Aarhus university

Computer-Science

Numerical Linear Algebra

Handin 1

Author
Søren M. Damsgaard

Student number
202309814

February 11, 2026

Rotation and bending of lines... The Handin *Dramatic intro music*

Exercise a - c

The following code combines exercise (a) and (c). Note that (b) is addressed after the figure.

The reasoning for (c) is that the rotation of a vector by 80° from an origin point C is the same a bending the line at point C by 80°, which is demonstrated in the figure below. The recipe given for the rotation is:

$$\text{bend}_C(S) = [\, a \mid b \mid c \mid R_C d \,] \tag{1}$$

```python
import numpy as np
import matplotlib.pyplot as plt

# Vectors (Horizontal)
zero = np.array([0, 0])
a = np.array([2, 5])
b = np.array([1.5, 0.2])
c = np.array([0.3, -1])
d = np.array([-0.1, -0.2])

# Rotation shenanigangs
theta = 80
rotation = np.array([[np.cos(theta),-np.sin(theta)],
                     [np.sin(theta),np.cos(theta)]])
dRot = rotation @ d

#coords, just sum the vectors to get the coords
figBefore = np.array([zero, a, a+b, a+b+c, a+b+c+d])
figAfter = np.array([zero, a, a+b, a+b+c, a+b+c+dRot])

fig, (ax1, ax2) = plt.subplots(1,2)
ax1.set_aspect('equal')
ax1.plot(figBefore[:,0], figBefore[:,1])
ax2.set_aspect('equal')
ax2.plot(figAfter[:,0], figAfter[:,1])
plt.show()
```

Listing 1: Python code for making handin figure with matplotlip.

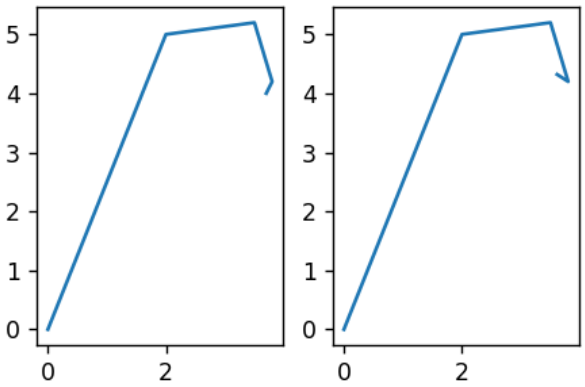

Figure 1: Before and after rotating vector $\vec{d}$ by 80° around point C

For (b), the vector $\vec{OP}$ is given as the sum of the vectors $\vec{OA}$, $\vec{AB}$, $\vec{BC}$, and $\vec{CP}$ which we designate a, b, c, and d respectively.
We determine the vectors as the difference between the coordinates of the points.

$$\begin{aligned}\vec{OP} &= a + b + c + d \\ &= \begin{pmatrix}2\\5\end{pmatrix} + \begin{pmatrix}1.5\\0.2\end{pmatrix} + \begin{pmatrix}0.3\\-1\end{pmatrix} + \begin{pmatrix}-0.1\\-0.2\end{pmatrix} \\ &= \begin{pmatrix}2+1.5+0.3-0.1\\5+0.2-1-0.2\end{pmatrix} \\ &= \begin{pmatrix}3.7\\4\end{pmatrix}\end{aligned}$$

Exercise d

An analogous recipe for only bending A.
Since the bend is around A, any subsequent vector will be affected, which is reflected in the recipe below.

$$\text{bend}_A(S) = [\ a \mid R_A b \mid R_A c \mid R_A d\ ] \tag{2}$$

The following code demonstrates the bending of vector $\vec{b}$ by 80° around point A.
This code is an extension of the previous code hence the ommision of vector/matrix definitions and misc. code.

```
bRot = rotation @ b

figD = np.array([zero, a, a+bRot, a+bRot+c, a+bRot+c+d])

ax3.set_aspect('equal')
ax3.plot(figD[:,0], figD[:,1])
plt.show()
```

Listing 2: Python code for bending vector b by 80° around point A.

The figure below demonstrates how the the rotation of b effects the subsequent vectors c and d.

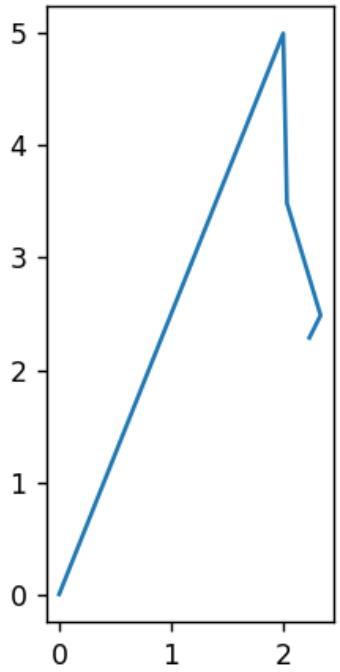

Figure 2: Bending vector $\vec{b}$ by 80° around point A

Effect is now shown and argued for #ToddlerArgument.

Exercise e

Algebraic proof based on the recipes

For the algabraic proof we apply the rotation recipes twice in different order.

$$\text{bend}_A(\text{bend}_C(S)) = [\ a \mid R_A b \mid R_A c \mid R_A(R_C d)\ ] \tag{3}$$

$$\text{bend}_C(\text{bend}_A(S)) = [\ a \mid R_A b \mid R_A c \mid R_C(R_A d)\ ] \tag{4}$$

In the first week of theoretical exercises, we showed that a rotation followed by another rotation is just a single rotation of the vector by the sum of the angles, and the order of the rotations did not matter. Based on that we conclude that $R_A(R_C d) = R_C(R_A d)$. This is also further reinforced by the fact that matrices are associative $A(BC) = B(AC)$.

Proof by numbers

```
# Rotations with theta = 80 (uses earlier defined rotations and angles)
figE1 = np.array([zero, a, a+bRot, a+bRot+c, (a+c+dRot)+bRot])
figE2 = np.array([zero, a, a+bRot, a+bRot+c, (a+bRot+c)+dRot])

```

```
# Rotation with theta = 40
theta2 = 40
rotation2 = np.array([[np.cos(theta2),-np.sin(theta2)],
                      [np.sin(theta2),np.cos(theta2)]])
bRot = rotation2 @ b
dRot = rotation2 @ d
figE3 = np.array([zero, a, a+bRot, a+bRot+c, (a+c+dRot)+bRot])
figE4 = np.array([zero, a, a+bRot, a+bRot+c, (a+bRot+c)+dRot])

# approximate equality check, since difference could be too small for
   python, and might make a rounding error.
one = np.allclose(figE3, figE4)
two = np.allclose(figE1, figE2)
print(one)
print(two)
```

Listing 3: Python showing non-commutativity numerically

```
True
True
```

Listing 4: Output of above code in Listing 3

The numerical proof reinforced the algebraic proof, and we can conclude that the order of the rotations does not matter, and that the two resulting figures are close enough to be equal, which is what we expected from the algebraic proof.

Thanks for coming to my TED talk.
please clap.

Appendix

Clearly a misguided appendixed left redundant after its creator realized that the code was not that long and could be easily included in the main text without a kerfuffle.