# linear_algebra

May 5, 2022

## 1 Lineær algebra med Jupyter Notebook

Her skal vi definere enkele vektoren $v$, for enkelte måtet vi bruker fiksert vektor $v = \begin{bmatrix} 1 \\ 2 \\ c\ 3 \\ 4 \\ 5 \end{bmatrix}$

[1]:
```
v = [1, 2, 3, 4, 5]
```

[1]:
```
5-element Vector{Int64}:
 1
 2
 3
 4
 5
```

Det er bare viktig for språk syntaks, hvordan er vektor eller matrise skevet. Formell det er det samme: vektor.

[2]:
```
mat = [11 22 33 44 55]
```

[2]:
```
1×5 Matrix{Int64}:
 11  22  33  44  55
```

Her skal vi demonstrere typiske operasjoner med vektorer. For eksampel, vi kan bruke funksjon **sum**, og noen andre sammen med VEC og MAT variebler.

[3]:
```
sum(mat)
```

[3]:
```
165
```

vektoren i potens 2

[4]:
```
v.^2
```

[4]:
```
5-element Vector{Int64}:
  1
  4
```

```
 9
16
25
```

Funksjonen **size** sier oss om en størrelse for hvert objekt som vektor eller matrise

[5]:
```
size(v)
```

[5]:
```
(5,)
```

[6]:
```
size(mat)
```

[6]:
```
(1, 5)
```

Det er to neste viktige operasjoner mellom vektorer: vektoriske og skalare multiplikasjoner. Det er viktig å forstå at for skalare multiplikasjonen vi får et tall. Når for vektorisk operasjonen får vi en vektor igjen.
Skalare multipliksjonen er: $v \cdot w = a$, hvor $a$ er et tall.

[7]:
```
using LinearAlgebra # denne linjen gir oss ekstra programming libs med funksjoner
w = [5,4,3,2,1] # en mer vektor
vw # skalare multiplikasjonen
```

[7]:
```
35
```

hva det betyr? Vi kan få det samme resultatet hvis bruker element ved element operasjonen: $v \cdot w = (v_1, ..., v_5) \cdot (w_1, ..., w_5) = v_1 w_1 + ... + v_5 w_5 = \sum_i v_i \cdot w_i$

[8]:
```
w[1]*v[1]+w[2]*v[2]+w[3]*v[3]+w[4]*v[4]+w[5]*v[5]
```

[8]:
```
35
```

Vi kan multiplere vektor ved vektor, vektor ved tall osv. Når kan vi komme nærmere til bildebehandlingen og bruke vektor og matriser som gjenstander for analyse. For dette la oss visualisere litt hva er vektor på geometriske måtet. Vi ser på to spesielle vektorer: $v = \begin{bmatrix} c \\ 4 \\ 2 \end{bmatrix}$ $w = \begin{bmatrix} c \\ -1 \\ 2 \end{bmatrix}$

[9]:
```
v = [4,2]; w = [-1,2];
vw
```

[9]:
```
0
```

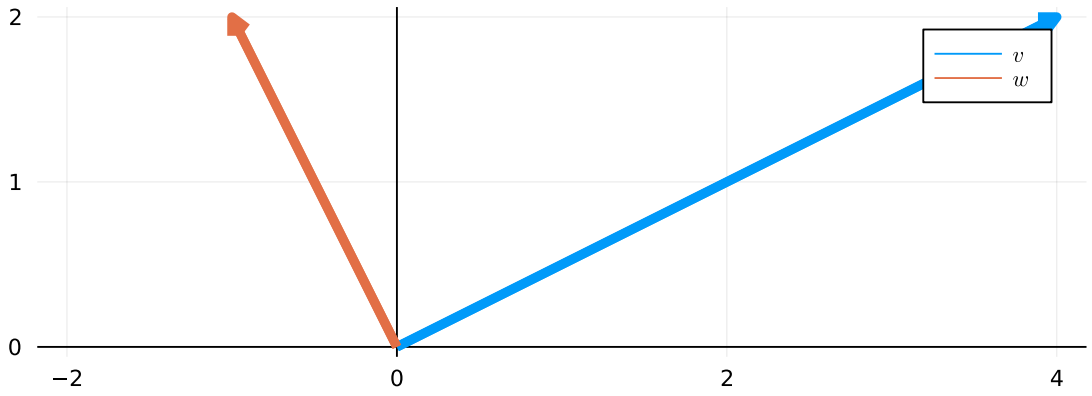
Et resultat av multiplikasjonen er 0!!! Hva kan det bety på geometriske måtet?

[10]:
```
using Plots
using LaTeXStrings

plot([0,4],[0,2],
    label = L"$v$",
```

```
    arrow = :head,
    linewidth = 5)
plot!([0,-1],[0,2],
    framestyle = :zerolines,
    label = L"$w$",
    arrow = :head,
    legend = :best,
    xlim = [-2, 4],
    ylim = [0, 2],
    ytick = [0, 1, 2],
    xtick = [-2,0,2,4],
    aspect_ratio = :equal,
    linewidth = 5)
```

[10]:




Som resultat, dere kan se at vektor har $90^o$ grad mellom hver andre. Hvis bruker vi andre formel for skalar produkt: $v \cdot w = |v| \cdot |w| \cos(\theta)$ da kan vi forstå at cos-funksjon gir oss svaret om vinkelen mellom vektorer. Da kan vi snakke litt om matriser og hva slags matriser kan man tenke på? symmetrisk: $S = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$ diagonal: $D = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$ identitetsmatrise $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

[11]:
```
A = [2 15; 8 45];
S = [1 2; 2 3];
D = [1 0; 0 2];
```

```
IM = [1 0; 0 1];
A*S
```

[11]:
```
2×2 Matrix{Int64}:
 32   49
 98  151
```

[12]:
```
S*A
```

[12]:
```
2×2 Matrix{Int64}:
 18  105
 28  165
```

Man kan se her at $S \cdot A$ er ikke det samme som $A \cdot S$. Dette trekket er kjent som non-kommutative matriser. Hvis $SA - AS = 0$ da det er kommutativ matriser. Det er veldig viktige resultatet fordi det bringer mange viktige konsekvenser for matimatikk.

Andre ting om matriser er eigenverdier, i.e. noen verdier hvilke er uavhengig av matrise formen eller matrise i eigenrom er altid diagonal.

[13]:
```
B = eigvals(A)
```

[13]:
```
2-element Vector{Float64}:
 -0.6298570240273023
 47.62985702402731
```

[14]:
```
C = eigvecs(A)
```

[14]:
```
2×2 Matrix{Float64}:
 -0.984976  -0.312291
  0.17269   -0.949986
```

Man kan forstå det som noen notasjon ned: $Au = \lambda u$, hvor A er matrisen vår, $u$ er eigenvektor og $\lambda$ er eigenverdi. Et viktig resultat er: noen symmetriske matrise kan bli representert som diagonal og eigenvektor. Faktisk betyr det følgende:

[15]:
```
A = [1 2 3; 2 4 5; 3 5 6]
```

[15]:
```
3×3 Matrix{Int64}:
 1  2  3
 2  4  5
 3  5  6
```

[16]:
```
Al = eigvals(A)
```

[16]:
```
3-element Vector{Float64}:
 -0.5157294715892576
  0.17091518882717918
```

```
11.344814282762076
```

[17]:
```
Au = eigvecs(A)
```

[17]:
```
3×3 Matrix{Float64}:
 -0.736976   0.591009  -0.327985
 -0.327985  -0.736976  -0.591009
  0.591009   0.327985  -0.736976
```

[18]:
```
Au'*A*Au
```

[18]:
```
3×3 Matrix{Float64}:
 -0.515729     -1.38778e-17  -6.10623e-16
 -5.20417e-16   0.170915      1.16573e-15
  0.0           8.88178e-16  11.3448
```

Au' betyr transponert matrise eller når vi skifter rekken og kolonnen.

[19]:
```
Au'
```

[19]:
```
3×3 adjoint(::Matrix{Float64}) with eltype Float64:
 -0.736976  -0.327985   0.591009
  0.591009  -0.736976   0.327985
 -0.327985  -0.591009  -0.736976
```

## 2 Hvorfor er viktig å jobbe med matriser og vektorer?

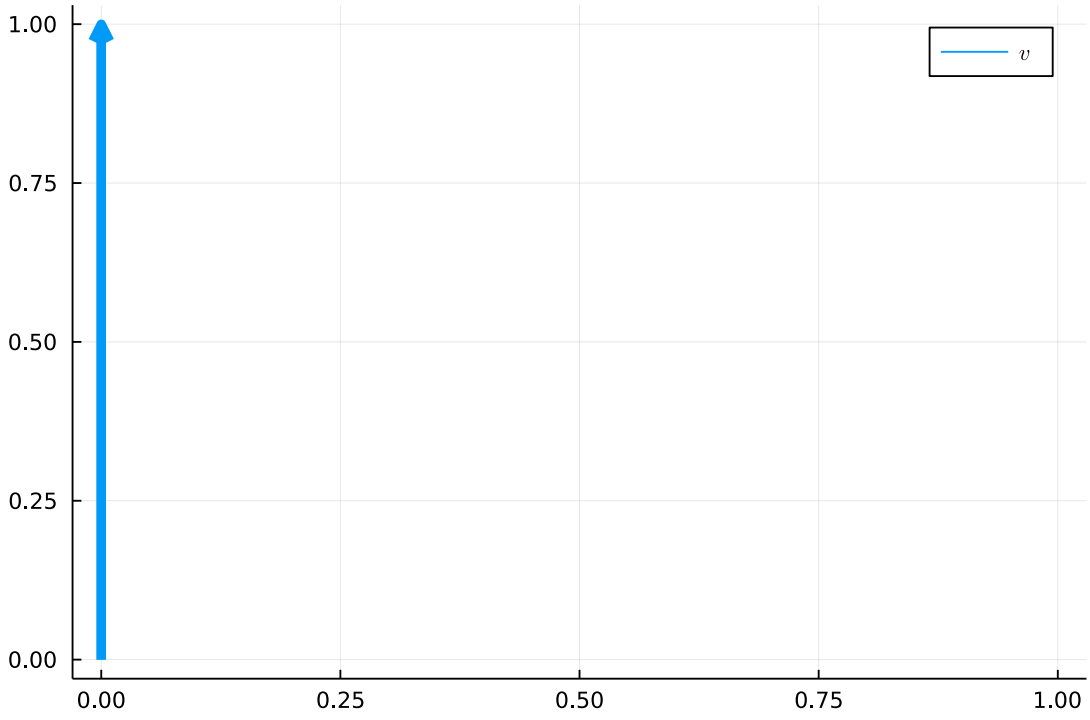
Vi kan manipulere med vectorer og matriser når vi bruker spesielle matriser. For eksempel, rotasjon matrise (også vi vet den som orthogonal matrise eller $R \cdot R^T = R^T \cdot R = I$ identitetsmatrisen.)
Rotasjon matrise for 2D tillfelet: $R = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}$

[20]:
```
θ = π/4
R = [cos(θ) sin(θ); -sin(θ) cos(θ)]
```

[20]:
```
2×2 Matrix{Float64}:
  0.707107  0.707107
 -0.707107  0.707107
```

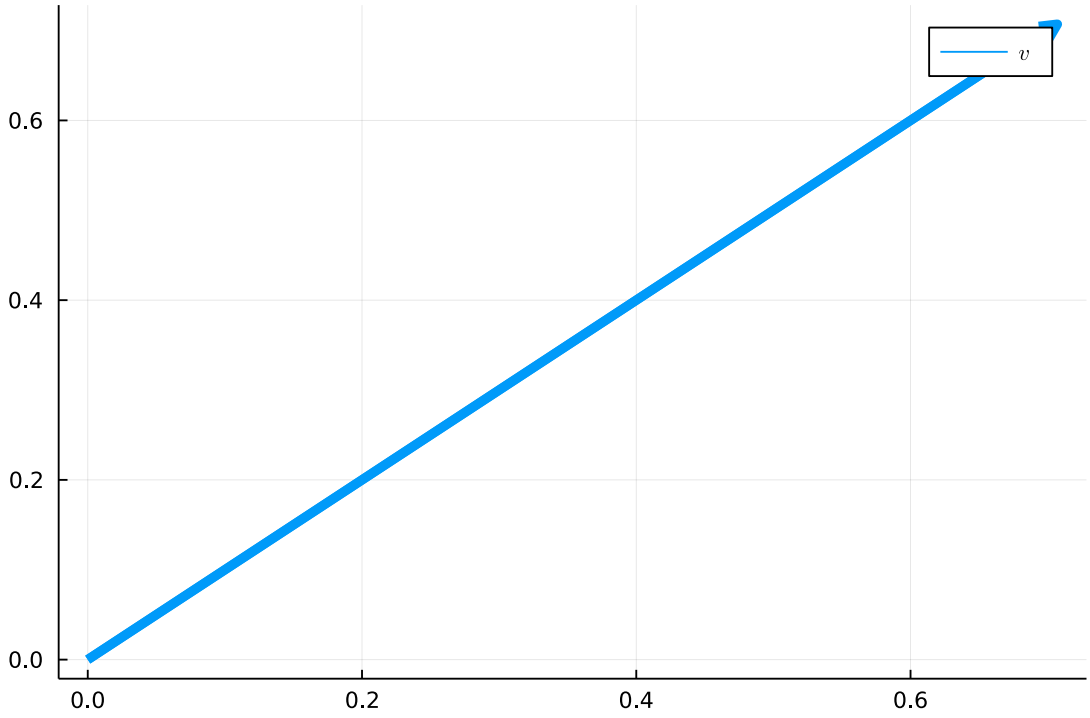
[21]:
```
v1 = [0; 1]
plot([0,v1[1]],[0,v1[2]],
    label = L"$v$",
    arrow = :head,
    linewidth = 5)
```

[21]:



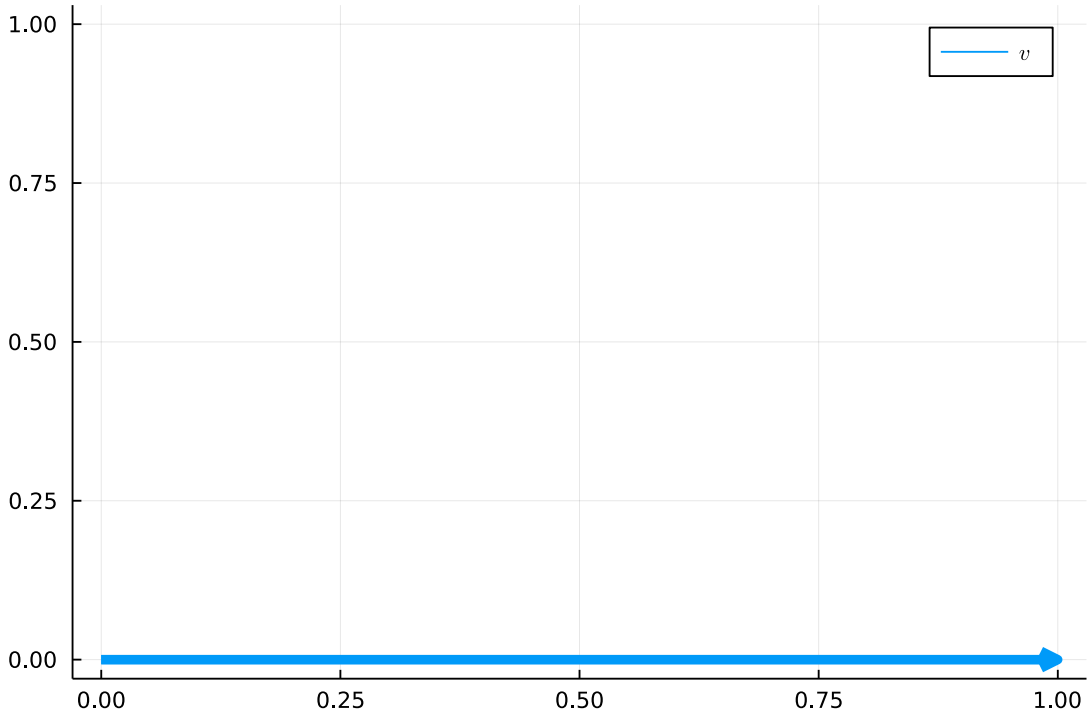

[22]:
```
v2 = R*v1
plot([0,v2[1]],[0,v2[2]],
    label = L"$v$",
    arrow = :head,
    linewidth = 5)
```

[22]:




[23]:
```
v3 = round.(R*v2)
plot([0,v3[1]],[0,v3[2]],
    label = L"$v$",
    arrow = :head,
    linewidth = 5)
```

[23]:



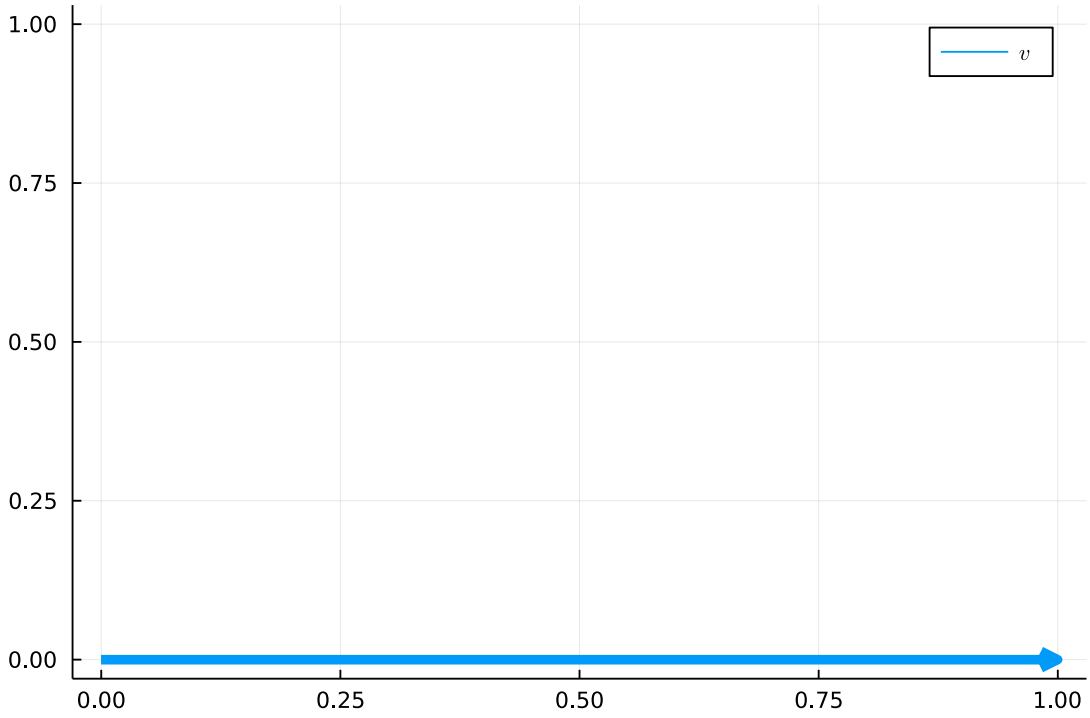

## 3 Manipulasjoner kan være sekvensielle

for eksampel, vi brukte to rotasjoner med vinkel $\theta = 45^o$ eller $\pi/4$ i stedet av to 45 grader rotasjoner vi kan bruke en rotasjon som 90 grader eller $R_{ny} = R(\theta) \cdot R(\theta)$ og R(45)R(45) = R(90)

## 4 sannelig

[24]:
```
θ = π/2
R90 = [cos(θ) sin(θ); -sin(θ) cos(θ)]
v4 = round.(R90*v1)
plot([0,v4[1]],[0,v4[2]],
    label = L"$v$",
    arrow = :head,
    linewidth = 5)
```

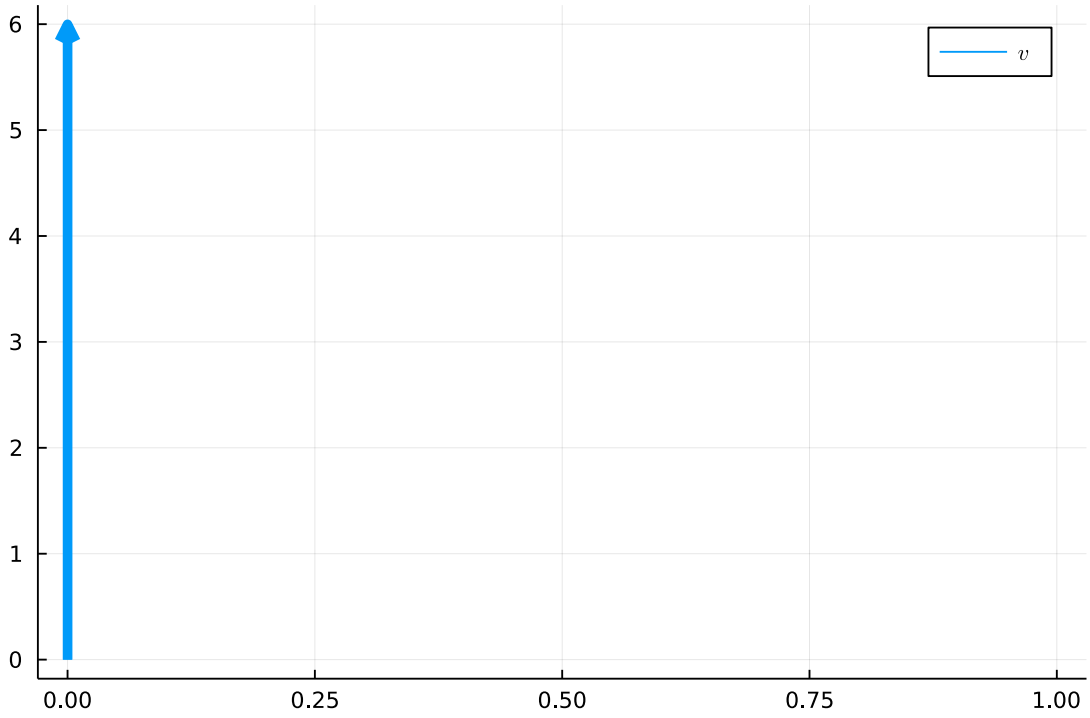
[24]:




vi kan også gjøre andre transformasjoner med vektor som et skift, en skalering etc

[25]:
```
scal=[1 0; 0 6]
v5 = scal*v1
plot([0,v5[1]],[0,v5[2]],
    label = L"$v$",
    arrow = :head,
    linewidth = 5)
```

[25]:



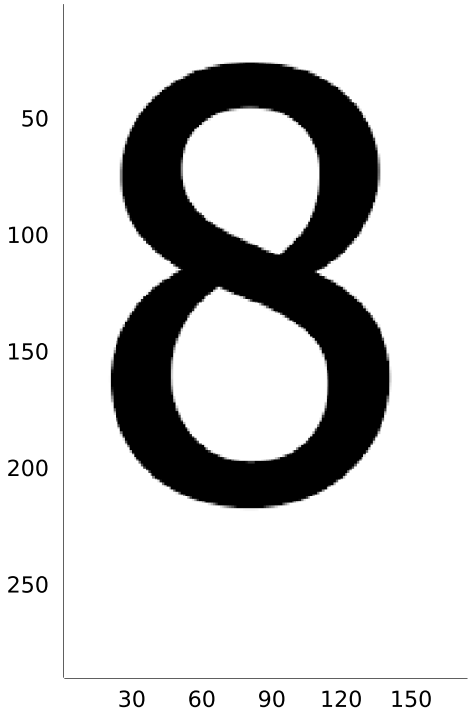

[26]:
```
using Images
png = "8.png"
img = load(png)
```

[26]:

Vi lastet opp bildet med tall 8. Det betyr at vi har matrisen med 0 og 1.

[27]:
```
plot(img)
```

[27]:




Vi kan også finne hva stor er matrisen

[28]:
```
size(img)
```

[28]: (290, 174)

## 5 Transponering er det samme som rotasjon for 90 grader

[29]:
```
A = img'
```

[29]:

## 6 Det er mulig å jobbe med bildet som en gjenstand og endrer sin oppløsning

[30]:
```
A = imresize(img,(30,30))
```

[30]:

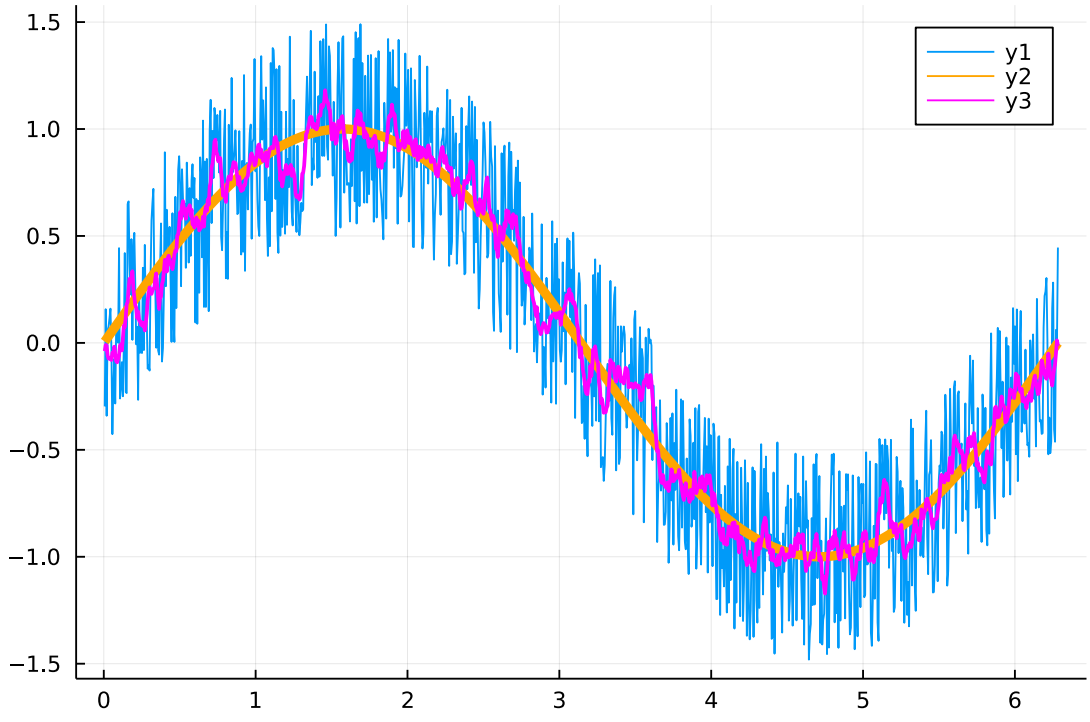
[31]:
```
size(A)
```

[31]: (30, 30)

## 7 Oppløsningen er et eksampel av smoothing teknikk.

[32]:
```
using NaNStatistics
N = 1000
x = 1:N
θ = x*2π/N
tr = sin.(θ)
y = sin.(θ) + rand(N,1) .-0.5
plot(θ,y)
plot!(θ,tr, linewidth = 5, color = :orange)
z = movmean(y,N/100)
plot!(θ,z,linewidth = 2, color = :magenta)
```

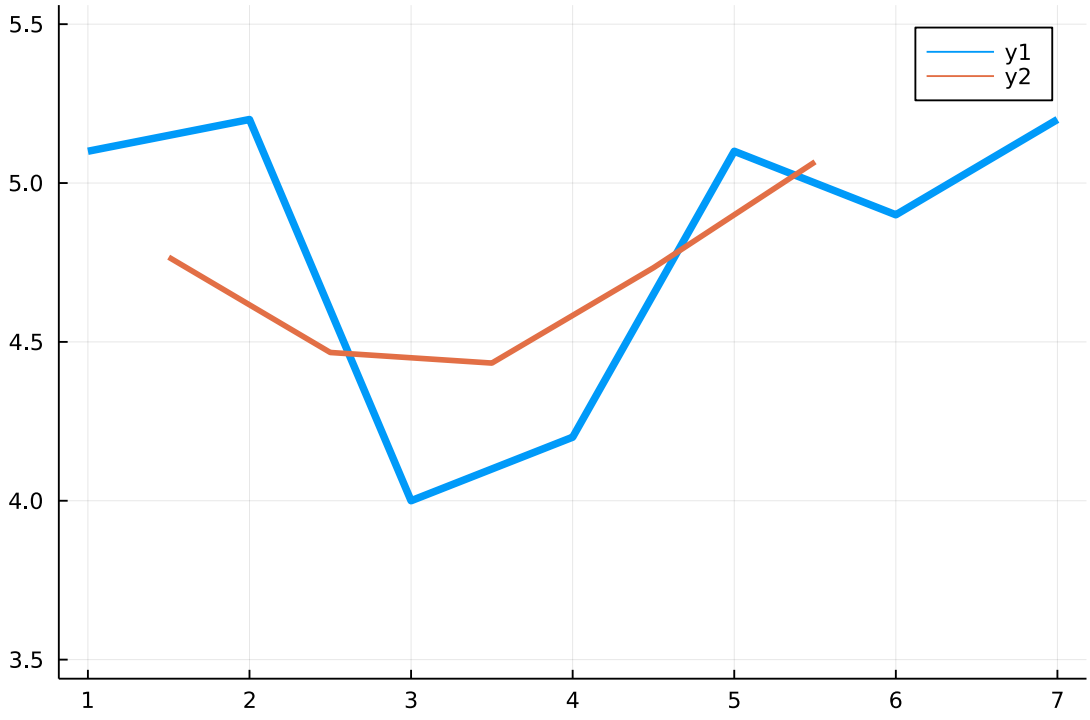
[32]:




## 8 Smoothing teknikk er ganske enkel

[33]:
```
a = [5.1, 5.2, 4, 4.2, 5.1, 4.9, 5.2]
plot(a, linewidth = 4)
function smoothing(a)
    N = length(a);
    b = zeros(N-2,1);
    for i = 1:(N-2)
        b[i] =  (a[i] + a[i+1]+a[i+2])/3;
    end
    return b
end
b = smoothing(a)
plot!([1.5, 2.5, 3.5, 4.5, 5.5],b,linewidth = 3, ylim = [3.5,5.5])
```

[33]:




## 9 Bilde skalering kan være noe forbedring. Men denne operasjonen har begrensning også.

[34]:
```
imresize(img, ratio=5)
```

[34]:

Mange matematiske operasjoner er allerede implementert i forskjellige pakker. Se på fslmaths, for eksampel.

[ ]: