Matematikk 1000

Øvingsoppgaver i numerikk – leksjon 8

Matriser

Løsningsforslag

Oppgave 1 – Redusert trappeform og løsning av lineære likningssystemer

a) Totalmatrisa blir

$$\begin{bmatrix} 5 & 1 & 1 & 2 & 2 \\ 1 & -1 & 1 & 7 & 30 \\ 2 & 1 & -5 & 0 & -16 \\ 2 & 0 & 1 & 3 & 17 \end{bmatrix}.$$

Vi tilordner dette i MATLAB:

```
>> format compact
>> T=[5 1 1 -2 2
1 -1 1 7 30
2 1 0 -5 -16
2 0 1 3 17]
T =
     5     1     1    -2     2
     1    -1     1     7    30
     2     1     0    -5   -16
     2     0     1     3    17
```

b) Vi lar MATLAB ta seg av rekkereduseringa:

```
>> rref(T)
ans =
     1     0     0     0     1
     0     1     0     0     2
     0     0     1     0     3
     0     0     0     1     4
```

Svar: $x_1 = 1, x_2 = 2, x_3 = 3$ og $x_4 = 4$. På vektor-form:

$$\vec{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} .$$

c) Vi setter opp totalmatrisa og bruker `rref`-funksjonen:

```
>> T=[1 2 0 -2 1 0; 2 -1 3 1 0 7; 3 1 3 1 -5 -8]
T =
     1     2     0    -2     1     0
     2    -1     3     1     0     7
     3     1     3     1    -5    -8
>> rref(T)
ans =
    1.0000         0    1.2000         0    0.2000    2.8000
         0    1.0000   -0.6000         0   -2.6000   -8.9000
         0         0         0    1.0000   -3.0000   -7.5000
```

Det gir

$$\begin{aligned} x_1 + 1.2x_3 + 0.2x_5 &= 2.8 \\ x_2 - 0.6x_3 - 2.6x_5 &= -8.9 \\ x_4 - 3x_5 &= -7.5 \end{aligned}$$

som igjen gir:

$$\begin{aligned} x_1 &= 2.8 - 1.2x_3 - 0.2x_5 \\ x_2 &= -8.9 + 0.6x_3 + 2.6x_5 \\ x_4 &= -7.5 + 3x_5 \end{aligned}$$

der x_3 og x_5 er frie. På parametrisk vektor-form:

$$\vec{x} = \begin{bmatrix} 2.8 \\ -8.9 \\ 0 \\ -7.5 \\ 0 \end{bmatrix} + x_3 \begin{bmatrix} -1.2 \\ 0.6 \\ 1 \\ 0 \\ 0 \end{bmatrix} + x_5 \begin{bmatrix} -0.2 \\ 2.6 \\ 0 \\ 3 \\ 1 \end{bmatrix} .$$

Oppgave 2 – Med og uten punktum

a) Vi kan for eksempel tilordne A og B slik:

```
>> A=[1 2 3; 4 5 6; 7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
>> B=[-2 -1 0; 0 1 0; 7 -2 1]
B =
    -2    -1     0
     0     1     0
     7    -2     1
```

b) Vi regner ut produktene i kommandovinduet:

```
>> A.*B
ans =
    -2    -2     0
     0     5     0
    49   -16     9
>> A*B
ans =
    19    -5     3
    34   -11     6
    49   -17     9
```

Vi ser at det i første produktet har matrisene blitt multiplisert elementvis; element 1, 1 i produktet er for eksempel $1 \cdot (-2)$ – altså element 1, 1 i A ganger element 1, 1 i B. Den andre produktet, uten punktum, er matriseproduktet:

$$AB = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} -2 & -1 & 0 \\ 0 & 1 & 0 \\ 7 & -2 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} -2+0+21 & -1+2-6 & 0+0+3 \\ -8+0+42 & -4+5-12 & 0+0+6 \\ -14+0+63 & -7+8-18 & 0+0+9 \end{bmatrix} = \begin{bmatrix} 19 & -5 & 3 \\ 34 & -11 & 6 \\ 49 & -17 & 9 \end{bmatrix}$$

Oppgave 3 – Noen spesielle matriser

Svarene i deloppgave a) - e) blir:

```
>> eye(3)
ans =
     1     0     0
     0     1     0
     0     0     1
>> ones(2,3)
ans =
     1     1     1
     1     1     1
>> -2*ones(2,3)
ans =
    -2    -2    -2
    -2    -2    -2
>> ones(3,3)-eye(3)
ans =
     0     1     1
     1     0     1
     1     1     0
>> diag(1:4)
ans =
     1     0     0     0
     0     2     0     0
     0     0     3     0
     0     0     0     4
```

I deloppgave e) og f) lager vi matriser med tilfeldige tall.

```
>> rand(2,2)
ans =
    0.8143    0.9293
    0.2435    0.3500
>> rand(2,2)
ans =
    0.1966    0.6160
    0.2511    0.4733
>> rand(2,2)
ans =
    0.3517    0.5853
    0.8308    0.5497
```

"`» rand(2,2)`" gir ei 2×2-matrise med tilfeldige tall mellom 0 og 1. Merk at vi får ei ny matrise hver gang selv om vi skriver den samme kommandoen.

Vi regner ut noen matriser med kommandoen gitt i deloppgave f):

```
>> 10*(rand(2,2)-0.5*ones(2,2))
ans =
   -3.3435   -2.3703
    1.0198    1.5408
>> 10*(rand(2,2)-0.5*ones(2,2))
ans =
    1.8921   -0.4946
    2.4815   -4.1618
>> 10*(rand(2,2)-0.5*ones(2,2))
ans =
   -2.7102   -3.4762
    4.1334    3.2582
>> 10*(rand(2,2)-0.5*ones(2,2))
ans =
    0.3834   -4.2182
    4.9613   -0.5732
```

Det ser ut til at vi får laga 2×2-matriser med tilfeldige tall i intervallet $[-5, 5]$. Når vi trekker `0.5*ones(2,2)` fra `rand(2,2)`, får vi tilfeldige tall mellom -0.5 og 0.5. Ganger vi dette med 10, får vi tilfeldige tall mellom -5 og 5.

Oppgave 4 – Testing av "regneregler"

I hver oppgave genererer vi tilfedige matriser A, B og eventuelt også C på samme måte som i deloppgave 2 f).

a)
```
>> A=10*(rand(3,3)-0.5*ones(3,3));
>> B=10*(rand(3,3)-0.5*ones(3,3));
>> A+B-(B+A)
ans =
     0     0     0
     0     0     0
     0     0     0
>> A=10*(rand(3,3)-0.5*ones(3,3));
>> B=10*(rand(3,3)-0.5*ones(3,3));
>> A+B-(B+A)
ans =
     0     0     0
     0     0     0
     0     0     0
>> A=10*(rand(3,3)-0.5*ones(3,3));
>> B=10*(rand(3,3)-0.5*ones(3,3));
```

```
>> A+B-(B+A)
ans =
     0     0     0
     0     0     0
     0     0     0
```

Her har vi laga tilfeldige A- og B-matriser tre ganger. Hver gang har vi fått at differansen $A + B - (B + A)$ blir nullmatrisa. Dette tyder på at regneregelen er rett.

b)
```
>> A=10*(rand(3,3)-0.5*ones(3,3));
>> B=10*(rand(3,3)-0.5*ones(3,3));
>> C=10*(rand(3,3)-0.5*ones(3,3));
>> (A*B)*C-A*(B*C)
ans =
   1.0e-14 *
    0.3553    0.7105    0.5329
         0   -0.7105   -0.1776
    0.3553         0    0.1776
>> A=10*(rand(3,3)-0.5*ones(3,3));
>> B=10*(rand(3,3)-0.5*ones(3,3));
>> C=10*(rand(3,3)-0.5*ones(3,3));
>> (A*B)*C-A*(B*C)
ans =
   1.0e-13 *
    0.0711   -0.0355    0.1066
    0.1421    0.0711         0
    0.0711         0         0
>> A=10*(rand(3,3)-0.5*ones(3,3));
>> B=10*(rand(3,3)-0.5*ones(3,3));
>> C=10*(rand(3,3)-0.5*ones(3,3));
>> (A*B)*C-A*(B*C)
ans =
   1.0e-13 *
    0.1421         0   -0.0711
         0    0.1421   -0.2842
   -0.1066         0    0.0355
```

Også her har vi testa regneregelen med tre ulike sett av tilfeldige matriser. Matrisene har blitt skrevt på en slik måte at alle tall i matrisa skal ganges med det tallet som står øverst. Hver gang har vi fått matriser der dette tallet er 10^{-13} eller mindre. Med andre ord er alle elementene sammenlignbare med maskinpresisjonen (`eps`). Derfor kan vi ikke uten videre si at dette er ulik null. Dette resultatet tyder altså på at regneregelen er rett.

c)
```
>> A=10*(rand(3,3)-0.5*ones(3,3));
>> B=10*(rand(3,3)-0.5*ones(3,3));
>> (A*B).'-(A.'*B.')
ans =
```

```
   -7.6774  -11.3022   24.8796
  -10.6053    0.2678  -15.6531
   -1.1223   38.7436    7.4096
```

Dette resultatet er slett ikke lik null-matrisa. Vi har funnet et moteksempel som viser tydelig at "regneregelen" ikke stemmer.

d) I første omgang kan vi fortsette med samme `A` og `B`:

```
>> (A*B).'-(B.'*A.')
ans =
     0     0     0
     0     0     0
     0     0     0
>> A=10*(rand(3,3)-0.5*ones(3,3));
>> B=10*(rand(3,3)-0.5*ones(3,3));
>> (A*B).'-(B.'*A.')
ans =
     0     0     0
     0     0     0
     0     0     0
>> A=10*(rand(3,3)-0.5*ones(3,3));
>> B=10*(rand(3,3)-0.5*ones(3,3));
>> (A*B).'-(B.'*A.')
ans =
     0     0     0
     0     0     0
     0     0     0
```

Mye tyder på at $(AB)^T$ faktisk *er* lik $B^T A^T$.

e)
```
>> det(A*B)-det(A)*det(B)
ans =
  -3.9080e-13
>> A=10*(rand(3,3)-0.5*ones(3,3));
>> B=10*(rand(3,3)-0.5*ones(3,3));
>> det(A*B)-det(A)*det(B)
ans =
  -4.5475e-13
>> A=10*(rand(3,3)-0.5*ones(3,3));
>> B=10*(rand(3,3)-0.5*ones(3,3));
>> det(A*B)-det(A)*det(B)
ans =
   9.0949e-13
```

Svaret, som denne gangen blir et tall, ikke ei matrise, ser ut til å bli veldig nær null. Det ser ut til at regneregelen stemmer.

f) Vi skal teste om vi får ulike svar om vi regner ut potensene med eller uten punktum:

```
>> A^4-A.^4
ans =
   1.0e+03 *
    0.1356   -0.9036   -0.5729
   -0.7370    1.2437    0.8025
   -0.3747    0.4655    0.2475
```

Som vi ser av faktoren "`1.0e+03`", altså 10^3, er denne matrisa langt fra null; regnereglen er feil. A^4 er altså noe helt anna enn det vi får om vi opphøyer hvert element i A i 4.

g) Vi kan lage 3×3-diagonalmatriser med tilfeldige tall langs diagonalen slik:

```
>> x=10*(rand(1,3)-0.5*ones(1,3))
x =
    1.9811    1.6653   -3.2187
>> D=diag(x)
D =
    1.9811         0         0
         0    1.6653         0
         0         0   -3.2187
```

Vi tester regneregelen fra f) igjen – denne gangen for diagonalmatriser:

```
>> D^4-D.^4
ans =
   1.0e-13 *
    0.0178         0         0
         0         0         0
         0         0    0.1421
>> x=10*(rand(1,3)-0.5*ones(1,3));
>> D=diag(x);
>> D^4-D.^4
ans =
   1.0e-13 *
         0         0         0
         0         0         0
         0         0   -0.1421
>> x=10*(rand(1,3)-0.5*ones(1,3));
>> D=diag(x);
>> D^4-D.^4
ans =
   1.0e-13 *
         0         0         0
         0   -0.0003         0
         0         0    0.2842
```

Det ser ut til at det er enklere å regne ut potenser av diagonalmatriser enn andre matriser.

Ekstra-oppgave

Denne oppgava går ut på å lage ei funksjonfil som rekkereduserer ei matrise til redusert trappeform. La det være sagt med én gang: Dette er *ikke* ei triviell oppgave. Den kan nok løses på mange måter, og den måten jeg har gjort det på kan helt sikkert forbedres.

Aller først: Vi skal gjøre mange rekkeoperasjoner på matrisa det er snakk om. Til å gjøre dette, har jeg laget funksjonsfiler som utfører disse. Ved å kalle disse i stedet for å gjøre disse operasjonene "for hånd" i hovedfila, blir nok koden noe mer ryddig. Som kjent har vi tre typer rekkepoerasjoner: 1) bytte om to rekker, 2) gange ei rekke med et tall (ulik null) og 3) legge et multiplum av ei rekke til ei anna rekke. Disse operasjonene har jeg implementert slik:

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

% Funksjon som byter om rekke m og n i ei matrise.

% Sjekker at verdien for m og n er ok
Rekker=size(A,1);
if m>Rekker | n>Rekker
    disp(['Matrisa har bare ',num2str(Rekker),' rekker.'])
    return
elseif m<0 | n<0
    disp('Rekkenumrene må være positive.')
    return
end

% Kopierer matrisa
M=A;
% Oppdaterer rekke m
M(n,:)=A(m,:);
% Oppdaterer rekke n
M(m,:)=A(n,:);
```

```
function M=Mult(A,k,m)

% Funksjon som multipliserer rekke m med k.

% Kontrollerer at k ikke er null
if k==0
    disp('Ikke tillatt; k kan ikke være null.')
    return
end

% Sjekker at verdien for m er ok
```

```
Rekker=size(A,1);
if m>Rekker
    disp(['Matrisa har bare ',num2str(Rekker),' rekker.'])
    return
elseif m<0
    disp('Rekkenummeret må være positivt.')
    return
end

% Kopierer matrisa
M=A;
% Ganger rekke m med k
M(m,:)=k*A(m,:);
```

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

% Funksjon som adderer k ganger rekke m til rekke n.

% Sjekker at verdien for m og n er ok
Rekker=size(A,1);
if m>Rekker | n>Rekker
    disp(['Matrisa har bare ',num2str(Rekker),' rekker.'])
    return
elseif m<0 | n<0
    disp('Rekkenumrene må være positive.')
    return
end

% Kopierer matrisa
M=A;

% Legger k ganger rekke m til rekke n
M(n,:)=M(n,:)+k*A(m,:);
```

Jeg har forsøkt å gjøre disse rutinene mer eller mindre idiotiskre i den forstand at de kontrollerer at rekkenumrene er positive og mindre enn rekketallet til matrisa. Videre kontrollerer vi at $k \neq 0$ i rutina for rekke-multiplikasjon.

Selve rekkereduksjonen til redusert trappeform blir utført av denne rutina (les kommentarene for mer informasjon):

```
function R=RedTrappeform(A);

% Funksjon som returnerer den reduserte trappeforma av input-matrisa.

% Finner formatet på A
[m n]=size(A);

% for-løkkene skal gå til og med det som er minst av rekke- og søylenr.
MaxInd=min(n,m);

% Kopierer matrisa:
R=A;

% Initierer matrise med søylenr. til pivot-søylene
PivotPos=[]; PivotIndex=1;

% Framover-fase - søyle for sølye
for s=1:MaxInd;
  SubMatr=R(s:end,s:end);              % Lager under-matrise
  [SubMatr Piv]=FramFaseStep(SubMatr);% Rekkereduserer søyle 1 i undermatr.
  if Piv==1                            % Dersom søyle s er ei pivot-søyle
    PivotPos(PivotIndex)=s;            % Identifiserer pivot-søyle
    PivotIndex=PivotIndex+1;
  end
  R(s:end,s:end)=SubMatr;              % Kopierer undermatr. tilbake til R
end

% Bestemmer antall ledende tall (rangen til matrisa)
Rank=length(PivotPos);

% Bakover-fase
for r=Rank:-1:2;                     % for hver pivot-rekke
  s=PivotPos(r);                     % Finner den aktuelle pivot-søyla
  R=BakFaseStep(R,r,s);              % Gjør alle tall over ledende til 0
end

% Gjør alle ledende tall til 1
for r=Rank:-1:1;
  s=PivotPos(r);                     % Finner den aktuelle pivot-søyla
  k=1/R(r,s);
  R=Mult(R,k,r);                     % Gjør det ledende tallet til 1
end
```

I et forsøk på å gjøre koden noe mer oversiktlig, har jeg lagt to funksjonsfiler som gjør deler av rekkeredusjonen: `FramFaseStep` og `BakFaseStep`. Den første undersøker først om alle tallene i søyle 1 i den aktuelle matrisa er lik null. Hvis ikke, sørger den for å få et tall ulik null i rekke 1, og så gjør den alle de andre

tallene i søyla til null ved hjelp av rekkeoperasjoner. Funksjonsfila ser slik ut:

```
1  function [M Piv]=FramFaseStep(A)
2
3  % Funksjon som rekkereduserer matrisa slik at alle tall - bortsett fra
4  % det første i søyle 1 er lik null.
5  % Dersom alle tallene i søyle 1 i A er null fra før, forblir matrisa
6  % uendra.
7  % Ut-variabelen Piv er 1 dersom søyla er ei pivot-søyle og 0 hvis ikke.
8
9  % Undersøker om alle elementene i søyle 1 er null
10 if sum(abs(A(1,:)))<2*eps
11     M=A;
12     Piv=0;
13     return
14 end
15
16 % Dersom ikke alle tallene er 0: Piv er 1
17 Piv=1;
18
19 % Finner formatet på A
20 [m n]=size(A);
21
22 % I det videre kan vi ta for gitt at A har minst ett tall ulik null i
23 % søyle 1.
24 % Vi sørger for at talle øverst til venstre er ulik null.
25 M=A;                          % Lager kopi av A
26 RekkeInd=1;
27 while M(1,1)==0
28     RekkeInd=RekkeInd+1;
29     M=Swap(M,1,RekkeInd);       % Bytter rekke dersom M(1,1)=0
30 end
31
32 % Sørger for at alle tall utenom det første i søyle 1 blir null.
33 for i=2:m
34     k=-M(i,1)/M(1,1);
35     M=AddMult(M,k,1,i);
36 end
```

Denne funksjonsfila identifiserer også søyler med ledende tall – se linje 12 og 17.

Etter linje 26 i hovedfila, `RedTrappeform`, har vi fått redusert matrisa til trappeform; alle tall under de ledende tallene er nå null. Men vi skal videre til redusert trappeform, vi må "fjerne" alle tall *over* de ledende tallene også. Dette blir gjort i linje 32-35. I `for`-løkka blir funksjonen `BakFaseStep` kalt. Den ser slik ut:

```
1  function M=BakFaseStep(A,r,s)
2
3  % Funksjon som rekkereduserer matrisa slik at alle tall bortsett fra den
4  % ledende eneren i rekke r, søyle s blir null.
5  % Det forutsettes at A(r,s) ikke er null;
6
7  % Sjekker at A(r,s) ikke er null
8  if abs(A(r,s))<2*eps
9      disp('A(r,s) er null.')
10     return
11 end
12
13 %Kopierer matrisa:
14 M=A;
15
16 % Fjerne alle tall over det ledende tallet
17 for rr=(r-1):-1:1;
18     k=-M(rr,s)/M(r,s);
19     M=AddMult(M,k,r,rr);
20 end
```

Det siste vi gjør, i linje 38-42 i `RedTrappeform`, er å sørge for at alle ledende tall er 1.

La oss sjekke om den fungerer:

```
>> A=[1 2 3; 3 -2 -1]
A =
     1     2     3
     3    -2    -1
>> RedTrappeform(A)
ans =
   1.000000000000000                   0   0.500000000000000
                   0   1.000000000000000   1.250000000000000
>> A=[0 9 0 1; 1 2 3 4; 78 9 0 2; 9 8 7 0]
A =
     0     9     0     1
     1     2     3     4
    78     9     0     2
     9     8     7     0
>> RedTrappeform(A)
ans =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
>> A=[0 0 0]
```

```
A =
     0     0     0
>> RedTrappeform(A)
ans =
     0     0     0
```

Dette ser jo vel og bra ut; alle matrisene vi får ut er på redusert trappeform. Men er det riktig? La oss sammenligne svarene som `RedTrappeform` gir med MATLABs egen rutine, `rref`:

```
>> A=rand(2,2);
>> RedTrappeform(A)-rref(A)
ans =
     0     0
     0     0
>> A=rand(4,3);
>> RedTrappeform(A)-rref(A)
ans =
   1.0e-15 *
                  0                   0                   0
   0.109791924442720                   0                   0
   0.235321355608084                   0  -0.111022302462516
                  0                   0                   0
>> A=rand(4,3);
>> RedTrappeform(A)-rref(A)
ans =
   1.0e-15 *
                  0                   0                   0
  -0.276742902922741  -0.111022302462516                   0
                  0                   0                   0
                  0                   0   0.222044604925031
>> A=rand(5,4);
>> RedTrappeform(A)-rref(A)
ans =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

Det ser ut til å fungere.