Eksamen Numerisk Linear algebra F 2021

Lucas Bagge 202004972

17. Juni 2021

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

Opgave 1)

Matricen M er en stokastisk matrice, da søjlerne summer til 1.

Her kan vi bruge proposition 24.4 og får at egenværdien er 1.

Dermed er svaret **B**.

Opgave 2)

Vi benytter os af definition 19.6 om koordinatvektor.

Vi kan danne en matrice og regne determinanten

```
opg_2 = np.array([3.0, 2.0])[:, np.newaxis]
opg_2_2 = np.array([-3.0, 3.0])[:, np.newaxis]

V = np.hstack([opg_2, opg_2_2])
print(f'Determinanten er forskellig fra nul \n: {np.linalg.det(V)}')
```

```
## Determinanten er forskellig fra nul
## : 15.0
```

Herefter kan vi bestteme koordinatvektoren $b = (-1, -1)$ med hensyn til base E.

$$x_0 \begin{bmatrix} 3 \\ 2 \end{bmatrix} + \begin{bmatrix} -3 \\ 3 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

Herefter udføre vi diverse rækkeoperationer på den udvided matrix

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

Det første trin vil være at dele første række med 1.

Herhefter kan vi ved at gange med 2 og trække fra anden række få elimeres første søjle.

Efterfølgende kan vi få et pivot element i anden række ved at dele anden række med 5.

Til slut får vi elimeneret anden kolonne.

Her får vi følgende koordinatvektor

$$[b]_E = \begin{bmatrix} -2/5 \\ -1/15 \end{bmatrix}$$

Som er svar mulighed C.

Det skal nævnes at jeg har fulgt eksempel 19.7 til udførelsen af denne opgave.

Opgave 3)

For at besvare dette spørgsmål omkring dimensionerne af søjlerummet, så benytter jeg mig af afsnit 19.4 og proposition 19.9 samt proposition 19.11 ligning

$$dim\ S(A) + dim(N) = n$$

Hvor vi har:

$$dimS(A) + 1 = 5$$

Derfor må svaret til denne opgave være B, altså **dimensionen af søjlerummet er 4**.

Opgave 4)

Opgave 5)

a)

For en lineær transofrmation benytter vi os af definition 18.1.

Vi ser at L er en lineær transformation da hver element er lineær og vi observer ingen konstanter der står alene. Så de to regler for lineær afbilding præ definition 18.1 er opfyldt.

Vi skal også kommentaer eller redeføre for at vi kan skrive standmatricen op

```
A = np.array([
  [1., 1., 0., -2.],
  [1., -1., 2., 0.],
  [0., 1., -1., -1.]])
A
```

```
## array([[ 1.,  1.,  0., -2.],
##        [ 1., -1.,  2.,  0.],
##        [ 0.,  1., -1., -1.]])
```

Hvis vi kigger på den første lignign i L

$$x + y - 2w$$

Hvor vi i alt har 4 variabler: x, y, z og w

I ligningen svarer hver variable til 1 eller den tilsvarende konstant foran. Så for x har vi i matrix notation det er 1, 1 for y, men for z har vi ingen variable og derfor er den nul. Til slut har vi -2 for w. Derfor når vi skriver den første række som: [1 1 0 -2] så svarer deres værdi til ligningen og det passer overens. Derfor kan vi godt opskrive A således.

b)

Til at besvare denne opgave gør jeg brug af kapitel 19, samt proposition 19.2 og underafsnittet 19.4.

Aller første skridt er at tage vores A matrice og reducer den til echelon form:

For at undgø at skrive alle matrice op skriver jeg hver række operation op:

$$R_2 : 2 = R_2 - R_1$$

$$R_2 = \frac{R_2}{2}$$

$$R_1 = R_1 - R_2$$

$$R_3 = R_3 - R_2$$

Her får jeg så at en række har kun nul elementer. Derfor bytter jeg rundt så jeg får følgende reducerede echelon form.

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

Nu kan vi så bestemme en basis for N(A): hvor vi skriver Ax=0 op:

$$x_0 + x_2 - x_3 = 0$$

$$x_1 - x_2 - x_3 = 0$$

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -x_2 + x_3 \\ x_2 + x_3 \\ x_2 \\ x_3 \end{bmatrix} = x_2 \begin{bmatrix} -1 \\ 1 \\ 1 \\ 0 \end{bmatrix} + x_3 \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Hvor de to vektor (-1, 1, 1, 0) og (1, 1, 0, 1) udspænder nulrummet og er en basus for N(A)

Så for søjlerummet skal vi tage pivot kolonner og gå tilbage til den oprindelig matrice og skrive det som S(A). Derfir passer det med at v0=første søjle i A og v1= anden søjle i A, da de netop udgør pivot elementerne i den reducerede echelon form.

c)

Til en start skal jeg vise v0 og v1 er ortogonale.

Her bruger jeg proposition 8.10 og beregner de indre produkter

```
v0 = np.array([1.0,   1.0,  0.0])
v1 = np.array([1.0,  -1.0,  1.0])
print("v0 og v1 er ortogonale da deres indre produkt er: \n ", np.dot(v0,v1))
```

```
## v0 og v1 er ortogonale da deres indre produkt er:
##   0.0
```

Herefter skal vi beregne projektionen af w.

```
w = np.array([1.0, 1.0, -1.0])

v0_norm = np.linalg.norm(v0)
v1_norm = np.linalg.norm(v1)

prov0 = (np.dot(v0, w) / v0_norm**2) * v0
prov1 = (np.dot(v1, w) / v1_norm**2) * v1

Px = prov0 + prov1

print("Projektionen er:  =\n", Px)
```

```
## Projektionen er:  =
##  [ 0.66666667  1.33333333 -0.33333333]
```

Opgave 6)

a)

I denne opgave skal vi ind og benytte os af de teknikker vi har lært i kap 16 om mindste kvadraters metode, som i bund og grund handler om at estimere parameter udfra nogle givet datapunker.

Derfor er ønske scenairet at vi sil løse en eksakt løsning til ligning systemet Aw=v og få de præcise værdier.

Det kan oftes dog ikke lade sig gøre da man har flere ligninger end variabler. Det gør sig også gældende i dette tilfælde da vi har 7 ligninger med 4 ubekendte.

Fordi vi ikke kan løse systemet så kigger vi på restvektoren og ønske den skal være så lille som muligt.

b)

Til en start sætter jeg matricen a som følger opgavebeskrivelsen:

```
x = np.array([0.2,  0.3, 1.1, 1.3, 2.2, 2.4,  2.5])
y = np.array([1.9, -0.9, 0.0, 2.4, 2.6, 0.1, -0.1])
a = np.vander(x, 4)
print(f'vores a matrice: \n {a}')
```

```
## vores a matrice:
##  [[8.0000e-03 4.0000e-02 2.0000e-01 1.0000e+00]
##  [2.7000e-02 9.0000e-02 3.0000e-01 1.0000e+00]
##  [1.3310e+00 1.2100e+00 1.1000e+00 1.0000e+00]
##  [2.1970e+00 1.6900e+00 1.3000e+00 1.0000e+00]
##  [1.0648e+01 4.8400e+00 2.2000e+00 1.0000e+00]
##  [1.3824e+01 5.7600e+00 2.4000e+00 1.0000e+00]
##  [1.5625e+01 6.2500e+00 2.5000e+00 1.0000e+00]]
```

Herefter skal vi beregne singulærværdierne og konditionstallet.

Her benytter jeg mig af afsnit 16.3 og lignign 17.2 til mine beregninger.

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

kappa_a = s[0] / s[-1]

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

```
## Værdien af s, som er vores singulærværdier:
##  [25.99958745  2.32596954  0.83772044  0.08837796]
```

```
print(f'Vores konditionstal \n {kappa_a}')
```

```
## Vores konditionstal
##  294.1863285756089
```

Således har vi fundet frem til vores singulærværdier og konditionstal.

c)

I denne opgave skal vi løse systemet med SVD.

Til det implemeter jeg ligning 16.5 og benyttter mig af algoritmen på side 186.

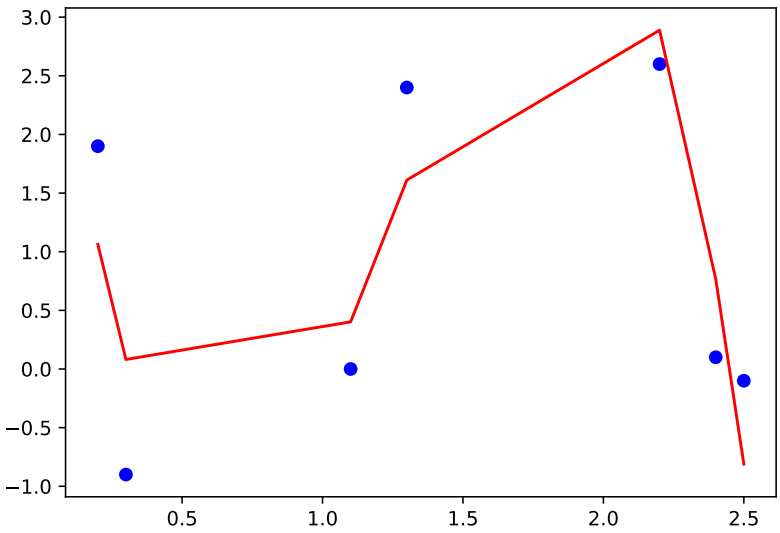
```
koeffs_svd = vt.T @ (np.diag(1 / s) @ (u.T @ y[:, np.newaxis]))

print(f'Vores koefficienter \n {koeffs_svd}')
```

```
## Vores koefficienter
##  [[ -5.18266996]
##  [ 19.63664881]
##  [-18.64277796]
##  [  4.0466531 ]]
```

Som er vores koefficienter.

d)

```
xaxis = np.linspace(0, 6.5, 100)
fig, ax = plt.subplots()
ax.plot(x, y, 'bo') #atl ax.scatter(x,y)
ax.plot(x, np.vander(x, 4) @ koeffs_svd, 'r');
plt.show()
```



Her ser vi vores plot af vores model.

Vores løsning er ikke for god. Vi rammer ikke punkter særlig godt.

Det kan tyde på at vores metode ikke er optimal og vi skal finde på en anden løsning.

Opgave 7)

a)

VI har systemer skrevet ud

$$\begin{bmatrix} \frac{-8}{320} & \frac{3}{160} \\ \frac{8}{320} & \frac{-8}{160} \end{bmatrix}$$

Jeg har svært ved at afgøre hvad c skal være. Hvis jeg læser opgave c, så kan jeg se at A=cB, derfor vil jeg så mene at ud fra løse det system kan vi beregne hvad c skal være.

Så hvis jeg gør det med den antagelse at jeg skal løse A=cB:

$$\frac{-8}{320} = c \cdot (-4)$$

Hvorfra jeg får $c = \frac{1}{160}$.

b)

Vi har fra opgavenbeskrivelsen giver B matricen og skal her finde egenværdier og egenvektor.

For at gøre det benytte jeg først proposition 21.15 til at beregne det karakteristisk polynomium.

$$det(B - \lambda I_n) = 0$$

$$\begin{bmatrix} -4-\lambda & 3 \\ 4 & -8-\lambda \end{bmatrix} = \lambda^2 + 12\lambda + 20$$

Hvor vi har et anden grads polynomium og så kan vi bruge

$$\frac{-12(+,-)\sqrt{12^2 - 4 * 1 * 20}}{2 * 1}$$

Som giver mig egenværdierne:

$$\lambda_1 = -2 \ \& \ \lambda_2 = -10$$

Fra egenværdier kan vi så beregne egenvektor. Jeg benytter eksempel 21.5 til dette formål.

For at spare tid undlade jeg at skrive matricerne op, men forklare fremgangsmåden.

Jeg tager for hver egenværdi og indsætter i funktionen $B - \lambda_i I$, hvorefter vi bruger række operation til at løse systemet til vi kommer til pivot elementerne.

Jeg viser for den første:

$$\begin{bmatrix} -4-\lambda & 3 \\ 4 & -8-\lambda \end{bmatrix}$$

$$\begin{bmatrix} -4-(-2) & 3 \\ 4 & -8-(-2) \end{bmatrix} = \begin{bmatrix} -2 & 3 \\ 4 & -6 \end{bmatrix}$$

Hvor vi reducer det til

og får egenvektor (-1,2).

Det samme gør vi for den anden og får egenvektor (3,2).

c)

I denne opgave skal jeg bestemme egenværdier og egenvektor for A, hvor jeg gør brug af mit resultat fra b.

Da jeg ved c = 1/160 så kan jeg bare gange det på egenværdier så får jeg egenværdier og egenvektor for A

```
print(f'første egenværdi \n {-2 * (1/160)}')
```

```
## første egenværdi
##  -0.0125
```

```
print(f'Anden egenværdi \n {-10 * (1/160)}')
```

```
## Anden egenværdi
##  -0.0625
```

Så for at løse for egenvektor så får jeg lidt nogle andre resultater end før. Men for at skrive den generelle løsning op så benytter jeg mig af dem fra opgave b.

Den generelle løssning kan skrives op som følgende af proposition 22.2.

$$y(t) = c_0 \begin{bmatrix} -1 \\ 2 \end{bmatrix} e^{-0.0125} + c_1 \begin{bmatrix} 3 \\ 2 \end{bmatrix} e^{-0.0625}$$

Ovenstående har vi den generelle løsning.

d)

I denne del skal vi i python løse den generelle løsning med start værdier 140 og 0. Samt plotte kvotienten.

Igen gør jeg brug af proposition 22.2.

Udregning kan laves i python:

```
A = np.array([
  [-1.0, 3.0],
  [2.0, 2.0]])

start = np.array([
  [140],
  [0]])

print(f'Bestem c0 og c1 i den gennerelle løsning: \n {np.linalg.inv(A).dot(start)}')
```

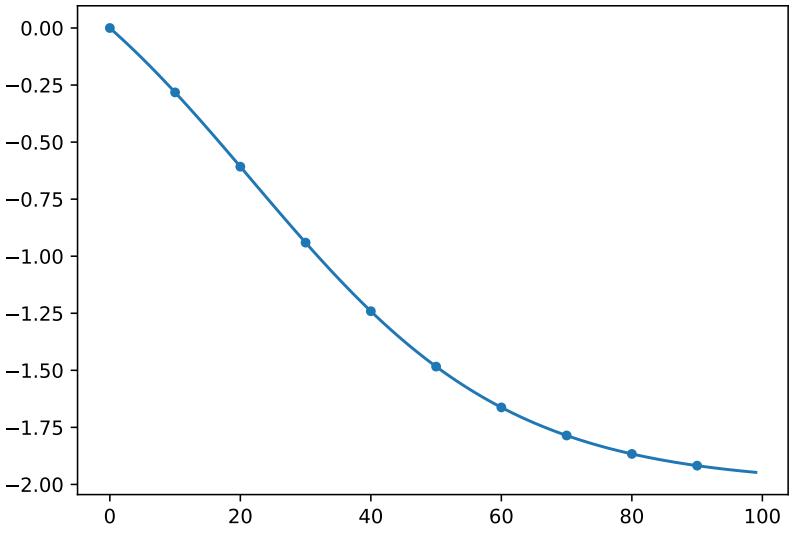
```
## Bestem c0 og c1 i den gennerelle løsning:
##  [[-35.]
##  [ 35.]]
```

Skrevet på fomlen får vi følgende:

$$y(t)$$

Nu kan vi så lave et plot af vores kvotienter som funktion af t. Det gør jeg her:

```
v0 = np.array([-1., 2.])[:, np.newaxis]
v1 = np.array([3. , 2.])[:, np.newaxis]

lambda0 = -0.0125
lambda1 = -0.0625

t = np.linspace(0, 100, 100)

løsning = (-35 * v0 * np.exp(lambda0 * t) + 35 * v1 * np.exp(lambda1 * t))

plt.plot(løsning[1]/løsning[0], marker='o', markevery=10, markersize=4)
plt.show()
```



Her har jeg implementeret løsning og ser på $y_1(t)/y_0(0)$.