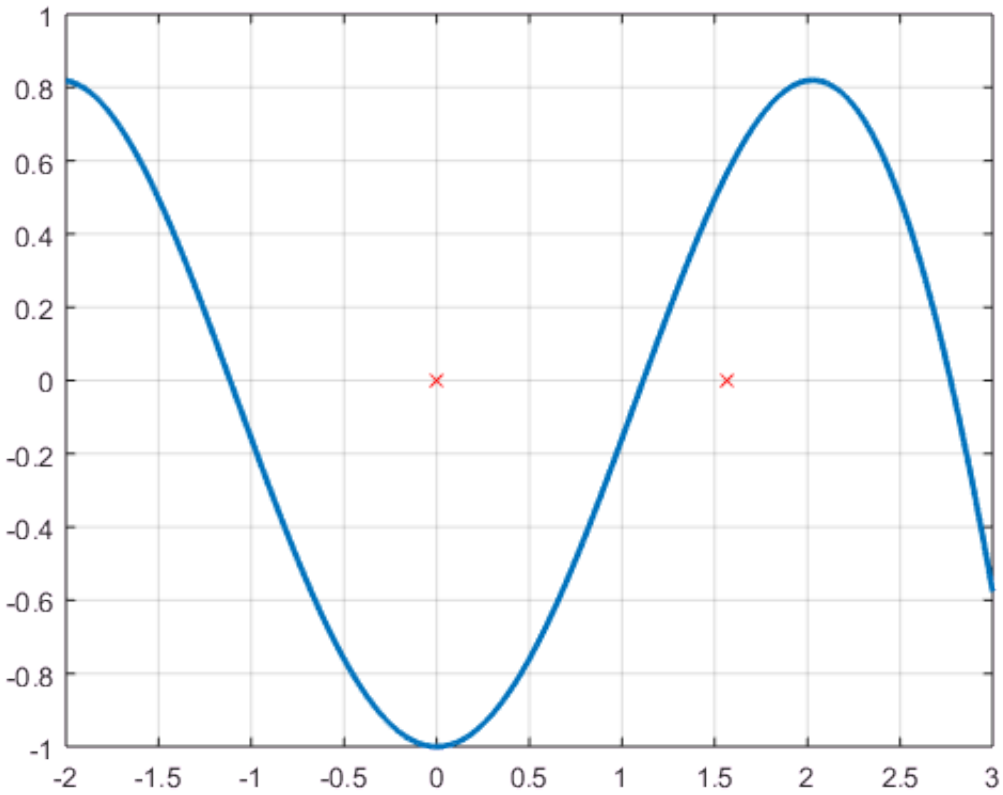
Innlevering nr. 3

Løysingsforslag

Oppgåve 1

Vi plottar funksjonen først:

```
x=-2:1e-2:3;
y=x.*sin(x)-1;
plot(x,y,'linewidth',2)
hold on
plot([0 pi/2],[0 0],'rx')
grid on
```



For det aktuelle nullpunktet, som skal ligge mellom 0 og $\pi/2$, kan $x_0 = 1$ vere eit greit utgongspunkt.

a) Vi implementerar Newtons metode med ei while-løkke og legg inn ein teljar som held styr på kor mange iterasjonar vi brukar:

```
% Implementering av Newtons metode.
% Funksjonen som skal vere null, den deriverte av denne,
% presisjonen og startpunktet for iterasjonen, x0, er hardkoda.

% Funksjon - og derivert
funk=@(x) x*sin(x)-1;
funkDeriv=@(x) sin(x)+x*cos(x);
```

```
% Startverdi for iterasjon
x=1;

% Preisjon
Pres=1e-5;

% Initerar den gamle x-verdien (til kva verdi som helst)
xGml=100;

% Initerar variabel som tel iterasjonane
Iterasjonar=0;

while abs(x-xGml) > Pres
  xGml=x;
  x=x-funk(x)/funkDeriv(x);
  Iterasjonar=Iterasjonar+1;
end

% Skriv resultat til skjerm
format long
x
```

```
x =    1.114157140871930
```

```
format short
Iterasjonar
```

```
Iterasjonar = 3
```

Vi ser at vi får svar med ein feil mindre enn 10^{-5} etter berre 3 iterasjonar.

Om vi no skal ha svaret med ein feil som er mindre enn 10^{-10}, endrar vi berre input-variabelen Pres i koden over til "1e-10". Svaret vi få får er

x=1.114157140871930

Det tok berre 4 iterasjonar å få denne presisjonen.

b) Aller først må vi skrive likninga på forma $x = g(x)$. Det er mange måtar å gjere det på. Vi kan for eksempel skrive ho som

$$x = \frac{1}{\sin x}$$

Fikspunktiterasjon kan implementerast på ein måte som er veldig likt det vi gjorde over. Men her treng vi ikkje ta med nokon derivert av funksjonen. Vi har derimot lagt til eit krav til i while-satsen. Dette legg føringar på kor mange iterasjonar vi kan tillate oss. Tanken er at skriptet skal stoppe dersom vi ikkje nærmar oss noko løysing. Dette kunne vi forresten godt ha gjort i implementeringa av Newtons metode også.

```
% Implementering av fikspunktiterasjon.
% Funksjonen som skal vere lik x, presisjonen og
% startpunktet for iterasjonen, x0, er hardkoda.

% Funksjon - og derivert
funk=@(x) 1/sin(x);
% Startverdi for iterasjon
x=1;
```

```
% Preisjon
Pres=1e-5;

% Initerar den gamle x-verdien (til kva verdi som helst)
xGml=100;

% Initerar variabel som tel iterasjonane
Iterasjonar=0;
ItMax=500;

while abs(x-xGml) > Pres & Iterasjonar<ItMax
  xGml=x;
  x=funk(x);
  Iterasjonar=Iterasjonar+1;
end

% Skriv resultat til skjerm
format long
if Iterasjonar <ItMax
  x
  Iterasjonar
else
  disp('Dette ser ikkje ut til å fungere')
end
```

```
x =    1.114154703730068
Iterasjonar =     18
```

```
format short
```

Vi ser at det fungerte. Men her måtte vi altså bruke fleire iterasjonar enn vi måtte med halveringsmetoden.

Vi kunne også ha formulert likninga på fleire andre måtar

$$\sin x = \frac{1}{x}$$

$$x = \arcsin\frac{1}{x} + n \cdot 2\pi \quad \text{eller} \quad x = \pi - \arcsin\frac{1}{x} + n \cdot 2\pi$$

der n er eit heiltal.

Om vi oppdaterar funksjonen i koden over til

funk=@(x) asin(1/x);

vil vi ikkje få noko svar og x_n blir ganske fort kompleks. Om vi derimot prøver oss med

funk=@(x) pi-asin(1/x);

får vi svaret

x=2.7716

Svaret er rett, det *er* eit nullpunkt, men det er eit anna enn det vi "sikta oss inn på".

Oppgåve 2

Desse oppgåvene løyser vi ved å setje opp totalmatrisa for likningssystema og så rekkeredusere denne til redusert trappeform.

Det blir knotete å skrive koreis vi gjer dette i eit Live Script, så det har vi gjort for hand. Sjå eige notat lenger bak.

MATLAB-kontrollen rekkereduksjonane, derimot, tar vi her:

a)

```
T=[1 2 3; 2 1 -2]
```

```
T =

     1     2     3
     2     1    -2
```

```
rref(T)
```

```
ans =

    1.0000         0   -2.3333
         0    1.0000    2.6667
```

```
-7/3
```

```
ans = -2.3333
```

```
8/3
```

```
ans = 2.6667
```

b)

```
A=[1 0 2; 0 3 3; 2 -2 2]; b=[1; 12; -6];
T=[A,b]
```

```
T =

     1     0     2     1
     0     3     3    12
     2    -2     2    -6
```

```
rref(T)
```

```
ans =

     1     0     2     1
     0     1     1     4
     0     0     0     0
```

c)

```
s1=[1; 1; 1]; s2= [1; 1; 0]; s3=[1; 0; 0]; b=[1; 0; -1];
T=[s1, s2, s3, b]
```

```
T =

     1     1     1     1
     1     1     0     0
     1     0     0    -1
```

```
rref(T)
```

```
ans =

     1     0     0    -1
     0     1     0     1
     0     0     1     1
```

Oppgåve 3

Dette kan implementerast iterativt; vi kan ta for oss søyle for søyle for stadig mindre likningssystem. Og på den måten blir sjølv relativt store likningssystem råd å handtere. Tanken er som følger:

Med totalmatrisa $T = [A|b]$ vil vi først sørge for at talet øvst til venstre er ulik null - og så at det blir én. Om første element i søyla er ulik null og ulik éin, gangar vi rekke 1 slik at dette talet blir ein. Hvis det er null, leitar vi gjennom søyla til vi finn eit tal ulik null - og byter rekker slik at dette kjem øvst. Deretter gjer vi det til éin. Dersom vi ikkje finn noko tal ulik null, betyr det at systemet ikkje har noko eintydig løysing, og vi gir oss der.

Dersom vi finn eit leiande tal, derimot, rekkeopererar vi slik at alle tala under den leiande einaren blir null.

Det fine no er at når vi er ferdig med søyle 1, gjer vi akkurat det same ein gong til - på ei mindre matrise. Totalmatrisa har no forma

$$T = \begin{pmatrix} 1 & * & * & \dots & * \\ 0 & * & * & \dots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & * & * & \dots & * \end{pmatrix}$$

der stjernene står for element i matrisa. Sidan alle tall under den leiande einaren i søyle 1 er null, vil ikkje rekkeoperasjonar endre søyle 1. Rekke 1 blir vil heller ikkje bli endra foreløpig. Dermed kan vi berre gjenta det vi gjorde over på undermatrisa som består av søylene frå og med søyle 2 - frå og med *rekke* 2 - og så sette denne inn igjen i den store totalmatrisa etterpå.

Og slik fortset vi heilt til vi evt. har kome til søyle (og rekke) n. Om vi no har funne leiande tal i kvar søyle, har vi eintydig løysing. Denne finn vi ved å rekkeredusere vidare slik at alle tala som står *over* leiande einarar også blir null - ikkje berre dei som står under. Løysinga av likningssystemet finn vi no i siste søyle i totalmatrisa.

Dette er tanken. Og implementeringa er absolutt gjerbar, men noko meir krevande enn dei andre implementeringane vi har gjort i dette kurset.

Denne implementeringa - med eksempel og litt dokumentasjon - finn de på dei siste sidene i løysingsforslaget.

Det heile kan gjerast enklare

Om vi tillet oss å bruke MATLAB sine funksjonar for rekne ut determinantar og for å invertere, derimot, blir implementeringa ganske enkel. Ei kvadratisk matrise er invertibel, har ein invers, hvis og berre hvis determinanten er ulik null. Med "ulik null" har eg i implementeringa under meint mindre enn 10^{-13} i absoluttverdi. Dersom A er invertibel, er løysinga av likningssystemet $x = A^{-1}\vec{b}$.

Implementeringa er gitt under:

```
% Skript som avgjer om et likningssystem på forma Ax=b
% med A kvadratisk har eintydig løysing eller ikkje.
% Løysinga blir skriven til skjerm dersom den er eintydig.
% Matrisa A og søylevektoren b er input.

% Les inn A og b
A=input('Gi koeffisientmatrisa (kvadratisk): ');
b=input('Gi høgresida (søylevektor): ');

% Sjekkar at formata er ok
[m n]=size(A);
[p q]=size(b);
if m~=n | q~=1 | p~=n
  disp('Her er det noko galt med formata.')
  return
end

% Determinanten
d=det(A);
% Undersøker om A er invertibel - og reknar ut x i så fall
if abs(d)<1e-13
  disp('Likningssystemet har ikkje noko eintydig løysing.')
else
  x=inv(A)*b   % Reknar ut løysinga som A^-1 b
end
```

```
x =

   -2.0000
    4.5000
   -2.0000
```

Vi kan også bruke rref-funksjonen for å avgjere om løysinga er eintydig - og kva ho er i så fall. Vi set opp totalmatrisa og rekkereduserar ho til redusert trappeform Om vi då har leiande einarar langs heile diagonalen i A-matrisa, er løysinga eintydig og vi finn ho i den siste søyle i totalmatrisa. Hvis ikkje, altså om eitt eller fleire av diagonalelementa er null, har vi ikkje eintydig løysing.

Denne framgogsmåten kan implementeriast slik:

```
% Skript som avgjer om et likningssystem på forma Ax=b
% med A kvadratisk har eintydig løysing eller ikkje.
% Løysinga blir skriven til skjerm dersom den er eintydig.
% Matrisa A og søylevektoren b er input.
```

```
% Les inn A og b
A=input('Gi koeffisientmatrisa (kvadratisk): ');
b=input('Gi høgresida (søylevektor): ');

% Sjekkar at formata er ok
[m n]=size(A);
[p q]=size(b);
if m~=n | q~=1 | p~=n
  disp('Her er det noko galt med formata.')
  return
end

% Set opp totalmatrisa og rekkereduserar ho til redusert trappeform
T=[A b];
T=rref(T);

% Undersøker om nokon av diagonalelementa er null
for i=1:n
  if T(i,i)==0
    disp('Likningssystemet har ikkje noko eintydig løysing.')
    return
  end
end

% Dersom vi ikkje har funne null-element, er løysigna eintydig-
% Vi skriv denne til skjerm
x=T(:,n+1)
```

```
x =

   -2.0000
    4.5000
   -2.0000
```

For å sjå eksempel på korleis kodane fungerar: Sjå lengst bak i løysingsforslaget; alle eksempla kan brukast i alle tre implementeringane. (Berre sjå bort frå utskriftene over.)

Oppgåve 4

a) Vi ser at med $h = 0.5$, kan $f'(x_1)$ estimerast slik ved hjelp av midtpunktsformelen:

$$f'(x_1) \approx \frac{f(x_1 + h) - f(x_1 - h)}{2h} = \frac{f(x_2) - f(x_0)}{2h}$$

Vi brukar dette og set det inn i den gitte likninga for $x = x_1$:

$$\frac{f(x_2) - f(x_0)}{2h} + x_1 f(x_1) = 3x_1^2 + x_1^4$$

Med $h = 0.5$, $f(x_0) = f(0) = 0$ og $x_1 = 0.5$ gir dette at

$$\frac{f_2 - f(0)}{2 \cdot 0.5} + 0.5 \cdot f_1 = 3 \cdot 0.5^2 + 0.5^4$$

$$0.5f_1 + f_2 = 0.8125$$

På same vis har vi at

$$\frac{f(x_3) - f(x_1)}{2h} + x_2 f(x_2) = 3x_2^2 + x_2^4$$

og

$$\frac{f(x_4) - f(x_2)}{2h} + x_3 f(x_3) = 3x_3^2 + x_3^4$$

Når vi set inn tala, får vi ut dei to neste likningane i det gitte likningssystemet.

For x_4 kan vi ikkje bruke midtpunktsformelen utan å gå forbi $x = x_4$. Om vi i staden brukar den gitte formelen, får vi

$$\frac{f(x_2) - 4f(x_3) + 3f(x_4)}{2h} + x_4 f(x_4) = 3x_4^2 + x_4^4$$

Når vi set inn tala, får vi at

$$\frac{f_2 - 4f_3 + 3f_4}{2 \cdot 0.5} + 2f_4 = 3 \cdot 2^2 + 2^4$$

$$f_2 - 4f_3 + 5f_4 = 28$$

b)

```
% Totalmatrisa
T=[.5 1 0 0 .8125
    -1 1 1 0 4
    0 -1 1.5 1 11.8125
    0 1 -4 5 28];
% Får ho på redusert trappeform
T=rref(T)
```

```
T =

    1.0000         0         0         0   -0.0355
         0    1.0000         0         0    0.8302
         0         0    1.0000         0    3.1343
         0         0         0    1.0000    7.9414
```

```
% Hentar ut siste søyle - og gjer ho til ei rekke
f=T(:,5).'
```

```
f =

   -0.0355    0.8302    3.1343    7.9414
```

c)

Vi sjekkar at $f(0) = 0$ først:

$$f(0) = 0^3 = 0$$

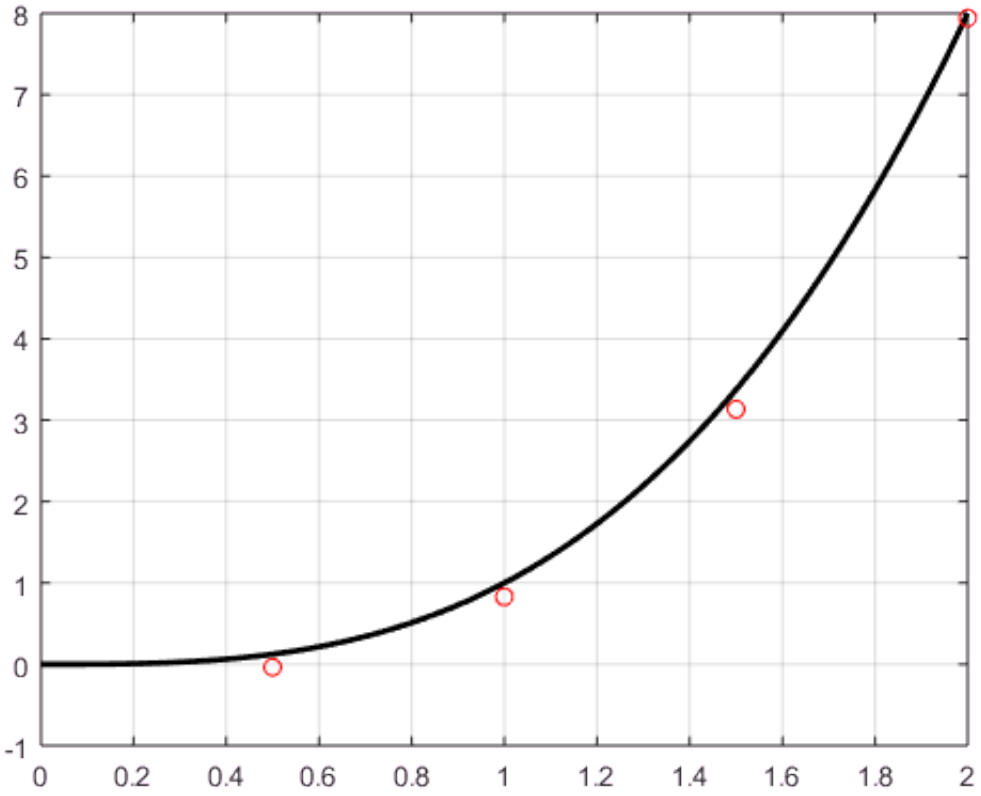
Så set vi inn den deriverte, $f'(x) = 3x^2$:

$$f'(x) + xf(x) = 3x^2 + x \cdot x^3 = 3x^2 + x^4$$

Vi ser at dette stemmer.

d) Når vi plottar, brukar vi f-vektoren som vi har tileigna

```
figure(2)
x=0:1e-2:2;
plot(x,x.^3,'k-','linewidth',2)
grid on
hold on
plot(0.5:0.5:2,f,'ro')
hold off
```



Vi ser at punkta ligg i nærleiken - sjølv om dei ikkje ligg akkurat oppå. Vi kunne ha gjort tilnærminga betre ved å ta med fleire punkt og dermed fått ein lågare h-verdi. Sjå gjerne kapittel 16 i numerikk-boka om du er interessert.

Oppg. 2

a) Totalmatrise:

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & -2 \end{pmatrix} \begin{matrix} -2 \\ \downarrow \end{matrix} \sim \begin{pmatrix} 1 & 2 & 3 \\ 0 & -3 & -8 \end{pmatrix} \leftarrow -\frac{1}{3} \sim$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & \frac{8}{3} \end{pmatrix} \begin{matrix} \hookrightarrow \\ -2 \end{matrix} \sim \begin{pmatrix} 1 & 0 & -\frac{7}{3} \\ 0 & 1 & \frac{8}{3} \end{pmatrix}$$

$x = -\frac{7}{3}$

$y = \frac{8}{3}$

b) Totalmatrise:

$$T = (A|\vec{b}) = \begin{pmatrix} 1 & 0 & 2 & 1 \\ 0 & 3 & 3 & 12 \\ 2 & -2 & 2 & -6 \end{pmatrix} \begin{matrix} \\ \leftarrow \frac{1}{3} \\ \leftarrow \frac{1}{2} \end{matrix} \sim$$

$$\begin{pmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 1 & 4 \\ 1 & -1 & 1 & -3 \end{pmatrix} \begin{matrix} \\ \\ -1 \\ \downarrow \end{matrix} \sim \begin{pmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 1 & 4 \\ 0 & -1 & -1 & -4 \end{pmatrix} \begin{matrix} \\ \\ 1 \\ \downarrow \end{matrix} \sim$$

$$\begin{pmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Med $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ har vi at

$x_1 + 2x_3 = 1$ og $x_2 + x_3 = 4$

$x_1 = 1 - 2x_3$ og $x_2 = 4 - x_3$

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1-2x_3 \\ 4-x_3 \\ x_3 \end{pmatrix} = \underline{\begin{pmatrix} 1 \\ 4 \\ 0 \end{pmatrix} + x_3 \begin{pmatrix} -2 \\ -1 \\ 1 \end{pmatrix}}$$

der $\underline{x_3 \text{ er fri.}}$

Geometrisk er løysinga ei linje i rommet. Linja går gjennom punktet $(1, 4, 0)$ og ho er parallell med vektoren $\begin{pmatrix} -2 \\ -1 \\ 1 \end{pmatrix}$.

c) $$x_1 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + x_2 \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + x_3 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$$

Likninga kan også skrivast som

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_3 \\ x_2 \\ x_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$$

Totalmatrise:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 \end{pmatrix} \begin{matrix} \\ \leftarrow -1 \\ \leftarrow -1 \end{matrix} \sim \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & -1 & -1 \\ 0 & -1 & -1 & -2 \end{pmatrix} \begin{matrix} \\ \leftarrow \\ \leftarrow \end{matrix} \sim$$

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & -1 & -2 \\ 0 & 0 & -1 & -1 \end{pmatrix} \begin{matrix} \\ \leftarrow (-1) \\ \leftarrow (-1) \end{matrix} \sim \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{matrix} \leftarrow \\ \leftarrow \\ -1 \end{matrix} \sim$$

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{matrix} \leftarrow \\ -1 \\ \\ \end{matrix} \sim \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

$\underline{x_1 = -1, \; x_2 = x_3 = 1}$

Kontroll:

$$(-1) \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + 1 \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + 1 \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -1+1+1 \\ -1+1+0 \\ -1+0+0 \end{pmatrix} = \underline{\begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}}$$ Ok

Oppgåve 3

Implementeringa er delt inn i 7 separate filer. Hovudskriptet, **KlassifisereLoeysing.m**, kallar 6 andre funksjonsfiler. Først kallar ho ei rutine som kontrollerer at koeffisientmatrisa *A* er kvadratisk og at høgresida b, har rett format. Denne heiter **KontrollFormat.m**.

Det neste som skjer, er at vi rekkereduserar totalmatrisa, *T=[A | b]*, til trappeform. Dette skjer i funksjonsfila **Trapp.m**. Dersom der skulle dukke opp søyler utan leiande tal, vil køyringa stoppe og vi konkluderer at systemet ikkje har eintydig løysing. Denne konklusjonen blir skriven til skjerm i så fall. Dette er den viktigaste delen av implementeringa.

Dersom vi finn leiande tal i kvar søyle, fortset vi med rekkereduksjonar til vi har totalmatrisa på *redusert* trappeform. Dette blir gjort i funksjonen **RedusertTrapp.m**. Som i Trapp.m er dette gjort blokk for blokk. Men her går vi frå små blokker til store – ikkje motsett, som i Trapp.m.

Til slutt, dersom skriptet når så langt, vil løysinga bli skriven til skjerm.

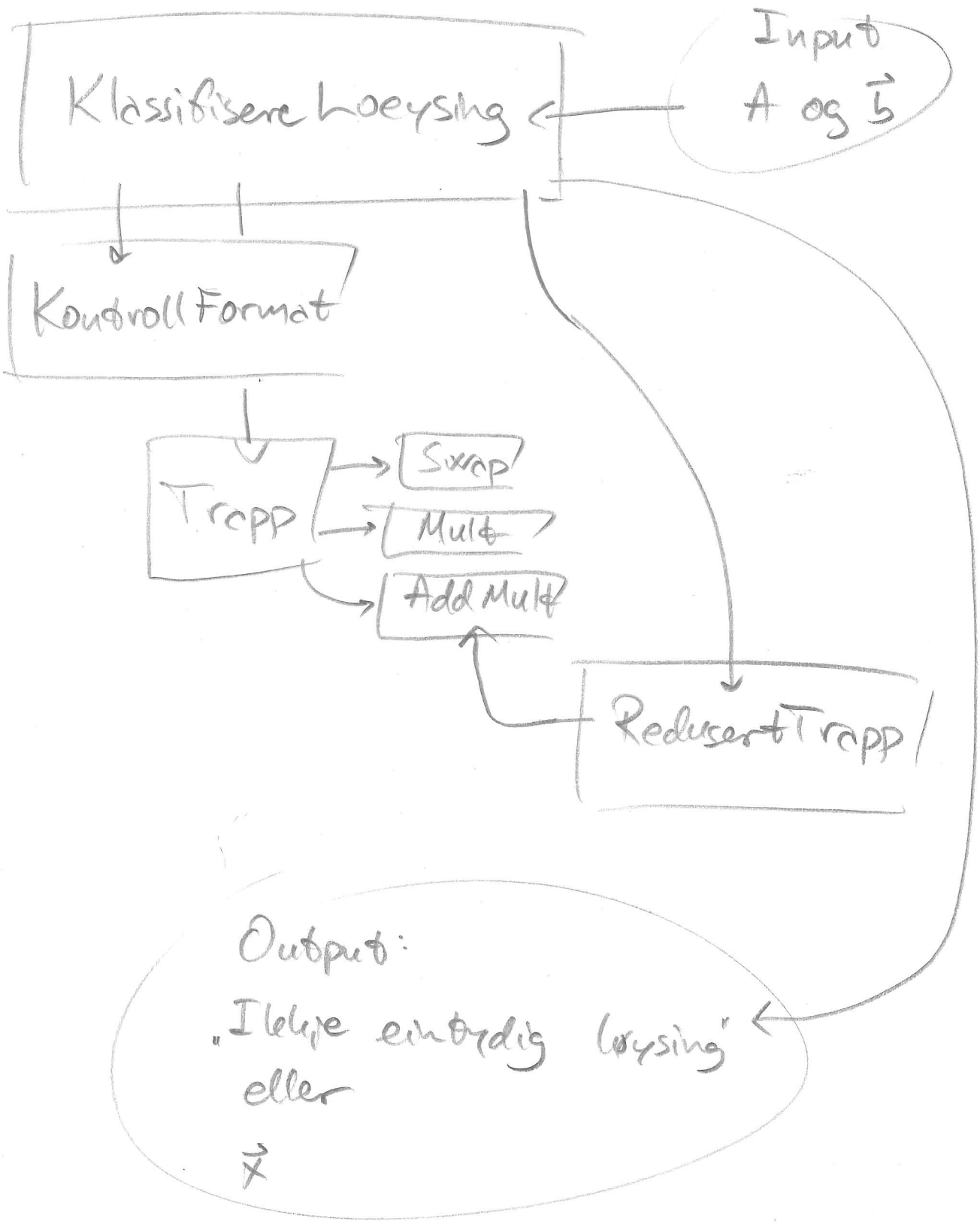
Trapp.m og RedusertTrapp.m brukar funksjonsfilene **Swap.m**, **Mult.m** og **AddMult.m**, som implementerar dei tre typane vi har av rekkeoperasjonar – altså å byte om rekkefølga på to rekker, å gange ei rekke med eit tal og å legge eit multiplum av ei rekke til ei anna.

For å få eit overblikk, har vi tatt med eit slags flytskjema på neste.

På sidene etter er kjeldekoden lagt ved.

Til sist kjem eksempel på bruk.

Flytskjema


Klassifisere Løsning
Input
A og $\vec{b}$
Kontroll Format
Trapp
Swap
Mult
Add Mult
Redusert Trapp
Output:
"Ikke entydig løsning"
eller
$\vec{x}$

```
% Skript som les inn eit genrelt likningssystem
% med n likningar i n ukjende. Skriptet avgjer om systemet
% eintydig løysing eller ikkje. Dersom løysinga er
% eintydig, vil ho rekne ho ut.
% Input er koeffisentmatrisa A og høgresida b,
% der likningssystemet er gitt på forma A*x=b.
% der x og b er søylevektorar og koeffisentmatrisa A
% må vere kvadratisk.
% Skriptet gir søylevektoren x til svar - dersom denne
% er veldefinert og eintydig.

% Les inn koeffisientmatrise
A=input('Gi koeffisientmatrisa (kvadratisk) ');
% Les inn høgresida
b=input('Gi høgresida i likningssystemet som ein søylevektor ');

% Sjekkar at formata er ok
[n Stop]=KontrollFormat(A,b);       % n er dimensjonen til A
% Stoppar skriptet dersom dei ikkje er det
if Stop
  return
end

% Set opp totalmatrisa
T=[A,b];

% Rekkereduserar denne til trappeform - dersom der er leiande
% tal i kvar søyle i A.
[T Stop]=Trapp(T,n);
% Stoppar skriptet dersom løysinga ikkje er eintydig
if Stop
  return
end

% Vi fortset med rekkereduksjon til ho er på redusert trappeform
T=RedusertTrapp(T,n);

% Hentar ut løysinga og skriv ho til skjerm
x=T(:,n+1)
```

Published with MATLAB® R2016b

```
function [n Stop]=KontrollFormat(A,b)

% Funksjon som kontrollerar at koeffisientmatrisa A
% for eit likningssystem er kvadratisk - og at vektoren
% på høgre side, b, er ein søylevektor med rett lengde.
% Dersom dette ikkje er oppfylt, vil køyringa stoppe.
% Funksjonsfila returnerar n, som er talet på rekker i A.
% Funksjonsfila returnerar også den logiske variabelen Stop,
% som er 1 dersom det er noko feil med formata på A og b.

% Les formata på matrisa og vektoren
[n m]=size(A);
[p q]=size(b);

% I utgangspunktet set vi Stop til 1
Stop=logical(1);

if n~=m
  disp('Koeffisientmatrisa må vere kvadatisk')
  return
elseif q~=1
  disp('Vektoren må vere ein søylevektor')
  return
elseif p~=n
  disp('Vektoren må ha like mange element som matrisa har rekker')
  return
else
  Stop=logical(0);  % Set Stop til 0 dersom ingenting er feil
end
```

Published with MATLAB® R2016b

```
function [T Stop]=Trapp(T,n)

% Funksjonsfila går gjennom søylene i totalmatrisa for eit
% likningssystem med like mange likningar som variablar.
% Ho avgjer om systemet har eintydig løysing. Dersom ho har det,
% vil fila returnere totalmatrisa redusert til trappeform der
% alle leiande tal i tillegg er éin.
% Fila returnerer også den logiske variabelen Stop, som
% er éin dersom der ikkje er leiande tal i kvar søyle (utanom
% den siste).
% I tillegg til sjølve matrisa, T, er talet på rekker, n, input.

Stop=logical(0);                % Set Stop til 0 - i utgongspunktet

% Jobbar seg gjennom søylene i totalmatrisa og avgjer om der
% er leiande tal i kvar søyle i koeffisientmatrisa
for i=1:n
  Blokk=T(i:n,i:(n+1));          % Hentar ut underblokk
  nBlokk=n-i+1;                  % Talet på rekker i undeblokka
  Soeyle=Blokk(:,1);             % Hentar ut første søyle i blokka
  % Finn første element i søyla ulik null - hvis der er eit
  j=1;
  while abs(Soeyle(j))<1e-13 & j<nBlokk % Undersøker opp til nest
 siste
    j=j+1;
  end
  if j==nBlokk & abs(Soeyle(j))<1e-13     % Undersøker evt. siste
 element
    % Dersom alle elementa i søyla er null, gir vi oss her
    disp('Likningssystemet har ikkje noko eintydig løysing.')
    Stop=logical(1);
    return
  elseif j>1
    Blokk=Swap(Blokk,1,j);       % Set det leiande talet øvst evt.
  end
  Faktor=1/Blokk(1,1);           % Bestemmer faktor vi skal gange med
  Blokk=Mult(Blokk,1,Faktor);    % Gjer det leiande talet til 1
  % Gjer alle tala under det leiande talet til null
  for k=(j+1):nBlokk
    faktor=-Blokk(k,1);          % Bestemmer faktoren
    Blokk=AddMult(Blokk,1,k,faktor);    % Legg til mult. av rekke
  end
  T(i:n,i:(n+1))=Blokk;          % Set blokka inn att i totalmatrisa
end
```

Published with MATLAB® R2016b

```
function T=RedusertTrapp(T,n)

% Funksjonsfil som tar inn ei totalmatrise på trappeform og
% returnerar ei totalmatrise på redusert trappeform.
% Input må tilsvare like mange likningar som variablar, altså
% må ho ha formatet n gonger n+1 (n er også input). Vidare må alle
% søyler utanom den siste ha eit leiande tal, og dessee skal vere éin.

% Går gjennom matrisa rekke for rekke frå den siste til den 2.
for i=n:-1:2
  Blokk=T(1:i,i:(n+1));        % Hentar ut underblokk
  % Gjer alle tala over leiande einar til 0
  for j=1:(i-1)
    faktor=-Blokk(j,1);        % Bestemmer faktor vi skal gange med
    Blokk=AddMult(Blokk,i,j,faktor);
  end
  T(1:i,i:(n+1))=Blokk;        % Set blokka inn att i totalmatrisa
end
```

Published with MATLAB® R2016b

```
function M=Swap(A,m,n)

% Funksjon som byter om på rekke m og rekke n
% i matrisa A

M=A;                    % Kopierar matrisa
% Byter rekker:
M(m,:)=A(n,:);
M(n,:)=A(m,:);
```

Published with MATLAB® R2016b

```
function M=Mult(A,n,faktor)

% Funksjon som gangar rekke n i matrisa A
% med ein faktor

M=A;                    % Kopierar matrisa
M(n,:)=faktor*A(n,:);   % Utfører rekkeoperasjon
```

Published with MATLAB® R2016b

```
function M=AddMult(A,m,n,faktor)

% Funksjon som legg ein faktor gonger rekke m til
% rekke n i matrisa A.

M=A;                    % Kopierar matrisa
M(n,:)=A(n,:)+faktor*A(m,:);    % Utfører rekkeoperasjonen
```

Published with MATLAB® R2016b

Testing av implementeringa

Vi kontrollerar først rutina som kontrollerar formata verkar:

```
>> A=[1 2 3; 0 1 2]
A =
     1     2     3
     0     1     2
>> b=[1; 2]
b =
     1
     2
>> KlassifisereLoeysing
Gi koeffisientmatrisa (kvadratisk) A
Gi høgresida i likningssystemet som ein søylevektor b
Koeffisientmatrisa må vere kvadratisk
>> A=[1 2 3; 0 1 2; -1 -2 -3]
A =
     1     2     3
     0     1     2
    -1    -2    -3
>> b=[1; 2; 3; 4]
b =
     1
     2
     3
     4
>> KlassifisereLoeysing
Gi koeffisientmatrisa (kvadratisk) A
Gi høgresida i likningssystemet som ein søylevektor b
Vektoren må ha like mange element som matrisa har rekker
```

Dette ser ut til å stemme

Så testar vi eit par elementære eksempel:

```
>> KlassifisereLoeysing
Gi koeffisientmatrisa (kvadratisk) zeros(3,3)
Gi høgresida i likningssystemet som ein søylevektor zeros(3,1)
Likningssystemet har ikkje noko eintydig løysing.
>> KlassifisereLoeysing
Gi koeffisientmatrisa (kvadratisk) eye(4)
Gi høgresida i likningssystemet som ein søylevektor [1; 2; 3; 4]
x =
     1
     2
     3
     4
```

Vi testar eit par eksempel til med singulære koeffisientmatriser:

```
>> A=[1 2; 2 4];
>> b=[0; 0];
>> KlassifisereLoeysing
Gi koeffisientmatrisa (kvadratisk) A
Gi høgresida i likningssystemet som ein søylevektor b
Likningssystemet har ikkje noko eintydig løysing.
>> A=[1 0 1; 0 2 0; 1 -2 1];
>> b=[0; 0; 0];
>> KlassifisereLoeysing
Gi koeffisientmatrisa (kvadratisk) A
Gi høgresida i likningssystemet som ein søylevektor b
Likningssystemet har ikkje noko eintydig løysing.
```

Til sist testar vi eit par større matriser med tilfeldige tal – og kontrollerar at `rref`-funksjonen i MATLAB gir same resultat:

```
>> A=rand(4,4)
A =
```

```
    0.8147    0.6324    0.9575    0.9572
    0.9058    0.0975    0.9649    0.4854
    0.1270    0.2785    0.1576    0.8003
    0.9134    0.5469    0.9706    0.1419
>> b=rand(4,1)
b =
    0.4218
    0.9157
    0.7922
    0.9595
>> KlassifisereLoeysing
Gi koeffisientmatrisa (kvadratisk) A
Gi høgresida i likningssystemet som ein søylevektor b
x =
   17.2819
    0.8395
  -15.9067
    1.0883
>> rref([A,b])
ans =
    1.0000         0         0         0   17.2819
         0    1.0000         0         0    0.8395
         0         0    1.0000         0  -15.9067
         0         0         0    1.0000    1.0883
>> A=rand(5,5)
A =
    0.6557    0.7577    0.7060    0.8235    0.4387
    0.0357    0.7431    0.0318    0.6948    0.3816
    0.8491    0.3922    0.2769    0.3171    0.7655
    0.9340    0.6555    0.0462    0.9502    0.7952
    0.6787    0.1712    0.0971    0.0344    0.1869
>> b=rand(5,1)
```

```
b =

    0.4898

    0.4456

    0.6463

    0.7094

    0.7547

>> KlassifisereLoeysing

Gi koeffisientmatrisa (kvadratisk) A

Gi høgresida i likningssystemet som ein søylevektor b

x =

    0.9484

    1.6954

   -0.6252

   -0.9387

   -0.4614

>> rref([A,b])

ans =

    1.0000         0         0         0         0    0.9484

         0    1.0000         0         0         0    1.6954

         0         0    1.0000         0         0   -0.6252

         0         0         0    1.0000         0   -0.9387

         0         0         0         0    1.0000   -0.4614
```

Så langt ser dette lovande ut.