Vektorer - 1 av 8

I MatLab finns det en mängd funktioner som har med vektorer att göra.

Allt ni redan gjort är egentligen vektorhantering. Även ett heltal är en vektor med längden 1.

Skapa en vektor:

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

v = [ 1 ; 2 ; 3 ; 4 ; 5 ];
v = [ 1 ;
      2 ;
      3 ;
      4 ;
      5 ];
v = [ 1
      2
      3
      4
      5 ];

v = 5;

v = 1:5;
v = 1:2:5;
v = 5:-1:1;

v = zeros(5, 1);
v = ones(1, 5);
```

Vektorer - 2 av 8

Skriva ut en vektor:

```
disp(v)

disp([ 1 2 3 ])

svar = 5;
disp([ 'Summan är ' num2str(svar) ])
```

Transponera (rotera 90 grader):

```
v = v';
```

Ta reda på information om vektorn:

```
Antag:  v = 1:3:9;  => [ 1 4 7 ]

l = length(v);  => 3
s = size(v);    => [ 1 3 ]
```

Indexera en vektor:

```
v(1)    => Värdet i index 1 i vektorn v

v(2:3)  => En vektor med de värden som
           finns i index 2-3 i v

Antag:  v = [ 3 1 2 ];

v(v)    => [ 2 3 1 ]
```

Vektorer - 3 av 8

For-satsen:

```
Antag:  v = 1:2:20

for i = 1:5
  disp(i)
end

for i = v
  disp(i)
end

for i = 1:length(v)
  disp(i)
  disp(v(i))
end

for i = [ 1 3 5 2 4 6 ]
  disp(v(i))
  i = i + 1;
  disp(i);
end   % => 1 2 5 4 9 6 3 3 7 5 11 7
      %    (på egna rader dock)
% Ändring av 'i' temporärt, men påverkar
% alltså ej for-satsens uppräkning.

for i = 1:inf
  disp(i)
end % => 1 2 3 ... 1.7977e+308 Inf Inf ...
    %   (realmax)
```

```
OBS!  realmax + power(10, 291) => realmax
      % Fortfarande realmax (avrundningsfel!!!)
```

Vektorer - 4 av 8

Funktionsreturvärden:

```
function v = create_vector(number)
  v = 1:2:number;
end

function [ a b ] = factorial_and_power(n)
  a = factorial(n);
  b = power(n, n);
end

function [ x y ] = swap(a, b)
  x = b;
  y = a;
end

% Alternativt bara:
function [ b a ] = swap(a, b)
end
```

Ovanstående kan anropas från ett huvudprogram ...

```
function main
  v = create_vector(4);
  [ f p ] = factorial_and_power(4);
  [ a b ] = swap(a, b);
end
```

Speciellt i MatLab: Inbyggda funktioner klara ofta hela vektorer även om det inte skall vara så enligt matematiken.

```
v = sin(v);  => ny vektor med sinusvärden
```

Vektorer - 5 av 8

Operationer som bearbetar hela vektorer:

```
Antag:   v1 = [ 1 2 3 ]

v2 = v1 * 2;    => [ 2 4 6 ]

v2 = v1 * v1;   => FEL!

v2 = v1 * v1';  => 14

v2 = v1 .* 2;   => [ 2 4 6 ]

v2 = v1 .* v1;  => [ 1 4 9 ]

v2 = v1' * v1;   => Matris!
                        [ 1 2 3 ;
                          2 4 6 ;
                          3 6 9 ]
```

Fundera på om det bör fungera för alla räknesätten ...

```
v2 = v1 ./ v1;   => [ 1 1 1 ]

v2 = v1 / v1;    => 1.0000 (flyttal???)

v2 = v1 / 2; => [ 0.5000 1.0000 1.5000 ]
```

OBS! .+ och .- finns ej!!!!

Vektorer - 6 av 8

Plottning av vektorer:

```
Antag:  x = 1:360

plot(x) => Rät linje: 1,1  2,2 ... 360,360

y = sind(x);
plot(y)  => Sinuskurva med x-koordinater
            1-360 (index i vektorn)
plot(x, y)  => Samma som ovan (plottar y-
               vektorn mot x-vektorn)
plot(x * 2, y)  => x-koordinaterna
                   "dubblade"
```

När man plottar vektorer binds punkterna automatiskt ihop. Nu gäller fler av argumenten till plot (d.v.s. dem som berör hur man binder ihop punkter m.m.)

```
% Endast x i punkterna (ej ihopbundet)
plot(x, y, 'x');

% x i punkterna, ihopbundet, streck-linje
plot(x, y, ':xr');

% Linjebredd 4 och som ovan ...
plot(x, y, ':xr', 'linewidth', 4);

% Linjebredd 4, heldragen röd linje,
% gröna *:or samt "dottad" linje
plot(x, y, '-xr', x, y, ':*g', ...
     'linewidth', 4);
```

Vektorer - 7 av 8

Uppgift: Skriv en funktion som ritar ut en kvadrat med storleken "n". Indata till funktionen skall vara storleken.

Lösning:

```
function square(n)
  figure;
  plot(-n, -n, 2*n, 2*n);
  hold on;
  plot([0 n n 0 0], [0 0 n n 0]);
  hold off;
end
```

Vektorer - 8 av 8

Fråga: Varför är nedanstående lösning ej så bra?

```
for i = 1:5
  hold on;
  plot( ... );
  hold off;
end
```

Svar: Kanske effektivare med "hold on" och "hold off" utanför for-satsen ...

Undantag: Om man använder "clf" rensas även "hold on". Detta ger att man måste göra hold on inuti for-satsen ...

Matriser - 1 av 6

Inbyggda funktioner klara ofta hela matriser även om det inte skall vara så enligt matematiken. Exempel på detta kommer på en senarte OH.

Vi börjar med lite enkla sätt att skapa matriser:

```
m = [ 1 2 3 ; 4 5 6 ];
m = [ 1, 2, 3 ;  4, 5, 6 ];

m = [ 1, 2 ; 3, 4 ; 5, 6 ];
m = [ 1 2 ; ...
      3 4
      5 6 ];

m = 1:5 , 6:10
m = 1:5 ; 6:10
   => m   = [ 1 2 3 4 5 ] samt
      ans = [ 6 7 8 9 10 ]
i första fallet skrivs både m och ans ut
i andra fallet skrivs endast ans ut

m = 1:2:5
m = 5:-1:1

m = zeros(5);  => 5x5-matris med nollor
m = ones(5);

m = zeros(5, 5, 5);
   => 5x5x5-matris med nollor
```

Matriser - 2 av 6

Skriva ut:

```
disp(m)
```

Transponera (spegla i diagonalen):

```
m = m';
```

Rotera (motsols):

```
m = rot90(m);     => Roterar 90 grader
m = rot90(m, n); => Roterar n*90 grader
```

Omforma:

```
reshape(v, rader, kolumner)

reshape(1:5, 5, 1)

ger
  1
  2
  3
  4
  5

reshape(1:5, 5, 1)'   (transponerar)

ger
   1 2 3 4 5
```

Matriser - 3 av 6

Diagonalmatriser:

```
diag(v)

diag([ 1 2 3 ])
=> matris med vektorn i diagonalen
  1 0 0
  0 2 0
  0 0 3


diag(m)

diag([ 1 2 3 ; 4 5 6 ; 7 8 9 ])
=> diagonalvektor
  1 5 9
```

Determinanter:

```
det(m)  => Determinanten av/för m
```

Matriser - 4 av 6

Indexering:

```
indexering av index r (rad), k (kolumn) i
matrisen m
m(r, k)

indexering i en 3-dimensionell matris
m(x, y, z)

v = m(1, :)    => Rad 1 i matrisen
v = m(1:1, :)  => Rad 1 i matrisen

n = m([1 3:size(m,1)], :);
=> Tar bort rad nummer 2 ur m

m1 = m(1:0, 1:4);  => Tom matris 0x4
m2 = m(1:0, 1:5);  => Tom matris 0x5

if m1 == m2
=> Felmeddelande att det är olika
   dimensioner :-)
```

Ta reda på information om matrisen:

```
Antag:
  m = reshape([1:20], 4, 5);

l = length(m);  => 5
s = size(m);    => [ 4, 5 ]
n = numel(m);   => 20
```

Matriser - 5 av 6

Funktionsreturvärden:

```
function m = create_matrix(number)
  m = round(rand(number) * 10);
end

function [ m1 m2 ] = create_matrices(m)
  m1 = factorial(m);
  m2 = power(m, 2);
end

function main
  m = create_matrix(4);
  [ a b ] = create_matrices(m);
end
```

Operationer som bearbetar hela vektorer:

```
Antag:
  m1 = [ 1 2 ; 3 4 ; 5 6 ]

m2 = m1 * 2   => [ 2 4 ; 6 8 ; 10 12 ]
m2 = m1 * m1  => FEL!
m2 = m1 * m1' => [  5 11 17 ;
                   11 25 39 ;
                   17 39 61 ]

m2 = m1 .* 2  => [ 2 4 ; 6 8 ; 10 12 ]
m2 = m1 .* m1 => [ 1 4 ; 9 16 ; 25 36 ]
m2 = m1' * m1 => [ 35 44 ; 44 56 ]
```

Matriser - 6 av 6

Speciella saker:

```
m = sin(m);
=> ny matris med sinusvärden för
   respektive värde i ursprungsmatrisen.

Inbyggda funktioner klara ofta hela
matriser även om det inte skall
vara så enligt matematiken.
```

Plottning av matriser:

```
plot3(x, y, z, ...);

imagesc(m);
=> Flera färger om man har många olika
   värden. Två färger om det endast är två
   olika värden.
```