F3: Funktioner (kap. 5)

Funktionsfil, funktionsanrop
in- och utparametrar, anropsin- och anropsutparametrar
lokala, globala och persistenta variabler
lokala funktioner
return
variabelt antal parameterar, skönsvärde
slumptal
avlusning, felsökning (debugging), programprofil

Funktioner forts.

En funktion är en följd kommandon i en m-fil som inleds med ett funktionshuvud:
`function [`*utparameter1*`, …] =` *funktionsnamn*`(`*inparameter1*`, …)`
eller
`function` *funktionsnamn*`(`*inparameter1*`, …)`

Direkt efter funktionshuvudet skriver man dokumentationskommentarer enligt reglerna för `lookfor` och `help`. Därefter står kommandon som utförs när funktionen anropas. Funktionen anropas med:
funktionsnamn`(`*anropsinparameter1*`, …)`
eller
`[`*anropsutparameter1*`, …] =` *funktionsnamn*`(`*anropsinparameter1*`, …)`

När kommandona i funktionen har utförts till slut fortsätter exekveringen efter anropet. En funktion bör lagras i en fil med namnet *funktionsnamn*.m.

Parametrar

Parametrarna och övriga variabler i en funktion är lokala dvs. har inget med variabler med samma namn i kommandofönstret eller andra kommando- eller funktionsfiler att göra. Undantag gäller *globala* och *persistenta* variabler. Funktioner kan skrivas så att de, olika gånger, kan anropas med olika antal in- och utparametrar. En m-fil kan innehålla mer än en funktion. Funktionerna efter den första kallas *lokala funktioner* och de kan anropas bara i samma m-fil.

Vi gör om kommandofilen för kuggväxel till en funktion:

```
function [m n finalgear] = findgear(gear,min,max)
% Hittar det kugghjulspar som bäst approximerar en
given utväxling.
% [m n finalgear] = findgear(gear,min,max)
% hittar den bästa approximationen m/n till
% gear för m och n mellan min och max för
% positiva min, max och gear.
```


findgear.m forts

```
if gear>0 & max>0 & min>0
  bestdiff = realmax;
  for m = min:max
    nprel = floor(m/gear);
    for n = nprel:nprel+1
      if n<min n=min; end; if n>max n=max; end;
      diff = abs(m/n-gear);
      if diff<bestdiff
         bestdiff = diff; bestm = m; bestn = n;
      end % if
    end % for n
  end % for m
  m = bestm; n = bestn; finalgear = m/n;
else error('Felaktiga indata.');
end
```

Kuggväxel forts.

Testkörning ger:

```
findgear(3.14159,5,50)
ans = 22
[m n g] = findgear(3.14159,5,50)
m = 22
n = 7
g = 3.1429
findgear(3.14159,5,-50)
??? Error using ==> findgear
Felaktiga indata.
```


Parametrar forts.

Anropsinparametrarna är uttryck. Vid anropet kopieras uttryckens värde till motsvarande inparameter. När funktionen har exekverat färdigt kopieras utparametrarna till motsvarande anropsutparametrar. Anropsutparametrarna får vara färre än utparametrarna. Funktionen har exekverat färdigt när sista kommandot eller sista kommandot före nästa funktion har utförts eller när kommandot `return` utförts.

Vi ändrar funktionen så att skönsvärdena för *max* och *min* är 100 resp. 5. För sådant använder man funktionerna `nargin`, `nargout`, `nargcheck`, `error`, `warning` och `inputname`.

```
function [m n finalgear] = findgear(gear,min,max)
% Hittar det kugghjulspar som bäst approximerar en
given utväxling.
% [m n finalgear] = findgear(gear,min,max)
% hittar den bästa approximationen m/n till
```



```
% gear för m och n mellan min och max för
% positiva min, max och gear. Skönsvärde för max
% och min är 100 resp. 5.
msg = nargchk(1,3,nargin);
error(msg);
if nargin<3 max = 100; end
if nargin<2 min = 5; end
...
```


Kuggväxel forts.

Testkörning ger:

```
[m n g] = findgear(pi,25)
m = 88
n = 28
g = 3.1429
[m n g] = findgear(3.14159)
m = 22
n = 7
g = 3.1429
[m n g] = findgear
??? Error using ==> findgear
Not enough input arguments.
[m n g] = findgear(17,10,50,50)
??? Error using ==> findgear
Too many input arguments.
```

Man kan på liknande sätt göra funktionen beroende av antalet anropsutparametrar.

```
m = findgear(3.14159)
m = 22
[m n] = findgear(3.14159)
m = 22
n = 7
```


Global variabel

Programmera en räknare som kan stegas (med `count`) och avläsas (med `getcount`). Själva räknaren måste nås av båda funktionerna. Vanliga variabler i funktioner är ju lokala. Vi gör en variabel, `COUNTER`, global. Man brukar skriva globala variabler med versaler.

I filen count.m:

```
function count
% Stegar räknaren COUNTER
global COUNTER
if isempty(COUNTER) COUNTER = 0; end
COUNTER = COUNTER+1;
```

I filen getcout.m:

```
function c = getcount
% Avläser räknaren COUNTER
global COUNTER
c = COUNTER;
```


Global variabel forts.

Testkörning:

```
getcount
ans = []
count; count; getcount
ans = 2
COUNTER
??? Undefined function or variable 'COUNTER'
```

En variabel som ska sparas mellan anrop men bara behöver nås i en funktion kan istället göras `persistent`.

Slumpvandring

Programmera Brownsk rörelse/slumpvandring med lika sannolikhet för alla riktningar, förflyttning sträckan 1 per steg och start i origo. Låt filen brown.m innehålla:

```
function pos = brown(steps,runs)
% Simulerar Brownsk rörelse/slumpvandring.
% pos = brown(steps) ritar en vandring med steps
% steg.
% pos = brown(steps,runs) ritar slutpunkterna av
% runs stycken vandringar med vardera steps steg.
```

Slumpvandring forts.

```
msg = nargchk(1,2,nargin);
error(msg);
if nargin==1
  pos = b(steps,1);
else
  pos = [];
  for run = 1:runs
    pos = [pos b(steps,0)];
  end
  plot(pos(1,:),pos(2,:),'ko');
  pos = pos(:,end);
end
```


Slumpvandring forts.

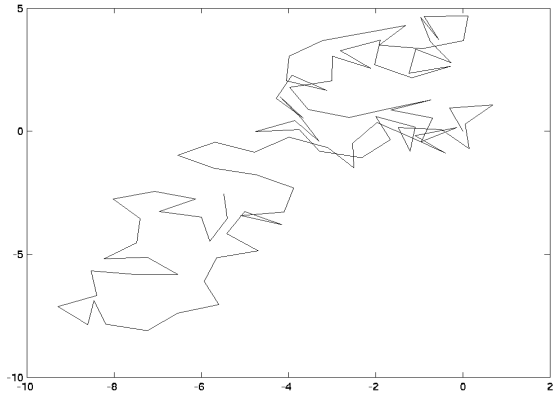
Som lokal funktion i samma funktionsfil:

```
function pos = b(steps,show)
% Simulerar en slumpvandring
% och plottar den om show är sant.
pos = [0;0]; x = pos;
for i = 1:steps
  a = rand*2*pi;
  pos = x(:,end)+[cos(a);sin(a)];
  x = [x pos];
end
if show
  plot(x(1,:),x(2,:),'k-');
end
```

Testkör:

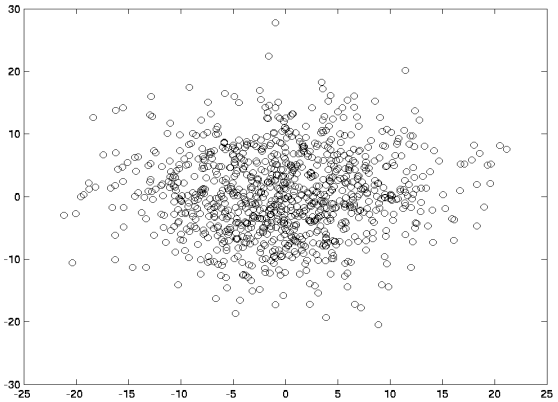
Slumpvandring forts.

`brown(100);` ger följande diagram:



Slumpvandring forts.

och `brown(100,1000);` ger:



Slumpvandring forts.

Området med slutpunkter efter n steg har praktiskt taget radien $\sqrt{n}$. Kommandofiler kan inte innehålla lokala funktioner. Utöver lokala funktioner kan man ha »privata» funktioner.

Funktionen rand ger ett värde som är större än eller lika med 0 och mindre än 1.

Effektivitet

Med kommandot `pcode` sparas p-koden som en fil med tillägget .p. Det finns hjälpmedel för att kombinera Matlab med andra språk såsom C, Fortran, Java och Ada.

Avlusning, felsökning, debugging

När programmet inte ger önskat resultat:

- Läs koden.
- Om ett fel hittas, rätta det och testa igen,
- annars lägg in spårutskrifter för att se i vilken ordning kommandona utförs och vissa variablers mellanresultat (ta bort `;`, lägg in `disp` eller `keyboard`).

I redigeringsfönstret finns möjligheter att sätta stoppunkter och stega.

Det finns ett särskilt fönster för att skapa profil för ett program, dvs. för att samla in och visa uppgifter om hur många gånger programmets olika delar har exekveras och om hur lång exekveringstid som går åt för respektive del.

Kommandot `tic` startar tidtagning och `toc` avläser och returnerar tid sedan senaste `tic`.