

Kap. 6: Plotting av kurver (kap 8.1)

Joakim Sundnes ved Simula Research Laboratory &
Universitetet i Oslo, Institutt for informatikk

Sep 24, 2026

Tirsdag 22. september

- Dictionaries (fra forrige uke)
- Live-programmering av oppg. 6.7
- Plotting med matplotlib (kap 8.1)
- Oppg. 5.9, 5.10, 5.11, 5.13

Torsdag 24. september

- Mer om plotting, animasjoner
- Oppg. 5.39
- Numerisk løsning av likninger

Plotting av en funksjon i Python

Plott av en funksjon $f(x)$ i Python skjer i to steg:

- Regn ut punkter (x, y) som ligger på kurven $y = f(x)$
 - Verktøy: Numpy (typisk `linspace` + vektoriserte beregninger)
- Plott punktene (vanligvis med rette streker mellom)
 - Verktøy: Matplotlib

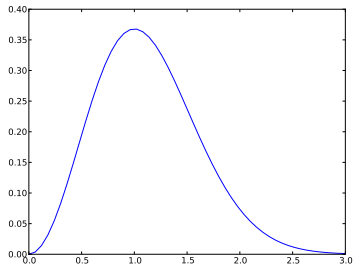
Enkelt eksempel på plotting

Plott kurven $y(t) = t^2 e^{-t^2}$

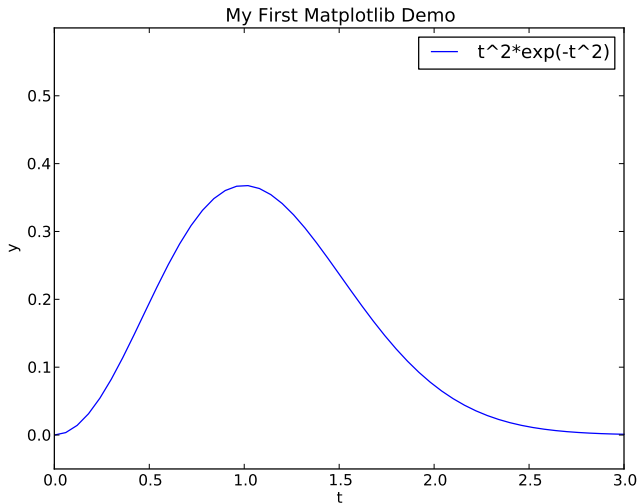
```
import matplotlib.pyplot as plt
import numpy as np

# Lag punkter langs kurven
t = np.linspace(0, 3, 51)          # 50 intervaller i [0, 3]
y = t**2*np.exp(-t**2)            # vektorisert uttrykk

plt.plot(t, y)                    # lag plott på skjermen
plt.show()
```



Et plott bør ha tittel, merklapper, etc



Koden som lager det forrige plottet

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

def f(t):
    return t**2*np.exp(-t**2)

t = np.linspace(0, 3, 51)      # t-koordinater
y = f(t)                      # tilsvarende y-verdier

plt.plot(t, y, label="t^2*exp(-t^2)")

plt.xlabel('t')                # merking av x-aksen
plt.ylabel('y')                # merking av y-aksen
plt.legend()                   # marker kurven
plt.axis([0, 3, -0.05, 0.6])  # [tmin, tmax, ymin, ymax]
plt.title('My First Matplotlib Demo')
plt.show()
```

Plotting av flere kurver i samme plott

Plott $t^2e^{-t^2}$ og $t^4e^{-t^2}$ i samme plott:

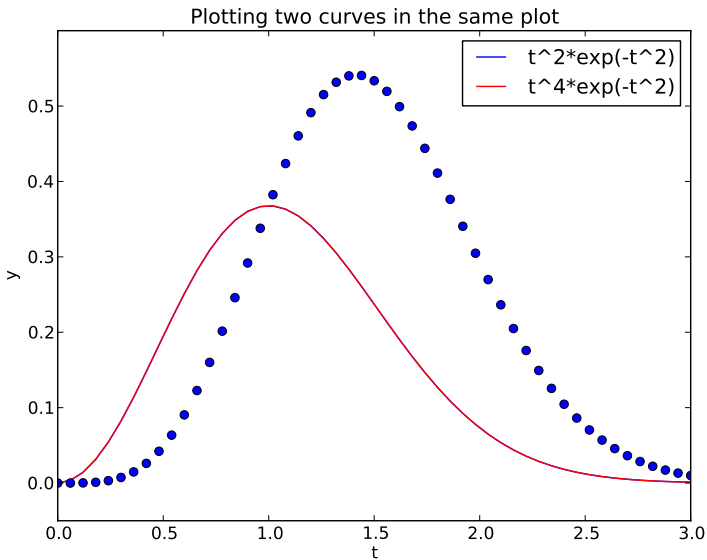
```
import matplotlib.pyplot as plt
import numpy as np

t = np.linspace(0, 3, 51)
y1 = t**2*np.exp(-t**2)
y2 = t**4*np.exp(-t**2)

plt.plot(t, y1, 'r-', label = 't^2*exp(-t^2)')
plt.plot(t, y2, 'bo', label = 't^4*exp(-t^2)')

plt.xlabel('t')
plt.ylabel('y')
plt.legend()
plt.title('Plotting av to kurver i samme plott')
plt.savefig('tmp2.png')
plt.show()
```

Plottet med to kurver



Matplotlib velger farger automatisk, men hvis vi ønsker det kan vi kontrollere linjetype og farge:

```
plot(t, y1, 'r-')    # rød (r) linje (-)  
plot(t, y2, 'bo')    # blå (b) sirkler (o)
```

```
# eller
```

```
plot(t, y1, 'r-', t, y2, 'bo')
```

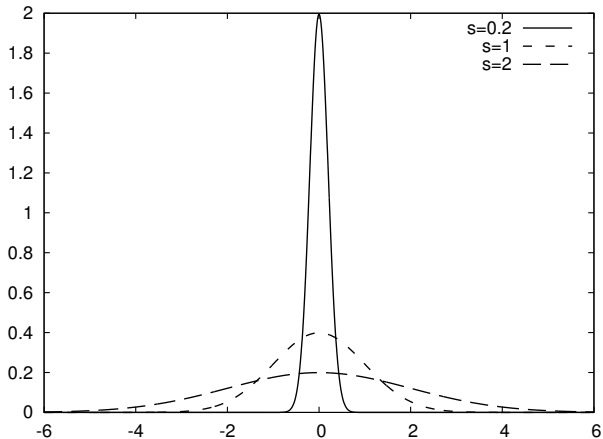
For dokumentasjon av farger og linjestiler, se Matplotlib-dokumentasjonen på nett.

Raskt plott med minimal skriving

Vi kan også skrive det matematiske uttrykket rett inn i argumentlisten:

```
t = np.linspace(0, 3, 51)
plt.plot(t, t**2*exp(-t**2), t, t**4*exp(-t**2))
plt.show()
```

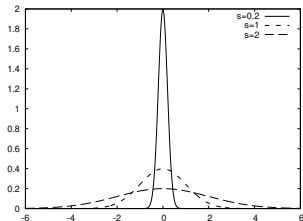
Animasjoner/filmer



Gauss-/klokkefunksjonen

$$f(x; m, s) = \frac{1}{\sqrt{2\pi}} \frac{1}{s} \exp \left[-\frac{1}{2} \left(\frac{x - m}{s} \right)^2 \right]$$

- m er plasseringen av toppen
- s er et mål på bredden av funksjonen
- Lag en film (animasjon) av hvordan $f(x; m, s)$ endrer form når s går fra 2 til 0.2



Filmer lages av et (stort) sett med individuelle plott

- Mål: lage en film som viser hvordan $f(x)$ varierer i form når s minker
- Idé: sett mange plott (for forskjellige s -verdier) sammen (som en tegnefilm)
- Veldig viktig: lås y -aksen! Ellers vil y -aksen alltid tilpasse seg funksjonens toppunkt, og det visuelle inntrykket blir helt feil.

Tre alternative oppskrifter

- ❶ La animasjonen kjøre *live*, uten å lagre noen filer
 - Ikke mulig å sette på pause, senke farten osv.
- ❷ Gå i løkke over alle dataverdier, plott og lag en kopi (fil) for hver verdi, kombiner alle filene til en film
 - Krever separat programvare (for eksempel *ImageMagick*) for å se animasjonen
- ❸ Bruk et 'FuncAnimation'-objekt fra 'matplotlib'
 - Spiller av animasjonen *live*
 - Avhenger av ekstern programvare for å lagre en filmfil

Alt. 1: Generell idé

- Fiksér aksene!
- Bruk en 'for'-løkke til å gå gjennom s -verdier
- Beregn nye y -verdier og oppdater plottet for hver runde i løkken

Alt. 1: Komplette kode

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

def f(x, m, s):
    return (1.0/(np.sqrt(2*np.pi)*s))*np.exp(-0.5*((x-m)/s)2)

m = 0; s_start = 2; s_stop = 0.2
s_values = np.linspace(s_start, s_stop, 30)

x = np.linspace(m - 3*s_start, m + 3*s_start, 1000)

# f er maks for x=m (mindre s gir større maksverdi)

max_f = f(m, m, s_stop)

y = f(x,m,s_stop)
lines = plt.plot(x,y) # Returnerer en liste med linje-objekter!

plt.axis([x[0], x[-1], -0.1, max_f])
plt.xlabel('x')
plt.ylabel('f')

for s in s_values:
    y = f(x, m, s)
    lines[0].set_ydata(y) # oppdater plottdata og tegn på nytt
    plt.draw()
    plt.pause(0.1)
```

- Samme 'for'-løkke som i alternativ 1
- Bruk f-streng-formatering for å generere et unikt filnavn for hvert plott
- Lagre fil

Alt. 2: Komplette kode

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

def f(x, m, s):
    return (1.0/(np.sqrt(2*np.pi)*s))*np.exp(-0.5*((x-m)/s)**2)

m = 0; s_start = 2; s_stop = 0.2
s_values = np.linspace(s_start, s_stop, 30)

x = np.linspace(m - 3*s_start, m + 3*s_start, 1000)

# f er maks for x=m (mindre s gir større maksverdi)

max_f = f(m, m, s_stop)

y = f(x,m,s_stop)
lines = plt.plot(x,y)

plt.axis([x[0], x[-1], -0.1, max_f])
plt.xlabel('x')
plt.ylabel('f')

frame_counter = 0
for s in s_values:
    y = f(x, m, s)
    lines[0].set_ydata(y) # oppdater plottdata og tegn på nytt
    plt.draw()
    plt.savefig(f'tmp_{frame_counter:04d}.png') # unikt filnavn
    frame_counter += 1
```

Hvordan kombinere plottfiler til en film (videofil)

Vi har nå mange filer:

`tmp_0000.png tmp_0001.png tmp_0002.png ...`

Vi bruker et program til å kombinere disse filene til en videofil:

- `convert` for animert GIF-format (hvis det bare er noen få plottfiler)
- `ffmpeg` (eller `avconv`) for MP4, WebM, Ogg, og Flash-formater

Lag og spill av animert GIF-fil

Verktøy: convert fra programvarepakken ImageMagick.

Unix-kommando:

```
Terminal> convert -delay 20 tmp_*.png movie.gif
```

Forsinkelse: 30/100 s, dvs. 0,5 s mellom hver ramme.

Spill av animert GIF-fil med animate fra ImageMagick:

```
Terminal> animate movie.gif
```

eller åpne filen i en nettleser.

Alt. 3: Generell idé

- Lag en funksjon for å oppdatere plottet:
- Oppdaterer plottet ved å beregne verdier og kalle `set_ydata`
- (Valgfri funksjon for å initialisere plottet)
- Lag en liste eller array av argumentet som endres (her `s`)
- Send funksjonen og listen som argumenter for å opprette et `FuncAnimation`-objekt
- Bruk funksjoner i dette objektet til å animere, lagre en filmfil, osv.

Alt. 3: Komplette kode

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

def f(x, m, s):
    return (1.0/(np.sqrt(2*np.pi)*s))*np.exp(-0.5*((x-m)/s)**2)

m = 0; s_start = 2; s_stop = 0.2
s_values = np.linspace(s_start,s_stop,30)

x = np.linspace(-3*s_start,3*s_start, 1000)

max_f = f(m,m,s_stop)

plt.axis([x[0],x[-1],0,max_f])
plt.xlabel('x')
plt.ylabel('y')

y = f(x,m,s_start)
lines = plt.plot(x,y) # innledende plott for å lage lines-objektet

def next_frame(frame):
    y = f(x, m, frame)
    lines[0].set_ydata(y)
    return lines

ani = FuncAnimation(plt.gcf(), next_frame, frames=s_values, interval=100)
ani.save('movie.mp4',fps=20)
plt.show()
```

Merknader om å lage filmer

- Å lage faktiske filmfiler krever ekstern programvare som ImageMagick eller ffmpeg
- Programvaren kan være vanskelig å installere (det finnes enkle oppskrifter, men erfaringen er at disse ikke alltid fungerer)
- For animasjonsoppgavene i dette kurset er det ikke nødvendig å lage selve filmfilene. Du kan enten:
 - Bruke Alt 1 eller Alt 3 for å la animasjonen kjøre *live*
 - Bruke Alt 2 til å lage mange bildefiler
- Hvis du også kan lage filmfilene, er dette flott, men ikke bruk for mye tid på det hvis det er vanskelig å installere verktøyene