EECS 465/ROB 422: Introduction to Algorithmic Robotics
Fall 2023
Homework Assignment #1
Due 9/18/2023 at 11:59pm

Rules:

1. **All homework must be done individually, but you are encouraged to post questions on Piazza.**
2. No late homework will be accepted.
3. **You must show all your work to receive credit for your solutions.**
4. The goal of this homework is to develop your linear algebra skills while also learning how to use python to do calculations. You should use python to do the calculations except where you are asked to perform calculations by hand. You may not use any other language, only python will be accepted.
5. Submit your python code to the Homework 1 Code assignment on Gradescope. Submit a pdf of your answers to the Homework 1 assignment on Gradescope.

Software

1. Read through the python tutorial here. You will need to use Numpy to complete parts of this assignment. Some other resources are also available:
   - Numpy tutorial from Stanford's CS231n (link)
   - Numpy for `matlab` users (link)
2. Install `matplotlib`. First do `sudo apt-get install python-pip`. Then do `pip install matplotlib`.

Vectors

1. (10 points) Given the following vectors:

$$p = \begin{bmatrix} 5 \\ -1 \\ 1 \end{bmatrix}, \quad q = \begin{bmatrix} -4 \\ -2 \\ 7 \end{bmatrix}, \quad r = \begin{bmatrix} -1 \\ -2 \\ 3 \end{bmatrix}$$

Determine the following **by hand** (remember to show your work):

a. $p + 2q$

b. $p \cdot r$ and $r \cdot p$ where "$\cdot$" denotes the dot product 向量相乘不需遵守矩阵相乘的条件

c. $q \times r$ and $r \times q$ where "$\times$" denotes the cross product

d. $||p||$, and $||q||$ where $|| \cdot ||$ denotes the Euclidean norm of a vector

e. distance between the tips of p and q

2. (10 points) Find all k such that $p = \begin{bmatrix} -2 \\ 1 \\ -k \end{bmatrix}$ and $q = \begin{bmatrix} 2 \\ -3k \\ -k \end{bmatrix}$ are orthogonal **by hand**.

Matrices

3. (5 points) Partition the following matrix into submatrices (i.e. find W, X, Y, and Z) **by hand**:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} = \begin{bmatrix} W & X \\ Y & Z \end{bmatrix}$$

where $W \in \mathbb{R}^{2\times 1}$ and $Z \in \mathbb{R}^{2\times 3}$.

4. (10 points) Perform the following matrix multiplication **by hand**:

$$\begin{bmatrix} 3 & -1 & -3 \\ -1 & 0 & 2 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & -1 \\ 0 & -2 & -1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ -2 \end{bmatrix}$$

5. (12 points) Solve the following systems of equations using `numpy`. Recall that there can be a unique solution, no solution, or infinitely many solutions.

a) $\begin{bmatrix} 0 & 0 & -1 \\ 4 & 1 & 1 \\ -2 & 2 & 1 \end{bmatrix} x = \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix}$

b) $\begin{bmatrix} 0 & -2 & 6 \\ -4 & -2 & -2 \\ 2 & 1 & 1 \end{bmatrix} x = \begin{bmatrix} 1 \\ -2 \\ 0 \end{bmatrix}$

c) $\begin{bmatrix} 2 & -2 \\ -4 & 3 \end{bmatrix} x = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

6. (14 points) Given the following matrices:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & -1 \end{bmatrix}, \qquad B = \begin{bmatrix} -2 & -2 \\ 4 & -3 \end{bmatrix}$$

Use `numpy` to calculate the following.

a. $A + 2B$

b. AB and BA

c. A^T, transpose of A

d. B^2

e. $A^T B^T$ and $(AB)^T$

f. $\det(A)$, determinant of A

g. B^{-1}, inverse of B

Rotation

7. (10 points) A rotation matrix R is defined by the following sequence of basic rotations:

   i. A rotation of $\pi/3$ about the z-axis

   ii. A rotation of $\pi/4$ about the new y-axis

   iii. A rotation of π about the new z-axis

   Compute the rotation matrix R.

8. (20 points) You would like to point a mobile robot so that it is looking at a target point. The camera of the robot is aligned with x in the robot's coordinate frame. The robot is assumed to be on flat ground ($z = 0$). The robot's pose in the world frame is:

$$T_r^0 = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 1.7 \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 2.1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

   a. The target point is at $p = [0.1, -1.4, 0]^T$ in the world frame. Assuming the robot stays at its current position, calculate the vector v in the world frame that the camera should align with.

   b. Use v to calculate the desired pose of the robot, again assume the robot does not change position. The robot must remain upright on the ground; i.e. it's z axis must not change.

   c. Prove that the rotation component of the pose meets **all** conditions for being a valid rotation matrix.

Least Squares

9. (30 points) You are testing one of the joints of a new robot arm and you notice there is some error when moving to a target position. To investigate, you command the joint to move through a series of positions and measure where the joint moves for each command. The recorded data is in `calibration.txt`, where the first column is the commanded position and the second column is the measured position. You see that there is significant error in the commanded vs. measured positions. Note: You are strongly advised to consult the linear algebra book, section 13.1 for this problem, especially part (c).

   a. (10 points) Use the psuedo-inverse to perform a least-squares fit of a line to the data. Produce a plot showing the data as blue xs and the line you computed in red and include it in the `pdf`.

Include the parameters of the line and the sum of squared error of the fit in your `pdf`. Submit your code as `leastsquares.py` to Gradescope. When we run the code, it should produce the plot and print out the parameter values and the sum of squared errors.

b. (5 points) Is this least-squares problem *underdetermined* or *overdetermined*? Explain your answer.

c. (15 points) You notice that the error is quite high using a linear fit and you notice that the error seems lower in the range [-0.5, 0.5] than elsewhere. Perform a *piece-wise* linear least-squares fit to the data using -0.5 and 0.5 as the knotpoints. You should *not* perform perform multiple least-squares fits to different parts of the data, i.e. you should set up the whole problem as $Ax = b$ and solve for all the parameters simultaneously.

   Produce a plot in the same format as in (a) and include it in your pdf along with the parameters you computed and the sum of squared errors. Use the parameters you computed in (c) to predict what the measured position will be when you apply the command 0.68. Include the prediction in your `pdf`. Submit your code as `pwleastsquares.py` to Gradescope. When we run the code, it should produce the plot and print out the parameter values, the sum of squared errors, and the prediction for 0.68.