Endimensjonale Numpy-arrays og matplotlib (TDT 4110)

Læringsmål

- Numpy-arrays
- Plotting

Det anbefales på det sterkeste at du gjør oppgaven som omhandler matplotlib før du begynner på denne øvingsoppgaven, ettersom vi her kommer til å anta at du har grunnleggende kunnskaper om plotting i python.

Generelt om numpy-arrays

Det kan være lurt å lese gjennom dette før du begynner på øvingen

Numpy-arrays kommer fra *biblioteket* `numpy` . Disse har noen likheter med lister i python, men er en del raskere til beregninger. Numpy-arrays blir mye brukt innen blant annet utregninger i fysikk fordi de har masse innebygde funksjoner, slik som matrisemultiplikasjon som dere skal lære om i matte 3.

Vi skal nå gå gjennom noen eksempler på hvordan `numpy` fungerer.

Før vi kan bruke numpy-biblioteket er vi nødt til å importere det. Vi kaller det for np, slik at vi kan skrive `np` i stedet for numpy.

In [ ]:

```
import numpy as np
```

Nå kan vi bruke alle funksjonene til numpy ved å skrive `np.<funksjonsnavn>`

Endimensjonale numpy-arrays

Det finnes flere måter å lage numpy-arrays på. En måte å putte inn en vanlig python-liste og få ut en numpy-array med funksjonen `np.array` , som under:

In [ ]:

```
import numpy as np
array = np.array([1, 2, 3])
print(array)
```

En av grunnene til at numpy-arrays er raskere enn vanlige python-lister er at den tvinger alle elementene i lista til å være av samme type. Dette kan gi noen sære konsekvenser hvis du ikke følger med. Kjør eksempelet under:

In [ ]:

```
import numpy as np
array = np.array([1, 2, 3, "mjau"])
print(array)
```

Her blir alle tallene i listen konvertert til strenger, siden typen til array´en må være streng ettersom vi har med `"mjau"` . Det samme skjer hvis vi for eksempel prøver å lage en numpy-array med både heltall og flyttall. Da vil alle heltallene bli gjort om til flyttall.

Numpy-arrays har en del av den samme funksjonaliteten som lister, som du kan se i eksempelet under:

In [ ]:

```
import numpy as np
array = np.array([1, 2, 3, 4, 5, 6])

print(array)

print(array[0])

array[5] = 3
print(array)

print(array[3:])
```

Matteoperasjoner: Når det kommer til numpy-arrays som inneholder nummer (som er det du som oftest jobber med) har du en del ekstrafunksjoner som ikke finnes på vanlige lister. Kjør kodeblokkene under én og én og se om du skjønner hva operasjonene under gjør:

In [ ]:

```
import numpy as np
a = np.array([1, 2, 3, 4, 5])
```

In [ ]:

```
print(a + 2)
print(np.add(a, 2))  # disse er like
```

In [ ]:

```
print(a**2)
```

In [ ]:

```
print(np.sqrt(a))
```

In [ ]:

```
print(np.max(a))
```

Operasjoner med flere numpy-arrays. Her er det viktig at array´ene er av samme størrelse. Hvis de ikke er det får du en error.

In [ ]:

```
import numpy as np
a = np.array([1, 2, 3, 4, 5])
b = np.array([8, 13, 4, 5, 3])

print(a + b)
print(np.add(a, b)) # Denne linja gjør det samme som den over

print(a * b)

print(a.dot(b)) # prikkprodukt
```

Arange-funksjonen og plotting

En mye brukt måte å lage numpy-arrays på er ved hjelp av `arange` -funksjonen. Denne vil lage en numpy-array av tall med like stor avstand mellom hvert tall. For eksempel vil array´en som blir laget under inneholde tallene fra og med 0.0 til (men ikke med) 5.0 med 0.1 i mellomrom. **Kjør koden og test selv**

In [ ]:

```
import numpy as np
array = np.arange(0, 5, 0.1)
print(array)
```

Et av tilfellene hvor `np.arange` er praktisk er hvis vi ønsker å printe en graf. For eksempel en graf som viser $f(x) = x$ mot $g(x) = x^2$. Når vi bruker `matplotlib` er det ikke noen forskjell om vi bruker en python-liste eller en numpy-array til å plotte grafen. Dere kan se et eksempel på plottingen under og se om dere skjønner hva som skjer:

Test ut koden under og prøv å forstå hva som skjer. Hvis du sitter helt fast kan det lønne seg å se igjen på øvingen om matplotlib.

In [ ]:

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

x_verdier = np.arange(0.0, 5.0, 0.1)

plt.plot(x_verdier, x_verdier, "r-")
plt.plot(x_verdier, x_verdier**2, "b-") # x_verdier**2 returnerer en numpy-array hvor x
-> x^2 for alle x
plt.title('f(x) = x (rød) mot g(x) = x^2 (blå)')
plt.show()
```

a) Tegning av sinusbølge

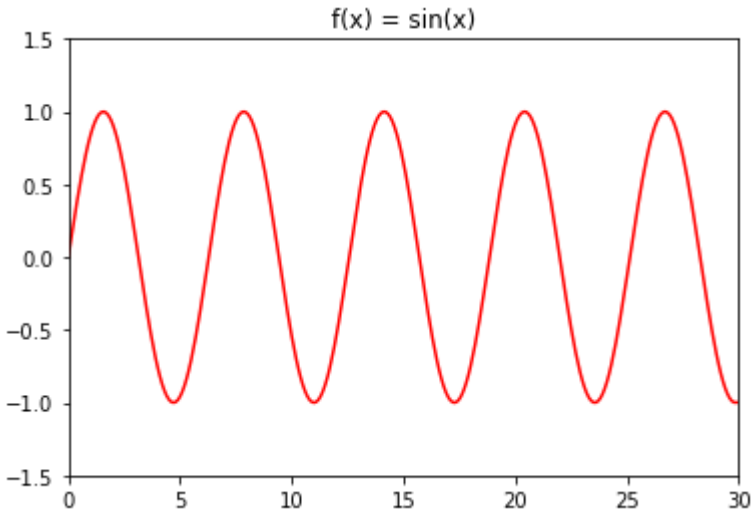
Nå skal du gjøre den samme oppgaven som i forrige plotteøving Matplotlib (Matplotlib.ipynb), bare med numpy-arrays. Plot en sinusfunksjon med `plt.plot` fra 0 til 30 ved hjelp av numpy-arrays.

Skriv koden din på markert sted i kodeblokken under:

In [3]:

```
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
# SKRIV DIN KODE HER
x_verdier = np.arange(0.0, 30.0, 0.1)
y_verdier = [np.sin(x) for x in x_verdier]

plt.title('f(x) = sin(x)')
plt.plot(x_verdier, y_verdier, 'r-')
plt.axis([0, 30, -1.5, 1.5]) # x-aksen går her fra 0 til 20, mens y-aksen går fra 0 til 15
plt.show()
```



Hint

Se på de tidligere eksemplene og bruk denne linken (https://docs.scipy.org/doc/numpy/reference/routines.math.html) for å få en oversikt over de matematiske operatorene som kan brukes med numpy-arrays

b) Flere sinus-bølger på hverandre

Du skal nå lage funksjonen `double_sine_plot(freq_mult)` . Denne skal ta inn "freq_mult", som skal være forholdet mellom frekvensene til to sinusbølger. Funksjonen skal lage et plot med:

- Den ene sinusbølgen i fargen gul med *dotted line style*
- Den andre sinusbølgen i fargen cyan, også med dotted line style
- De to sinusbølgene plusset sammen, plottet med en heltrukket linje.
- En svart heltrukket (horisontal) linje i y=0

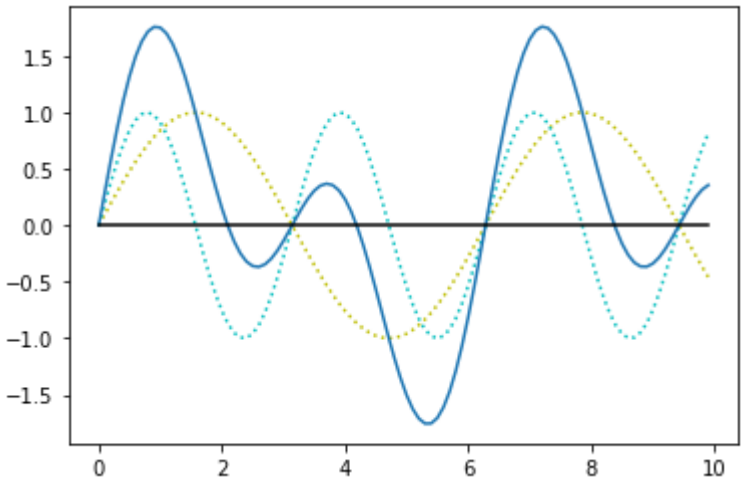
Oversikten over farger og linjestiler kan du finne her : https://matplotlib.org/2.1.1/api/_as_gen/matplotlib.pyplot.plot.html (https://matplotlib.org/2.1.1/api/_as_gen/matplotlib.pyplot.plot.html).

Skriv koden på markert sted i kodeblokken under:

In [12]:

```
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
# SKRIV DIN KODE HER
def double_sine_plot(freq_mult):
    x_verdier = np.arange(0.0, 10.0, 0.1)
    y_verdier = np.array([np.sin(x) for x in x_verdier])
    plt.plot(x_verdier, y_verdier, 'y:')
    y_verdier2 =np.array([np.sin(x*freq_mult) for x in x_verdier])
    plt.plot(x_verdier, y_verdier2, 'c:')
    plt.plot(x_verdier, y_verdier+y_verdier2, '-')
    plt.plot(x_verdier,[0 for x in x_verdier],'k-')
    plt.show

double_sine_plot(2)
```



c) (frivillig) frekvensforhold i tillegg

Kopier koden din fra oppgave b under og endre den slik at funksjonen også tar inn variabelen `amp_mult`. Denne skal beskrive forholdet mellom amplitudene til sinus-bølgene. Test så ut funksjonen din med forskjellige tall og se på grafene som oppstår.

Kopier koden din fra oppgave b inn i feltet under og endre den:

In [ ]:

```

```