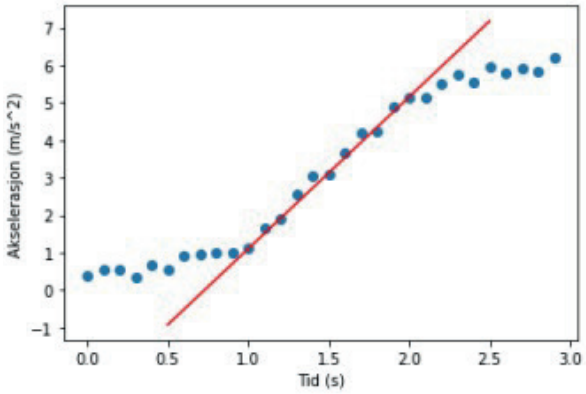
Opplegg 10 – **Momentan vekstfart**

To former for vekstfart

Vi skil mellom gjennomsnittleg og momentan vekstfart. Den gjennomsnittlege vekstfarten fortel oss kor raskt noko aukar mellom to x-verdiar, medan den momentane vekstfarten fortel oss kor rask aukinga er i eit visst punkt. Vi byrjar med den gjennomsnittlege.

Gjennomsnittleg vekstfart

I følgande kode går vi ut frå at vi alt har importert eit datasett med x- og y-verdiar. Vi startar med å plotte punkta. Deretter bereknar vi den gjennomsnittlege akselerasjonen med å bruke $a=\Delta x/\Delta y$. I figuren vår var N = 30, og vi ønsker å berekne den gjennomsnittlege vekstfarten for indeksverdiar 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. Berekner deretter den gjennomsnittlege akselerasjonen.

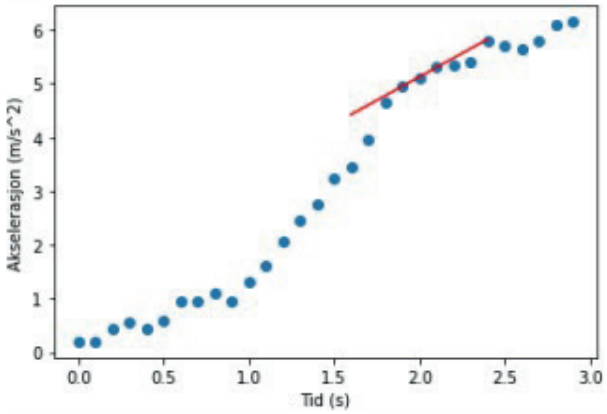
```
def f(x,a,x1,y1):
    return a*(x-x1)+y1

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

→ ruker eittpunktsformelen til å finne funksjonsuttrykket til den raude lina på figuren. Til sist plotter vi lina.

Momentan vekstfart

Dersom vi vil finne den momentane vekstfarten, gjer vi akkurat som for den gjennomsnittlege, men vi vel berre to punkt som ligg rett i nærleiken av kvarandre. Dermed får vi eit uttrykk for korleis funksjonsverdiane er i ferd med å endre seg akkurat der. I dømet nedanfor har vi valt indeksverdiar 19 og 21 for å sjå på veksten rundt i=20, altså x=2,0.



Plottar grafen og bereknar gjennomsnittleg akselerasjon for x-verdiar i nærleiken av kvarandre.

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

Bruker eittpunkts-formelen og plottar den raude 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')
```

Lag en utskytningsmekanisme

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

Oppgåve

Lag ein utskytingsmekanisme som skyt ein micro:bit med støytdemping rundt slik at «micro:bit-kula» får størst mogleg rykk (akselerasjonsendring). Micro:biten skal måle akselerasjonen, og desse dataene skal de bruke til å berekne den største momentane vekstfarten for akselerasjonen (rykket) micro:biten hadde på flygeturen.

Fase 1: Gjennomfør informasjonsinnhenting for å få idéar. Kva slags utskytningsmekanismer finnest?

Fase 2: Idémyldre og planlegge
Det er viktig at de er opne for alle slags idear og ikkje er for kritiske, da kan nyttige framlegg bli kutta ut for tidleg.

1. Tenk sjølv først og teikn gjerne skisser.
2. Forklar ideen din for dei andre på gruppa.
3. Heile gruppa diskuterer dei ulike ideane, og lager ein felles plan for bygging.

Fase 3: Gjennomføring

- Bygg utskytningsmekanismen
- Lag programmet for å måle akselerasjonen med micro:biten og pakk micro:biten inn i støytdempande materiale.

Fase 4: Test om utskytningsmekanismen og programmene verkar som planlagt.

Fase 5: Samanlikne resultata med andre i klassen.

- Fekk nokon andre større akselerasjon?
- kvifor det?
- Kan de bruke noko av det same på dykkar utskytingsmekanisme for å få han meir effektiv?

Fase 6: Gå attende til dei andre fasane for å gjere dei planlagte forbetringane.

Fase 7: Gjennomfør dei siste målingane, og dokumenter prosjektet på ein valfri måte

---

Programmeringsoppgåver

1. Lag ein regresjonsmodell for akselerasjonen som funksjon av tida og lag eit plott. Sjå tips på side 4 og 5. Diskuter kva som kan vere gyldigheitsområde for modellen.
2. Deriver modellen frå 1. og plott resultatet. Korleis vil du beskrive akselerasjonsendringa ifølge denne modellen?
3. Bruk dei målte verdiane for tid og akselerasjon, og lag eit program som finn momentan vekstfart numerisk. Lag ein graf av akselerasjonsendringa som ein funksjon av tida. Sjå tips på side 3. Ser grafen ut slik de forventa? Når er endringa størst?
4. Samanlikne grafen frå 3. med den deriverte regresjonsmodellen. Kva for ein er mest nøyaktig?
5. Ser det ut til at akselerasjonsendringen minkar med tida? Korleis heng dette saman med utskytingsmekanismen?

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 modellar. Hadde det ikkje vore kult om vi kunne fått micro:biten til å lage slike filer av målingane sine? Da kan vi bruke micro:biten til å samle inn data over ein viss periode, utan å vere kopla til ein datamaskin. Datasettet blir lagra i micro:biten som ei .txt-fil, og vi kan legge henne over i datamaskinen, før vi plottar resultata eller lager ein matematisk modell med Python.

Det aller første vi må gjere, er å opprette ei tekstfil, det vil seie ei fil som sluttar på .txt, og det er enklast å gjere i Mu. Lagre fila i same mappe som resten av Python Mu-filene dine.

Deretter må den fila du har laga, kopierast over på micro:biten. Dette gjer du på same måten som for sonaren.

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

Opnar fila som heiter filnavn.txt og tildeler den til eit objekt som vi kallar fil. Sidan vi har 'w' (write) i kommandoen gjerest det klar til å skrive inn i fila.

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

Tilsvarande som lina over, men sidan vi har 'r' (read) i kommandoen gjerest det klar til å lese fila. Det er valfritt å ta med 'r'.

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

Skriv variabelen y til fila vår. +'n' gjer at det blir eit lineskift etter kvart tal, og det må vi ha for at python skal kunne lese filene. Må stå med innrykk.

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

Gjer om variabelen x til ein streng (tekstverdi). Må stå med innrykk.

```
print(z)
```

Skriv variabelen z til skjermen på PC. Må IKKJE stå med innrykk.

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

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

Da gjenstår det å lage sjølve programmet. Under er eit døme der micro:biten måler temperaturen og legg målinga i ein .txt-fil med namn datasett.txt. Det kan vere greit å kunne sjekke korleis datasett.txt-fila ser ut utan å overføre til datamaskinen og opne den, difor er dette og med i puslespel-oppgåva.

Puslespill-oppgåve

```
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 *
```

Ekstraoppgåve

Lag eit program som bruker ein sensor du koplar på micro:biten for å samle inn data som du lagrar i ei fil på micro:biten. Kva må du gjere annleis?

Kva må du forandre i programmet for å lage ei fil med to kolonnar der den første kolonnen er kor lang tid det har gått før målinga er gjort?

Av: Ellen Rekaa

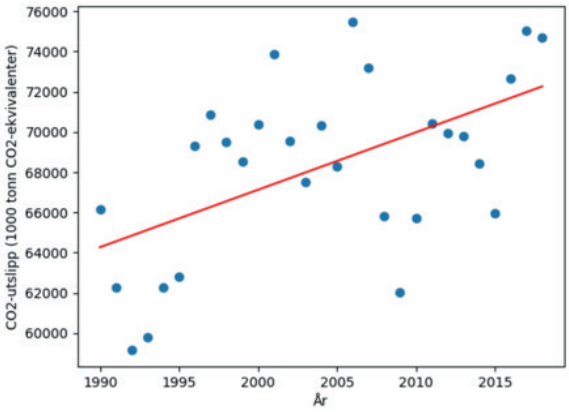
Lineær regresjon

Lineær regresjon med Python

De har alt utført mange lineære regresjonar med Geogebra, så kva er poenget med å nytte Python i staden?

I Geogebra må vi skrive inn alle målingane våre for hand, eller kopiere frå t.d. Excel, og da kan det vere vanskeleg å gjere ein regresjon dersom vi har mykje data. Tenk deg at du har 100 målingar, da tek det veldig lang tid å skrive inn alle målingane i reknearket i Geogebra. Det kan fort bli ein feil eller to undervegs og. Tenk om du har endå fleire målingar, kanskje fleire enn 1000! Da blir det ganske keisamt å skrive inn alle tala.

Samlet klimagassutslipp for Norge 1990 – 2018



I Python kan du bruke data som ligg i ferdige filer. Du slepp å skrive dei inn! Det som er viktig, er at filene er rett type. Slik som med plotting av grafar i Python.

Ein annan føremon med å bruke Python, er at det er eit programmeringsspråk som blir brukt av forskarar som jobbar med modellering. Andre forskarar kan bruke andre programmeringsspråk som er spesialtilpassa akkurat den modelleringa dei jobbar med, men dei følger dei same prinsippa som Python. Så dersom du kan modellere med Python, er det enklare å modellere i eit anna programmeringsspråk enn om du berre har brukt Geogebra tidlegare.

Regresjon i Python er samansett av seks deler:

1. Importere dei biblioteka som trengs

2. Importere data frå ei .txt-fil

3. Legge dataene i en tabell for x-verdiene og en for y-verdiene

4. Lage ein programmeringsfunksjon som seier at vi skal bruke ein lineær modell i regresjonen

5. Utføre regresjonen, og vise på skjermen kva den matematiske modellen blir, stigningstalet (a) og skjæringspunktet med y-aksen (b)

6. Teikne grafen med resultatet – tilpass koden for plotting av graf
   - Plotte punkta med eit scatterplot
   - Plotte regresjonsgrafen – korleis få programmet til å rekne ut y-verdiane for modellen?

Puslespeloppgåve
Sorter kodelinene etter nummera i tekstboksen over, om regresjon i Python. Da vil du få den rette 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]
```

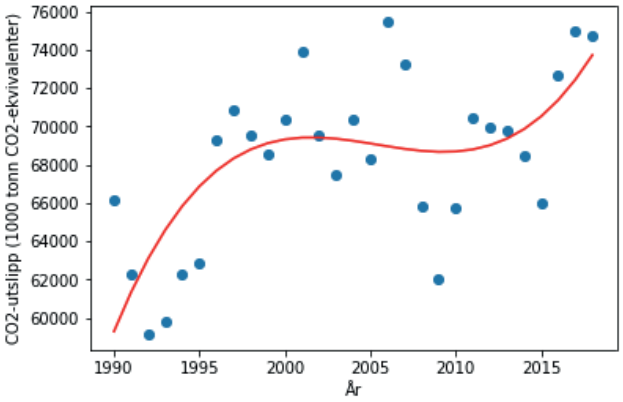
6
```
def funksjon(x, a, b):
    return (a * x) + b
```

Ikkje-lineær regresjon med Python

Skilnad på lineær og ikkje-lineær regresjon i Python

Det er liten skilnad på korleis programma for lineær og ikkje-lineær regresjon ser ut. Den viktigaste skilnaden er å endre på kva for ein funksjonstype datasettet skal tilpassast. Det står eit oversyn over nokre funksjonstypar, og korleis ein skriv dei i python, på dei to neste sidene.

Samla klimagassutslepp for Noreg 1990 – 2018



Men korleis skal vi vite kva for ein funksjonstype som passar til datasettet vårt? Går det an å finne det ut på førehand? Svaret på det spørsmålet avheng av datasettet ditt. Du kan lage eit scatterplot av datasettet ditt, og sjå om du kjenner att ein av funksjonstypane. Som regel er det likevel vanskeleg å gjette seg til ein funksjonstype på førehand, og vi kan prøve oss fram med ulike funksjonstypar.

Programmeringa er veldig lik for lineær og ikkje-lineær regresjon. Vi må berre endre den lineære funksjonen til ein ikkje-lineær funksjon. Da vil modellen vår tilpasse seg datasettet vårt best mogleg, og vi får vite koeffisientane som avgjer funksjonen for modellen vår. I tillegg må vi endre til rett tal på koeffisientar i programmet vårt. Sjå dømet under på kva som må endrast.

```
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()
```