Matlab-Introductie (les 1)

Wat is Matlab?

MATLAB staat voor MATrix LABoratory.

Opstarten van Matlab

Dit hangt af van het onderliggende systeem (Windows, Linux, ...), Maar kortweg geldt bijna altijd: ga met de muis naar het juiste menuutje!

Afsluiten van Matlab

Gebruik: **quit** (↩) of **exit** (↩) ⇒ de Matlab-windows verdwijnen weer.

De 'help'-functies

Voor het snel zoeken naar en een korte uitleg van Matlab-commando's en functies is het vaak handig om gebruik te maken van de 'help'-mogelijkheid. Type in: **help trefwoord** (↩). Tevens bestaat er een handige help-functie (zie menuutje bovenaan het window onder **Help**, dan **Matlab Help** en vervolgens **Search** aanklikken).

Basis-rekenregels

De bewerkingen *, /, +, - spreken voor zich. Voor het machtsverheffen gebruikt Matlab het teken ∧. Indien de berekening wordt afgesloten met een ; dan wordt die wel uitgevoerd, maar zie je niet direkt het resultaat op het scherm terug. Het weglaten van ; geeft *ook* het antwoord in het scherm. Dit is vooral van belang als je met variabelen werkt die lange vectoren of grote matrices voorstellen.
Check **help format** voor de vorm waarin de getallen worden uitgewerkt.

Inlezen van files

M.b.v. het commando **load** kunnen datafiles worden ingevoerd. Zie ook **help load**.

Het bewaren van Matlab-variabelen

Met **save pietje** (↩) worden alle berekeningen (lees: variabelen en hun waarden) tot dan toe gestopt in de file **pietje.mat**. Wil je een volgende keer weer verder gaan met deze variabelen, dan kun je hiervoor ook het commando **load** gebruiken: **load pietje** (↩). NB. Deze file is *niet* met een tekstverwerker te lezen (heeft ook weinig zin)!

Het bewaren van een Matlab-sessie

Voor het bewaren van alle (letterlijke) teksten, getallen, en bewerkingen van een sessie (ook de onvermijdelijke tikfouten en foutmeldingen!) moet je *voor*dat je daarmee begint het commando **diary jantje** geven. Hiermee wordt alles in de

file **jantje** gestopt. Let op: als je klaar bent, moet je nog wel afsluiten met **diary off**. Anders weet Matlab natuurlijk niet, dat je klaar bent met deze actie. Deze file kun je *wel* gewoon in een tekstverwerker inladen, lezen en aanpassen (en dat is ook de bedoeling bij de opdrachten)!

Een paar nuttige commando's

Voor het schoonvegen van alle tot dan toe gebruikte variabelen: **clear all** (↩)
Voor het schoonvegen van, bijv., de variabele **klaasje**: **clear klaasje** (↩)
Om oudere commando's uit een sessie terug te halen: gebruik de Control toets in combinatie met de letter P (van Previous). De andere kant op: met Control N (van Next). In sommige typen schermpjes werken de 'pijltjes'-toetsen ook.

For loops: for...end

Voor het meerdere keren herhalen van eenzelfde formule zijn 'for'-lussen ('loops') handig. Voorbeeld: stel we willen de termen van een Fibonacci-rij berekenen die gedefinieerd is door de recursie:

$$a_{n+1} = a_n + a_{n-1}, \quad n = 1,\ 2,\ ...$$

Als we de eerste 100 termen berekenen met startwaarden $a_0 = 1,\ a_1 = -2$ dan kan dat in Matlab alsvolgt:

```
>> a0 = 1
>> a1 = -2
>> for n = 2:100
    a = a1 + a0
    a0 = a1;
    a1 = a;
   end
```

While loops: while...end

Voor het uitvoeren van een opdracht, zolang een bepaalde conditie geldt, zonder dat je van tevoren weet wanneer je moet stoppen, is een 'while'-lus nuttig. Voorbeeld: stel we willen de grootste waarde van n bepalen in de som $1^2 + 2^2 + ... + n^2$ zodanig dat de som kleiner is dan 100. Het volgende stukje Matlab voert dit voor je uit:

```
>> S = 1; n = 1;
>> while S + (n+1)^2 < 100
    n = n + 1; S = S + n^2;
   end
>> n
```

If...elseif...else...end

Voor het uitvoeren van verschillende opdrachten die afhangen van bepaalde condities bestaan constructies met 'if...else...end'. Bijvoorbeeld, voor het testen of π^e wel of niet groter is dan e^π kan je de volgende stappen uitvoeren in Matlab:

```
>> a = pi^exp(1); c = exp(pi);
>> if a >= c
     b = sqrt(a^2 - c^2)
   end
```

Hiermee krijgt b slechts een waarde toegekend als $a \geq c$. Er is geen uitvoer zodat we afleiden dat $a = \pi^e < c = e^\pi$. Handiger is de volgende constructie:

```
>> if a >= c
     b = sqrt(a^2 - c^2)
   else
     b = 0
   end
```

die er voor zorgt dat b altijd een waarde krijgt (in dit geval de waarde 0) en bevestigt dat $a < c$. Een iets meer uitgebreide versie met meerdere testmogelijkheden is de volgende:

```
>> if a >= c
     b = sqrt(a^2 - c^2)
   elseif a^c > c^a
     b = c^a/a^c
   else
     b = a^c/c^a
   end
```

De algemene vorm van een 'if-statement' is derhalve:

if *logische test 1*
    commando's uitvoeren als test 1 waar is
elseif *logische test 2*
    commando's uitvoeren als test 2 waar is
elseif *logische test 3*
    commando's uitvoeren als test 3 waar is
... net zo veel situaties (↝ **elseif***) beschouwen als nodig*
else
    uit te voeren in alle overige gevallen!
end

Voor de 'logische test' kan gebruik worden gemaakt van == ('gelijk aan'; niet te verwarren met de toekennning =), ~= ('niet gelijk aan'), >, <, >=, <=. Zie de Matlab-primer voor meer details.

Script-files

Een script-file, aangemaakt in een tekst-editer, krijgt een naam eindigend op **.m**, bijvoorbeeld: **henkie.m**. Een dergelijke file bestaat uit allerlei Matlab-commando's onder elkaar in de file ingetypt en door Matlab achter elkaar uitgevoerd, als je het volgende doet:

```
>> henkie
```

Function-files

Een function-file is hetzelfde als een script-file, maar dan met de extra mogelijkheid van één of meerdere invoerparameters en idemdito uitvoerwaarden. Voorbeeld: voor

het berekenen van het gemiddelde van een rij getallen, die we voor het gemak even in een vector z stoppen: **z=[1;2;3;4;5;6;7;8;9;]**, kan je de volgende function-file **gemid.m** aanroepen:

```
function y=gemid(x)
% dit is een commentaar-regel: eigen tekst dus invullen, indien gewenst...
m=length(x);
y=sum(x)/m;
```

In Matlab zeg je dan (in de juiste directory...):

```
>> z=[1;2;3;4;5;6;7;8;9;];
>> antwoord=gemid(z);
```

Indien je tekst in je scherm wilt krijgen, dan gebruik je, in een script/function-file of achter de Matlab-prompt, het commando **disp**:

```
>>disp('dit is een tekst die ik in het Matlab-schermpje wil zien');
```

Het tekenen van een grafiek

Stel we willen de grafiek van $\sin(x)$ op het interval $[-2, 3]$ tekenen met 'stapjes' van 0.01. In Matlab kan dit als volgt:

```
>> x = -2.0:0.01:3.0;
>> y = sin(x);
>> plot(x,y)
```

Hiermee wordt er een figure-window ('Figure No.1') geopend en de grafiek geplot.
• Een nieuw schermpje voor eventueel een andere grafiek wordt geopend met **figure(2)** ($\hookleftarrow$).
• Om te checken in welke window je op een bepaald moment aan het werk bent: **gcf** ($\hookleftarrow$); gcf staat voor get current figure.
• Veranderen van schermpje: **figure(NUMMER)** ($\hookleftarrow$) intypen met NUMMER=gewenste figure-window.
• Het plaatje schoonvegen kan met: **clf** ($\hookleftarrow$) en de 'huidige' window weggooien met **close** ($\hookleftarrow$).
• Alle figure-windows weg: **close all** ($\hookleftarrow$).
• LET OP (addertje onder het gras): hoe plotten we de grafiek van bijv. x^5 of van $\frac{x}{x+10}$ (probeer dit eens)?

'Versiering'

• Een titel meegeven: **title 'tekst boven grafiek'** ($\hookleftarrow$)
• Tekst onder de x-as: **xlabel 'tekst onder de x-as'** ($\hookleftarrow$)
• Tekst links van de y-as: **ylabel 'tekst links van de y-as'** ($\hookleftarrow$)
• Een vierkante figuur: **axis square** ($\hookleftarrow$)
Etcetera, etcetera (zie bijbehorende **help**-mogelijkheden)

Meerdere figuren in één window

Hiervoor is **subplot** handig. Voor meer info, type in: **help subplot** ($\hookleftarrow$).

Een plaatje bewaren

Om een plaatje te bewaren ('saven') zijn er meerdere opties, zie: **help print** ($\hookleftarrow$). Voor onze toepassingen is handig:

```
>> print -deps wimpie
```

Dit levert de file **wimpie.eps** op, die weer bekeken, eventueel afgedrukt, of in een .tex-file ingelezen kan worden.

Het tekenen van oppervlakken in 3d

Voor het plotten van oppervlakken, gedefinieerd door $z = f(x, y)$ kan o.a. gebruik worden gemaakt van **mesh** of **surf** in Matlab. Voorbeeld: stel we willen het oppervlak plotten gegeven door $f(x, y) = (x - 3)^2 - (y - 2)^2$ op het gebied $2 \leq x \leq 4, \quad 1 \leq y \leq 3$ met stapjes van 0.2 in beide richtingen, dan wordt dat in Matlab:

```
>> [x,y] = meshgrid(2.0:0.2:4.0, 1.0:0.2:3.0);
>> z = (x-3).^2-(y-2).^2;
>> mesh(x,y,z)
>> title('Zadel'); xlabel('x'); ylabel('y');
```

- Bedenk zelf waarom het commando **meshgrid** nodig is!
- Waarom staat hier .$\wedge$ en niet gewoon $\wedge$?
- Wat gebeurt er als we x en y vergeten in de 3e regel? D.w.z.

```
>> mesh(z)
```

i.p.v.

```
>> mesh(x,y,z)
```