www.math-stockholm.se/cirkel

15 april 2021

Välkomna!

Idag:

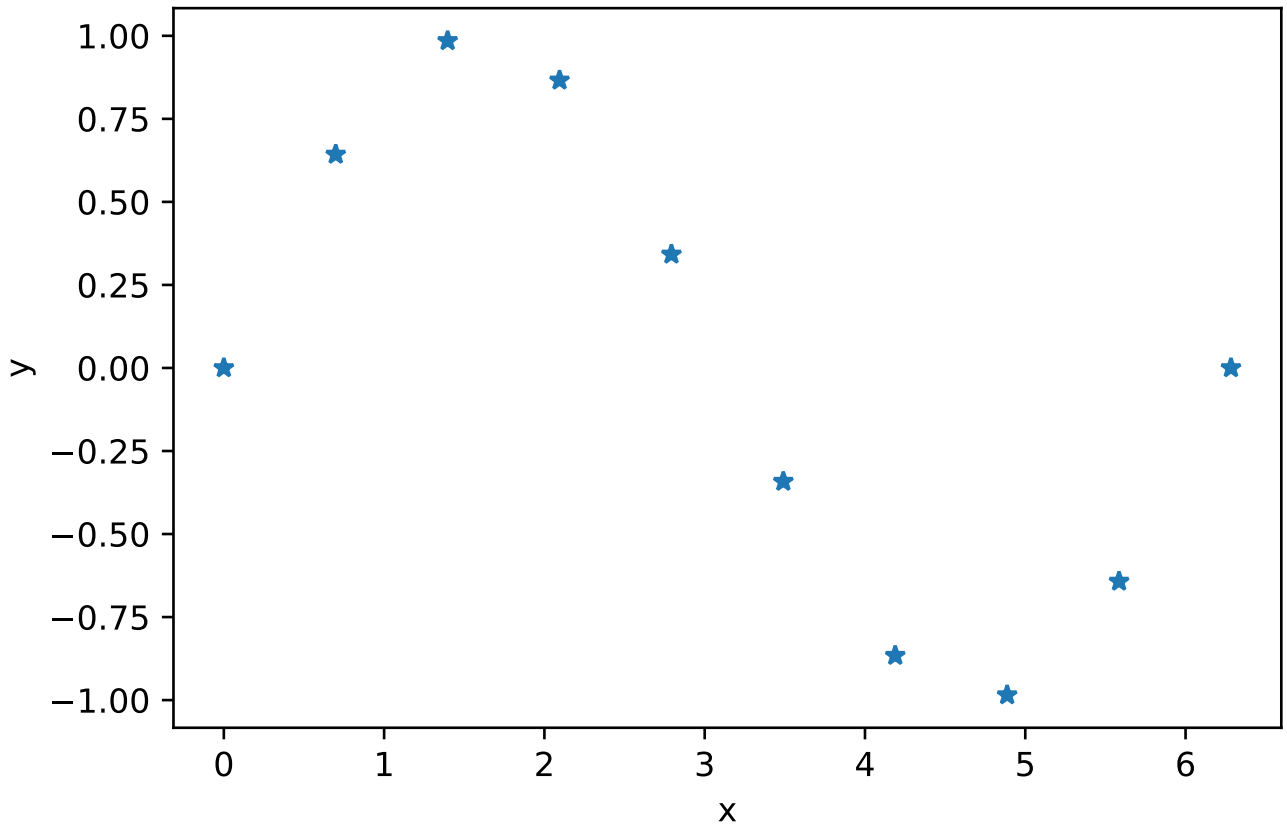
- Skapa grafer i Python
- Skapa diskreta sinuskurvor (toner) i Python
- Att skapa en `.wav` fil vi kan lyssna på (Audacity)
- Lyssna på det som vi har skrivit
  - Exempel på A-durskalen, A-durtreklang
  - En melodi
- Se hur man kan använda FFT för att se vilka toner spelas i en ackord
- (Lite mer praktiskt än innan)

Att skapa grafer i Python

Sinus med 10 punkter

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0,2*np.pi,10)
y = np.sin(x)
plt.plot (x,y,'*')
plt.xlabel('x')
plt.ylabel('y')
```

Att skapa grafer i Python



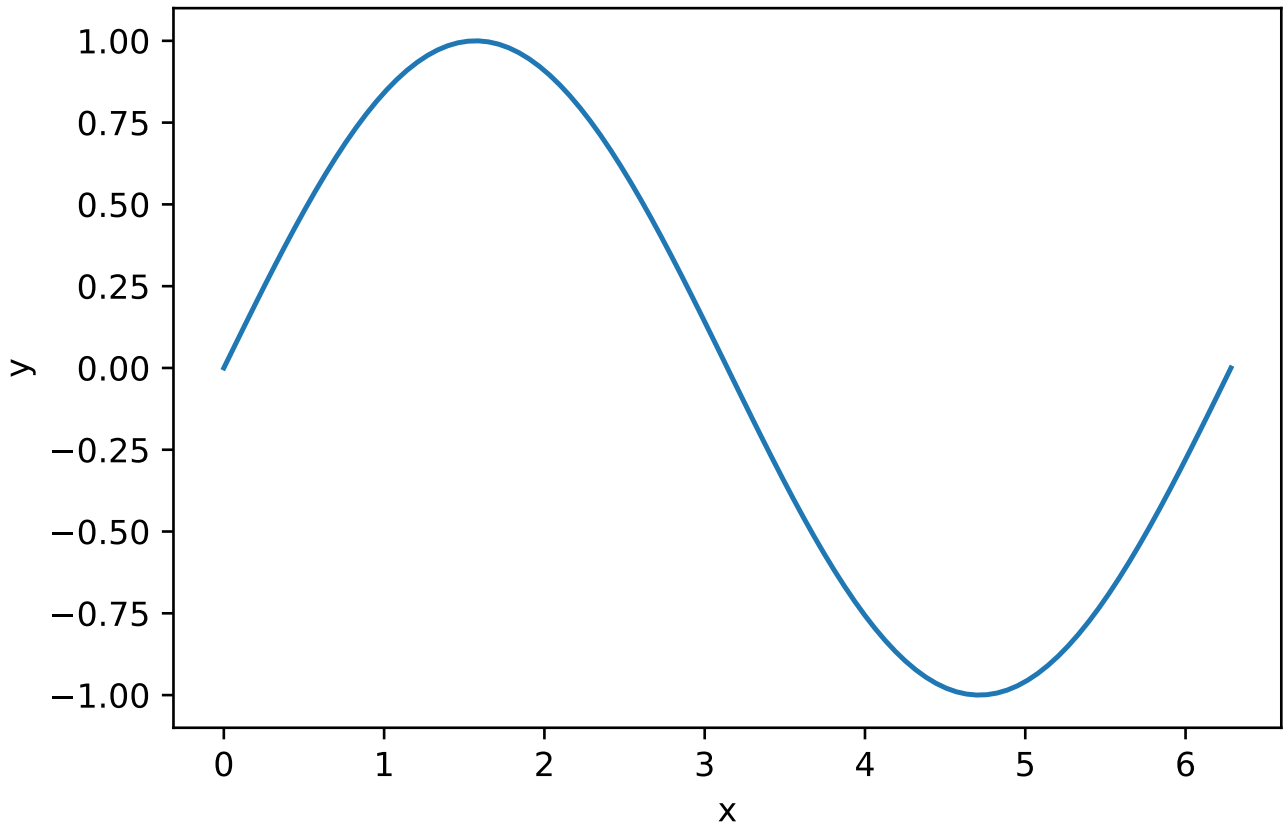
Figur: $y = \sin(x)$ med 10 punkter

Att skapa grafer i Python

Sinus med 100 punkter

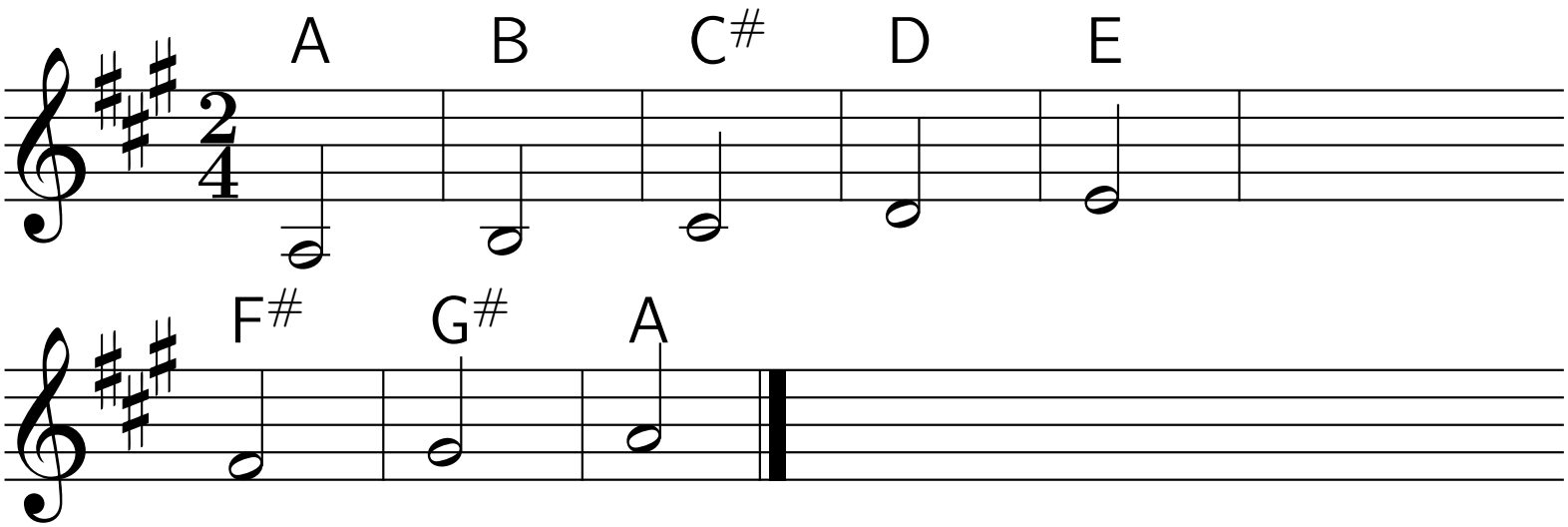
```
import numpy as np
import matplotlib.pyplot as plt
x2 = np.linspace(0,2*np.pi,100)
y2 = np.sin(x2)
plt.show()
plt.plot (x2,y2)
plt.xlabel('x')
plt.ylabel('y')
```

Att skapa grafer i Python



Figur: $y = \sin(x)$ med 100 punkter

A-durskalan



- Obs! Helt okej om ni aldrig har sett det här innan
- Nästa föreläsning
- Men vi kan lyssna
- Varning: högljud!

Spara alla frekvens (i Hertz) som konstanter

```
import numpy as np


#Toner i A-durskalan
f0 = 220 # A
f1 = 247 # B
f2 = 277 # C #
f3 = 294 # D
f4 = 330 # E
f5 = 370 # F #
f6 = 416 # G #
f7 = 440 # A
f0_a = f0*2 # A (samma som f7)
f1_a = f1*2 # B (nästa ton)
```

Skapa olika toner

```
N = 44100 #Hur många punkter
x = np.linspace(0, 1, N)

#Skapa vektorer av diskreta sinuskurvor av alla toner
y0 = np.sin(2*np.pi*f0*x)
y1 = np.sin(2*np.pi*f1*x)
y2 = np.sin(2*np.pi*f2*x)
y3 = np.sin(2*np.pi*f3*x)
y4 = np.sin(2*np.pi*f4*x)
y5 = np.sin(2*np.pi*f5*x)
y6 = np.sin(2*np.pi*f6*x)
y7 = np.sin(2*np.pi*f7*x)
y0_a = np.sin(2*np.pi*f0_a*x)
y1_a = np.sin(2*np.pi*f1_a*x)
```

Skapa olika toner (2)

```
N = 44100; x = np.linspace(0, 1, N)
y0 = np.sin(2*np.pi*f0*x)
y1 = np.sin(2*np.pi*f1*x)
y2 = np.sin(2*np.pi*f2*x)
y3 = np.sin(2*np.pi*f3*x)
y4 = np.sin(2*np.pi*f4*x)
y5 = np.sin(2*np.pi*f5*x)
y6 = np.sin(2*np.pi*f6*x)
y7 = np.sin(2*np.pi*f7*x)
```

Notera:

- Varje ton har $N = 44100$ punkter (samma längd)
- Varje ton ska spelas i en sekund (mer nästa föreläsning)
- Kan lyssna

Skriva en melodi

Ta olika toner från A-durskalen (diskreta sinuskurvor med olika frekvenser) i en särskild ordning

Python kod för att skapa en `.wav` fil

```
f = open('do_re_mi.wav','wb') # Öppna filen
f.write(y4); f.write(y0); f.write(y5)
f.write(y3); f.write(y2); f.write(y0)
f.write(y1); f.write(y1);
f.write(y4); f.write(y0); f.write(y5); f.write(y6)
f.write(y0_a); f.write(y1_a); f.write(y0_a)
f.close() # Stänger filen
```

Och sedan . . .

Kan vi lyssna på den i Audacity

Skriva samma melodin men långsammare

Ta **samma** olika toner från A-durskalen (diskreta sinuskurvor med olika frekvenser) i **samma** ordning, men skriv dubbelt så många!

```
f = open('do_re_mi_lang.wav','wb')
f.write(y4); f.write(y4); f.write(y0); f.write(y0);
f.write(y5); f.write(y5); f.write(y3); f.write(y3);
f.write(y2); f.write(y2); f.write(y0); f.write(y0);
f.write(y1); f.write(y1); f.write(y1); f.write(y1);
f.write(y4); f.write(y4); f.write(y0); f.write(y0);
f.write(y5); f.write(y5); f.write(y6); f.write(y6);
f.write(y0_a); f.write(y0_a); f.write(y1_a); f.write(y1_a);
f.write(y0_a); f.write(y0_a); f.write(y0_a); f.write(y0_a);
f.close()
# Stänger filen
```

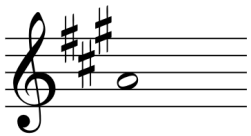
Skriva samma melodin men långsammare

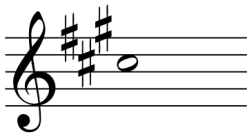
```
f = open('do_re_mi_lang.wav','wb')
f.write(y4); f.write(y4); f.write(y0); f.write(y0);
f.write(y5); f.write(y5); f.write(y3); f.write(y3);
f.write(y2); f.write(y2); f.write(y0); f.write(y0);
f.write(y1); f.write(y1); f.write(y1); f.write(y1);
f.write(y4); f.write(y4); f.write(y0); f.write(y0);
f.write(y5); f.write(y5); f.write(y6); f.write(y6);
f.write(y0_a); f.write(y0_a); f.write(y1_a); f.write(y1_a);
f.write(y0_a); f.write(y0_a); f.write(y0_a); f.write(y0_a);
f.close()
# Stänger filen
```

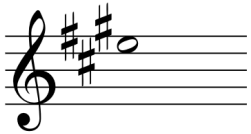
- Varje ton spelas nu i två sekunder
- Vi kan lyssna på den också

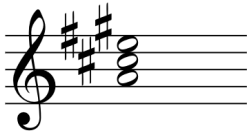
A-durtreklang

Tre toner samtidigt (första, tredje, femte toner av A-durskalen)

A 440Hz 

$C\#$ 554Hz 

E 660Hz 

A-durtreklang 

Att skriva en treklang

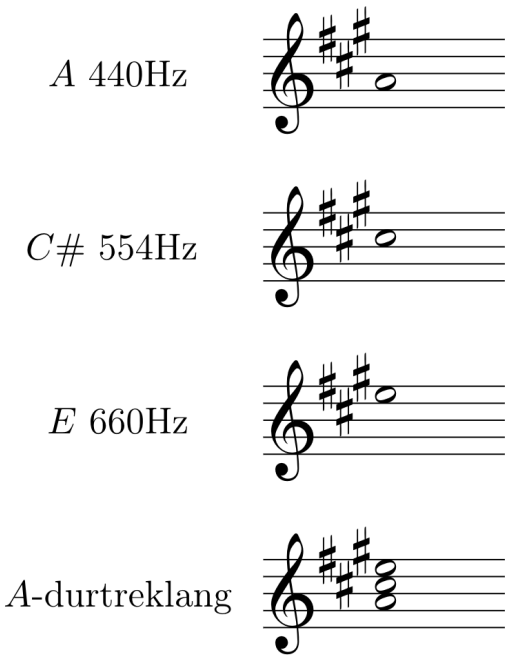
- Adderar tre toner (diskreta sinuskurvor) ihop
- Multiplicerar en sinuskurva med en konstant är det **volymen** som ändras, inte **tonen**

Python

```
#3 toner spelade en i taget och efter hörs en treklang
f = open('ackord.wav','wb')
f.write(y0_a)
f.write(y2_a)
f.write(y4_a)
f.write((1/3)*y0_a + (1/3)*y2_a + (1/3)*y4_a)
f.close()
# Stänger filen
```

A-durtreklang

Tre toner samtidigt (första, tredje, femte toner av A-durskalen)



- Lyssna på den
- Varning: högljud!

Låter fint

```
#A och E spelar samtidigt
f = open('fin.wav','wb')
f.write((1/2)*y0_a + (1/4)*y4_a)
f.close()
```

- Två toner som låter fint när de spelas samtidigt
- Lyssna
- Varning: högljud!

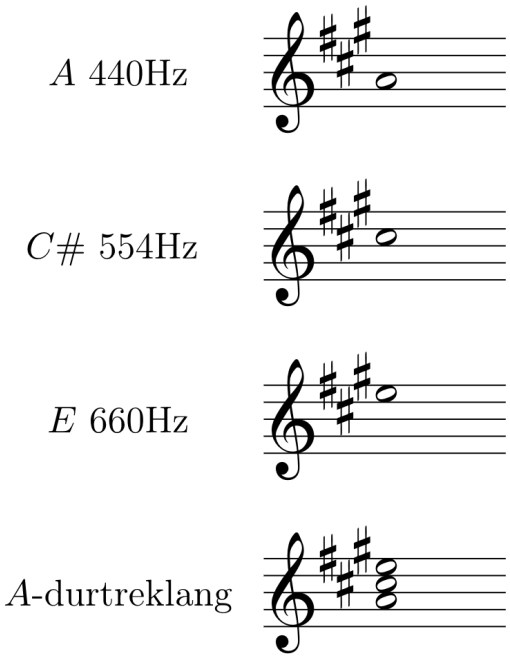
Låter inte så fint...

```
# G# och A spelar samtidigt
f = open('ugly.wav','wb')
f.write((1/2)*y6_a + (1/2)*y7_a)
f.close()
```

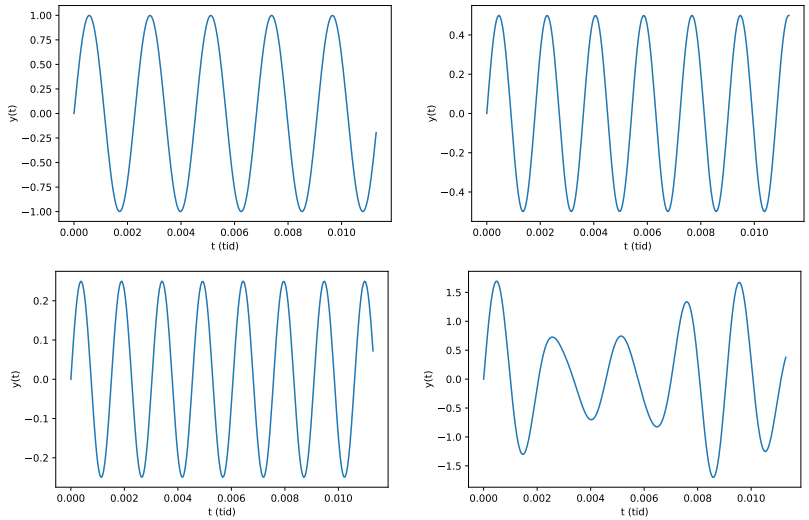
- Två toner som **inte** låter fint när de spelas samtidigt
- Lyssna
- Varning: högljud!

Från första föreläsningen

A-durtreklang: Noter



A-durtreklang: Sinus sinuskurvor

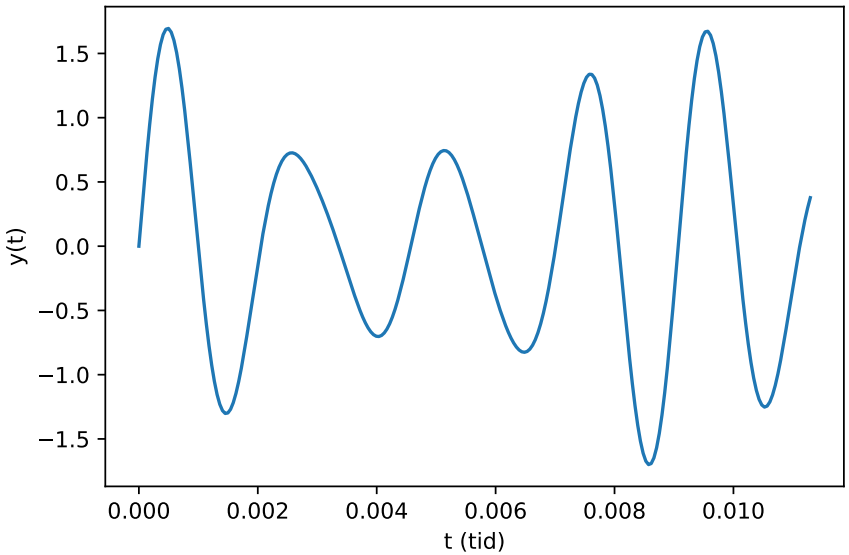


Kom ihåg!

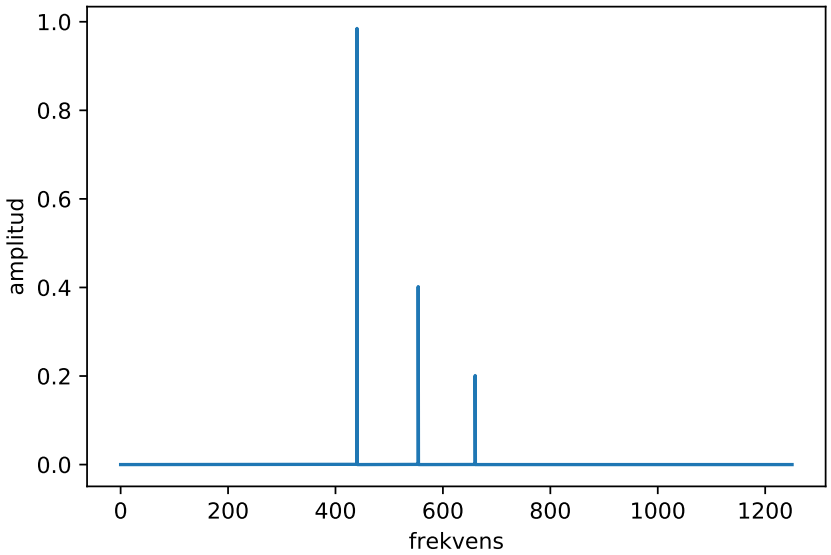
- Amplitud är volymen
- Vilken ton det är som spelas beror på frekvensen

Från första föreläsningen

Tidsdomän:



Frekvensdomän:



- A 440Hz med amplitud 1
- C# 554Hz med amplitud .5
- E 660Hz med amplitud .25

- Och vi ska använda FFT för att gå från tidsdomän till frekvensdomän

Obs!

Notera: För att få en graf med sinuskurvornas amplituder på y-axeln måste vi använda följande vektor till y-axeln i vår plot.

$$\hat{y} = \frac{2}{n}|y|$$

(Redan skrivit i koden.)

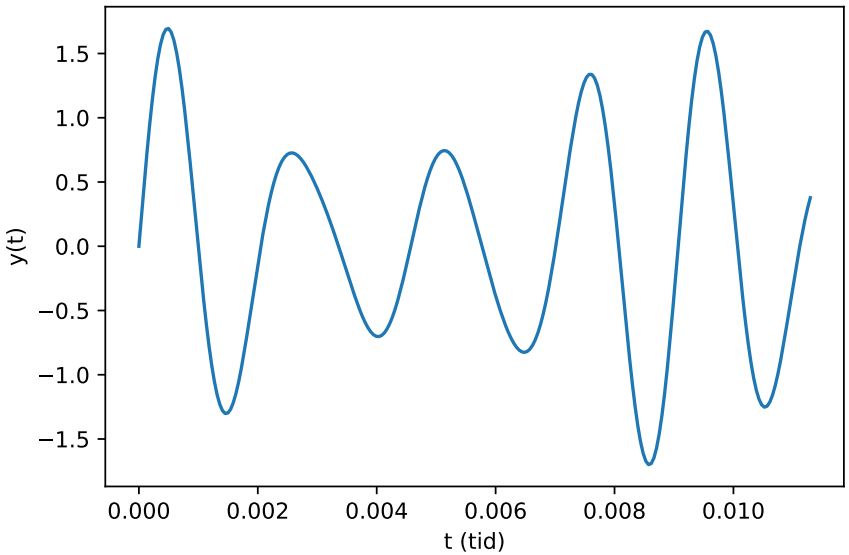
Kod för att skapa FFT graf

```
N = 2**24 #Storlek av vektorn är 2^24 (kan ta mindre N)
T = 1.0/5000
x = np.linspace(0.0,N*T,N)

#A-durtreklang
y = 1*np.sin(2*np.pi*f0_a*x) + (1/2)*np.sin(2*np.pi*f2_a*x) +
(1/4)*np.sin(2*np.pi*f4_a*x)

xf = np.linspace(0.0, 1.0/(2.0*T), N//2)
yf = FFT(y) #Anropa till funktionen vi skrev innan för FFT
fig, ax = plt.subplots() # Skapa plotten
ax.plot(xf, 2.0/N * np.abs(yf[:N//2]))
plt.xlabel('frekvens')
plt.ylabel('amplitud')
```

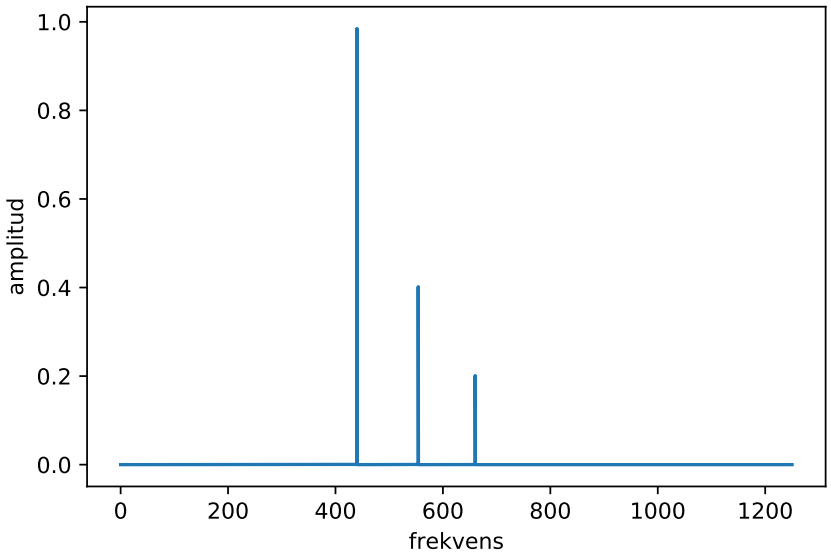
Då har vi FFT

A-durtreklang: A 440Hz, C# 554Hz, E 660Hz

Tidsdomän
Vilka toner??



Frekvensdomän:
Lätt att se vilka



Snabbhet

- Notera hur stort N är $N = 2^{24} = 16777216 \approx 10^7$
- Från Avsnitt 6.2: Den rekursiva algoritmen har komplexiten $\mathcal{O}(n \log n)$
- Gausselimination för att beräkna de trigonometriska koefficienterna (Exempel 4.2.2)
- Komplexiten för Gausselimination är $\mathcal{O}(n^3)$
- Tänk hur många beräkningar vi slipper göra för att vi använder FFT istället för Gausselimination

Tillämpningar

- Vi kan ändra felaktiga ton i en ackord
- Någon spelar ett piano och använder många ackord
- Om de spelade en fel ton skulle bara de som har mycket erfarenhet av musik kunna höra vilken ton som var fel och vad den borde var
- Vi kan använda algoritmen för att se vilken ton som var fel i ackorden
- Kan också laga tonen och sedan använda den inversa transformen för att gå tillbaka till tidsdomänen

En varning

- Hoppas att ni vill prova!
- Bara en varning:
  - Den kan låta hemskt!!
  - Ta ner volymen på datorn
  - Var försiktig med hörlurar
- Det kan lätt bli alldeles för högt ljud!!