Pythonprogrammering med Tekna

Befolkningsvekst og pandemier

kodeskolen simula

kodeskolen@simula.no

I dette kompendiet skal vi bruke rekursive tallfølger til å simulere befolkningsvekst. Vi skal se på bærekraft og hva som skjer når vi har både rovdyr og byttedyr i samme biologiske system. Til slutt vil vi se hvordan den samme teknikken vi brukte til å simulere befolkningsvekst kan brukes for pandemisimuleringer, det har seg nemlig slik at den vanligste matematiske modellen for pandemier, *SIR-modellen* kan simuleres på liknende måter som befolkninger av dyr.

Innhold

1 Populasjonsdynamikk, eksponensiell vekst og pandemier

I biologi er man ofte interessert i hvordan mengden dyr (eller mikroorganismer) forandrer seg over tid. Såkalt populasjonsdynamikk. For eksempel, hvis vi har en petri-skål med bakterier i, så vil antallet bakterier i morgen øke proposjonalt med hvor mange bakterier som er i skåla i dag. Dette er et eksempel på eksponensiell vekst, som kommer igjen mange ganger når man studerer populasjonsdynamikk. Denne eksponensielle veksten er bare et av de matematiske fenomenene vi ser når vi studerer populasjonsdynamikk, men det er langt fra det eneste.

I dette prosjektet skal vi se hvordan en matematiker kunne gått frem for å beskrevet populasjonsvekst, men først la oss se et eksempel som illustrerer hvorfor det er viktig at biologer og matematikere arbeider sammen for å løse slike problem!

1.1 Matematikk for biologi og biologi for matematikk

På starten av 1900-tallet var den italienske biologen, Umberto D'Ancona, interessert i å vite hvordan mengden fisk endrer seg fra år til år. Det er jo tross alt nyttig å vite for å forhindre overfiske! Dessverre var det ingen åpenbar måte måle hvor mye fisk som var i havet på starten av 1900-tallet. Derfor bestemte han seg for å måle andelen fisk og andelen hai som ble fanget av fiskebåter ved ulike havner i Italia. Her er en figur som viser andelen hai som ble fanget ved havna i Fiume:

Det samme mønsteret fant D'Ancona ved flere andre havner i Italia — det var mange flere haier enn fisk fanget i 1918 og 1919 enn andre år. Hvordan kan dette ha seg?

Vi vet at 1918 og 1919 var på høyden av første verdenskrig, og da var det mye mindre fiske enn normalt. Det er fort gjort å tenke at det vil påvirke mengden fisk og hai, men ikke forholdet mellom dem. Som vi så i grafen over, så var det altså ikke sånn. Etter å ha sittet lenge med denne nøtta uten å finne en overbevisende forklaring, pratet D'Ancona med svigerfaren sin, Vito Volterra, som var en annerkjent italiensk matematiker. Sammen snakket de om problemet og kom frem til verdens mest kjente modell for rovdyr-pattedyr dynamikk, den såkalte Lotka-Volterra modellen. Denne modellen har blitt studert utrolig mye av matematikere i etterkant, og vi skal se mer på den mot slutten av dette prosjektet.

PS: Hvis du lurer på hvorfor modellen heter Lotka-Volterra istedenfor D'Ancona-Volterra modellen så kommer det av at den amerikanske matematikeren Alfred J. Lotka kom på samme modell et tiår tidligere, men da var det for å beskrive kjemiske reaksjoner, ikke biologiske populasjoner. Vi kan faktisk bruke de samme likningene for å simulere kjemiske reaksjoner og biologiske populasjoner!

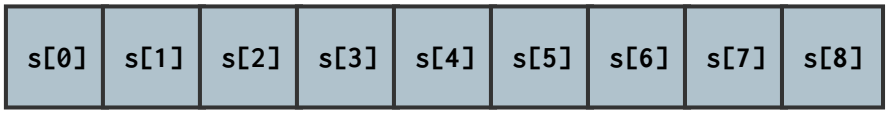
1.2 Kilder

- Datagrunnlaget vi brukte for å lage linjeplott-figuren over kan du finne det i en italienske artikkel av D'Ancona og Volterra, Leçons sur la théorie mathématique de la lutte pour la vie, som ligger tilgjengelig her: https://gallica.bnf.fr/ark:/12148/bpt6k29088s/
- Historien er gjenfortalt basert to verk:
    1. Den originale forskningsartikkelen til Vito Volterra: Variations and Fluctuations of the Number of Individuals in Animal Species living together, som ligger tilgjengelig her: https://academic.oup.com/icesjms/article/3/1/3/737415
    2. Læreboken: Differential Equation Models skrevet av Martin Braun, Courtney S. Coleman og Donald A. Drew og som ble utgitt i 1983 (kan kjøpes som e-bok her: https://link.springer.com/book/10.1007/978-1-4612-5427-0)

2 Arrayer og arrayindeksering

I dette opplegget skal vi altså simulere biologiske system ved hjelp av rekursive rekker. Med slike simuleringer kan vi se på utviklingen av en dyrebefolkning over tid. Den enkleste måten å se resultatet fra simuleringen er bare å printe ut befolkningstallet for hvert år i simuleringen, men det er kanskje ikke den beste måten. Isteden ønsker vi å plotte resultatet fra simuleringen. For å få til det, må vi regne ut befolkningstallet for hvert år, og lagre det tallet i en *array*.

Det har seg nemlig slik at om vi har en array, `x`, så kan vi hente ut element fra denne arrayen med klammeparanteser, det første elementet i x-arrayen vår kan vi få tak i ved å skrive `x[0]`, det andre kan vi få ved å skrive `x[1]`, osv. Vi teller altså fra en her og. Under har vi en figur som viser hvordan dette virker for en array med ni element i seg.



La oss nå se på et lite eksempel:

```
1 x = arange(5)  # Lag en array med heltallene fra og med 0
     til men ikke med 5
2 print(x)
3 print(x[0])
```

```
[0 1 2 3 4]
0
```

Vi kan og endre verdien til et element i arrayet, hvis vi vil endre verdien til det første elementet i arrayen vår til `9`, kan vi skrive `x[0] =9`. La oss se på det.

```
1 x[0] = 9
2
3 print(x)
```

```
[9 1 2 3 4]
```

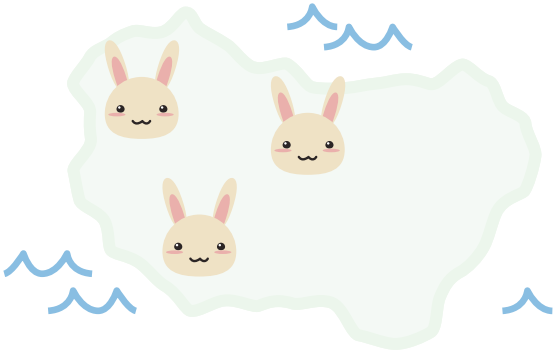
Noe vi kommer til å trenge for dette prosjektet er altså "tomme" arrayer som vi kan lagre simuleringsresultatene våre i. Den vanligste måten å lage slike "tomme" arrayer er ved å bruke `zeros`-funksjonen i `pylab`:

```
from pylab import zeros

x = zeros(5)
print(x)
```

```
[0 0 0 0 0]
```

Denne funksjonaliteten er veldig nyttig. Før simuleringen sier vi hvor mange år vi vil simulere, og lager en tom array med et element for hvert år. Deretter kan vi bare putte inn befolkningen for for hvert år i simuleringen, som gjør at vi kan plotte og se på befolkningsveksten etterpå!

3 Eksponensiell vekst av populasjoner



Nå skal vi simulere hvordan dyrepopulasjoner utvikler seg. La oss se for oss at vi har en øy med 200 kaniner på, også skal vi simulere hvordan kaninbefolkningen endrer seg. Det første, enkleste eksempelet vil jo være at det er en konstant økning av kaniner hvert år, f.eks. kan det bli født 100 nye kaniner hvert år. Matematisk, så kan vi skrive den modellen slik:

$$k_t = k_{t-1} + 200, \tag{1}$$

hvor k_t er antall kaniner etter t år. Denne lineære veksten er veldig urealistisk. Vekstraten til dyrene er jo avhengig av hvor mange dyr vi har på øya vår. Hvis vi ikke har noen dyr på øya, så kan vi vel ikke få 200 kaniner neste år!? Hvis vi har to kaniner, så vil vekstraten være mindre enn hvis vi har 4 kaniner, for det er jo bare halvparten så mange som kan få nye kaninunger. Det er derfor naturlig å tenke at vekstraten til kaniner er proposjonal med antall kaniner vi har på øya, altså blir den slik:

$$k_t = k_{t-1} + fk_{t-1}, \tag{2}$$

hvor k_t er antall kaniner i år t, k_{t-1} er antall kaniner året før og f er fødselsraten. Om vi bruker denne modellen for å simulere systemet vårt så vil jo ingen kaniner dø. Igjen så vet vi at hvis vi har 100 kaniner så kan dobbelt så mange dø som om vi har 50 kaniner, og det er derfor logisk å tenke at dødsraten er proposjonal med antall kaniner på øya fra før. Matematisk blir det

$$k_t = k_{t-1} + fk_{t-1} - dk_{t-1}, \tag{3}$$

hvor d er dødsraten til kaninene. Dette ser vi at vi kan skrive om til å bli

$$k_t = k_{t-1} + rk_{t-1}, \tag{4}$$

hvor $r = (f - d)$ er vekstraten til kaninene våre. Hvis vi bruker programmering, så får vi:

```
k[t] = k[t-1] + r*k[t-1]
```

La oss simulere systemet med en dødsrate, $d = 0.1$ og fødselsrate, $f = 0.2$:

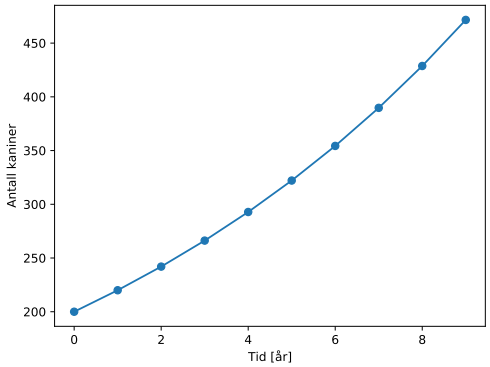
```
from pylab import zeros

antall_år = 10

dødsrate = 0.1
fødselsrate = 0.2
vekstrate = fødselsrate - dødsrate

k = zeros(antall_år)
k[0] = 200

for t in range(1,antall_år):
    k[t] = k[t-1] + vekstrate*k[t-1]

print(k)
```

```
[200.         220.         242.         266.2        292.82
        322.102    354.3122    389.74342   428.717762
   471.5895382]
```

Med dette programmet har vi simulert kaninpopulasjonen på en øy, men hvordan har det egentlig gått på øya? Vi printet ut resultatet fra simuleringen, men det er vanskelig å se hvordan populasjonsveksten går bare basert på denne tallrekka. Vi kan jo ikke se resultatet! La oss isteden bruke pylab for å plotte denne kaninbestanden.

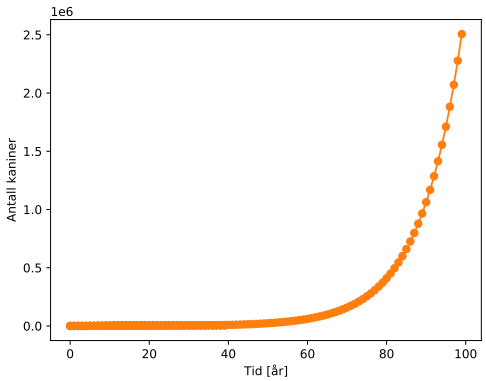
```
from pylab import plot, show, xlabel, ylabel

plot(range(antall_år), k, '-o')
xlabel('Tid [år]')
```

```
ylabel('Antall kaniner')
show()
```

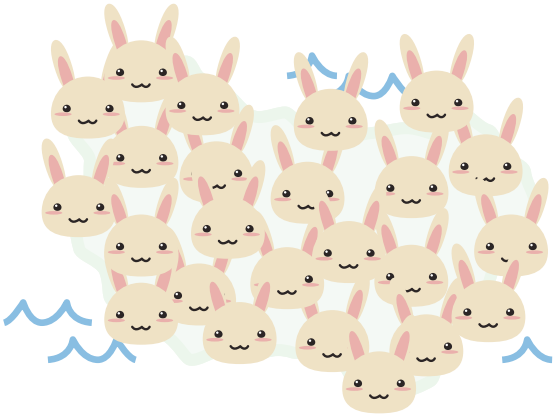


Dette ser jo bra ut! kaninbestanden øker, og det er jo bare å forvente, det fødes jo tross alt flere kaniner enn det dør. La oss derfor simulere dyrepopulasjonen over flere år på øya

```
antall_år = 100

dødsrate = 0.1
fødselsrate = 0.2
vekstrate = fødselsrate - dødsrate

k = zeros(antall_år)
k[0] = 200

for t in range(1, antall_år):
    k[t] = k[t-1] + vekstrate*k[t-1]


plot(range(antall_år), k, '-o')
xlabel('Tid [år]')
ylabel('Antall kaniner')
show()
```



Oisann, i følge simuleringa får vi ca 2.5 millioner dyr på øya. Med denne modellen vil befolkningen bare vokse helt uhemmet, og det har jo ikke øya ikke plass til!



4 Bærekratig vekst

Den matematiske modellen vi har lagd så langt har ikke noen form for bærekraftig vekst. Øya vår har jo ikke mat, eller plass, til uendelig med kaniner. Derfor ønsker vi en modell hvor vekstraten endrer seg basert på hvor mange kaniner som er på øya fra før. Vi ønsker altså en modell på formen:

$$k_t = k_{t-1} + r(k_{t-1})k_{t-1}, \tag{5}$$

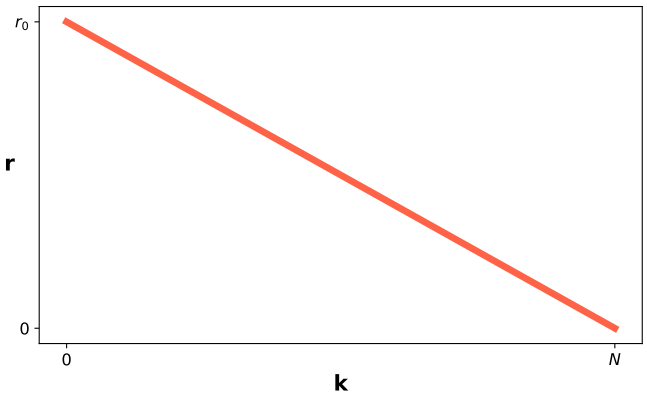
som med kode blir dette:

```
k[t] = k[t-1] + vekstrate(k[t-1])*k[t-1]
```

Det naturlige spørsmålet vårt nå er hvilken form vi tror at vekstraten, $r(k)$, skal ha. La oss starte med noen observasjoner.

- Øya har en bæreevne, N, som er antallet kaniner øya har plass og mat til. Hvis $k_t \approx 0$ så ønsker vi eksponensiell vekst for da er bæreevnen mye høyere enn antallet dyr.
- Altså er den ubegrensede vekstraten, $r(0) = r_0 > 0$. Hvis $k_t \approx N$ så ønsker vi at det skal være ca like mange dyr på øya hvert år.
- Altså er $r(N) = 0$.

Den letteste måten å få en funksjon på denne formen er å tenke seg en rett linje som går igjennom punktene $(0, r_0)$ og $(K, 0)$, slik vi ser i figuren under:



Hvis vi bruker denne lineære funksjonen for den relative vekstraten, får vi dette uttrykket for r:

$$r(k) = r_0 \left(1 - \frac{k}{N}\right), \tag{6}$$

hvor $r(k)$ altså er kaninvekstraten på en øy med en bæreevne på N kaniner mens det er k kaniner på øya.

4.1 Oppgaver

Oppgave 1 *Vekstratefunksjon*

a) Lag en funksjon, `vekstrate(antall_kaniner)` som tar inn antall kaniner og returnerer vekstraten for en øy med en bæreevne på 5000 kaniner og en ubegrenset vekstrate på 0.5.

Nå har vi en funksjon som regner ut vekstraten, men hvordan vet vi at den funker? Vi vet at dersom det ikke er noen kaniner på øya, så skal vekstraten være lik den ubegrensede vekstraten, altså $r(0) = 0.5$. Vi vet også at dersom det er 5000 kaniner på øya, så skal vekstraten være 0.

b) Sjekk om din `vekstrate` funksjon returnerer `0.5` dersom `antall_kaniner=0` og at den returnerer `0` dersom `antall_kaniner=5000`.

Slik testing du akkurat gjorde er veldig lurt. Det er lett å skrive feil når man koder, og det kan være vanskelig å finne feilen i et stort program! Tenk hvis vi hadde skrevet `1 + antall_kaniner/5000` istedenfor `1 - antall_kaniner/5000`? Det er fort gjort, men hvis vi tester koden underveis kan vi enkelt finne mange av disse feilene.

Hint: I Python ser syntaksen til funksjoner slik ut:

```
def funksjonsnavn (inputvariabel):
    # [instruksjonene funksjonen skal utføre]
    return #[det som skal returneres]
```

Oppgave 2 *Simulere bærekraftig vekst*

a) Nå har vi kode for eksponensiell vekst (se side 9). Modifiser den koden slik at du simulerer en øy med bærekraftig vekst. Den ubegrensede vekstraten skal være 0.5, bæreevnen skal være 5000 kaniner og simuleringen skal vare i 40 år og du skal starte med 200 kaniner på øya.

b) Bruk plottefunksjonaliteten i Pylab for å plotte resultatet fra simuleringen. Ser det mer realistisk ut enn den bærekraftige veksten? Stabiliserer populasjonen seg? i såfall, hva stabiliserer den seg til?

Oppgave 3 *Å utforske bærekraftig vekst*

a) Modifiser bæreevnen til systemet. Hvordan oppfører populasjonen til en øy med en bæreevne på 10 000 kaniner?

Nå skal vi utforske hva som skjer dersom det oppstår en naturkatastrofe på øya. For å gjøre det må vi kunne sette bæreevnen til øya manuelt inne i simuleringsløkka vår.

b) Lag en funksjon, `vekstrate(antall_kaniner, ubegrenset_vekstrate, bæreevne)`, som tar inn antall kaniner på øya, den ubegrensede vekstrate og øyas bæreevne og returnerer vekstraten for en øy med det gitte antallet kaniner, ubegrensede vekstrate og bæreevne.

c) Modifiser koden du skrev i Oppgave 2 slik at den bruker den nye vekstratefunksjonen din. Sett den ubegrensede vekstraten lik 0.5 og bæreevnen lik 5000 kaniner.

Med dette har vi alle puslespillbrikkene vi trenger for å simulere en naturkatastrofe på øya! Det kan vi gjøre ved å endre bæreevnen på øya etter et gitt antall år.

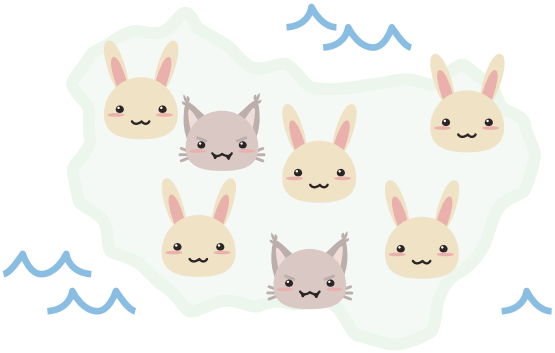
d) Bruk en betingelse (`if`) for å sjekke om det har gått mer enn 20 år siden starten av simuleringen (`t > 20`). Hvis det har gått mer enn 20 år, skal bæreevnen være lik 2000, hvis ikke (`else`) skal bæreevnen være lik 5000.

Hint: Syntaksen til betingelser ser slik ut:

```
x = 5
if x < 3:
    print("x er mindre enn 3")
else:
    print("x er ikke mindre enn 3")
```

5 Rovdyr-byttedyr dynamikk

La oss nå tenke oss at vi, istedenfor å introdusere bare kaniner til øya vår, introduserer både kaniner og gauper. Før vi kan beskrive et slikt system matematisk så bør vi tenke hvordan det gir mening at det biologiske systemet utvikler seg. Så, når vi har kommet frem til det, kan vi tenke på hvordan vi kan beskrive et slikt system matematisk.



Hvis vi skal tenke oss hvordan øya er, kan vi komme frem til disse fire reglene:

- Hvis det ikke er noen gauper tilstede, så utvikler kaninpopulasjonen seg som om det bærekraftig vekst.
- Jo flere gauper jo høyere blir dødsraten til kaninene.
- Bæreevnen til gaupene er avhengig av antallet kaniner.
- Gaupene dør også av naturlige årsaker som ikke er relatert til mengden kaniner som er tilgjengelig. La oss nå sette opp likninger som tar hensyn til alle disse reglene.

5.1 Modellere kaniner

Vi har allerede modellert kaninbestanden med en bærekraftig vekst-likning:

$$k_{t+1} = r(k_t)k_t. \tag{7}$$

Kaniner som blir jaktet:

Når kaninene blir jaktet på trenger vi vite hvor mange kaniner gaupene spiser. Hvis hver gaupe har en apetitt, altså et visst antall kaniner de vil spise, vil vi få en proposjonalitetssammenheng mellom antall gauper og dødsraten. Vi kan bruke variabelen, a_u for å beskrive gaupeappetitten, og få denne likningen for kaninbestanden:

$$k_{t+1} = r(k_t)k_t - (a_u g_t)k_t, \tag{8}$$

hvor g_t er antallet gauper i år t. Den relative vekstraten for kaniner:

Siden kaniner blir jaktet på vet vi empirisk at kaninbestanden ikke vil nærme seg bæreevnen til øya — de blir spist av gaupene lenge før det. Dermed trenger vi ikke bruke den kompliserte formelen for relativ vekstrate fra forrige avsnitt. Istedenfor kan vi modellere fødselsraten til kaninene slik vi gjorde i første avsnitt, og dermed få likninger som er litt lettere å håndtere. Hvis vi gjør det får vi denne likningen for harebestanden:

$$k_{t+1} = k_t + f_k k_t - (a_g g_t)k_t. \tag{9}$$

5.2 Modellere gauper

Bæreevnen til gaupepopulasjonen er gitt ved antallet kaniner på øya. Igjen, hvis hver gaupe må spise 500 kaniner i året for å overleve, så er bæreevnen til Gaupene proposjonal med antallet kaniner. Dermed, hvis vi antar at antallet gaupeunger som overlever er proposjonalt med antall gauper og antall kaniner får vi denne likningen for gaupepopulasjonen:

$$g_{t+1} = g_t + (f_g k_t)g_t \tag{10}$$

Gauper dør også:

Siden det ikke er noen som jakter på gauper så kan vi anta at de dør med en konstant frekvens. Altså, hvor mange gauper som dør i løpet av et år er proposjonalt med hvor mange gauper som er på øya. Hvis vi kombinerer det med likningen over

får vi:

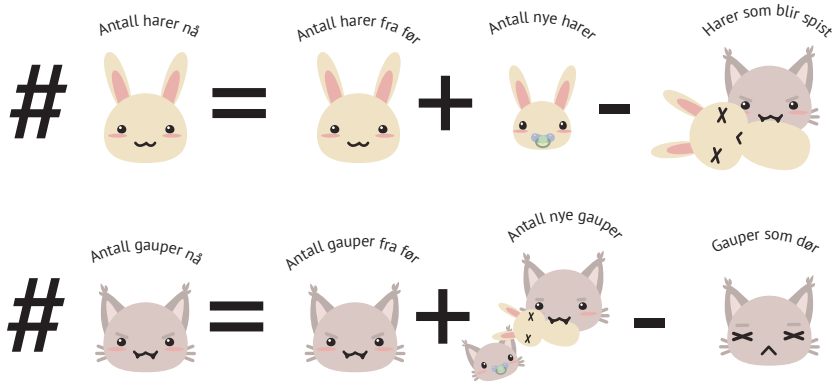
$$g_{t+1} = g_t + (f_g k_t) g_t - d_g g_t. \tag{11}$$

Sammendrag av modellen:

Bestanden kaniner og gauper kan vi altså beskrive med disse to likningene:

$$k_{t+1} = k_t + f_k k_t - (a_g g_t) k_t, \tag{12}$$
$$g_{t+1} = g_t + (f_g k_t) g_t - d_g g_t. \tag{13}$$

Her ble det mange konstanter gitt! Jeg liker å visualisere disse likningene slik:



Antallet kaniner neste år er gitt ved antallet kaniner i år, pluss antallet nye kaniner, minus antallet kaniner som blir spist av gaupene. Tilsvarende er antallet gauper neste år gitt ved antallet gauper dette året, pluss antallet nye gauper som det er mat til, minus antallet gauper som dør. Hvis vi vil programmere det, kan vi skrive det slik:

```
k[t+1] = k[t] + f_k * k[t] - a_g * k[t] * g[t]
g[t+1] = g[t] + f_g * k[t] * g[t] - d_g * g[t]
```

5.3 Sammendrag over størrelsene

Her ble det mange variabler, så la oss se grundigere på dem

- k_t - Antall kaniner i år t.
- g_t - Antall gauper i år t.
- f_k - Den relative fødselsraten til kaniner.
- f_g - Den relative fødselsraten til gauper.
- a_g - Appetitten til gauper.
- d_g - Den relative dødsraten til gauper.

Vi har altså et system hvor antallet kaniner kan vokse fritt om vi ikke har noen gauper til stede. Gaupene jakter på kaninene så hvis det er mange gauper tilstede så dør mange kaniner. Problemer med denne modellen:

Det er dessverre et problem med denne modellen, og det er at den ser på år for år. Dessverre er det slik at utviklingen år for år er mer komplisert enn det vi har skrevet over. Men, fortvil ikke! For alt vi har sagt gir jo mening, men sammenhengene gjelder ikke for år, men for kortere tidsintervall, slik som dager. Det eneste vi trenger å gjøre for at modellen skal gjelde dag-for-dag istedenfor år-for-år er å dele konstantene våre på antall dager det er i et år.

5.4 Valg av størrelser

I denne simuleringen kommer vi til å bruke parametre som ikke helt gir mening hvis vi har et system med kun gauper kaniner. Grunnen til det er at hvis vi har fornuftige"verdier for parameterne vil vi få så mange flere kaniner enn gauper at vi vil ikke kunne se noen endring i gaupebestanden over tid. Vi setter disse verdiene for parameterne:

- $k_0 = 10.0$ - Antall kaniner det første året
- $g_0 = 2.00$ - Antall gauper det første året
- $f_k = 5.00$ - Den relative fødselsraten til kaniner per år.
- $f_g = 0.15$ - Den relative fødselsraten til gauper per år.
- $a_g = 2.50$ - Appetitten til gauper per år.
- $d_g = 0.50$ - Den relative dødsraten til gauper per år.

5.5 Simulere systemet

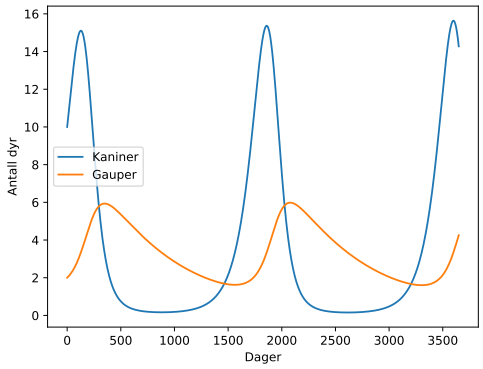
Under har vi koden som kreves for å simulere dette rovdyr-byttedyr-systemet.

```
from pylab import zeros

dager_i_år = 365
antall_år = 10
antall_dager = antall_år*dager_i_år

f_k = 5/dager_i_år
f_g = 0.15/dager_i_år
a_g = 1.5/dager_i_år
d_g = 0.5/dager_i_år

k = zeros(antall_dager)
k[0] = 10 # start mengde kaniner
g = zeros(antall_dager)
g[0] = 2

for t in range(antall_dager - 1):
    k[t+1] = k[t] + f_k*k[t] - a_g*g[t]*k[t]
    g[t+1] = g[t] + f_g*g[t]*k[t] -  d_g*g[t]
```

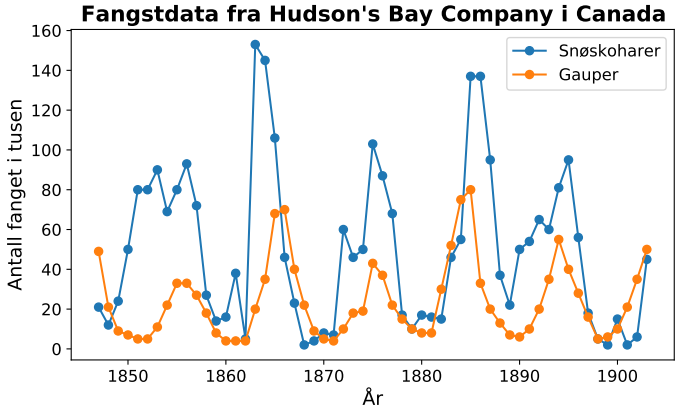
Så koden over kan brukes for å simulere kanin- og gaupebestanden på øya vår. Men, det er en forskjell mellom denne simuleringskoden, og koden for å simulere kaniner alene. Her simulerer vi dag-fordag, ikke år-for-år. Hvorfor gjør vi det slik? Vel, likningene våre beskriver hvordan dyrebestanden utvikler seg, men hvis vi bare ser på årlige målinger, så vil vi miste noe av samspillet mellom gaupene og kaninene. Hva om det er veldig få kaniner et år? Da vil det være vanskelig for gaupene å få mat, så mange av de vil dø, og kaninbestanden vil derfor vokse. På slutten av året kan det derfor være mange kaniner og få gauper! Men, hvis vi kun ser på hvor mange kaniner vi har en gang i året, så får vi ikke med oss at gaupebestanden synker raskt nok, så på slutten av året er det kanskje få gauper og få kaniner.

Vi har simulert kanin- og gaupebestanden, og nå vil vi se hvor mange kaniner og gauper vi har på øya vår. Da må vi bruke `pylab` for å plotte resultatet. Her har vi to kurver vi vil se på samtidig, kaninkurven og gaupekurven. Vi kan faktisk plotte begge disse samtidig, og bruke `legend` funksjonen fra pylab for å legge på en merkelapp som viser hva de forskjellige linjene representerer

```
from pylab import plot, xlabel, ylabel, legend, show

plot(range(antall_dager), k, label='Kaniner')
plot(range(antall_dager), g, label='Gauper')

legend()
xlabel('Dager')
ylabel('Antall dyr')
show()
```



Her har vi altså sett hvordan vi kan modellere gaupe og kanin populasjoner matematisk, men stemmer disse tallene overens med virkeligheten? Vel, det kan vi faktisk sjekke. Det har seg nemlig slik at det kanadiske selskapet Hudson's Bay Company samlet inn data på hvor mye hareskinn og gaupeskinn de fikk inn hvert år.



Vi ser altså at mengden harer og gauper følger et periodisk mønster som likner veldig på det vi har i simuleringen vår!

5.6 Oppgaver

Oppgave 4

a) Modifiser koden vi har for å simulere en øy med gauper og kaniner slik at gaupenes dødsrate er lik 0.8 ($d_g = 0.8$). Hvordan påvirker det kaninbestanden på øya?

b) Hva skjer om du også starter med 4 gauper istedenfor 2 gauper?

Oppgave 5

Bytt ut kaninvekstraten i koden over slik at vi tar hensyn til bærekraftig vekst. Sett bæreevnen til øya lik 5000 kaniner og den ubegrensede vekstraten til `5 / dager_i_år`.

Du skal altså modifisere kanin-oppdateringsreglene til å bli

$$k_t = k_{t-1} + f_k(k_{t-1})k_{t-1} - a_g g_{t-1} k_{t-1}, \tag{14}$$

hvor kaninfødselsraten er gitt ved

$$f_k(k) = \frac{5}{365}\left(1 - \frac{k}{5000}\right). \tag{15}$$

Hvordan påvirker dette kanin- og gaupepopulasjonen på øya? Var det store endringer?

Hint: Du kan f.eks. ta utgangspunkt i funksjonen under

```
def f_k(k):
    return (1 - k/5000) * 5 / 365
```

Oppgave 6

Problemløsningsstrategiene vi har brukt her likner faktisk veldig på de man bruker for å simulere pandemier. Den mest kjente pandemisimuleringsmodellen heter SIR-modellen, eller susceptible (smittbar), infected (smittet) og removed (fjernet) modellen. Grunnen til at modellen heter dette er at den deler en befolkning i tre grupper, en gruppe smittbare personer, en gruppe smittede personer og en gruppe med både døde og immune personer. En smittbar person kan bli smittet og en smittet person kan bli fjernet fra simuleringen (altså enten dø eller bli immun).

SIR modellen kan beskrives med den rekursive sammenhengen

$$S_t = S_{t-1} - iI_{t-1}S_{t-1} \tag{16}$$
$$I_t = I_{t-1} + iI_{t-1}S_{t-1} - dI_{t-1} \tag{17}$$
$$R_t = R_{t-1} + dI_{t-1}. \tag{18}$$

i er relatert til smitteraten og d henger sammen med hvor lang tid det tar for en syk person å enten dø eller bli frisk.

Modifiser rovdyr-byttedyr systemet over til å simulere en pandemi med SIR-modellen. Start med 99998 friske personer ($S_0 = 99998$), 2 syke personer ($I_0 = 2$) og 0 fjernede personer ($R_0 = 0$). Sett $i = 0.0000125$ og $d = 0.1$. Simuler systemet i 365 dager.

Hint: Hvis du står fast, kan du se på neste del av kompendiet, som går mer i dybden på SIR modellen.

6 Modellering av pandemier med SIR modellen

Vi har sikkert alle hørt om simuleringsmodeller for pandemier. Disse modellene brukes for å informere politikerene om hva som er lurt å gjøre og hvordan vi bør oppføre oss, men hvordan fungerer egentlig disse modellene?

I dette prosjektet skal vi se litt på en av de enkleste modellene for sykdomsmodellering: den såkalte SIR modellen. SIR står for Susceptible, infected og removed, eller smittbar, smittet og fjernet på norsk, og representerer de tre forskjellige gruppene et menneske kan tilhøre i en slik modell.

La oss nå se nærmere på de tre kategoriene i SIR modellen

- Smittbar (Susceptible): De som kan bli smittet
- Smittede (Infected): De som kan smitte andre
- Fjernet (Removed): De som har blitt immune eller dødd

I pandemimodellene som lages av eksperter er det så klart flere kategorier enn dette, men de tar gjerne utgangspunkt i en SIR modell og legger til flere grupperinger. Hvis vi forstår SIR modellen så har vi godt grunnlag til å forstå modellene som det snakkes om i avisene!

6.1 Matematisk modellering av SIR dynamikk

For å danne en matematisk SIR modell må vi først beskrive hvordan de forskjellige gruppene oppfører seg påvirker hverandre.

6.1.1 Smittbare personer

En person kan bli smittet, og for hver person som smittes, reduseres antall smittbare personer med 1. Vi antar at alle personer har like stor sannsynlighet for å bli smittet, p_i, og får dermed denne likningen:

$$S_{t+1} = S_t - p_i S_t, \tag{19}$$

hvor S_t er antall smittbare ved dag nr. t.

Et klart spørsmål videre er hva som bestemmer sannsynligheten for å bli smittet. Vi vet at sannsynligheten må være avhengig av antall smittede, I_t, for hvis ingen er smittet, er sannsynligheten for å bli smittet lik 0. Tilsvarende, hvis vi går fra 10 til 20 smittede personer, er det logisk at smittesannsynligheten dobles. Det gir altså mening å ha en proposjonalitetsrelasjon mellom smittesannsynligheten og antall smittede. Hvis vi har en slik relasjon, får vi denne likningen

$$S_{t+1} = S_t - iI_tS_t, \tag{20}$$

hvor S_t og I_t er antall smittbare og smittede ved dag nr. t og i er smitteraten.

6.1.2 Antall smittede

Vi vet at antall nye smittede en dag er likt med antall friske som blir smittet den dagen. Hvis vi skriver det matematisk får vi

$$I_{t+1} = I_t + iI_tS_t, \tag{21}$$

hvor S_t og I_t er antall smittbare og smittede ved dag nr. t og i er smitteraten. Hvis vi kun har den relasjonen så vil jo ingen bli immune eller dø. Vi må også ha med en sammenheng for det. Hver dag er det en fast sannsynlighet for at en smittet person enten blir frisk eller dør, som betyr at antall smittede synker proposjonalt med antall smittede. Hvis vi skriver det matematisk får vi denne sammenhengen for antall smittede:

$$I_{t+1} = I_t + iI_tS_t - dI_t, \tag{22}$$

hvor S_t og I_t er antall smittbare og smittede ved dag nr. t, i er smitteraten og d er sannsynligheten for å gå fra å være syk til å enten bli frisk og immun eller dø.

6.1.3 Antall fjernede

Antallet som er fjernet fra simuleringen, altså de som er immune eller døde, har vi jo allerede funnet ut at er proposjonalt med antall syke. Hvis vi skriver det, så får vi denne sammenhengen.

$$R_{t+1} = R_t + dI_t, \tag{23}$$

hvor R_t og I_t er antall fjernede og smittede ved dag nr. t og d er sannsynligheten for at en syk person enten bli frisk eller dør i løpet av et tidssteg.

6.1.4 Kombinere alle likningene

Hvis vi kombinerer alle likningene våre får vi dette uttrykket:

$$S_{t+1} = S_t - iS_tI_t, \tag{24}$$
$$I_{t+1} = I_t + iS_tI_t - dI_t, \tag{25}$$
$$R_{t+1} = R_t + dI_t, \tag{26}$$

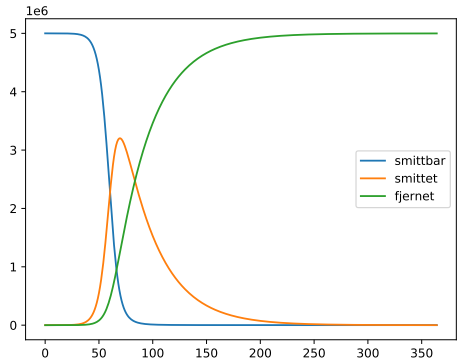
som vi kan programmere slikt vi programmerte rovdyr-byttedyr modellen.

6.1.5 Simulere SIR-systemet

La oss først skrive kode for å simulere en pandemi ved hjelp av SIR modellen, før vi diskuterer hvordan man kan estimere smitteraten, i, og fjerneraten, d, for en sykdom. La oss starte med en fjerne-rate på 0.03 og en infeksjonsrate på 0.00000005

```
from pylab import zeros, plot, legend, show

# Vi definerer modell-parametrene
fjerne_rate = 0.03
infeksjonsrate = 0.00000005

# Vi lager variabler som vi skal lagre
   simuleringsresultatene i
smittbar = zeros(365)
smittet = zeros(365)
fjernet = zeros(365)

# Vi setter antall smittbare og antall smittede i
   befolkningen vår
smittbar[0] = 5_000_000
smittet[0] = 30
fjernet[0] = 0

# Vi kjører simuleringen
for t in range(364):
    smittbar[t+1] = smittbar[t] - infeksjonsrate*smittet[t
       ]*smittbar[t]
    smittet[t+1] = smittet[t] + infeksjonsrate*smittbar[t]
       *smittet[t] - fjerne_rate*smittet[t]
    fjernet[t+1] = fjernet[t] + fjerne_rate*smittet[t]
```

```
22
23  # Vi plotter resultatet fra simuleringen
24  plot(smittbar, label="smittbar")
25  plot(smittet, label="smittet")
26  plot(fjernet, label="fjernet")
27  legend()
28  show()
```



6.2 Estimere parameterne til SIR-modellen

6.2.1 Valg av initialpopulasjoner

Så nå har vi en matematisk modell for å simulere antallet syke i befolkningen, men hvordan kan vi egentlig velge disse ratene?

La oss ta utgangspunkt i en by som Oslo, hvor det er ca 600 000 innbyggere. Hvis vi starter med 2 syke og 599 998 friske personer, så får vi at

$$S_0 = 599998, \tag{27}$$
$$I_0 = 2, \tag{28}$$
$$R_0 = 0. \tag{29}$$

6.2.2 Valg av d

d representerer sannsynligheten for at en smittet person blir frisk eller dør i løpet av en dag. Hvis den forventede syketiden er T, vil d være gitt ved

$$d = \frac{1}{T}. \tag{30}$$

Hvis vi sier at man i snitt er syk i 10 dager, vil får vi at

$$d = 0.1. \tag{31}$$

6.2.3 Valg av i

i er den parameteren som er mest innviklet å velge. Det er en del logikk bak det, men hvert steg er ganske rett frem.

For å velge i kan vi starte med å se på likningen for antall smittbare:

$$S_{t+1} = S_t - iS_tI_t. \tag{32}$$

Vi ser at hver smittede person smitter totalt iS_t nye personer hver dag. Hva betyr dette? La oss ta utgangspunkt i noen eksempelscenarioer og tolke modellen for å finne en fornuftig verdi for dette.

Eksempelscenario: Nesten ingen har blitt syk
Hvis nesten ingen har blitt syke ennå, må nesten alle en syk person møter i løpet av en dag være smittbare. Det betyr at antallet friske som en syk person smitter er gitt ved smittesannsynligheten ganget med antall personer en syk person møter i løpet av en dag. Vi kan skrive det slik

$$S_{t+1} = S_t - mp_{\text{smitte}}I_t, \tag{33}$$

hvor m er hvor mange personer en gjennomsnittsperson møter i løpet av en dag og p_{smitte} er sannsynligheten for å smitte en person du møter.

Eksempelscenario: Halve befolkningen er syk
Hvis halve befolkningen er blitt syk uten at noen smitteværnstiltak er satt inn så vil smittesannsynligheten, p_{smitte}, være konstant. Den eneste endringen vil altså være at antall smittbare personer en syk person møter i løpet av en dag vil være halvert. Hvis vi skriver dette matematisk får vi denne sammenhengen:

$$S_{t+1} = S_t - \frac{m}{2}p_{\text{smitte}}I_t, \tag{34}$$

hvor m nok en gang er hvor mange personer en gjennomsnittsperson møter i løpet av en dag og p_{smitte} er sannsynligheten for å smitte noen om du er syk.

Eksempelscenario: Hele befolkningen har vært eller er syke
Hvis hele befolkningen har vært eller er syke, så vil ingen nye personer bli smittet, altså er

$$S_{t+1} = S_t - 0p_{\text{smitte}}I_t. \tag{35}$$

Hva betyr disse eksempelscenarioene?

Vi ser at antall friske personer som blir smittet i løpet av en dag, t, er gitt ved

$$\text{nye syke} = p_{\text{smitte}}n_t I_t, \tag{36}$$

hvor n_t er antall smittbare personer en gjennomsnittlig syk person møter i løpet av en dag, p_{smitte} er smittesannsynligheten og I_t er antallet syke personer.

Den neste parameteren vi må finne er hvor mange smittbare personer en syk person møteri løpet av en dag, n_t. Dette er gitt ved andelen smittbare i befolkningen

$$n_t = m\frac{S_t}{N}, \tag{37}$$

hvor m er det totale antallet personer en syk person møter i løpet av en dag, S_t er antall smittbare personer den gjeldende dagen og N er den totale befolkningen.

La oss nå sette sammen alt dette for å finne en logisk verdi for i:

$$S_{t+1} = S_t + p_{\text{smitte}}m\frac{S_t}{N}I_t, \tag{38}$$

hvis vi flytter på brøken, får vi at

$$i = \frac{mp_{\text{smitte}}}{N}, \tag{39}$$

hvor m er det totale antallet personer en syk person møter i løpet av en dag, p_{smitte} er sannsynligheten for at en syk person smitter en smittbar person han eller hun møter og N er det totale befolkningstallet.

Overslag av verdien til i

- Det er en 5% sannsynlighet for å smitte noen
  - $p_{\text{smitte}} = 0.05$
- Hver person møter i snitt 25 nye i løpet av en dag

- $m = 25$

Hvis vi kombinerer disse antakelsene med et innbyggertall på 600000, får vi

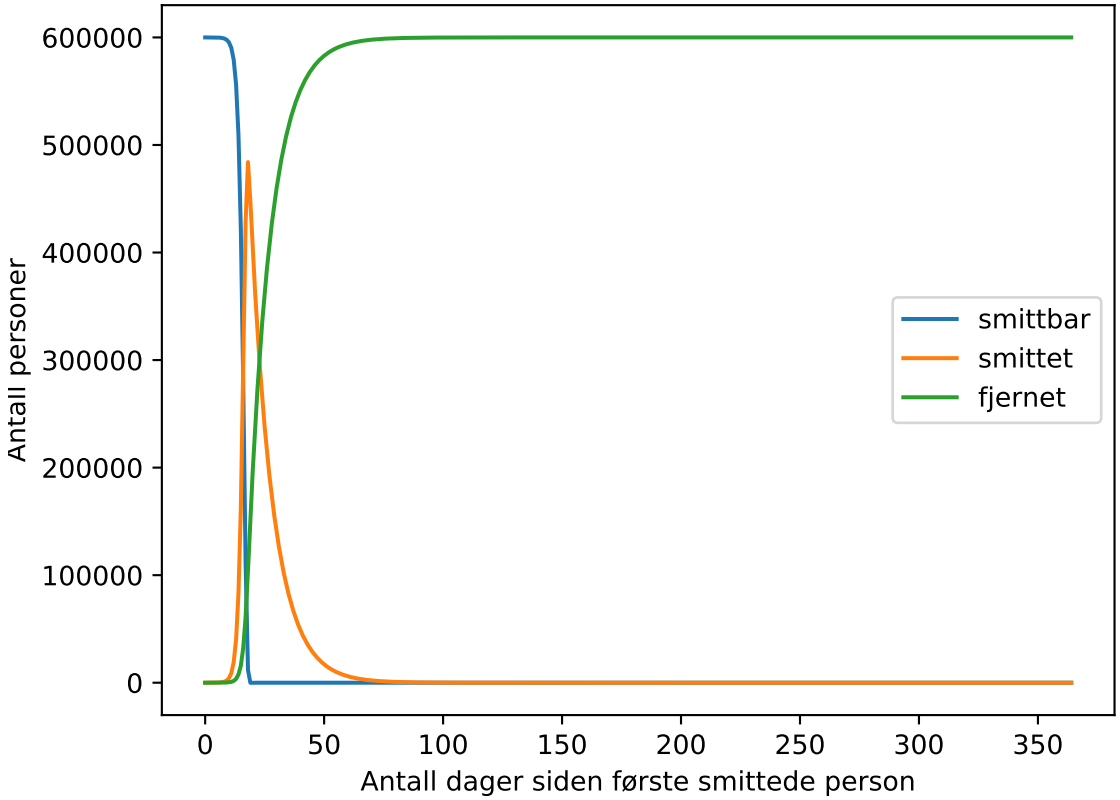
$$i = \frac{0.05 \times 25}{600000}. \tag{40}$$

6.3 Implementere en sykdomssimulering

La oss nå simulere sykdomsforløpet til en kontrollert befolkning. Vi setter antallet personer vi møter i løpet av en dag til å være 25, smittesannsynligheten lik 0.05 (5%) og befolkningen til 600 000.

```
from pylab import zeros, plot, show, legend, xlabel,
    ylabel

# Befolkningsparametre
befolkning = 600_000
antall_syke_dag_en = 2

# Smitteparametre
antall_man_møter = 25
smittesannsynlighet = 0.05

# Syketid
gjennomsnittlig_syketid = 10

# Modellparametre
fjerne_rate = 1 / gjennomsnittlig_syketid
infeksjonsrate = antall_man_møter*smittesannsynlighet/
    befolkning

# Simuleringslengde
simuleringslengde = 365

# Startverdier til simuleringen
smittbar = zeros(simuleringslengde)
smittet = zeros(simuleringslengde)
fjernet = zeros(simuleringslengde)

```

```
26
27 smittbar[0] = befolkning - antall_syke_dag_en
28 smittet[0] = antall_syke_dag_en
29
30 for i in range(simuleringslengde - 1):
31     smittbar[i+1] = smittbar[i] - infeksjonsrate*smittet[i
       ]*smittbar[i]
32     smittet[i+1] = smittet[i] + infeksjonsrate*smittbar[i]
       *smittet[i] - fjerne_rate*smittet[i]
33     fjernet[i+1] = fjernet[i] + fjerne_rate*smittet[i]
34
35 plot(smittbar, label="smittbar")
36 plot(smittet, label="smittet")
37 plot(fjernet, label="fjernet")
38 xlabel("Antall dager siden første smittede person")
39 ylabel("Antall personer")
40 legend()
41 show()
```


600000
500000
400000
300000
200000
100000
0
Antall personer
0
50
100
150
200
250
300
350
Antall dager siden første smittede person
smittbar
smittet
fjernet

A Relevante kompetansemål

I dette kompendiet går vi igjennom simulering av populasjoner og pandemier ved hjelp av rekursive sammenhenger.

Prosjektet kombinerer algoritmisk tenking, populasjonsdynamikk, bærekraftig vekst, rekursive sammenhenger og tallfølger. Det rører derfor ved flere forskjellige kompetansemål. Man kan også tilpasse hvilke kompetansemål man vil rette seg mot ved å endre hvilke biter av matematikken/koden elever må løse selv og hvilke som gis ferdig utfylt. Under følger et utplukk relevante kompetansemål.

1T

- formulere og løyse problem ved hjelp av algoritmisk tenking, ulike problemløysingsstrategiar, digitale verktøy og programmering
- identifisere variable storleikar i ulike situasjonar, setje opp formlar og utforske desse ved hjelp av digitale verktøy
- modellere situasjonar knytte til ulike tema, drøfte, presentere og forklare resultata og argumentere for om modellane er gyldige

S2

- utforske egenskaper ved ulike rekker og gjøre rede for praktiske anvendelser av egenskaper ved rekker
- utforske rekursive sammenhenger ved å bruke programmering og presentere egne framgangsmåter
- modellere og analysere eksponentiell og logistisk vekst i reelle datasett

R1

- modellere og analysere eksponentiell og logistisk vekst i reelle datasett

R2

- utforske rekursive sammenhenger ved å bruke programmering og presentere egne framgangsmåter
- utforske egenskaper ved ulike rekker og gjøre rede for praktiske anvendelser av egenskaper ved rekker
- gi eksempler på ulike situasjoner som kan modelleres ved å bruke ulike matematiske funksjoner, og modellere og analysere slike situasjoner ved å bruke reelle datasett

Naturfag

- vurdere og lage programmer som modellerer naturfaglige fenomener

B PyLab, NumPy og Matplotlib

I dette kompendiet har vi brukt PyLab-biblioteket. Grunnen til det er at vi i Python har mange forskjellige bibliotek for vitenskapelige anvendelser. De mest brukte er NumPy (numerical python) og Matplotlib. Det PyLab gjør, er å samle de nyttigste verktøyene fra disse bibliotekene et sted, slik at det er lettere å skrive kode for å løse problemer. Hvis du leser bøker eller guider som bruke NumPy-arrayer, så er det faktisk det du gjør med PyLab og. På samme måte, hvis du ser noen prate om plotting i Matplotlib, så er det faktisk Matplotlib vi bruker for plotting her, men vi henter plotte-kommandoene fra `pylab` istedenfor `matplotlib.pyplot`.

C Bonus for R2: Differensiallikninger

I R2 lærer man om enkle differensiallikninger og hvordan vi kan løse dem eksakt. Dessverre er det slik at det nesten ikke er noen differensiallikninger som vi kan finne en analytisk løsning på. Isteden bruker vi to verktøy:

- Numeriske løsninger

- Kvalitative tolkninger og stabilitetsanalyse

I dette heftet har vi faktisk løst differensiallikninger numerisk uten å si noe om det! La oss for eksempel ta den første likningen for eksponensiell vekst:

$$k_t = k_{t-1} + fk_{t-1}, \tag{41}$$

hvor k_t er antall kaniner etter t år. her ser vi på et år om gangen, men vi kan alltids endre på dette og se på dager istedenfor:

$$k_t = k_{t-1} + \frac{fk_{t-1}}{365}, \tag{42}$$

hvor k_t er antall kaniner etter t dager. Vi kan se på enda mindre tidssteg, Δt, og da får vi denne likningen:

$$k_t = k_{t-1} + \Delta t f k_{t-1}, \tag{43}$$

som er måten vi numerisk ville løst differensiallikningen

$$\frac{dk(t)}{dt} = fk(t) \tag{44}$$

numerisk med Euler's metode.

Ok, nå har vi en differensiallikning, men hvorfor kan vi si at (43) er en numerisk måte å løse differensiallikningen (44)? La oss bruke definisjonen på den deriverte:

$$\frac{dk(t)}{dt} = \lim_{\Delta t \to 0} \frac{k(t + \Delta t) - k(t)}{\Delta t} = fk(t). \tag{45}$$

Datamaskiner kan ikke regne ut grenseverdier, så istedenfor å ha grensen når Δt går mot null, kan vi bare sette Δt til et veldig lavt tall. Da får vi likningen

$$\frac{k(t + \Delta t) - k(t)}{\Delta t} \approx fk(t). \tag{46}$$

Hvis vi ganger med Δt og legger til $k(t)$ på begge sider av likhetstegnet, får vi

$$k(t + \Delta t) \approx k(t) + \Delta t f k(t), \tag{47}$$

som er ekvivalent med (43) om vi setter $k_{t-1} = k(t)$.