Av: Ellen Rekaa

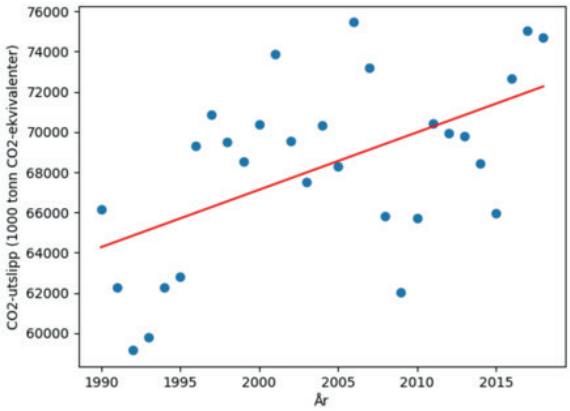
Opplegg 11 – **Temperaturendring i solfanger og regresjon**

Lineær regresjon med Python

Dere har allerede utført mange lineære regresjoner med Geogebra, så hva er poenget med å bruke Python i stedet?

I Geogebra må vi skrive inn alle målingene våre for hånd, eller kopiere fra f.eks. Excel, og da kan det være vanskelig å gjøre en regresjon dersom vi har mye data. Tenk deg at du har 100 målinger, da tar det veldig lang tid å skrive inn alle målingene i regnearket i Geogebra. Det kan fort bli en feil eller to underveis også. Tenk om du har enda flere målinger, kanskje flere enn 1000! Da blir det ganske kjedelig å skrive inn alle tallene.

Samlet klimagassutslipp for Norge 1990 – 2018



I Python kan du bruke data som ligger i ferdige filer. Du slipper å skrive dem inn! Det som er viktig er at filene er riktig type. Slik som med plotting av grafer i Python.

En annen fordel med å bruke Python, er at det er et programmeringsspråk som blir brukt av forskere som jobber med modellering. Andre forskere kan bruke andre programmeringsspråk som er spesialtilpasset akkurat den modelleringen de jobber med, men de følger de samme prinsippene som Python. Så dersom du kan modellere med Python, er det enklere å modellere i et annet programmeringsspråk enn om du bare har brukt Geogebra tidligere.

Regresjon i Python består av seks deler:

1. Importere de bibliotekene som trengs
2. Importere data fra en .txt-fil
3. Legge dataene i en tabell for x-verdiene og en for y-verdiene
4. Lage en programmeringsfunksjon som sier at vi skal bruke en lineær modell i regresjonen
5. Utføre regresjonen, og vise på skjermen hva den matematiske modellen blir, stigningstallet (a) og skjæringspunktet med y-aksen (b)
6. Tegne grafen med resultatet – tilpass koden for plotting av graf
   - Plotte punktene med et scatterplot
   - Plotte regresjonsgrafen – hvordan få programmet til å regne ut y-verdiene for modellen?

Puslespilloppgave
Sorter kodelinjene etter numrene i tekstboksen over, om regresjon i Python. Da vil du få den riktige rekkefølgen i programmet ditt.

1
```
co2data = loadtxt("co2utslipp.txt")
```

2
```
from pylab import *
from scipy.optimize import curve_fit
```

3
```
scatter(x_verdier, y_verdier)
plot(x_verdier, funksjon(x_verdier, a, b), color="red")
title("Samlet klimagassutslipp for Norge 1990 - 2018")
xlabel("År")
ylabel("CO2-utslipp (1000 tonn CO2-ekvivalenter)")
show()
```

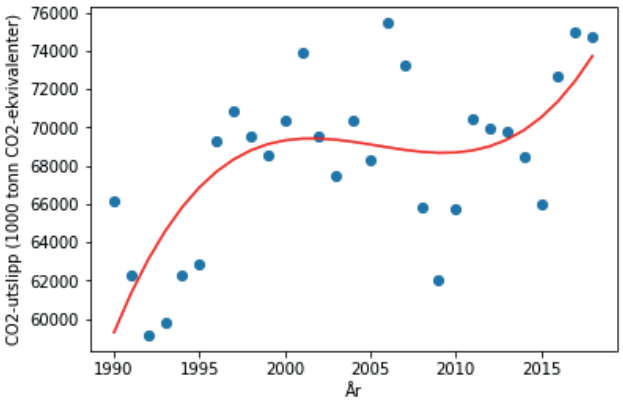
4
```
koeffisienter, kovarianser = curve_fit(funksjon, x_verdier, y_verdier)
a, b = koeffisienter
print(a,b)
```

5
```
x_verdier = co2data[:,0]
y_verdier = co2data[:,1]
```

6
```
def funksjon(x, a, b):
    return (a * x) + b
```

Ikke-lineær regresjon med Python

Forskjell på lineær og ikke-lineær regresjon i Python



Det er liten forskjell på hvordan programmene for lineær og ikke-lineær regresjon ser ut. Den viktigste forskjellen er å forandre på hvilken funksjonstype datasettet skal tilpasses. Det står en oversikt over noen funksjonstyper, og hvordan man skriver dem i python på de to neste sidene.

Men hvordan skal vi vite hvilken funksjonstype som passer til datasettet vårt? Går det an å finne det ut på forhånd? Svaret på det spørsmålet avhenger av datasettet ditt. Du kan lage et scatterplot av datasettet ditt, og se om du kjenner igjen en av funksjonstypene. Som regel er det likevel vanskelig å gjette seg til en funsjonstype på forhånd, og vi kan prøve oss fram med forskjellige funksjonstyper.

Programmeringen er veldig lik for lineær og ikke-lineær regresjon. Vi må bare endre den lineære funksjonen til en ikke-lineær funksjon. Da vil modellen vår tilpasse seg datasettet vårt best mulig, og vi får vite koeffisientene som bestemmer funksjonen for modellen vår. I tillegg må vi endre til riktig antall koeffisienter i programmet vårt. Se eksempelet under på hva som må endres.

```
def funksjon(x, a, b):
    return (a * x) + b
```

```
def funksjon(x, a, b, c, d):
    return (a*x**3)+(b*x**2)+(c*x)+d
```

```
koeffisienter, kovarianser = curve_fit(funksjon, x_verdier, y_verdier)
a, b = koeffisienter
print(a,b)
```

```
koeffisienter, kovarianser = curve_fit(funksjon, x_verdier, y_verdier)
a, b, c, d = koeffisienter
print(a,b,c,d)
```

```
scatter(x_verdier, y_verdier)
plot(x_verdier, funksjon(x_verdier, a, b), color="red")
title("Samlet klimagassutslipp for Norge 1990 - 2018")
xlabel("År")
ylabel("CO2-utslipp (1000 tonn CO2-ekvivalenter)")
show()
```

```
scatter(x_verdier, y_verdier)
plot(x_verdier, funksjon(x_verdier, a, b, c, d), color="red")
title("Samlet klimagassutslipp for Norge 1990 - 2018")
xlabel("År")
ylabel("CO2-utslipp (1000 tonn CO2-ekvivalenter)")
show()
```

Lag en solfanger

- mål tid og temperatur med micro:bit og lagre til fil

Oppgave

Lag en solfanger som skal varme opp et begerglass med 100 ml vann. Mål tiden og temperaturen mens solfangeren står ute i sola, ved å legge en micro:bit som lagrer data til en fil en frysepose og senk ned i begerglasset med vann. Dette datasettet med tid og temperatur skal dere bruke for å modellere temperaturendringen ved programmering.

Fase 1: Hva er en solfanger og hva kan den lages av?

Fase 2: Planlegg design og gjennomføring

1. Tenk selv først og tegn gjerne skisser fra forskjellige vinkler.
2. Forklar ideen din for de andre på gruppa. Bruk gjerne skissene i forklaringen.
3. Hele gruppa diskuterer de ulike ideene, og lager en felles plan for byggingen.

Fase 3: Gjennomfør planen deres for å lage solfangeren og lag programmet for å måle tid og temperatur og lagre på micro:biten. Se tips på side 3 og 4.

Gjennomfør fase 4 – 7.

Oppgaver

1. Bruk solas strålingstetthet for å beregne hvor stor strålingseffekt sola sender ut.
2. Tenk at denne effekten spres utover et kuleskall, og når strålingen når jorda vil dette kuleskallet ha en radius som er lik avstanden mellom jorda og sola. Finn strålingstettheten fra sola når den når jorda.
3. Bruk denne strålingstettheten for å estimere hvor mye varme som kan overføres til vannet i solfangeren deres per sekund. Finn total overført varme i løpet av den perioden dere målte temperaturendringen med micro:biten.
4. Bruk formelen $Q=mc(T-T\_0\ )$ der c er vannets spesifikke varmekapasitet for å finne ut hvor stor temperaturendringen til vannet skulle ha blitt.
5. Sammenlign svaret deres med den målte verdien for sluttemperaturen. Hvorfor er verdiene forskjellige? (Hvilke tilnærminger blir gjort i denne utregningen?)
6. Anta at det etter hvert blir strålingsbalanse mellom solinnstrålingen og varmeutstrålingen fra vannet i solfangeren. Gjør de antagelsene dere trenger for å finne vannets overflateareal og bruk dette for å finne vannets temperatur når det er strålingsbalanse.
    a. Hva skjer med denne temperaturen dersom dere fester plastfolie over åpningen på solfangeren, og puster inn mest mulig CO2?
    b. Hvordan tror dere hva solfangeren deres består av, vil påvirke denne temperaturen?
    c. Hvordan ville denne temperaturen endret seg dersom sola hadde vært mer rød/blå?

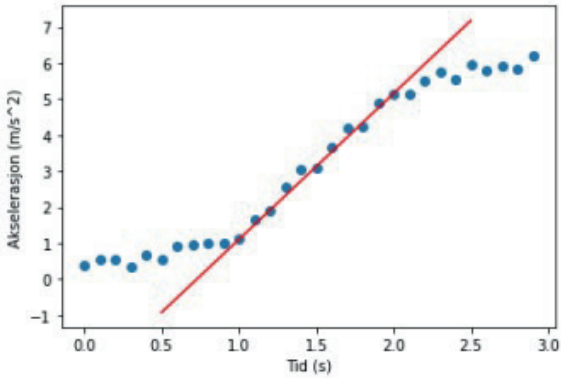
Programmeringsoppgaver

1. Lag en regresjonsmodell for temperaturen som funksjon av tiden og lag et plott. Diskuter hva som kan være gyldighetsområde for modellen.
2. Deriver modellen fra 1. og plott resultatet. Hvordan vil du beskrive temperaturendringen ifølge denne modellen?
3. Bruk de målte verdiene for tid og temperatur, og lag et program som finner momentan vekstfart numerisk. Lag en graf av temperaturendringen som en funksjon av tiden. Se tips på side 4. Ser grafen ut slik dere forventet? Når er endringen størst?
4. Sammenlign grafen fra 3. med den deriverte regresjonsmodellen. Hvilken er mest nøyaktig?
5. Ser det ut til at temperaturendringen minker med tiden? Hvordan henger dette sammen med en eventuell strålingsbalanse?

Momentan vekstfart

To former for vekstfart

Vi skiller mellom gjennomsnittlig og momentan vekstfart. Den gjennomsnittlige vekstfarten forteller oss hvor raskt noe øker mellom to x-verdier, mens den momentane vekstfarten forteller oss hvor rask økningen er i et bestemt punkt. Vi begynner med den gjennomsnittlige.



Gjennomsnittlig vekstfart

I følgende kode forutsetter vi at vi allerede har importert et datasett med x- og y-verdier. Vi starter med å plotte punktene. Deretter beregner vi den gjennomsnittlige akselerasjonen ved å bruke $a=\Delta x/\Delta y$. I vår figur var N = 30, og vi ønsker å beregne den gjennomsnittlige vekstfarten for indeksverdier mellom 10 og 20.

```
scatter(x,y)
xlabel("Tid (s)")
ylabel("Akselerasjon (m/s^2)")

aks_gj = (y[20]-y[10])/(x[20]-x[10])
```

→ Plotter x- og y-verdiene. Beregner deretter den gjennomsnittlige akselerasjonen.

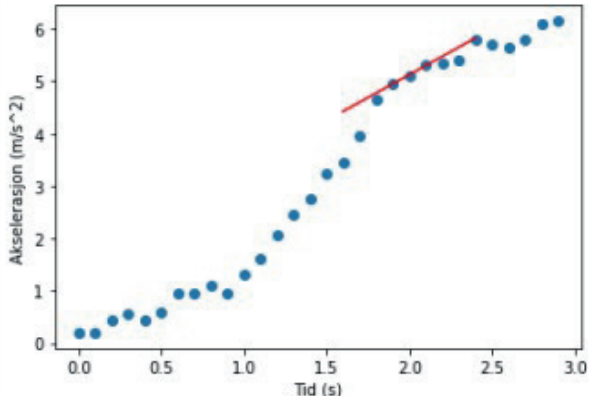
```
def f(x,a,x1,y1):
    return a*(x-x1)+y1

s = linspace(0.5,2.5)
t = f(s,aks_gj,x[10],y[10])
plot(s,t,'r')
```

→ Bruker ettpunktsformelen til å finne funksjonsuttrykket til den røde linja på figuren. Til sist plotter vi linja.

Momentan vekstfart

Hvis vi vil finne den momentane vekstfarten, gjør vi akkurat som for den gjennomsnittlige, men vi velger bare to punkter som ligger rett i nærheten av hverandre. Dermed får vi et uttrykk for hvordan funksjonsverdiene er i ferd med å endre seg akkurat der. I eksempelet nedenfor har vi valgt indeksverdier 19 og 21 for å se på veksten rundt i=20, altså x=2,0.

Plotter grafen og beregner gjennomsnittlig akselerasjon for x-verdier i nærheten av hverandre.



↓

```
scatter(x,y)
xlabel("Tid (s)")
ylabel("Akselerasjon (m/s^2)")
aks_gj = (y[21]-y[19])/(x[21]-x[19])
```

Bruker ettpunkts-formelen og plotter den røde linja. →

```
def f(x,a,x1,y1):
    return a*(x-x1)+y1
s = linspace(1.6,2.4)
t = f(s,aks_gj,x[19],y[19])
plot(s,t,'r')
```

Skrive til fil på micro:bit med Python Mu

I slutten av kapittel 9 brukte vi datasett i form av tekstfiler for å lage matematiske modeller. Hadde det ikke vært kult om vi kunne fått micro:biten til å lage slike filer av sine målinger? Da kan vi bruke micro:biten til å samle inn data over en viss periode, uten å være koblet til en datamaskin. Datasettet blir lagret i micro:biten som en .txt-fil, og vi kan legge den over i datamaskinen, før vi plotter resultatene eller lager en matematisk modell med Python.

Det aller første vi må gjøre, er å opprette en tekstfil, det vil i en fil som slutter på .txt, og det er enklest å gjøre i Mu. Lagre den samme i samme mappe som resten av dine Python Mu-filer.

Deretter må den fila du har laget, kopieres over på micro:biten. Dette gjør du på samme måten som for sonaren.

```
with open ('filnavn.txt', 'w') as fil:
```

Åpner fila som heter filnavn.txt og tildeler den til et objekt som vi kaller fil. Siden vi har 'w' (write) i kommandoen gjøres det klar til å skrive inn i fila.

```
with open ('filnavn.txt', 'r') as fil:
```

Tilsvarende som linja over, men siden vi har 'r' (read) i kommandoen gjøres det klar til å lese fila. Det er valgfritt å ta med 'r'.

```
fil.write(y+'\n')
```

Skriver variabelen y til fila vår. +'n' gjør at det blir et linjeskift etter hvert tall, og det må vi ha for at python skal kunne lese filene.
Må stå med innrykk.

```
y = str(x)
```

Gjør om variabelen x til en streng (tekstverdi). Må stå med innrykk.

```
z = fil.read()
```

Leser det som står i fila vår, og legger det inn i variabel z. Må stå med innrykk.

```
print(z)
```

Skriver variabelen z til skjermen på PC. Må IKKE stå med innrykk.

Da gjenstår det å lage selve programmet. Under er et eksempel der micro:biten måler temperaturen og legger målingen i en .txt-fil ved navn datasett.txt. Det kan være greit å kunne sjekke hvordan datasett.txt-fila ser ut uten å overføre til datamaskinen og åpne den, derfor er dette også med i puslespill-oppgaven.

Puslespill-oppgave

```
print(innhold)
```

```
t = temperature()
```

```
while not button_a.is_pressed():
```

```
sleep(1000)
```

```
fil.write(str(t)+'\n')
```

```
with open('datasett.txt') as fil:
```

```
with open('datasett.txt', 'w') as fil:
```

```
innhold = fil.read()
```

```
from microbit import *
```

Ekstraoppgave

Lag et program som bruker en sensor du kobler på micro:biten for å samle inn data som du lagrer i en fil på micro:biten. Hva må du forandre på?

Hva må du forandre i programmet for å lage en fil med to kolonner der den første kolonnen er hvor lang tid det har gått før målingen er gjort?