Sheet 2 solutions

April 27, 2018

Exercise 1: Linear Algebra

(a) *Consider the matrices*

$$\mathbf{A} = \begin{pmatrix} 0.25 & 0.1 \\ 0.2 & 0.5 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 0.25 & -0.3 \\ -0.3 & 0.5 \end{pmatrix}.$$

Are they symmetric positive definite?

A symmetric matrix is a square matrix that is equal to its transpose: $\mathbf{M} = \mathbf{M}^T$. Its entries are symmetric with respect to the main diagonal, for example:

$$\mathbf{M} = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

A symmetric $n \times n$ matrix $\mathbf{M}$ is positive definite if the scalar $z^T\mathbf{M}z$ is positive for every non-zero column vector z of n real numbers. It is negative definite if $z^T\mathbf{M}z$ is negative, positive-semidefinite if $z^T\mathbf{M}z \geq 0$ and negative-semidefinite if $z^T\mathbf{M}z \leq 0$ for every non-zero vector z with appropiate dimension. A symmetric positive definite matrix has only positive eigenvalues. This is often easier to check than $z^T\mathbf{M}z$.

- Matrix $\mathbf{A}$ is not symmetric!
- Matrix $\mathbf{B}$ is symmetric. The eigenvalues are calculated as:

$$\begin{aligned} |\mathbf{B} - \lambda\mathbf{I}| &= 0 \\ \left| \begin{pmatrix} 0.25 & -0.3 \\ -0.3 & 0.5 \end{pmatrix} - \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} \right| &= 0 \\ (0.25 - \lambda)(0.5 - \lambda) - 0.09 &= 0 \\ \lambda^2 - 0.75\lambda + 0.035 &= 0 \end{aligned}$$

$$\begin{aligned} \lambda_{1,2} &= \frac{0.75}{2} \pm \sqrt{\left(\frac{0.75}{2}\right)^2 - 0.035} \qquad \text{(pq-formula)} \\ \lambda_1 &= 0.7, \lambda_2 = 0.05 \end{aligned}$$

Both eigenvalues are positive, the matrix is symmetric and positive definite.

(b) *For*

$$\mathbf{C} = \begin{pmatrix} -3 & 0 \\ 0 & 1 \end{pmatrix},$$

find the largest value for $\mu \in \mathbb{R}$ for which $C + \mu I$ is not symmetric positive definite.

We can check the eigenvalues for the largest value of μ for which $C + \mu I$ is not symmetric positive definite. The matrix

$$C + \mu I = \begin{pmatrix} -3 + \mu & 0 \\ 0 & 1 + \mu \end{pmatrix}$$

is in diagonal form, the eigenvalues are the entries of the diagonal. If at least one of the eigenvalues is smaller or equal to zero, the matrix is not symmetric positive definite:

$$\begin{aligned} (-3 + \mu) \leq 0 \quad &\text{or} \quad (1 + \mu) \leq 0 \\ \mu \leq 3 \quad &\text{or} \quad \mu \leq -1 \end{aligned}$$

The largest value for μ for which $C + \mu I$ is not symmetric positive definite is 3.

(c) *Write a program in Python that determines whether a matrix is orthogonal.*

A square matrix is orthogonal, if its columns and rows are orthogonal unit vectors, which is equivalent to:

$$\mathbf{M}^T \mathbf{M} = \mathbf{I}. \tag{1}$$

As an example, rotation matrices are always orthogonal. Please find the code listing below.

(d) *Use this program to investigate whether*

$$\mathbf{D} = \frac{1}{3} \begin{pmatrix} 2 & 2 & -1 \\ 2 & -1 & 2 \\ -1 & 2 & 2 \end{pmatrix}$$

is orthogonal.

Please find the code listing below. Matrix D is orthogonal.

```
import numpy as np

# c) program to verify matrix orthogonality

def check_orthogonal(M):
    # make sure the input is a matrix
    if len(np.shape(M)) != 2:
        print("error: input is not a matrix")
        return
    # make sure the input is a square matrix
```

```
    dim = np.shape(M)[0]
    if dim != np.shape(M)[1]:
        print("error: input is not a square matrix")
        return
    A = np.dot(M, M.T)
    if np.array_equal(A, np.identity(dim)):
        print("matrix is orthogonal")
    else:
        print("matrix is not orthogonal")

# d) apply to given matrix

D = 1./3. * np.array(
    [[2, 2, -1],
     [2, -1, 2],
     [-1, 2, 2]])

check_orthogonal(D)
```

Exercise 2: 2D Transformations as Affine Matrices

The 2D pose of a robot w.r.t. a global coordinate frame is commonly written as $\mathbf{x} = (x, y, \theta)^T$*, where* (x, y) *denotes its position in the* xy*-plane and* θ *its orientation. The homogeneous transformation matrix that represents a pose* $\mathbf{x} = (x, y, \theta)^T$ *w.r.t. to the origin* $(0, 0, 0)^T$ *of the global coordinate system is given by*

$$X = \begin{pmatrix} \mathbf{R}(\theta) & \mathbf{t} \\ 0 & 1 \end{pmatrix}, \mathbf{R}(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}, \mathbf{t} = \begin{pmatrix} x \\ y \end{pmatrix}$$

(a) *While being at pose* $\mathbf{x_1} = (x_1, y_1, \theta_1)^T$*, the robot senses a landmark* l *at position* (l_x, l_y) *w.r.t. to its local frame. Use the matrix* X_1 *to calculate the coordinates of* l *w.r.t. the global frame.*

$$\text{Let } {}^{g}\mathbf{T}_{x_1} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & x_1 \\ \sin\theta_1 & \cos\theta_1 & y_1 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } {}^{x_1}\mathbf{l} = \begin{bmatrix} l_x \\ l_y \\ 1 \end{bmatrix} \text{ then}$$

${}^{g}\mathbf{T}_{x_1}$ is the matrix expression in homogeneous form of pose $\mathbf{x_1}$ w.r.t. the global reference frame, while ${}^{x_1}\mathbf{l}$ is the vector expression in homogeneous form of the landmark w.r.t. the robot reference frame $\mathbf{x_1}$.

The question asks to compute the landmark coordinate w.r.t. the global frame, i.e. ${}^{g}\mathbf{l}$:

$${}^{g}\mathbf{l} = {}^{g}\mathbf{T}_{x_1} \cdot {}^{x_1}\mathbf{l} \tag{2}$$

(b) *Now imagine that you are given the landmark's coordinates w.r.t. to the global frame. How can you calculate the coordinates that the robot will sense in his local frame?*

We are given ${}^{g}\mathbf{l}$ and ${}^{g}\mathbf{T}_{x_1}$ and we want to compute ${}^{x_1}\mathbf{l}$. We can solve this either by taking (2) and solving with respect to ${}^{x_1}\mathbf{l}$ by multiplying to the left and right hand side $({}^{g}\mathbf{T}_{x_1})^{-1}$). Or we follow the same logic as the previous exercise, i.e.

$$ {}^{x_1}\mathbf{l} = {}^{x_1}\mathbf{T}_{g} \cdot {}^{g}\mathbf{l} = ({}^{g}\mathbf{T}_{x_1})^{-1} \cdot {}^{g}\mathbf{l} $$

(c) *The robot moves to a new pose $\mathbf{x_2} = (x_2, y_2, \theta_2)^T$ w.r.t. the global frame. Find the transformation matrix T_{12} that represents the new pose w.r.t. to $\mathbf{x_1}$.* Hint: *Write T_{12} as a product of homogeneous transformation matrices.*

$$ \text{Let } {}^{g}\mathbf{T}_{x_2} = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & x_2 \\ \sin\theta_2 & \cos\theta_2 & y_2 \\ 0 & 0 & 1 \end{bmatrix} $$

${}^{g}\mathbf{T}_{x_2}$ is the matrix expression in homogeneous form of pose $\mathbf{x_2}$ w.r.t. the global reference frame. This time we need to compute the homogeneous matrix form of the pose $\mathbf{x_2}$ expressed w.r.t. the reference frame of $\mathbf{x_1}$, i.e. ${}^{x_1}\mathbf{T}_{x_2}$. Again, we follow the rules of transformation concatenation and we find:

$$ {}^{x_1}\mathbf{T}_{x_2} = {}^{x_1}\mathbf{T}_{g} \cdot {}^{g}\mathbf{T}_{x_2} = ({}^{g}\mathbf{T}_{x_1})^{-1} \cdot {}^{g}\mathbf{T}_{x_2} = T_{12} $$

(d) *The robot is at position $\mathbf{x_2}$. Where is the landmark $\mathbf{l} = (l_x, l_y)$ w.r.t. the robot's local frame now?*

Compute the landmark l w.r.t. the reference frame of $\mathbf{x_2}$, i.e. ${}^{x_2}\mathbf{l}$!

Since we computed ${}^{x_1}\mathbf{T}_{x_2}$ in the previous exercise we can just reuse it as follows:

$$ {}^{x_2}\mathbf{l} = {}^{x_2}\mathbf{T}_{x_1} \cdot {}^{x_1}\mathbf{l} = ({}^{x_1}\mathbf{T}_{x_2})^{-1} \cdot {}^{x_1}\mathbf{l} $$

In case we want to express it only in terms of the ${}^{g}\mathbf{T}_{x_1}$ and ${}^{g}\mathbf{T}_{x_2}$, we can apply the matrix inversion property $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$ and find:

$$ ({}^{x_1}\mathbf{T}_{x_2})^{-1} = \left(({}^{g}\mathbf{T}_{x_1})^{-1} \cdot {}^{g}\mathbf{T}_{x_2}\right)^{-1} = ({}^{g}\mathbf{T}_{x_2})^{-1} \cdot {}^{g}\mathbf{T}_{x_1} $$
$$ \Longrightarrow {}^{x_2}\mathbf{l} = ({}^{g}\mathbf{T}_{x_2})^{-1} \cdot {}^{g}\mathbf{T}_{x_1} \cdot {}^{x_1}\mathbf{l} $$

We could have found the same result by directly applying the rules of transformation concatenation.

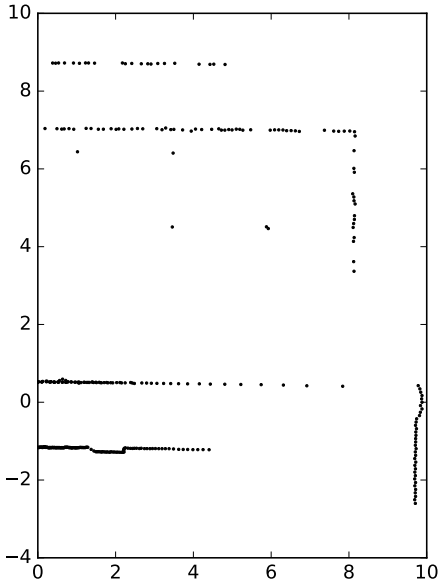
Exercise 3: Sensing

A robot is located at $x = 1.0m$, $y = 0.5m$, $\theta = \frac{\pi}{4}$. Its laser range finder is mounted on the robot at $x = 0.2m$, $y = 0.0m$, $\theta = \pi$ (with respect to the robot's frame of reference).

The distance measurements of one laser scan can be found in the file `laserscan.dat`*, which is provided on the website of this lecture. The first distance measurement is taken in the angle* $\alpha = -\frac{\pi}{2}$ *(in the frame of reference of the laser range finder), the last distance measurement has* $\alpha = \frac{\pi}{2}$ *(i.e., the field of view of the sensor is* π*), and all neighboring measurements are in equal angular distance (all angles in radians).*

(a) *Use Python to plot all laser end-points in the frame of reference of the laser range finder.*

Please find the code listing below.

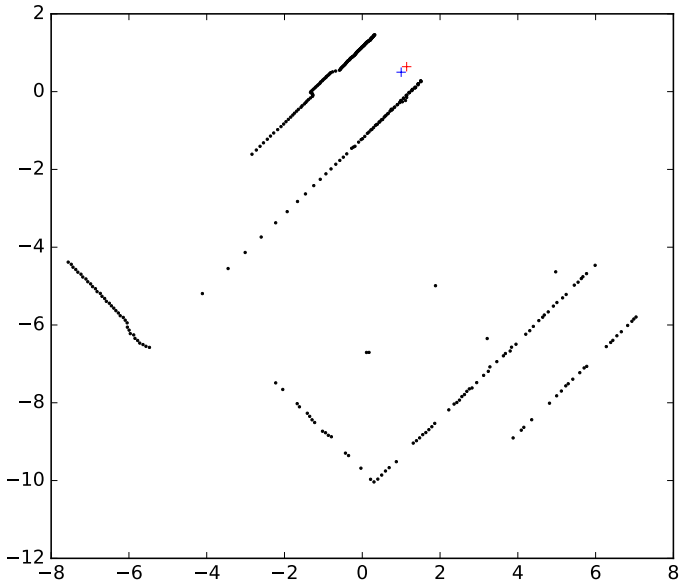


(b) *The provided scan exhibits an unexpected property. Identify it an suggest an explanation.*

It appears as the laser can, at times, see through the walls, which shouldn't be possible. This can indeed happen if the "wall" is actually a semi transparent obstacle, such as a grid, a fence, or a glass.

(c) *Use homogeneous transformation matrices in Octave to compute and plot the center of the robot, the center of the laser range finder, and all laser end-points in world coordinates.*

Please find the code listing below.




```
import numpy as np
import matplotlib.pyplot as plt
import math
pi = math.pi

# a) Load laserscan and plot in scanner frame

scan = np.loadtxt('laserscan.dat')
angle = np.linspace(-pi/2, pi/2, np.shape(scan)[0], endpoint='true')

x = scan * np.cos(angle);
y = scan * np.sin(angle);

plt.plot(x, y, '.k', markersize=3)

# Set the same scale on both axes
plt.gca().set_aspect('equal')
plt.savefig('scan1.pdf')

# c) Transform to global frame

# Define the transformation matrices
T_global_robot = np.array(
    [[np.cos(pi/4), -np.sin(pi/4), 1],
     [np.sin(pi/4),  np.cos(pi/4), 0.5],
     [0, 0, 1]])
```

```
T_robot_laser = np.array(
    [[np.cos(pi), -np.sin(pi), 0.2],
     [np.sin(pi),  np.cos(pi), 0.0],
     [0, 0, 1]])


# Compute the laser frame w.r.t. the global frame
T_global_laser = np.dot(T_global_robot, T_robot_laser)

# Apply the transformation to the scan points
w = np.ones((1, np.shape(x)[0]))[0]
scan_laser =  np.array([x, y, w])
scan_global = np.dot(T_global_laser, scan_laser)

# Plot the laser points
plt.figure()
plt.plot(scan_global[0,:], scan_global[1,:], '.k', markersize=3)

# Plot robot pose in blue
plt.plot(T_global_robot[0,2], T_global_robot[1,2], '+b');

# Plot laser pose in red
plt.plot(T_global_laser[0,2], T_global_laser[1,2], '+r');

# Set the same scale on both axes
plt.gca().set_aspect('equal')
plt.savefig('scan2.pdf')
```