10
P = np.random.randint(0, 10, (3, n))
t = np.random.randint(0, 10, (1, 3)).T

R = Rot.random().as_matrix()
Q = R @ P + t
Q = Q + 0.2*np.random.normal(size=(3, n))
t_, R_ = findtrans(P, Q)

calc_error = lambda x: np.mean(np.square(x))
trans_error = calc_error(t_-t)
rot_error = calc_error(R_-R)

print("Translation error: {:2f}\n Rotation error: {:2f}".format(trans_error,␣
 ↪rot_error))
```

```
Translation error: 0.004584
 Rotation error: 0.000053
```

[ ]: