NTNU

Kunnskap for en bedre verden

TDT4110 Informasjonsteknologi grunnkurs:
Uke 41:

«Matlab programs» (kapittel 6)

Amanuensis Terje Rydland
Kontor: ITV-021 i IT-bygget vest (Gløshaugen)
Epost: terjery@idi.ntnu.no
Tlf: 735 91845

Noen innebygde funksjoner - Vektorisering

- sum
  - sum(<vektor>) -> summen av elementene
  - sum(<matrise>) -> kolonnesummer
- cumsum
  - Kumulative summer
- max (min)
  - max(<vektor>) -> største element
  - max(<matrise>) -> max element i hver kolonne
- find(<betingelse>)
  - I vektor: Indeksene
  - I matrise: Lineær indeks (1. kolonne, 2. kolonne, …)

Eksempler

vektorisering_4.m

```
clear, clc

v = [1 2 3 4 5];
m = [1:4; 5:8; 9:12];

sum(v), sum(m), sum(sum(m))

max(v), max(m), max(max(m))

find(v>3)
find(m==11)
```

```
ans =
    15
ans =
    15    18    21    24
ans =
    78
ans =
     5
ans =
     9    10    11    12
ans =
    12
ans =
     4     5
ans =
     9
>>
```


Preallokering av plass

- Allokering vil si å avsette plass til en variabel i minnet
- Tabeller som vokser gradvis er svært lite effektivt
  - Finne ny plass
  - Kopiere gamle verdier til ny plass
- Lønner seg å sette av nødvendig plass (pre-allokering)
  - zeros(n), zeros(n,m)
  - ones(n), ones(n,m)
  - true(n), true(n,m) / false(n), false(n,m) – logiske verdier

5

Kumulative summer

```
clear
clc

N = 100000;

% Uten preallokering av tabell
tic
sum1(1) = 1;
for i = 2:N
    sum1(i) = sum1(i-1) + i;
end

toc

% med preallokering av tabell

tic
sum2 = ones(1,N);
for i = 2:N
    sum2(i) = sum2(i-1) + i;
end

toc

% med innebygd funksjon

tic
sum3 = cumsum(1:N);
toc
```

vektorisering_3.m

```
Elapsed time is 0.020361 seconds.
Elapsed time is 0.000974 seconds.
Elapsed time is 0.000986 seconds.
>>
```

NTNU

6

Læringsmål og pensum

- Læringsmål:
  - Synlighet av variabler (scope)
  - Mer om funksjoner
    - Flere ut-variabler
    - Lokale funksjoner
    - Persistente variabler
  - Feilfinning:
    - Feiltyper, Tracing og Debugger
- Pensum
  - Kapittel 6 “Matlab Programs”

NTNU

Synlighet av variabler (scope)

- Synlighet av en variabel (scope)
  - Arbeidsområdet der den er gyldig
  - Command Window: base workspace
- Lokale variabler
  - Variabler definert i funksjoner
  - Synlige inne i funksjonen
  - Eksisterer ikke utenfor funksjonen
- Skript "ser" variabler definert i Command Window
  - Kan være kilde til feil/problemer
  - clear: tømmer arbeidsområdet
- Globale variabler
  - Ikke synlig inne i funksjoner uten å bli sendt som parameter
  - Synlig i alle scriptene våre.
  - Dårlig programmeringsskikk å dele globale variabler!

Eksempel: rest i a/b

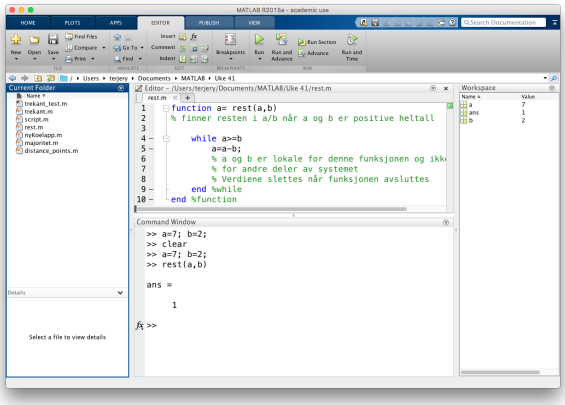
- Kommandovinduet og funksjonen har egne arbeidsområder
  - Parameterne overfører data inn i funksjonens arbeidsområde
  - Funksjonen returnerer verdi til kallende arbeidsområde
  - Samme navn på variabler i ulike arbeidsområder referer ikke til en og samme variabel
  - Funksjonens arbeidsområde aktiveres på nytt hver gang funksjonen kalles.
    - Husker ingenting mellom kallene

```
>> format compact
>> a = 7; b = 2;
>> c = rest(a, b)
c =
     1
>> a
a =
     7
>> b
b =
     2
>> a = 13; b = 7;
>> c = rest(a, b)
c =
     6
>> a
a =
    13
>> b
b =
     7
>>
```

```
function a = rest(a, b)
% finner resten i a/b når a og b
% er positive heltall

   while a >= b
       a = a - b;
   end % while

end % function
```

Arbeidsområder

- **Command window** har sitt Workspace (synlig i Matlabvinduet)
- Funksjonen **rest** har sitt arbeidsområde (usynlig) som er frikoblet fra **Command windows** arbeidsområde



```
function a= rest(a,b)
% finner resten i a/b når a og b er positive heltall

    while a>=b
        a=a-b;
        % a og b er lokale for denne funksjonen og ikke kjent
        % for andre deler av systemet
        % Verdiene slettes når funksjonen avsluttes
    end %while
end %function
```

rests workspace

(eksisterer bare så lenge funksjonen kjører)

a

b

Returnere flere argumenter (parametere)

- [rader, kolonner] = size(<matrise>)
- Eksempel: Er en verdi i en matrise i majoritet?
  - Majoritet: Flere av denne verdien enn av de andre tilsammen - mer enn 50% av verdiene i matrisen.
- Pseudokode:
  - Input: Matrise med verdier
  - Gjør om matrisen til en vektor
  - Sorter vektoren
  - Tell opp for verdien til elementet "i midten"
  - Sjekk om det er mange nok (antall > n – antall)
  - Returner:
    - Flagg (true/false),
    - Verdi
    - Antall

11

```
function [flagg, verdi, antall] =
majoritet(m)
% returnerer flagg = true hvis verdi er i
% majoritet i m – mer enn 50% av tallene
% i m er verdi
% verdi finnes da antall ganger i m

% Flytter verdiene inn i en vektor
v = m(:);
% Sorterer vektoren
v = sort(v);

% Finner antall elementer
n = length(v);

% Indeks til elementet i midten
if odd(n)
    midtIndeks = (n+1)/2;
else
    midtIndeks = n/2;
end

% Verdien i midten
verdi = v(midtIndeks);

% Teller antallet av verdien
antall = sum(v == verdi);
```

```
% Sjekker om det er mange nok
if antall > n – antall
    flagg = true;
else
    flagg = false;
end

end % function

function retur = odd(tall)
% Finner ut om tall er ett oddetall

if mod(tall,2) == 1
    retur = true;
else
    retur = false;
end

end % function odd
```


12

```
>> m
m =
     1     2     3     4
     5     6     7     8
     9    10    11    12
>> [stemmer, tall, antall] =  majoritet(m)
stemmer =
     0
tall =
     6
antall =
     1
>> n
n =
     1     2     3     4
    12    12    12    12
    12    12    12    13
>> [stemmer, tall, antall] =  majoritet(n)
stemmer =
     1
tall =
    12
antall =
     7
>>
```

Sub-funksjoner

- Kan deklarere mer enn en funksjon i en m-fil
- Etter den primære funksjonen
  - Sub-funksjoner, lokale funksjoner eller hjelpefunksjoner
- Usynlige og utilgjengelige utenfor m-filen
- Kalles fra primærfunksjonen som andre funksjoner
- Kan bidra til å dele opp programkoden
  - Enklere, bedre kode
- Hvorfor ikke skrive som vanlig funksjon?

Eksamen august 2012

Oppgave 2 a) (5 %)
Gitt funksjonen unknown vist under.

```
function [a b c] = unknown(a, b, c)

    if (b >= a) && (a > 10)
        b = a;
    elseif (a <=c) && (b >= a)
        if (c ~= a)
            c = a;
        elseif (b > c)
            a = b;
        else
            b = a;
        end
    else
        b = c;
    end
end
```

Anta at x=4, y=6 og z=4.
Hva blir verdien til x, y og z etter at vi har utført setningen

```
[x y z] = unknown(x, y, z)?
```

Funksjoner: Persistente variabler

- Variabler som beholder verdien mellom hver kjøring av funksjonen.
  - Husker verdien fra avslutningen av forrige utførelse av funksjonen.

```
function retur = funksjonsnavn()

persistent variabel

if isempty(varabel)
    variabel = <startverdi>;
end
```

- Bør brukes med forsiktighet
- Verdiene nullstilles:
  - clear all / clear functions / clear <funksjonsnavn>
  - Ved omstart av Matlab


```
function nyttNr = nyKoelapp()
      % trekker neste koelapp

      % største nr på kølapp
      maxNr = 5;

      % holder rede på forrige nr
      persistent forrigeNr
      if isempty(forrigeNr)
            forrigeNr = maxNr;
      end

      % finner nytt nr
      if forrigeNr == maxNr
            nyttNr = 1;
      else
            nyttNr = forrigeNr + 1;
      end

      % oppdaterer forrige nr
      forrigeNr = nyttNr;

end % function
```

nyKoelapp.m

Kjøring av kølapp

```
function nyttNr = nyKoelapp()
      % trekker neste koelapp

      % største nr på kølapp
      maxNr = 5;

      % holder rede på forrige nr
      persistent forrigeNr
      if isempty(forrigeNr)
            forrigeNr = maxNr;
      end

      % finner nytt nr
      if forrigeNr == maxNr
            nyttNr = 1;
      else
            nyttNr = forrigeNr + 1;
      end

      % oppdaterer forrige nr
      forrigeNr = nyttNr;

end % function
```

Start 1. kjøring	Arbeidsområde nyKoelapp	
Slutt 1. kjøring	Arbeidsområde nyKoelapp	maxNr : 5 nyttNr : 1 forrigeNr : 1
Start 2. kjøring	Arbeidsområde nyKoelapp	forrigeNr : 1
Slutt 2. kjøring	Arbeidsområde nyKoelapp	maxNr : 5 nyttNr : 2 forrigeNr : 2

Eksempelkjøringer

```
>> nyKoelapp()
ans =
   1
>> nyKoelapp()
ans =
   2
>> nyKoelapp()
ans =
   3
>> nyKoelapp()
ans =
   4
>> nyKoelapp()
ans =
   5
>> nyKoelapp()
ans =
   1
>> nyKoelapp()
ans =
   2
>>
```


```
>> nyKoelapp()
ans =
   1
>> nyKoelapp()
ans =
   2
>> clear nyKoelapp
>> nyKoelapp()
ans =
   1
>>
```

Feil i programmer

- Syntaksfeil (syntax errors)
  - Feil i bruken av språket – Matlab gir beskjed

```
>> v = 1::10;
??? v = 1::10;
          |
Error: Unexpected MATLAB operator.
```

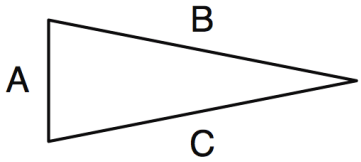
- Kjøretidsfeil (runtime errors)
  - Feil som oppstår under kjøring

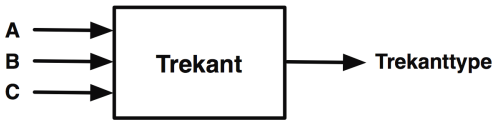
```
>> v(3)
??? Index exceeds matrix dimensions.
```

- Logiske feil
  - Programmet virker ikke som tiltenkt

Trekanter

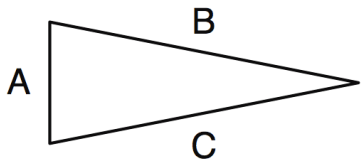
- Forutsetter: A <= B <= C
- Trekanttyper:
  - 0: Umulig (A + B <= C)
  - 1: Ubestemt
  - 2: Likebeint (A = B eller B = C)
  - 3: Rettvinklet ($A^2 + B^2 = C^2$)
  - 4: Likebeint og rettvinklet (A = B og $A^2 + B^2 = C^2$)
  - 5: Likesidet (A = B = C)
- Viktig å teste i riktig rekkefølge





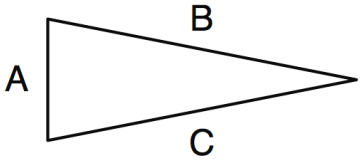
trekant.m

```
function type = trekant(a, b, c)
% finner trekanttypen definert av a, b, c

% validerer inn-parametrene
if (a>b) || (b>c)
    error('Trekantsidene maa gis i stigende orden');
end;

% bestemmer trekanttype
if (a+b) <= c
    % Ikke en trekant
    disp('Umulig trekant');
    type = 0;
elseif (a==b) && (b==c)
    % Likesidet trekant
    disp('Likesidet trekant');
    type = 5;
```



Forts.

```
elseif (a==b) || (b==c)
    % Likebeint trekant
    if rettvinklet(a, b, c)
        disp('Likebeint, rettvinklet trekant');
        type = 4;
    else
        disp('Likebeint trekant');
        type = 2;
    end
elseif rettvinklet(a, b, c)
    % Rettvinklet trekant
    disp('Rettvinklet trekant');
    type = 3;
else
    % Ubestemt trekant
    disp('Ubestemt trekant');
    type = 1;
end

end % function
```



Forts. (lokal funksjon)

```
function r = rettvinklet(a, b, c)
% Sjekker om trekanten er rettvinklet

toleranse = 0.001;

if (a*a + b*b - c*c) < toleranse
    r = true;
else
    r = false;
end

end % function
```


trekant_test.m

```
clear, clc

trekant(1, 2, 4);          % Umulig
trekant(1, 1, 1);          % Likesidet
trekant(1, 1, 1.5);        % Likebeint
trekant(1, 1, sqrt(2));    % Likebeint og rettvinklet
trekant(3, 4, 5);          % Rettvinklet
trekant(3, 4, 6);          % Ubestemt
```

Her ble det noe feil…

```
Umulig trekant
Likesidet trekant
Likebeint, rettvinklet trekant
Likebeint, rettvinklet trekant
Rettvinklet trekant
Rettvinklet trekant
>>
```


Debugger

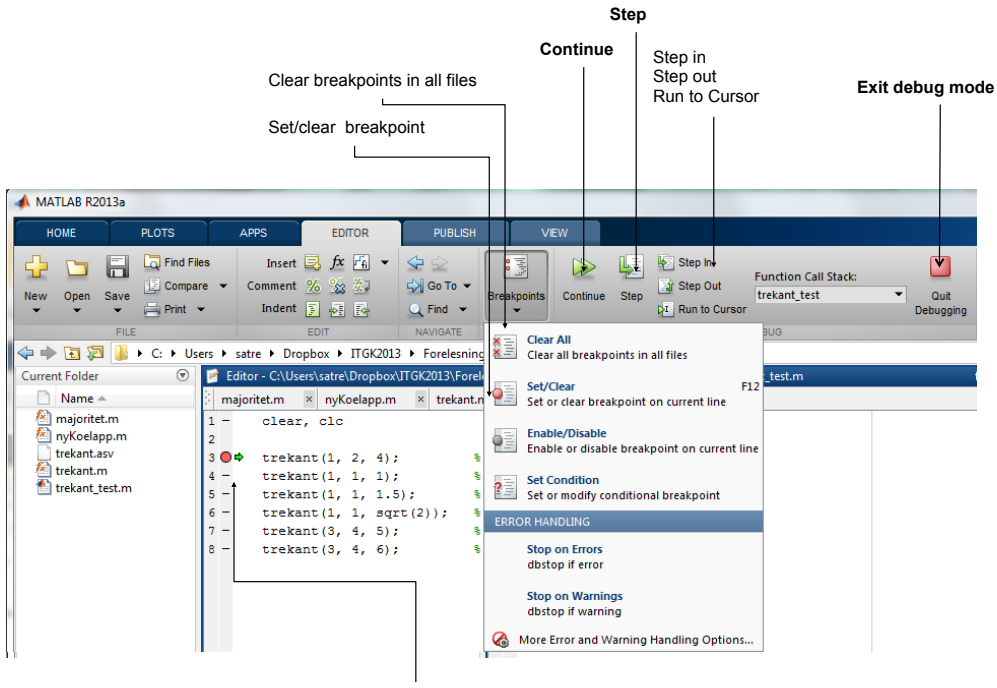
- Program eller funksjonalitet for å finne feil i programmer.
- Breakpoints
  - Programmet stopper på spesifiserte steder
- Kan se på variabler
- Steg-for-steg-utførelse
  - Følge programflyten
  - Inn i funksjoner som kalles (eller ikke)
- Fortsette (Continue)
- Alternativer:
  - Stirre og tenke, trace på papir (i hodet)
  - Ekstra utskriftskommandoer

På trekantproblemet

```
14 –    elseif (a==b) && (b==c)
15          % Likesidet trekant
16 –        disp('Likesidet trekant');
17 –        type = 5;
18 ●➡   elseif (a==b) || (b==c)
19          % Likebeint trekant
20 –        if rettvinklet(a, b, c)
21 –            disp('Likebeint, rettvinklet trekant');
22 –            type = 4;
23 –        else
```

Breakpoints markeres med rød prikk, grønn pil viser neste kommando som vil bli utført

R2015a: EDITOR-tab→Debug



Breakpoints markeres med rød prikk, grønn pil viser neste kommando som vil bli utført

Feil i rettvinklet

- Når c*c er større enn a*a+b*b blir "venstresiden" negativ og ulikheten blir sann uansett størrelsen på avviket. Må sjekke absoluttverdien!

```
function r = rettvinklet(a, b, c)
% Sjekker om trekanten er rettvinklet

toleranse = 0.001;

if (a*a + b*b - c*c) < toleranse
    r = true;
else
    r = false;
end

end % function
```

Eksamen 2010

Oppgave 3: Programmering (15 %)

I skihopp gis det poeng for hopplengde og stil.

a) (5 %) Poeng for hopplengde beregnes med følgende formel:

distance_points = 60 + (*jump_distance* – *kpoint*) *meter_value*

Tallene *kpoint* og *meter_value* er fastsatt for hver hoppbakke.

Hoppbakken i Granåsen har *kpoint* 124 og *meter_value* 1,8. En skihopper som hopper 140 meter i denne bakken vil få 60 + (140 – 124) 1,8 = 88,8 lengdepoeng (distance points).

Skriv en funksjon *distance_points* som tar inn parametrene *distance* (hopplengde), *kpoint* (k-punkt) og *meter_value* (meterverdi), og returnerer lengdepoeng.

b) (10 %) Et skihopp belønnes med 0–60 stilpoeng. Fem dommere gir 0–20 poeng hver. Den laveste og den høyeste poengsummen strykes, og de tre resterende poengsummene legges sammen og utgjør hoppets stilpoeng.

Hvis et skihopp får poengsummene 17, 17,5, 17,5, 18, 19, strykes 17 og 19, og stilpoengene blir 17,5 + 17,5 + 18 = 53.

Skriv en funksjon *style_points* som tar inn en usortert liste *points* med de fem dommerpoengsummene, og returnerer hoppets stilpoeng.

a)

```
function points = distance_points(jump_distance, kpoint, meter_value)
     points = 60 + (jump_distance - kpoint) * meter_value;
end
```

b)

Alternativ 1:

```
function points = style_points(refpoints)
     points = sum(refpoints) - min(refpoints) - max(refpoints);
end
```

Alternativ 2:

```
function points = style_points(points_list)
     points_list = sort(points_list);
     points = sum(points_list(2:4));
end
```