MATLAB OPPGAVER I MATEMATIKK 1 (IGR1600)
HØST 2016

UIT THE ARCTIC UNIVERSITY OF NORWAY

Irina Pettersson, UiT.

Innhold

1. Vektorer 3
2. Komplekse tall 5
3. Funksjoner og grafer 10
4. Derivasjon 15
5. Ekstremalpunkter 20
6. Newtons metode 22
7. Integrasjon 26
8. Differensiallikninger 29
9. Matriser. Lineære likningssett. 33
10. Matriser. Egenvektorer og egenverdier. 34
11. Fasit 38

1. Vektorer

Eksempel 1.1. Regn ut prikkprodukt $\vec{a}\cdot\vec{b}$, der $\vec{a}=\vec{i}+2\vec{j}+\vec{k}$ og $\vec{b}=2\vec{i}-\vec{j}+\vec{k}$.

Løsning. Vi definerer først de to vektorene i Matlab kode og deretter regner vi ut skalarproduktet mellom dem. Man kan bruke definisjonen av prikkproduktet og regne ut det direkt eller anvende en innbygd funksjon `dot`. For å skrive ut svaret fint utnytter vi `fprintf`. Symbolet `\%f` viser at vi skal skrive ut en variabel av **floating-point number** type (variabelen PRODUCT1 i vårt tilfelle), mens `\n` skifter linje. Bruk gjerne `help` for å finne ut mer om `fprintf`.

```
% Input: två vektorer
% Output: prikkprodukt
% Definer de to vektorene
a = [1, 2, 1]
b = [2, -1, 1]
% Regn ut prikkproduktet direkt
PRODUCT1 = a(1)*b(1) + a(2)*b(2) + a(3)*b(3);
% Eller ved hjelp av "dot"-funksjon
PRODUCT2 = dot(a,b);
% Skriv ut svaret
fprintf('Skalarproduktet er lik %f\n', PRODUCT1);
```

□

Eksempel 1.2. Regn ut skalart trevektorprodukt $\vec{a}\cdot(\vec{b}\times\vec{c})$, der $\vec{a}=\vec{i}+2\vec{j}+\vec{k}$, $\vec{b}=2\vec{i}-\vec{j}+\vec{k}$ og $\vec{c}=2\vec{i}-2\vec{j}+\vec{k}$.

Løsning. Vi kan bruke definisjonene av prikk- og kryssproduktet:

```
% Input: tre vektorer
% Output: trevektorprodukt
% Definer tre vektorer
a = [1 2 1]
b = [2 -1 1]
c = [2 -2 1]
% Regn ut cryssproduktet av b og c dikrekt
PROD = [(b(2)*c(3)-b(3)*c(2))...
        (b(3)*c(1)-b(1)*c(3))...
        (b(1)*c(2)-b(2)*c(1))];
% Regn ut prikkrpoduktet av resultatet med vektor a
ScalarTripleProduct = a(1)*PROD(1)+a(2)*PROD(2)+a(3)*PROD(3);
% Skriv ut svaret
fprintf('Trevektorproduktet = %d\n', ScalarTripleProduct);
```

OBS! Legg merke til at vi bruker "...for å skrive et langt utrykk på flere linjer.

En annen mulighet er å bruke de innebygde funksjonene `dot` og `cross`:

```
% Input: tre vektorer
% Outpit: trevektorprodukt
% Definer tre vektorer
a = [1 2 1]
b = [2 -1 1]
c = [2 -2 1]
% Regn ut trevektorprodukt
PROD = dot(a, cross(b,c));
% Skriv ut svaret
fprintf('Trevektorproduktet = %d\n', PROD);
```

□

Oppgaver

1. Lag en MATLAB kode som regner ut prikkproduktet mellom to vektorer og bruk den for å kontrollere svaret i opp. 4.3.1.
2. Lag et program som tar imot en vektor fra brukeren og gir en enhetsvektor i samme retning.

   Hint: google `input` funksjon.
3. Lag et program som tar imot to vektorer fra brukeren og finner en enhetsvektor som står vinkelrett på begge to.

   Hint: Dersom du vil output vektorer eller matriser, er det enklere å bruke `disp` enn `fprintf`.
4. Lag et program som tar imot tre vektorer fra brukeren og regner ut volumet av parallellepipedet dannet av de tre vektorene.
5. 

   Lag en MATLAB kode som tar imot tre vektorer fra brukeren og avgjør om de tre vektorene ligger i samme plan eller ikke.

2. Komplekse tall

Gitt et komplekstall $z = a + b\,i$, der i er den imaginære enheten, vi kan

- Regne ut modulen og hovedargumentet til z ved hjelp av `abs(z)` og `angle(z)`.
- Finne real- og imaginærdelen til z ved hjelp av `real(z)` og `imag(z)`.
- Finne det konjugerte komplekse tallet $\overline{z}$ ved hjelp av `conj(z)`.
- Finne **numerisk** nullpunkter til et polynom C med komplekse koeffisienter ved hjelp av `roots(C)`.
- Utføre **symbolske** utregninger: forenkle, forkorte, utføre delbrøkspalting, løse likninger, osv..

Ta en titt i `help` for å finne ut mer om disse funksjonene.

Eksempel 2.1. Regn ut $(-2 + 3i/4)^2 * (i - 5)$.

Løsning. Det finnes to måter å gjøre det: numerisk og symbolsk.
Numerisk:

```
>> (-2+3i/4)^2*(1i-5)

ans =

 -14.1875 +18.4375i
```

Numerisk løsning viser svaret i **default format**, som er

> ...the short, fixed-decimal format for floating-point notation and loose line spacing for all output lines.

Symbolsk:
Noen gang vil man ha et eksakt uttrykk, da setter man beregningen inn som argument for funksjonen `sym`.

```
>> sym((-2+3i/4)^2*(1i-5))

ans =

- 227/16 + (295*i)/16
```

Vil du sjekke svar til oppgavene i boka, er det lurt å bruke symbolsk utregning. □

Nederfor finner du noen eksempler som viser hvordan man løser likninger med komplekse koeffisienter både numerisk og symbolsk.

Eksempel 2.2. Lag en MATLAB-kode som løser likningen

$$z^2 + (1 + i)z - \frac{i}{2} = 0.$$

Løsning.
Numerisk

Fra `help`:

> roots(C) computes the roots of the polynomial whose coefficients are the elements of the vector C. If C has $N + 1$ components, the polynomial is $C(1) * X^N + ... + C(N) * X + C(N + 1)$.

La oss se nærmere på likningen vi har. Vi vil finne nullpunkter (roots) av polynomet $z^2 + (1 + i)z - \frac{1}{2}i$. Dette er et andregradspolynom (så $N = 2$) med komplekse koeffisienter

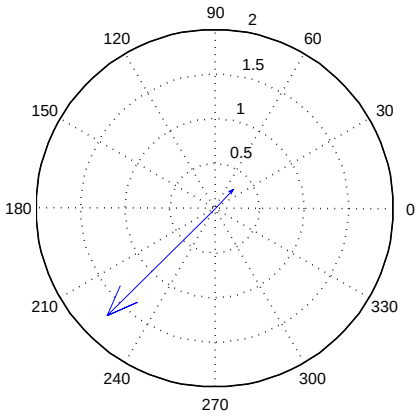
$$C(1) = 1, \quad C(2) = 1 + i, \quad C(3) = -\frac{i}{2}.$$

OBS: Den imaginære enheten i skriver vi som $1i$ i MATLAB.

```
% Løser z^2+(1+i)z - i/2 = 0
% Input: Koeffisienter til et polynom
% Output: Nullpunkter til polynomet
% Definerer en vektor av koeffisientene
C = [1 1+1i -0.5i];
% Regner ut nullpunktene
Z = roots(C)
% Tegner løsningene
compass(Z)
```

Output:

```
ans =
    -1.2071 - 1.2071i
     0.2071 + 0.2071i
```



FIGUR 1. Løsningene til $z^2 + (1 + i)z - i/2 = 0$.

Symbolsk

```
% Løser z^2+(1+i)z - i/2 = 0
% Definerer en symbolsk variabel z
syms z
% Løser likningen symbolsk.
% Resultatet er en 2-vektor Z (to løsninger)
Z = solve(z^2 + (1+1i)*z -1i/2==0)
% Forenkler uttrykket ved å regne ut real-
% og imaginærdelene for hver løsning
Z1 = real(Z(1))+1i*imag(Z(1))
Z2 = real(Z(1))+1i*imag(Z(1))
```

Output:

```
Z =
   (4*i)^(1/2)/2 - 1/2 - i/2
 - (4*i)^(1/2)/2 - 1/2 - i/2
Z1 =
    2^(1/2)*(1/2 + i/2) - 1/2 - i/2
Z2 =
    2^(1/2)*(1/2 + i/2) - 1/2 - i/2
```

Legg merke til at den første løsningen Z vi får er resultatet av ABC-formelen. Vi er ikke helt fornøyde med tallet $(4*i)^{1/2}$, det skal regnes ut. Derfor separerer vi real- og imaginærdelen og får de to ferdige løsninger $Z1$ og $Z2$. MATLAB grupperer leddene som den selv synes er best, så vil man ha en løsning på formen $a+ib$, skal man bruke `real(Z)` og `imag(Z)` separat.

□

Eksempel 2.3. Lag en MATLAB-kode som løser likningen

$$z^3 = -8i.$$

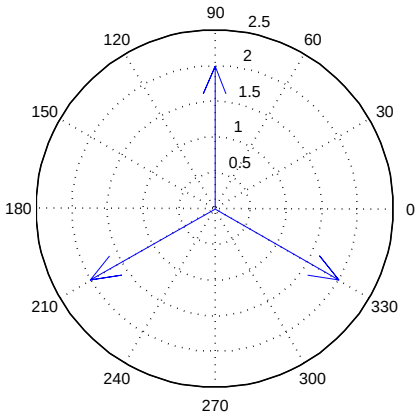
Løsning.

Numerisk:

Vi skriver om likningen til et tredjegradspolynom: $z^3 + 8i = 0$. Koeffisientene er $C(1) = 1, C(2) = 0, C(3) = 0, C(4) = 8i$. Da er det bare å bruke `roots` for å regne ut nullpunktene.

```
% Løser z^3 = -8i
% Input: koeffisienter til et polynom
% Output: nullpunkter til polynomet
% Definerer en vektor av koeffisientene
C = [1 0 0 8i];
% Regner ut nullpunktene
roots(C)
% Tegner nullpunktene
compass(roots(C))
```

Output:



FIGUR 2. Løsningene til $z^3 = -8i$.

```
ans =
    -1.7321 - 1.0000i
    0.0000 + 2.0000i
    1.7321 - 1.0000i
```

Symbolsk:

```
>> syms z
>> solve(z^3==-8i)

ans =
           2*i
   3^(1/2) - i
 - 3^(1/2) - i
```

Løsningen til denne delen er så kort at det er enklest å taste inn likningen direkte i Command Window.

□

Les mer om symbolsk MATLAB i Tore Gaupseths kompendium.

OPPGAVER

1. Trykk inn tallene

$$1 + \sqrt{-4} \quad \text{og} \quad \frac{5}{8} + \sqrt{-\frac{7}{64}}$$

i MATLAB.

2. Lag en MATLAB-kode som regner ut $(1+i)^2 \cdot (-3+2i)$. Du kan også bruke denne koden for å kontrollere svar i opp. 4.4.4.

3. Lag en MATLAB-kode som finner den reelle og den imaginære delen av $(1+i)^2(-3+2i)$. Sammenlikn svaret med det i den forrige oppgaven.

4. Løs likningene:

(a) $z^4 = 16$
(b) $z^2 - 2iz - \sqrt{3}i = 0$.

Du kan bruke denne koden til å kontrollere svar i opp. 4.4.17 og 4.4.18.

5. 

Lag en MATLAB-kode som tar imot to komplekse tall fra brukeren og forteller om produktet av de to tallene er et realtall eller ikke.

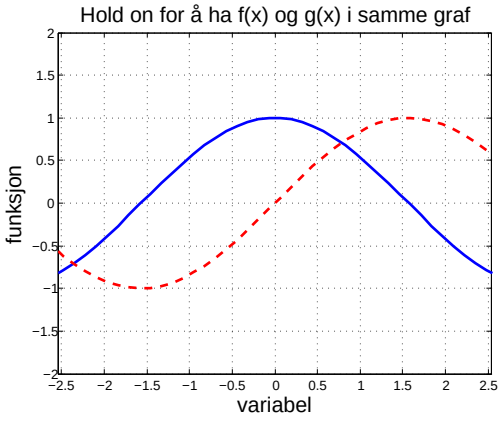
3. Funksjoner og grafer

MATLAB er et bra grafisk verktøy. Vi skal bruke de enkleste funksjonene `plot` og `plot3` for å tegne kurver både i 2D og 3D. Flere eksempler finner du i MATLAB Documentation.

Eksempel 3.1. Tegn $f(x) = \cos(x)$ og $g(x) = \tan(x)$ i samme graf.

Løsning.

```
close all;
% x vektor: 100 punkter mellom -2pi og 2pi
x = linspace(-2*pi, 2*pi);
% Plot f(x) i blått
plot(x, cos(x), 'b', 'LineWidth', 2)
% La den stå
hold on
% Plot g(x) i rødt (dashed line)
plot(x, sin(x), '--r', 'LineWidth', 2)
grid on
axis equal
% Vi begrenser y-verdier til [-2, 2]
ylim([-2, 2])
xlabel('variabel', 'FontSize', 16)
ylabel('funksjon', 'FontSize', 16)
title('Hold on for å ha f(x) og g(x) i samme graf', 'FontSize', 16)
```



Figur 3. Grafene i Eksempel 3.1.

Mer om `linspace` kan du lese i `help` eller her.
Mer om `Line Specifications`: her. □

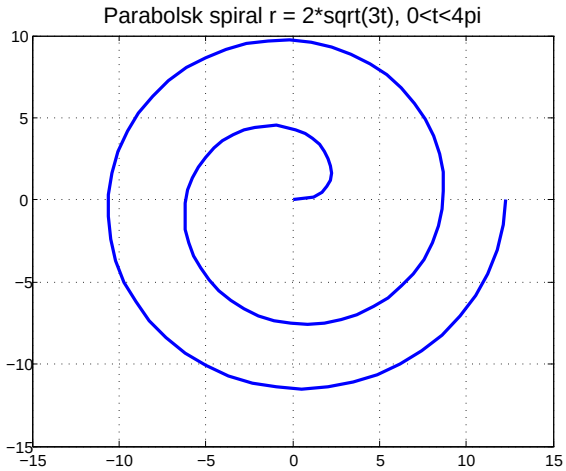
I neste eksempel viser vi hvordan man kan tegne en polar kurve, det vil si en kurve gitt ved $r = r(\theta)$, der r er polarradius og θ er polarvinkel. En måte er å bytte til kartesiske koordinater $x = r\cos(\theta), y = r\sin(\theta)$ og bruke `plot(x,y)`. En annen måte er å bruke `polarplot`. Vi viser den første måten.

Eksempel 3.2. Tegn en *parabolsk spiral*

$$r = 2\sqrt{3\theta}, \quad \theta \in [0, 4\pi].$$

Løsning.

```
% Tegner en parabolsk spiral r = 2*sqrt(3*t), 0 < t < 4pi.
close all;
% t - vektor med lengde 100
t = linspace(0, 4*pi);
% Definerer polar radius r=r(t)
r = 2*sqrt(3*t);
% Byter til kartesiske koordinater
x = r.*cos(t);
y = r.*sin(t);
% Tegner kurven
plot(x,y, 'LineWidth', 2)
grid on
title('Parabolsk spiral r = 2*sqrt(3t), 0<t<4pi', 'FontSize', 14)
```



FIGUR 4. Parabolsk spiral i Eksempel 3.2.

□

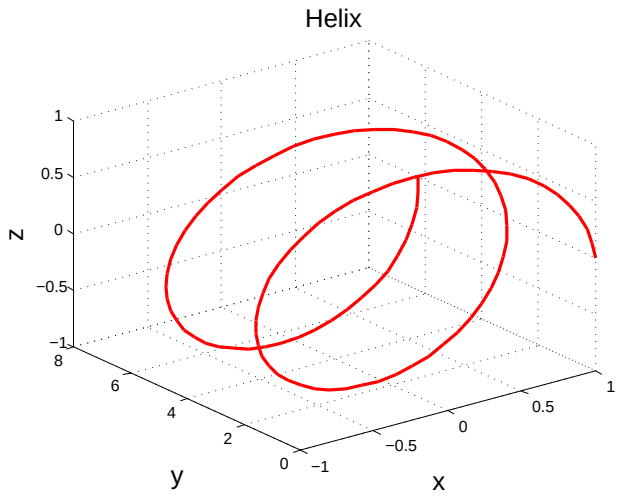
Kurver i 3D tegnes med `plot3`. Les mer om `plot3` her.

Eksempel 3.3. Tegn en romkurve

$$x(t) = \cos(2\pi t), \quad y(t) = \pi t, \quad z(t) = \sin(2\pi t), \quad t \in [0, 2].$$

Løsning.

```
% Tegner en romkurve r(t)=[cos(2pi*t), pi*t, sin(2pi*t)]
% 0<t<2
t = linspace(0, 2);
% Definerer x(t), y(t), z(t)
xt = cos(2*pi*t);
yt = pi*t;
zt = sin(2*pi*t);
% Tegner grafen
plot3(xt,yt,zt, 'r', 'Linewidth', 2)
grid on
xlabel('x', 'FontSize', 16)
ylabel('y', 'FontSize', 16)
zlabel('z', 'FontSize', 16)
title('Helix', 'FontSize', 16)
```



FIGUR 5. Romkurven $r(t) = [\cos(2\pi t), \pi t, \sin(2\pi t)], t \in [0, 2]$ i Eksempel 3.3.

□

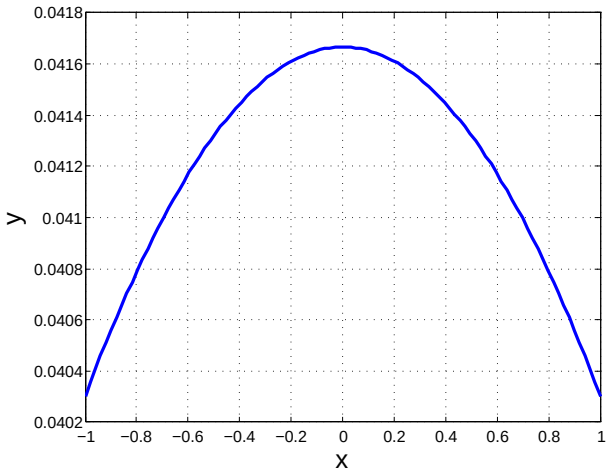
Eksempel 3.4. Gitt

$$f(x) = \frac{\cos x - 1 + x^2/2}{x^4},$$

anslå grenseverdien $\lim_{x\to 0} f(x)$ ved å tegne grafen $y = f(x)$. Beregn siden grenseverdien symbolsk i MATLAB og kontroller svaret.

Løsning.

```
% Tegner f= (cos(x) - 1 + x^2/2)/x^4
close all;
% x-vektor
x = linspace(-1, 1);
% Definerer y
y = (cos(x)-1+x.^2/2)./x.^4;
% Tegner
plot(x, y, 'LineWidth', 2)
grid on
xlabel('x', 'FontSize', 16)
ylabel('y', 'FontSize', 16)
% Regner ut symbolsk grenseverdi når x -> 0
syms t
LIMIT = limit((cos(t)-1+t^2/2)/t^4, t, 0);
disp('Grenseverdi lim_{x -> 0} (cos(x)-1+x^2/2)/x^4 =');
disp(LIMIT)
```



FIGUR 6. $f(x) = \dfrac{\cos x - 1 + x^2/2}{x^4}, x \in [-1, 1]$ i Eksempel 3.4.

Output:

```
Grenseverdi lim_{x -> 0} (cos(x)-1+x^2/2)/x^4 =
1/24
```

Vi ser på grafen at grenseverdien ligger mellom 0.0416 og 0.0418 ($1/24 = 0.0417$). □

OPPGAVER

1. Tegn $f(x) = -x^2 + 2x + 2$ og vis at $f(x)$ har minst ett nullpunkt i $[-1, 1]$.
2. Tegn linjen mellom to punkter $A(2, -1, 0)$ og $B(4, 4, -3)$.
3. Tegn romkurven

$$x(t) = 2t, \quad y(t) = t^2, \quad z(t) = t^3, \quad 0 \leq t \leq 3.$$

4. Tegn en kurve i polar koordinater $r = 8/(5 - 3\cos\theta)$, $\theta \in [0, 2\pi]$.

5. Tegn $f(x) = \dfrac{\sin x}{x}$ i MATLAB, regn ut (symbolsk) grenseverdien

$$\lim_{x \to 0} \frac{\sin x}{x}$$

og sjekk om svaret stemmer med grafen. Les mer om hvordan man regner en grenseverdi symbolsk i Tore Gaupseths kompendium.

4. Derivasjon

I MATLAB finnes det muligheter for både numerisk (diff) og symbolsk derivasjon (symbolsk diff).

For eksempel, for å derivere $f(x) = (3x^2 + 5)^4$ symbolsk, definerer vi en symbolsk variabel x og bruker deretter `diff`.

```
>> syms x
>> diff((3*x^2 + 5)^4)

ans =

24*x*(3*x^2 + 5)^3
```

Du kan bruke denne oppskriften for å kontrollere svar i derivasjonsoppgaver.

Eksempel 4.1. Sett opp likninger for tangenten og normalen til kurven $y = (x-1)^2$ i punktet $(2, 1)$. Bruk MATLAB til å beregne den deriverte $y'(x)$. Tegn grafen til $y = y(x)$, tangenten og normalen i ett og samme koordinatsystem.

Løsning.

Når funksjonen er gitt eksplisitt, kan man selvsagt regne ut den deriverte for hånd eller symbolsk ved hjelp av MATLAB. I dette eksemplet viser vi hvordan man regner ut (approksimerer) den deriverte numerisk ved å bruke definisjonen. Ideen er å erstatte $f'(x_0)$ med en lineær approksimasjon

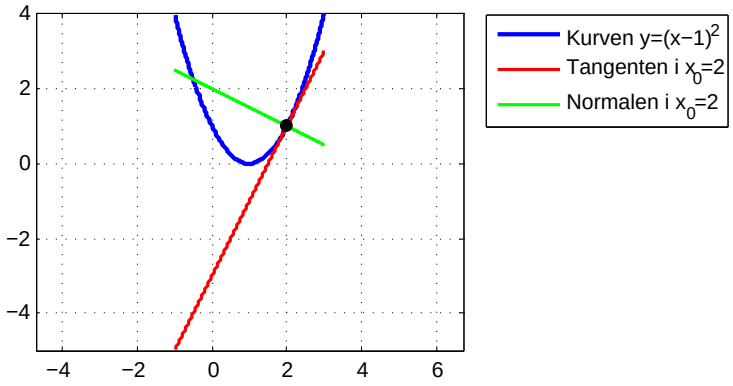
$$f'(x_0) \approx \frac{f(x_0 + h) - f(x)}{h}.$$

Vi sier da at vi har *diskretisert* f'. Det finnes et lite teknisk problem: en diskretisert x-vektor (linja 5 i koden nederfor: `x = -1:h:3`) inneholder ikke nødvendigvis x_0. Så vi må finne et punkt i x som ligger nærmest x_0. For å gjøre det, tar vi alle punkter i x og ser på avstanden til x_0: $abs(x - x_0)$. Avstanden blir minimal for det punktet som ligger nærmest. Variabelen `val` gir verdien av $abs(x - x_0)$, mens `idx` gir posisjonen av det nærmeste til x_0 punktet i x (les mer om funskjonen `min` i Documentation).

Fordeler med en sånn framgangsmåten er at man ikke er avhengig av innbygde MATLAB funksjoner og ikke er nødt til å regne ut deriverte eksplisit, som er spesielt viktig når vi har en komplisert sammensatt funskjon.

```
1  % Input: en funksjon, et punkt
2  % Output: plot av funskjonen, tangenten og normalen
3  close all
4  h = 0.001;          % skrittlengde
5  x = -1: h:3;        % variabel
6  x0=2;               % et gitt punkt
7  f = (x-1).^2;       % en gitt funskjon
8  plot(x, f, 'LineWidth', 2); % tegner f
9  hold on; grid on;axis equal;
10 % Finner punktet i x nærmest x0=2
11 [val, idx] = min(abs(x-x0))
12 y0 = f(idx) % funksjonens verdi i det punktet
13 % Approksimasjon av den deriverte y'(x0)
```

```
df = (f(idx+1)-f(idx))/h;
% Tangenten y=y0 + y'(x0)*(x-x0)
T = y0 + df*(x-x(idx));
% Normalen y = y0 -1/y'(x0)*(x-x0)
N = y0 - (x-x(idx))/df;
plot(x, T, 'r', 'LineWidth', 1.5) % Tegner tangenten i rødt
plot(x, N, 'g', 'LineWidth', 1.5) % Tegner normalen i grønt
plot(x0,y0,'.k', 'MarkerSize', 20) % Markerer (x0, y0)
legend('Kurven y=(x-1)^2', 'Tangenten i x_0=2',...
       'Normalen i x_0=2',...
        'location', 'NorthEastOutside');
```



FIGUR 7. Eksempel 4.1.

Output:

```
val =
     0
idx =
      3001
y0 =
     1
```

□

Eksempel 4.2. La $f(x) = e^{-x}$. Tegn grafen til $y = f(x)$ og grafen til et polynom P av grad mindre eller lik 2 med

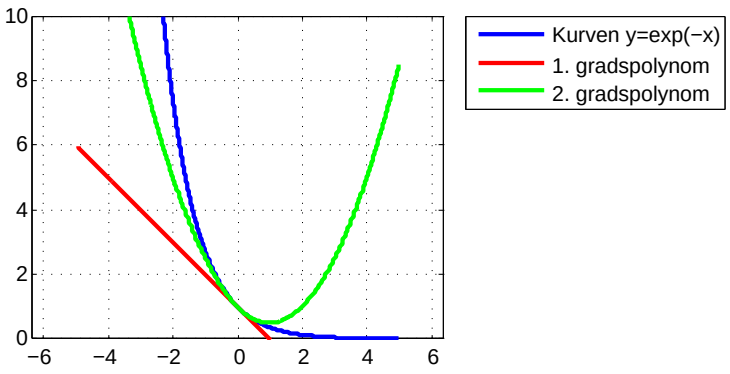
$$P^{(k)}(0) = f^{(k)}(0), \quad k = 0, 1, 2.$$

Dette eksemplet viser hvordan man approksimerer en funksjon med et polynom i et gitt punkt. Det er artig å se hvordan grafen til polynomet nærmer seg funskjonens graf når vi øker polynomets grad.

Løsning.

```
% Approksimerer f=exp(3x) med et polynom i x=0
close all;
h = 0.0001; % skrittlengde
```

```
x = -5:h:5;
x0 = 0;
f = exp(-x);
plot(x, f, 'LineWidth', 2); % tegner f
xlim([-5, 5]); ylim([0, 10]);
hold on; grid on;axis equal;
% Finner punktet i x nærmest x0=2
[val, idx] = min(abs(x-x0))
y0 = f(idx) % funksjonens verdi i det punktet
% Approksimasjon av den deriverte f'(0)
df = (f(idx+1)-f(idx))/h;
% Approksimasjon av den andre deriverte f"(0)
ddf = (f(idx+1) - 2*f(idx)+f(idx-1))/h^2;
% Polynom p=C1x+C0, p(0)=f(0) og p'(0)=f'(0)
C0 = f(idx); % C0 = tilnærmet f(0)
C1= df; % C1 = tilnærmet f'(0)
% Polynom av første grad
P1 = C1*x + C0;
% Skriver ut P1
fprintf('P1 = %fx + %f', C1, C0);
plot(x, P1, 'Color', 'r', 'LineWidth', 2); % Tegner P1
% Polynom p=C2* x^2 + C1*x+C0: p(0)=f(0), p'(0)=f'(0)
% og p"(0) = f"(0)
C0 = f(idx); % C0 = tilnærmet f(0)
C1 = df; % C1 = tilnærmet f'(0)
C2 = ddf/2; % C2 = tilnærmet f"(0)
% Polynom av andre grad
P2 = C2*x.^2 + C1*x + C0;
% Skriver ut P2
fprintf('\nP2 = %f x^2 + %fx + %f', C2, C1, C0);
plot(x, P2, 'Color', 'g', 'LineWidth', 2); % Tegner P2
legend('Kurven y=exp(-x)', '1. gradspolynom',...
        '2. gradspolynom','location', 'NorthEastOutside');
```



FIGUR 8. Eksempel 4.2.

I dette eksemplet regner vi ut den deriverte numerisk. Utregningen er basert på definisjonen av den deriverte:

$$f'(x_0) \approx \frac{f(x_0+h) - f(x_0)}{h}.$$

Problemet igjen er at x ikke nødvendigvis inneholder $x0 = 0$ (den gjør det likevel i vårt tilfelle pga at vi starter i -5 og velger $h = 0.0001$). Så vi skal finne et punkt i x som ligger nærmest x_0. For å gjøre det, tar vi alle punkter i x og ser på avstanden $abs(x - x_0)$. Avstanden blir minimal for det punktet som ligger nærmest. Variabelen `val` gir verdien av $abs(x - x_0)$, mens `idx` gir posisjonen av det nærmeste til x_0 punktet i x (les mer om funskjonen `min` i Documentation). Som du ser i **Output**, `val`= 0 (viser at x_0 er i x), og `idx`= 50001 (element nr.50001 i x). Tenk at du tar utgangspunktet i -5 og går med skritt 0.0001, hvor mange steg skal du ta for å komme til 0?

Nå kan vi regne ut funskjonens verdi $f(idx)$ og approksimere den deriverte:

$$f'(idx) \approx \frac{f(idx+1) - f(idx)}{h}.$$

Husk at f er også en vektor, med samme lengde som x. Når vi skriver $f(idx)$, får vi den komponenten av f som tilsvarer elementet i x med nummer idx (det nærmeste punktet til $x_0 = 0$). Derimot gir $f(idx + 1)$ funskjonens verdi i neste punkt, $x_0 + h$.

Den andre deriverte $f''(x_0)$ approksimerer vi som følge:

$$f''(x_0) \approx \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{h^2}.$$

Nå har vi de første to deriverte og kan konstruere polynomene. Et polynom $P(x)$ slik at

$$P(x_0) = f(x_0), P'(x_0) = f'(x_0), P''(x_0) = f''(x_0), \cdots P^{(n)}(x_0) = f^{(n)}(x_0)$$

kalles Taylorpolynomet av orden n. Det finnes bare ett sånt polynom:

$$P(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n.$$

Det betyr at koeffisientene til $P(x)$ kan beregnes i termer av deriverte av f.

I vårt eksempel, $x_0 = 0$. For 1. gradspolynomet $C1x + C0$, må vi ha

$$C0 = f(0), C1 = f'(0).$$

For 2. gradspolynomet $C2x^2 + C1x + C0$, har vi

$$C0 = f(0), C1 = f'(0), C2 = \frac{f''(0)}{2}.$$

Legg merke til at `df` og `ddf` er skalare (ikke vektorer som `f`), derfor skriver vi ikke noe argument (som i `f(idx)`).

Output:

```
val =
     0
idx =
       50001
y0 =
     1
P1 = -0.999950x + 1.000000
P2 = 0.500000 x^2 + -0.999950x + 1.000000
```

□

1. Bruk MATLAB for å finne første og andrederiverte av $f(x) = \cos\sqrt{2+x^4}$ symbolsk.
2. Gitt $f(x) = \dfrac{-2x^2+1}{x^2+3}$, sett opp likninger for tangenten og normalen til grafen i punktet $x_0 = 3$. Tegn grafene til funskjonen, tangenten og normalen i ett og samme koordinatsystem.
3. Gitt funksjonen $f(x) = \ln x$, regn ut de første fem deriverte for hånd eller ved hjelp av MATLAB symbolic. Tegn grafene til funksjonen og taylorpolynomene av orden 1 til 5 om punktet $x = 1$.
4. La $f(x) = \cos(x/2)$. Sett opp likningene til polynom P av grad 1 og 2 med
$$P^{(k)}(1) = f^{(k)}(1), \quad k = 1, 2, 3.$$
Tegn grafene til funskjonen og de to polynomene i ett og samme koordinatsystem.
5. 

La $f(x) = \sin(\cos(x))$. Sett opp likningen til Taylorpolynom P av grad 3 med
$$P^{(k)}(1) = f^{(k)}(1), \quad k = 0, 1, 2, 3.$$
Tegn grafene til funskjonen og Taylorpolynomet i ett og samme koordinatsystem.

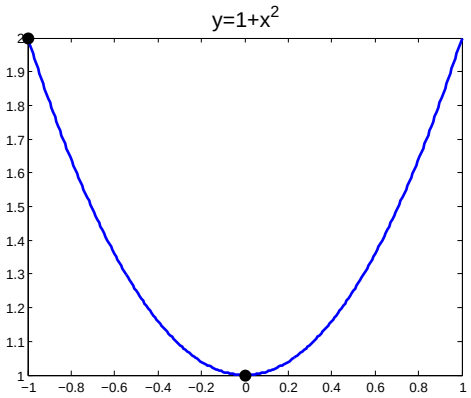
5. Ekstremalpunkter

Funksjonene `max` og `min` returnerer det største og det minste elementet i en tabell (array). Les mer om max og min i **help**.

Eksempel 5.1. Finn maksimal og minimal verdier til $f(x) = 1 + x^2$, $x \in [-1, 1]$. Har du fått alle maksimalpunkter? Om ikke, kan du endre koden slik at den gir alle maksimalpunkter?

Løsning.

```
% Input: en funksjon f(x)
% Output: grafen og ekstremalpunkter
close all;
% x varierer fra -1 til 1
x = -1:2^(-7):1;
% Definerer f= 1+x^2
f = 1 + x.^2;
% maxval= maksimalverdi, imax=posisjon i x
[maxval, imax] = max(f)
% minval=minimalverdi, imin=posisjon i x
[minval, imin] = min(f)
% Tegner y=f(x)
plot(x, f, 'LineWidth', 2);
hold on
% Markerer ekstremalpunktene
plot(x(imax), maxval, '.k', 'MarkerSize', 30);
plot(x(imin), minval, '.k', 'MarkerSize', 30);
title('y=1+x^2', 'FontSize', 16);
% Skriver ut max og min
fprintf('max=(%f, %f), min=(%f, %f)', ...
        x(imax), maxval, x(imin), minval);
```



Figur 9. $y = 1 + x^2$ og ekstremalpunkter.

Output:

```
maxval =
     2
imax =
     1
minval =
     1
imin =
   129
max=(-1.000000, 2.000000), min=(0.000000, 1.000000)
```

Vi har fått bare ett av to maksimalpunkter (det andre er $x = 1$). Grunnen til det er at `max` returnerer bare det første elementet som gir maksimum. I vårt tilfelle er $imax = 1$ og $x_{max} = -1$.

Kan du endre koden slik at den returnerer alle maksimalpunkter?
Hint: En mulighet er å bruke `find`.

□

Oppgaver

1. Bruk koden fra Eksempel 5.1 til å finne maksimal- og minimalverdi av $f(x) = \dfrac{-2x^2+1}{x^4+3}$, $x \in [-1, 2]$.

6. Newtons metode

Newtons metode er en metode til å løse likninger på formen $F(x) = 0$. Med andre ord, metoden gir en tilnærming til nullpunktene til funskjonen F. Hvis likningen har mer enn en løsning, gir metoden én løsning av gangen.

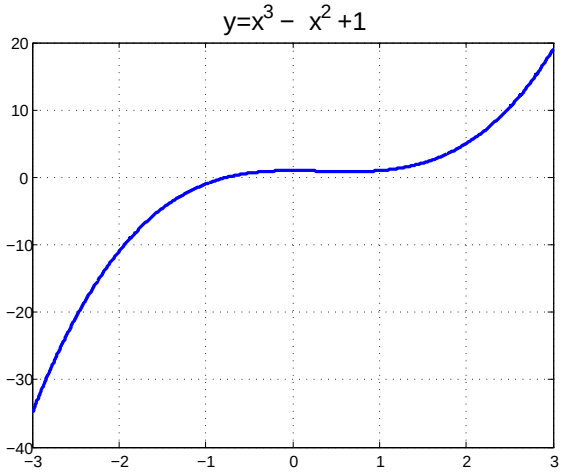
En tilnærmet løsning x_n til likningen $F(x) = 0$ er gitt ved rekursjonsformelen

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, 2, \cdots .$$

Før man setter i gang med rekursjon, lønner det seg å tegne grafen til $y = F(x)$. Det kan hjelpe å velge et punkt nære et nullpunkt, som kommer til å gi en bedre approximasjon.

Eksempel 6.1. Finn en løsning av likningen $x^3 - 17x + 1 = 0$ ved bruk av Newtons metode.

Løsning. Først spør man seg hvor mange nullpunkter har $F(x) = x^3 - 17x + 1$. Vi ser nærmere på grafen i Figur 10. Det ser ut som om $F(x)$ har bare ett nullpunkt



Figur 10. $y = x^3 - x^2 + 1, x \in [-3, 3]$ i Eksempel 6.1.

som ligger mellom -1 og 0. Vi gir et bevis for at det stemmer:

1. F er en kontinuerlig funskjon, $F(-1) = -1$ og $F(0) = 1$, så $F(x)$ må ha minst ett nullpunkt i intervalet $(-1, 0)$ (skjæringssetningen).

2. $F'(x) = 3x^2 - 2x = 3x(x - 2/3)$, så funskjonen F er stigende når $-1 < x < 0$ og

$$F'(x) > 0 \quad \text{for} \quad x < 0 \quad \Rightarrow \quad F \quad \text{er en stigende funskjon,}$$

$$F'(x) < 0 \quad \text{for} \quad 0 < x < \frac{2}{3} \quad \Rightarrow \quad F \quad \text{er en fallende funskjon,}$$

$$F'(x) > 0 \quad \text{for} \quad x > \frac{2}{3} \quad \Rightarrow \quad F \quad \text{er en stigende funskjon.}$$

Det betyr at det ikke finnes flere nullpunkter når $x < 0$ eller $x > 2/3$. Fordi $F(0), F(2/3) > 0$ og funskjonen er fallende, kan det ikke heller finnes flere nullpunkter i intervallet $[0, 2/3]$.

La oss velge startpunktet for rekursjonen $x_0 = -0.5$. Newtons metode gir for

$$n = 0 : \qquad x_1 = -0.5 - \frac{F(0.5)}{F'(0.5)},$$

$$n = 1 : \qquad x_2 = x_1 - \frac{F(x_1)}{F'(x_1)}, \cdots .$$

Vi tar $n = 2$ og håper å få en god approksimasjon.

```
% Input: funksjon F(x)=x^3-x^2+1
% Output: grafen til y=F(x),
%         nullpunkter til F ved Newtons metode
close all;
h = 0.0001; % skrittlengde
% Lager x vektor
x = -2:h:2;
% Definerer F
F = x.^3-x.^2 + 1;
% Tegner y=F(x)
plot(x, F, 'LineWidth', 2);
grid on;
title('y=x^3 -  x^2 +1', 'FontSize', 16);
% Velger statpunkt for Newtons metode
x0 = -0.5;
% Finner et punkt i x vektor nærmest x0
[val0, idx0] = min(abs(x-x0));
% Approksimerer F'(x0)
dF0 = (F(idx0+1)-F(idx0))/h;
% Første steg i Newtons metode
x1 = x(idx0) - F(idx0)/dF0;
% Finner et punkt i x nærmest x1
[val1, idx1] = min(abs(x-x1));
% Approksimerer F'(x1)
dF1 = (F(idx1+1)-F(idx1))/h;
% Andre steg i Newtons metode
x2 = x1 - F(idx1)/dF1;
% Finner et punkt i x vektor nærmest x2
[val2, idx2] = min(abs(x-x2));
% Approksimerer F'(x2)
dF2 = (F(idx2+1)-F(idx2))/h;
% Tredje steg i Newtons metode
x3 = x(idx2) - F(idx2)/dF2;
% Skriver ut x_n
fprintf('Newtons metode gir:\n x0=%f, x1=%f, x2=%f, x3=%f \n ',...
        x0, x1, x2, x3);
% Nullpunkter til F(x) ved hjelp av roots
C = [1 -1 0 1];
disp('Nullpunkter av F gitt av "roots":'); roots(C)
```

Output:

```
Newtons metode gir:
     x0=-0.500000, x1=-0.857194, x2=-0.764132, x3=-0.754962
Nullpunkter av F gitt av "roots":
ans =
   0.8774 + 0.7449i
   0.8774 - 0.7449i
  -0.7549 + 0.0000i
```

Koden gir oss en tilnærmet verdi $x \approx -0.7549$. Stopper vi her, kan vi garantere en løsning -0.7549 med fire sikre desimaler ($x2$ og $x3$ samsvarer så langt).

Koden kan skrives kortere om man anvender `while`- eller `for`- løkke. For eksempel, om vi vil sikre fem desimaler, kan vi bruke `while`:

```
% Input: funksjon F(x)=x^3-x^2+1
% Output: løsning av F=0 med 5 sikre desimaler
close all;
h = 0.0001; % skrittlengde
% Lager x vektor
x = -2:h:2;
% Definerer F
F = x.^3-x.^2 + 1;
% Velger statpunkter for Newtons metode
x0 = 0; x1 = -0.5;
% Iterasjonssteg
k = 0;
while (abs(x0-x1)>0.00001)
    x0 = x1;
    k = k+1;
    % Finner et punkt i x vektor nærmest x0
    [val0, idx0] = min(abs(x-x0));
    % Approksimerer F'(x0)
    dF0 = (F(idx0+1)-F(idx0))/h;
    % Første steg i Newtons metode
    x1 = x(idx0) - F(idx0)/dF0;
    % Skriver ut x1
    fprintf('x%d=%f \n ', k, x1);
end
```

Output:

```
x1=-0.857194
x2=-0.764138
x3=-0.754962
x4=-0.754878
x5=-0.754878
```

□

Oppgaver

1. Finn en tilnærmet verdi for π med 8 sikre desimaler ved å bruke Newtons metode til å løse likningen $\cos x = 1/2$.

2. Vis at funksjonen $F(x) = \sin x - x - 1$ har nøyaktig ett nullpunkt, og bruk Newtons metode til å finne en tilnærmet verdi for det punktet med tre sikre desimaler.

3. 

Bruk tre trinn av Newtons metode til å bestemme radien i den største sirkelen det er plass til mellom x-aksen og grafen til $y = e^{-|x|}$.

7. Integrasjon

MATLAB kan regne integraler både numerisk og symbolsk. For eksempel, for å beregne antiderivert (symbolsk) til $f(x) = 1 + x^2$ skriver vi

```
>> syms x
>> f = 1+x^2;
>> int(f)

ans = (x*(x^2 + 3))/3
```

For å regne ut et bestemt integral $\int_0^1 (1 + x^2)dx$ symbolsk skriver vi

```
>> syms x
>> f = 1+x^2;
>> int(f, 0, 1)

ans =  4/3
```

MATLAB kan også utføre delbrøkspalting (prøv gjerne funskjonen `partfrac`). En måte er å lure MATLAB: integrere brøken og siden derivere resultatet. For eksempel, til å delbrøkoppspalte $\dfrac{x^5 + x^3 + 1}{x^3 - 7x + 6}$, skriver vi

```
>> syms x
>> f = (x^5 + x^3 +1)/(x^3 -7*x+6);
>> diff(int(f))
ans =
    41/(5*(x - 2)) - 3/(4*(x - 1)) - 269/(20*(x + 3)) + x^2 + 8
```

Symbolsk integrasjon fungerer bra når man har en funksjon i hånden. Et problem oppstår når man har et diskret datasett i stedet for en eksplisit funksjon. Da er vi nødt til å bruke numeriske integrasjonsmetoder.

Eksempel 7.1. En bil beveger seg rett østover. Farten er målt hver minutt og er gitt i tabell

tid	0	1	2	3	4	5	6	7	8	9	10	11	12
fart	0	30	35	47	52	49	80	82	82	79	80	78	85
tid	13	14	15	16	17	18	19	20	21	22	23	24	25
fart	61	63	62	62	30	33	37	34	34	60	65	70	73

Her er tiden målt i minutter og farten i km/t. Bruk trapesmetoden til å anslå hvor langt bilen har kommet i løpet av de 25 minuttene.

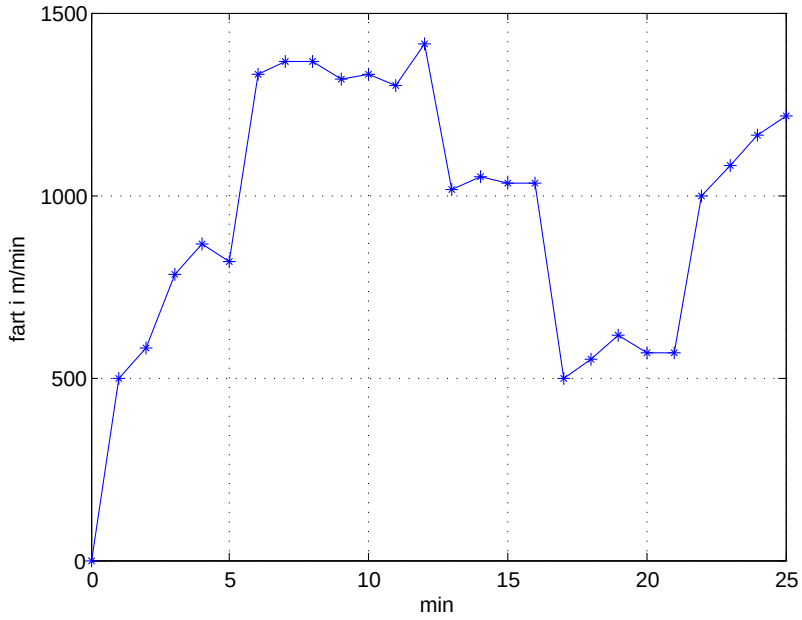
Løsning.

Vi begynner med å tegne data. Farten skal vi oversette i meter per minutt (gange med 100/6).

```
fart=[0 30 35 47 52 49 80 82 82 79 80 78 85 61 63 62 62 30 33
      37 34 34 60 65 70 73]*100/6;
tid = 0:25;
```

```
>>plot(tid, fart, '-*')
>> grid on
>> xlabel('min')
>> ylabel('fart i m/min')
```

Resultatet er vist i Fugur 11.



FIGUR 11. Eksempel 7.1.

Til å regne ut den totale avstanden bilen har kjørt, vil vi helst ta et integral av farten med hensyn på tiden fra 0 til 25 (minutter). Men fartens verdier er gitt som et diskret sett, slik at vi kan bare bruke numeriske metoder og finne en tilnærming til avstanden. Det finnes flere metoder for numerisk integrasjon. Vi skal benytte trapesmetoden.

La oss beskriver kort hvordan trapesmetoden fungerer. Anta at vi vil beregne det bestemte integralet $\int_a^b f(x)dx$. Vi deler intervallet $[a, b]$ i n like store deler med bredde $(b-a)/n$. Delingspunktene har abscissene $a = x_0, x_1, \cdots, x_{n-1}, x_n = b$. En tilnærmet verdi for integralet er gitt ved

$$\int_a^b f(x)dx \approx \frac{b-a}{2n}(f(x_0) + 2f(x_1) + \cdots + 2f(x_{n-1}) + f(x_n)).$$

Du kan lese mer om trapesmetoden for numerisk integrasjon i tekstboka. Her viser vi hvordan metoden kan implementeres i MATLAB.

```
1 % Input: fart data for en bil, tidsintervall
2 % Output: Plot av data, avstanden bil har kjørt
3 close all; clear all;
4 % Farten målt hver minutt i m/min
5 fart=[0 30 35 47 52 49 80 82 82 79 ...
```

```
6         80 78 85 61 63 62 62 30 33 37 ...
7         34 34 60 65 70 73]*100/6;
8  % Tid i min
9  a = 0; b = 25; n = 25;
10 tid = a:(b-a)/n:b;
11 % Tegner data
12 plot(tid, fart, '-*');
13 grid on;
14 xlabel('min');
15 ylabel('fart i m/min');
16 % Regner ut avstanden med trapesmetode med n=25
17 % fart(1)+2fart(2)+2fart(3)+...+2fart(25)+fart(26)
18 % = 2*sum(fart)-fart(1)-fart(26)
19 avstand = ((b-a)/(2*n))*(2*sum(fart)-fart(1)-fart(n+1))
```

Output:

```
avstand = 2.3775e+04
```

Legg merke til at det første elementet i tabellen `fart` er `fart(1) = 0`, og det siste er `fart(26)=73*100/6` (for $n = 25$!).

Nå, når vi vet hvordan trapesmetoden fungerer, kan vi sjekke svaret vi får ved hjelp av en innbygd funskjon `trapz` som utgjør diskret integrasjon ved hjelp av trapesmetoden (les mer om trapz):

```
>>trapz(tid, fart)
ans = 2.3775e+04
```

□

Oppgaver

1. Bruk MATLAB symbolic til å kontrollere svar i minst fem integrasjonsoppgaver fra tekstboka (velg de vanskeligste).
2. Bruk MATLAB til å delbrøkoppspalte
$$\frac{100(x^3+2x+8)}{x^5+7x^4+18x^3+14x^2-15x-25}.$$
3. Beregn de bestemte integralene $\int_0^1(2x+1)dx$ og $\int_0^1 \cos x dx$ ved hjelp av trapesmetoden med $n = 50$. Regn ut disse integralene eksakt/symbolsk og sammenlikn svar.
4. 

En sirkelskive av radius $R = 10$ cm er laget av et materiale med massetettheten $\delta(r) = 10 + r^2\,\mathrm{kg/m}^2$, der r er avstanden fra skivens sentrum målt i meter. Regn ut skivens masse ved hjelp av trapesmetoden.

   Hint: Massen av sirkelringen med tykkelse Δr i avstand r fra sentrum, er omtrent lik $\delta(r) \times$ arealet av sirkelringen. Legger du sammen alle ringene, får du den riemannsummen som approksimerer den totale massen.

8. Differensiallikninger

MATLAB kan løse differensiallikninger både numerisk og symbolsk. Vi skal se på hvordan vi kan løse symbolsk førstegrads differensiallikninger og lineære differensiallikninger med konstante koeffisienter. Du kan bruke kodene i denne kapittelen til å kontrollere svar i oppgavene fra Kapittel 9.2-9.5 i tekstboka.

Eksempel 8.1. Bestem løsningen til differesiallikningen

$$2y'' - 5y' + 2y = 0, \quad y(0) = -3,\ y'(0) = 0.$$

Løsning.

En måte er å gå gjennom alle steg for hånd: lage den karakteristiske likningen, løse den, lage allmenn løsning og siden bruke intialbetingelsene til å finne konstantene. Du finner koden nederfor.

```
% Input: differentiallikning 2y"-5y'+2y=0
% Output: løsningen (symbolsk) til likningen
clear all; close all;
% Definerer en symbolsk variabel
syms r
% Løser den karakteristiske likningen
l = solve(2*r^2 -5*r+2, r)
% Allmenn løsning
syms t A B
y = A*exp(l(1)*t) + B*exp(l(2)*t)
% Setter t=0 inn i y og i y' for å finne A og B
Const = solve(subs(y, t, sym(0))+3, ...
        subs(diff(y,t), t, sym(0))-0, A, B);
% Skriver ut konstantene A og B
C1 = Const.A
C2 = Const.B
% Setter inn konstantene A og B i løsningen
y  = subs(y, {A, B}, {C1, C2})
% Tegner løsningen
x = linspace(0,10,100);
plot(x, subs(y,t,x), 'LineWidth', 2)
grid on;
```

Output:

```
l =
   2
 1/2

y = A*exp(2*t) + B*exp(t/2)

C1 = 1
C2 = -4
y = exp(2*t) - 4*exp(t/2)
```

En annen måte å løse en differentiallikning symbolsk er å bruke `dsolve`. Les gjerne mer om denne funksjonen i `help`.

```
% Input: likningen 2y"-5y'+2y=0,
%                    y(0)=-3, y'(0)=0
% Output: løsningen til likningen

% Definerer en symbolsk funksjon
syms y(t)
% Første derivert
Dy = diff(y);
% Andre derivert
D2y = diff(y,2);
% Løser likningen
dsolve(2*D2y -5*Dy+2*y == 0, y(0) == -3, Dy(0) == 0)
```

Output:

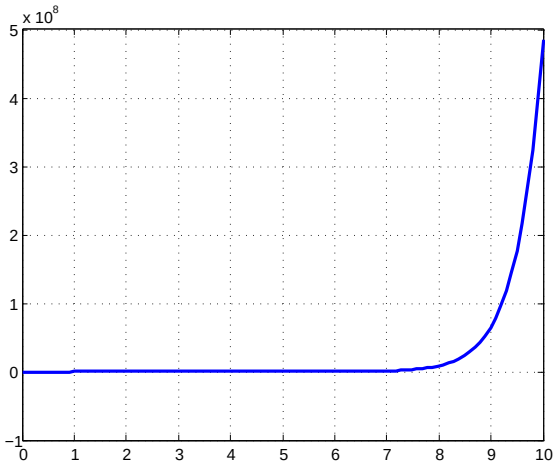
```
ans = exp(2*t) - 4*exp(t/2)
```

Du liker sikkert den andre løsningen best: den er mye kortere. Fordelen med den første lønsingen er at du har full kontroll over operasjonene og framgangsmåten.

Vi kan også tegne løsningen. Husk å sette inn en diskret variabel x i det symbolske uttrykket for løsningen.

```
x = linspace(0,10,100);
plot(x, subs(y,t,x))
grid on;
```

Grafen er vist i Figur 12.



FIGUR 12. $y(t) = e^{2t} - 4e^{t/2}$, løsningen til $2y'' - 5y' + 2y = 0$.

□

Vi fortsetter med inhomogene likninger.

Eksempel 8.2. Bestem løsningen til differesiallikningen

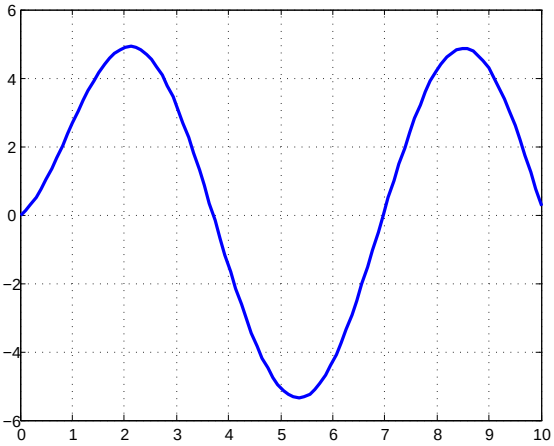
$$4y'' + 4y' + y = 25\cos t, \quad y(0) = 0,\ y'(0) = 1.$$

Løsning.

```
% Input: likningen 4y"+4y'+y=25*cos(t)
%                    y(0)=0, y'(0)=1
% Output: løsningen til likningen

% Definerer en symbolsk funksjon
syms y(t)
% Første derivert
Dy = diff(y);
% Andre derivert
D2y = diff(y,2);
% Løser likningen
y = dsolve(4*D2y +4*Dy+y == 25*cos(t), y(0) == 0, Dy(0) == 1)
% Tegner løsningen
x = linspace(0,10,100);
plot(x,subs(y,t,x), 'LineWidth',2);
grid on;
```

Output:

```
ans =
3*exp(-t/2) - 3*cos(t) + 4*sin(t)
- (3*t*exp(-t/2))/2 + 5*t*(cos(t)/2 + sin(t))
- (5*t*cos(t))/2 - 5*t*sin(t)
```



FIGUR 13. Løsningen til $4y'' + 4y' + y = 25\cos t$, $y(0) = 0, y'(0) = 1$.

□

Oppgaver

1. Bestem løsningen til differesiallikningen

$$y'' + 4y' + 4y = 0, \quad y(0) = -2,\ y'(0) = 7.$$

2. Bestem løsningen til differesiallikningen

$$y'' - 2y' + 2y = 8e^{4t} - 2\cos(3t) + 5\sin(3t)$$
$$y(0) = 44/85,\ y'(0) = 0.$$

9. Matriser. Lineære likningssett.

MATLAB er en riktig proffs når det gjelder matriseregning. Det ser man også i navnet: MATLAB er en forkortelse for "matrix laboratory".

For å løse et lineært likningssytem, skriver vi det om på formen $Ax = b$, der A er en matrise og b er en vektor. Siden kan vi bruke innebygde MATLAB funksjoner til å løse likningssettet. Les mer om hvordan MATLAB håndterer lineære likningssett i Documentation.

Eksempel 9.1. Løs et likningssystem

$$\begin{cases} 3x_1 + 2x_2 - 2x_3 = 1 \\ x_2 + 5x_3 + x_4 = 9 \\ -2x_1 - 3x_2 - 3x_3 = 2. \end{cases}$$

Løsning. Vi skriver opp koeffisientmatrisen

```
>> A = [3 2 -2; 1 5 1; -2 3 -3]
A =  3     2     -2
     1     5      1
    -2     3     -3
```

og vektoren med elementer fra høyresiden i likningssystemet

```
>> b = [1; 9; 2]
b =  1
     9
     2
```

Løsningen til $Ax = b$ finner vi ved å taste inn $x = A\backslash b$. Merk at løsningen til $xA = b$ gis ved $x = b/A$.

```
>> x = A\b
x =-2.5000
    2.6250
   -1.6250
```

□

Oppgaver

1. Løs likningssystemet

$$\begin{cases} x_1 - x_2 + 3x_3 = 3 \\ 4x_1 + 3x_2 + 4x_3 = 2 \\ 2x_2 - 3x_3 + 2x_4 = 0 \\ x_3 - x_4 = -1. \end{cases}$$

2. Løs likningssystemet for hånd og ved hjelp av MATLAB. Har du fått samme svar? Forklar.

$$\begin{cases} x_1 + x_2 + x_3 = 1 \\ x_1 + x_2 + 2x_3 = 3. \end{cases}$$

10. Matriser. Egenvektorer og egenverdier.

Eksempel 10.1. Finn egeverdier og egenvektorer til

$$A = \begin{pmatrix} 2 & -1 & 3 \\ 1 & -1 & -2 \\ 0 & 0 & 1 \end{pmatrix}$$

Løsning.

Numerisk:

```
>> A=[2 -1 3;1 -1 -2; 0 0 1]
A =
     2    -1     3
     1    -1    -2
     0     0     1

>> [vektor, verdi]=eig(A)
vektor =
    0.9342    0.3568   -0.8433
    0.3568    0.9342   -0.5270
         0         0    0.1054

verdi =
    1.6180         0         0
         0   -0.6180         0
         0         0    1.0000
```

Matrisen har tre egeverdier $1.6180, -0.6180, 1$ som står i diagonalen i `verdi`, og tre tilsvarende egenvektorer, kolonner i `vektor`.

Symbolsk:

```
>> A = sym([2 -1 3; 1 -1 -2; 0 0 1])
A =
[ 2, -1,  3]
[ 1, -1, -2]
[ 0,  0,  1]

>> [vektor, verdi]=eig(A)
vektor =
[ -8, 3/2 - 5^(1/2)/2, 5^(1/2)/2 + 3/2]
[ -5,               1,               1]
[  1,               0,               0]

verdi =
[ 1,               0,               0]
[ 0, 1/2 - 5^(1/2)/2,               0]
[ 0,               0, 5^(1/2)/2 + 1/2]
```

□

Eksempel 10.2. Regn ut egenverdier og egenvektorer til

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Løsning.

```
>> A=[1 0 0;0 0 0; 0 0 0]
A =
     1     0     0
     0     0     0
     0     0     0

>> [vektor, verdi]=eig(A)
vektor =
     0     0     1
     1     0     0
     0     1     0

verdi =
     0     0     0
     0     0     0
     0     0     1
```

Den første egenverdien $\lambda_1 = 0$ har multiplisitet 2, det betyr at det finnes to egenvektorer som tilsvarer denne egenverdien: $[0, 1, 0]$ og $[0, 0, 1]$. Disse vektorene er orthogonale og følgelig lineært uavhengige. Den siste egevektoren $[1, 0, 0]$ tilhør $\lambda_2 = 1$. □

Eksempel 10.3. Avgjør om matrisen

$$A = \begin{pmatrix} 2 & 2 & -6 \\ 2 & -1 & -3 \\ -2 & -1 & 1 \end{pmatrix}$$

er diagonaliserbar. Dersom den er det, finn den diagonale matrisen $D = P^{-1}AP$, der P er matrisen av egenvektorer.

Løsning.

En kvadratisk $n \times n$ matrise A kan diagonaliseres dersom den har n lineært uavhengige egenvektorer $v_1, v_2, \cdots, v_n$ med tilhørende egenverdier $\lambda_1, \lambda_2, \cdots, \lambda_n$. Danner vi matrisen av egenvektorene $P = [v_1, v_2, \cdots, v_n]$, blir $D = P^{-1}AP$ en diagonalmatrise:

$$D = P^{-1}AP = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_n. \end{pmatrix}$$

Vi begynner ved å finne egenverdier og egenvektorer til A.

```
>> A = [2 2 -6; 2 -1 -3; -2 -1 1]
A =
     2     2    -6
     2    -1    -3
    -2    -1     1

>> [vec, val] = eig(A)
vec =
    0.8165    0.8165    0.5575
    0.4082   -0.4082    0.6015
   -0.4082    0.4082    0.5722
val =
     6     0     0
     0    -2     0
     0     0    -2
```

Matrisen A har én egenverdi av multiplisitet 1, $\lambda_1 = 6$, og én egenverdi av multiplisitet 2, $\lambda_2 = -2$. Matrisen `vec` består av egenvektorer. De tre vektorene er lineært uavhengige. La os sjekke dette.

La oss ta en lineær kombinasjon av egenvektorene og sette den lik null:

$$\alpha \begin{bmatrix} 0.8165 \\ 0.4082 \\ -0.4082 \end{bmatrix} + \beta \begin{bmatrix} 0.8165 \\ -0.4082 \\ 0.4082 \end{bmatrix} + \gamma \begin{bmatrix} 0.5575 \\ 0.6015 \\ 0.5722 \end{bmatrix} = 0.$$

Om det viser seg at alle konstantene α, β, γ er null, er egenvektorene lineært uavhengige. Vi skriver om denne likningen som et likningssett:

$$\begin{pmatrix} 0.8165 & 0.8165 & 0.5575 \\ 0.4082 & -0.4082 & 0.6015 \\ -0.4082 & 0.4082 & 0.5722 \end{pmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = 0.$$

Den eneste løsningen til dette likningssytemet er $[\alpha, \beta, \gamma] = 0$, så egenvektorene er lineært uavhengige. Legg merke til at kolonnenene av en kvadratisk matrise er lineært uavhengige dersom det $A \neq 0$ (for at da har den homogene likningen $Ax = 0$ bare en triviel løsning).

Følgelig er matrisen A diagonaliserbar og vi kan regne ut $D = \text{vec}^{-1} A \text{vec}$:

```
>> vec^(-1)*A*vec
ans =
    6.0000   -0.0000    0.0000
   -0.0000   -2.0000   -0.0000
   -0.0000   -0.0000   -2.0000
```

Resultatet er en diagonalmatrise med egenverdiene i diagonalen.

Er du i tvil at beregningen er nøyaktig nok, kan du gjøre det samme i MATLAB **symbolic**:

```
>> A = sym([2 2 -6; 2 -1 -3; -2 -1 1])
A =
```

```
[  2,  2, -6]
[  2, -1, -3]
[ -2, -1,  1]

>> [vektor, verdi]=eig(A)
vektor =
[ -2, -1/2, 3/2]
[ -1,    1,   0]
[  1,    0,   1]

verdi =
[ 6,  0,  0]
[ 0, -2,  0]
[ 0,  0, -2]
```

Determinanten til matrisen `vektor` er ikke null

```
>> det(vektor)
ans = -4
```

Det betyr at egenvektorene er lineært uavhengige og matrisen A er diagonaliserbar:

```
>> vektor^(-1)*A*vektor
ans =
[ 6,  0,  0]
[ 0, -2,  0]
[ 0,  0, -2]
```

□

Oppgaver

1. Bruk MATLAB til å regne ut egenverdier og egenvektorer i opp. 11.3.6 i tekstboka.
2. Regn ut egenverdier og egenvektorer til
$$A = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 2 & 2 \\ 0 & 0 & 2 \end{pmatrix}$$
3. Avgjør om matrisen
$$A = \begin{pmatrix} -2 & -3 & -3 \\ 4 & 3 & 1 \\ -2 & 0 & 2 \end{pmatrix}$$
er diagonaliserbar. Dersom den er det, finn den diagonale matrisen $D = P^{-1}AP$, der P er matrisen av egenvektorer.

11. Fasit

Kapitel 1

1. Lag en MATLAB-kode som regner ut prikkproduktet mellom to vektorer og bruk den for å kontrollere svaret i opp. 4.3.1.

Løsning.

```
% Input: två vektorer
% Output: Prikkprodukt
% Definer de to vektorene
a = [1 1 2];
b = [-1 2 3]
% Regn ut prikkproduktet
PRODUCT = dot(a,b);
% Skriv ut svaret
fprintf('Skalarproduktet = %f\n', PRODUCT);
```

2. Lag et program som tar i mot en vektor fra brukeren og gir en enhetsvektor i samme retning.

Løsning.

Gitt en vektor $\vec{a}$, kan vi dele den med lengden $|\vec{a}|$ og få en enhetsvektor i samme retning som $\vec{a}$: $\vec{u} = \vec{a}/|\vec{a}|$. Husker du at divisjon ikke er definert for vektorer? Det vil si at man kan dele en vektor med et skalar (et tall), men ikke med en vektor.

For å gjøre det enklere å endre en vektor, skal vi spørre brukeren om å trykke inn en. Vi skal bruke `input` funksjon (bruk gjerne `help` for å finne ut mer om `input`). Symbolet `\n` starter en ny linje. Her er MATLAB koden som regner ut $\vec{u}$, gitt $\vec{a}$.

```
% Input: en vektor fra brukeren
% Output: en enhetsvektor i samme retning
% Spør etter en vektor
a = input('Skriv en vektor på formen [a1 a2 a3] \n');
% Deler a vektor med sin absoluttverdi
u = a./norm(a);
% Skriver ut svaret
disp('Enhetsvektoren i retningen av a er'); disp(u)
```

Legg merke til at for $\vec{a} = [a_1, a_2, a_3]$, gir `norm(a)` absoluttverdien

$$|\vec{a}| = \sqrt{|a_1|^2 + |a_2|^2 + |a_3|^2}.$$

3. Lag et program som tar i mot to vektorer fra brukeren og finner en enhetsvektor som står vinkelrett på begge to.

Løsning.

Gitt to vektorer $\vec{a}$ og $\vec{b}$, for å finne en vektor som står vinkelrett på begge to, skal vi ta kryssproduktet $\vec{a} \times \vec{b}$. Det er enklest å bruke en innebygd funksjon `cross`.

```
% Input: To vektorer a og b fra brukeren
% Output: En vektor orthogonal til a og b
% Spør brukeren etter den første vektoren
a = input('Trykk inn en vektor på formen [a1 a2 a3] \n');
% Spør brukeren etter en til vektor
b = input('Trykk inn en til vektor på formen [b1 b2 b3] \n');
% Regn ut vektorproduktet
c = cross(a,b);
disp('En vektor som står vinkelrett på a og b er'); disp(c)
```

4. Lag et program som tar imot tre vektorer fra brukeren og regner ut volumet av parallellepipedet dannet av de tre vektorene.

Løsning.

Volumet av parallellepipedet dannet av $\vec{a}, \vec{b}$ og $\vec{c}$ er gitt ved skalar trevektorproduktet $\vec{a} \cdot (\vec{b} \times \vec{c})$. Her er MATLAB koden.

```
% Input: Tre vektorer
% Output: Volumet av parallellepipedet (trevektorprodukt)
% Spør etter tre vektorer:
a = input('Første vektor \n');
b = input('Andre vektor \n');
c = input('Tredje vektor \n');
% Regn ut trevektorprodukt
volum = dot(a, cross(b, c));
% Skriv ut svaret
fprintf('Volumet av parallellepipedet = %d\n', volum);
```

Kapitel 2

1. Trykk inn de tallene

$$1 + \sqrt{-4} \quad \text{og} \quad \frac{5}{8} + \sqrt{-\frac{7}{64}}$$

i MATLAB og kontroller svaret.

Løsning.

```
>> 1 + sqrt(-4)

ans =

   1.0000 + 2.0000i
```

```
>> 5/8+sqrt(-7/64)

ans =
```

```
    0.6250 + 0.3307i
```

2. Regn ut $(1+i)^2 \cdot (-3+2i)$.
Løsning.

```
>> (1+1i)^2*(-3+2i)

ans =

  -4.0000 - 6.0000i
```

3. Finn den reelle og den imaginære delen av $(1+i)^2 \cdot (-3+2i)$.
Løsning.

```
>> real((1+1i)^2*(-3+2i))

ans =

    -4

>> imag((1+1i)^2*(-3+2i))

ans =

    -6
```

4. Løs likningene:

(a) $z^4 = 16$
Løsning.

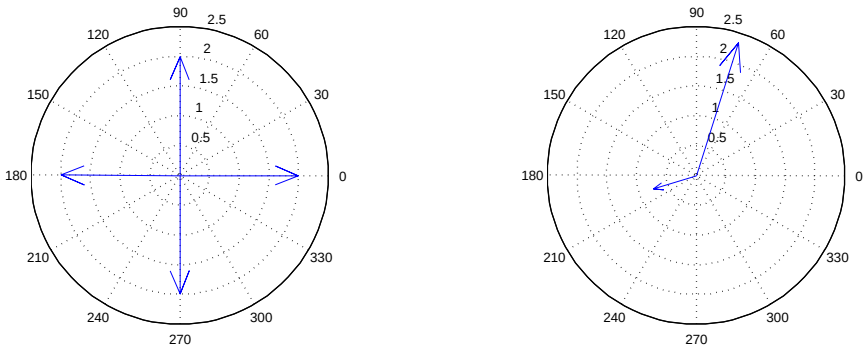
```
1 % Løser z^4 = 16
2 % Input: Koeffisienter til et polynom
3 % Output: Nullpunkter til polynomet
4 % Definerer en vektor av koeffisientene
5 C = [1 0 0 0 -16];
6 % Regner ut nullpunktene
7 roots(C)
8 % Tegner nullpunktene
9 compass(roots(C))
```

Figur 14(a) viser fire løsninger i det komplekse planet.

Output:

```
ans =

  -2.0000 + 0.0000i
   0.0000 + 2.0000i
   0.0000 - 2.0000i
   2.0000 + 0.0000i
```

(b) $z^2 - 2iz - \sqrt{3}i = 0$.



(a) $z^4 = 16$ (b) $z^2 - 2iz - \sqrt{3}i = 0$

FIGUR 14. Løsningene til oppgave 4 .

Løsning.

```
% Løser z^2-2iz - sqrt(3)i = 0
% Input: Koeffisienter til et polynom
% Output: Nullpunkter til polynomet
% Definerer en vektor av koeffisientene
C = [1 -2i -sqrt(3)*1i];
% Regner ut nullpunktene
roots(C)
% Tegner nullpunktene
compass(roots(C))
```

Figure 14(b) viser de to løsningene i det komplekse planet.

Output:

```
ans =

   0.7071 + 2.2247i
  -0.7071 - 0.2247i
```

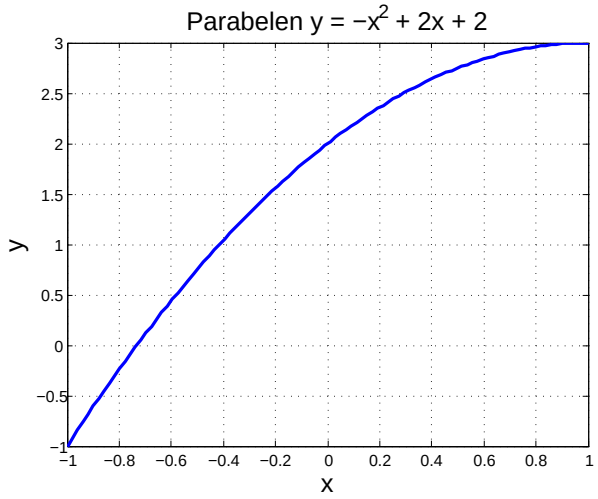
Kapitel 3

1. Tegn $f(x) = -x^2 + 2x + 2$ og vis at $f(x)$ har minst ett nullpunkt i $[-1, 1]$.

Løsning.

```
% Tegner f(x) = -x^2 + 2x +2 for -1 < x < 1.
close all;
% x-vektor
x = linspace(-1, 1);
% Definerer y
```

```
y = -x.^2 + 2.*x +2;
% Tegner
plot(x, y, 'LineWidth', 2)
grid on
xlabel('x', 'FontSize', 16)
ylabel('y', 'FontSize', 16)
title('Parabelen y = -x^2 + 2x + 2', 'Fontsize', 16)
% Sjekker om f(x) har et nullpunkt
% Koeffisientene til -x^2 + 2x + 2
C = [-1, 2, 2];
% Nullpunktene til polynomet med koeffisientene C
disp('Nullpunkter til f(x)=-x^2 + 2x + 2'); disp(roots(C))
```



FIGUR 15. Parabelen $f(x) = -x^2 + 2x + 2$.

Output:

```
Nullpunkter til f(x)=-x^2 + 2x + 2
    2.7321
   -0.7321
```

Vi ser at funksjonen har ett nullpunkt i $[-1, 1]$: $x_0 = -0.7321$.

OBS! Funksjonen `roots` anvender vi for polynomer. For andre funksjoner bruk "fzero". □

2. Tegn linjen mellom to punkter $A(2, -1, 0)$ og $B(4, 4, -3)$.

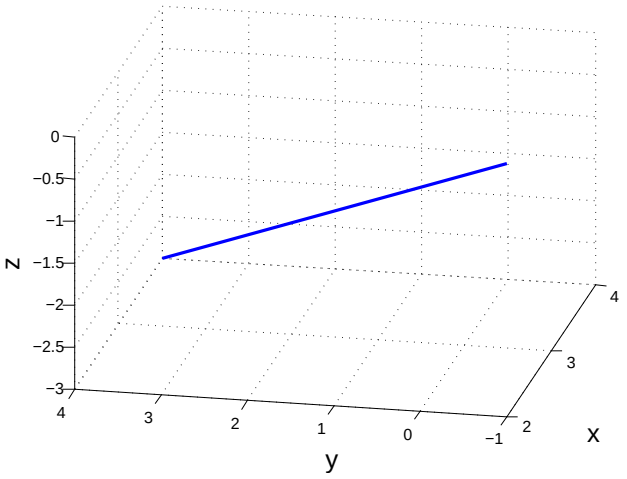
Løsning.

`plot(x,y)` for $x = [x1, x2, ..., x_n]$ og $y = [y_1, y_2, \cdots, y_n]$ tegner stien som består av rette linjestyllene som forbinder punktene $(x_1, y_1), (x_2, y_2), \cdots (x_n, y_n)$.

I 3D bruker har vi `plot3(x, y, z)` med x, y, z vektorer med lengde lik antall punkter.

```
% Tegner linjen mellom (2,-1, 0) og (4, 4, -3)
close all;
```

```
% x koordinatene
x = [2, 4];
% y koordinatene
y = [-1, 4];
% z koordinatene
z = [0,-3];
% Tegner linja
plot3(x,y,z, 'LineWidth', 2)
grid on
xlabel('x', 'FontSize', 16);
ylabel('y', 'FontSize', 16);
zlabel('z', 'FontSize', 16);
```



FIGUR 16. Linjen mellom $A(2,-1,0)$ og $B(4,4,-3)$.

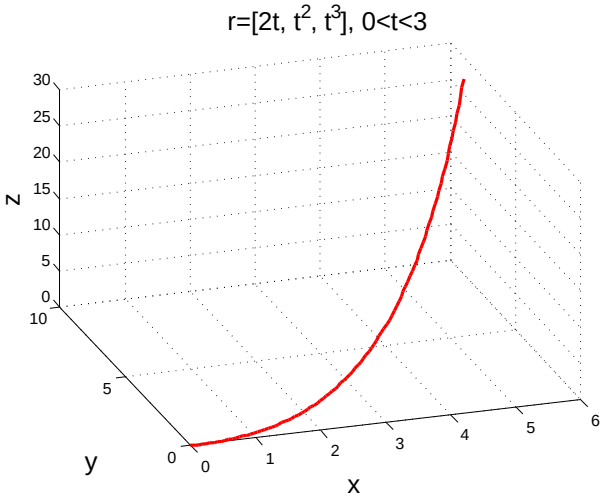
Vil du markere visse punkter på en kurve, les mer om markers. □

3. Tegn romkurven

$$x(t) = 2t, \quad y(t) = t^2, \quad z(t) = t^3, \quad 0 \leq t \leq 3.$$

Løsning.

```
% Tegner en romkurve r(t)=[2t, t^2, t^3], 0<t<3
t = linspace(0, 3);
% Definerer x(t), y(t), z(t)
xt = 2*t;
yt = t.^2;
zt = t.^3;
% Tegner grafen
plot3(xt,yt,zt, 'r', 'Linewidth', 2)
grid on
xlabel('x', 'FontSize', 16)
ylabel('y', 'FontSize', 16)
zlabel('z', 'FontSize', 16)
title('r=[2t, t^2, t^3], 0<t<3', 'FontSize', 16)
```



FIGUR 17. Kurven $r(t) = [2t, t^2, t^3], \quad 0 \leq t \leq 3$.

□

4. Tegn en kurve i polar koordinater $r = 8/(5 - 3\cos\theta)$, $\theta \in [0, 2\pi]$.

Løsning.

```
% Tegner polarkurven r = 8/(5-3cos(theta)), 0<theta<2pi.
close all;
% theta - vektor med lengde 100
theta = linspace(0, 2*pi);
% Definerer polar radius r=r(t)
r = 8./(5-3*cos(theta));
% Byter til kartesiske koordinater
x = r.*cos(theta);
y = r.*sin(theta);
% Tegner kurven
plot(x,y, 'LineWidth', 2)
grid on
```

□

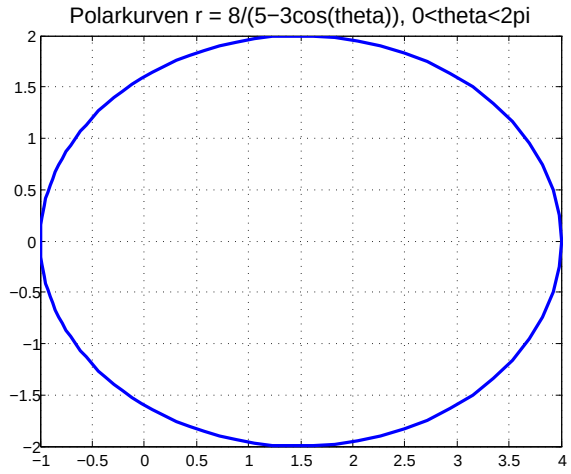
5. Tegn $f(x) = \dfrac{\sin x}{x}$ i MATLAB, regn ut (symbolsk) grenseverdien

$$\lim_{x \to 0} \frac{\sin x}{x}$$

og sjekk om svaret stemmer med grafen.

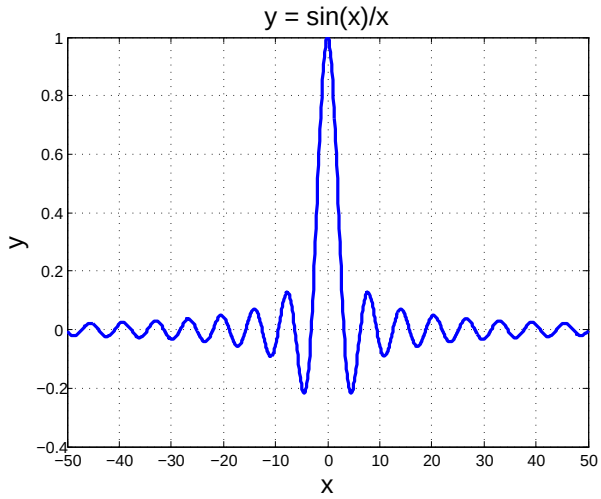
Løsning.

```
% Tegner
close all;
% x-vektor
x = linspace(-50, 50, 1000);
```



FIGUR 18. Polarkurven $r = 8/(5 - 3\cos\theta)$, $\theta \in [0, 2\pi]$.

```
5  % Definerer y
6  y = sin(x)./x;
7  % Tegner
8  plot(x, y, 'LineWidth', 2)
9  grid on
10 xlabel('x', 'FontSize', 16)
11 ylabel('y', 'FontSize', 16)
12 title('y = sin(x)/x', 'Fontsize', 16)
13 % Regner ut symbolsk grenseverdi når x -> 0
14 syms t
15 LIMIT = limit(sin(t)/t, t, 0);
16 disp('Grenseverdi lim_{x -> 0} sin(x)/x =');
17 disp(LIMIT)
```



FIGUR 19. $f(x) = \dfrac{\sin x}{x}$, $x \in [-50, 50]$.

Output:

```
Grenseverdi lim_{x -> 0} sin(x)/x =
1
```

□

Kapitel 4

1. Bruk MATLAB symbolic for å finne første og andrederiverte av $f(x) = \cos\sqrt{2+x^4}$.

Løsning.
```
>> syms x
>> diff(cos(sqrt(2+x^4)))

ans =
    -(2*x^3*sin((x^4 + 2)^(1/2)))/(x^4 + 2)^(1/2)

>> diff(-(2*x^3*sin((x^4 + 2)^(1/2)))/(x^4 + 2)^(1/2))

ans =

(4*x^6*sin((x^4 + 2)^(1/2)))/(x^4 + 2)^(3/2)
- (6*x^2*sin((x^4 + 2)^(1/2)))/(x^4 + 2)^(1/2)
- (4*x^6*cos((x^4 + 2)^(1/2)))/(x^4 + 2)
```

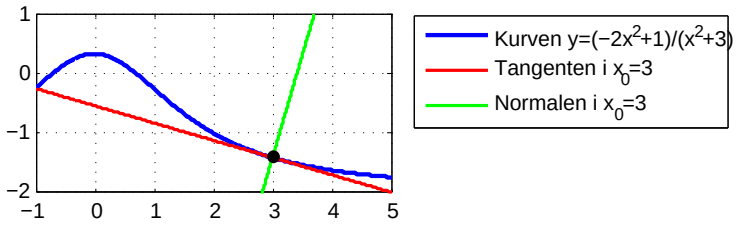
□

2. Gitt $f(x) = \dfrac{-2x^2+1}{x^2+3}$, sett opp likninger for tangenten og normalen til grafen i punktet $x_0 = 3$. Tegn grafene til funskjonen, tangenten og normalen i ett og samme koordinatsystem.

Løsning.

```
% Input: en funksjon, et punkt
% Output: plot av funskjonen, tangenten og normalen
close all
h = 0.001;          % skrittlengde
x = -1: h:5;        % variabel
x0=3;               % et gitt punkt
f = (-2*x.^2+1)./(x.^2+3);       % en gitt funskjon
plot(x, f, 'LineWidth', 2); % tegner f
hold on; grid on;axis equal;
xlim([-1,5]); ylim([-2, 1]);
% Finner punktet i x nærmest x0=3
[val, idx] = min(abs(x-x0))
y0 = f(idx) % funksjonens verdi i det punktet
% Approksimasjon av den deriverte y'(x0)
df = (f(idx+1)-f(idx))/h;
% Tangenten y=y0 + y'(x0)*(x-x0)
T = y0 + df*(x-x(idx));
```

```
% Normalen y = y0 -1/y'(x0)*(x-x0)
N = y0 - (x-x(idx))/df;
plot(x, T, 'r', 'LineWidth', 1.5) % Tegner tangenten i rødt
plot(x, N, 'g', 'LineWidth', 1.5) % Tegner normalen i grønt
plot(x0,y0,'.k', 'MarkerSize', 20) % Markerer (x0, y0)
legend('Kurven y=(-2x^2+1)/(x^2+3)', 'Tangenten i x_0=3', ...
        'Normalen i x_0=3',...
        'location', 'NorthEastOutside');
```



FIGUR 20. $f(x) = \dfrac{-2x^2+1}{x^2+3}$, tangenten og normalen i punktet $x_0 = 3$.

Output:

```
val =
     0
idx =
        4001
y0 =
   -1.4167
```

□

3. Gitt funksjonen $f(x) = \ln x$, regn ut de første fem deriverte for hånd eller ved hjelp av MATLAB symbolic. Tegn grafene til funksjonen og taylorpolynomene av orden 1 til 5 om punktet $x = 1$.

Løsning.

For å gjøre det enklere, deriverer vi funksjonen fem ganger symbolsk:

```
>> syms x
>> diff(log(x))
ans = 1/x

>> diff(log(x),2)
ans = -1/x^2

>> diff(log(x),3)
ans = 2/x^3
```

```
>> diff(log(x),4)
ans = -6/x^4

>> diff(log(x),5)
ans = 24/x^5
```

Nå kan vi bruke de deriverte til å konstruere taylorpolynomene.

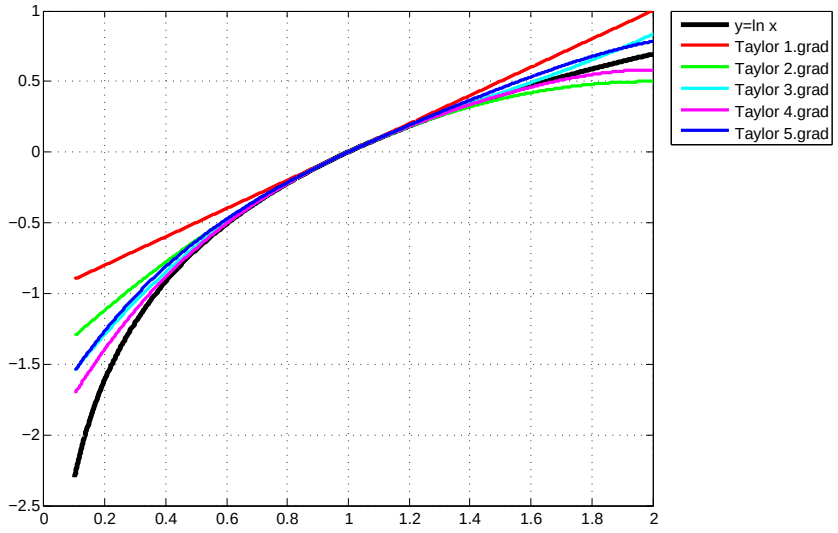
```
% Input: f(x) og de første fem deriverte
% Output: grafen til f(x) og taylorpolynomene
%         av grad fra 1 til 5
close all;
% Definerer x
x = 0.1: 0.001: 2;
% Definerer f=ln(x)
f = log(x);
% Definerer deriverte til f
d1f = 1./x;
d2f = -1./x.^2;
d3f = 2./x.^3;
d4f = -6./x.^4;
d5f = 24./x.^5;
% Tegner f
plot(x, f, 'Color', 'k', 'LineWidth', 3);
hold on; grid on;
% Finner et punkt x_idx i x vektoren nære x0=1
[val,idx] = min(abs(x - 1));
% Definerer og tegner taylorpolynomet av 1.grad
P1 = f(idx) + d1f(idx)*(x - x(idx));
plot(x, P1, 'Color', 'r', 'LineWidth', 2);
% Definerer og tegner taylorpolynomet av 2.grad
P2 = P1 + d2f(idx)*(x - x(idx)).^2/factorial(2);
plot(x, P2, 'Color', 'g', 'LineWidth', 2);
% Definerer og tegner taylorpolynomet av 3.grad
P3 = P2 + d3f(idx)*(x - x(idx)).^3/factorial(3);
plot(x, P3, 'Color', 'c', 'LineWidth', 2);
% Definerer og tegner taylorpolynomet av 4.grad
P4 = P3 + d4f(idx)*(x - x(idx)).^4/factorial(4);
plot(x, P4, 'Color', 'm', 'LineWidth', 2);
% Definerer og tegner taylorpolynomet av 5.grad
P5 = P4 + d5f(idx)*(x - x(idx)).^2/factorial(5);
plot(x, P5, 'Color', 'b', 'LineWidth', 2);
legend('y=ln x','Taylor 1.grad','Taylor 2.grad',...
'Taylor 3.grad','Taylor 4.grad', 'Taylor 5.grad',...
'location', 'NorthEastOutside');
```

Figur 21 viser grafene til funksjonen og taylorpolynomene av grad fra 1 til 5.

□

4. La $f(x) = \cos(x/2)$. Sett opp likningene til polynomene P av grad mindre eller lik n med

$$P^{(k)}(0) = f^{(k)}(0), \quad k = 0, 1, \cdots n$$



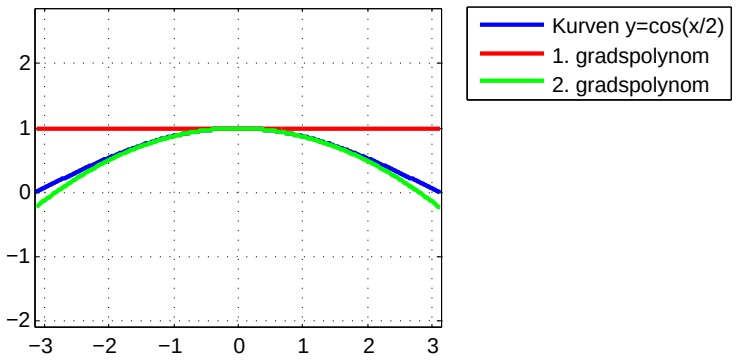
FIGUR 21. $f(x) = \ln x$ og de første fem taylorpolynomene i $x_0 = 1$.

for $n = 2$. Tegn grafene til funskjonen og de tre polynomene i ett og samme koordinatsystem.

Løsning.

```
% Approksimerer f=cos(x/2) med polynomer i x=0
close all;
h = 0.0001; % skrittlengde
x = -pi:h:pi;
x0 = 0;
f = cos(x./2);
plot(x, f, 'LineWidth', 2); % tegner f
xlim([-pi, pi]); ylim([-1, 1]);
hold on; grid on;axis equal;
% Finner punktet i x nærmest x0=0
[val, idx] = min(abs(x-x0))
y0 = f(idx) % funksjonens verdi i det punktet
% Approksimasjon av den deriverte f'(0)
df = (f(idx+1)-f(idx))/h;
% Approksimasjon av den andre deriverte f"(0)
ddf = (f(idx+1) - 2*f(idx)+f(idx-1))/h^2;
% Polynom p=C1x+C0, p(0)=f(0) og p'(0)=f'(0)
C0 = f(idx); % C0 = tilnærmet f(0)
C1= df; % C1 = tilnærmet f'(0)
% Polynom av første grad
P1 = C1*x + C0;
% Skriver ut P1
fprintf('P1 = %fx + %f', C1, C0);
plot(x, P1, 'Color', 'r', 'LineWidth', 2); % Tegner P1
% Polynom p=C2* x^2 + C1*x+C0: p(0)=f(0), p'(0)=f'(0)
% og p"(0) = f"(0)
```

```
C0 = f(idx); % C0 = tilnærmet f(0)
C1 = df; % C1 = tilnærmet f'(0)
C2 = ddf/2; % C2 = tilnærmet f"(0)/2
% Polynom av andre grad
P2 = C2*x.^2 + C1*x + C0;
% Skriver ut P2
fprintf('\nP2 = %f x^2 + %fx + %f', C2, C1, C0);
plot(x, P2, 'Color', 'g', 'LineWidth', 2); % Tegner P2
legend('Kurven y=cos(x/2)', '1. gradspolynom',...
        '2. gradspolynom','location', 'NorthEastOutside');
```



FIGUR 22. $f(x) = \cos(x/2)$ og polynomer av grad 1 og 2 som approximerer $f(x)$ i $x = 0$.

```
val =
   7.3464e-06
idx =
       31417
y0 =
    1.0000
P1 = -0.000014x + 1.000000
P2 = -0.125000 x^2 + -0.000014x + 1.000000
```

□

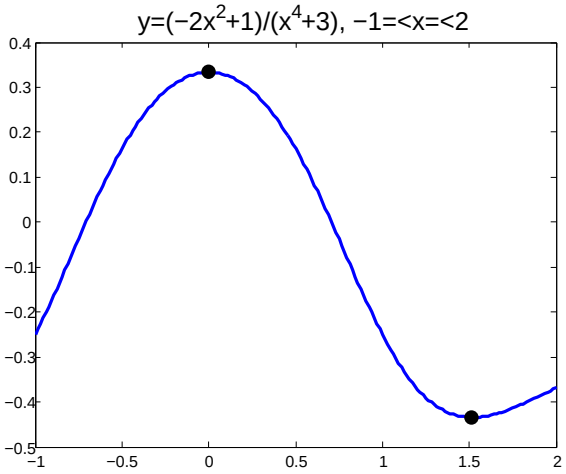
Kapitel 5

1. Bruk koden fra Eksempel 5.1 til å finne maksimal- og minimalverdi av $f(x) = \dfrac{-2x^2+1}{x^4+3}$, $x \in [-1, 2]$.

Løsning.

```
% Input: en funksjon f(x)
% Output: grafen og ekstremalpunkter
close all;
```

```
4  % x varierer fra -1 til 2
5  x = -1:2^(-6):2;
6  % Definerer f= (-2x^2+1)/(x^4+3)
7  f = (-2*x.^2+1)./(x.^4+3);
8  % maxval= maksimalverdi, imax=posisjon i x
9  [maxval, imax] = max(f)
10 % minval=minimalverdi, imin=posisjon i x
11 [minval, imin] = min(f)
12 % Tegner y=f(x)
13 plot(x, f, 'LineWidth', 2);
14 hold on
15 % Markerer ekstremalpunktene
16 plot(x(imax), maxval, '.k', 'MarkerSize', 30);
17 plot(x(imin), minval, '.k', 'MarkerSize', 30);
18 title('y=(-2x^2+1)/(x^4+3), -1=<x=<2', 'FontSize', 16);
19 % Skriver ut max og min
20 fprintf('max=(%f, %f), min=(%f, %f)', ...
21         x(imax), maxval, x(imin), minval);
```



FIGUR 23. Minimal- og maksimalverdi av $f(x) = \frac{-2x^2+1}{x^4+3}$, $x \in [-1, 2]$.

```
maxval =
    0.3333
imax =
    65
minval =
   -0.4343
imin =
   162
max=(0.000000, 0.333333), min=(1.515625, -0.434257)
```

□

Kapitel 6

1. Finn en tilnærmet verdi for π med 8 sikre desimaler ved å bruke Newtons metode til å løse likningen $\cos x = 1/2$.

Løsning.
Innen vi kaster oss i det og skriver koden, la oss planlegge litt.

Likningen $\cos x = 1/2$ for $x \in [0, \pi]$ har en løsning: $x = \pi/3$. Så om vi klarer av å finne en tilnærmet verdi x_n slik at $\cos x - 0.5 = 0$, får vi $\pi \approx 3x_n$. La oss velge startpunktet $x_0 = 1$ som er nære $\pi/3$.

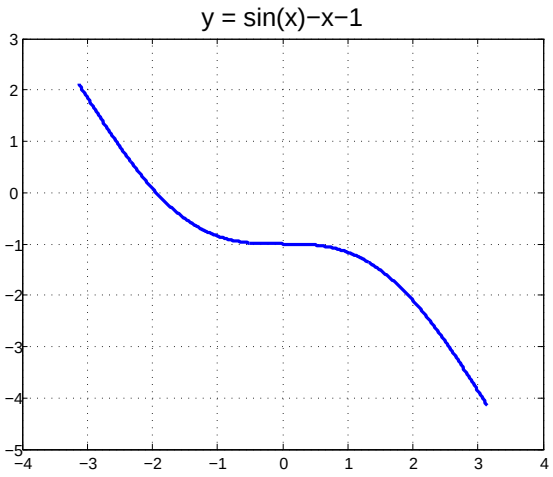
```
% Input: cos x = 0.5
% Output: pi med 8 sikre desimaler
close all;
h = 0.0001; % skrittlengde
% Lager x vektor
x = 0:h:pi;
% Definerer F
F = cos(x)-0.5;
% Velger statpunkter for Newtons metode
x0 = 0; x1 = 1;
% Iterasjonssteg
k = 0;
while (abs(x0-x1)>10^(-8))
    x0 = x1;
    k = k+1;
    % Finner et punkt i x vektor nærmest x0
    [val0, idx0] = min(abs(x-x0));
    % Approksimerer F'(x0)
    dF0 = (F(idx0+1)-F(idx0))/h;
    % Første steg i Newtons metode
    x1 = x(idx0) - F(idx0)/dF0;
    % Approksimasjon av pi
    ppi = 3*x1;
    % Skriver ut x1
    % %.8f gir 8 desimaler
    fprintf('x%d=%.8f \n ', k, ppi);
end
```

```
x1=3.14368058
x2=3.14159314
x3=3.14159265
x4=3.14159265
```

□

2. Vis at funksjonen $F(x) = \sin x - x - 1$ har nøyaktig ett nullpunkt, og bruk Newtons metode til å finne en tilnærmet verdi for det punktet med tre sikre desimaler.

Løsning.
Vi begynner ved å tegne $y = F(x)$ (se Figur 24).



FIGUR 24. $y = \sin x - x - 1$, $x \in [-\pi, \pi]$.

Den deriverte $F'(x) = \cos x - 1$ er negativ for alle x, så funksjonen F er fallende. Fordi $F(-\pi) = \pi - 1 > 0$ og $F(\pi) = -\pi - 1 < 0$, funksjonen F har ett nullpunkt for alle x.

```
% Input: F = sin(x) - x -1
% Output: nullpunkter til F
close all;
h = 0.0001; % skrittlengde
% Lager x vektor
x = -pi:h:pi;
% Definerer F
F = sin(x) -x-1;
% Tegner y=F(x)
plot(x, F, 'LineWidth', 2);
grid on;
title('y = sin(x)-x-1', 'FontSize', 16);
% Velger statpunkter for Newtons metode
x0 = 0; x1 = 1;
% Iterasjonssteg
k = 0;
while (abs(x0-x1)>10^(-3))
    x0 = x1;
    k = k+1;
    % Finner et punkt i x vektor nærmest x0
    [val0, idx0] = min(abs(x-x0));
    % Approksimerer F'(x0)
    dF0 = (F(idx0+1)-F(idx0))/h;
    % Første steg i Newtons metode
    x1 = x(idx0) - F(idx0)/dF0;
    % Approksimasjon av pi
    ppi = 3*x1;
    % Skriver ut x1
    % %.8f gir 8 desimaler
    fprintf('x%d=%.8f \n ', k, ppi);
```

```
31 end
```

```
x1=-4.55979915
x2=-6.07320022
x3=-5.81135176
x4=-5.80369597
x5=-5.80368963
```

□

Kapitel 7

1. Bruk MATLAB til å delbrøkoppspalte

$$\frac{100(x^3+2x+8)}{x^5+7x^4+18x^3+14x^2-15x-25}.$$

Løsning.

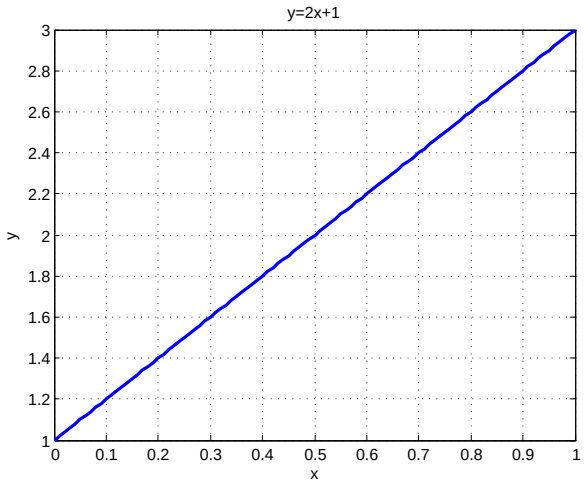
```
>> syms x
>> f = (100*(x^3+2*x+8))/(x^5+7*x^4+18*x^3+14*x^2-15*x-25);
>> diff(int(f))

ans =
11/(x - 1) + (- 11/2 - 51*i)/(x + 2 - i)
+ (- 11/2 + 51*i)/(x + 2 + i) + 35/(x^2 + 4*x + 5)
- ((2*x + 4)*(35*x + 275))/(x^2 + 4*x + 5)^2
```

□

2. Beregn de bestemte integralene $\int_0^1(2x+1)dx$ og $\int_0^1 \cos x dx$ ved hjelp av trapesmetoden med $n = 50$. Regn ut disse integralene eksakt/symbolsk og sammenlikn svar.

Løsning. Vi begynner med $f(x) = 2x + 1$.

```
% Input: f(x) = 2x+1, a=0, b=1, n=50
% Output: int_a^b f dx med trapesmetoden
close all; clear all;
% Data
a = 0; b = 1; n = 50;
% Definerer x variabel
x = a:(b-a)/n:b;
% Definerer f(x)
f = 2*x +1;
% Tegner data
plot(x, f, 'LineWidth', 2);
title('y=2x+1');
grid on; xlabel('x'); ylabel('y');
% Regner ut integralet med trapesmetode
% f(1)+2f(2)+2f(3)+...+2f(100)+f(101)
% = 2*sum(f)-fart(1)-fart(101)
int = ((b-a)/(2*n))*(2*sum(f)-f(1)-f(n+1))
trapz(f)*(b-a)/n
```



FIGUR 25. $y = 2x + 1$, $x \in [0, 1]$.

Output:

```
int = 2
```

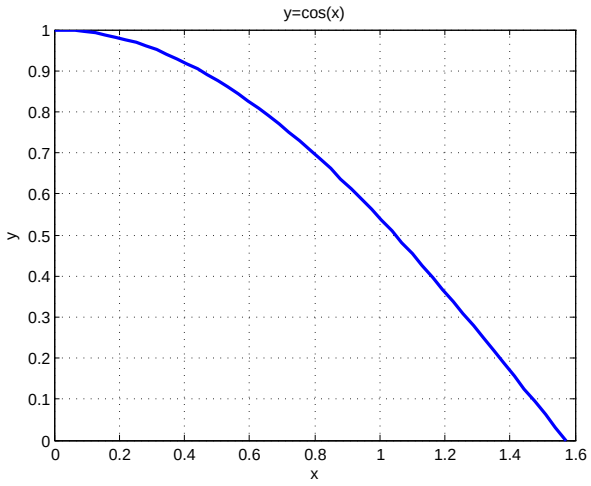
Symbolsk beregning av integralet:

```
>> syms x
>> f = 2*x+1;
>> int(f,0,1)

ans = 2
```

La oss regne ut integralet av $f(x) = \cos x$.

```
% Input: f(x) = cos(x), a=0, b=pi/2, n=100
% Output: int_a^b f dx med trapesmetoden
close all; clear all;
% Data
a = 0; b = pi/2; n = 50;
% Definerer x variabel
x = a:(b-a)/n:b;
% Definerer f(x)
f = cos(x);
% Tegner data
plot(x, f, 'LineWidth', 2);
title('y=cos(x)');
grid on; xlabel('x'); ylabel('y');
% Regner ut integralet med trapesmetode
% f(1)+2f(2)+2f(3)+...+2f(100)+f(101)
% = 2*sum(f)-fart(1)-fart(101)
int = ((b-a)/(2*n))*(2*sum(f)-f(1)-f(n+1))
```

Output:



FIGUR 26. $y = 2x + 1$, $x \in [0, 1]$.

```
int = 0.9999
```

Symbolsk beregning av integralet:

```
>> syms x
>> f = cos(x);
>> int(f,0,1)

ans = 1
```

Vi ser at trapesmetoden gir eksakt svar i første tilfelle. Grunnen til det er at funksjonen vi integrer er lineær, så at grafen kan approksimeres med trapeser eksakt. Jo større n vi tar, desto bedre blir approksimasjonen. Prøv og ta $n = 100$.

Til slutt merker vi at man også kunne bruke den innbygde funksjonen `trapz`

```
Z = trapz(Y) computes an approximation of the integral of Y
    via the trapezoidal method (with unit spacing).
    To compute the integral for spacing different from one,
    multiply Z by the spacing increment.
```

Vi skriver

```
>> trapz(f)*(b-a)/n
```

som gir (selvfølgelig) samme svar.

□

Kapitel 8

1. Bestem løsningen til differesiallikningen

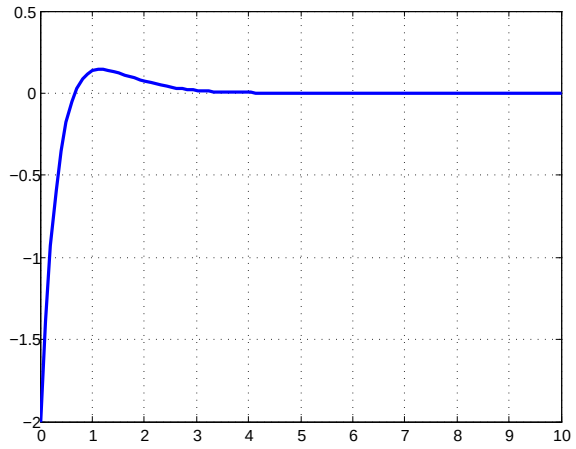
$$y'' + 4y' + 4y = 0, \quad y(0) = -2, \ y'(0) = 7.$$

Løsning.

```
% Input: likningen y"+4y'+4y=0,
%                    y(0)=-2, y'(0)=7
% Output: løsningen til likningen
close all;
% Definerer en symbolsk funksjon
syms y(t)
% Første derivert
Dy = diff(y);
% Andre derivert
D2y = diff(y,2);
% Løser likningen
y = dsolve(D2y +4*Dy+4*y == 0, y(0) == -2, Dy(0) == 7)
% Tegner løsningen
x = linspace(0,10,100);
plot(x, subs(y,t,x), 'LineWidth', 2)
grid on;
```

Output:

```
y = 3*t*exp(-2*t) - 2*exp(-2*t)
```



FIGUR 27. Løsningen til $y'' + 4y' + 4y = 0$, $y(0) = -2, y'(0) = 7$.

□

2. Bestem løsningen til differensiallikningen

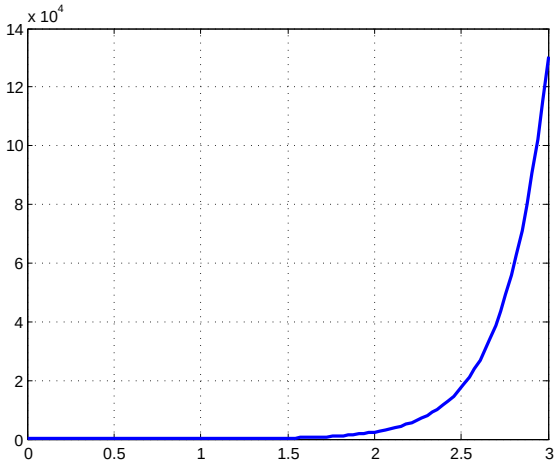
$$y'' - 2y' + 2y = 8e^{4t} - 2\cos(3t) + 5\sin(3t)$$
$$y(0) = 44/85,\ y'(0) = 0.$$

Løsning.

```
% Input: likningen y"-2y'+2y=f(t),
% f=8*exp(4t)-2*cos(3t)+5*sin(3t),
% y(0)=-2, y'(0)=7
% Output: løsningen til likningen
close all;
% Definerer en symbolsk funksjon
syms y(t)
% Første derivert
Dy = diff(y);
% Andre derivert
D2y = diff(y,2);
% Høyre side
f = 8*exp(4*t)-2*cos(3*t)+5*sin(3*t)
% Løser likningen
y = dsolve(D2y -2*Dy+2*y == f, y(0) == 44/85, Dy(0) == 0)
% Tegner løsningen
x = linspace(0,3,100);
plot(x, subs(y,t,x), 'LineWidth', 2)
grid on;
```

Output:

```
y = 3*t*exp(-2*t) - 2*exp(-2*t)
```



FIGUR 28. Løsningen til $y'' - 2y' + 2y = 8e^{4t} - 2\cos(3t) + 5\sin(3t)$, $y(0) = 44/85, y'(0) = 0$.

□

Kapitel 9

1. Løs et likningssystem

$$\begin{cases} x_1 - x_2 + 3x_3 = 3 \\ 4x_1 + 3x_2 + 4x_3 = 2 \\ 2x_2 - 3x_3 + 2x_4 = 0 \\ x_3 - x_4 = -1. \end{cases}$$

Løsning.
```
>> A = [1 -1 3 0; 4 3 4 0; 0 2 -3 2; 0 0 1 -1]
A =
     1    -1     3     0
     4     3     4     0
     0     2    -3     2
     0     0     1    -1

>> b = [3; 2; 0; -1]
b =
     3
     2
     0
    -1

>> x = A\b
x =
    0.3333
   -0.6667
    0.6667
    1.6667
```

□

2. Løs et likningssystem

$$\begin{cases} x_1 + x_2 + x_3 = 1 \\ x_1 + x_2 + 2x_3 = 3 \end{cases}$$

for hånd og ved hjelp av MATLAB. Har du fått samme svar? Forklar.

Løsning. Når vi løser likningssettet for hånd, subtraherer vi den andre likningen fra den første og får

$$x_3 = 2, \quad x_1 + x_2 = -1.$$

Det betyr at det finnes uendelig mange løsninger på formen $[x_1, -1 - x_1, 2]$, der $x_1 \in \mathbb{R}$.

```
>> A=[1 1 1; 1 1 2]
A =
     1     1     1
     1     1     2

>> b = [1;3]
b =
```

```
     1
     3
>> x = A\b
x =
          0
   -1.0000
    2.0000
```

MATLAB viser bare én av de uendelig mange løsninger, som tilsvarer $x_1 = 0$. Hvordan kan vi få MATLAB å vise alle løsningene?

Den generelle løsninger er en sum av løsningen til det homogene likningssystemet og en partikulær løsning av den inhomogene likningssytemet. For å løse det homogene likningssettet $Ax = 0$ skriver vi

```
>> null(A)
ans =
    -0.7071
    0.7071
    0.0000
```

Denne funksjonen regner ut *nullrommet* til en matrise, det vil si mengden av alle løsninger til $Ax = 0$. Tast inn `null` i `help` for å finne ut mer om funksjonen.

Den generelle løsningen blir da en lineær kombinasjon

$$[0, -1, 2] + C_1[-0.7071, 0.7071, 0]$$

som er det samme som

$$[C, -1 - C, 2], \quad C \in \mathbb{R}.$$

Den siste vektoren representerer alle løsninger vi fikk tidligere for hånd ($[x_1, -1 - x_1, 2]$, der $x_1 \in \mathbb{R}$). □

Kapitel 10

1. Regn ut egenverdier og egenvektorer til

$$A = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 2 & 2 \\ 0 & 0 & 2 \end{pmatrix}$$

Løsning.

```
>> A = [1 0 -1; 2 2 2; 0 0 2]
A =
     1     0    -1
     2     2     2
     0     0     2
>> [vec, val] = eig(A)
vec =
```

```
         0    0.4472   -0.7071
    1.0000   -0.8944         0
         0         0    0.7071
val =
     2     0     0
     0     1     0
     0     0     2
```

Den første egenverdien $\lambda_1 = 1$ har multiplisitet 1, mens den andre $\lambda_2 = 2$ har multiplisitet 2. Det betyr at det finnes to egenvektorer som tilsvarer denne egenverdien: $[0, 1, 0]$ og $[-0.7071, 0, 0.7071]$. Fordi $\det(\text{vec}) \neq 0$, er disse vektorene lineært uavhengige. □

2. Avgjør om matrisen

$$A = \begin{pmatrix} -2 & -3 & -3 \\ 4 & 3 & 1 \\ -2 & 0 & 2 \end{pmatrix}$$

er diagonaliserbar. Dersom den er det, finn den diagonale matrisen $D = P^{-1}AP$, der P er matrisen av egenvektorer.

Løsning.

Vi begynner ved å finne egenverdier og egenvektorer til A.

```
>> A = [-2 -3 -3; 4 3 1; -2 0 2]
A =
    -2    -3    -3
     4     3     1
    -2     0     2
>> [vec, val]=eig(A)
vec =
   -0.4575   -0.2673   -0.0000
    0.7625    0.8018   -0.7071
   -0.4575   -0.5345    0.7071

val =
    0.0000         0         0
         0    1.0000         0
         0         0    2.0000
```

Matrisen A har tre egenverdier av multiplisitet 1: $\lambda_1 = 0$, $\lambda_2 = 1$ og $\lambda_3 = 2$. Matrisen `vec` består av egenvektorer. Matrisen `vec` har $\det(vec) \neq 0$, så de tre vektorene er lineært uavhengige.

Følgelig er matrisen A diagonaliserbar og vi kan regne ut $D = \text{vec}^{-1}A\text{vec}$:

```
>> vec^(-1)*A*vec
ans =
   -0.0000   -0.0000    0.0000
    0.0000    1.0000   -0.0000
    0.0000    0.0000    2.0000
```

Resultatet er diagonalmatrisen med egenverdiene i diagonalen. □