Afleveringsopgave 5

Denne opgave må laves som en gruppeopgave, med gruppestørrelse mellem 1 og 4 personer fra samme TØ hold. Opgaveløsningen forsynes med navnene af alle gruppemedlemmer, og hver gruppemedlem uploader en kopi af besvarelsen, som én pdf fil, brightspace under "Course Tools > Assignments > Aflevering 5". Afleveringsfristen bestemmes af din instruktor, men ligger i uge 12.

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

$$x(t) = 4\cos(t), \quad y(t) = \sin(2t), \qquad \text{for } 0 \leqslant t \leqslant 2\pi.$$

Vi vil danne nogle datapunkter, som ligger tæt på en drejet version af denne figur.

(a) Plot kurven $(x(t), y(t))$ i python.

(b) Brug

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

til at vælge en tilfældig vinkel θ mellem $\pi/7$ og $6\pi/7$. Drej kurven med rotationsmatricen $R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$ og plot resultatet.

(c) For et rimeligt stort n, f.eks. $n = 2000$, 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.

Nu vil vi forsøge at opdage hvordan figuren fra A kan bringes tilbage til den oprindelige figur, uden kendskab til matricen R.

(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(..., axis=..., keepdims=...)`.

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

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

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

Andrew Swann

Funktioner brugt i forbindelse med opgaveløsning

In [12]:
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # Nogle miljøer kræver dette import

def center_matrix(A):
    """
    (d) Fjern middelværdi fra hver række i A.
```

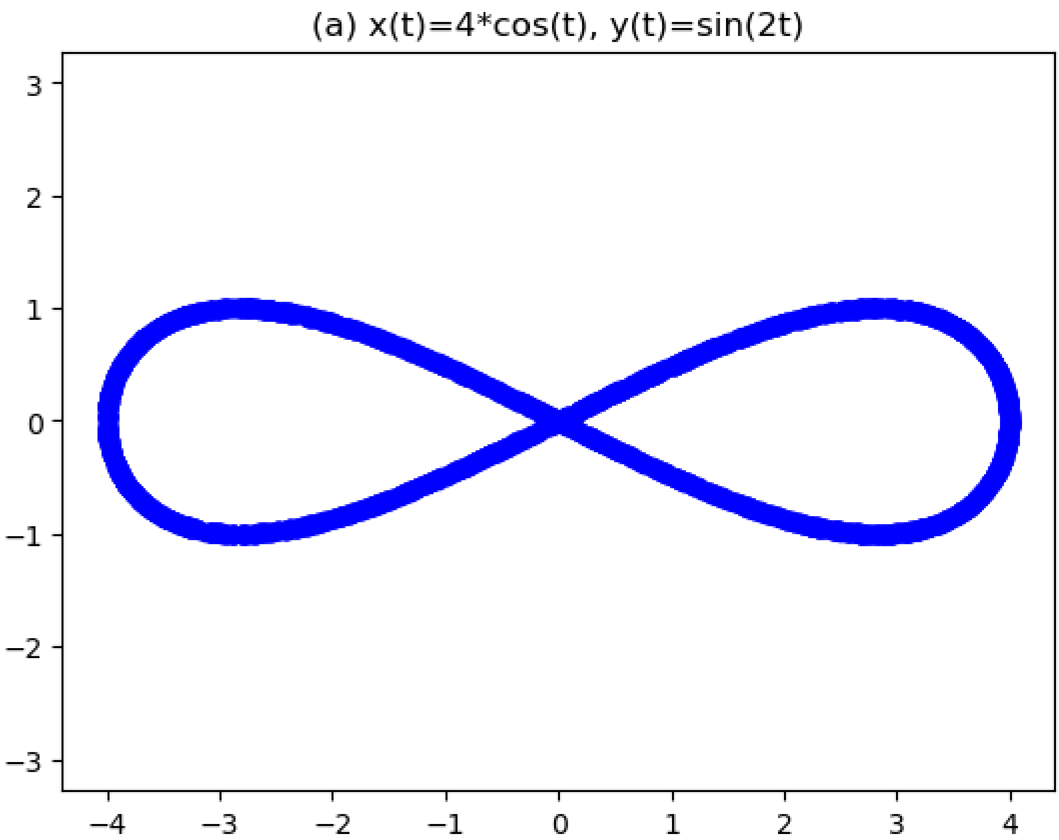
```
    B = A - mean(A) pr. række.
    """
    row_means = np.mean(A, axis=1, keepdims=True)
    B = A - row_means
    return B

def compute_svd(A, full_matrices=False):
    """
    (e) Udfør SVD: A = U * Σ * V^T. Formel kommer fra ligning 10.5 i notessæt.
    Returnerer (U, s, Vt).
    """
    U, s, Vt = np.linalg.svd(A, full_matrices=full_matrices)
    return U, s, Vt

def tjek_ortogonalitet(U):
    """
    Tjekker om en matrix U er ortogonal ved at verificere om U^T * U = I.
    I henhold til Proposition 9.5 bevarer en ortogonal matrix længder og
    indre produkter: <Uu, Uv> = <u, v> for alle vektorer u, v.
    """
    # Identitetsmatrix med samme dimension som U
    I = np.eye(U.shape[0])

    # Beregn U^T * U
    UTU = U.T @ U

    # Tjek om UTU er lig med I (inden for en numerisk tolerance)
    if np.allclose(UTU, I, atol=1e-10):
        print("Matrix U er ortogonal, fordi U^T * U = I (inden for numerisk tole
    else:
        # Viser eventuelt afvigelsens størrelse
        afvigelse = np.linalg.norm(UTU - I)
        print(f"Matrix U er ikke ortogonal, fordi U^T * U != I. (Afvigelse: {afv


import numpy as np
import matplotlib.pyplot as plt

def compute_svd(A, full_matrices=False):
    """
    Udfører SVD: A = U * Σ * V^T.
    Returnerer (U, s, Vt).
    """
    U, s, Vt = np.linalg.svd(A, full_matrices=full_matrices)
    return U, s, Vt
```

Kode til selve opgaveløsning

a)

In [13]:
```
#(a)
# Parametrisk kurve
rng = np.random.default_rng()
n_a = 2000  #Vi sætter n til at være stor fra start af,
            #så vi ikke behøver på den senere i c)

t = rng.uniform(0, 2*np.pi, n_a)  # Tilfældige t i [0, 2*pi]
```

```
# x(t) = 4*cos(t), y(t) = sin(2t)
x_a = 4 * np.cos(t)
y_a = np.sin(2 * t)

#Plot kurven
plt.figure()
plt.scatter(x_a, y_a, color='blue')
plt.title('(a) x(t)=4*cos(t), y(t)=sin(2t)')
plt.axis('equal')
plt.show()
```

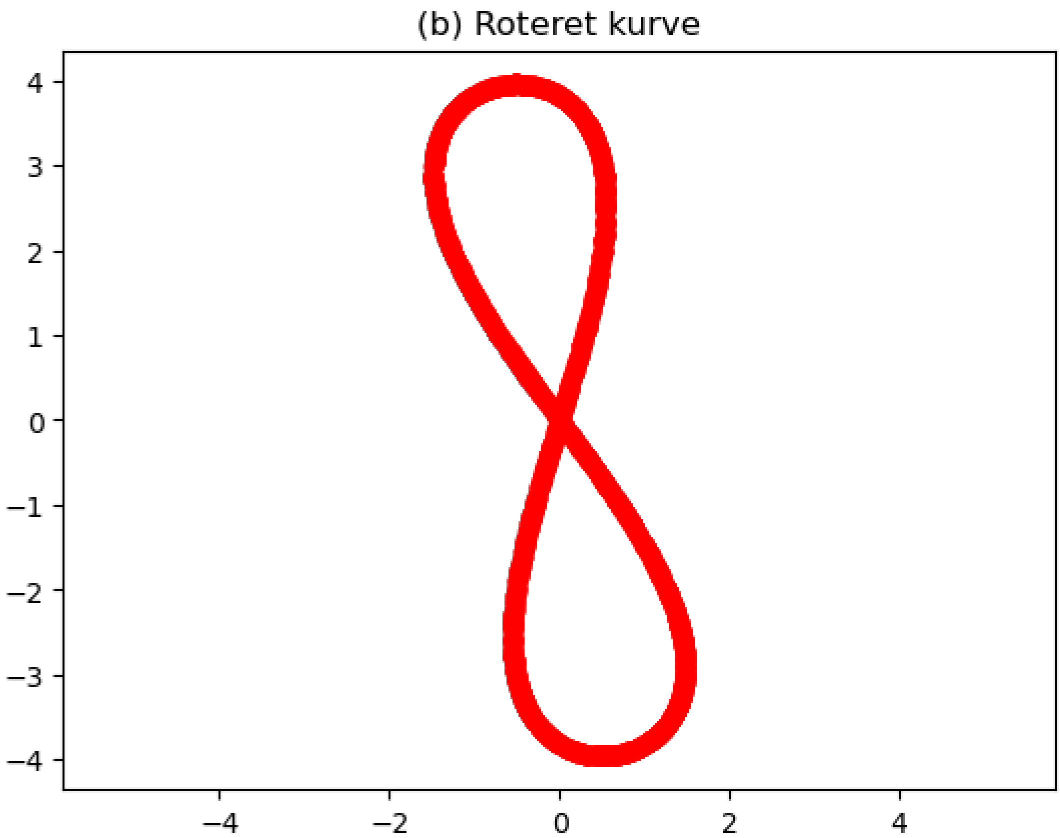


Hermed er vores kurve plottet til a)

b)

In [14]:
```
#(b)
# Roteret kurve
kurve = np.vstack((x_a, y_a))

rng = np.random.default_rng()
theta = rng.uniform(np.pi/7,6*np.pi/7)

#Vi bruger rotationsmatrix angivet i Opgaven til at rotere kurven
R = np.array([[np.cos(theta), -np.sin(theta)],
              [np.sin(theta), np.cos(theta)]])

#Brug matrixmultiplikation til at rotere kurven
kurve_rotate = kurve.T @ R

#Plot den roterede kurve
plt.figure()
plt.scatter(kurve_rotate[:,0], kurve_rotate[:,1], color='red')
plt.title('(b) Roteret kurve')
```

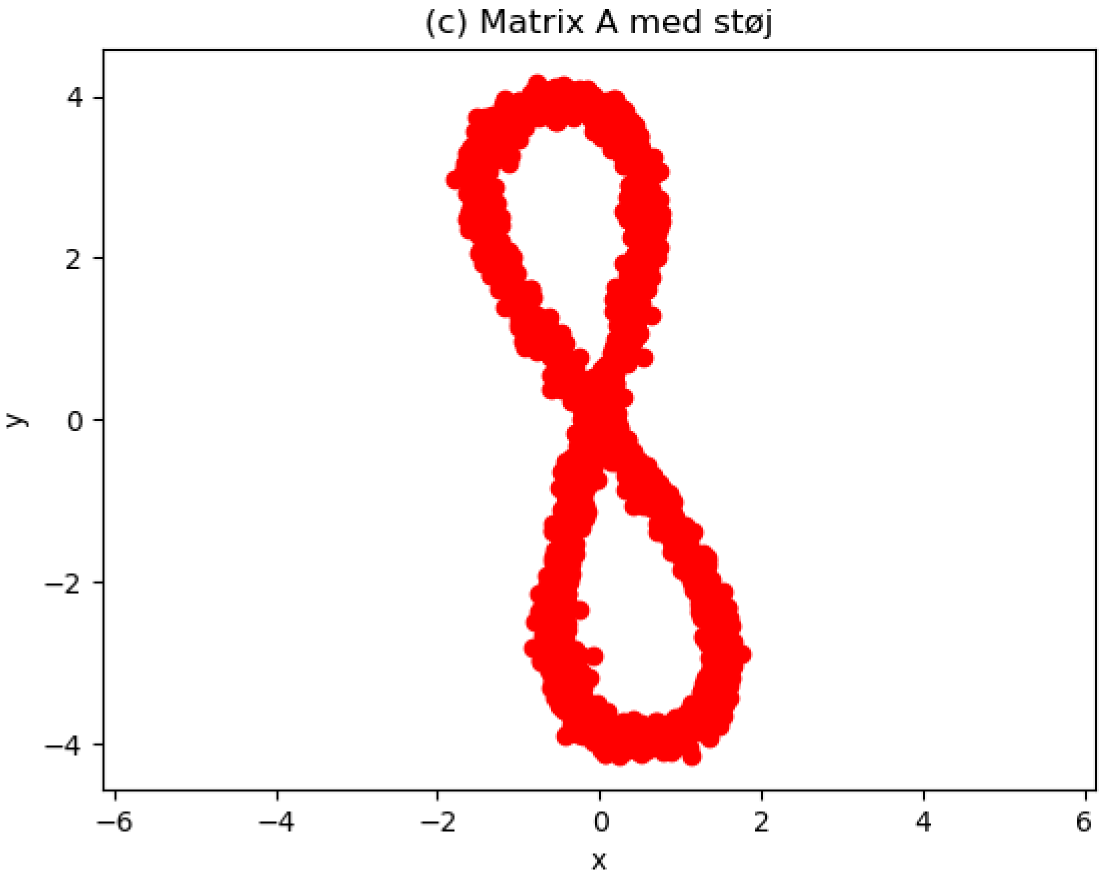
```
plt.axis('equal')
plt.show()
```



Hermed ses vores roteret kurve efter vi har ganget vores rotationsmatrix på til b).

c)

In [15]:
```
#  (c)
#  Dan en 2 * n matrix A, hvor hver kolonne er et punkt på den roterede kurve
#  og tilføj støj til punkterne. Plot punkterne i resultatet.

#Opret A ud fra vores roteret kurve og læg støj med rng.normal.
#Transponer herefter A til at få 2*n matrix og plot den.
A = kurve_rotate + rng.normal(0.0, 0.1, (2000, 2))
A = A.T

#Plot A
plt.figure()
plt.scatter(A[0,:], A[1,:], color='red')
plt.title('(c) Matrix A med støj')
plt.axis('equal')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Hermed ses vores roteret matrix A med tilføjet støj.

d)

In [16]:
```
#  (d) Fjern middelværdi fra hver række (centrering).
# Vi bruger np.mean i funktion.
B = center_matrix(A)
B
```

Out[16]:
```
array([[-0.01756426,  0.78764479,  0.53599487, ..., -1.45828361,
        -0.91318661, -1.32585247],
       [ 3.76923234, -1.25169776, -0.66656542, ...,  2.74623265,
         3.98331544,  2.61085076]])
```

Hermed ses vores matrix B hvor vi har fjernet middelværdien fra hver række vha vores center_matrix funktion.

e)

In [17]:
```
#  (e) Beregn SVD af B og check ortogonalitet af U.
#Beregn SVD af B og print singulærværdierne.
U, s, Vt = compute_svd(B, full_matrices=False)
print("e) SVD af B:")
print("Singulærværdier (s):", s)
print("Form U:", U.shape, "Form Vt:", Vt.shape)

# Bekræft at U er ortogonal. For at gøre dette, konstruerer vi Gram-matricen
# G = U^T * U og ser om den er identitetsmatricen. Gør vi med vores funktion.
print("\n e) U matrix:")
print(U)
```

```
print("Check af ortogonalitet af U vha funktion:")
is_U_orthogonal = tjek_ortogonalitet(U)

print("\n e): Første to singulærværdier:", s[:2])
print("Første venstre singularvektor, U[:,0]:", U[:, 0])
print("Anden venstre singularvektor, U[:,1]:", U[:, 1])
```

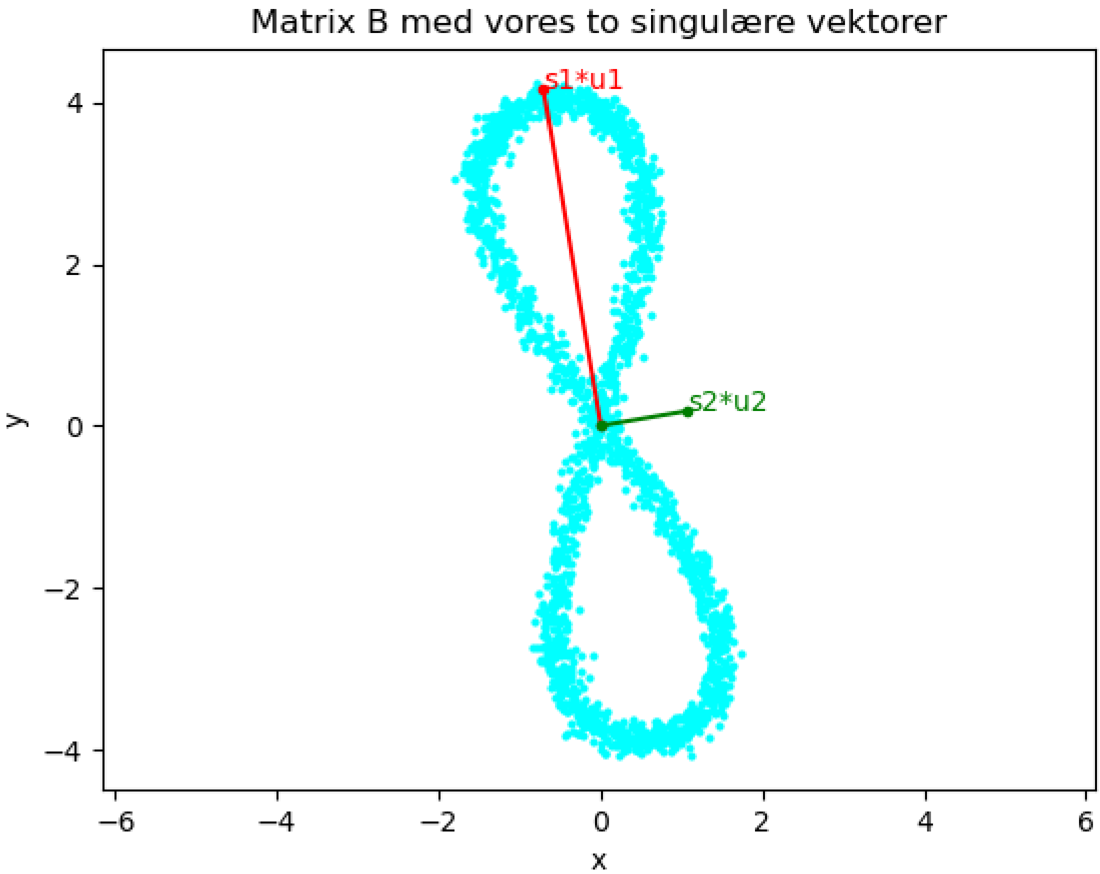
```
e) SVD af B:
Singulærværdier (s): [126.18667906  32.11325038]
Form U: (2, 2) Form Vt: (2, 2000)

 e) U matrix:
[[-0.16746856  0.98587742]
 [ 0.98587742  0.16746856]]
Check af ortogonalitet af U vha funktion:
Matrix U er ortogonal, fordi U^T * U = I (inden for numerisk tolerance).

 e): Første to singulærværdier: [126.18667906  32.11325038]
Første venstre singularvektor, U[:,0]: [-0.16746856  0.98587742]
Anden venstre singularvektor, U[:,1]: [0.98587742 0.16746856]
```

Hermed har vi beregnet SVD af matrix B vha formel 10.5 i notessæt og checket om U er ortogonal ud fra prop 9.5 i notessæt.

f)

Vi kan først tegne de venstre singulærvektorer fra e) oven på vores 8 tals figur med følgende kode baseret på eksempel 10.3.1 i notessæt.

In [18]:
```
# 1) Plot B som i notessættet eksempel 10.3.1
plt.figure()
plt.plot(B[0, :], B[1, :], 'o', markersize=2, color='cyan')
plt.title('Matrix B med vores to singulære vektorer')
plt.axis('equal')

# 2) Definér "origo", scale og tscale
origo = np.zeros((2, 1))          # (0,0) i det centrerede koordinatsystem
scale = 1.5 / np.sqrt(B.shape[1]) # fx 2/sqrt(n)
tscale = scale             # lidt større for tekstplacering

# 3) Tegn den første venstre singularvektor: s[0]*u1
#    notessættet bruger ofte np.hstack([origo, s[0]*U[:,0]*scale])
v1 = np.hstack([origo, (s[0] * U[:, 0] * scale).reshape(2,1)])  # (2,2)
plt.plot(v1[0, :], v1[1, :], color='red', marker='.')
plt.text(s[0]*U[0, 0]*tscale, s[0]*U[1, 0]*tscale, 's1*u1', color='red')

# 4) Tegn den anden venstre singularvektor: s[1]*u2
v2 = np.hstack([origo, (s[1] * U[:, 1] * scale).reshape(2,1)])
plt.plot(v2[0, :], v2[1, :], color='green', marker='.')
plt.text(s[1]*U[0, 1]*tscale, s[1]*U[1, 1]*tscale, 's2*u2', color='green')

plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



I e) ser vi, at den første singularværdi

$$\sigma_0 = 127$$

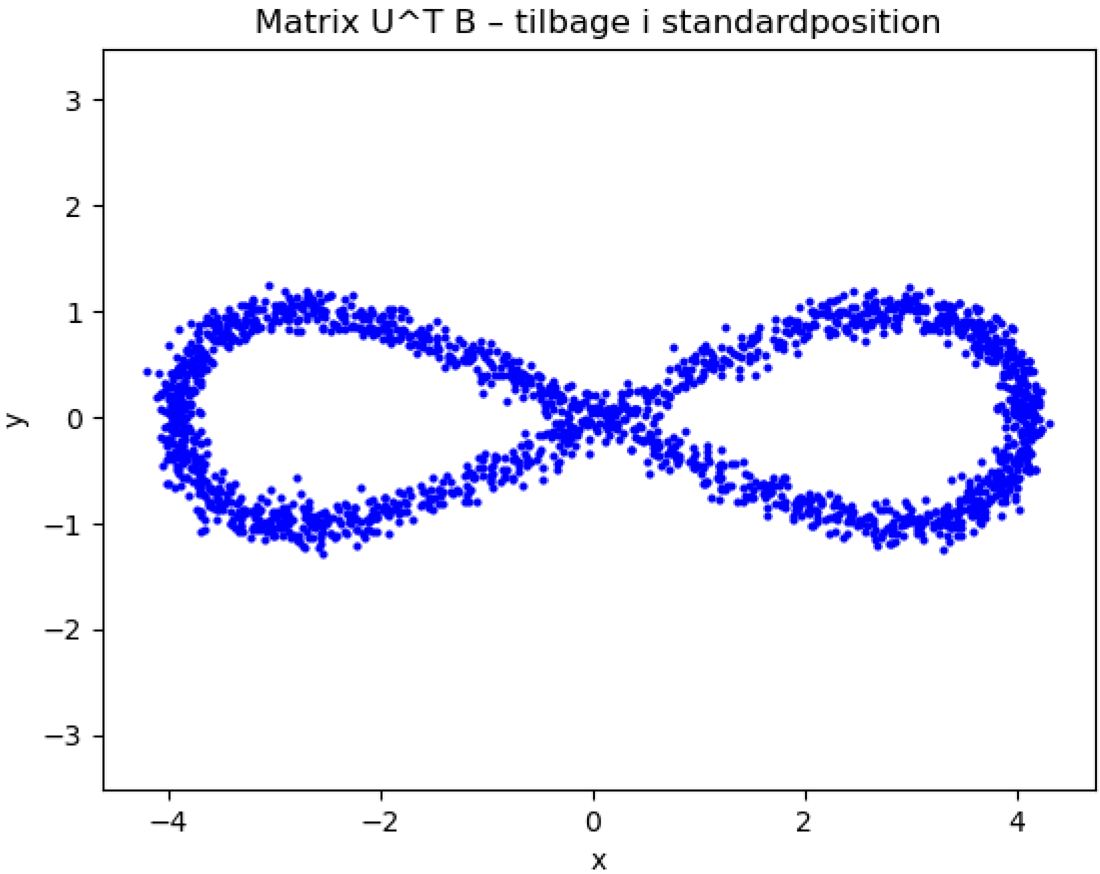
er meget større end næste singularværdi

$$\sigma_1 = 31$$

Dette stemmer med ligning 10.6, hvor singularværdierne altid er sorteret ikke-stigende. Dermed dominerer u_1 (rød pil) datasættets største variation langs ottetallets hovedakse, mens u_2 (grøn pil) peger i en retning med mindre spredning.

g)

In [19]:
```
#Transformér B tilbage ved at gange med U^T (g) og plot
B_std = U.T @ B
plt.figure()
plt.plot(B_std[0, :], B_std[1, :], 'o', markersize=2, color='blue')
plt.title('Matrix U^T B – tilbage i standardposition')
plt.axis('equal')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Da U er en ortogonal matrix:

$$U^T = U^{-1}$$

gælder def 9.1:

$$U^T * U = I$$

, hvor I er identitetsmatrixen. Dette betyder vi kan fjerne den venstre transformation U ved at gange med U^T fra venstre.

$$B_{std} = U^T B$$

Ved at indsætte SVD fra ligning 10.3

$$B = U \sum V^T$$

får vi

$$U^T B = U^T (U \sum V^T) = I \sum V^T = \sum V^T$$

Så kun diagonal/skaleringsmatrixen og højretranformationen bliver tilbage og den venstre ortogonale del(U).

Konklusion: Ved at gange med U^T fra venstre, bringer vi figuren tilbage til en standard position, hvor rotationen(eller spejlingen) fra U er fjernet.

Noter brugt fra notessæt til løsning af opgave

Proposition 9.5. *For $A \in \mathbb{R}^{n \times n}$ ortogonal, gælder*

$$\langle Au, Av \rangle = \langle u, v \rangle, \qquad \textit{for alle } u, v \in \mathbb{R}^n. \tag{9.2}$$

Specielt gælder $\|Au\|_2 = \|u\|_2$ for alle $u \in \mathbb{R}^n$.

Omvendt, hvis $A \in \mathbb{R}^{n \times n}$ opfylder (9.2), så er A en ortogonal matrix.

10.2 SVD generelt

Sætning 10.2. *Enhver matrix $A \in \mathbb{R}^{m \times n}$ har en* singulærværdidekomponering

$$A = U \Sigma V^T \tag{10.5}$$

hvor $U \in \mathbb{R}^{m \times m}$ og $V \in \mathbb{R}^{n \times n}$ er ortogonale matricer og $\Sigma \in \mathbb{R}^{m \times n}$ er en diagonal matrix

$$\Sigma = \operatorname{diag}(\sigma_0, \dots, \sigma_{k-1}) = \begin{bmatrix} \sigma_0 & 0 & 0 & \dots \\ 0 & \sigma_1 & 0 & \dots \\ \vdots & & \ddots & \vdots \end{bmatrix}, \qquad k = \min\{m, n\}, \tag{10.6}$$

med $\sigma_0 \geqslant \sigma_1 \geqslant \cdots \geqslant \sigma_{k-1} \geqslant 0$.

Tallene σ_i kaldes *singulærværdier* for A. Søjlerne af U er *venstresingulærvektorer*; søjlerne af V er *højresingulærvektorer*.

```
print(u[:, [0]])
```

```
[[-0.98579704]
 [ 0.16794106]]
```

Vi kan tegne de venstresingulærvektorer ovenpå punktskyen:

```
origo = np.zeros((2, 1))
scale = 2 / np.sqrt(n)
tscale = 1.4 * scale

fig, ax = plt.subplots()

ax.set_xlim(-20, 20)
ax.set_ylim(-20, 20)
ax.set_aspect('equal')

ax.plot(*a, 'o', markersize = 2, color='cyan')

ax.plot(*(np.hstack([origo, u[:, [0]]*s[0]*scale])),
        color='red', marker='.')
ax.text(*u[:, [0]]*s[0]*tscale, 's0*u0', color='red')

ax.plot(*(np.hstack([origo, u[:, [1]]*s[1]*scale])),
        color='red', marker='.')
ax.text(*u[:, [1]]*s[1]*tscale, 's1*u1', color='red')

ax.set_xlabel('$x$')
ax.set_ylabel('$y$')
```

Figur 10.2 demonstrerer at de venstresingulærvektorer giver retningerne hvor variationen af punkterne er hhv. størst og mindst.

134

$$\Sigma = \operatorname{diag}(\sigma_0, \dots, \sigma_{k-1}) = \begin{bmatrix} \sigma_0 & 0 & 0 & \dots \\ 0 & \sigma_1 & 0 & \dots \\ \vdots & & \ddots & \vdots \end{bmatrix}, \qquad k = \min\{m, n\}, \qquad (10.6)$$

med $\sigma_0 \geqslant \sigma_1 \geqslant \cdots \geqslant \sigma_{k-1} \geqslant 0$.

Tallene σ_i kaldes *singulærværdier* for A. Søjlerne af U er *venstresingulærvektorer*; søjlerne af V er *højresingulærvektorer*.

Beviset gives i afsnit 10.4. Her vil vi fokusere på fortolkningen og anvendelser.

dige vektorer.

Lad os samler u_0, u_1 og v_0, v_1 til (2×2)-matricer

$$U = \begin{bmatrix} u_0 \mid u_1 \end{bmatrix} \quad \text{og} \quad V = \begin{bmatrix} v_0 \mid v_1 \end{bmatrix}.$$

Så er U og V ortogonale matricer, da deres søjler er enhedsvektorer, der står vinkelret på hinanden. Sættes

$$u_0 = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}, \quad u_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}.$$

kan vi omskrive (10.1) til

$$\begin{aligned} AV &= \begin{bmatrix} Av_0 \mid Av_1 \end{bmatrix} = \begin{bmatrix} \sigma_0 u_0 \mid \sigma_1 u_1 \end{bmatrix} = \begin{bmatrix} x_0\sigma_0 & x_1\sigma_1 \\ y_0\sigma_0 & y_1\sigma_1 \end{bmatrix} \\ &= \begin{bmatrix} x_0 & x_1 \\ y_0 & y_1 \end{bmatrix} \begin{bmatrix} \sigma_0 & 0 \\ 0 & \sigma_1 \end{bmatrix} = U\Sigma, \end{aligned} \tag{10.2}$$

hvor

$$\Sigma = \begin{bmatrix} \sigma_0 & 0 \\ 0 & \sigma_1 \end{bmatrix}.$$

Da V er en ortogonal matrix, er den invertibel med invers V^T. Ganges (10.2) med $V^{-1} = V^T$ fra den højre side, fås

$$A = U\Sigma V^T. \tag{10.3}$$

Definition 9.1. En kvadratisk matrix $A \in \mathbb{R}^{n \times n}$ er *ortogonal* hvis

$$A^T A = I_n.$$