Assignment 1

Xuxin Cheng

Problem 1

I implemented $PA = LDU$ decomposition in Python. Please refer to Jupyter Notebook attached for 2 examples of code running correctly.

Problem 2

1. LDU decomposition:

$$
\begin{array}{cc}
\text{L} & \text{A'} \\
\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 7 & 6 & 1 \\ 4 & 5 & 1 \\ 7 & 7 & 7 \end{bmatrix} \\
\begin{bmatrix} 1 & 0 & 0 \\ \frac{4}{7} & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 7 & 6 & 1 \\ 0 & \frac{11}{7} & \frac{3}{7} \\ 0 & 1 & 6 \end{bmatrix} \\
\begin{bmatrix} 1 & 0 & 0 \\ \frac{4}{7} & 1 & 0 \\ 1 & \frac{7}{11} & 1 \end{bmatrix} & \begin{bmatrix} 7 & 6 & 1 \\ 0 & \frac{11}{7} & \frac{3}{7} \\ 0 & 0 & \frac{63}{11} \end{bmatrix}
\end{array}
$$

$$
P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, L = \begin{bmatrix} 1 & 0 & 0 \\ \frac{4}{7} & 1 & 0 \\ 1 & \frac{7}{11} & 1 \end{bmatrix}, D = \begin{bmatrix} 7 & 0 & 0 \\ 0 & \frac{11}{7} & 0 \\ 0 & 0 & \frac{63}{11} \end{bmatrix}, U = \begin{bmatrix} 1 & \frac{6}{7} & \frac{1}{7} \\ 0 & 1 & \frac{3}{11} \\ 0 & 0 & 1 \end{bmatrix}
$$

SVD decomposition:

$$
U = \begin{bmatrix} -0.55 & 0.64 & 0.53 \\ -0.39 & 0.36 & -0.85 \\ -0.74 & -0.68 & 0.05 \end{bmatrix}, \Sigma = \begin{bmatrix} 16.06 & 0.00 & 0.00 \\ 0.00 & 4.03 & 0.00 \\ 0.00 & 0.00 & 0.97 \end{bmatrix}, V^T = \begin{bmatrix} -0.66 & -0.65 & -0.38 \\ 0.30 & 0.23 & -0.92 \\ 0.69 & -0.72 & 0.04 \end{bmatrix}
$$

2. LDU decompostion:

$$\begin{array}{cc} \text{L} & \text{A'} \\ \begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\end{bmatrix} & \begin{bmatrix}12&12&0&0\\3&0&-2&0\\0&1&-1&0\\0&0&0&1\\0&0&1&1\end{bmatrix} \end{array}$$

$$\begin{bmatrix}1&0&0&0&0\\\frac{1}{4}&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\end{bmatrix} \quad \begin{bmatrix}12&12&0&0\\0&-3&-2&0\\0&1&-1&0\\0&0&0&1\\0&0&1&1\end{bmatrix}$$

$$\begin{bmatrix}1&0&0&0&0\\\frac{1}{4}&1&0&0&0\\0&-\frac{1}{3}&1&0&0\\0&0&0&1&0\\0&0&0&0&1\end{bmatrix} \quad \begin{bmatrix}12&12&0&0\\0&-3&-2&0\\0&0&-\frac{5}{3}&0\\0&0&0&1\\0&0&1&1\end{bmatrix}$$

$$\begin{bmatrix}1&0&0&0&0\\\frac{1}{4}&1&0&0&0\\0&-\frac{1}{3}&1&0&0\\0&0&0&1&0\\0&0&-\frac{3}{5}&1&1\end{bmatrix} \quad \begin{bmatrix}12&12&0&0\\0&-3&-2&0\\0&0&-\frac{5}{3}&0\\0&0&0&1\\0&0&0&0\end{bmatrix}$$

$$P = \begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\end{bmatrix}, L = \begin{bmatrix}1&0&0&0&0\\\frac{1}{4}&1&0&0&0\\0&-\frac{1}{3}&1&0&0\\0&0&0&1&0\\0&0&-\frac{3}{5}&1&1\end{bmatrix}, D = \begin{bmatrix}12&0&0&0&0\\0&-3&0&0&0\\0&0&-\frac{5}{3}&0&0\\0&0&0&1&0\\0&0&0&0&0\end{bmatrix},$$

$$U = \begin{bmatrix}1&1&0&0\\0&1&\frac{2}{3}&0\\0&0&1&0\\0&0&0&1\\0&0&0&0\end{bmatrix}$$

SVD decomposition:

$$U = \begin{bmatrix}-0.99&0.13&-0.01&0.03&-0.03\\-0.13&-0.94&0.28&0.00&0.13\\-0.04&-0.09&-0.51&-0.76&0.39\\0.00&0.04&0.44&-0.63&-0.65\\0.00&0.29&0.69&-0.17&0.65\end{bmatrix}, \Sigma = \begin{bmatrix}17.12&0.00&0.00&0.00\\0.00&3.00&0.00&0.00\\0.00&0.00&1.61&0.00\\0.00&0.00&0.00&1.13\\0.00&0.00&0.00&0.00\end{bmatrix},$$

$$V^T = \begin{bmatrix} -0.72 & -0.70 & 0.02 & 0.00 \\ -0.44 & 0.47 & 0.76 & 0.11 \\ 0.42 & -0.42 & 0.40 & 0.70 \\ 0.35 & -0.34 & 0.51 & -0.71 \end{bmatrix}$$

3. LDU decomposition:

$$\begin{array}{cc} \text{L} & \text{A'} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 7 & 6 & 4 \\ 0 & 3 & 3 \\ 7 & 3 & 1 \end{bmatrix} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 7 & 6 & 4 \\ 0 & 3 & 3 \\ 0 & -3 & -3 \end{bmatrix} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & -1 & 1 \end{bmatrix} & \begin{bmatrix} 7 & 6 & 4 \\ 0 & 3 & 3 \\ 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & -1 & 1 \end{bmatrix}, D = \begin{bmatrix} 7 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix}, U = \begin{bmatrix} 1 & \frac{6}{7} & \frac{4}{7} \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

SVD decomposition:

$$U = \begin{bmatrix} -0.79 & -0.21 & -0.58 \\ -0.21 & -0.79 & 0.58 \\ -0.58 & 0.58 & 0.58 \end{bmatrix}, \Sigma = \begin{bmatrix} 12.68 & 0.00 & 0.00 \\ 0.00 & 4.14 & 0.00 \\ 0.00 & 0.00 & 0.00 \end{bmatrix}, V^T = \begin{bmatrix} -0.75 & -0.56 & -0.34 \\ 0.63 & -0.45 & -0.63 \\ 0.20 & -0.69 & 0.69 \end{bmatrix}$$

Problem 3

(a) A is a square matrix and has full rank. The system has a unique exact solution.

$$x = \bar{x} = V\Sigma^{-1}U^T = \begin{bmatrix} -1 \\ 2 \\ 1 \end{bmatrix}$$

(b) A has rank 2, is not full rank. b is in the column space of A.

$$\bar{x} = V\Sigma^{-1}U^T = \begin{bmatrix} \frac{70}{51} \\ -\frac{31}{102} \\ -\frac{71}{102} \end{bmatrix}$$

In the SVD decomposition of $A = U\Sigma V^T$, The last row $v_3^T \approx [0.2\ -0.69\ 0.69]$ in V^T is the basis vector of the null space of A. Thus,

$$x = \bar{x} + x'$$

where $x' = a \cdot v_3^T$ is a vector in the null space of A. $a \in \mathbb{R}$.

(c)

$$\bar{x} = \begin{bmatrix} \frac{70}{51} \\ -\frac{31}{102} \\ -\frac{71}{102} \end{bmatrix}$$

b is not in the column space of A. $\bar{x}$ is the least-square solution of $Ax = b||$ (i.e. $\bar{x}$ minimizes $||Ax - b||$).

(b) and (c) have the same SVD solution. In (b), b is in the column space of A, so $Ax = b$ has infinite solutions with the form $x = \bar{x} + v$, where v is an arbitrary vector in the null space of A. However in (c), b is not in the column space of A, the SVD solution $\bar{x}$ is the least-square solution.

Problem 4

(a) Let $v \in \mathbb{R}^n \perp u$.

$$\begin{aligned} Au &= (I - uu^T)u \\ &= u - uu^T u \\ &= 0 \\ Av &= (I - uu^T)v \\ &= v \end{aligned}$$

Any vector $x \in \mathbb{R}^n$ can be written as $x = au + bv$, where a, b are coefficients. Thus A transforms the portion of vectors which is parallel to u to the origin (zero vector), and the portion perpendicular to u remain unchanged.

(b) From (a) we know that eigenvalue 0 corresponds to eigenvector u and eigenvalue 1 corresponds to eigenvector v.

(c) The null space of A is the span of vector u.

(d)

$$\begin{aligned} A^2 &= (I - uu^T)(I - uu^T) \\ &= I - 2uu^T + uu^T uu^T \\ &= I - uu^T \\ &= A \end{aligned}$$

Probelem 5

Suppose the rotation matrix we want to get is $R \in \mathbb{R}^{3\times 3}$. The translation vector is $t \in \mathbb{R}^3$. We want to get R and t, s.t.

$$f(R, t) = \sum_{i=0}^{n} ||Rp_i + t - q_i||^2$$

is minimized. Let's make $\frac{\partial f}{\partial t} = 0$ to compute the optimal t,

$$\begin{aligned}\frac{\partial f}{\partial t} &= 0 \\ &= 2(R\sum_{i=0}^{n} p_i + nt - \sum_{i=0}^{n} q_i) \\ t &= \frac{\sum_{i=0}^{n} q_i}{n} - \frac{R\sum_{i=0}^{n} p_i}{n}\end{aligned}$$

Let $\bar{p} = \frac{\sum_{i=0}^{n} p_i}{n}$, $\bar{q} = \frac{\sum_{i=0}^{n} q_i}{n}$,

$$t^* = \bar{q} - R\bar{p}$$

Now we have computed the best translation vector. Let's put t^* back in $f(R, t)$,

$$f(R, t^*) = \sum_{i=0}^{n} ||R(p_i - \bar{p}) - (q_i - \bar{q})||^2$$

Denote $x_i = p_i - \bar{p}$, $y_i = q_i - \bar{q}$, our objective becomes,

$$\min_R \sum_{i=0}^{n} ||Rx_i - y_i||^2$$

where R is a rotation matrix. Let $x'_i = Rx_i \in \mathbb{R}^3$, we can write the objective as:

$$\min_R \sum_{i=0}^{n} ||x'_i - y_i||^2$$

which means we want to minimize the sum of distance between points x'_i and y_i. In other words, we want to maximize the sum of projections from vectors x'_i to y_i (inner product of x'_i and y_i). We can rewrite the objective as:

$$\max_R \sum_{i=0}^{n} y_i^T x'_i \Leftrightarrow \max_R \sum_{i=0}^{n} y_i^T R x_i$$

Let $X = [x_1, x_2, ..., x_n]^T$, $Y = [y_1, y_2, ..., y_n]^T$

$$\begin{aligned}\sum_{i=0}^{n} y_i^T R x_i &= Tr(\begin{bmatrix} y_1^T Rx_1 & y_1^T Rx_2 & \dots & y_1^T Rx_n \\ y_2^T Rx_1 & y_2^T Rx_2 & \dots & y_2^T Rx_n \\ \vdots & \vdots & \ddots & \vdots \\ y_n^T Rx_1 & y_n^T Rx_2 & \dots & y_n^T Rx_n \end{bmatrix}) \\ &= Tr(\begin{bmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_n^T \end{bmatrix} R \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}) \\ &= Tr(Y^T(RX)) \\ &= Tr((RX)Y^T) \\ &= Tr(RXY^T)\end{aligned}$$

Let the SVD decomposition be $XY^T = U\Sigma V^T$, we have:

$$\begin{aligned} Tr(RXY^T) &= Tr(RU\Sigma V^T) \\ &= Tr(\Sigma V^T RU) \end{aligned}$$

R is a rotation matrix, so R is an orthogonal matrix. U and V_T are also orthogonal matrices. So $M = V^T RU$ is also an orthogonal matrix.

$$\begin{aligned} Tr(\Sigma M) &= Tr(\begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma 2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}) \\ &= \sum_{i=1}^{3} \sigma_i m_{ii} \end{aligned}$$

Since $M = [m_1 \; m_2 \; m_3]$ is an orthogonal matrix,

$$||m_j||^2 = 1 \Longrightarrow \sum_{i=0}^{3} m_{ij}^2 = 1 \Longrightarrow m_{ij}^2 \leq 1 \Longrightarrow |m_{ij}| \leq 1$$

Since $\sigma_1, \sigma_2, \sigma_3 \geq 0$, $Tr(\Sigma M)$ can achieve maximum if $m_{11} = m_{22} = m_{33} = 1$ (M is an identity matrix).

$$\begin{aligned} M = V^T RU &= \mathbb{I} \\ R &= VU^T \end{aligned}$$

So far we have found the orthogonal matrix to maximize $\sum_{i=0}^{n} y_i^T R x_i$. However, rotation matrix requires $det(R) = 1$. When $det(VU^T) = -1$,

$$R = VMU^T$$

$$det(R) = det(VMU^T) = det(VU^T)det(M) = -det(M) = 1 \Rightarrow det(M) = -1$$

We choose $M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$

$$R = V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} U^T$$

Combining the 2 situations discussed above, we can get the optimal R^*

$$\begin{aligned} R^* &= V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & sign(det(VU^T)) \end{bmatrix} U^T \\ t^* &= \bar{q} - R\bar{p} \end{aligned}$$

Please refer to Jupyter Notebook for some examples. Even with some Gaussian noise applied to q_i, the algorithm can still acquire the rotation matrix and translation with a small error.

References

[1] https://igl.ethz.ch/projects/ARAP/svd_rot.pdf

hw1_code

September 16, 2021

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

1 Problem 1

[ ]:
```python
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
```

[ ]:

```

```