TANA81/82 Föreläsning 4

- Logik och Jämförelser.
- Styrsatser: Villkorssatsen `if` och repetitonssatsen `for`.
- Repetitonssatsen `while`. Avbrott med `break`.
- Exempel: En Talföljd och en enkel simulering.
- Egna funktioner. Skalärprodukt. Funktioner som inargument.

Logiska variabler och jämförelser

En *Logisk variabel* kan ha värdet *sant* eller *falskt*. Logiska värden skapas vid jämförelser mellan två tal.

Exempel Låt `x=3.0` och `y=6.5`.

```
>> x > y
 ans =
     0
```

Kommentar I MATLAB tolkas värdet 0 som falskt och 1 som sant.

I MATLAB finns
Relationsoperatorer <, <=, >, >=, ==, och ~=
Logiskaoperatorer `&`, `|` och ~

Exempel Låt $x = 3.0$ och $y = 6.5$. Beräkna uttrycket

```
>> z = x + y >= x^2 | x == y
```

Prioritet Aritmetriska > Relations > Logiska. Kan använda parenteser.

Jämförelser med Matriser eller Vektorer

Jämförs två vektorer eller matriser så beräknas jämförelsen elementvis. Bägge operanderna måste ha samma dimension.

Exempel Låt `x=[1 3]` och `y=[2 2]`. Då gäller att

```
>> x < y
 ans =
   1   0
```

Även logiska operatorer beräknas elementvis.

Funktionen `randi` returnerar slumpmässiga heltal i ett givet intervall. Exempelvis `A=randi([a b],n,m)` skapar en $n \times m$ matris med slumpmässiga heltal i intervallet $[a, b]$.

Exempel Hur många femmor får man i genomsnitt om man slår en tärning 8 gånger? Gör $N = 10^4$ experiment och undersök detta.

Exempel Antag att `x` är en vektor som innehåller både positiva och negativa tal. Beräkna summan av de positiva talen.

I Matlab skriver vi

```
>> S = sum( (x>0) .* x );
```

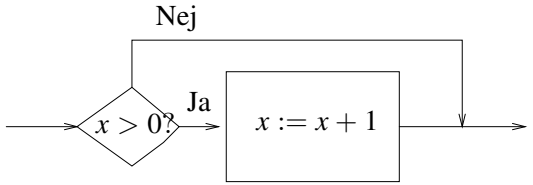
Vad händer om $x = (2\,,\ -3\,,\ 0\,,\ 1\,,\ 2\,,\ -1)^T$?

Villkorssatser

Definition En *styrsats* används för att bestämma vilka delar av ett program som skall exekveras.

Definition En *villkorssats* används då ett antal kommandon endast skall exekveras om ett visst villkor har värdet *sant*.

Exempel Värdet på x avgör vilken väg som tas genom programmet.



I Matlab finns

```
if  <logiskt uttryck>
     <satser>
end
```

där kommandona `<satser>` endast utförs om det *logiska uttrycket* beräknas till värdet *sant*.

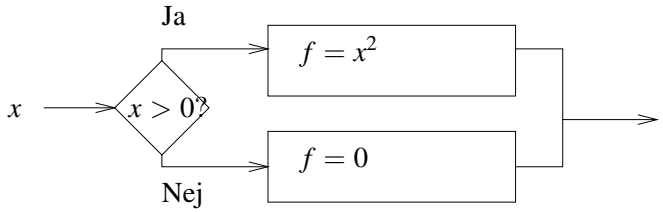
Exempel Antag att vi har beräknat ett funktionsvärde $f(x)$. Gör en utskrift om värdet är positivt.

Ett alternativ är

```
if  <logiskt uttryck>
     <satser 1>
else
     <satser 2>
end
```

där antigen `<satser 1>` eller `<satser 1>` utförs.

Exempel Låt $f(x)$ ges av två olika uttryck beroende på om $x > 0$.



Exempel En funktion ges av uttrycket

$$f(x) = \begin{cases} 0, & x \leq 0, \\ x^2, & x > 0. \end{cases}$$

I Matlab kan vi skriva:

```
x=3.5;
if (x <= 0)
   f=0;
else
   f=x^2;
end;
```


Exempel Antag att $a = 5$ och att variabeln `flagga` har värdet falskt, dvs `flagga=0`. Vad händer i följande fall?

```
if flagga
   if  a<10
      a=a+1
   else
      a=a-1
   end
end
```

```
if flagga
     if a<10
        a=a+1
     end
 else
     a=a-1
 end
```

Vad skulle ha hänt om istället $a = 5$ och variabeln `flagga` hade haft värdet sant?

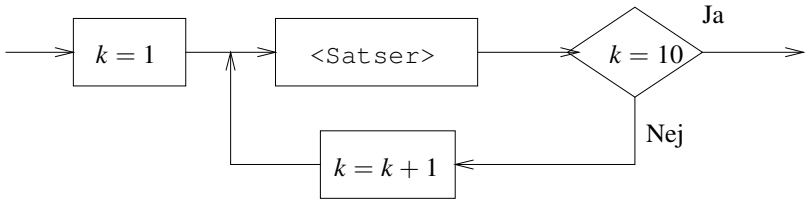
Exempel Testa om ett värde x ligger i intervallet $2 \leq x < 5$. Skriv isåfall ut ett meddelande.

Tänk på att jämförelser görs mellan *två* tal.

Repetitonssatser

Definition En *repetitionssats* används då ett antal satser skall upprepas flera gånger.

Då ett antal satser skall utföras, ett på förhand känt, antal gånger används en `for`-sats.



I Matlab finns konstruktionen

```
for <variabel>= start:steg:slut
        <satser>
end
```

där `<satser>` utförs en gång för varje värde på `<variabel>`.

Exempel Beräkna summan $S = \sum_{k=1}^{1000} \frac{1}{k^2}$. I Matlab skriver vi

```
S=0;
for k=1:1000
   S=S+k^(-2);
end
```

Detta ger `S=1.6439`.

Exempel Vad blir `x(5)` då följande program exekveras?

```
x=zeros(5,1);
for k=2:1:5
  x(k)=x(k-1)+k
end
```

Exempel Låt x vara vektorn

$$x = (1, -1, 5, 2, 3, -2, 4)^T.$$

Använd en repetitionssats för att hitta det största elementet i x.

Nästlade repetitionssatser

Det går bra att ha flera `for`-satser innuti varandra.

Exempel Hilbert matrisen är ett exempel på en matris som är svår att arbeta med. Den definieras av att elementen sätts till

$$h_{ij} = \frac{1}{i+j}.$$

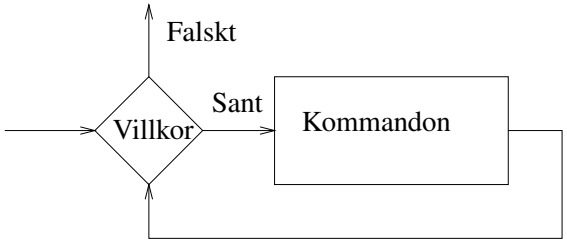
Skriv ett Matlab program som skapar en Hilbert matris av dimension $n = 10$.

Repetitionssatser

Kommandot `while` exekverar en satsgrupp så länge som ett logiskt villkor är *sant*. Den generella formen är:

```
while <logiskt villkor>
   <satsgrupp>
end
```

Det är viktigt att en `while`–sats kan avbrytas.



Exempel Fibonacci talen ges av $F_1 = 0$, $F_2 = 1$, och $F_n = F_{n-1} + F_{n-2}$. Vilket är det största talet F_k som forfarande är mindre än 100?

I Matlab skriver vi:

```
F(1)=0;F(2)=1;n=2;
while ( F(n)< 100 )
  n=n+1;
  F(n)=F(n-1)+F(n-2);
end
n-1,F(n-1)
```

Repetitionssatsen avbryts när $n = 13$ så det största talet blir $F_{12} = 89$.

Att avbryta en repetitionssats

Kommandot `break` avbryter en `while` eller `for` sats direkt.

Exempel I ett spel försöker vi slå en tärning högst 10 gånger i följd. Målet är att få samma resultat två gånger i rad.

Skriv ett Matlab program som simulerar en sådan spelomgång. För varje lyckad omgång skriver vi ut antalet kast som krävdes.

Exempel En talföld $\{x_k\}_{k=0}^{\infty}$ genereras ifrån ett positivt heltal x_0 enligt följande regler:

1. Om x_k *udda* så fås nästa tal som $x_{k+1} = 3x_k + 1$.
2. Om x_k är jämnt fås $x_{k+1} = x_k/2$.

Om vi startar med talet 6 får vi talföljden:

$$6 \quad 3 \quad 10 \quad 5 \quad 16 \quad 8 \quad 4 \quad 2 \quad 1.$$

Övning Vi antar att att alla sådana talföljder slutar med 1. Skriv ett program som verifierar detta för startvärden mellan 1 and 100.

Hitta dessutom det startvärde som ger den längsta talföljden.

Funktioner i MATLAB

En *funktion* har ett antal *inagrument* och beräknar ett antal *utargument*.

Exempel Funktionen `zeros` har som inparameter två heltal N och M. Utparameter är en matris av dimension $N \times M$ med nollor.

```
>> Z = zeros( N , M );
```


Funktioner skapas genom att man samlar kommandon på en fil, exemelvis `min_funktion.m`.

Dessutom skall man skriva ett *funktionshuvud*.

```
% Inledande kommentar
%
function [ut1,ut2]=min_funktion( in1,in2,in3 )

   <beräkna ut1,ut2 givet in1,in2,in3>
end
```

En funktion kan ha godtyckligt antal in-, respektive ut-parametrar. Den inledande kommentaren skrivs ut om man skriver

```
>> help min_funktion
```


Exempel På filen `serie.m` har vi skrivit

```
function [S]=serie(N)
   S=0;
   for k=1:N,S=S+1/k^2;,end
end
```

Skriver vi följande kommandon

```
>> S=serie( 100 );
>> disp(S)
     1.6350
>> disp(k)
     Undefined function or variable 'k'.
```

Variabler är *lokala*. Skapas, eller ändras, en variabel i en funktion är det en lokal kopia som ändras. Då funktionen avslutas är det endast utparametrar som sparas.

Exempel Skalärprodukten mellan två vektorer x och y kan beräknas med formeln

$$x \cdot y = \sum_{i=1}^{n} x_i y_i.$$

Skriv en funktion som beräknar skalärprodukten. Funktionen skall användas enligt

```
>> S = ScalarProd( x , y );
```

På filen `ScalarProd.m` skriver vi

```
% ScalarProd: Beräkna skalärprodukt mellan två vektore
% y. Anropas enligt:
%
%   >> S = ScalarProd( x , y );
%
function [S]=ScalarProd( x , y )
  n=length(x);
  S=0;
  for i=1:n
       S=S+x(i)*y(i);
  end;
```

Kommentar Då $x \cdot y = x^T y$ kan vi skriva funktionen enklare. Det finns även en fördefinierad funktion `dot()`.

Rekursiva funktioner

En funktion som anropar sig själv kallas *rekursiv*. Vid varje funktionsanrop skapas nya lokala kopior av variabler.

Exempel Fakulteten är definerad genom att

$$n! = n \cdot (n-1)!, \qquad 0! = 1.$$

Skriv en MATLAB funktion som beräknar $n!$ för ett givet heltal n.

På filen `Fakultet.m` skriver vi

```
function [F]=Fakultet(N)

    if N==0,
      F=1;
    else
      F=N*Fakultet(N-1);
    end;
end
```

Vad händer om vi skriver

```
>> Fakultet( 4 );
```


Funktioner som inargument

Kommandot `feval` kan användas för att anropa en funktion med givna inparametrar.

Exempel Har vi skrivit en fil `funk.m` som innehåller

```
function [z]=funk( x , y )
   z=x^2+3*y;
end
```

så kan vi anropa funktionen med inargument $x = 2$ och $y = 2.5$ genom att skriva

```
>> feval( @funk , 2 , 2.5 )
 ans =
  11.5000
```

Funktioner som inargument

Exempel Definitionen av derivata är

$$f'(x) = \lim_{h\to 0} \frac{f(x+h)-f(x)}{h}.$$

Välj ett fixt $h > 0$ för att approximera derivatan av $f(x)$.

Skriv en funktion `Derivata.m` som kan anropas med

```
>> Df = Derivata( @funk , x , h );
```

Använd funktionen för att plotta derivatan av $\sin(x)$ på intervallet $[0, 2\pi]$.

Lösning På filen `Derivata.m` skriver vi

```
function [Df]=Derivata( f , x , h )
     f1=feval( f , x + h );
     f0=feval( f , x );
     Df=(f1-f0)/h;
end
```

Vi kan sedan plotta derivatan av $\sin(x)$ genom att skriva

```
>> x = 0:0.01:2*pi;
>> plot( x , Derivata( @sin , x , 10^-3 ) );
```


Enkla funktioner

Enkla funktioner kan skapas direkt i Matlab. Man skriver

```
>> f = @( <variabler> )  <uttryck>;
```

Man får då ett handtag till en funktion.

Exempel Skriv

```
>> f = @(x) exp(2*x).*cos(x.^2)+4*x
  f =
    @(x)exp(2*x).*cos(x.^2)+4*x
```

Funktionen kan beräknas med `feval` eller direkt `f(4)`.

Exempel Skapa funktionerna

$$f(x) = \sqrt{1-x^2}e^{-2x}, \quad \text{och} \quad g(x) = (x_1^2 - 1\,,\, \sqrt{1+x_2})^T.$$

Tänk på att $g(x)$ måste ha en vektor som inagrument och returnera en vektor i $\mathbb{R}^2$.

Försök få $f(x)$ att acceptera vektor argument så att det går att skriva

```
>> x = 0:0.01:1;
>> plot( x , f(x) )
```

Funktionen `integral` beräknar integraler. Den anropas

```
>> I = integral( fun , a , b )
```

där `f` är ett funktionshandtag.

Exempel Beräkna samma integral som tidigare genom

```
>> f = @(x) exp(2*x).*cos(x.^2)+4*x;
>> I = integral( f , 0 , 1)
  I =
   4.6736
```

Det finns även `integral2` för dubbelintegraler.

Funktionen `fzero` hittar en rot till ekvationen $f(x) = 0$. Den anropas

```
>> x = fzero( fun , x0 )
```

där `f` är ett funktionshandtag och x_0 är en start gissning.

Exempel Lös ekvationen $f(x) = \mathrm{e}^{-2x} - x = 0$.

Skapa en funktion och använd `fzero` för att hitta roten

```
>> f = @(x) exp(-2*x)-x;
>> x = fzero( f , 1 )
 x =
  0.4263
```

Kommentar Funktionen `fzero` använder en avancerad numerisk metod. Det finns färdiga metoder för att lösa en stor mängd problem.