MAL600, MATLABprogrammering 1

Enkla slingor och m-filer

Exempel 1. Beräkna $\sum_1^{10} k$.

Lösning 1. Med hjälp av en for-slinga:

```
>>s=0;for k=1:10,s=s+k;end,s
```

Lösning 2. Med hjälp av vektoroperationer:

```
>>k=1:10;sum(k)
```

Exempel 2. Kan $\sum_{k=1}^{n} k = 257$ för något n?

Lösning 1. Med hjälp av en while-slinga:

```
k=1;s=0;while s<257,s=s+k;k=k+1;end,s
```

Lösning 2. Med vektoroperationer:

```
>>k=1:100;K=cumsum(k);I=(K==257);s=sum(I)
```

Lösning 3. Variant där vi får svaret Ja respektive Nej:

```
>>k=1:100;K=cumsum(k);
I=(K==257);s=sum(I);
if s=0
svar='Nej'
else
svar='Ja'
end
```

Exempel 3. Skriv ett betygsättarprogram.

Lösning. Programmet skall till en poängsiffra p ge betyget U om $p < 12$, G om $12 \leq p < 18$ och VG om $p \geq 18$.

```
>>if 0<=p&p<12
betyg='U'
elseif 12<=p&p<18
betyg='G'
elseif 18<=p&p<=25
betyg='VG'
else
'Va,fuskar du?'
end
```

Studera §12.1-4 i MATLABboken för mer om programmering i MATLAB.

I stället för att utföra MATLABkommandon i kommandofönstret kan man skapa (med hjälp av någon editor) så kallade m-filer. Information om detta hittar du i §2.8 i MATLAB-boken.

Lite om m-filer och editorer.
Gör om Exempel 3 som en m-fil.
Funktionsfiler.

Exempel 4. Simulering av tärningskast.

Lösning 1. Med en kommandofil:
Filnamn: tarn1.m
Innnehåll:

```
r=6*rand
if r<1
  t=1
  elseif r<2
    t=2
    elseif r<3
      t=3
      elseif r<4
        t=4
        elseif r<5
          t=5
          else
            t=6
end
```

Lösning 2. Som en funktionsfil:
Filnamn: tarn2.m
Innehåll:

```
function t=tarn2
r=6*rand;
if r<1
  t=1;
  elseif r<2
    t=2;
    elseif r<3
      t=3;
      elseif r<4
        t=4;
        elseif r<5
          t=5;
          else
            t=6;
end
```

Lösning 3. Lite listigare kod:
Filnamn: tarn3.m
Innehåll:

```
function t=tarn3
r=6*rand;
t=ceil(r);
```

Lösning 4. : Här kan vi göra flera kast:
Filnamn: tarn4.m
Innehåll:

```
function t=tarn4(n)
if nargin==0
  n=1;
    end
    for k=1:n
r=6*rand;
t(k)=ceil(r);
end
```

Vi skall nu använda tarn-filerna för att undersöka stopptiden τ som definieras på följande sätt. Låt $T_1, T_2, T_3, \ldots$ vara en följd av tärningskast. Då

definierar vi τ som det antal kast som behövs för att summan av tärningskasten för första gången blir minst n, dvs. $T_1 + T_2 + T_3 + \ldots + T_{\tau-1} < n$ och $T_1 + T_2 + T_3 + \ldots + T_\tau \geq n$.

Lösning 5.
Filnamn: tarn5.m
Innehåll:

```
function k=tarn5(n)
s=0;k=0;
while s<n
  s=s+tarn4;
  k=k+1;
  end
```

För att undersöka $E[\tau]$ gör vi försöket flera (k) gånger och beräknar medelvärdet.

Lösning 6.
Filnamn: tarn6.m
Innehåll:

```
function m=tarn6(n,k)
s=0;
for i=1:k
  s=s+tarn5(n);
  m=s/k;
  end
```

Uppvärmningsövningar

Man kan skapa egna kommandon och funktioner i MATLAB. Dessa ska ha namn som slutar med **.m** och skapas som filer i en editor. MATLAB har en egen editor men Du kan också använda någon Du redan behärskar exempelvis EMACS, som Du har tillgång till på Din arbetsstation. Innan Du gör denna uppgift bör Du läsa avsnitt 2.8 i boken ganska noggrant.

Upppgiften i övning 1 får ses som ett pedagogiskt enkelt exempel då det knappast är relevant att skapa m-filer för så elementära beräkningar.

1. Skriv en kommandofil med namnet **addera.m** som helt enkelt består av MATLAB-beräkningen $c = a + b$. Spara filen och testa att exekvera den från MATLAB. Variablerna **a** och **b** måste ha fått värden innan förstås.

Lägg nu till kommentaren "Jag adderar a plus b i variabeln c" **först** i filen addera.m. (Kommentarer skrivs med procent-tecken i första position.) Vad händer om Du nu skriver **help addera** i MATLAB. Just det, Du har skapat en egen help-information.

Gör om **addera.m** till en funktionsfil i stället, med parametrarna **a** och **b**, och anropa den från MATLAB som en funktion.

Ge nu kommandot **type addera** och konstatera att din funktion betraktas som MATLAB's egna, detta under förutsättning att den ligger i aktuellt MATLAB-bibliotek dvs aktuellt directory där Du startade MATLAB från. Man kan kolla vilka egna m-filer man har i biblioteket genom kommandot **what** i MATLAB. (Med hjälp av kommandot **matlabpath** kan man komplettera de bibliotek som man vill att MATLAB ska söka i)

2. Skriv en for-sats för beräkning av summan av de 20 första termerna i serien 1, 1/2, 1/4, 1/8, 1/16, ...

3. Skriv en while-sats för beräkning av en delsumma av serien i uppgift 2, där summeringen avbryts när termerna är mindre än en miljondel.

4. Harmoniska serien är 1, 1/2, 1/3, 1/4, ... Skriv en funktion **harmonisk.m** med parameter **n**, som beräknar partialsumman av **n** termer (1+1/2+1/3+1/4+...+1/n) genom att på lämpligt sätt använda en for-sats.

Harmoniska serien är inte konvergent dvs summan blir godtyckligt stor bara man tar med tillräckligt många termer. Vi vill veta hur många termer som behövs för att summan ska överstiga värdet **tak**. Skriv en ny funktion, som har **tak** som parameter och som returnerar antalet termer. Använd lämpligen en while-sats. Hur många termer behövs då tak=12?

Förslag till lösningar

1.

```
%jag adderar a plus b i varabeln c
c=a+b
```

respektive

```
function c=addera(a,b)
%jag adderar b plus c
c=a+b
```

som sedan kan användas i MATLAB genom att man t.ex. skriver `>>sum=addera(3,5)`.

2.

```
s=0;
 for i=1:20
s=s+1/2^(i-1);
 end,
 s
```

3.

```
s=1;term=1;
while term>=1e-6
term=term/2;
s=s+term;
end
s
```

4.

```
function s=harmonisk(n)
s=0;
for i=1:n
   s=s+1/i
end
```

respektive

```
function n=harmonisk2(tak)
s=0;
i=1;
while s<=tak
s=s+1/i;
i=i+1;
end
n=i-1;
```

tak=12 kräver 91380 termer.