Aflevering 5

Lucas Bagge

Vi skal se hvordan vi kan flytte en figur i planen til en standard position. Betragt et ottetalsfigur i planen givet ved

$$(3\cos(t), \sin(2t)), \qquad \text{for } 0 \leqslant t \leqslant 2\pi.$$

Vi vil danne nogle datapunkter der ligge tæt på en drejet version af denne figur.

(a) Plot kurven i python.

a)

I den første del af opgaven skal vi tage de angivende funktioner:

$$(3cos(t), sin(2t)), \text{for } 0 \leq t \leq 2\pi$$

Hertil vil jeg hente ‘numpy‘ og matplotlib modulerne.

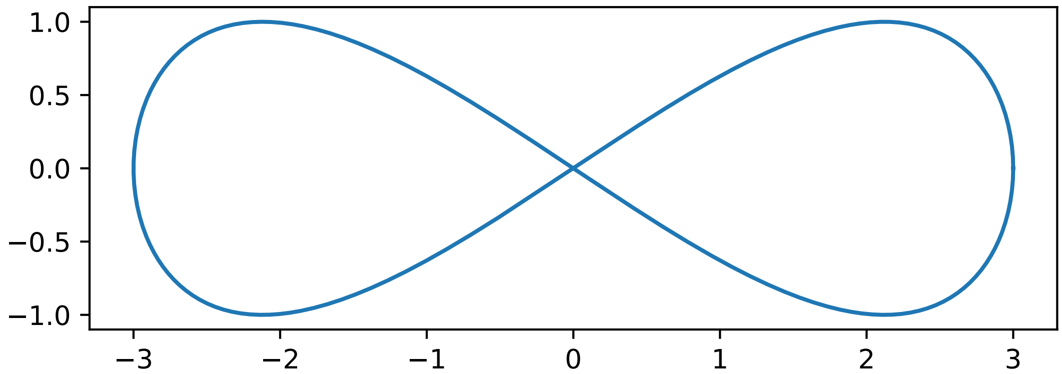
```
import numpy as np
import matplotlib.pyplot as plt
```

Nu kan jeg således danne vores punkter til figuren.

```
t = np.linspace(0, 2*np.pi, 1000)
y_1 = 3 * np.cos(t)
y_2 = np.sin(2*t)
eight = np.array([y_1, y_2])
```

Herefter kan vi plotte det.

```
fig, ax = plt.subplots()
ax.set_aspect('equal')
ax.plot(*eight)
plt.show()
```



Som giver mig det ønskede 8 tal.

(b) Brug

```
rng = np.random.default_rng()
theta = rng.uniform(...)
```

til at vælge en tilfældig vinkel θ mellem $\pi/10$ og $9\pi/10$. Drej kurven med rotationsmatricen $R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$ og plot resultatet.

b)

Jeg opskriver funktioner der skal til for at lave vores plot med støjen.

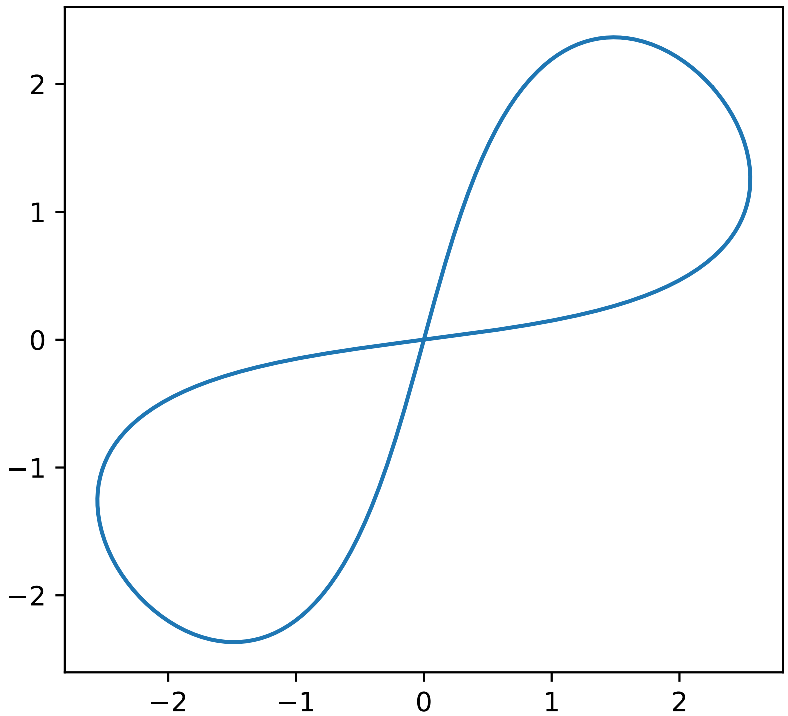
```
rng = np.random.default_rng()
theta = rng.uniform(np.pi / 10, (9 * np.pi) / 10)
```

Herefter definerer jeg min rotationsmatrice:

```
c = np.cos(theta)
s = np.sin(theta)
R = np.array([[c, -s],
              [s, c]])
```

```python
rotated = R @ eight
fig, ax = plt.subplots()
ax.set_aspect('equal')
ax.plot(*(rotated))

plt.show()
```



Her kan vi se at vores ottetals er blevet roteret

c)

(c) For et rimeligt stort n, f.eks. $n = 1000$, dan en $(2 \times n)$-matrix hvis søjler er tilfældige punkter fra den drejede kurve. Ved hjælp af

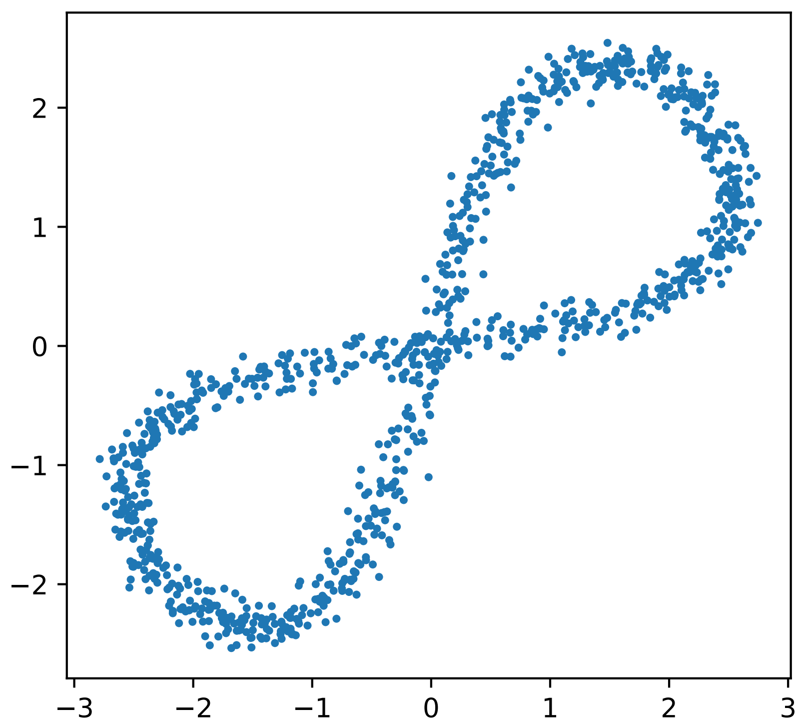
`rng.normal(0.0, 0.1, (2,n)),`

eller noget lignende, tilføj støj til alle indgange til at få en matrix A. Plot punkterne i resultatet.

Dan den matricen angivet i opgave beskrivelsen:

```python
noise = rng.normal(0.0, 0.1, (2, 1000))
A = rotated + noise
```

```
fig, ax = plt.subplots()
ax.set_aspect('equal')
ax.plot(*A, 'o', markersize = 2)
plt.show()
```



Jeg har tilføjet noget støj til ottetallet og herefter plottet det.

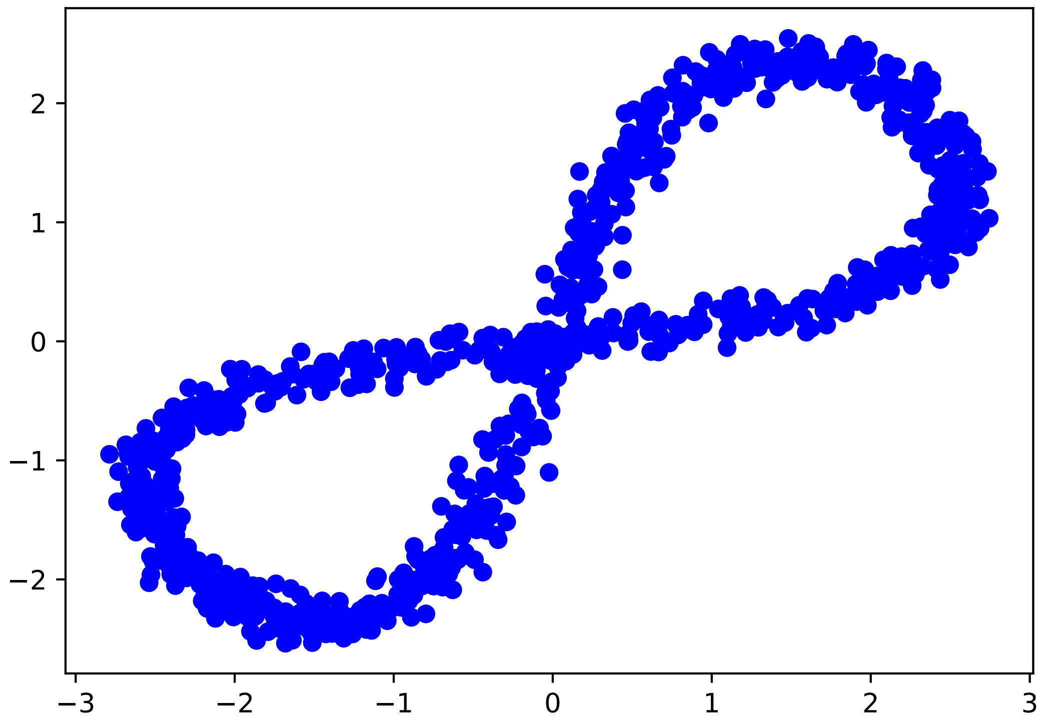
d)

(d) For hver række i A, træk middelværdien fra, og dermed dan en ny matrix B hvor hver række har middelværdi 0. Der må gerne anvendes np.mean().

For at gøre ovenstående skal jeg trække mean fra hver enkelt vektor i A.

```
B = np.vstack([A[0] - np.mean(A[0]), A[1] - np.mean(A[1])])
```

```
fig, ax = plt.subplots()
ax.scatter (*B, color = "blue")
plt.show()
```



e)

(e) Brug python til at beregne singulærværdidekomponeringen $B = U\Sigma V^T$ af B. Angiv $U \in \mathbb{R}^{2\times 2}$ og singulærværdierne.

SVD er en matrix dekomponering metode der reducere en matrice i tre komponenter for senre at gøre udregninger til andre matricer simpler.

U er en m x m matic, Σ er en m x n diagonal matrix og V^T er den transponerede af en n x n matrix.

De diagonale værdier i Σ kendes som singulærværdierne af den orginale matrice. Kolonnerne af U kaldes venstre-singulær vektor af den orginale matrix og kolonnen af V kaldes højre singulær vektor af A.

For vores B matrices komponenter er givet som følgende:

```
u, s, vt = np.linalg.svd(B, full_matrices = False)

print("Dimensioner af vores singulærværdier: \n",u.shape, s.shape, vt.shape)
```

```
## Dimensioner af vores singulærværdier:
##  (2, 2) (2,) (2, 1000)
```

```
print("Værdien af u: \n", u)
```

```
## Værdien af u:
##  [[-0.75371777 -0.65719824]
##  [-0.65719824  0.75371777]]
```

```
print("Værdien af s, som er vores singulærværdier: \n", s)
```

```
## Værdien af s, som er vores singulærværdier:
##  [67.23050729 22.63884049]
```

```
print("Værdien af vt: \n", vt)
```

```
## Værdien af vt:
##  [[-0.04505677 -0.04451329 -0.04819938 ... -0.04754126 -0.04169439
##   -0.04530317]
##  [-0.00317803  0.00361635  0.00391478 ...  0.00160556 -0.00187981
##    0.00747618]]
```

f)

I opgave f skal vi se på hvordan singulærværdierne for B er relateret til den første figur.

Det jeg vil starte med a gøre er at plotte matricen U:

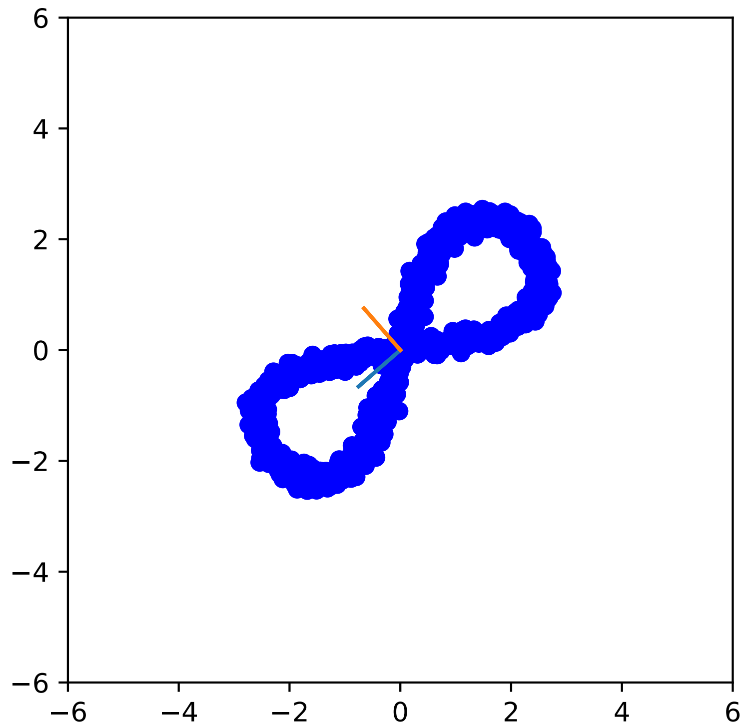
```
n = 1000
scale = 2/np.sqrt(n)
tscale = 1.2*scale
origo = np.zeros((2,1))
fig, ax = plt.subplots()
ax.set_aspect('equal')
ax.scatter (*B, color = "blue")
ax.plot(*np.hstack([origo, u[:,[0]]]))
ax.plot(*np.hstack([origo, u[:,[1]]]))
plt.ylim([-6.0,6.0])
```

```
## (-6.0, 6.0)
```

```
plt.xlim([-6.0, 6.0])
```

```
## (-6.0, 6.0)
```

```
plt.show()
```



Ud fra ovenstående figur kan vi se hvilken retning der giver mest varians. I dette tlfælde er det de venstresingulærvektorer, som også bliver af vores singulærværdier:

```
print("Singulærværdier: \n", s)
```

```
## Singulærværdier:
##  [67.23050729 22.63884049]
```

Her kan vi se at 67 er størst som er vores s[0].

Denne singulærværdi har venstesingulærvektor som er den første føjle af u. Vores figur foroven viser at det således er de venstresingulærvektorer giver retningerne hvor variationen af punkterne er størst.

(f) Beskriv hvordan singulærværdierne og de venstre singulærvektorer for B er relateret til den oprindelige figur.

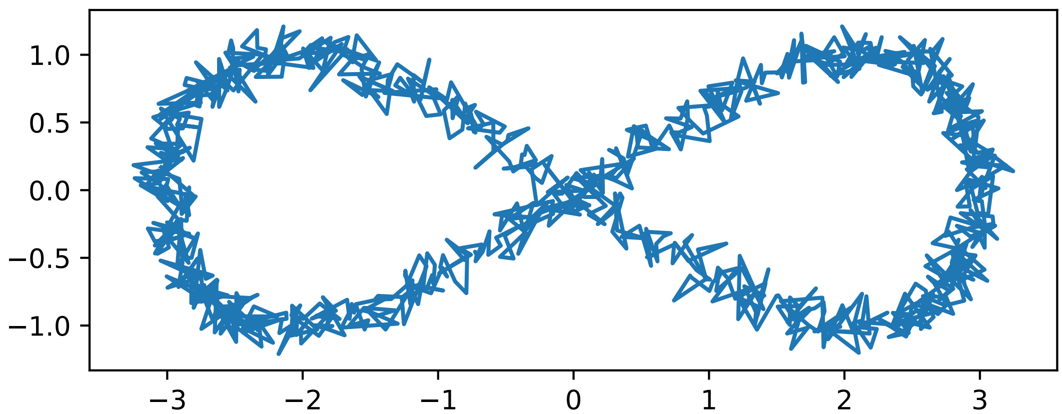
(g) Vis hvordan den ortogonale matrix U kan bruges til at flytte figuren givet ved B, så den ligger tæt på den oprindelige ottetalsfigur.

g)

Her forsøger jeg at gange u med B:

```
opg_g = u @ B
```

```
fig, ax = plt.subplots()
ax.set_aspect('equal')
ax.plot(*opg_g)
plt.show()
```



Her foroven ser vi at vi tilnærmelsesvis har fået den samme figur som i a.