

**ELE130 Anvendt fysikk og matematikk i
robotprogrammering**

ELE100 Elektroteknikk 1

Øving 1

**Introduksjon til L^AT_EX/Overleaf og
beregningsorientert programmering i Matlab.** ¹

Svar:

Levert av Lasse Lorentzen

¹Tilhørende filer: `oving1_skallfil.m` og L^AT_EX-mal i `oving1.zip`

Om øvingen og innleveringen

- For å bli kjent med Matlab på en effektiv måte, er alle oppgavene i denne øvingen svært nyttige og de kan fungere som et oppslagsverk for deg.
- For å tilrettelegge for at du skal gjøre flest mulig oppgaver, så skal du laste ned skallfilen `oving1_skallfil.m` som inneholder uferdig kode for alle oppgavene i denne øvingen. Mange av oppgavene virker kanskje omfattende, men svært mye av "infrastruktur-koden" er allerede gitt.

For å kjøre ferdig kode i et skript fullt av uferdig kode så bør du benytte (en av) teknikkene beskrevet i kapittel 2.2 i kompendiet.

- Hver celle starter med `clear; close all; clc` slik at du unngå å gjøre beregninger på gamle verdier av variable.
- **Hensikten med øvingen er du skal bli kjent med programmeringsmiljøet i Matlab og et utvalg av de viktigste funksjonene.**
- Dersom du ikke er en erfaren programmerer, så anbefales det at du prøver deg på alle oppgavene slik at du får mengdetrenings og at du forhåpentligvis opplever mestringsfølelse (som er helt avgjørende for å trives med programmering).
- **For å få øvingen godkjent må du gjøre minst følgende oppgaver:**
`a), b), c), f), g), i),`
- For å gjøre bl.a. oppgave g) så må du ha installert Signal processing toolbox. For å finne ut om du har denne tilgjengelig, skriv `>> ver` i *Command Window* og sjekk listen over installerte toolbox'er.
- Oppgavene `d), e), h), j)` er også lærerike og nyttige, men frivillige. Dette er markert i toppteksten.
- **For å ta eksamen i emnet ELE130 så må denne øvingen være godkjent. Husk at øvingene må leveres individuelt.**
- I øvingen skal du også lære deg grunnleggende L^AT_EX og Overleaf funksjonalitet. Du må levere øvingen ved å bruke L^AT_EX-malen gitt i filen `oving1.zip`.
- Basis for denne øvingen er kapittel 1, 2 og 3 samt vedlegg B og C i kompendiet.

Om L^AT_EX og Overleaf, og litt Matlab-tips

- Gjør først det som står i vedlegg B i kompendiet hvor du registrerer deg som bruker av Overleaf.
- Last deretter opp i OverLeaf .zip-filen **oving1.zip** som blant annet inneholder malen **ELE130 innlevering oving1.tex** som er et L^AT_EX-skall med en ferdig struktur som du skal bruke for å levere denne øvingen.
- Kompiler **ELE130 innlevering oving1.tex** og forsikre deg om at det ikke er noen feil i kompileringen. Som du ser i OverLeaf så er det en mappe som heter **figurer** hvor du kan laste opp figurer som du lager. Ser du i linje 16 i denne .tex-filen så er figurmappen lagt inn i søkerstien til LaTeX ved hjelp av **figurepath**-pakken.
- Dersom du får uforståelige feilmeldinger i L^AT_EX så kan du flytte rundt på kommandoen **\end{document}** inn i teksten og på den måte lokalisere feilen.
- I filen **startup.m** har vi definert L^AT_EX som tekst-tolker i Matlab. Husk derfor på at “æ”, “ø” og “å” må skrives som **{\ae}**, **{\o}** og **{\aa}** i title-, legend- og text-kommandoene.
- For å avbryte et Matlabprogram som kjører trykker du **ctrl-C/cmd-C**.
- For å se verdien av en variable så kan du skrive variabelnavnet i **Command Window**. Dette er en veldig nyttig offline feilsøkingsteknikk.
- For å få hjelp til Matlab-funksjoner, bruk **help**, **lookfor** og **doc**-kommandoene.

Hvordan gjøre innleveringen?

- I filen **ELE130 innlevering oving1.tex** finner du følgende seksjon:

```
\section*{Mal for {\LaTeX}-kode til hver deloppgave}
% ****
% Mal som du kan bruke for de deloppgavene du gjør.
% Marker og kopier LaTeX-koden fra og med
% \begin{solution}
%
% til og med
%
% \end{solution}
% og kopier den inn rett etter \include{oving1_x} for de
% deloppgavene du gjør.
% Pass på å oppdater/endre figur-, kode- og ligningsreferansene.
% ****
```

LATEX-koden innenfor området mellom `\begin{solution}` og `\end{solution}` er vist i grått felt på neste side og fungerer som en mal som du kopierer til de deloppgavene du skal (og eventuelt vil) gjøre.

- Generelt for mange av deloppgavene i denne øvingen er at du skal skrive ut svaret i **Command Window** med `disp`-kommandoen, se kapittel 2.3 i kompendiet, og deretter ta en skjermdump av resultatet og levere inn. På en Mac kan du bruke `Shift+Cmd+4` for å velge et utsnitt til skjermdump, og tilsvarende på PC (win11) er `Windows+Shift+S`.

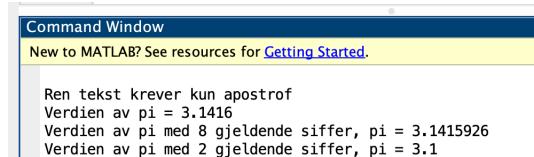
I tillegg til en skjermdump av resultatet i **Command Window** skal du i noen deloppgaver ta med Matlab-koden din og figurer som du plotter. Benytt da funksjonen `SaveMyFigure.m` til å lage pdf-fil av figuren(e). Figurene i øvingen er laget med denne funksjonen, men er etterpå blitt redigert til png-formatet.

- Et eksempel på inkludering av kode og tilhørende skjermdump er vist under.

Kode 1: Kode som viser bruk av `disp`-kommandoen.

```
disp('Ren tekst krever kun apostrof')
% for å bruke både tekst og tall trengs hakeparentes og num2str()
a = 3.14159256;
disp(['Verdien av pi = ',num2str(a)])
disp(['Verdien av pi med 8 gjeldende siffer, pi = ',num2str(a,8)])
disp(['Verdien av pi med 2 gjeldende siffer, pi = ',num2str(a,2)])
```

som gir følgende resultat i **Command Window**



Mal for L^AT_EX-kode til hver deloppgave

Svar:

- Forslag til løsning er gitt i kodeutdrag ??.

Kode 2: Kode for oppgaven.

```
1 Kopier
2 inn
3 koden her
```

- Skjermdump av *Command Window* er gitt i figur ??.

Alternativ tekst: Koden produserer figuren vist i figur ??.



Figur 1: Skjermdump av *Command Window*.

- Svaret er gitt i ligning (1).

$$y = 4 \tag{1}$$

Beregningsorientert programmering i Matlab

- a) I denne oppgaven skal du bli kjent med et par Matlab-funksjoner. Følgende tekst står i skallfilen

Kode 3: Heading i oppgave a).

```
%% -----  
% a)  
%  
% Innhold:  
% - Vektor med tilfeldige tall  
% - Bruk av diff-funksjonen  
% - Plotte ulike elementer  
% - Finne maksimum og mininimum  
%  
% Eksempler på nyttige Matlab-funksjoner  
% - rng  
% - randn  
% - num2str  
% - plot  
% - max  
% - min  
% - diff  
% - subplot  
% - title med bruk av num2str  
% - title som deles på 2 linjer  
% -----
```

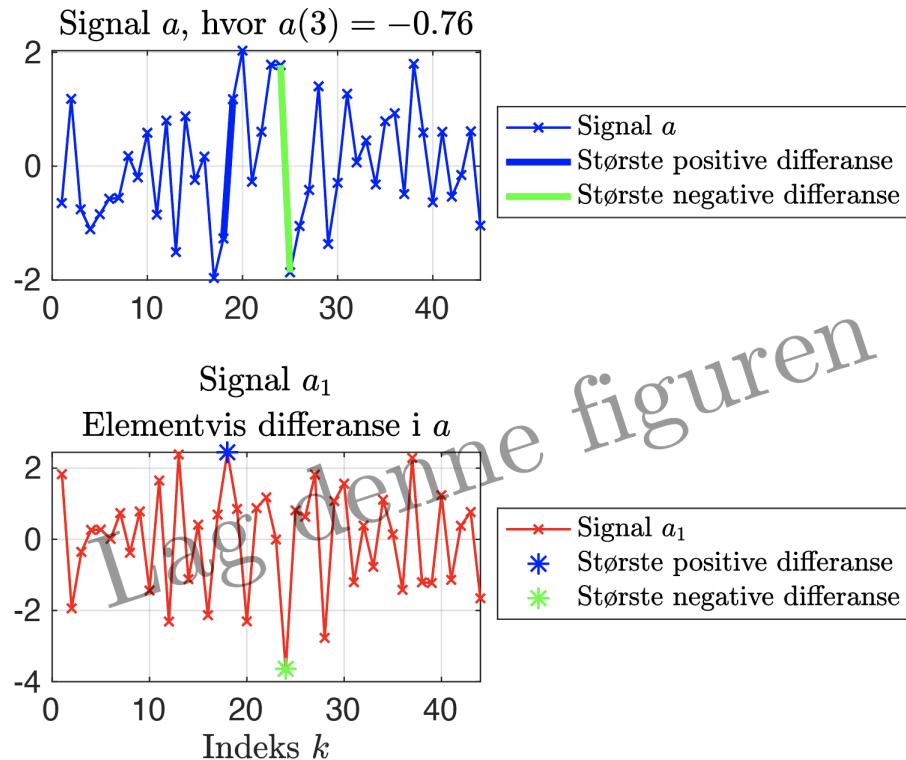
For å få hjelp til hvordan du bruker funksjonene, benytt `help` og `doc`-funksjonene.

Ta utgangspunkt i `oving1_skallfil.m` og fyll inn det som mangler.

Som du ser tar koden utgangspunkt i variabelen `a` som inneholder tilfeldige tall laget med `randn`-funksjonen.

For at de tilfeldige tallene skal være like for hver gang du kjører koden så bestemmer du dette med å gi en `seed`-verdi til `rng`-funksjonen. Fjerner du `rng`-funksjonen vil du få nye tall for hver kjøring. Endre gjerne på `seed`-verdien for å studere effekten.

Vis at du får resultatene vist i figur 2 dersom `seed = 1`.



Figur 2: Resultat med 2 subplot over hverandre, hvor `seed = 1`.

Svar:

- Forslag til løsning er gitt i kodeutdrag 4.

Kode 4: Kode for oppgaven.

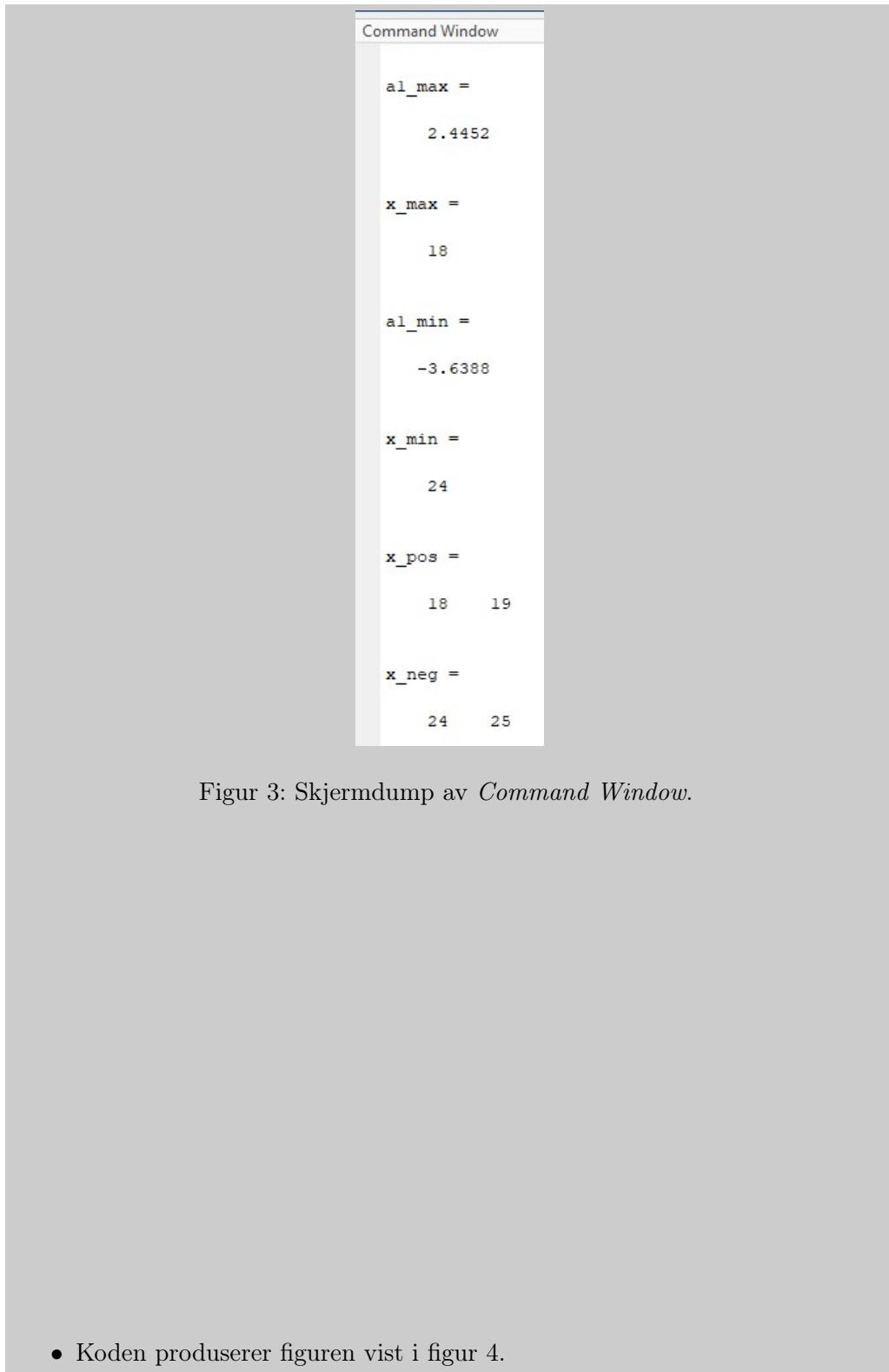
```

1      clear; close all; clc
2
3  seed = 1
4  rng(seed)
5  a = randn(1,45)
6
7  figure
8  % First subplot
9  subplot(2,1,1)
10
11 % Plot a with blue color, solid line, and 'x' markers
12 plot(a, 'b-x')
13 grid
14 hold on

```

```
15
16 % Title
17 title(sprintf('Signal $a$, hvor $a$(3) = %.2f', a(3)))
18
19 % Second subplot
20 subplot(2,1,2)
21
22 % Diff function on a
23 a1 = diff(a)
24
25 % Plot a1 with red color, solid line, and 'x' markers
26 plot(a1, 'r-x')
27 grid
28 hold on
29
30 % X-label and title
31 xlabel('Indeks $k$')
32 title({'Signal $a_1$', 'Elementvis differanse i $a$'})
33
34 % Max difference
35 [a1_max, x_max] = max(a1)
36 plot(x_max, a1_max, 'b*', 'MarkerSize', 10)
37
38 % Min difference
39 [a1_min, x_min] = min(a1)
40 plot(x_min, a1_min, 'g*', 'MarkerSize', 10)
41
42 % Legend for second subplot
43 legend('Signal $a_1$', 'St{\o}rste positive differanse', ...
        'St{\o}rste negative differanse', ...
        'Location','eastoutside')
44
45 % Return to the first subplot
46 subplot(2,1,1)
47
48 % Line for max positive difference
49 x_pos = [x_max, x_max+1]
50 plot(x_pos, a(x_pos), 'b', 'LineWidth', 3)
51
52 % Line for max negative difference
53 x_neg = [x_min, x_min+1]
54 plot(x_neg, a(x_neg), 'g', 'LineWidth', 3)
55
56 % Legend for first subplot
57 legend('Signal $a$', 'St{\o}rste positive differanse', ...
        'St{\o}rste negative differanse', ...
        'Location','eastoutside')
```

- Skjermdump av *Command Window* er gitt i figur 3.

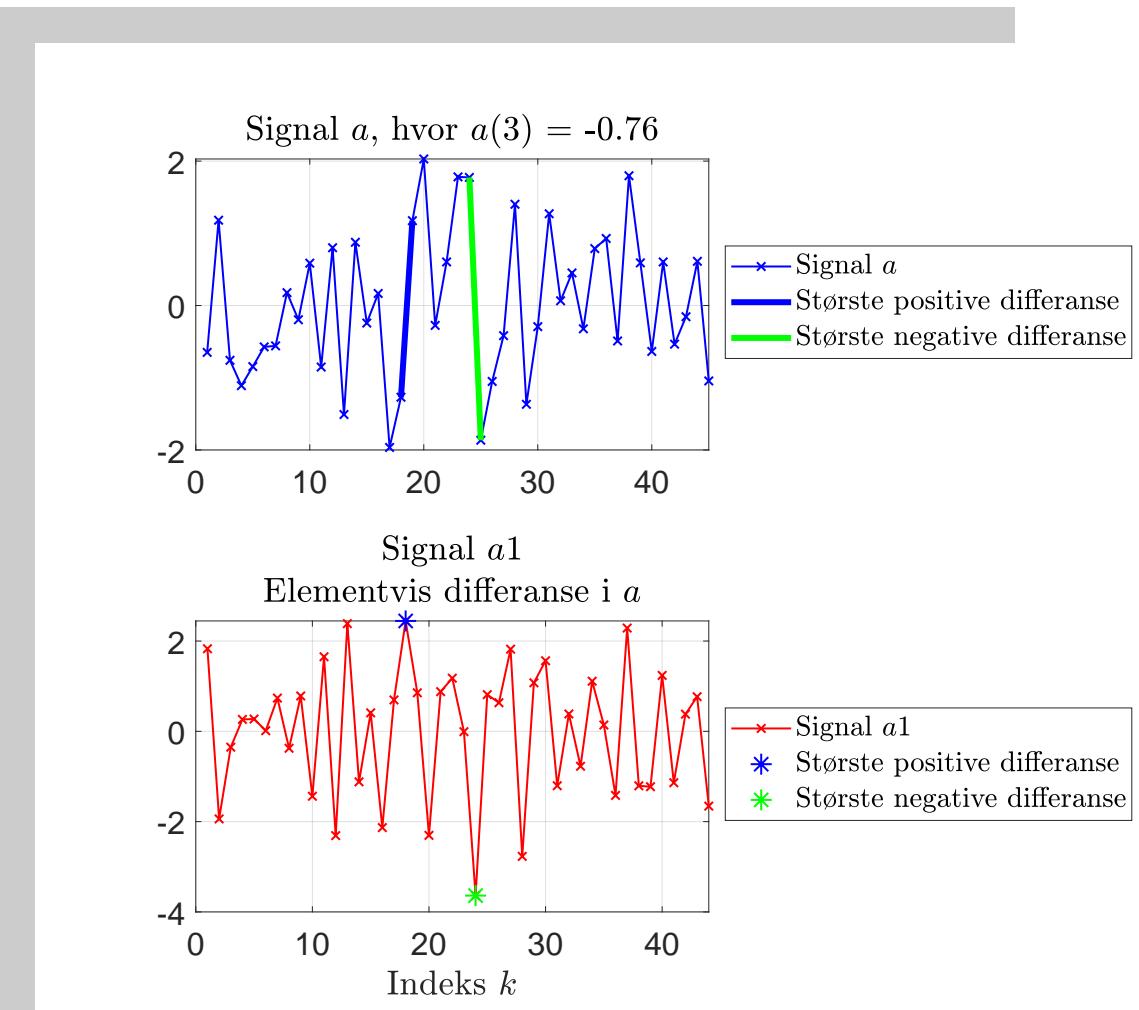


The screenshot shows the MATLAB Command Window with the following variable assignments:

```
al_max =  
2.4452  
  
x_max =  
18  
  
al_min =  
-3.6388  
  
x_min =  
24  
  
x_pos =  
18 19  
  
x_neg =  
24 25
```

Figur 3: Skjermdump av *Command Window*.

- Koden produserer figuren vist i figur 4.



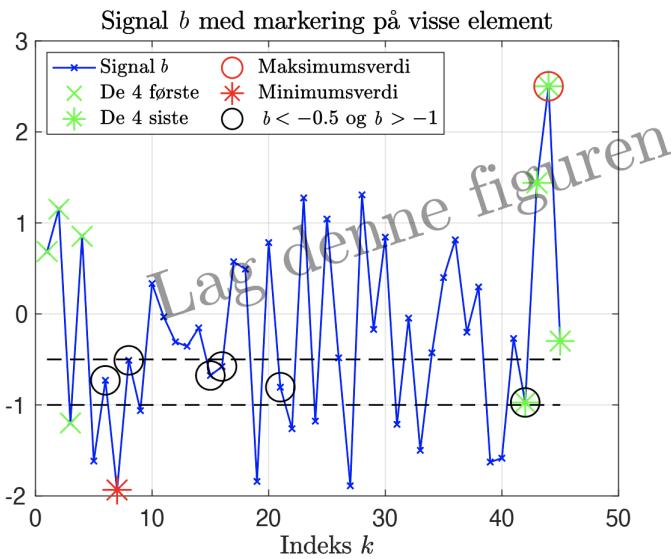
Figur 4: Signal a og elementvis differanse for a, a1

- b) I denne oppgaven skal du bli kjent med endel flere Matlab-funksjoner. Følgende tekst står i skallfilen

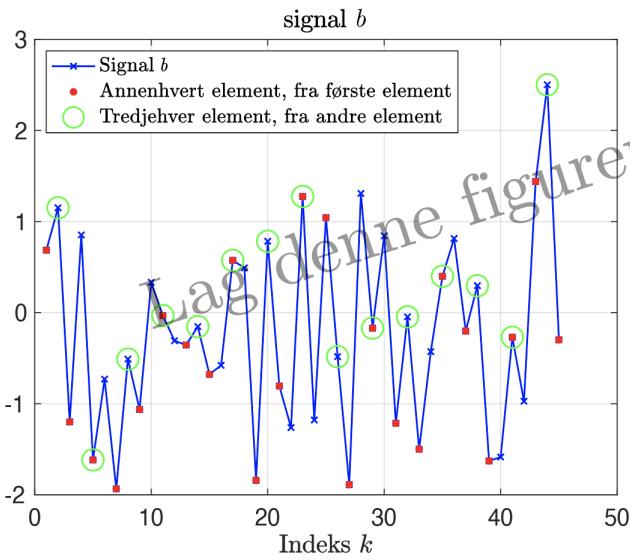
Kode 5: Heading i oppgave b).

```
%% -----
% b)
%
% Innhold:
% - Matrise og vektor med tilfeldige tall
% - Utrekk av forskjellige elementer
% - Bruk av find-funksjonen til å finne utvalgte verdier
% - LaTeX-tips
%
% Eksempler på nyttige Matlab-funksjoner
% - length, numel og size
% - disp og sprintf
% - for-løkker og if-setninger
% - plot, grid, hold, xlabel
% - ones
% - find
% - title med LaTeX-kode for ø og å, {\o} og {\aa}
% - legend med LaTeX-kode for ø og å, {\o} og {\aa}
% - legend med styring av plassering
%
```

Ta utgangspunkt i `oving1_skallfil.m` og fyll inn det som mangler. Som du ser tar koden utgangspunkt i matrisen `B_mat` som inneholder tilfeldige tall laget med `randn`-funksjonen. Vis at du får resultatene vist i figurene 5 - 8, hvor `seed = 3`.



Figur 5: Første figur, hvor `seed = 3`.



Figur 6: Andre figur, hvor seed = 3.

```
s > tormod > Dropbox > faglig > fag > ELE130_Anvendt_matematikk_og_fysikk_i_robotprogrammering
Command Window
New to MATLAB? See resources for Getting Started.
length gir største dimensjon en vei: 45 elementer
numel gir totalt antall: 135 elementer
size gir dimensjon [3 x 45]

Første element = 0.68584
Siste element = -0.29811
4 første element = 0.68584      1.1512     -1.1992      0.85343
4 siste element = -0.97261     1.4397     2.5015     -0.29811

Ved indeks k = 44 er største verdi a = 2.5015
Ved indeks i = 7 er minste verdi a = -1.9337
Ved indeks i = [6   8   15  16  21  42]
er verdiene av a = [-0.73    -0.51    -0.677   -0.578   -0.805   -0.973]

Ved indeks i = 6 er verdien av a = -0.73
Ved indeks i = 8 er verdien av a = -0.51
Ved indeks i = 15 er verdien av a = -0.677
Ved indeks i = 16 er verdien av a = -0.578
Ved indeks i = 21 er verdien av a = -0.805
Ved indeks i = 42 er verdien av a = -0.973
```

Figur 7: Skjermdump av Command Window.

Svar:

- Forslag til løsning er gitt i kodeutdrag 6.

Kode 6: Kode for oppgaven.

```
1      clear; close all; clc
2
3 %-----
4 % Spesifiserer styrte tilfeldige tall med seed
5 % Lager matrisen B_mat
6 %-----
7 seed = 3;
8 rng(seed)
9 B_mat = randn(3,45);
10
11 % Forskjellige kommandoer for å finne ulike størrelser ...
12 % fra matrisen B_mat.
12 k = length(B_mat);
13 tot_elements = numel(B_mat);
14 [len,bre] = size(B_mat);
15
16 % Alternativ bruk av size
17 len = size(B_mat, 1); % size kan gi kun lengde
18 bre = size(B_mat, 2); % size kan gi kun bredde
19
20 disp(['length gir største dimensjon en vei: ', k , ' ...
21 % elementer'])
21 disp(['numel gir totalt antall: ',num2str(tot_elements), ' ...
22 % elementer'])
22 str = sprintf(['size gir dimensjon ',num2str(len), ' x ...
23 % num2str(bre), ']\n']);
23 disp(str)
24
25 %-----
26 % Lager vektor b som rad 2, alle kolonner fra B_mat
27 %-----
28 b = B_mat(2, :);
29
30 % Plot de tilfeldige tallene b med blå strek og 'x' som ...
31 % markør
31 % Lager først indeksvektor
32 x = 1:length(b);
33
34 figure(1)
35 set(1, 'position', [600 400 700 400])
36 plot(x, b, 'b-x')
37
38 % Siden du skal plotta flere ting, benytter vi "hold on".
39 % Du kan også bruke bare "hold", men en ny "hold" slår av ...
40 % holdingen
40 hold on
```

```
41
42 % Kan også bruke "grid on", for å låse grid på. Ny "grid" ...
43 % slår av.
44 grid
45 title('Signal $b$ med markering p{\aa} visse element')
46 xlabel('Indeks $k$')
47
48 %-----
49 % Uttrekk og presentasjon/plotting av elementer fra ...
50 %-----vektor b
51
52 % Første element av b
53 b1 = b(1);
54 disp(['Første element = ',num2str(b1)])
55
56
57 % Siste element av b
58 b2 = b(end);
59 disp(['Siste element = ',num2str(b2)])
60
61
62 % De 4 første elementene
63 b3 = b(1:4);
64 disp(['4 første element = ',num2str(b3)])
65 % Lager tilhørende indeksvektor for plotting
66 x3 = 1:4;
67 % Plot disse med en relativt stor 'x' som markør i grønn ...
68 % farge
68 plot(x3,b3, 'gx', 'MarkerSize',10)
69
70
71 % De 4 siste elementene
72 b4 = b(end-3:end);
73 disp(['4 siste element = ',num2str(b4)])
74 disp(' ') % alternativ til \n i sprintf
75 % Lager tilhørende indeksvektor for plotting, bruk indeks k
76 x4 = (k-3):k;
77 % Plot disse med en relativt stor '*' som markør i grønn ...
78 % farge
78 plot(x4,b4, 'g*' , 'MarkerSize',10)
79
80
81 % max-funksjonen finner indeks x5 og største verdi av b ...
82 % som vi kaller b5
83 [b5 , x5] = max(b);
84 disp(['Ved indeks k = ',num2str(x5),...
84 ' er største verdi b = ',num2str(b5)])
85 % Plot inn største verdi med en relativt stor 'o' som ...
86 % markør i rød farge
86 plot(x5,b5, 'ro' , 'MarkerSize',10)
87
```

```

88 % min-funksjonen finner indeks x6 og minste verdi av b, b6
89 [b6, x6] = min(b);
90 disp(['Ved indeks i = ', num2str(x6), ...
91       ' er minste verdi b = ', num2str(b6)])
92 % Plot inn minste verdi med en relativt stor '*' som ...
93 % markør i rød farge
94 plot(x6,b6, 'r*', 'MarkerSize',10)
95
96
97 % find-funksjonen gir indeksene x7 til alle verdier ...
98 % mellom b_min og b_max
99 b_max = -0.5;
100 b_min = -1;
101 x7 = find(b >= b_min & b < b_max);
102 b7 = b(x7); % plukker ut tilhørende element
103 disp(['Ved indeks i = [', num2str(x7), ']'])
104 disp(['er verdiene av b = [', num2str(b7,3), ']'])
105 disp(' ')
106 % Plot verdiene mellom b_min og b_max rett plass
107 % Bruk en relativt stor 'o' som markør i svart farge
108 plot(x7,b7, 'ko', 'MarkerSize',10)
109
110 % Indikerer max- og min-verdiene med en vannrett svart ...
111 % stiplet strek
112 plot(b_min*ones(1, k), 'k--')
113 plot(b_max*ones(1, k), 'k--')
114 % sjekk oppgaveteksten hvordan legendene skal se ut
115 legend('Signal $b$', ...
116       'De 4 f{\o}rste', ...
117       'De 4 siste', ...
118       'Maksimumsverdi', ...
119       'Minimumsverdi', ...
120       ['$b<', num2str(b_max), '$ og $b>', num2str(b_min), ...
121        '$'], ...
122       'Location','northwest','numcolumns',2);
123
124 % Alternativ måte å finne verdiene mellom b_min og b_max
125 for i = 1:length(b)
126     if b(i) >= b_min && b(i) < b_max
127         disp(['Ved indeks i = ', num2str(i), ...
128               ' er verdien av b = ', num2str(b(i), 3)])
129     end
130 end
131
132 %-----
133 % Plot de tilfeldige tallene b i ny figur, figur 2
134 %-----
135 %
136 figure(2)

```

```
137 set(gcf,'position',[400 100 700 400])
138
139 % Plasserer figuren slik at den ikke kommer rett over
140 % den første. Flytt figuren der du vil og bruk kommandoen
141 %   >> get(gcf,'position')
142 % og kopier inn verdien i vektoren under.
143 plot(x,b,'b-x')
144 hold on
145 grid
146 title('Signal $b$')
147 xlabel('Indeks $k$.')
148
149
150 % Plukker ut annenhvert element, fra og med første.
151 b8 = b(1:2:end);
152 % Lager indeksvektor for plotting
153 x8 = 1:2:length(b);
154 % Plot disse med en relativt stor '.' som markør i rød farge
155 plot(x8, b8, 'r.', 'MarkerSize', 12)
156
157 % Plukker ut tredjehver element, starter fra andre element.
158 b9 = b(2:3:end);
159 % Lager indeksvektor for plotting
160 x9 = 2:3:length(b);
161 % Plot disse med en relativt stor 'o' som markør i grønn ...
162 % farge
163
164
165 legend('Signal $b$',...
166     'Annenhvert element, fra f{\o}rste element',...
167     'Tredjehver element, fra andre element',...
168     'location','northwest')
```

- Skjermdump av *Command Window* er gitt i figur 8.

```
Command Window
length gir største dimensjon en vei: - elementer
numel gir totalt antall: 135 elementer
size gir dimensjon [3 x 45]

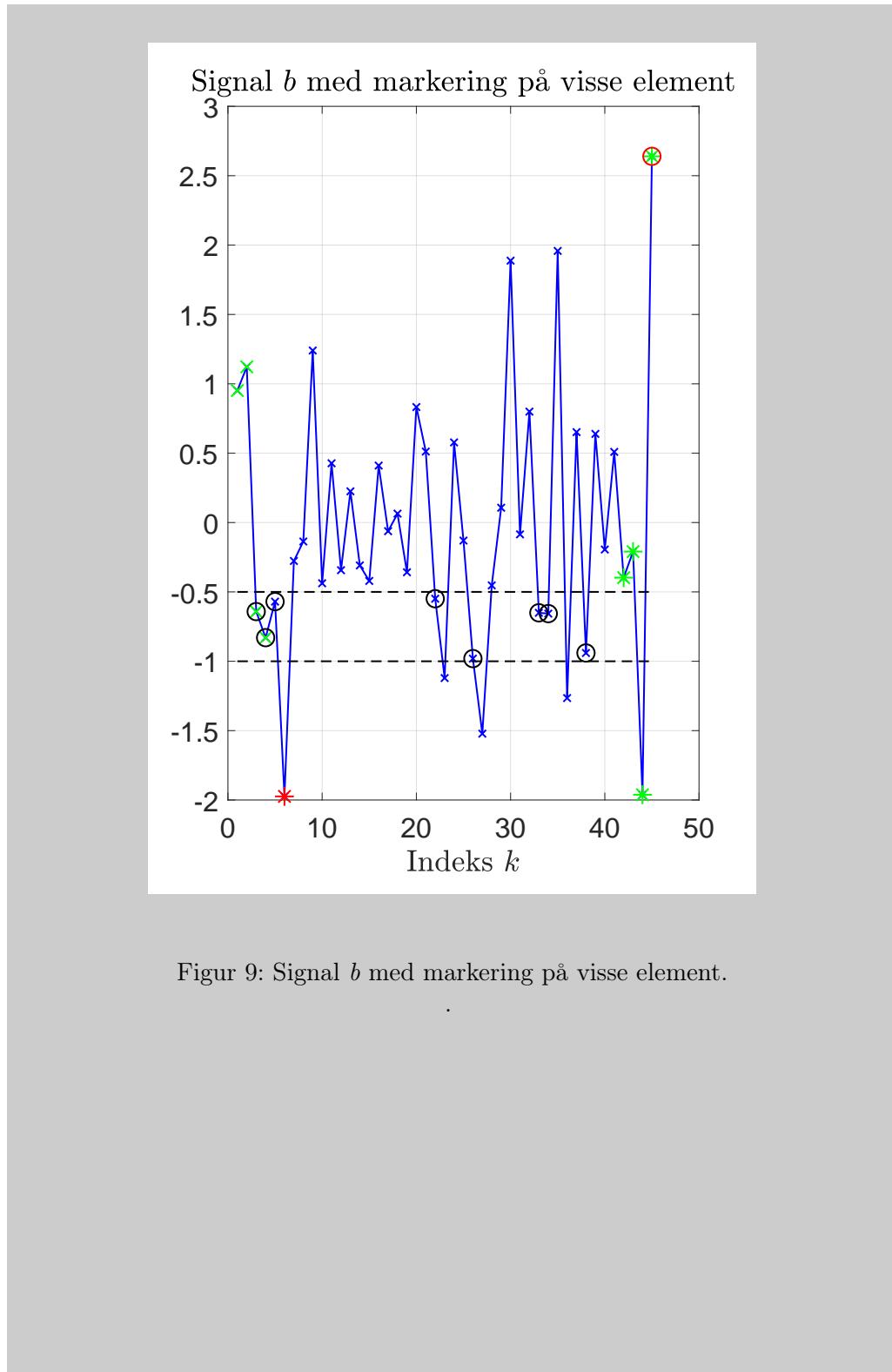
Første element = 0.95115
Siste element = 2.6395
4 første element = 0.95115      1.1221     -0.64179     -0.82988|
4 siste element = -0.39757     -0.20951     -1.9621      2.6395

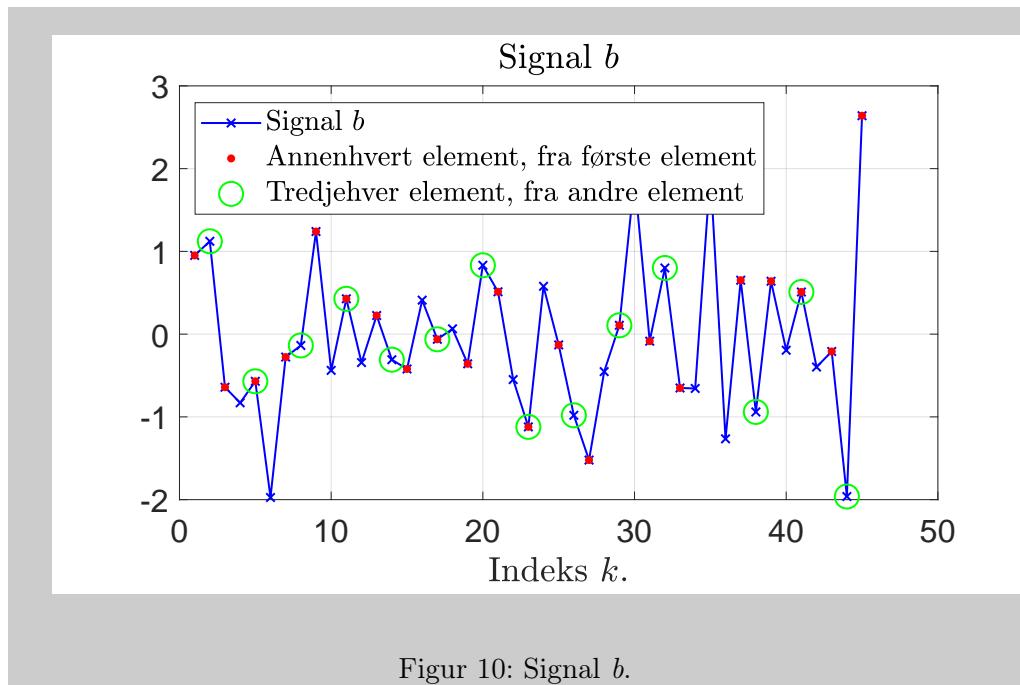
Ved indeks k = 45 er største verdi b = 2.6395
Ved indeks i = 6 er minste verdi b = -1.9741
Ved indeks i = [3    4    5    22   26   33   34   38]
er verdiene av b = [-0.642      -0.83      -0.57      -0.549      -0.981      -0.651      -0.656      -0.94]

Ved indeks i = 3 er verdien av b = -0.642
Ved indeks i = 4 er verdien av b = -0.83
Ved indeks i = 5 er verdien av b = -0.57
Ved indeks i = 22 er verdien av b = -0.549
Ved indeks i = 26 er verdien av b = -0.981
Ved indeks i = 33 er verdien av b = -0.656
Ved indeks i = 34 er verdien av b = -0.656
Ved indeks i = 38 er verdien av b = -0.94
```

Figur 8: Skjermdump av *Command Window*.

- Koden produserer figurene vist i figur 9 og 10.

Figur 9: Signal b med markering på visse element.

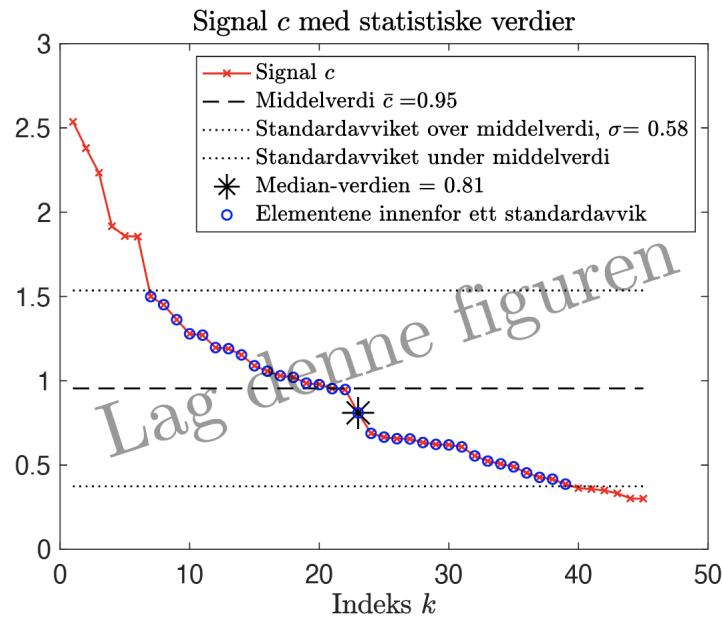
Figur 10: Signal b .

- c) I denne oppgaven skal du bli kjent med noen statistikk- og avrundingsfunksjoner. Følgende tekst står i skallfilen

Kode 7: Heading i oppgave c).

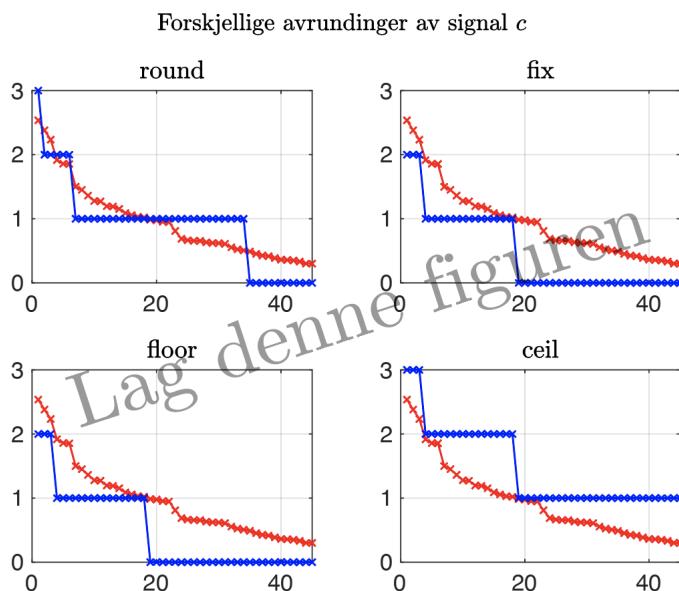
```
%% -----
% c)
%
% Innhold:
% - Vektor med tilfeldige tall sortert
% - Bruk av statistiske operasjoner
% - Bruk av avrundingsfunksjoner
% - Overordnet tittel i subplot
%
% Eksempler på nyttige Matlab-funksjoner
% - sort
% - abs
% - mean, std og median
% - ones
% - round, fix, floor, ceil
% - sum og prod
% - sgtitle
%
```

Ta utgangspunkt i `oving1_skallfil.m` og fyll inn det som mangler. Vis at du får resultatene vist i figurene 14 - 13, hvor `seed = 7`.



Figur 11: Resultat, hvor `seed = 7`.

```
's > tormod > Dropbox > faglig > fag > ELE130_Anvendt_matematikk_og_fys
Command Window
New to MATLAB? See resources for Getting Started.
Middelverdi = 0.95
Standardavvik = 0.58
Median = 0.81
Andel elementer innenfor ett standardavvik: 73.3%
Sum av alle element = 42.961
Produkt av alle element = 5.3624e-05
fx >>
```

Figur 12: Skjermdump av *Command Window*.Figur 13: Resultat, hvor $\text{seed} = 7$.

Svar:

- Forslag til løsning er gitt i kodeutdrag 8.

Kode 8: Kode for oppgaven.

```

1      clear; close all; clc
2
3  % Absoluttveriden av tilfeldige tall, sortert fra høyest ...
   % til lavest verdi
4  seed = 7;
5  rng(seed)
6  c = sort(abs(randn(1,45))+0.3, 'descend');
7  k = length(c);
8
9  % Plot de til sorterte tallene i stigende rekkefølge
10 figure
11 plot(c, 'r-x')
12 hold on
13 title('Signal $c$ med statistiske verdier')
14 xlabel('Indeks $k$')
15
16 % Beregner middelverdi c1, standardavvik c2 og ...
   % medianverdi c3,
17 % og plotter inn disse som hhv. linjer og punkt
18 c1 = mean(c);
19 c2 = std(c);
20 c3 = median(c);
21 x3 = find(c == c3);    % finn indeksen x3 til ...
   % medianverdien ut fra c3
22
23 plot(c1*ones(1,k), 'k--')      % middelverdi
24 plot(c1 + c2*ones(1,k), 'k:')  % standardavvik over ...
   % middelverdi
25 plot(c1 - c2*ones(1,k), 'k:')  % standardavvik under ...
   % middelverdi
26 plot(x3, c3, 'k*', 'MarkerSize', 20)
27
28 % Skriver ut verdiene med redusert antall desimaler,
29 % ved bruk av disp og sprintf
30 disp(['Middelverdi = ', num2str(c1, '%.2f')])
31 disp(['Standardavvik = ', num2str(c2, '%.2f')])
32 str = sprintf(['Median = ', num2str(c3, '%.2f'), '\n']);
33 disp(str)
34
35 % Beregner, finner og indikerer verdiene innenfor ett ...
   % std-avvik.
36 % Bruk kombinasjon av c1 og c2 til i find(..) til å finne ...
   % indeksene x4.
37 x4 = find(c>=c1-c2 & c<=c1+c2);
38 c4 = c(x4);
39 plot(x4, c4, 'bo')

```

```
40
41 % Beregner %-andelen elementer innenfor ett standardavvik
42 andel = length(x4)/length(c) * 100;
43 disp(['Andel elementer innenfor standardavviket: ...
44     ',num2str(andel,3),'%'])
45 disp(' ')
46 legend('Signal $c$',...
47     ['Middelverdi $\bar{c}=$',num2str(c1,2)],...
48     ['Standardavviket over middelverdi, $\sigma$= ...
49         ',num2str(c2,2)],...
50     ['Standardavviket under middelverdi',...
51     ['Median-verdien = ',num2str(c3,2)],...
52     'Elementene innenfor ett standardavvik',...
53     'location','northeast')
54 %-----
55 % Ulike avrundinger av c
56 %
57 c5 = round(c);      % round
58 c6 = fix(c);       % fix
59 c7 = floor(c);     % floor
60 c8 = ceil(c);      % ceil
61
62 figure
63 set(gcf,'position',[400 100 700 400])
64 subplot(2,2,1)
65 plot(c,'r-x')
66 grid
67 hold on
68 plot(c5,'b-x')
69 title('round')
70
71 subplot(2,2,2)
72 plot(c,'r-x')
73 grid
74 hold on
75 plot(c6,'b-x')
76 title('fix')
77
78 subplot(2,2,3)
79 plot(c,'r-x')
80 grid
81 hold on
82 plot(c7,'b-x')
83 title('floor')
84
85 subplot(2,2,4)
86 plot(c,'r-x')
87 grid
88 hold on
89 plot(c8,'b-x')
90 title('ceil')
```

```

91
92 sgttitle('Forskjellige avrundinger av signal $c$')
93
94 % Beregn summen c7 og produktet c8 av alle elementene i c
95 c7 = sum(c);      % summen
96 disp(['Sum av alle element = ',num2str(c7)])
97
98 c8 = prod(c);    % produktet
99 disp(['Produkt av alle element = ',num2str(c8)])

```

- Skjermdump av *Command Window* er gitt i figur 14.

Command Window

```

Middelverdi = 0.95
Standardavvik = 0.58
Median = 0.81

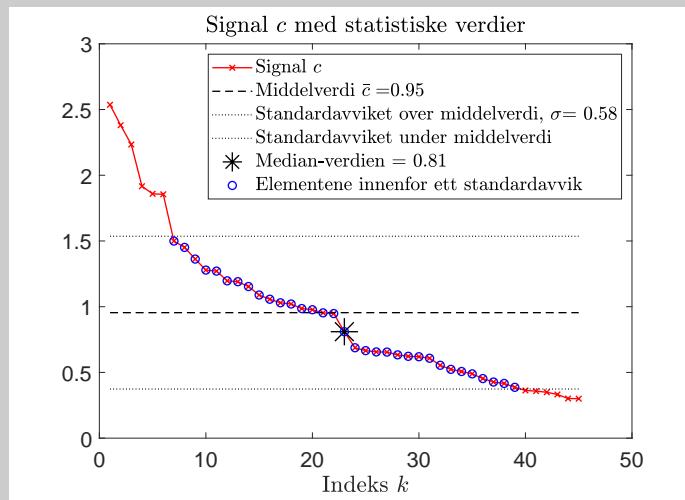
Andel elementer innenfor standardavviket: 73.3%

Sum av alle element = 42.961
Produkt av alle element = 5.3624e-05

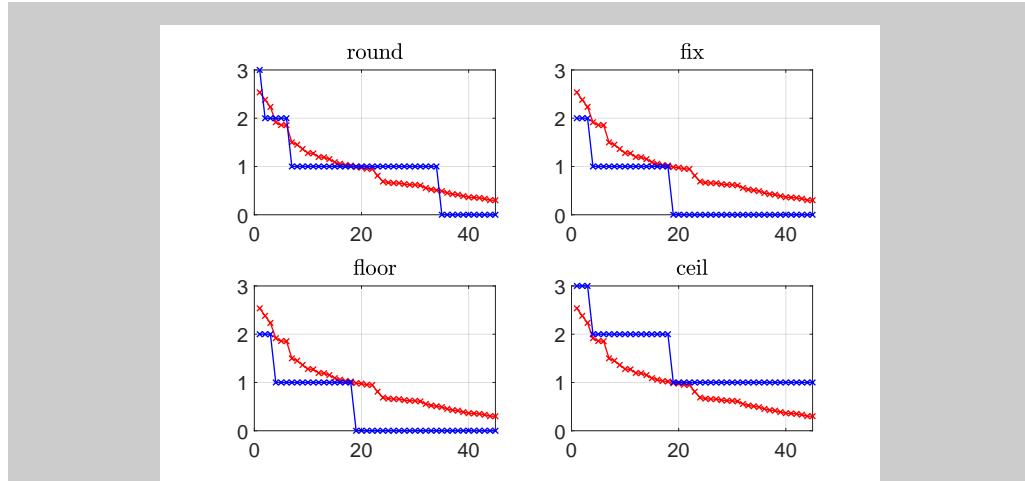
```

Figur 14: Skjermdump av *Command Window*.

- Koden produserer figuren vist i figur 15 og 16.



Figur 15: Skjermdump av *Command Window*.

Figur 16: Skjermdump av *Command Window*.

- d) I denne oppgaven skal du bli kjent med rekursiv kode i for-løkker og elementvis operasjoner. Følgende tekst står i skallfilen

Kode 9: Heading i oppgave d).

```
%% -----
% d)
%
% Innhold:
%   - Bruk av for-løkker til rekursiv beregning.
%   - Sammenligning av for-løkker med elementvise operasjoner
%
% Eksempler på nyttige Matlab-funksjoner
%   - sqrt
%   - ^
%   - .^ (elementvise operasjoner)
%   - for-løkker
%   - set og gcf
%   - stem
%   - bar
%
```

Oppgaven tar utgangspunkt i vektoren x gitt som har M elementer

$$x = [1.5, 2, 2.5, 3, \dots, 21] \quad (2)$$

- Du skal første beregne d_1 gitt som

$$d_{1,k} = \sum_{n=1}^k \sqrt{x_n} \quad (3)$$

på en *rekursiv* måte i en **for**-løkke med bruk av indeks k . Skallfilen inneholder to alternative måter å gjøre det på.

- Videre skal du beregne d_2 fra følgende uttrykk

$$d_2 = \sum_{n=1}^M \sqrt{x_n} \quad (4)$$

ved å kun bruke **sum**-funksjonen. Du skal vise at d_2 er identisk med siste element i d_1 .

- Deretter skal du beregne d_3 gitt som

$$d_{3,k} = \sum_{n=1}^k (x_n)^2 \quad (5)$$

på en rekursiv måte i en ny **for**-løkke.

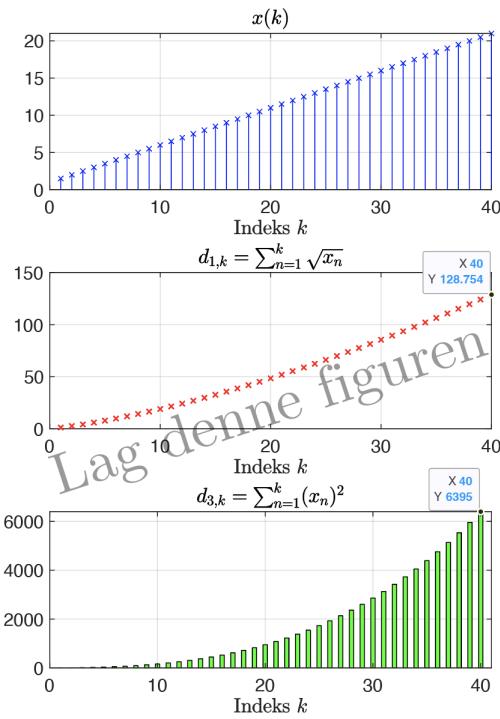
- Til slutt skal du beregne d_4 fra følgende uttrykk

$$d_4 = \sum_{n=1}^M (x_n)^2 \quad (6)$$

ved å kun bruke `sum`-funksjonen og elementvise operasjoner. Du skal vise at d_4 er identisk med siste element i d_3 .

- I siste del av koden plottes x , d_1 og d_3 i hvert sitt `subplot(..)`. Du skal klikke deg inn på begge kurvene og avlese siste verdi. Ta med figur i innleveringen.

Vis at du får følgende figur og resultater i Command Window.



Figur 17: Resultat fra kjøring av koden.

```
Command Window
d1(end) = 128.8
d2 = 128.7538
d3(end) = 6395
d4 = 6395
fx >>
```

Figur 18: Skjermdump av Command Window.

- e) I denne oppgaven skal du bli kjent med rekkeutviklingen bak eksponentialfunksjonen² og mer om elementvise operasjoner. Følgende tekst står i skallfilen

Kode 10: Heading i oppgave e).

```
%%
% -----
% e)
%
% Innhold:
% - Analyse av rekkeutviklingen bak eksponentialfunksjonen
% - Sammenligning mellom Matlab-funksjon og egenprodusert kode
% - Sammenligning av resultat fra for-løkke og elementvise ...
%   operasjoner
%
% Eksempler på nyttige Matlab-funksjoner
% - exp
% - factorial
% - semilogy
% - log10 og log
% - ./ (elementvise operasjoner)
% - .* (elementvise operasjoner)
% - format
% - sgtitle
% - axis
% -----
```

Oppgaven tar utgangspunkt i uttrykket for funksjonen $e(x, M)$ gitt som

$$e(x, M) = \sum_{n=0}^M \frac{x^n}{n!} \quad (7)$$

- Du skal først implementere kode for ligning (7) ved bruk av en **for**-løkke, hvor vi benytter **x=2** og **M=40**. Svarvariabelen kalles for **e_xM1**.
- Deretter skal du implementere kode for ligning (7) med kun elementvise operasjoner. Svarvariabelen kalles for **e_xM2**. Du skal vise at differensen mellom **e_xM2** og **e_xM1** er 0.
- Dersom $M=\infty$ så tilsvarer uttrykket i ligning (7) eksponentialfunksjonen e^x skrevet som

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \quad (8)$$

Du skal derfor benytte **exp**-funksjonen til å beregne e^2 . Svarvariabelen kalles for **e_x**. Du skal vise at avviket mellom **e_x** og **e_xM2** er litt forskjellig fra 0.

²..som er bakgrunnen for at dette er deloppgave e) nerdehumor.

- For å undersøke hvordan avviket mellom `e_x` og `e_xM2` utvikler seg som funksjon av M , er det lagt inn en `for`-løkke hvor M øker fra 0 til 100. For hver runde beregnes avviket `avvik(k)`, og til slutt plottes avviket som funksjon av M ved bruk av lineær skala i `subplot(1,3,1)`.
- Ut fra `subplot(1,3,1)` er det vanskelig å se detaljene i hvordan avviket nærmer seg 0 når M blir stor. En vanlig teknikk for å øke den visuelle informasjonen/innsikten i slike figurer hvor en variabel har verdier som strekker seg fra svært små verdier til svært store verdier, er å anvende logaritmisk skala på det som skal plottes. Dette gjøres i `subplot(1,3,2)` med bruk av `semilogy`-kommandoen. Du ser da at det fremkommer detaljer rundt $M \approx 40$ som ikke vises i `subplot(1,3,1)`.
- Et tredje alternativ er å først beregne 10'er-logaritmen og deretter plotte resultatet. Verdiene på y-aksen vil da representere eksponenten vist på y-aksen til `subplot(1,3,2)`. Matlabkommandoene og den matematiske syntaksen for henholdsvis 10'er-logaritmen `log10()` og den naturlige logaritmen `ln()` er noe forvirrende. Et eksempel på bruk av 10'er-logaritme er

$$\log_{10}(1000) = \log_{10}(10^3) = 3, \quad (9)$$

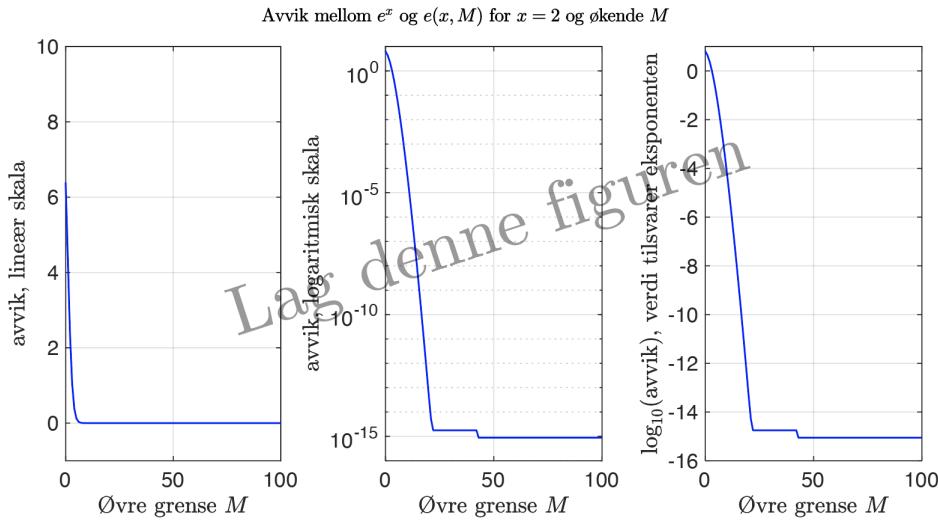
og den tilhørende Matlab-funksjonen er `log10()`. Et eksempel på matematisk syntaks for den naturlige logaritmen er

$$\ln(7.3891) = \ln(e^2) = 2 \quad (10)$$

og den tilhørende Matlab-funksjonen er `log()`. Årsaken til dette er nok at den naturlige logaritmen også skrives som $\log_e()$ i noen lærebøker.

Moralen i historien er uansett at Matlab ikke har en kommando som heter `ln`. I skallfilen skal du derfor beregne både den naturlige logaritmen og 10'er logaritmen av `e_x` slik at du husker på at du aldri må blande disse funksjonene.

I `subplot(1,3,3)` plottes til slutt 10'er-logaritmen av avviket, hvor verdiene på y-aksen tilsvarer eksponentene i `subplot(1,3,2)`. Vis at du får følgende figur.



Figur 19: Resultat.

- Deretter er det klargjort noe kode for at du skal bli kjent med kommandoene `format long` og `format short`. Vis også at du nå har fått følgende resultat i Command Window.

```

ilig > fag > ELE130_Anvendt_matematikk_og_fysikk_i_robotprogrammering > Studie2024 > Ovi
Command Window
New to MATLAB? See resources for Getting Started.
for-løkke:      e_xM1 = 7.3891
elementvis:      e_xM2 = 7.3891
Avvik mellom for-løkke og elementvis = 0
exp-funksjonen: e_x = 7.3891
Avvik mellom exp-funksjon og for-løkke = 1.7764e-15

Naturlige logaritmen log(e_x) = 2
10'er logaritmen log10(e_x) = 0.86859

Siste element i log10(avvik) med 'format short' og 'format long':
verdi =
-15.0515
verdi =
-15.051499783199059

```

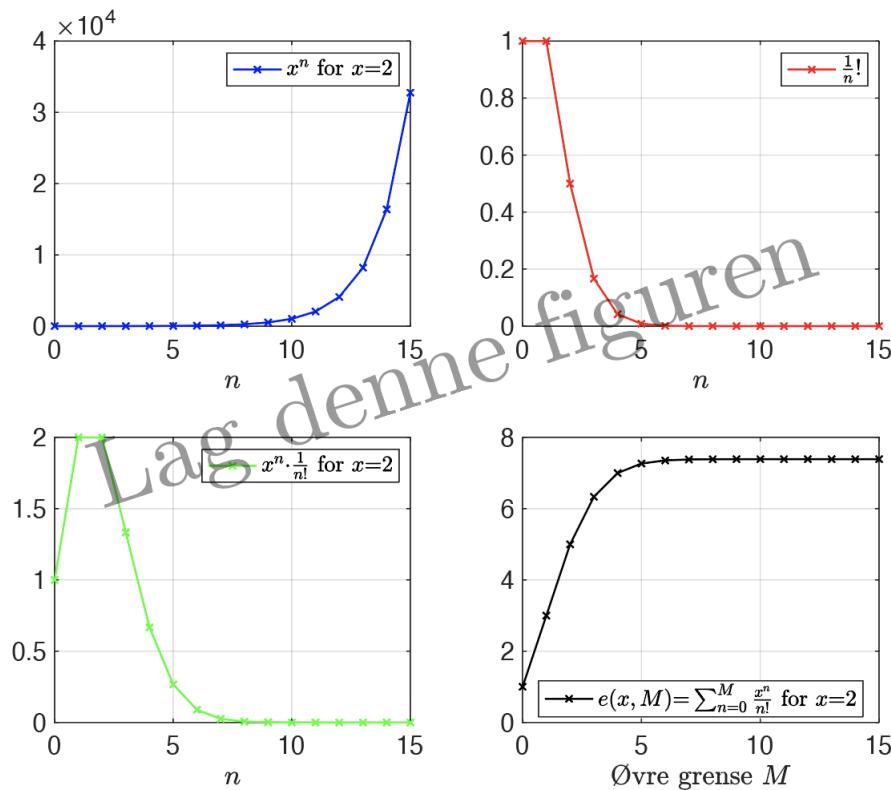
Figur 20: Skjermdump av *Command Window*.

- Med utgangspunkt i at

$$\log_{10}(avvik) = -15.0515 \quad (11)$$

vis i innleveringen din hvordan du matematisk finner verdien av *avvik*? Stemmer denne verdien med siste element i `avvik(k)`

- Den siste delen av oppgaven undersøker hvorfor summen i ligning (7) faktisk konvergerer, selv for veldig store x -verdier. For å bli overbevist om hvorfor det er slik, skal du ferdigstille koden og forhåpentligvis la deg fascinere over at produktet mellom x^n og $\frac{1}{n!}$ blir et lite tall for store verdier av n , og uavhengig av verdien på x . Vis at du får følgende figur.



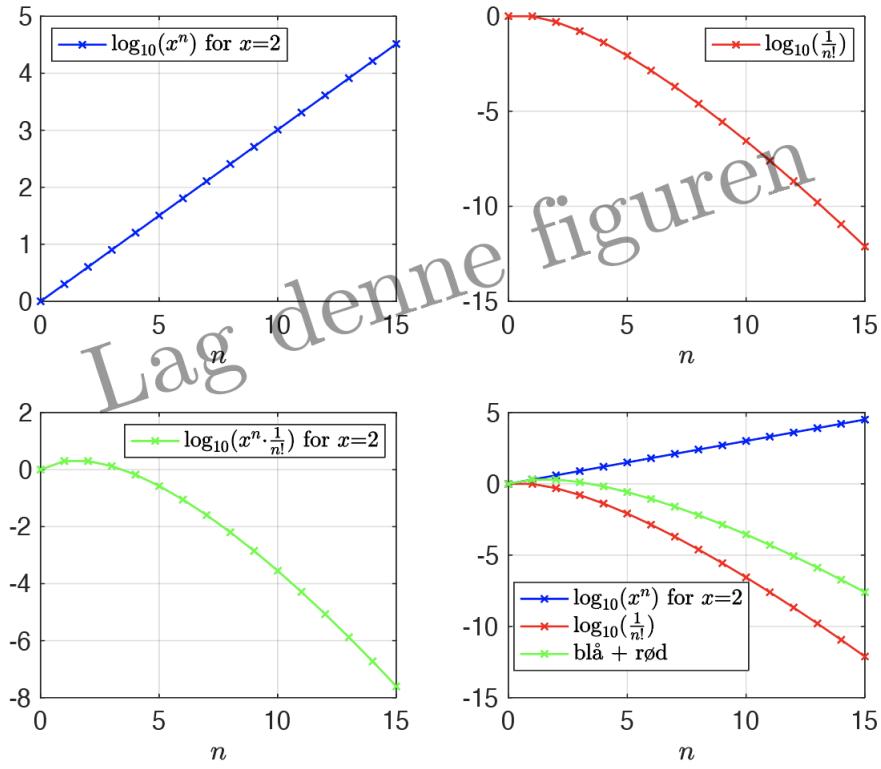
Figur 21: Resultat fra kjøring av kode.

Gi en forklaring på hva figuren viser.

- På samme måte som i `subplot(1,3,1)` i figur 19 er det vanskelig å se detaljer i hva som skjer med den grønne kurven i figur 21 når n blir stor. En måte å få innsikt i dette på er igjen å beregne 10'er-logaritmen til uttrykket $x^n \cdot \frac{1}{n!}$ og plotte resultatet. På denne måten får vi samtidig demonstrert at

$$\log_{10}\left(x^n \cdot \frac{1}{n!}\right) = \log_{10}(x^n) + \log_{10}\left(\frac{1}{n!}\right) \quad (12)$$

Vis at du får følgende figur som har tilnærmet samme struktur som figur 21. Eneste forskjellen er at `subplot(2,2,4)` viser alle de andre 3 delfigurene samlet slik at du kan se ligning (12) på en grafisk måte.



Figur 22: Resultat.

Gi en forklaring på hva resultatene i figuren viser.

- Basert på koden til figur 22, lag en ny figur som viser den alternative måten å skrive ligning (12) på, nemlig

$$\log_{10}\left(\frac{x^n}{n!}\right) = \log_{10}(x^n) - \log_{10}(n!) \quad (13)$$

- Du må gjerne kjøre hele koden på ny for en MYE større x -verdi, og bli overbevist om at uttrykket for eksponentialfunksjonen faktisk konvergerer uansett.

- f) I denne oppgaven skal du bli kjent med histogram-funksjonen. Følgende tekst står i skallfilen.

Kode 11: Heading i oppgave f).

```
%% -----
% f)
%
% Innhold:
%   - Bruk av histogram, styring av intervall
%   - Styring av akseverdier
%   - Plassere tekst i figurer
%
% Eksempler på nyttige Matlab-funksjoner
%   - histogram
%   - ceil, floor
%   - axis
%   - xlim
%   - text
%   - sprintf og ylabel
%
```

Koden tar utgangspunkt i at du skal plotte fordelingen av de tilfeldige tallene gitt i variabelen `x`.

- Først plottes `x` i `subplot(2,1,1)` og deretter histogrammet til `x` ved bruk av `histogram`-funksjonen i `subplot(2,1,2)`. Som du ser benyttes en returnvariabel `x_prop` som viser alle egneskapene til histogrammet. Disse skrives ut i Command Window.
- Legg også merke til at den lange tekststrengen til `ylabel` blir delt opp ved bruk av `sprintf` og `\n`.
- Deretter plukkes elementene i intervallet

$$40 < x < 50 \quad (14)$$

ut og plottes i `subplot(2,1,1)`. På denne måten kan du se sammenhengen mellom tidssserien og det tilhørende histogrammet.

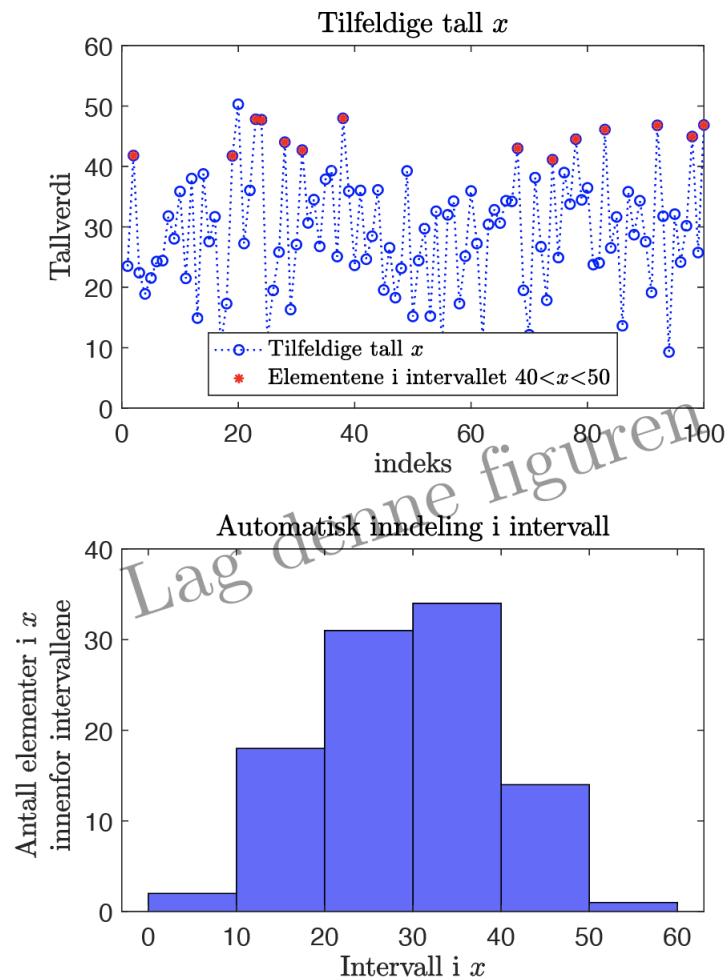
- I `figure(2)` er poenget å vise hvordan du kan styre histogrampresentasjonen med enten

- å bestemme antall intervall `nbins`
- eller å bestemme grenseverdiene og intervallbredden med `edges`

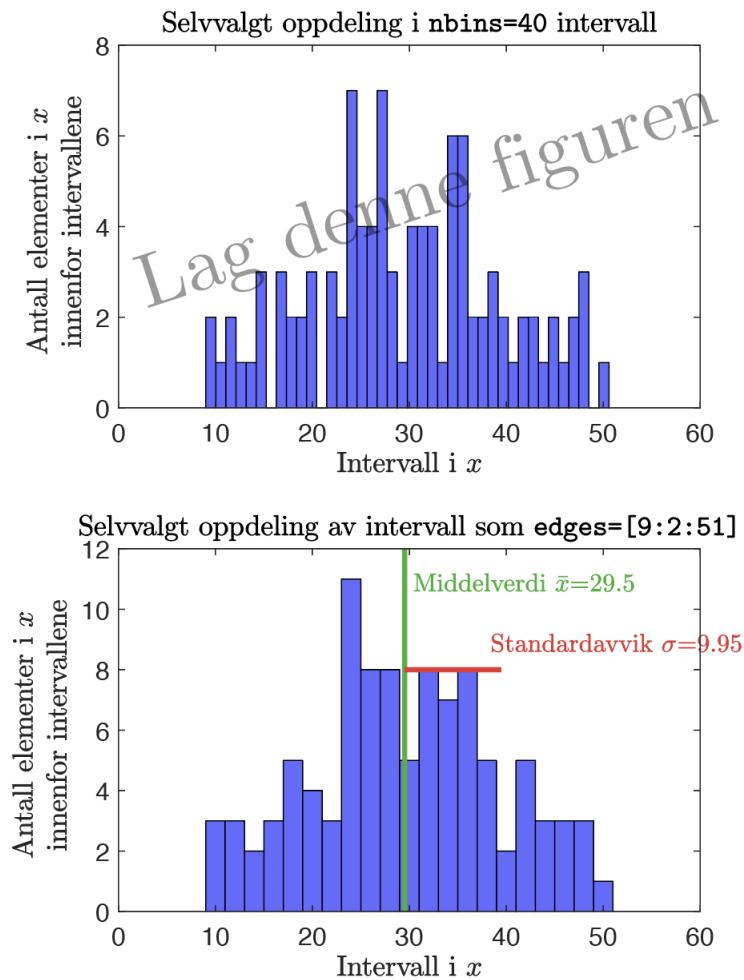
For å bestemme grenseverdiene skal du finne minimums- og maksimumsverdiene av `x` og runde av henholdsvis nedover og oppover.

- Til slutt avleses akseverdiene `ax = axis;`, hvor `ax` blir en 1×4 vektor, og akseverdiene kan hentes ut som `ax(1)`, `ax(2)`, `ax(3)` og `ax(4)`. Disse verdiene brukes for å plotte linjer og plassere tekst i det siste histogrammet.

Vis at du får resultatene vist i figurene 23 - 28, hvor `seed = 1`.



Figur 23: Resultat, hvor `seed = 1`.

Figur 24: Resultat, hvor `seed = 1`.

```
tormod > Dropbox > faglig > fag > ELE130_Anvendt_matematikk_og_fysikk_i_robotprog
Command Window
New to MATLAB? See resources for Getting Started.
x_prop =
Histogram with properties:
    Data: [23.5099 41.8117 22.4155 18.9039 21.5445 24.2734 24
    Values: [2 18 31 34 14 1]
    NumBins: 6
    BinEdges: [0 10 20 30 40 50 60]
    BinWidth: 10
    BinLimits: [0 60]
    Normalization: 'count'
    FaceColor: 'auto'
    EdgeColor: [0 0 0]
Show all properties
```

Figur 25: Skjermdump av *Command Window*.

Svar:

- Forslag til løsning er gitt i kodeutdrag 12.

Kode 12: Kode for oppgaven.

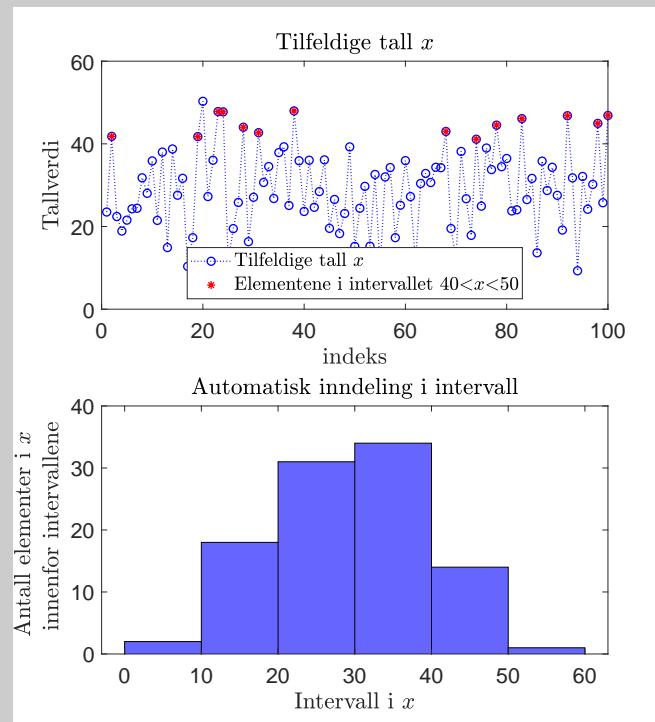
```
1      clear; close all; clc
2
3  seed = 1;
4  rng(seed)          % styring av tilfeldige tall
5
6  % Signal x
7  n = 100;           % antall elementer i x-vektoren
8  x = abs(randn(1,n)*10 + 30);
9
10 %
11 % Plotter tallene i x-vektor i subplot(2,1,1)
12 %
13 subplot(2,1,1)
14 plot(x,'b:o')
15 ylabel('Tallverdi')
16 xlabel('indeks')
17 title('Tilfeldige tall $x$')
18
19 %
20 % Plotter histogrammet til x i subplot(2,1,2)
21 % samtidig som vi henter ut x_prop som gir
22 % mye informasjon om histogrammet, se Command Window.
23 %
24 % I koden nedenfor kan du velge å benytte
25 %   - x_prop.BinEdges
26 %   - x_prop.BinWidth
27 %   - x_prop.BinLimits
28 % til å hente ut data fra x og plotte i subplot(2,1,1).
29 % Alternativt kan du hardkode grenseverdiene.
30 %
31 figure(1)
32 set(1,'position',[400 400 700 400])
33 subplot(2,1,2)
34 x_prop = histogram(x) % ---> sjekk Command Window
35
36 % Tekststreng for ylabel.
37 % Bruker sprintf for å kunne fordele teksten opp på 2 linjer
38 str = sprintf('Antall elementer i $x$ \n innenfor ...
39     intervallene');
40 ylabel(str)
41 xlabel('Intervall i $x$')
42 title('Automatisk inndeling i intervall')
43 %
44 % Fremheving av elementene fra x-vektor i ett intervall
45 %   x_low < x < x_high
```

```
46 %      40 < x < 50
47 % Bruk enten xprop fra histogrammet
48 % eller hardkodede verdier.
49 %-----
50
51 % Velg riktig element BinEdges som gir x_min = 40
52 x_low = x_prop.BinEdges(5);
53 % x_low = 40;    % alternativt hardkodet
54
55 % x_high er ett intervall større
56 x_high = x_low + x_prop.BinWidth;
57
58 % Velger ut indeks-verdier fra intervallet
59 i = find(x>x_low & x<x_high);
60
61 subplot(2,1,1)
62 hold on
63 plot(i, x(i), 'r*')
64 legend('Tilfeldige tall $x$',...
65     ['Elementene i intervallet $',num2str(x_low),...
66     '{<}x{<}$',num2str(x_high)],...
67     'location','south')
68
69
70 %-----
71 % Ny figur.
72 % Plotter histogrammet til x hvor vi
73 % styrer antall intervall (nbins)
74 %-----
75 figure(2)
76 set(2,'position',[400 100 700 400])
77 subplot(2,1,1)
78 % velg selv antall intervall
79 nbins = 40;
80 histogram(x,nbins)
81 ylabel(str)
82 xlabel('Intervall i $x$')
83 title(['Selvvalgt oppdeling i \tt ...
84 nbins=',num2str(nbins),'] intervall')
85 xlim(x_prop.BinLimits)
86
87 %-----
88 % Alternativ måte å styre histogrammet på er
89 % å styrer intervallgrensene og
90 % intervallbredden (edges) som
91 % fra:bredde:til
92 %-----
93 subplot(2,1,2)
94
95 % Intervallgrensene, runder oppover og nedover
96 max_x = ceil(max(x));
97 min_x = floor(min(x));
```

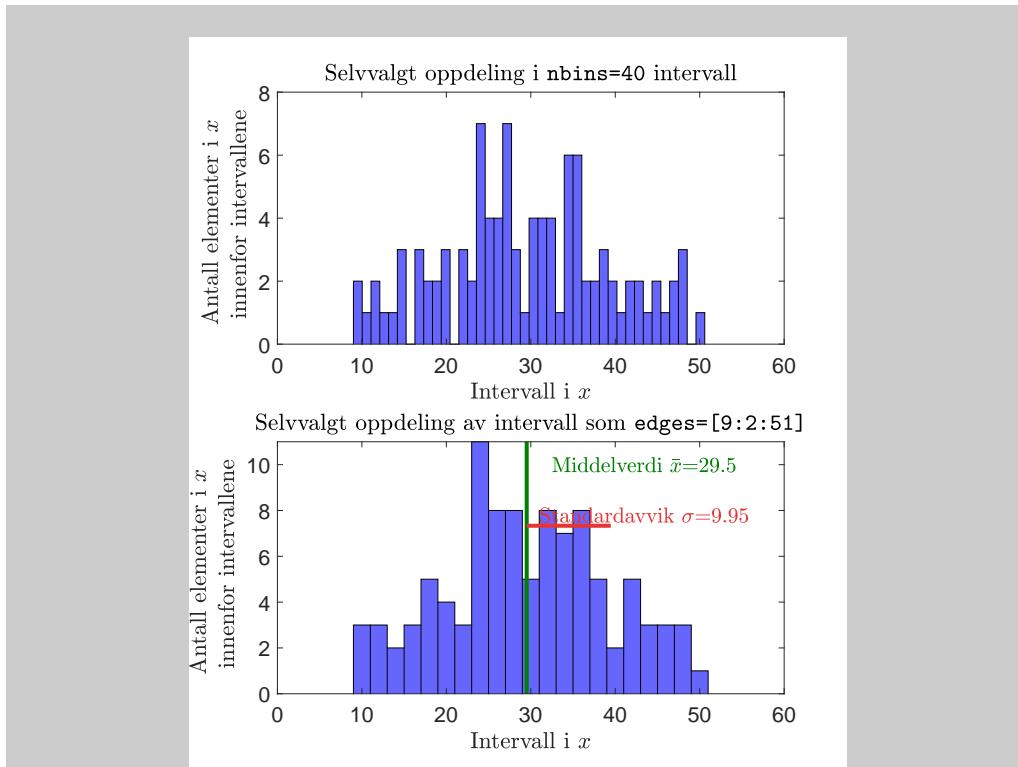
```
98
99 % Velg intervallbredde
100 int_width = 2;
101 edges = min_x:int_width:max_x;
102 histogram(x,edges)
103 ylabel(str)
104 xlabel('Intervall i $x$')
105 title(['Selvvalgt oppdeling av intervall som {\tt ...
106     edges=[,...,
107         num2str(min_x),':',num2str(int_width),':',num2str(max_x),']}]')
108 xlim(x_prop.BinLimits)
109 hold on
110 %-----
111 % Skissering av middelverdi og standardavvik i histogrammet
112 %
113 mean_value = mean(x);
114 std_value = std(x);
115
116 % x-koordinaten til text-funksjonen
117 x1 = mean_value;
118
119 % Benytter y-akseinformasjon for å både plotte loddrett
120 % strek samt å plassere tekst
121 ax = axis;    % leser av aksene
122 y1 = ax(3);  % plukker ut laveste y-akseverdi
123 y2 = ax(4);  % plukker ut høyeste y-akseverdi
124
125
126 % Definerer fargen darkgreen ut fra [R G B],
127 % hvor [1 1 1] = hvitt og [0 0 0] = svart
128 darkgreen = [0 0.5 0.0];
129
130 % Plotter loddrett strek i farge darkgreen
131 plot([x1 x1],[y1 y2],'color',darkgreen, 'LineWidth',3)
132
133 % Plasserer inn en tekststreng hvor vi hardkoder ...
134 % justering på
135 % plasseringen med å gange med en tallverdi større eller ...
136 % mindre enn 1
137 text(x1*1.1, y2*0.9, ...
138     ['Middelverdi $\bar{x}$=',num2str(mean_value,3)],...
139     'color',darkgreen)
140
141 % x- og y-koordinatene til text
142 x1 = mean_value; % gjentar denne
143 x2 = mean_value + std_value;
144 y1 = (ax(4)+ax(3))/1.5;
145
146 % Definerer fargen darkred ut fra [R G B]
147 darkred = [0.9 0.2 0.2];
148
149 % Plotter vannrett strek i farge darkred
```

```
148 plot([x1 x2],[y1 y1], 'color', darkred,'LineWidth',3)
149 text(x1*1.05, y1*1.05 ,...
150     ['Standardavvik $\sigma$=', num2str(std_value,3)],...
151     'color',darkred)
```

- Koden produserer figuren vist i figur 26 og 27.



Figur 26: Resultat, der seed = 1.



Figur 27: Resultat, der seed = 1.

- Skjermdump av *Command Window* er gitt i figur 28.

```
Command Window

x_prop =
Histogram with properties:
Data: [23.5099 41.8117 22.4155 18.9
Values: [2 18 31 34 14 1]
NumBins: 6
BinEdges: [0 10 20 30 40 50 60]
BinWidth: 10
BinLimits: [0 60]
Normalization: 'count'
FaceColor: 'auto'
EdgeColor: [0 0 0]
Show all properties
```

Figur 28: Skjermdump av *Command Window*.

- g) I denne oppgaven skal du bli kjent med `sin`, `findpeaks` og `text` funksjonene. For å kunne bruke `findpeaks` må du ha Signal processing toolbox installert. I tillegg skal du anvende `arrow`-funksjonen som er hentet fra Matlab File Exchange og som er en av filene du pakket opp fra `mylib_SavemMyFigure_startup.zip`. Sjekk at du har denne filen ved å skrive

`>> which arrow.m` i Command Window. Følgende tekst står i skallfilen

Kode 13: Heading i oppgave g).

```
%%
% -----
% g)
%
% Innhold:
%   - Bruk av findpeaks-funksjonen
%
% Eksempler på nyttige Matlab-funksjoner
%   - sin
%   - findpeaks
%   - text
%   - arrow (hentet fra Mathworks File Exchange)
%     https://se.mathworks.com/matlabcentral/fileexchange/278-arrow
% -----
```

Utgangspunktet for oppgaven er inngangssignalet $u(t)$ og utgangssignalet $y(t)$ for en gitt prosess (begge har $\omega=4$ rad/s)

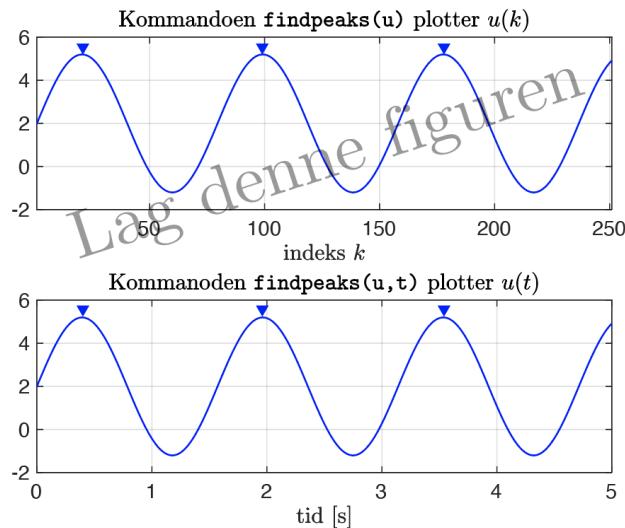
$$u(t) = 3.2 \sin(4t) + 2 \quad (15)$$

$$y(t) = 1.4 \sin(4t - 1.9) + 3 \quad (16)$$

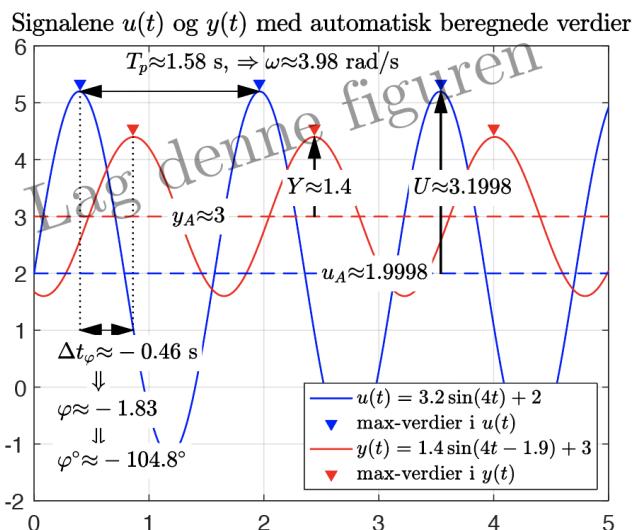
- Som mange andre Matlab-funksjoner kan `findpeaks` benyttes både med og uten returargument. Første del av oppgaven viser effekten av disse to variantene anvendt på på signalet u . Varianten med returargument viser informasjonen i Command Window.
- Koden beregner automatisk følgende størrelser:
 - Periodetiden T_p og vinkelfrekvens ω
 - Amplitudene U og Y
 - Likevektsverdiene u_A og y_A
 - Tidsforskyvningen Δt_φ og faseforskyvningene φ [rad] og φ° [grader]
- Endre gjerne på tidsoppløsningen gitt av `delta_t` og studer resultatet i forhold til de beregnede størrelsene.

Vis at du får resultatene vist i figurene 32 - 34.

tormod > Dropbox > faglig > fag > ELE130_Anven
Command Window
New to MATLAB? See resources for [Getting Started](#).
Ved t = [0.393 1.96 3.53]
er u = [5.2 5.2 5.2]

Figur 29: Skjermdump av *Command Window*.

Figur 30: Eksempel på resultat.



Figur 31: Eksempel på resultat.

Svar:

- Forslag til løsning er gitt i kodeutdrag 14.

Kode 14: Kode for oppgaven.

```

1      clear; close all; clc
2
3  delta_t = 0.02;
4  t = 0:delta_t:5;
5  u = 3.2*sin(4*t)+2;
6
7  % Det er to måter å bruke findpeaks på:
8
9  % -----
10 % 1) Uten returargument plottes figur med toppunkter,
11 %     enten som funksjon av indeksene eller
12 %     som funksjon av en definert tidsvektor t.
13 %
14 figure
15 subplot(2,1,1)
16 findpeaks(u)
17 xlabel('indeks $k$')
18 title('Kommandoen {\tt findpeaks(u)} plotter $u(k)$')
19
20 subplot(2,1,2)
21 findpeaks(u, t)
22 xlabel('tid [s]')
23 title('Kommanoden {\tt findpeaks(u,t)} plotter $u(t)$')
24
25 %
26 % 2) Med bruk av returargument plottes ikke figuren,
27 %     men toppunkt-verdiene og indeksene returneres,
28 %     her kalt u_pk og u_locs
29 %
30 [u_pk, u_locs] = findpeaks(u);
31
32 % Tidspunkt for maksverdi
33 t_pk = t(u_locs);
34
35 % skriver ut til Command Window
36 disp(['Ved t = [', num2str(t_pk, 3), ']'])
37 disp([' er u = [', num2str(u_pk, 3), ']'])
38
39
40 %
41 % Automatisk beregning av periode, vinkelfrekvens,
42 % likevektsverdier og amplituder.
43 % Plotter først signalene u(t) og y(t) med ...
44 %         findpeaks-funksjonen
45 y = 1.4*sin(4*t-1.9)+3;

```

```

46
47 % Plotter toppene i u(t) og y(t)
48 figure
49 findpeaks(u,t)
50 hold on
51 findpeaks(y,t)
52
53
54 % Finner maksverdiene og tilhørende indeks
55 [u_pk, u_locs] = findpeaks(u);
56 [y_pk, y_locs] = findpeaks(y);
57
58 % Finner tidspunktet tilsvarende indeksene
59 t_pk_u = t(u_locs);
60 t_pk_y = t(y_locs);
61
62 % Beregner T_p og omega
63 T_p = mean(diff(t_pk_y));
64 w = 2 * pi / T_p;
65
66 % Beregner likevekstverdi/konstantleddet
67 u_A = mean(u);
68 y_A = mean(y);
69
70 % Beregner amplitudene
71 U = (max(u)-min(u)) / 2;
72 Y = (max(y)-min(y)) / 2;
73
74 % Beregner faseforsyningen
75 delta_t_phi = t_pk_u(1) - t_pk_y(1);
76 phi = w * delta_t_phi;
77 phi_d = phi * (180 / pi);
78
79
80 %-----
81 % Legger på piler og linjer og tekst som viser
82 % beregnede verdier.
83 %-----
84
85 % Periode T_p og vinkelfrekvens w
86 xy1 = [t_pk_u(1), u_pk];
87 xy2 = [t_pk_u(2), u_pk];
88 arrow(xy1, xy2, 'Ends', [1, 2])
89 text(0.8, 5.7, ['$T_p$\{ \approx \}$', num2str(T_p, 3), ' s, ...',
90 '$\rightarrow$', ...
91 '$\omega$\{ \approx \}', num2str(w, 3), '$ rad/s$'])
92
93 % Tidsforskyvning delta_t_phi og faseforskyvning phi
94 xy1 = [t_pk_u(1), 1];
95 xy2 = [t_pk_y(1), 1];
96 arrow(xy1, xy2, 'Ends', [1, 2])
97 text(0.2, 0.6, ['$\Delta ...',
98 '$\varphi$\{ \approx \}', num2str(delta_t_phi), '$ s$'], ...)

```

```

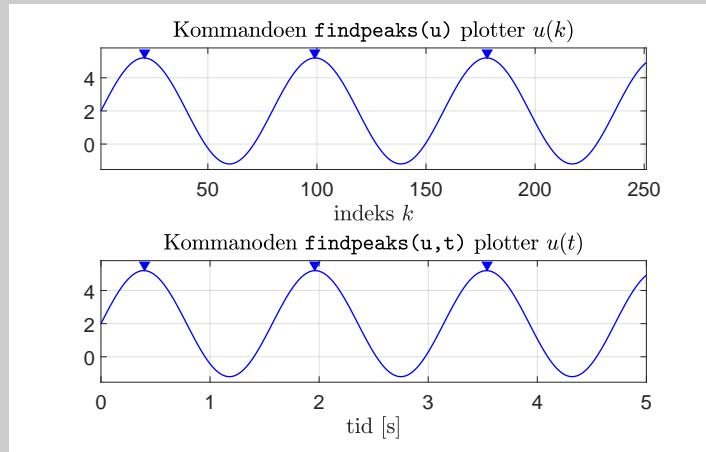
97      'backgroundcolor','white')
98 text(0.5,0.1,'$\Downarrow$','backgroundcolor','white')
99 text(0.2,-0.4,['$\varphi${\approx}',num2str(phi,3),'$'],...
100    'backgroundcolor','white')
101 text(0.5,-0.9,'$\Downarrow$','backgroundcolor','white')
102 text(0.2,-1.3,['$\varphi${\circ}{\approx}',num2str(phi_d,4),'^{\circ}$'],...
103    'backgroundcolor','white')
104
105 % Første loddrette strek
106 x = [t_pks_u(1),t_pks_u(1)];
107 y = [1,u_pks(1)];
108 plot(x,y,'k:')
109 % Andre loddrette strek
110 x = [t_pks_y(1),t_pks_y(1)];
111 y = [1,y_pks(1)];
112 plot(x,y,'k:')
113
114 % Likevektslinjen u_A
115 x = [0,t(end)];
116 y = [u_A,u_A];
117 plot(x,y,'b--')
118 text(2.5,u_A,['$u_A${\approx}',num2str(u_A)],'backgroundcolor','white')
119
120 % Likevektslinjen y_A
121 x = [0,t(end)];
122 y = [y_A,y_A];
123 plot(x,y,'r--')
124 text(1.2,y_A,['$y_A${\approx}',num2str(y_A)],'backgroundcolor','white')
125
126 % Amplituden U
127 xy1 = [t_pks_u(3),u_A];
128 xy2 = [t_pks_u(3),u_pks(3)];
129 arrow(xy1,xy2,'Ends',1,'width',1)
130 text(3.3,3.5,['$U${\approx}',num2str(U)],'backgroundcolor','white')
131
132 % Amplituden Y
133 xy1 = [t_pks_y(2),y_A];
134 xy2 = [t_pks_y(2),y_pks(2)];
135 arrow(xy1,xy2,'Ends',1,'width',1)
136 text(2.2,3.5,['$Y${\approx}',num2str(Y)],'backgroundcolor','white')
137
138 legend('$u(t)=3.2\sin(4t)+2$',...
139   'max-verdier i $u(t)$',...
140   '$y(t)=1.4\sin(4t-1.9)+3$',...
141   'max-verdier i $y(t)$',...
142   'location','southeast')
143
144 title('Signalene $u(t)$ og $y(t)$ med automatisk ...
        beregnede verdier')

```

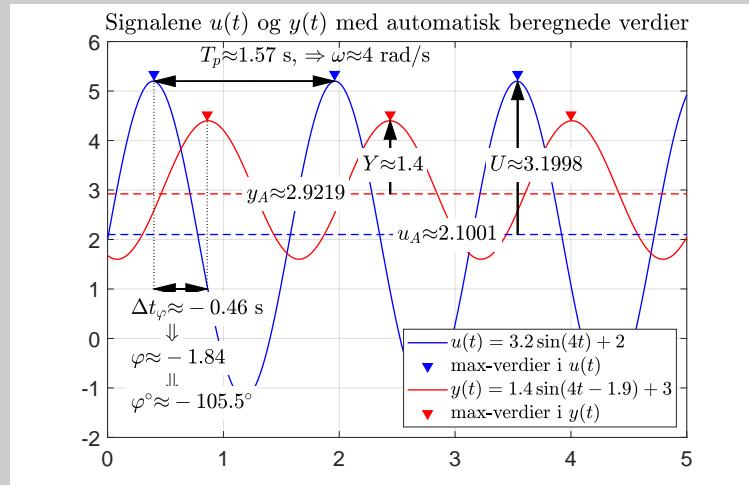
- Skjermdump av *Command Window* er gitt i figur 32.

Figur 32: Skjermdump av *Command Window*.

- Koden produserer figuren vist i figur 33 og 34.



Figur 33: Eksempel på resultat.



Figur 34: Eksempel på resultat.

- h) I denne oppgaven skal du bli mer kjent med `sin`-funksjonen, i tillegg til `pause`-funksjonen og `ctrl C` for å avbryte kjøring. Følgende tekst står i skallfilen

Kode 15: Heading i oppgave h).

```
%%
% -----
% h)
%
% Innhold:
%   - Bruk av pause til å demonstrere utvikling i for-løkker
%
% Eksempler på nyttige Matlab-funksjoner
%   - sin
%   - pause
%   - ctrl-C
%   - hold on/hold off
%   - tic og toc
%
```

Utgangspunktet for oppgaven er signalene $y_1(t)$ og $y_2(t)$ gitt som

$$y_1(t) = \sum_{n=1}^M \frac{1}{2n-1} \sin((2n-1)\cdot\omega\cdot t) \quad (17)$$

$$y_2(t) = \sum_{n=1}^M \frac{1}{n} \sin(n\cdot\omega\cdot t) \quad (18)$$

- Ferdigstill koden for $y_1(t)$ og $y_2(t)$ i `for`-løkken.
- `pause`-funksjonen uten argument venter på at du skal trykke en tast. Dersom du ikke vil trykke en tast for å få videre i koden, men heller avbryte, så må du trykke `ctrl C`. `pause`-funksjonen kan også ta et argument som angir hvor mange sekund pausen skal være. Test disse tingene ut
- Funksjonen `tic` starter en stoppeklokke, og den kan brukes både *med* og *uten* returvariabel. Den tilhørende funksjonen `toc` leser av medgått tid, og du kan avlese medgått tid så mange ganger du vil.
- Dersom du har flere parallelle stoppeklokker må du benytte en unik returvariabel på `tic`, og for å lese av medgått tid til en bestemt stoppeklokke må du benytte returvariabelen som argument til `toc()`. I skallfilen er begge disse demonstrert.
- Lek litt med øvre grense M og pausetidene, slik at du ser og opplever effekten.
- For tydeligere å se utviklingen mellom hver runde i `for`-løkka kan velge å benytte `hold on` på subplotene underveis (er i utgangspunktet kommentert bort).
- Hvor mye kortere tid, relativt sett, bruker koden når du anvender `hold on`?

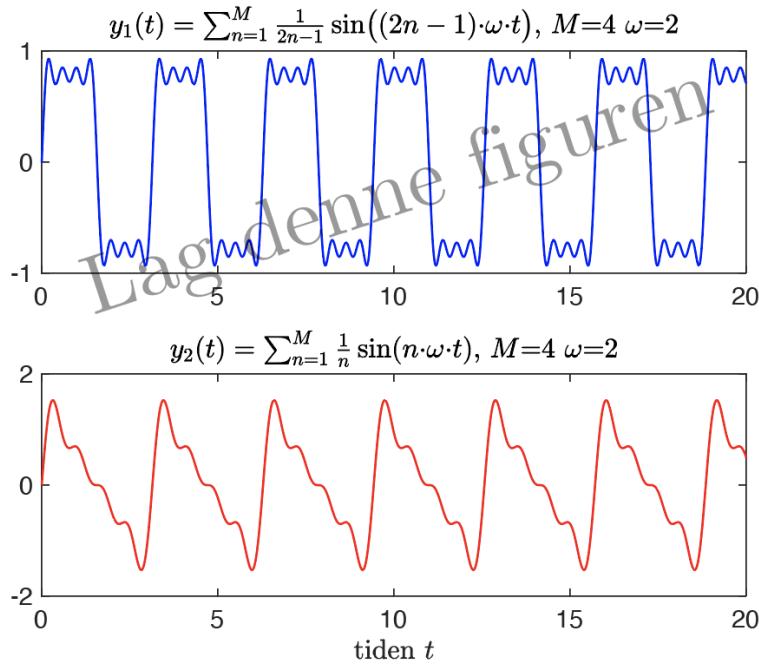
Vis at du får resultatene vist i figurene 35 - 37.

```

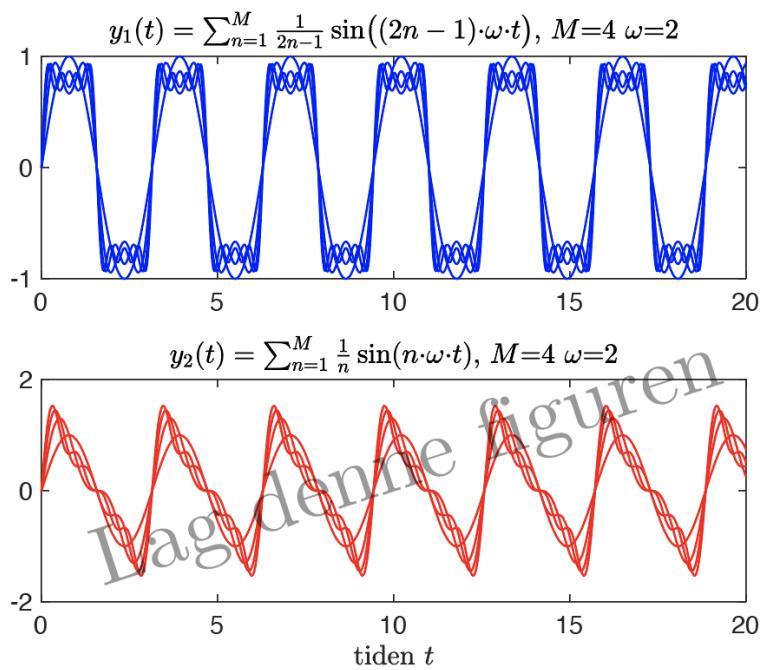
faglig > fag > ELE130_Anvendt_matematikk_og_fysikk_i_robotprogr
Command Window
New to MATLAB? See resources for Getting Started.
Runde 1 av 4
Pause. Mulig du må trykke en tast.
Pausen varte i 0.23536 sekund
Medgått tid siden start er 0.34203 sekund
Runde 2 av 4
Pause. Mulig du må trykke en tast.
Pausen varte i 0.20011 sekund
Medgått tid siden start er 0.55587 sekund
Runde 3 av 4
Pause. Mulig du må trykke en tast.
Pausen varte i 0.20073 sekund
Medgått tid siden start er 0.7658 sekund
Runde 4 av 4
Pause. Mulig du må trykke en tast.
Pausen varte i 0.20011 sekund
Medgått tid siden start er 0.97642 sekund
Hele for-løkken brukte 0.97704 sekund

```

Figur 35: Skjermdump av *Command Window*.



Figur 36: Eksempel på resultat.



Figur 37: Eksempel på resultat.

- i) I denne oppgaven skal du bli kjent med det bestemte integralet og hvordan det relaterer seg til areal under en kurve. Følgende tekst står i skallfilen

Kode 16: Heading i oppgave i).

```
%% -----
% i)
%
% Innhold:
% - Bruk av area-funksjonen
%
% Eksempler på nyttige Matlab-funksjoner
% - area
% - cos
%
```

Oppgaven tar utgangspunkt i følgende sinusuttrykk.

$$u(t) = U \cdot \sin(\omega \cdot t) + C \quad (19)$$

hvor vi i utgangspunktet spesifiserer amplituden $U=4$, $\omega=7$ rad/s og $C=0.4$.

- Med utgangspunkt i den generelle ligningen (19), finn først det analytiske uttrykket for det bestemte integralet $y(t)$ gitt som

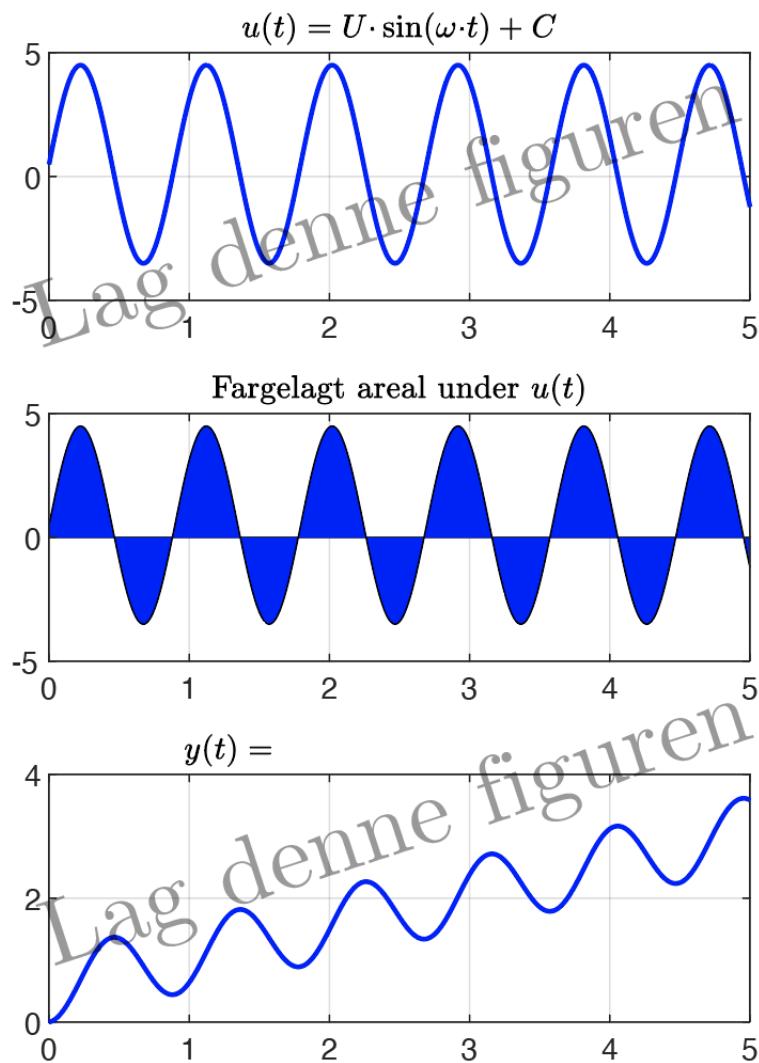
$$y(t) = \int_0^t u(\tau) d\tau \quad (20)$$

- Implementer deretter kode for både $u(t)$ og $y(t)$ i begynnelsen av for-løkken, samt kode for titlene for delfigurene. Velg passelige verdier for oppdelingen av tidsvektoren slik at kurvene blir fine. La variabelen `ant_runder = 1;`.
- Avles en vilkårlig verdi på $y(t)$ -kurven ved å bruke avlesningsverktøyet. Kall tidspunktet du leser av for t_{avlest} og integralverdien for y_{avlest} . Ta med figuren med avlesninger i innleveringen din (samme som figur 38 men med avlesning en vilkårlig plass).
- Vis matematisk at det bestemte integralet A fra uttrykket

$$A = \int_0^{t_{avlest}} u(t) dt \quad (21)$$

tilsvarer den avleste verdien y_{avlest} .

- Kan du ut fra arealbetraktingene av $u(t)$ vist i `subplot(3,1,2)` forklare hvorfor $y(t)$ stiger jevnt?



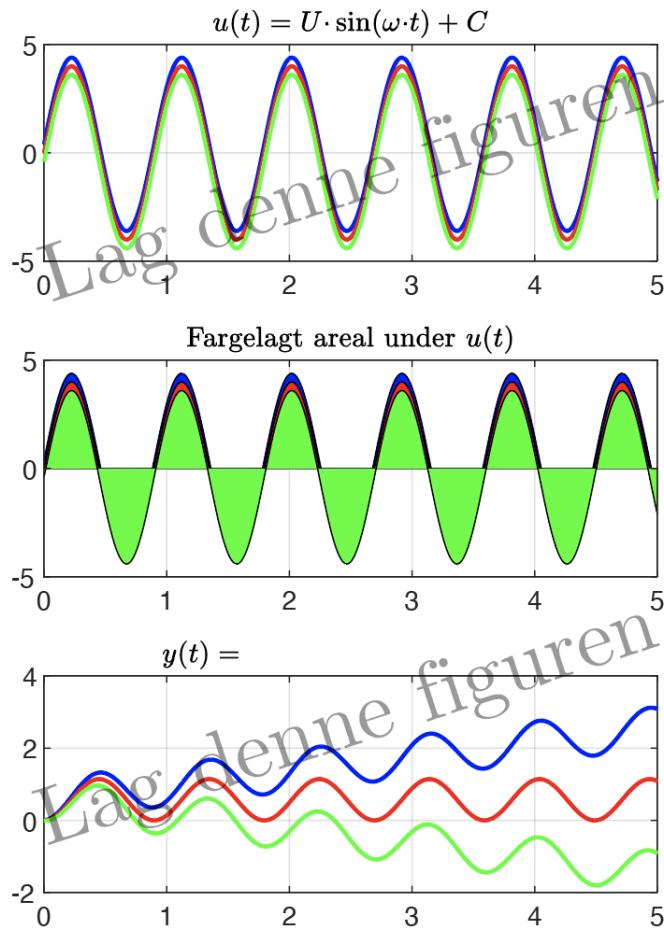
Figur 38: Figur som produseres med `ant_runder = 1;`.

- Spesifiser deretter variablene `ant_runder = 3;` og sørge for at koden nederst i for-løkken er

Kode 17: Heading i oppgave i).

```
% justering av C og/eller vinkelfrekvens w
C = C - 0.4;
%w = w - 2;
end
```

Kjør koden og vis at du får resultatet i figur 39.

Figur 39: Figur som produseres med `ant_runder = 3;` og med justeringer på C .

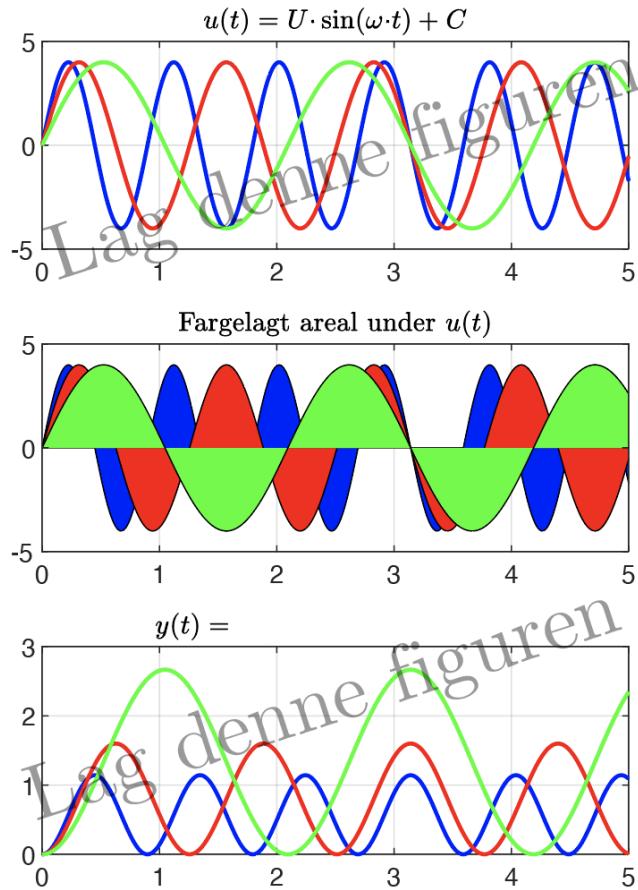
Hva er det egentlig som demonstreres i figuren?
Endres amplituen til $y(t)$ når C endres?

- Endre verdien av C fra $\boxed{C = 0.4;}$ til $\boxed{C = 0;}$ i koden før *for*-løkken. Sørg for at koden nederst i *for*-løkken er

Kode 18: Heading i oppgave i).

```
% justering av C og/eller vinkelfrekvens w
%C = C - 0.4;
w = w - 2;
end
```

Kjør deretter koden, og vis at du får resultatet i figur 40.



Figur 40: Figur som produseres med $\boxed{\text{ant_runder} = 3;}$, $\boxed{C = 0;}$ og med justeringer på ω .

Hva er det resultatet viser? Beskriv resultatet med ord. Gi gjerne svar på spørsmålet: Hvorfor skjer det endringer i amplituden til $y(t)$ når ω endres? Forklar det både ut fra arealbetraktninger og det analytisk uttrykket til $y(t)$.

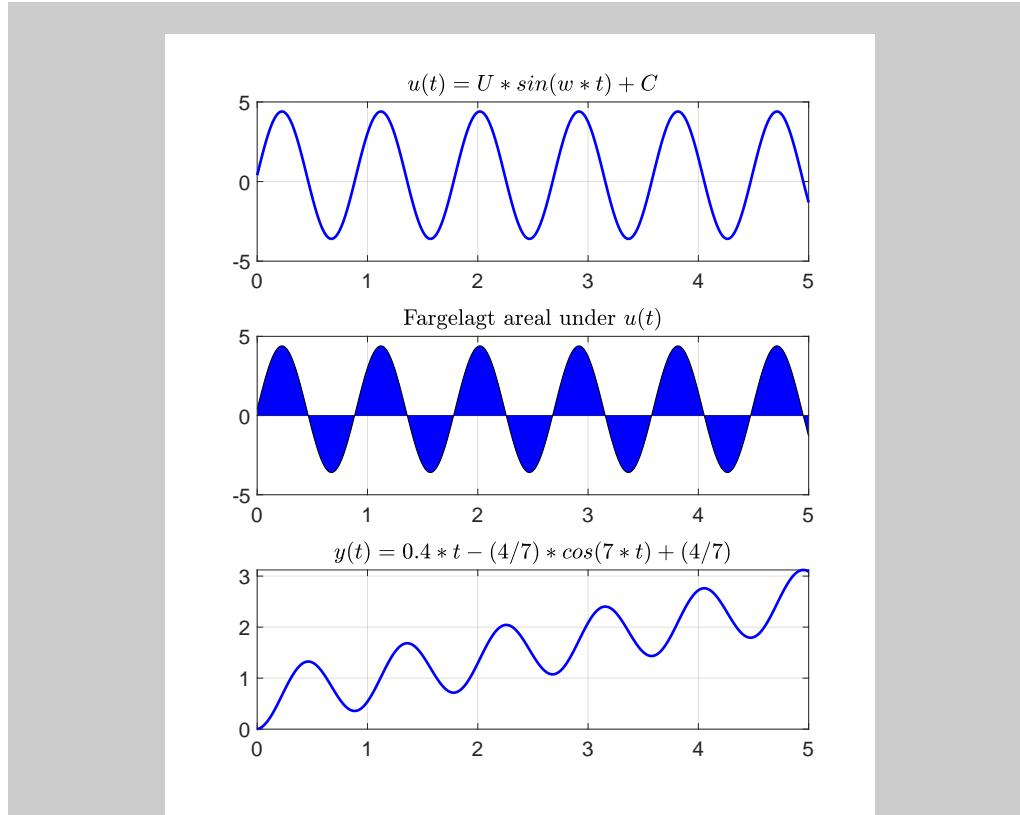
Svar:

- Forslag til løsning er gitt i kodeutdrag 19.

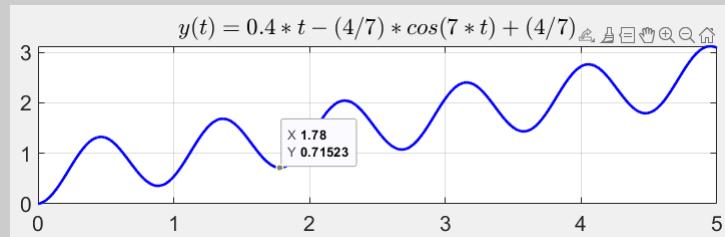
Kode 19: Kode for oppgaven.

```
1      clear; close all; clc
2
3  U = 4;
4  w = 7;
5  C = 0;
6
7  figure(1)
8  set(gcf,'position',[500 100 450 600])
9
10 % Juster på denne
11 ant_runder = 3;
12
13 for i = 1:ant_runder
14     t = 0:0.01:5;
15     u = U*sin(w*t)+C;
16     y = (-U/w) * (cos(w * t) - 1) + C * t;
17
18     subplot(3,1,1)
19     plot(t,u,'LineWidth',2)
20     hold on
21     grid
22     title('$u(t) = U*sin(w*t)+C$')
23
24     subplot(3,1,2)
25     area(t,u)
26     grid
27     hold on
28     title('Fargelagt areal under $u(t)$')
29
30     subplot(3,1,3)
31     plot(t,y,'LineWidth',2)
32     grid
33     hold on
34     title('$y(t) = (-U/w) * (cos(w * t) - 1) + C * t$')
35
36     % justering av C og/eller vinkelfrekvens w
37     % C = C - 0.4;
38     w = w - 2;
39 end
```

- Koden produserer figuren vist i figur 41.

Figur 41: Figur som produseres med `ant_runder = 1;`.

- Koden produserer figuren vist i figur 42.



Figur 42: Skjermdump av avleste verdier.

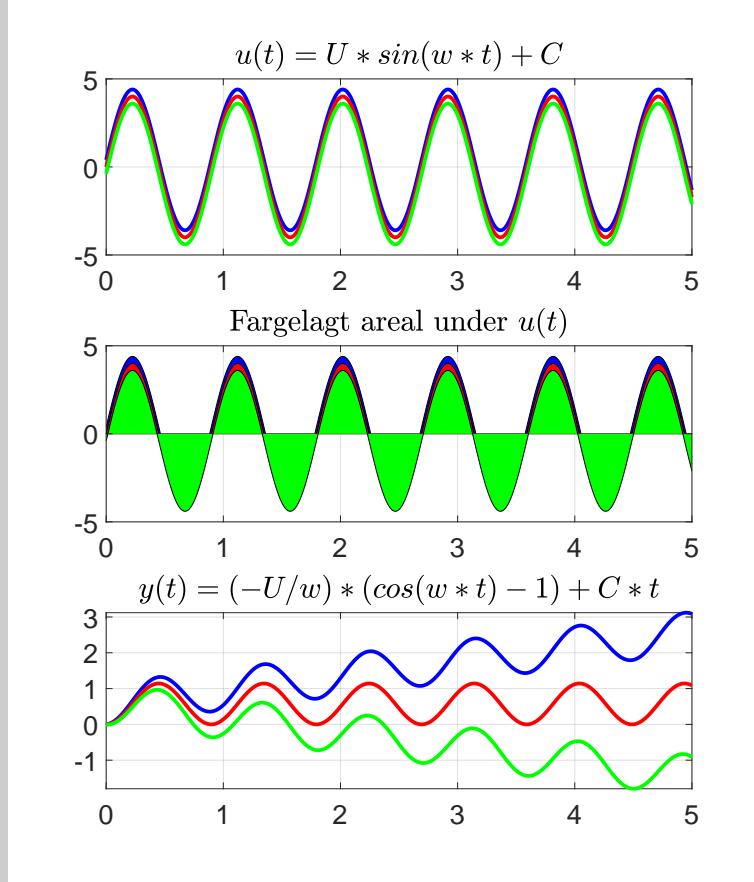
- Vi vet at det bestemte integralet av $u(t)$ fra 0 til t kan skrives som

$$y(t) = \frac{-U}{\omega} * (\cos(\omega * t) - 1) + C * t \quad (22)$$

- Ved å sette inn t_{avlest} for t kan vi beregne $y(t)$ og sammenligne med y_{avlest} .

$$y(1.78) = \frac{-4}{7} * (\cos(7 * 1.78) - 1) + 0.4 * 1.78 = 0.71523 \quad (23)$$

- $y(t)$ stiger jevnt fordi konstantleddet, C , bidrar til at arealet akkumulerer jevnt over tid. Det positive bidraget er likt for hver oscillasjon.
- Koden produserer figuren vist i figur 43.

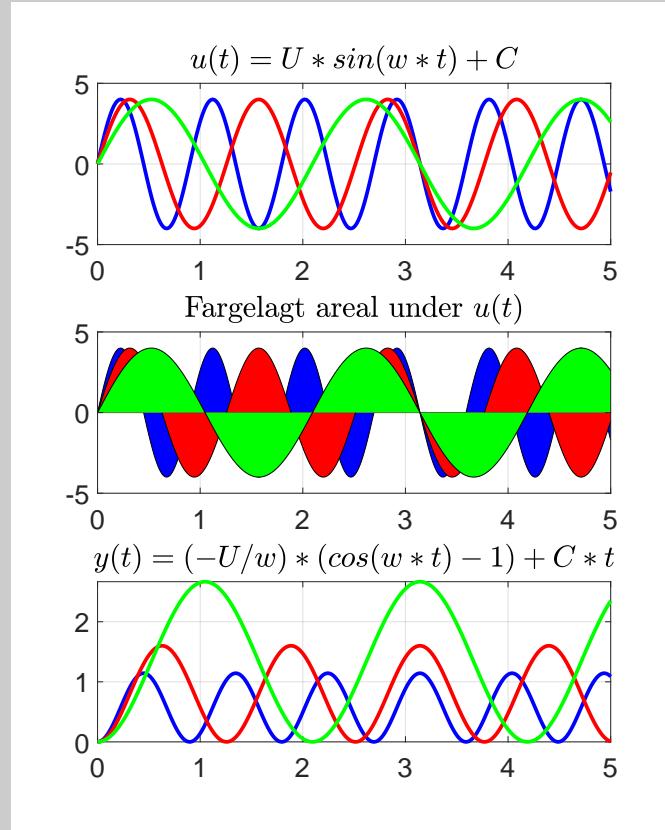


Figur 43: Figur som produseres med `ant_runder = 3;` og med justeringer på C .

- Her demonstrerer vi effekten av å endre konstantleddet. Som nevnt over er det konstantleddet som gir forskyvning på y-aksen, som igjen bidrar til akkumulert areal - positivt eller negativt. Fordi vi nå kjører for-løkken 3 runder med $C = C - 0.4$, endrer vi effektivt konstantleddet for hver

iterasjon, og dermed også $u(t)$ og $y(t)$. Utslaget kan bl.a. sees i det nedste subplottet: For hver iterasjon reduseres C og dermed det akkumulerete arealet.

- Koden produserer figuren vist i figur 44.



Figur 44: Figur som produseres med `ant_runder = 3;`, `C = 0;` og med justeringer på ω .

- Fordi $u(t)$ er en sinusfunksjon vil endringer i w gi utslag i hvor raskt $u(t)$ oscillerer. ω reduseres nå for hver iterasjon, altså lavere frekvens. Dette gir økt bølgelengde.
- Når ω reduseres påvirkes amplituden av de oscillerende delene av $y(t)$.
- Arealbetraktnng: Når bølgelengden til $u(t)$ øker akkumuleres det større areal under grafen mellom hver oscillasjon.

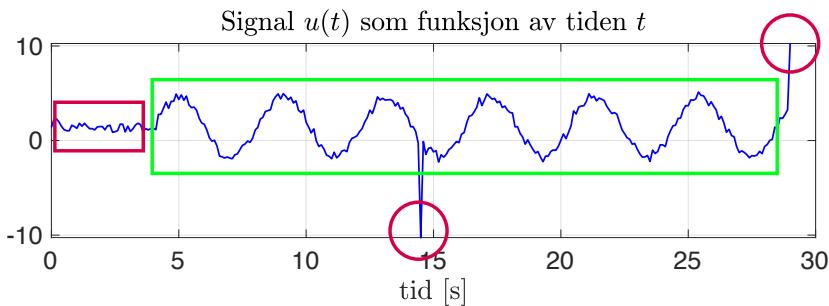
- Analytisk: Amplituden er omvent proporsjonal med vinkelfrekvensen, gitt ved $\frac{-U}{w}$ i ligning 22.

- j) I denne oppgaven skal du bli kjent med teknikker for å “vaske” et signal. Følgende tekst står i skallfilen

Kode 20: Heading i oppgave j).

```
%% -----
% j)
%
% Innhold:
% - Fjerne uteliggere, konstantledd og deler av et signal
% - Relevant for Lego-prosjektet
%
% Eksempler på nyttige Matlab-funksjoner
% - randi, rand og randn
% - zeros
% - NaN
% - findpeaks med MinPeakDistance-argument
%
```

I Legoprosjektet skal du blant annet lage sinussignal med lyssensoren, og du vil ofte ende opp med et signal som vist i figur 45, hvor signalet har en tilnærmet konstant verdi i begynnelsen (rødt rektangel), og en eller flere feilaktige måleverdier (røde ringer). Slike feilaktige målinger kalles uteliggere (eng: *outliers*).



Figur 45: Et typisk resultat fra Lego-prosjektet.

Signalet som vi typisk er interessert i er rammet inn i det grønne rektangelet, og ser vi bort fra at signalet er forsinket, er signalet prinsipielt gitt som

$$u(t) = U \sin(\omega t) + C \quad (24)$$

I denne oppgaven skal du “vaske” (eller justere om du vil) signalet slik at du kun står igjen med et “rent” signal som ikke er forsinket (som starter i $t=0$) og som svinger omkring $C = 0$, formulert som

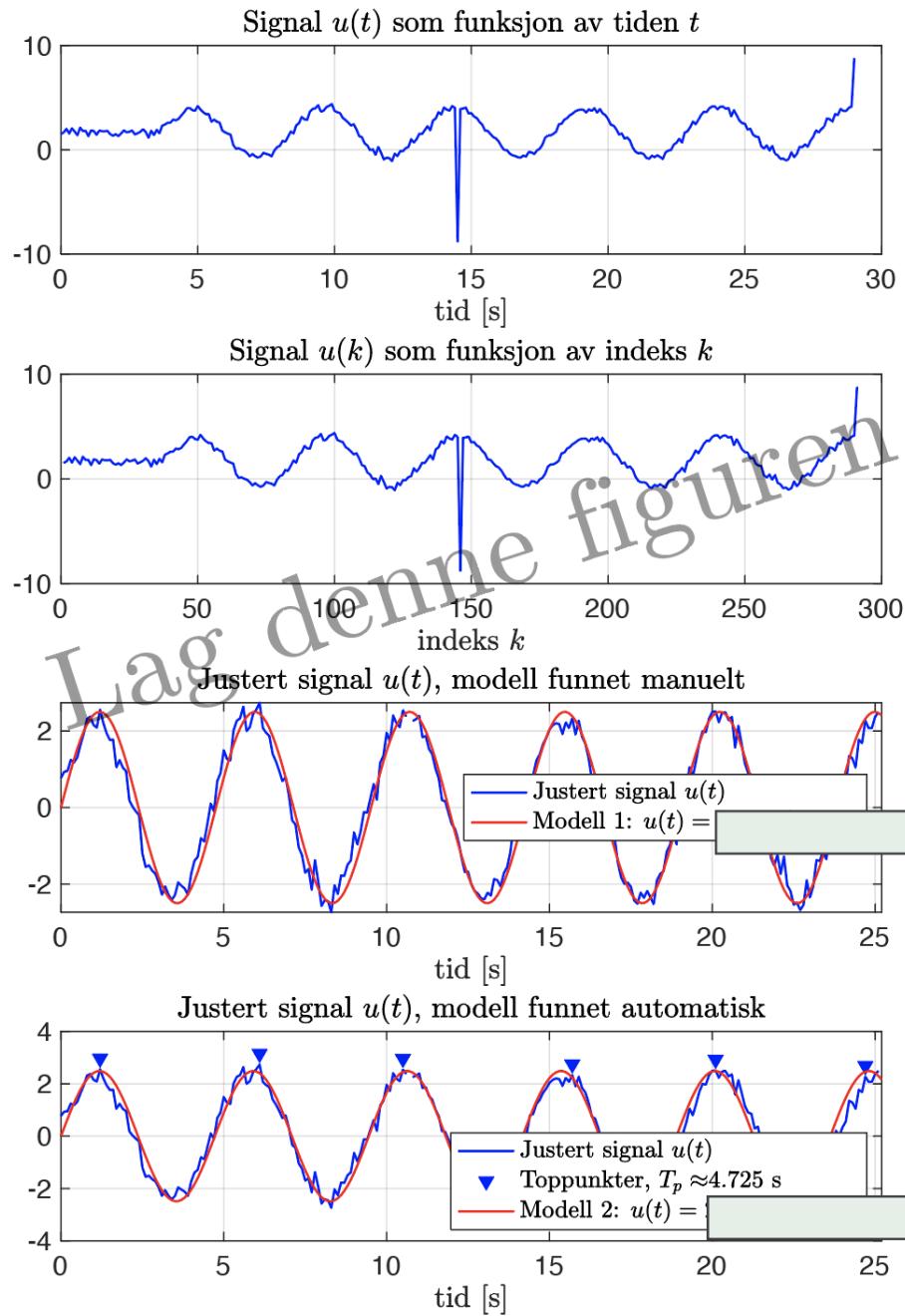
$$u(t) = U \sin(\omega t) \quad (25)$$

Du skal gjøre denne justeringen på et tilfeldig sinussignal, og du skal gjøre det både manuelt og automatisk. Målet er altså å fjerne det konstante delen i begynnelsen, uteliggerne og likeveksverdien C .

Når Legoprosjektet begynner kan du anvende denne teknikken på dine egenproduserte sinussignal. Gjør følgende i skallfilen `oving1_skallfil.m`:

- Lag ferdig koden som setter inn k stykk 0'ere i begynnelsen av `u_delay`.
- Henskten med figuren i `subplot(4,1,2)` er du lett skal kunne lese av/estimere ved hvilken indeks (k -verdi) sinussignalet starter på. Alternativt måtte du brukt figuren i `subplot(4,1,1)`, men denne gir deg ikke indeksinformasjon direkte. Skriv den avleste k -verdien inn i linjen som begynner med `k0 =` .
- Fullfør koden for å fjerne de k_0 første elementene i både `t` og `u`, samt på en smart måte justere tidsvektoren på slik at `t(1)=0`.
- Avgjør hvilke grenseverdier `u_max` og `u_min` du vil benytte for å fjerne ute-liggere. Alle verdier større enn grenseverdien settes lik `NaN`, som betyr *Not A Number*.
- Finn et estimat av likevektslinjen `C_est`, gjerne ved bruk av en ligning som automatisk finner verdien ut fra signalet `u`. Deretter plottes det justerte signalet `u` mot tiden i `subplot(4,1,3)`.
- Du skal videre estimere hvilket sinussignal du har fått i i `subplot(4,1,3)` ved prøve- og feilemetoden. For å finne amplituden `U_est` og vinkelfrekvensen `w_est` kan du enten
 - lese av amplituden direkte fra kurven i `subplot(4,1,3)`, samt lese av perioden T_p og beregne `w_est`
 - eller bare velge verdier for `U_est` og `w_est` og kjøre koden om og om igjen til kurvene overlapper.
- Til slutt skal du implementere en metode som estimerer amplituden `U_est` og vinkelfrekvensen `w_est` automatisk. Metoden benytter funksjonen `findpeaks` med argumentet `MinPeakDistance` slik at den ikke identifiserer topper forårsaket av støy som ligger tett etter hverandre.

Ferdigstill koden og vis at du får følgende figur, hvor `seed = 1`. Legg merke til at vi har maskert de manuelt avleste og de automatisk beregnede verdiene. Det er ingen eksakt fasitsvar for de manuelt avleste verdiene.

Figur 46: Resultat for denne oppgaven, hvor `seed = 1`.