Pythonoving5(1)

October 13, 2024

Pythonøving 5 - TKT4118/MEKT1101 Mekanikk 1

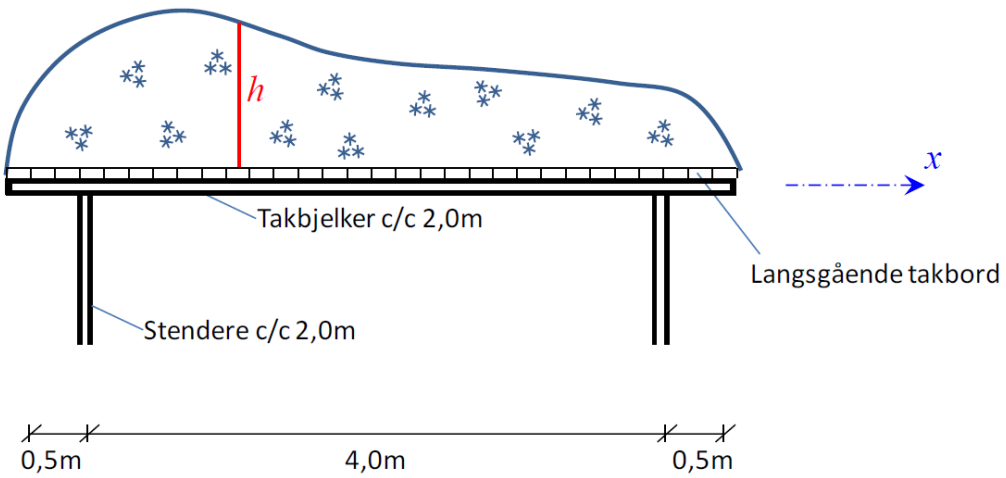
Snølast 2

I denne oppgaven skal vi fortsette med den samme problemstillingen som vi var innom i forrige Pythonøving: Det dreier seg om snø på et flatt tak på et uthus, se figur nedenfor, og vi skal analysere snølasten på takbjelken. Senteravstand (ut av planet) mellom takbjelkene er c/c 2,0 m. Måledataene for snøen (koordinat x [m] og vertikal snødybde h [m]) er stadig gitt i fila snødybde.txt. Densiteten til snøen er $\rho_{sn} = 400$ kg/m^3.

Som nevnt i oppgaveteksten, skal det brukes regresjon for å tilpasse måledataen til et polynom som videre kan evalueres for ønskede x-verdier.

---

1 Definisjon av parametre



Nok en gang er snømålingene lagret i en tekstfil med navn "snødybde.txt", men denne gangen må du lese inn filen selv. Tekstfilen er fortsatt på formen:

x-koordinat [m]	Snødybde [m]
0.0	0.0
0.5	0.95
1.0	1.25
1.5	1.30
2.0	1.20
2.5	1.10
3.0	1.05
3.5	0.95
4.0	0.85
4.5	0.55
5.0	0.0

Før snødatane kan tilpasses et polynom, må flere parametre allerede være definert. Gjenta de første stegene som ble gjort i pythonøving 4. Hvis du ikke har gjort øvingen og trenger hjelp, kan du finne løsningsforslag på Blackboard, eller se hintet nedenfor.

Linjelasten $q(x)$ fra snøen er fortsatt:

$$q(x) = \rho_{sn} * h(x) * c/c * g$$

- Importer verktøyene for beregning (Numpy) og plotting (Matplotlib).
- Definer konstanter som trengs
- Les tekstfilen med snodata (Se hint under!)
- Beregn linjelasten $q(x)$

---

[1]:
```
#importerer viktige biblioteker
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
#definerer parametere
g = 9.82  #[m/s^2]
rho = 400. #[kg/m^3]
senteravstand = 2.0 #[m]
```

[4]:
```
sno_data = np.genfromtxt("snodybde.txt")
```

[5]:
```
x = sno_data[:,0]
h = sno_data[:,1]

q = rho * h * senteravstand * g
```

```
print(q)
```

```
[    0.   7463.2  9820.  10212.8  9427.2  8641.6  8248.8  7463.2  6677.6
  4320.8     0. ]
```

Trykk for hjelp med å lese inn .txt filen

```
sno_data = np.genfromtxt("snodybde.txt")  # Les tekstfilen med snodata
```

2 Regresjon

Vi skal nå se på hvordan vi kan "forbedre" snødataene ved bruk av regresjon.

Linjelasten er bare beskrevet for noen få punkter. Vi kan gjøre et estimat av hva linjelasten er mellom punktene ved å anta at kurven er glatt og dermed at et polynom kan tilnærme linjelasten. Hvis vi har et polynom som beskriver linjelasten kan vi enkelt regne ut verdien av linjelasten ved så mange punkter vi ønsker!

For å finne et passende polynom kan vi bruke ferdige funksjoner gjort tilgjengelig i Numpy. Den første av disse er er `np.polyfit("x-verdier", "y-verdier", "polynomorden")`. Den finner koeffisientene som gir best mulig passform på polynomet. Den gir tilbake et array med koeffisienter:

$$\text{koeff} = \text{polyfit}(x_i, f(x_i), \text{polynomorden})$$

hvor funksjonen vi tilpasser er et polynom bygd opp som

$$f(x) = \text{koeff}[0] + \text{koeff}[1] \cdot x + \text{koeff}[2] \cdot x^2 + ... + \text{koeff}[n] \cdot x^n$$

Disse koeffisientene kan gis videre til en funksjon som heter `np.poly1d("koeffisienter")`, som lager en funksjon som lar oss evaluere polynomet for de verdiene vi ønsker.

Under er et generisk eksempel på en slik prosedyre:

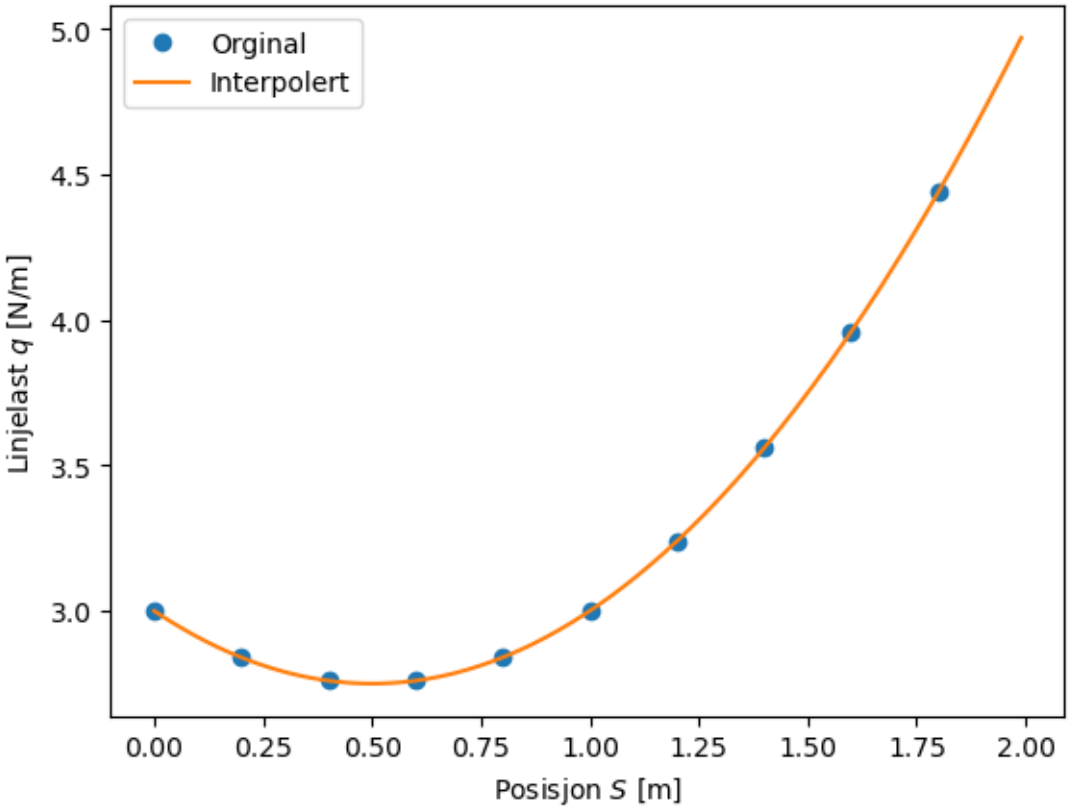
[6]:
```
### EKSEMPEL ###
x_ex = np.arange(0.,2.,0.2) # Vilkårlig eksempeldata for x . x går fra 0 til 2,␣
 ↪med steglengde 0.2.
q_ex = x_ex**2.-x_ex+3.     # Vilkårlig eksempeldata for q(x)

# Lag et array med samme endepunkter som x_ex men med mindre skritt.
x_fin = np.arange(0.,2.,0.01)  # x går fra 0 til 2, med steglengde 0.01.

# Finn koeffisientene til et 5. grads polynom og evaluer det for alle verdier i␣
 ↪x_fin
koeff_ex   = np.polyfit(x_ex,q_ex,5)
polynom_ex = np.poly1d(koeff_ex)   # polynom_ex er nå en funksjon q(x) som kan␣
 ↪evalueres som:
q_ex_fin   = polynom_ex(x_fin)     # Funksjonen blir så evaluert for x definert␣
 ↪med kort steglengde.
```

Så kan resultatene plottes som:

[7]:
```
plt.plot(x_ex, q_ex,'o', label="Orginal")
plt.plot(x_fin,q_ex_fin,label="Interpolert")
plt.xlabel("Posisjon $S$ [m]")
plt.ylabel("Linjelast $q$ [N/m]")
plt.legend()
plt.show()
```



Bruk fremgangsmåten presentert ovenfor til å interpolere linjelasten ved bruk av et 5. grads polynom og evaluer det for x-verdier med kortere intervaller for å få flere datapunkter.

- Definer et nytt array x_{fin} med steglengde $\Delta x = 0.05$ m. Dette tilsvarer 10 ganger så mange punkter som den originale målingen.
- Utfør kurvetilpassningen av polynomet og evaluer polynomet med den kortere steglengden.
- Plot de interpolerte målingene sammen med de originale for å se forskjellen.

Bruk kodecellen under.

---

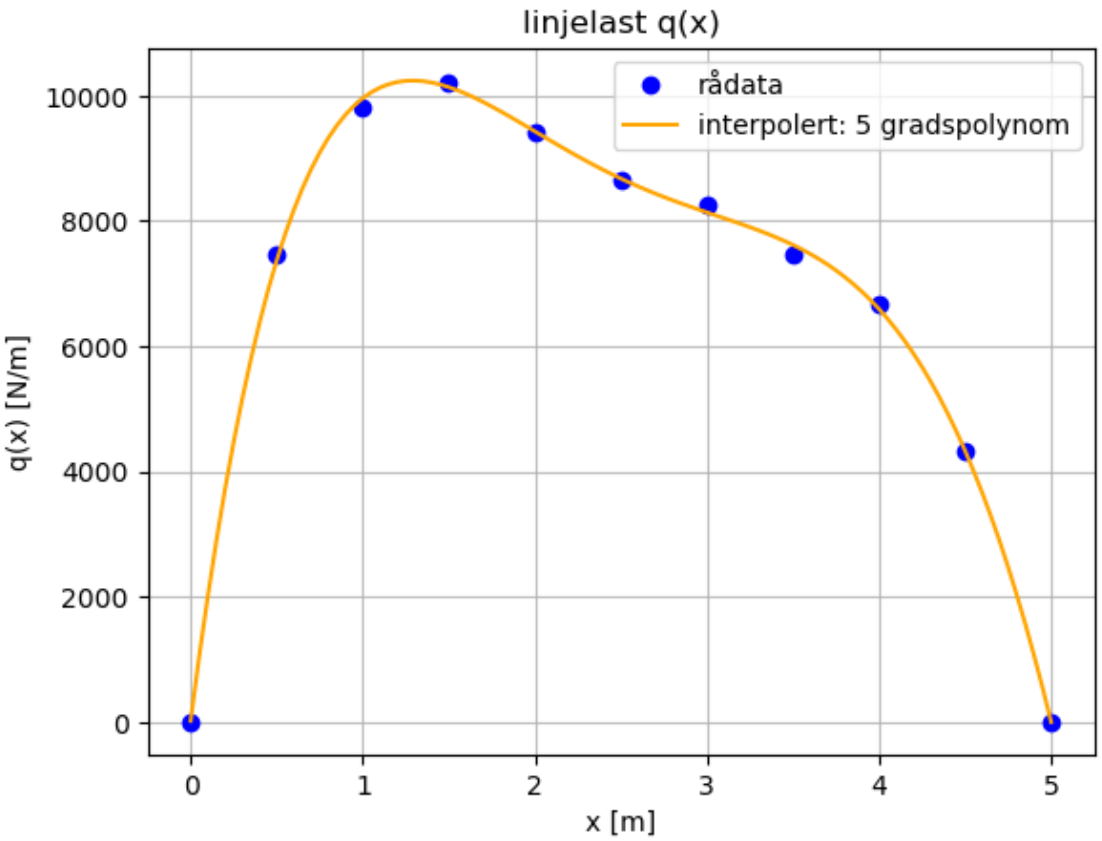
[8]:
```
#lager 5. ordens polynom regresjon for linjelasten
dx = 0.05
order = 5

x_fin_5 = np.arange(x[0],x[-1]+dx,dx)
print(x_fin_5)

coeff = np.polyfit(x, q, order)
polynom_func = np.poly1d(coeff)
q_fin_5 = polynom_func(x_fin_5)
```

```
[0.   0.05 0.1  0.15 0.2  0.25 0.3  0.35 0.4  0.45 0.5  0.55 0.6  0.65
 0.7  0.75 0.8  0.85 0.9  0.95 1.   1.05 1.1  1.15 1.2  1.25 1.3  1.35
 1.4  1.45 1.5  1.55 1.6  1.65 1.7  1.75 1.8  1.85 1.9  1.95 2.   2.05
 2.1  2.15 2.2  2.25 2.3  2.35 2.4  2.45 2.5  2.55 2.6  2.65 2.7  2.75
 2.8  2.85 2.9  2.95 3.   3.05 3.1  3.15 3.2  3.25 3.3  3.35 3.4  3.45
 3.5  3.55 3.6  3.65 3.7  3.75 3.8  3.85 3.9  3.95 4.   4.05 4.1  4.15
 4.2  4.25 4.3  4.35 4.4  4.45 4.5  4.55 4.6  4.65 4.7  4.75 4.8  4.85
 4.9  4.95 5.  ]
```

[9]:
```
#plotter rådataene og polynomet evaluert med kortere intervaller
fig, ax = plt.subplots()
ax.plot(x,q, 'o', color = 'blue', label = 'rådata')
ax.plot(x_fin_5, q_fin_5, color = 'orange', label = 'interpolert: 5␣
 ↪gradspolynom')
ax.legend()
ax.set_title("linjelast q(x)")
ax.set_xlabel("x [m]")
ax.set_ylabel("q(x) [N/m]")
ax.grid(alpha = 0.8)
plt.show()
```



Trykk for tips

```
sno_posisjon_fin = np.arange(0, 5, 0.05)
koeff = np.polyfit(sno_posisjon, linjelast, 5)
```

Gjenta regresjonen med et 3. gradspolynom og beregn resultantkraften for: * De originale målingene * 5. gradspolynomet * 3. gradspolynomet

Resultantkraften kan stadig regnes ut fra:

$$R_{sn} = \Delta x \sum_{i=1}^{n} q(x_i)$$

- Plot de tre tilfellene i samme graf. Holder det å tilpasse med et 3. gradspolynom?

---

[10]:
```
# lager 3. ordenspolynom for linjelasten
n_order = 3

x_fin_3 = np.arange(x[0],x[-1]+dx,dx)
```

```
print(x_fin_3)

coeff_3 = np.polyfit(x, q, n_order)
polynom_func_3 = np.poly1d(coeff_3)
q_fin_3 = polynom_func_3(x_fin_3)
```

```
[0.   0.05 0.1  0.15 0.2  0.25 0.3  0.35 0.4  0.45 0.5  0.55 0.6  0.65
 0.7  0.75 0.8  0.85 0.9  0.95 1.   1.05 1.1  1.15 1.2  1.25 1.3  1.35
 1.4  1.45 1.5  1.55 1.6  1.65 1.7  1.75 1.8  1.85 1.9  1.95 2.   2.05
 2.1  2.15 2.2  2.25 2.3  2.35 2.4  2.45 2.5  2.55 2.6  2.65 2.7  2.75
 2.8  2.85 2.9  2.95 3.   3.05 3.1  3.15 3.2  3.25 3.3  3.35 3.4  3.45
 3.5  3.55 3.6  3.65 3.7  3.75 3.8  3.85 3.9  3.95 4.   4.05 4.1  4.15
 4.2  4.25 4.3  4.35 4.4  4.45 4.5  4.55 4.6  4.65 4.7  4.75 4.8  4.85
 4.9  4.95 5.  ]
```

[13]:
```
#kraftresultant
dx_original = (x[-1] - x[0])/(len(x)-1)
R_original = 0.0
R_3 = 0.0
R_5 = 0.0

#bruker trapesmetoden
for i in range(len(x)-1):
    R_original = R_original + dx_original * (q[i] + q[i+1])/2

for i in range(len(x_fin_5)-1):
    R_3 = R_3 + dx *(q_fin_3[i] + q_fin_3[i+1])/2
    R_5 = R_5 + dx * (q_fin_5[i] + q_fin_5[i+1])/2

print(f'R_original = {R_original}')
print(f'R_3 = {round(R_3,1)}')
print(f'R_5 = {round(R_5,1)}')
```

```
R_original = 36137.6
R_3 = 35737.5
R_5 = 36803.7
```

[14]:
```
#plotter rådataene og polynomet evaluert med kortere intervaller
fig, ax = plt.subplots()
ax.plot(x,q, '-o', color = 'blue', label = 'rådata')
ax.plot(x_fin_3, q_fin_3, color = 'red', label = 'interpolert: 3. gradspolynom')
ax.plot(x_fin_5, q_fin_5, color = 'orange', label = 'interpolert: 5.␣
 ↪gradspolynom')
ax.legend()
ax.set_title("linjelast q(x)")
ax.set_xlabel("x [m]")
ax.set_ylabel("q(x) [N/m]")
```

```
ax.grid(alpha = 0.8)
plt.show()
```