Assignment 1

Math Fundamentals for Robotics 16-811, Fall 2018
DUE: Thursday, September 13, 2018

1. Implement the $PA = LDU$ decomposition algorithm directly yourself (in other words, do not just call a built-in Gaussian elimination algorithm in MatLab, for instance). You may assume that the matrix A is square and invertible. Also: Do not worry about column interchanges, just row interchanges. Demonstrate that your implementation works properly.

   Solution: The Matlab implementation is in file **code/q1.m**

   There are some test cases provided in the script file, according to the assumption that the matrix A is square and invertible. The decomposition method `LDU_Decompose` requires an input A, and returns the decomposed matrices P, L, D and U.

   After the decomposition, the correctness of the decomposition is also judged by checking whether PA equals to LDU using the returned values. This is finished by calling the function `check`, which returns the check status indicating whether the decomposition is correct or not. Here are some samples of the decomposition results. All the results are obtained by running the code:

   (a) $A = \begin{pmatrix} 1 & -2 & 1 \\ 1 & 2 & 2 \\ 2 & 3 & 4 \end{pmatrix}$

   The returned values are as below:

   $$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

   $$L = \begin{pmatrix} 1.0000 & 0 & 0 \\ 1.0000 & 1.0000 & 0 \\ 2.0000 & 1.7500 & 1.0000 \end{pmatrix}$$

   $$D = \begin{pmatrix} 1.0000 & 0 & 0 \\ 0 & 4.0000 & 0 \\ 0 & 0 & 0.2500 \end{pmatrix}$$

   $$U = \begin{pmatrix} 1.0000 & -2.0000 & 1.0000 \\ 0 & 1.0000 & 0.2500 \\ 0 & 0 & 1.0000 \end{pmatrix}$$

   By checking $PA = LDU$, it returns True.

(b) $A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 2 \\ 4 & 2 & 3 \end{pmatrix}$

The returned values are as below:

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

$$U = \begin{pmatrix} 1.0000 & 1.0000 & 0 \\ 0 & 1.0000 & -1.5000 \\ 0 & 0 & 1.0000 \end{pmatrix}$$

By checking $PA = LDU$, it returns True.

(c) $A = \begin{pmatrix} 0 & -4 & 1 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & -1 \end{pmatrix}$

The returned values are as below:

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$L = \begin{pmatrix} 1.0000 & 0 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 \\ 1.0000 & -0.2500 & 1.0000 & 0 \\ 0 & 0.2500 & 0.3333 & 1.0000 \end{pmatrix}$$

$$D = \begin{pmatrix} -1.0000 & 0 & 0 & 0 \\ 0 & -4.0000 & 0 & 0 \\ 0 & 0 & -0.7500 & 0 \\ 0 & 0 & 0 & 1.0000 \end{pmatrix}$$

$$U = \begin{pmatrix} 1.0000 & 0 & -1.0000 & 0 \\ 0 & 1.0000 & -0.2500 & 0 \\ 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 \end{pmatrix}$$

By checking $PA = LDU$, it returns True.

2. Compute the $PA = LDU$ decomposition and the SVD decomposition for each of the following matrices:
(You may use any method or tools you wish to solve this problem, including pre-defined routines from MATLAB, your code, and/or hand calculations. Show how you obtained your solutions.)

$$A_1 = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 2 & -1 \\ 1 & 0 & -1/2 \end{pmatrix} \quad A_2 = \begin{pmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -4 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -2 & 1 \end{pmatrix} \quad A_3 = \begin{pmatrix} 2 & 2 & 5 \\ 1 & 1 & 5 \\ 3 & 2 & 5 \end{pmatrix}$$

Solution: The Matlab implementation is in file **code/q2.m**.

For the LDU and SVD decomposition, I implemented my own code to solve the problem. For the LDU decomposition part, the code is improved based on the code in problem 1, since the input matrix A in problem 1 is assumed as square and invertible while in this problem it is not guaranteed. With the improvement, the algorithm `LDU_Decompose` can handle the situation when A is not square or not invertible. For the SVD decomposition part, the method `SVD_Decompose` requires an input matrix A, and returns the decomposed matrices U, S and V, which make $A = USV^T$. All the solutions in this problem are obtained by calling methods in code/q2.m.

To make sure that the algorithm implementation works properly, the decomposed results are also checked by calling the functions `check_LDU` and `check_SVD` for LDU and SVD decomposition respectively. The returned status shows 'True' when $PA = LDU$ and $A = USV^T$.

(a) $A_1 = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 2 & -1 \\ 1 & 0 & -1/2 \end{pmatrix}$

The returned matrices after calling function `LDU_Decompose` are as below:

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad L = \begin{pmatrix} 1.0000 & 0 & 0 \\ 0 & 1.0000 & 0 \\ 1.0000 & 0.5000 & 1.0000 \end{pmatrix}$$

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad U = \begin{pmatrix} 1.0000 & -1.0000 & 0 \\ 0 & 1.0000 & -0.5000 \\ 0 & 0 & 1 \end{pmatrix}$$

The returned matrices after calling function `SVD_Decompose` are as below:

$$U = \begin{pmatrix} 0.4427 & 0.5963 & 0.6667 \\ -0.8944 & 0.2981 & 0.3333 \\ 0 & 0.7454 & -0.6667 \end{pmatrix} \quad S = \begin{pmatrix} 2.4495 & 0 & 0 \\ 0 & 1.5000 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$V = \begin{pmatrix} 0.1826 & 0.8944 & 0.4082 \\ -0.9129 & 0 & 0.4082 \\ 0.3651 & -0.4472 & 0.8165 \end{pmatrix}$$

(b) $A_2 = \begin{pmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -4 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -2 & 1 \end{pmatrix}$

The returned matrices after calling function `LDU_Decompose` are as below:

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad L = \begin{pmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 1.0000 & 1.0000 & 0 & 0 & 0 \\ 0 & 4.0000 & 1.0000 & 0 & 0 \\ 0 & 1.0000 & 0.3333 & 1.0000 & 0 \\ 0 & 0 & 0.6667 & 1.0000 & 1.0000 \end{pmatrix}$$

$$D = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad U = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

The returned matrices after calling function `SVD_Decompose` are as below:

$$U = \begin{pmatrix} 0.2273 & -0.1886 & 0.6002 & -0.3657 & -0.6472 \\ -0.0556 & -0.4309 & 0.6230 & 0.0650 & 0.6472 \\ -0.9380 & 0.0273 & 0.0653 & -0.3264 & -0.0925 \\ -0.2236 & 0.2663 & 0.3671 & 0.8170 & -0.2774 \\ 0.1238 & 0.8409 & 0.3356 & -0.2968 & 0.2774 \end{pmatrix}$$

$$S = \begin{pmatrix} 4.3870 & 0 & 0 & 0 \\ 0 & 2.5053 & 0 & 0 \\ 0 & 0 & 1.4110 & 0 \\ 0 & 0 & 0 & 0.6975 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$V = \begin{pmatrix} -0.0391 & 0.2473 & -0.8669 & 0.4311 \\ 0.9581 & -0.2252 & -0.0200 & 0.1760 \\ -0.2829 & -0.8324 & 0.0122 & 0.4764 \\ -0.0227 & 0.4419 & 0.4979 & 0.7458 \end{pmatrix}$$

(c) $A_3 = \begin{pmatrix} 2 & 2 & 5 \\ 1 & 1 & 5 \\ 3 & 2 & 5 \end{pmatrix}$

The returned matrices after calling function `LDU_Decompose` are as below:

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad L = \begin{pmatrix} 1.0000 & 0 & 0 \\ 1.5000 & 1.0000 & 0 \\ 0.5000 & 0 & 1.0000 \end{pmatrix}$$

$$D = \begin{pmatrix} 2.0000 & 0 & 0 \\ 0 & -1.0000 & 0 \\ 0 & 0 & 2.5000 \end{pmatrix} \quad U = \begin{pmatrix} 1.0000 & 1.0000 & 2.5000 \\ 0 & 1.0000 & 2.5000 \\ 0 & 0 & 1.0000 \end{pmatrix}$$

The returned matrices after calling function `SVD_Decompose` are as below:

$$U = \begin{pmatrix} 0.5859 & 0.0444 & -0.8091 \\ 0.5182 & -0.7882 & 0.3319 \\ 0.6231 & 0.6138 & 0.4849 \end{pmatrix} \quad S = \begin{pmatrix} 9.7910 & 0 & 0 \\ 0 & 1.4162 & 0 \\ 0 & 0 & 0.3606 \end{pmatrix}$$
$$V = \begin{pmatrix} 0.3635 & 0.8063 & 0.4666 \\ 0.2999 & 0.3729 & -0.8781 \\ 0.8820 & -0.4591 & 0.1062 \end{pmatrix}$$

By calling function `check_LUD` and `check_SVD` to check the correctness of the LUD and SVD decomposition respectively, all the checking results above return 'True', which means all the decomposition results above satisfy $PA = LUD$ and $A = USV^T$.

3. Solve the system of equations $Ax = b$ for the given values of A and b. Specify whether each system has zero, one, or more solutions. If the system has zero solutions give "the SVD solution" (as defined in class). If the system has a unique solution compute that solution. If the system has more than one solution specify both "the SVD solution" and all solutions. Relate your answers to the SVD decomposition. Show how you obtained your solutions.

Solution:

(a) $A = \begin{pmatrix} 2 & 2 & 5 \\ 1 & 1 & 5 \\ 3 & 2 & 5 \end{pmatrix} \quad b = \begin{pmatrix} 5 \\ -5 \\ 0 \end{pmatrix}$

In this problem, the matrix A is square and invertible, so "the SVD solution" definitely is a solution of $Ax = b$ and it is the unique solution.

By calling function `SVD_Decompose` implemented in problem 2, the decomposition results are as below:

$$U = \begin{pmatrix} 0.5859 & 0.0444 & -0.8091 \\ 0.5182 & -0.7882 & 0.3319 \\ 0.6231 & 0.6138 & 0.4849 \end{pmatrix} \quad S = \begin{pmatrix} 9.7910 & 0 & 0 \\ 0 & 1.4162 & 0 \\ 0 & 0 & 0.3606 \end{pmatrix}$$
$$V = \begin{pmatrix} 0.3635 & 0.8063 & 0.4666 \\ 0.2999 & 0.3729 & -0.8781 \\ 0.8820 & -0.4591 & 0.1062 \end{pmatrix}$$

so "the SVD solution" (unique solution) is:

$$\bar{x} = VS^{-1}U^T b = \begin{pmatrix} -5 \\ 15 \\ -3 \end{pmatrix}$$

(b) $A = \begin{pmatrix} -3 & -4 & -1 \\ 2 & 3 & 1 \\ 3 & 5 & 2 \end{pmatrix} \quad b = \begin{pmatrix} 3 \\ 5 \\ 1 \end{pmatrix}$

In this problem, the rank of the matrix A is 2. A is square but not invertible. The basis for column space of A are: $(-3, 2, 3)^T$ and $(-4, 3, 5)^T$. The vector b is

not in the column space of A. So this system has zero solutions.

By calling function `SVD_Decompose` implemented in problem 2, the matrices after decomposition are as below:

$$U = \begin{pmatrix} -0.5762 & -0.7597 & -0.3015 \\ 0.4248 & 0.0368 & -0.9045 \\ 0.6982 & -0.6493 & 0.3015 \end{pmatrix} \quad S = \begin{pmatrix} 8.8076 & 0 & 0 \\ 0 & 0.6522 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$
$$V = \begin{pmatrix} 0.5306 & 0.6206 & -0.5774 \\ 0.8028 & -0.1492 & 0.5774 \\ 0.2722 & -0.7698 & -0.5774 \end{pmatrix}$$

So "the SVD solution" (a least square solution) is:

$$\bar{x} = V \frac{1}{S} U^T b = \begin{pmatrix} -2.5455 \\ 0.7273 \\ 3.2727 \end{pmatrix}$$

(c) $A = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -2 \\ 1 & 0 & -2 \end{pmatrix} \quad b = \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}$

In this problem, the rank of matrix A is 2. A is square but not invertible. The basis for column space of A are: $(1, 0, 1)^T$ and $(-1, 1, 0)^T$. The vector b is in the column space of A. Thus, for this problem, "the SVD solution" $\bar{x}$ is a particular solution and the system has solutions in form $\bar{x} + x_N$ with x_N is the null space of A.

By calling function `SVD_Decompose` implemented in problem 2, the matrices after decomposition are as below:

$$U = \begin{pmatrix} 0.0000 & 0.8165 & 0.5774 \\ 0.7071 & -0.4082 & 0.5774 \\ 0.7071 & 0.4082 & -0.5774 \end{pmatrix} \quad S = \begin{pmatrix} 3.0000 & 0 & 0 \\ 0 & 1.7321 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$
$$V = \begin{pmatrix} 0.2357 & 0.7071 & 0.6667 \\ 0.2357 & -0.7071 & 0.6667 \\ -0.9428 & -0.0000 & 0.3333 \end{pmatrix}$$

So "the SVD solution" (a particular solution) is:

$$\bar{x} = V \frac{1}{S} U^T b = \begin{pmatrix} 1.4444 \\ -0.5556 \\ -1.7778 \end{pmatrix}$$

The basis for null space of A is: $(2, 2, 1)^T$. So all solutions can be written as:

$$x = \bar{x} + x^N = \begin{pmatrix} 1.4444 \\ -0.5556 \\ -1.7778 \end{pmatrix} + k \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}$$

where $k \in \mathbb{R}$.

4. Suppose that u is an n-dimensional column vector of unit length in $\Re^n$, and let u^T be its transpose. Then uu^T is a matrix. Consider the $n \times n$ matrix $A = I - uu^T$.

Solution:

(a) Describe the action of the matrix A geometrically.

Obviously, the vector space $\Re^n = \{u\} \oplus \{u\}^\perp$, where $\{u\}^\perp$ stands for the hyperplane that is perpendicular to vector u. Each vector $x \in \Re^n$ can be uniquely represented as $x = \alpha + \beta$ where $\alpha \in \{u\}$ and $\beta \in \{u\}^\perp$.
First we consider arbitrary vector $\alpha \in \{u\}$. The vector α can be written as $\alpha = ku, k \in \mathbb{R}$. When α is pre-multiplied by A,

$$A\alpha = (I - uu^T)\alpha = ku - kuu^T u = ku - ku = 0$$

which means that α is projected to 0 by the operation.
Then we consider arbitrary vector $\beta \in \{u\}^\perp$. When β is pre-multiplied by A,

$$A\beta = (I - uu^T)\beta = \beta - uu^T\beta = \beta$$

which means that β remains unchanged after projection.
Thus, for arbitrary vector $x \in \Re^n$,

$$Ax = A(\alpha + \beta) = \beta$$

This means that, when vector $x \in \Re^n$ is pre-multiplied by A, it is projected onto the hyperplane that is perpendicular to vector u.

(b) Give the eigenvalues of A.

To get the eigenvalues, we just need to solve the function below:

$$det(A - \lambda I) = det((1 - \lambda)I - uu^T) = 0$$

According to Sylvester's formula, we have

$$\begin{aligned} det((1 - \lambda)I - uu^T) &= (1 - \lambda)^{n-1} det((1 - \lambda)I - u^T u) \\ &= (1 - \lambda)^{n-1} det(1 - \lambda - 1) \\ &= -(1 - \lambda)^{n-1}\lambda \end{aligned}$$

So we can know that the eigenvalues of A are $n - 1$ 1's and a single 0.

(c) Describe the null space of A.

According to the analysis in (a), we know that for each vector $\alpha \in \{u\}$, there is $A\alpha = 0$. For vector $x \in \Re^n$, it would be projected onto the hyperplane that is perpendicular to $\{u\}$. So the null space of A is $\{u\}$, the space spanned by vector u.

(d) What is A^2?

$$A^2 = (I - uu^T)(I - uu^T) = I - 2uu^T + uu^T uu^T$$
$$= I - 2uu^T + uu^T = I - uu^T = A$$

5. The following problem arises in a large number of robotics and vision problems: Suppose $\mathbf{p}_1, ..., \mathbf{p}_n$ are the 3D coordinates of n points located on a rigid body in three-space. Suppose further that $\mathbf{q}_1, ..., \mathbf{q}_n$ are the 3D coordinates of these same points after the body has been translated and rotated by some unknown amount. Devise and demonstrate an algorithm in which SVD plays a central role for inferring the body's translation and rotation.

   Comment: Your algorithm should make use of all the information available. True, in principle you only need three pairs of points – but if you use more points your solution will be more robust, something that might come in handy some day when you need to do this for real with noisy data.

   Caution: A common mistake is to find the best affine transformation, rather than the best rigid body transformation.

   Hint: It may help to think in terms of inertia and/or correlation matrices.

   One more comment: This problem requires some thought. You can probably find a solution on the web or in a vision text book. Try to solve it yourself before looking at any such sources. Spend some time on the problem. It is good practice to develop your analytic skills. Feel free to discuss among yourselves.

   Solution: The Matlab implementation is in file **code/q5.m**.

   Suppose that the rotation is determined by matrix $A \in \mathbb{R}^{3\times3}$, and the rotation is determined by vector $t \in \mathbb{R}^3$. Then we have:

   $$A(p_1, p_2, ..., p_n) + (t, t, ..., t) = (q_1, q_2, ..., q_n)$$

   namely, for $k \in [1, n]$, $Ap_k + t = q_k$.

   Since there are many noisy data among these n points, we can not work out an exactly true solution for the equation above. The only thing we can do is to minimize the average deviation between the points after rotation and translation and the true points. The average deviation can be represented as:

   $$D = \frac{1}{n}\sum_{k=1}^{n} ||(Ap_k + t - q_k)||^2$$

   The aim of this problem would be to solve A and t subject to minimizing D, namely:

   $$\arg\min_{A,t} \frac{1}{n}\sum_{k=1}^{n} ||(Ap_k + t - q_k)||^2$$

We consider the translation first. Differentiate D with respect to t, there is:

$$\begin{aligned}
\frac{\partial D}{\partial t} &= \frac{1}{n}\frac{\partial}{\partial t}\sum_{k=1}^{n}||(Ap_k + t - q_k)||^2 \\
&= \frac{1}{n}\sum_{k=1}^{n}\frac{\partial (Ap_k + t - q_k)^T(Ap_k + t - q_k)}{\partial t} \\
&= \frac{1}{n}\sum_{k=1}^{n}\frac{\partial (Ap_k + t - q_k)^T(Ap_k + t - q_k)}{\partial (Ap_k + t - q_k)}\frac{\partial (Ap_k + t - q_k)}{\partial t} \\
&= \frac{1}{n}\sum_{k=1}^{n}(Ap_k + t - q_k) \\
&= t + \frac{1}{n}\sum_{k=1}^{n}(Ap_k - q_k)
\end{aligned}$$

When setting the equation above to 0, we get:

$$t = \frac{1}{n}\sum_{k=1}^{n}(q_k - Ap_k) = \frac{1}{n}\sum_{k=1}^{n}q_k - A\frac{1}{n}\sum_{k=1}^{n}p_k = \bar{q} - A\bar{p}$$

where $\bar{p}$ and $\bar{q}$ are the center vectors of the points before and after projection respectively. This means that, given a certain rotation matrix A, by applying translation $t = \bar{q} - A\bar{p}$, the deviation D would be minimized.

Now, let's solve the rotation matrix A. Since the translation vector t is determined, the deviation would be:

$$\begin{aligned}
D &= \frac{1}{n}\sum_{k=1}^{n}||(Ap_k + \bar{q} - A\bar{p} - q_k)||^2 \\
&= \frac{1}{n}\sum_{k=1}^{n}||A(p_k - \bar{p}) - (q_k - \bar{q})||^2
\end{aligned}$$

To simplify the expression, denote $p_k^{'} = p_k - \bar{p}$, $q_k^{'} = q_k - \bar{q}$. Then the aim of the solution would be:

$$\begin{aligned}
\arg\min_A \frac{1}{n}\sum_{k=1}^{n}||Ap_k^{'} - q_k^{'}||^2 &= \arg\min_A \frac{1}{n}\sum_{k=1}^{n}(Ap_k^{'} - q_k^{'})^T(Ap_k^{'} - q_k^{'}) \\
&= \arg\min_A \frac{1}{n}\sum_{k=1}^{n}({p_k^{'}}^T A^T Ap_k^{'} - {p_k^{'}}^T A^T q_k^{'} - {q_k^{'}}^T Ap_k^{'} + {q_k^{'}}^T q_k^{'}) \\
&= \arg\min_A \frac{1}{n}\sum_{k=1}^{n}({p_k^{'}}^T A^T Ap_k^{'} - {p_k^{'}}^T A^T q_k^{'} - {q_k^{'}}^T Ap_k^{'})
\end{aligned}$$

Apply SVD decomposition to A and obtain $A = S_A\Sigma_A V_A$. Because that A is a rotation matrix without scaling, we have ${p_k^{'}}^T A^T Ap_k^{'} = {p_k^{'}}^T p_k^{'}$. Besides, ${p_k^{'}}^T A^T q_k^{'}$ and ${q_k^{'}}^T Ap_k^{'}$ are

both scalars and $p_k^{'T} A^T q_k^{'} = q_k^{'T} A p_k^{'}$, so the equation above would be:

$$\arg\min_A \frac{1}{n}\sum_{k=1}^{n} ||Ap_k^{'} - q_k^{'}||^2 = \arg\max_A \frac{1}{n}\sum_{k=1}^{n} q_k^{'T} A p_k^{'}$$

Let $q_k^{'} = (q_{k1}^{'}, q_{k2}^{'}, q_{k3}^{'})^T$ and $p_k^{'} = (p_{k1}^{'}, p_{k2}^{'}, p_{k3}^{'})^T$, then

$$\begin{aligned} q_k^{'T} A p_k^{'} &= \begin{pmatrix} q_{k1}^{'} & q_{k2}^{'} & q_{k3}^{'} \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} p_{k1}^{'} \\ p_{k2}^{'} \\ p_{k3}^{'} \end{pmatrix} \\ &= \sum_{i=1}^{3}\sum_{j=1}^{3} q_{ki}^{'} a_{ij} p_{kj}^{'} \\ &= tr(A^T q_k^{'} p_k^{'T}) \end{aligned}$$

Thus, the aim equation would be:

$$\begin{aligned} \arg\min_A \frac{1}{n}\sum_{k=1}^{n} ||Ap_k^{'} - q_k^{'}||^2 &= \arg\max_A \frac{1}{n}\sum_{k=1}^{n} tr(A^T q_k^{'} p_k^{'T}) \\ &= \arg\max_A tr(\frac{1}{n}\sum_{k=1}^{n} A^T q_k^{'} p_k^{'T}) \\ &= \arg\max_A tr(A^T \frac{1}{n}\sum_{k=1}^{n} q_k^{'} p_k^{'T}) \end{aligned}$$

Apply SVD decomposition to matrix $\frac{1}{n}\sum_{k=1}^{n} q_k^{'} p_k^{'T}$ and get $\frac{1}{n}\sum_{k=1}^{n} q_k^{'} p_k^{'T} = U\Sigma V^T$, then

$$\begin{aligned} \arg\max_A tr(A^T \frac{1}{n}\sum_{k=1}^{n} q_k^{'} p_k^{'T}) &= \arg\max_A tr(A^T U\Sigma V^T) \\ &= \arg\max_A tr(V^T A^T U\Sigma) \end{aligned}$$

Since Σ is a diagonal matrix, only the diagonal entries of matrix $V^T A^T U$ would contribute to the trace. What's more, $(V^T A^T U)(V^T A^T U)^T = V^T A^T U U^T A V = I$, so matrix $V^T A^T U$ is an orthogonal matrix. In order to maximize the sum of diagonal entries of $V^T A^T U$, it should be equal to I. This means $A = UV^T$.

Thus, finally we obtain the rotation matrix A and the translation vector t:

$$A = UV^T, \qquad t = \bar{q} - A\bar{p}$$

where $U\Sigma V^T = \frac{1}{n}\sum_{k=1}^{n} q_k^{'} p_k^{'T} = \frac{1}{n}\sum_{k=1}^{n}(q_k - \bar{q})(p_k - \bar{p})^T$, $\bar{p} = \frac{1}{n}\sum_{k=1}^{n} p_k$ and $\bar{q} = \frac{1}{n}\sum_{k=1}^{n} q_k$ are the centroids of points before and after projection respectively.

Given the 3D coordinates of points $p_1, p_2, ..., p_n$ and $q_1, q_2, ..., q_n$, the Matlab implementation is **code/q5.m**. The calculation of rotation matrix A and translation vector t is complemented in function `transform_info`, which requires the inputs P and Q. $P = [p_1, p_2, ..., p_n]$ is a $3 \times n$ matrix in which each column is the 3D coordinates of a point, and $Q = [q_1, q_2, ..., q_n]$ is a $3 \times n$ matrix in which each column is the 3D coordinates of the corresponding point in P after translated by unknown amount. This function would return the rotation matrix A and the translation vector t, where A is a 3×3 matrix and t is a 3×1 vector. Besides, function `plot_points` is implemented to visualize the points in P and Q, and also the points in $R = AP + t$, namely the points after rotation and translation by the obtained A and t.

In this script, 10 pairs of points are provided to test the algorithm as below:

$$P = \begin{pmatrix} 0 & -1 & -1 & -2 & -2 & -0.5 & -0.2 & -1.3 & -1.5 & -0.4 \\ 0 & -1 & -1 & -3 & -1 & -2 & -0.5 & -0.3 & -1.5 & -0.5 \\ -1 & -3 & 0 & -1 & -1 & -1.5 & -1 & 0 & -1.5 & 0.8 \end{pmatrix}$$

$$Q = \begin{pmatrix} 2.5 & 3.5 & 3.5 & 4.5 & 4.5 & 3 & 2.7 & 3.8 & 4 & 3 \\ 0 & -1 & 1.5 & 1.6 & 0.6 & 0.7 & 0.4 & 0.8 & 0.5 & 2 \\ 2 & 4 & 2.5 & 4.6 & 2.8 & 4 & 2.4 & 1.8 & 3.5 & 1.5 \end{pmatrix}$$

So, we calculate the center points of P and Q respectively:

$$\bar{p} = (-0.99, -1.08, -0.92)^T, \qquad \bar{q} = (3.5, 0.71, 2.91)^T$$

By calling function `transform_info`, the returned rotation matrix A and translation vector t are as below:

$$A = \begin{pmatrix} -0.9996 & -0.0092 & 0.0263 \\ 0.0271 & -0.5370 & 0.8431 \\ -0.0064 & -0.8435 & -0.53718 \end{pmatrix}, \qquad t = \begin{pmatrix} 2.5246 \\ 0.9325 \\ 1.49868 \end{pmatrix}$$

Now, we call the function `plot_points` to project points P using the obtained A and t and then plot the points.

$$R = AP + t$$
$$= \begin{pmatrix} 2.4983 & 3.4546 & 3.5334 & 4.5252 & 4.5068 & 3.0034 & 2.7029 & 3.8269 & 3.9984 & 2.9501 \\ 0.0894 & -1.0869 & 1.4424 & 1.6462 & 0.5722 & 0.7283 & 0.3525 & 1.0584 & 0.4327 & 1.8647 \\ 2.0357 & 3.9597 & 2.3485 & 4.5789 & 2.8919 & 3.9944 & 2.4587 & 1.7599 & 3.5790 & 1.4933 \end{pmatrix}$$

Then all the points are plotted in an image, shown as Figure 1, where points in P are in blue, points in Q are in red and points in R are in green. From the image, the rotation matrix A and translation vector t we obtained with this algorithm can project the points P to R, which is very close to P. The center point of Q is exactly the same as the center point of R. This makes sense since in the algorithm, we obtained t as $t = \bar{q} - A\bar{p}$, namely $A\bar{p} + t = \bar{q}$.

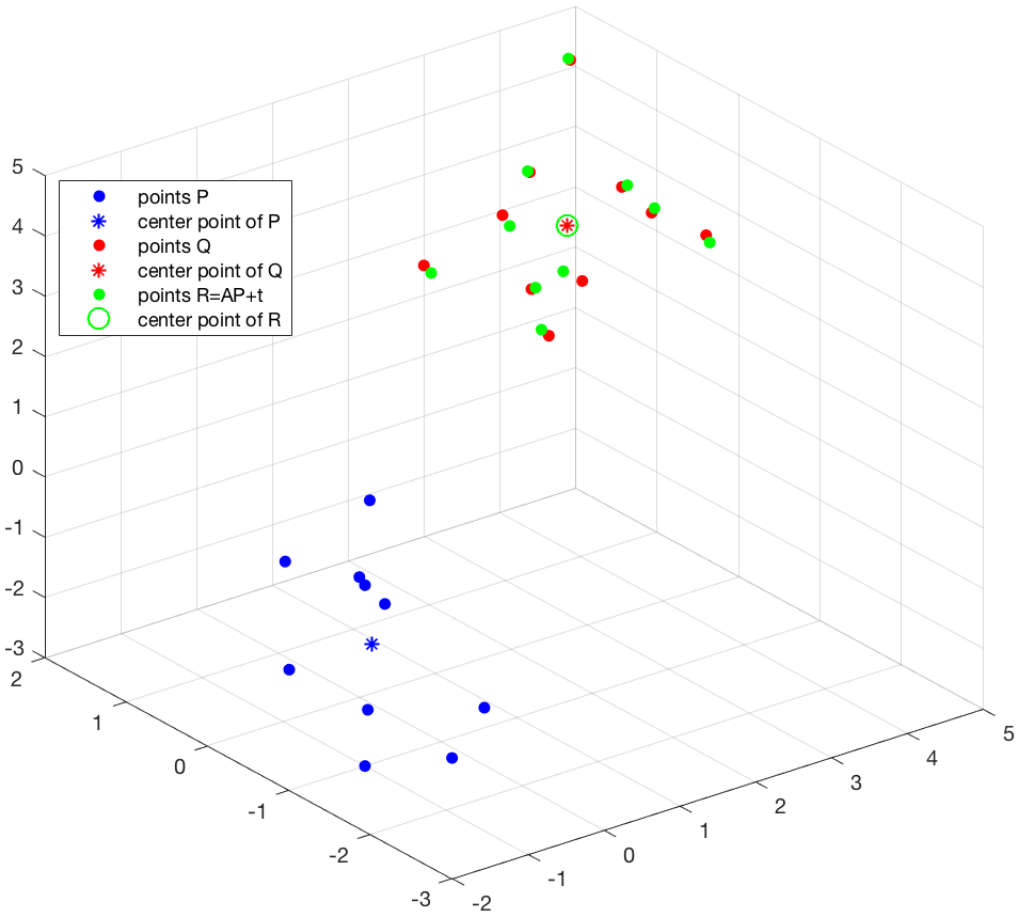

Figure 1: Points distribution in 3D space