Aflevering 3

Lucas Bagge

2021-03-05

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

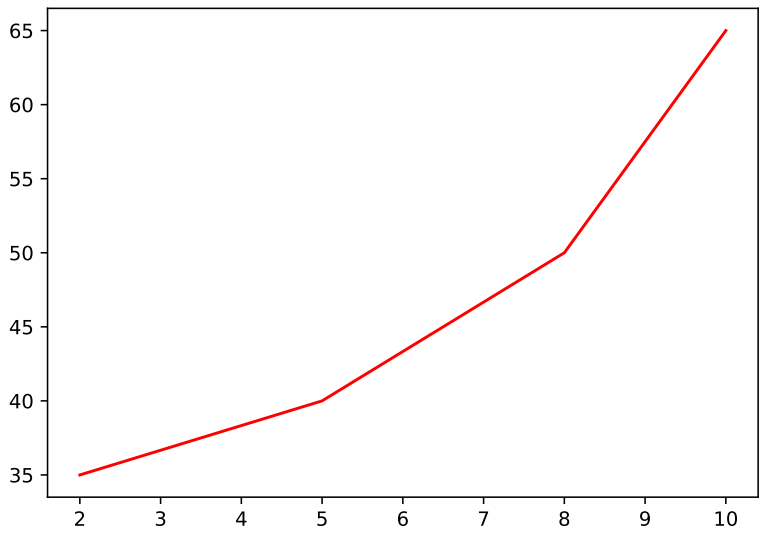
a) Plot grafen svarende til de overstående datapunkter.

Ud fra opgave beskrivelse får vi angiver nogle data punkter for en drone vi er ved at bygge.

Data består af tid og temperatur for droen.

Her skal vi starte med at visualizer det.

```
x = np.array([2.0, 5.0, 8.0, 10.0])
y = np.array([35.0, 40., 50., 65.])
fig, ax = plt.subplots()
ax.plot(x, y, 'r')
plt.show()
```



I ovenstående plot ser vi vores datapunkter.

b) Opstil et lineært ligningssystem for at p går igennem de sidste tre datapunkter. Løs ligningssystemet ved hjælp af elementære rækkeoperationer. Plot resultatet

I den anden del af opgaven opstiller jeg mit koefficient matrict for systemet og benytter mid af Andrew python metodik til at løse systemet.

```
a = np.array([[1, 5, 25, 40],
              [1, 8, 64, 50],
              [1, 10, 100, 65]])
```

```
aub = a
aub
```

```
## array([[  1,   5,  25,  40],
##        [  1,   8,  64,  50],
##        [  1,  10, 100,  65]])
```

```
aub[1, :] += (-1) * aub[0, :]
aub
```

```
## array([[  1,   5,  25,  40],
##        [  0,   3,  39,  10],
##        [  1,  10, 100,  65]])
```

```
aub[2, :] += (-1) * aub[0, :]
aub
```

```
## array([[ 1,  5, 25, 40],
##        [ 0,  3, 39, 10],
##        [ 0,  5, 75, 25]])
```

For bytte rundt om rækkerne så laver jeg en funktionen `switch_rows` til det formål.

```
def switch_rows(A,i,j):
    "Switch rows i and j in matrix A."
    n = A.shape[0]
    E = np.eye(n)
    E[i,i] = 0
    E[j,j] = 0
    E[i,j] = 1
    E[j,i] = 1
    return E @ A
```

```
aub = switch_rows(aub, 1, 2)
aub
```

```
## array([[ 1.,  5., 25., 40.],
##        [ 0.,  5., 75., 25.],
##        [ 0.,  3., 39., 10.]])
```

```
aub[1,:] *= 1/5
aub
```

```
## array([[ 1.,  5., 25., 40.],
##        [ 0.,  1., 15.,  5.],
##        [ 0.,  3., 39., 10.]])
```

```
aub[2,:] += (-3 * aub[1,:])
aub
```

```
## array([[ 1.,  5., 25., 40.],
##        [ 0.,  1., 15.,  5.],
##        [ 0.,  0., -6., -5.]])
```

```
aub[2,:] *= -1/6
aub
```

```
## array([[ 1.        ,  5.        , 25.        , 40.        ],
##        [ 0.        ,  1.        , 15.        ,  5.        ],
##        [-0.        , -0.        ,  1.        ,  0.83333333]])
```

```
aub[1,:] += - 15 * aub[2,:]
aub
```

```
## array([[ 1.        ,  5.        , 25.        , 40.        ],
##        [ 0.        ,  1.        ,  0.        , -7.5       ],
##        [-0.        , -0.        ,  1.        ,  0.83333333]])
```

```
aub[0,:] += - 25 * aub[2,:]
aub
```

```
## array([[ 1.        ,  5.        ,  0.        , 19.16666667],
##        [ 0.        ,  1.        ,  0.        , -7.5       ],
##        [-0.        , -0.        ,  1.        ,  0.83333333]])
```

```
aub[0,:] += - 5 * aub[1,:]
aub.round(2)
```

```
## array([[ 1.  ,  0.  ,  0.  , 56.67],
##        [ 0.  ,  1.  ,  0.  , -7.5 ],
##        [-0.  , -0.  ,  1.  ,  0.83]])
```

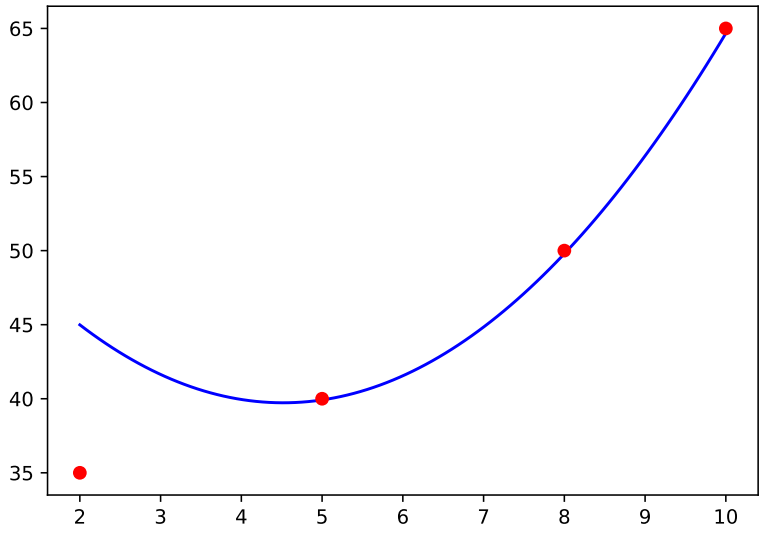
Nu har vi ved hjælp af rækkeoperationer fået udregnet løsningerne til systemet.

$$a = 56.67b = -7.5c = 0.83$$

Yderligere bliver vi bedt om at plotte ovenstående

Laver efterfølgende plottet af de to optimale punkter.

```
x1 = np.linspace(2, 10, 100)
y1 = 56.67 - 7.5 * x1 + 0.83 * x1 **2
fig, ax = plt.subplots()
ax.plot(x1, y1, color = "blue")
ax.plot(x, y, linestyle = '', marker = 'o', color = 'red')
plt.show()
```



Jeg har løst lignignsystemet og lave ovenstående plot. Dog mistænker jeg der er en fejl et eller andet sted, da polynomiet ikke ligger oven på punkterne.

c) Hvis man vil have et polynomium p(x) der går igennem alle 4 datapunkter, hvad er den mindst mulige grad for p? Bestem sådan et polynomium, denne gang må I bruge np.linalg.solve(), og plot resultatet.

For at udvælge den mindste mulige grad vælger jeg at tage 3. Dermed skal vi have et polynomium på formen:

$$p(x) = a + bx + cx^2 + dx^3$$

Nu skal jeg således definere mit koefficientmatrix, løsning og løse ligninssystemet, som vi må gøre med `np.linalg.solve`.

```
x2 = np.array([
  [1, 2, 4, 8],
  [1, 5, 25, 125],
  [1, 8, 64, 512],
  [1, 10, 100, 1000]
])

y2 = np.array([35, 40, 50, 65])[:, np.newaxis]

np.linalg.solve(x2,y2)
```

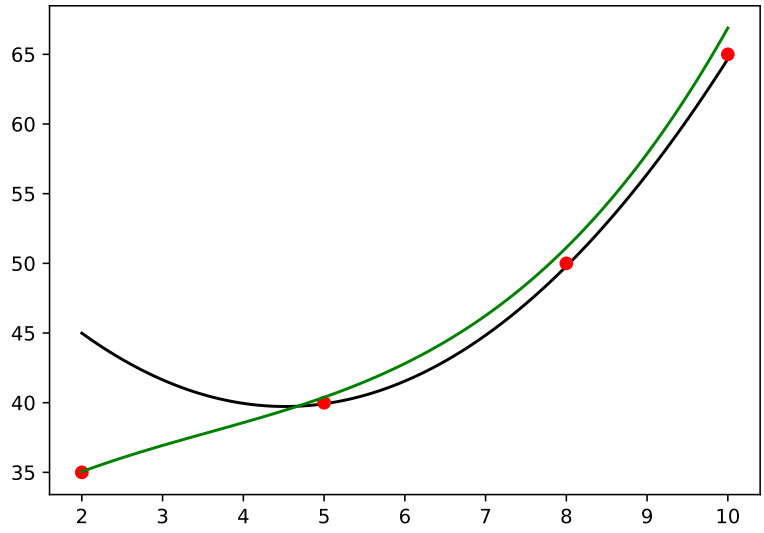
```
## array([[28.88888889],
##        [ 4.30555556],
##        [-0.76388889],
##        [ 0.06944444]])
```

Dermed bliver mit polynomium:

$$p(x) = 28.89 + 4.31x - 0.76x^2 + 0.07x^3$$

Lad os som før også plotte dette:

```
y2 = 28.89 + 4.3 * x1 - 0.75 * x1 ** 2 + 0.07 * x1 **3
fig, ax = plt.subplots()
ax.plot(x1, y1, color = "black")
ax.plot(x, y, linestyle = '', marker = 'o', color = 'red')
x1 = np.linspace(2, 10, 100)
ax.plot(x1, y2, color = 'green')
plt.show()
```



Her ser vi et meget bedre resultat end det forrig.

d) Opstil et lineært ligningssystem for en funktion

$$f(x) = p_1(x),\ for\ 5.\ \leq x \leq 8. f(x) = p_2(x),\ for\ 8.\ \leq x \leq 5.$$

hvor p1, p2 er andengradspolynomier således at

i) p1 går igennem datapunkterne ved tid 5,0 og 8,0,

ii) p2 går igennem datapunkterne ved tid 8,0 og 10,0, og

iii) i tid 8,0 har p1 og p2 den samme hældning.

Her til en start kan vi se at $p_1(x)$ og $p_2(x)$ skal begge gå igennem (8,55).
Ud fra iii) har vi den sammen hældming som medføre

$$p_1'(8) = b_1 + 16c_1 = p_2'(8) = b_2 + 16c_2 \iff b_1 + 16c_1 - b_2 - 16b_2 = 0$$

Dermed bliver min udvidede koefficientmatrix:

```
a = np.array([
  [1, 5, 25, 0, 0, 0, 40],
  [1, 8, 64, 0, 0, 0, 50],
  [0, 0, 0, 1, 8, 64, 50],
  [0, 0, 0, 1, 10, 100, 65],
  [0, 1, 16, 0, -1, -16, 0]
  ])
a
```

```
## array([[  1,   5,  25,   0,   0,   0,  40],
##        [  1,   8,  64,   0,   0,   0,  50],
##        [  0,   0,   0,   1,   8,  64,  50],
##        [  0,   0,   0,   1,  10, 100,  65],
##        [  0,   1,  16,   0,  -1, -16,   0]])
```

e) Vis a systemet har mere end en løsning.

Her udfører vi række operationer så vi får løsningen:

```
a_solve = np.array([
  [1, 0, 0, 0, 0, 80/3, 710/9],
  [0, 1, 0, 0, 0, -26/3, -265/18],
  [0, 0, 1, 0, 0, 2/3, 25/18],
  [0, 0, 0, 1, 0, -80, -10],
  [0, 0, 0, 0, 1, 18, 15/2]
  ])
a_solve.round(2)
```

```
## array([[  1.  ,   0.  ,   0.  ,   0.  ,   0.  ,  26.67,  78.89],
##        [  0.  ,   1.  ,   0.  ,   0.  ,   0.  ,  -8.67, -14.72],
##        [  0.  ,   0.  ,   1.  ,   0.  ,   0.  ,   0.67,   1.39],
##        [  0.  ,   0.  ,   0.  ,   1.  ,   0.  , -80.  , -10.  ],
##        [  0.  ,   0.  ,   0.  ,   0.  ,   1.  ,  18.  ,   7.5 ]])
```

Herfra kan vi se at der ikke er et pivotelement i søjle 7 og fra 6.4 medføre det at der er uendelig mange løsninger.

f) Ved at sætte én betingelse på den afedede af p1 i 5,0, dan et system med

en entydig løsning og plot den resulterende funktion f. Gør rede for jeres valg af betingelse på p1.

$$p_1(x) = a + bx + cx^2$$

$$p_1'(x) = b + 2 \cdot x$$
$$p_1'(2) = 0$$

Her ved får vi følgende system

Så får jeg en nyt system som jeg også løser:

```
a = np.array([
  [1, 5, 25, 0, 0, 0],
  [1, 8, 64, 0, 0, 0],
  [0, 0, 0, 1, 8, 64],
  [0, 0, 0, 1, 10, 100],
  [0, 1, 16, 0, -1, -16],
  [0, 1, 10, 0, 0, 0]
  ])
b = np.array([40,50,50,65,0,2])[:, np.newaxis]
```

Ved at løse systemet får vi den entydig løsning:

```
np.linalg.solve(a,b)
```

```
## array([[ 41.11111111],
##        [ -2.44444444],
##        [  0.44444444],
##        [103.33333333],
##        [-18.        ],
##        [  1.41666667]])
```

Her for oven har vi valgt betingelsen p'(2)=0. Det har jeg gjort ved en snak med Simon.

g) Hvilke af modellerne (b), (c) eller (f) vil I helst bruge for at estimere hvornår temperaturen passerer 55 C?

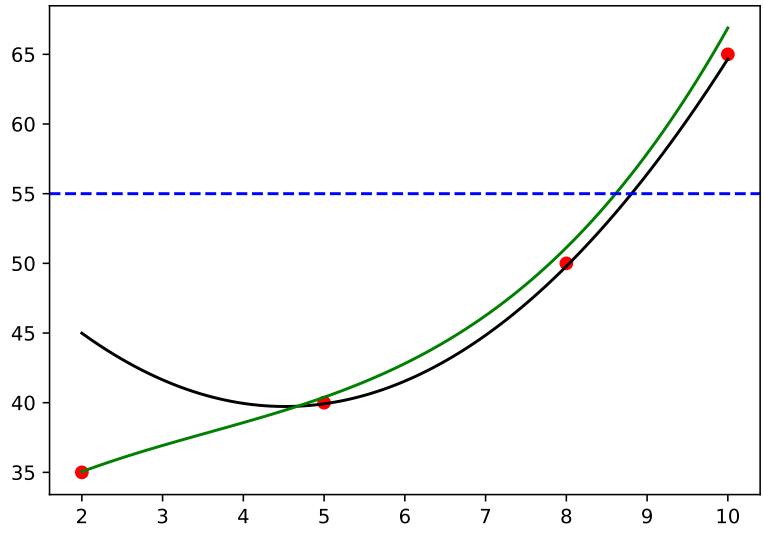
Her i den sidste opgave skal vi samle vores modeller og udfra det vurdere hvilken model vi vil vælge.

$$p(x) = 28.89 + 4.31x - 0.76x^2 + 0.07x^3$$

```
# plot b i grøn, c i sort
n = 2
fig, ax = plt.subplots()
ax.plot()
```

```
## []
```

```
x1 = np.linspace(2, 10, 100)
ax.plot(x, y, linestyle = '', marker = 'o', color = 'red')
ax.plot(x1, y1, color = "black")
ax.plot(x1, y2, color = 'green')
plt.axhline(y = 55, color = "blue", linestyle = "--")
plt.show()
```



Her vil jeg vælge den model jeg har lavet i opagev b, da det fitter bedre til data. Der kan dog være et problem at vi overfitter til vores data. At overfitte er et begreb vi kommer ind på i forhold til kurset Machine Learning and Statistical Learning.