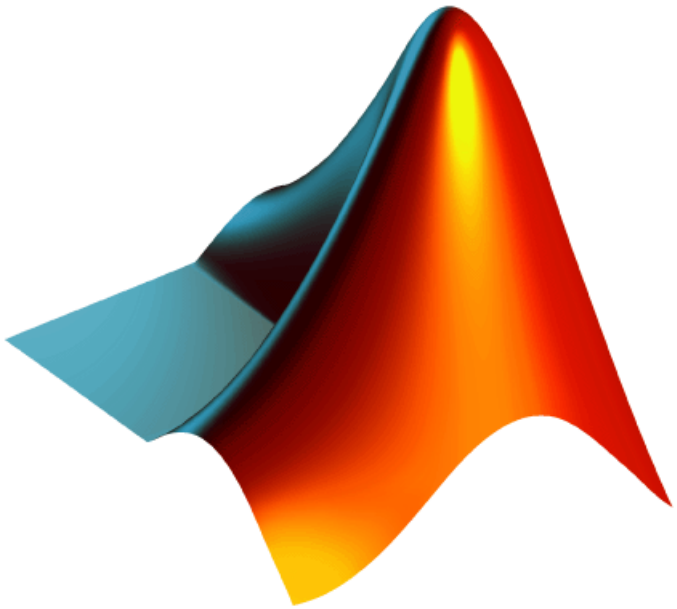
Een korte MATLAB introductie



K. Dekker, I.A.M. Goddijn & P. Sonneveld
2014: Aangepast door Tobias de Jong & Irene Verstraten, De Leidsche Flesch
2015: Wijzigingen door Jan van Staalduinen, De Leidsche Flesch

Inhoud

1. Hoe werkt MATLAB

1.1. MATLAB op de NUWD

MATLAB staat wel geïnstalleerd op de NUWD-computers, maar ze (de computers) weten standaard niet waar het geinstalleerd staat.

Om MATLAB werkend te krijgen, typ in een terminal:

```
echo ". /vol/share/software/matlab/mi/r2013a-bashrc" >> ~/.bashrc
```

Start vervolgens MATLAB door in een nieuwe (!) terminal te typen:

```
matlab
```

Het opstarten kan even duren!

1.2. Het invoeren van opdrachten

MATLAB is een interactief systeem: de ingetikte commando's (opdrachten), afgesloten door `Enter', worden direct uitgevoerd. Indien gewenst verschijnen de resultaten direct op het scherm. Om uitvoer op het scherm te onderdrukken moet een commando met een puntkomma worden afgesloten.

MATLAB is een numeriek pakket, dat wil zeggen dat er geen exacte berekeningen uitgevoerd kunnen worden. Het hangt van de opgegeven nauwkeurigheid af in hoeveel decimalen er wordt gerekend. MATLAB rekent in ongeveer 16 decimalen nauwkeurig (waarvan er standaard vijf op het scherm verschijnen).

Het intikken van opdrachten vraagt wel enige nauwkeurigheid. Het vergeten van bijvoorbeeld een `*` of een `)` kan ertoe leiden dat de opdracht niet of verkeerd wordt uitgevoerd. Hieronder volgt een lijst met links de in de wiskunde gebruikelijke notatie en rechts de vorm waarin het ingetikt moet worden (`a`; `b`; `x` en `y` zijn getallen).

Wiskundige notatie	Intikken
$a + b$	`a+b`
$a - b$	`a-b`
$a \cdot b$	`a*b`
$3xy$	`3*x*y`
$\frac{a}{b}$	`a/b`
a^b	`a^b`
$\sqrt{x}$	`sqrt(x)`
π	`pi`
e^x	`exp(x)`
$3 - 4i$	`3-4*i`
$\sin x$; $\arctan x$; ...	`sin(x), atan(x), ...`
e^x; $\ln x$	`exp(x), log(x)`
${}^{10}\log x$	`log10(x)`
$\lvert x \rvert$	`abs(x)`
∞	`Inf`

(i) Houd er rekening mee dat MATLAB case sensitive is. Dat wil zeggen dat er verschil is tussen hoofdletters en kleine letters. Het getal bijvoorbeeld wordt binnen MATLAB geschreven als `pi`, het invoeren van `Pi` zal een foutmelding opleveren.
(ii) Opdrachten in MATLAB worden afgesloten door het indrukken van de Enter. Het antwoord verschijnt direct op het scherm. Door het commando af te sluiten met een ; (puntkomma) wordt de uitvoer op het scherm onderdrukt.

1.3. De help-functie

MATLAB beschikt over een uitgebreide help-functie. Maak hiervoor gebruik van het commando `help` *onderwerp*. Met behulp van dit commando verschijnt er een toelichting bij onderwerp. Het typen van bijvoorbeeld:

```
>> help elfun                    (Druk nu de Enter in)
```

geeft informatie over de elementaire functies die MATLAB kent. Lees (via of Page Up en Page Down) wat er staat. Met alleen het commando `help` verschijnt een lijst van mogelijke onderwerpen.

1.4. Rekenen met getallen

MATLAB is in feite een zeer uitgebreide programmeerbare rekenmachine. Om met het pakket kennis te maken en te wennen aan het invoeren van commando's zullen we een aantal korte opdrachten uitvoeren.

Opgaven

(1) Voer in: `>> 12/9`. Op het scherm verschijnt de uitvoer van het commando in de volgende vorm:

```
ans =
    1.3333
```

Voer ook eens in: `>> 12\3`. Wat valt er op ?
(2) Vraag eens met het commando `help` informatie op over `SIN`, `Sin` en `sin` en probeer vervolgens eens $\sin(\frac{\pi}{2})$ uit te rekenen. Wat valt op ?

Rekenen met complexe getallen is voor MATLAB geen probleem. Het getal i kent MATLAB ook als i. Het getal $a + bi$ voer je in als `a+b*i`. Met de commando's `abs` en `angle` kun je de modulus resp. het argument van een complex getal vinden. Verder zijn er de commando's `real`, `imag`, `conj` en `exp` met voor de hand liggende acties (zo niet: raadpleeg `help`).

(3) Bereken $(1 + 2i)(5 - 3i)$ en $(1 + i\sqrt{3})^5$, en ga na dat $e^{\frac{3}{4}\pi i} = -\frac{1}{2}\sqrt{2} + \frac{1}{2}\sqrt{2}i$.

Standaard geeft MATLAB op het scherm 5-cijferige getallen met een vaste komma. Met behulp van het `format`-commando kun je opgeven welke notatievorm je wilt gebruiken. In onderstaande tabel staan verschillende opties:

In de rechterkolom staan de representaties van π, van $1.23456 \cdot 10^3$ en van $1.23456 \cdot 10^6$.

Commando	**Resultaat**	**Voorbeeld**
`format short`	5 cijfers,	3.1416
	scaled fixed point	0.0012
		$1.2346e-006$
`format short e`	5 cijfers,	$3.1416e+000$
	floating point	$1.2346e-003$
		$1.2346e-006$
`format long`	15 cijfers,	3.14159265358979
	scaled fixed point	0.00123456000000
		$1.23456000000000e-006$

Zonder toevoegingen is `format` hetzelfde als `format short`. Voor overige mogelijkheden: raadpleeg `help`.

(4) Ga eens na hoeveel verschillende formats MATLAB kent.

1.5. Toekenningen

We gaan de volgende matrix maken en toekennen aan een variabele.

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & -1 & 2 \\ 1 & 3 & -2 \end{bmatrix}$$

Voer in:

```
>> A = [1, 2, 1; 2, -1, 2; 1, 3, -2]
```

(i.p.v. komma's kun je ook spaties nemen)
Op het scherm verschijnt:

```
A =

    1    2    1
    2   -1    2
    1    3   -2
```

Voer in:

```
>> A2 = A^2; A3 = A.^2; A4 = A^4;
```

Omdat de MATLAB-commando's zijn afgesloten met `;` worden de resultaten onderdrukt. Wat is het verschil tussen `A2` en `A3`? (Je kan de matrices bekijken door `>> A2` in te voeren).

Een paar commando's om de actuele stand van zaken op te vragen of te beïnvloeden:

Intikken

`who`	de namen van alle variabelen die een waarde hebben worden opgevraagd;
`size(x)`	de afmetingen (aantal rijen, kolommen) van de variabele `x` worden opgevraagd;
`clear x`	de variabele `x` wordt verwijderd;
`clear all`	alle variabelen worden verwijderd;
`clc`	het scherm wordt leeg gemaakt (idem: Clear Command Window).

In MATLAB is het alleen mogelijk aan een variabele een *gedefinieerde* waarde toe te kennen Het volgende geeft bijvoorbeeld problemen:

```
>> clear x;
functie = x2 + 4 * sin(x)
```

Om een functie in te voeren moet deze door de gebruiker eerst geprogrammeerd worden. De gebruiker breidt dan de lijst MATLAB-commando's uit met een nieuw commando. In deze fase gaan wij hier niet verder op in. Het is niet [1] mogelijk formulemanipulatie uit te voeren met MATLAB, daartoe zijn andere pakketten beschikbaar (bijvoorbeeld: Sage, Maple (niet in Leiden), Mathematica). Het berekenen van de functiewaarde van boven bedoelde functie f in bijv. het punt $\pi/3$ kan als volgt gebeuren:
`>> x = pi/3`
`>> y = x2 + 4*sin(x)` Voer dit uit.

Opmerking
Met de ↑-toets kun je voorgaande commando's terughalen en, al of niet gewijzigd, op- nieuw uit laten voeren.

1.6. Werken met matrices

Zojuist heb je al een matrix ingevoerd. Een (kolom)vector kan worden gezien als een matrix met één kolom en kan dus als volgt worden ingevoerd:

```
b = [ 1; 6; 7; 5; -3 ]
```

Matrix-elementen kunnen afzonderlijk of in groepen worden opgeroepen of van een waarde voorzien worden, onderstaande tabel geeft een overzicht van een aantal mogelijkheden:

[1] Althans niet eenvoudig. Mocht je hierin geinteresseerd zijn, kijk dan eens naar het `syms` pakket. Helaas niet beschikbaar op de installatie op de MI computers.

Intikken

`A(i,j)`	het element in de i-de rij en j-de klom van de matrix A (a_{ij});
`A(:,j)`	de j-de kolom van de matrix A (a_j);
`A(i,:)`	de i-de rij van de matrix A;
`A(k:l, m:n)`	de deelmatrix van de matrix A bestaand uit de k-de t/m de l-de rij en de m-de t/m de n-de kolom (a_{ij} met ($k \leq i \leq l$ en $m \leq j \leq n$));
`v(k:l)`	de deelvector van v met als elementen $v_i (k \leq i \leq l)$.

En voor een vierkante matrix:

`diag(A)`	de hoofddiagonaal van A (als kolomvector);
`diag(A,k)`	de k-de nevendiagonaal van de matrix A. (Als $k > 0$ dan wordt de k-de nevendiagonaal boven de hoofddiagonaal genomen en als $k < 0$ dan de k-de nevendiagonaal onder de hoofddiagonaal.)

Opgave

(5) Voer de volgende 3×4-matrix in:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Voorspel en controleer wat het resultaat is van de volgende commando's:

```
>> B = A(:, 3)
>> C = A(1:3,2:4)
>> D = diag(C,-1)
>> A(1,1) = 9
>> A(2:3, 1:3) = [0 0 0; 0 0 0]
>> A(1:3, 1) = [1, 1, 5]
>> A(1:2, 1:2) = [-1, -1; 3, 3]
>> A = A(3:3, 1:4)
```

Opmerking
Bij één van de voorgaande commando's zou het voor de hand liggen als MATLAB een foutmelding zou geven. Welke, en waarom ?

1.7. Matrix-bewerkingen

Wat nu volgt is een greep uit het grote arsenaal aan matrix-bewerkingen waar de MATLAB- gebruiker standaard de beschikking over heeft.

Intikken

`C = A+B`	Matrixoptelling
`C = A-B`	Matrixverschil
`C = A*B`	Matrixvermenigvuldiging (onjuiste afmetingen geven een foutmelding)
`C = A.*B`	Componentsgewijze Matrixvermenigvuldiging
`C = A+k`	Bij elke element van de matrix wordt het getal k opgeteld
`C = A^k`	de gewone machtsverheffing ($k \in \mathbb{Z}$)
`C = A.^k`	de elementsgewijze machtsverheffing
`C = A'`	de geadjungeerde $A^\top$ (pas op met complexe getallen!)
`C = A\ B`	berekent voor een niet-singuliere, vierkante matrix A de unieke oplossing van de vergelijking $AX = B$ en voor een overbepaald stelsel (meer vergelijkingen dan onbekenden) een kleinste kwadraten oplossing.
`C = A/B`	oplossing van $XA = B$, analoog aan vorige commando
`[L,U,P] = lu(A)`	berekent de LU-ontbinding van PA
`L = eig(A)`	L is een kolomvector met de eigenwaarden van de matrix A
`[X,D] = eig(A)`	D is een diagonaalmatrix met op de diagonaal de eigenwaarden van de matrix A , de kolommen van X zijn bijbehorende eigenvectoren.
`[Q,R] = qr(A)`	produceert een QR-ontbinding van matrix A
`A = diag(v)`	geeft diagonaalmatrix A met diagonaal $v(1), v(2), ..., v(n)$
`A = zeros(n)`	A wordt de $n \times n$ nulmatrix
`A = zeros(n,m)`	A wordt de $n \times m$ nulmatrix
`A = eye(n)`	A wordt de $n \times n$ identiteitsmatrix
`A = ones(n,m)`	A wordt de $n \times m$ enenmatrix
`x = a:b`	x wordt de vector $(a, a+1, a+2, ..., b)$ $a \leq b$
`x = a:h:b`	x wordt de vector $(a, a+h, a+2h, ..., a+nh)$ $a \leq b$, waarbij $n = \lfloor \frac{b-a}{h} \rfloor$

Opgaven

(6a) Voer, door gebruik te maken van het commando `a:h:b` de volgende vector in:

```
x = [1 3 5 7 9 11 13 15]
```

(b) Zoek uit hoe het commando `linspace(a,b,n)` gebruikt kan worden om dezelfde matrix in te voeren. (Wat is het verschil met de `a:h:b` methode?)

(c) Voer, op twee verschillende manieren, de vector $y = [x_0, x_1, x_2, \cdots, x_n]$ in zodat $x_0 = 2, x_n = 22, n = 10$ en het verschil tussen twee opeenvolgende elementen steeds gelijk is.

(7) Voer (op een handige manier) de volgende matrix in:

$$A = \begin{bmatrix} 1 & 3 & 5 & \dots & 15 \\ 1 & 3 & 5 & \dots & 15 \\ 1 & 3 & 5 & \dots & 15 \\ 1 & 3 & 5 & \dots & 15 \\ 1 & 3 & 5 & \dots & 15 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & 3 & 5 & \dots & 15 \end{bmatrix}$$

(A heeft 50 rijen. Gebruik de commando's `a:h:b` en `ones`.)

(8) Gegeven is de rijvector $v = (1, 2, ..., 100)$. Bereken de som van de elementen van v op twee verschillende manieren.

a. Door het product te nemen met een geschikte kolomvector
b. Door een geschikt MATLAB-commando te gebruiken.
c. Bereken analoog de som van de eerste honderd kwadraten, dat wil zeggen: $1^2 + 2^2 + 3^2 + ... + 99^2 + 100^2$.

1.8. Plotten

Met het `plot`-commando kun je vectoren (en matrices) grafisch laten weergeven. Als je een functie invoert door een rij x-waarden met bijbehorende y-waarden te geven, krijg je met het commando `plot(x,y)` min of meer de grafiek op het scherm:

Intikken

`plot(y)`	tekent de punten $(1, y(1)), (2, y(2)), ..., (n, y(n))$ en verbindt ze met rechte lijnen
`plot(x,y)`	doet hetzelfde voor de punten $(x(1), y(1)), (2, y(2)), ..., (x(n), y(n))$ x en y mogen hier rij- of kolomvectoren met een gelijk aantal elementen zijn
`figure`	er wordt een nieuw Plot Window geopend.

Voorbeelden

Voer in:
```
x = [2,5,4,8,10,4];
y = [6,3,9,1,4,0];
plot(x,y);
```

Voer in:
```
x = 0:0.1:3.2;
y = sin(x);
plot(x,y);
```

Voer in:
```
x = linspace(0,3.2,33);
y = sin(x);
plot(x,y);
```

Nog een voorbeeld:

Voer in:
```
x = 1:10;
y = 20:30;
plot(x,y);
```

Wat gaat hier fout?

Bij het plotten van meerdere grafieken in één plaatje is het soms handig om verschillende kleuren en tekens te gebruiken. De volgende tabel biedt voldoende variatie-mogelijkheden; voor het volledige aanbod: raadpleeg `help plot`.

Symbool	Kleur	Symbool	Stijl
y	geel (yellow)	.	stip
r	rood	o	rondje
g	groen	-	lijn
b	blauw	-.	streep-stip
w	wit	*	ster

Voorbeeld

Voer in:
```
x = 0:0.1:3.2;
y = sin(x)
z = x.*sin(x);          (Elementsgewijze matrixsvermenigvuldiging!)
plot(x,y,' y-',x,z,'r-.');
```

Er zijn twee manieren om meerdere grafieken in één plaatje te krijgen: met één `plot`-opdracht, zoals boven, of met een aantal `plot`-opdrachten na elkaar, met tussendoor het commando `hold on`.

Voorbeeld

Voer in:
```
x = 0:0.1:3.2;
y = sin(x)
x2 = 0:0.3:3;
y2 = sin(x2)
plot(x,y,'g')
```

Ga nu naar het `Command Window` terug. Het `Plot Window` is dan nog steeds aanwezig.

Voer vervolgens in:
```
hold on;
plot(x2,y2,'yo');
```

en de twee grafieken staan in één plaatje (achter het `Command Window`). Alle volgende `plot`-opdrachten geven grafieken in hetzelfde plaatje. Wil je een heel nieuw plaatje maken dan kun je dat doen door het commando `hold off` te gebruiken, of door het `Plot Window` te sluiten.

Met onderstaande commando's kun je de 'layout' van een bestaande plot beïnvloeden:

Intikken

`axis([xmin xmax ymin ymax])`	de 'x-coördinaat' loopt van `xmin` tot `xmax`, de 'y-coördinaat' van `ymin` tot `ymax`;
`axis('off')`	de assen worden niet getekend;
`title('`$tekst$`')`	geeft de titel $tekst$ ($tekst$ moet tussen quotes gezet worden.);
`xlabel('`$tekst$`')`	er wordt $tekst$ langs de x-as gezet;
`ylabel('`$tekst$`')`	analoog aan `xlabel`;
`gtext('`$tekst$`')`	doet het (evt. nog lege) `Plot Window` in beeld verschijnen, en zet $tekst$ op de positie die je met de muis aanklikt.

Ook kun je de 'layout' van een bestaande plot beïnvloeden door op het item `Insert` in het menu van het `Plot Window` te klikken en vervolgens op `X label, Y label, Title` etc.

Voorbeeld

Voer in:
```
p = [2, 5, 5, 2, 2];
q = [2, 2, 5, 5, 2];
plot(p,q);
title('vierkant?')
```

Het resultaat is het `vierkant' met hoekpunten (2,2), (5,2), (5,5) en (2,5).

Door op de x-as en y-as het interval $[0,7]$ te nemen krijg je een duidelijker beeld:
```
>> axis([0,7,0,7]);
```

Wil je tenslotte dat het vierkant er als een vierkant uitziet, geef dan de opdracht:
```
>> axis('square');
```

Probeer tenslotte het volgende uit:
```
>> gtext('Dit is het punt (5,5)');
```

Opmerking
De zelf gedefinieerde instellingen m.b.v. `axis` blijven geldig zolang je met hetzelfde plaatje werkt.

Opgave

(9) Teken de (gladde) grafiek van $f(x) = \sin x$ op het interval $[0, 4\pi]$. Verdeel hierbij $[0, 4\pi]$ in 100 deelintervallen. Geef daarna in de grafiek de punten $(\frac{k\pi}{6}, \sin(\frac{k\pi}{6}))$, $k = 0, 1, \ldots, 24$ met een `*` aan.

1.9. Printen en opslaan

Als je het plaatje op het scherm er acceptabel uit vindt zien, dan kun je het als volgt vastleggen op papier (printen): klik in het menu van het `Plot Window` achtereenvolgens op de items `File` en `Print...` en tenslotte in het `Print Window` wat dan zichtbaar wordt op de `OK`-knop. Als de opdracht is verwerkt kom je weer terug in het `Plot Window`. Als je de plot wilt opslaan om bijvoorbeeld in een latex-verslag te verwerken om om met paint of een ander programma te bekladden, ga dan naar `File` en `save`, en kies vervolgens als extensie in plaats van `.fig` de gebruikelijke `.png` of `.pdf`.

1.10. Thuis werken met MATLAB

Bij Modelleren wordt uitgelegd hoe je MATLAB op je computer kan krijgen.

2. Programmeren in MATLAB (m-files)

2.1. Scripts

Het is mogelijk een aantal commando's in een zogenaamd *script* te zetten en die vervolgens uit te voeren.
Ook kunnen er procedures en functies, zogenaamde *function files*, worden gemaakt, die vanuit het `Command Window` of vanuit een script kunnen worden aangeroepen.

Voor het maken van een script file kun je op de volgende manier te werk gaan.

Klik achtereenvolgens op de menu-items `File, New` en `M-file`. De *M-file Editor/Debugger* opent een nieuw document, waarin een programma kan worden geschreven. Ben je klaar met schrijven dan moet je het 'programma' (dat nu alleen nog maar op het scherm staat) opslaan in een daarvoor geschikte map. Klik de menu-items `File` en `Save As` aan en maak eventueel eerst een map aan waarin je het script wilt opslaan. Geef vervolgens het bestand een naam, bijvoorbeeld: `naam.m` en bewaar het. De extensie `.m` is belangrijk! (Als je hem niet geeft, dan voegt de editor die standaard toe.)

Voor het uitvoeren van het script kun je nu op de volgende twee manieren te werk gaan:

(i) Voer in het `Command Window` in:
```
>> naam
```
(ii) Klik in het menu van de *M-file Editor/Debugger* op `Debug` en `Run`.

Let op: je functie moet je in dezelfde map opslaan als waar MATLAB drt.

Voorbeeld
Maak een .m-file waarin de volgende commando's staan:

```
% Plotten van de sinus op het interval [0,4pi]
x = 0:0.1:4*pi;
y = sin(x);
plot(x,y);
hold on;
x = 0:pi/6:4*pi;
y = sin(x);
plot(x,y,'r*');
```

Bewaar dit onder de naam `plotsin.m`, ga naar het `Command Window` en voer in:

```
>> plotsin
```

Opmerkingen

(i) Het is een goede gewoonte om een programma te beginnen met één of enkele regels commentaar over de werking van het programma (Een commentaarregel wordt voorafgegaan door %.). Door in te voeren:
`>> help naam`
krijg je dit commentaar later snel in beeld. Commentaarregels na een blanco regel of na een MATLAB-opdracht worden niet getoond. Zij worden opgevat als intern commentaar.

(ii) Krijg je na een poging naam uit te voeren de melding:
`??? Undefined function or variable 'naam'.`
dan heb je een tikfout gemaakt of MATLAB kent het pad naar de file `naam.m` niet. Door de `Current Directory' te wijzigen in het pad naar de file `naam.m` kun je het laatste probleem opheffen. Dit kan door op de knop te klikken en vervolgens het juiste pad te kiezen.

(iii) Zoals de naam al aangeeft kun je de *M-file Editor/Debugger* ook gebruiken om fouten in je programma's op te sporen en te verwijderen (`*debuggen*') Kijk eens onder de menu-items `Debug` en `Breakpoints` en let eens op de kleuren die de editorgebruikt.

(iv) Laat de naam van een `.m file` nooit met cijfers beginnen! MATLAB herkent scripts met dergelijke namen niet.

2.2. Function files

Zoals gezegd, er kunnen twee soorten `.m-files` onderscheiden worden: *scripts* en *function files*.
In scripts worden simpelweg een aantal commando's in een `.m-file` gegroepeerd. Door middel van function files kun je zelf functies en procedures aan MATLAB toevoegen. Scripts kun je uitvoeren, function files roep je aan.

De eerste regel van een function file luidt in het algemeen als volgt:

```
function y = functienaam(x)
```

Ook hier is het een goede gewoonte om te te beginnen met een of meer regels commentaar over de werking van de functie.

Met `help functienaam` kun je dit commentaar later eenvoudig opvragen.

Voorbeelden

(i) De functie $f : \mathbb{R} \to \mathbb{R}$ met functievoorschrift $f(x) = x^2 e^{-x^2}$ kun je toevoegen als m-file `newfunc`:

```
function y = newfunc(x)
% newfunc(x) berekent (elementsgewijs) x^2 * exp(-x^2)
y = x.^2 .* exp(-x.^2);
```

Oefening: doe dit, en maak een plot van deze functie op het interval $[0,5]$.

(ii) De volgende functie berekent het grootste verschil tussen twee opeenvolgende elementen van een vector:

```
function result = maxdiff(x)
% maxdiff(x) berekent het grootste verschil |x(i) - x(i-1)|
% van vector x
L = length(x);
v = x(2:L) - x(1:L-1);
result = max(abs(v));
```

Als je deze `m-file` opslaat als `maxdiff.m`, dan kan bijvoorbeeld het `maximale verschil' tussen twee opeenvolgende elementen van de vector $y = (f(0), f(0.1), f(0.2), \ldots, f(3,0))$ worden berekend door in te voeren:

```
>> x = 0:0.1:3;
>> y = newfunc(x);
>> z = maxdiff(y)
```

of door het volgende script te schrijven en te `runnen':

```
x = 0:0.1:3.0;
y = newfunc(x);
z = maxdiff(y);
disp(z);
```

Opmerkingen

(i) De variabelen `result` en `x` in de regel `function result = functienaam(x)` zijn, (hoe kan het anders in MATLAB), matrices.
`x` heet de invoervariabele, `result` de uitvoervariabele.
In plaats van een enkele variabele, kun je ook géén of meerdere variabelen als invoer en/of uitvoer nemen. Enkele voorbeelden van een mogelijke aanheffen zijn:

```
function result = functienaam
```

(geen getalsmatige invoer, één uitvoergetal)

```
function [res1, res2, ..., resk] = functienaam
```

(geen getalsmatige invoer, meerdere uitvoergetallen)

```
function functienaam(x)
```

(geen getalsmatige uitvoer, één invoer)

```
function functienaam(x1, x2, ..., xn)
```

(geen getalsmatige uitvoer, n invoer)

```
function [res1, res2, ..., resk] = functienaam(x1, x2,..., xn)
```

(meerdere in- en uitvoervariabelen)

(ii) Alle MATLAB-opdrachten in function files behalve de aanhef moeten worden afgesloten met een ; (punt-komma). Doe je dit niet kun je ongewenste uitvoer krijgen. Kijk ook nog eens terug naar de voorbeelden van function files.

(iii) Een belangrijk verschil tussen scripts en function files is het karakter van de variabelen. De variabelen die in een function file gebruikt worden zijn lokaal. Ze leven als het ware alleen binnen de procedure. We gaan hier niet verder in op dit punt.

(iv) Voor één-regelige functies als in voorbeeld (i) kent MATLAB (vanaf versie 5) het commando `inline`. De functie `newfunc` is daarmee in te voeren als

```
>> newfunc = inline('x.^2.* exp(-x.^2)','x');
```

waarna je bijvoorbeeld *newfunc*(5) kunt berekenen via

```
>> newfunc(5)
```

Vraag desgewenst hulp op over `inline`.

De volgende opgaven zijn nu niet nodig om te maken.

Opgaven

(10) Maak een function file `verwissel.m` die als invoer een matrix heeft en twee indices i, j en als uitvoer een matrix die gelijk is aan de ingevoerde matrix op de i-de en j-de rij na; de i-de en j-de rij zijn verwisseld. Maak hierbij gebruik van zo min mogelijk hulpvariabelen.

(11) Maak een function file `inwprod.m` die als invoer twee kolomvectoren heeft en als uitvoer het inwendig product van deze vectoren.

(12) Maak een function file `stat.m` die als invoer een vector x heeft, die bijvoorbeeld tentamenresultaten bevat, en als uitvoer het gemiddelde en de standaarddeviatie van deze resultaten.

(13) Maak een function file `plaatje.m` die als invoer $a, b \in \mathbb{R}$ en $n \in \mathbb{N}\setminus\{0\}$ heeft en als uitvoer een plaatje van de grafiek van de functie $f : [\alpha; \beta] \to \mathbb{R}$ gegeven door $f(x) = x^n$. Hierbij $\alpha = \min(a, b)$ en $\beta = \max(a, b)$.

3. Control-flow (for, while, if ...)

3.1. De for-loop

Het commando voor een for-loop in MATLAB is:

```
for k = 1:n
    k
end;
```

Opmerking
Vermijd `for-loops` als er een (efficïenter) alternatief voorhanden is. Onder andere door handig gebruik van matrices (bijvoorbeeld met `linspace()`): is dit vaker mogelijk (vooral de wat meer ervaren programmeur!) dan je zou verwachten, en het kan de rekentijd aanzienlijk bekorten.

Voorbeeld
De rij van Fibonacci $0, 1, 1, 2, 3, 5, 8, 13, \ldots$ is vastgelegd door de eerste twee waarden en het voorschrift $f_{k+1} = f_k + f_{k-1}$.

Het volgende `programma' genereert de eerste vijftig Fibonacci-getallen.

```
format short e;
f = zeros(50,1);
f(2) = 1; (Merk op dat f(1) = 0.)
for k = 3:50
f(k) = f(k-1) + f(k-2);
end;
disp(f);
format;
```

Na het uitvoeren van de `for-loop` is f een vector met 50 getallen, en het k-de element van f is precies het k-de Fibonacci-getal.
Als je alleen geïnteresseerd bent in de waarde f_{50}, dan spaar je op de volgende manier enige geheugenruimte uit (maak een `m-file fibo.m`):

```
format short e;
f1 = 0; f2 = 1;
for k = 1:48
f3 = f1+f2;
f1 = f2;
f2 = f3;
end;
```

```
disp(f2);
```

Er zijn nu slechts drie variabelen f1, f2 en f3 in het spel; na elke stap wordt de waarde die niet meer nodig is `weggegooid'.

Voorbeeld
De som van de eerste N kwadraten kun je berekenen met de volgende commando's. (Maak een function-file `somkwad.m` waarbij N in- en het resultaat uitgevoerd kan worden.)

```
s = 0; for k = 1:N
s = s + k^2;
end;
```

Maar kan ook worden berekend met de volgende commando's. (Maak opnieuw een function file waarbij N ingevoerd en het resultaat uitgevoerd kan worden.)

```
v = 1:N;
enen = ones(N,1);
kwad = v.^2;
somkwad = kwad*enen
```

Met de commando's `tic` en `toc` kun je de verstreken tijd in beeld krijgen.
Voeg in de twee bovengenoemde function files als eerste regel toe:

```
tic
```

en als laatste regel:

```
toc
```

en roep ze aan voor N = 10, 100, 1000 en 10000.

Opgaven

(14) $A = \begin{bmatrix} 1 & 3 & 5 & \dots & 15 \\ 1 & 3 & 5 & \dots & 15 \\ 1 & 3 & 5 & \dots & 15 \\ 1 & 3 & 5 & \dots & 15 \\ 1 & 3 & 5 & \dots & 15 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & 3 & 5 & \dots & 15 \end{bmatrix}$ (A heeft 50 rijen)

Voer matrix A op drie verschillende manieren in.

a. Door een dubbele for-loop te gebruiken.
b. Door één loop te gebruiken.
c. Door geen enkele loop te gebruiken (zie opgave 7).

(15) Construeer de 30 bij 30 matrix A met op de hoofddiagonaal 2-en, direct boven en onder de diagonaal 1-en en verder overal 0-en. Doe dit met behulp van het commando `diag(..,..)`: als `v` een (rij- of kolom-)vector is van lengte m, dan wordt met
`V = diag(v,k)`
een vierkante matrix V geconstrueerd met de vector `v` als de k-de nevendiagonaal, en met verder overal 0-en.

3.2. De while-loop

Het commando voor een `while-loop` in MATLAB is:

```
while  cond
       <opdr.1>;
       <opdr.2>;
       ...
       ...
       <opdr.m>;
end
```

Hier wordt de serie opdrachten `<opdr.1>,<opdr.2>,...,<opdr.m>` uitgevoerd zolang voldaan is aan de voorwaarde *cond*.

De voorwaarden zullen zijn van het type $n < N, n = p, n \neq m$, of combinaties ervan.
De volgende tabel geeft een overzicht van de `logische operatoren':

`n > m`	n is groter dan m;
`n < m`	n is kleiner dan m;
`n >= m`	n is groter dan of gelijk aan m;
`n <= m`	n is kleiner dan of gelijk aan m;
`n == m`	n is gelijk aan m;
`n ~= m`	n is niet gelijk aan m;

Opgave

(16) Hoeveel van de getallen sin(1), sin(2),...,sin(1000) zijn groter dan 0.5?
(Antwoord: 332)
(Ook dit KAN zonder for- of while-loops)

Met de volgende commando's kun je het rekenproces tijdens een loop in de gaten houden en beïnvloeden.

`disp(`$tekst$`')`	toont de tekst $tekst$ op het scherm;
`disp(X)`	toont de inhoud (zonder de naam) van matrix X (X zal in het algemeen een `tekstvector' zijn.);
`x = input(`$tekst$`')`	toont $tekst$ op het scherm en wacht op een waarde van de gebruiker (gevolgd door Enter), en kent deze waarde aan x toe. (Indien de gebruiker direct Enter geeft dan wordt x de lege matrix.);
`str = input(`$tekst$`,`s`')`	toont $tekst$ op het scherm en wacht op een antwoord van de gebruiker, bijvoorbeeld ja of nee (gevolgd door Enter) en kent dit antwoord aan str toe;
`[x,y] = ginput(n)`	(`graphical input') vraagt $n$ punten $(x_i, y_i)$, die worden opgeslagen in de vectoren `x` en `y`, de gebruiker klikt de punten aan met de muis;
`[x,y] = ginput`	gaat door met punten op te slaan tot de gebruiker Enter geeft;
`pause`	onderbreekt een loop tot de eerstvolgende toetsaanslag
`pause(n)`	onderbreekt de loop n seconden (bijvoorbeeld om even een tussenresultaat te bekijken);
`break`	breekt een loop af;
`error(`$tekst$`')`	breekt een loop af en geeft tevens de (fout)melding $tekst$.

Hoe via het `Command Window` gecommuniceerd kan worden maakt het volgende voorbeeld van een `m-file` duidelijk:

```
\% De som van twee getallen!
a ='';
while (isempty(a))||(length(a) > 1)
disp('Geef a een waarde !')
a = input('a = ');
end;
b = '';
while (isempty(b))||(length(b) > 1)
disp('Geef b een waarde !')
b = input('b = ');
end;
som = a + b;
disp(['De som van a en b is : ',num2str(som)]);
```

Als isempty(a) = 0 of isempty(b) = 0 hoeft de tweede voorwaarde niet meer gecontroleerd te worden!

3.3. Het if-statement

Het commando voor een `if-statement` in MATLAB is:

```
if      cond
        <opdrachten A>;
else
        <opdrachten B>;
end;
```

wordt eerst gekeken of aan de voorwaarde $cond$ is voldaan. Zo ja, dan worden de opdrachten A uitgevoerd, zo nee, opdrachten B. Het gedeelte `else ...` mag ontbreken, en het mag ook op zijn beurt een `vertakking' geven. De constructie gaat er dan als volgt uitzien:

```
if      cond_A
        <opdrachten A>;
elseif  cond_B
        <opdrachten B>;
else
        <opdrachten C>;
end;
```

Opgave

(17) ('bubble sort')
Schrijf een function file `sorteer.m` die de elementen van een vector `x` naar oplopende grootte sorteert, en wel op de volgende wijze:
- `doorloop' de vector `x` van voor naar achter,
- en verwissel `buren' die niet in de juiste volgorde staan,
- houdt met een variabele $change$ bij of er verwisselingen plaatsvinden. Zo ja, herhaal dan de stappen en zo nee, dan is x geordend.

4. Uitdagende opgaven

4.1. Impliciete plots

Bekijk de help van `ezplot`. Deze matlab routine is handig om snel een indruk te krijgen van een functie waarvan je alleen een impliciete formule hebt, dus van de vorm $f(x, y) = 0$.

Gebruik `ezplot` om plots te maken van de volgende functie:

$$f(x, y) = y^2 - \frac{1}{2}x^4 + x^2 = c$$

Met $c \in [-5, 5]$. Als je dit leuk vind, speel dan ook eens met `ezcontour` of `ezsurf`

4.2. Mooie kleurtjes

Bij vorige opgaven (bijvoorbeeld bij de impliciete plots) ben je vast in de situatie geweest dat je meerdere functies met een for-loop wil plotten, en deze dan verschillende kleuren wil geven. De essentie om dit te doen zijn zogenaamde colormaps. Deze bestaan uit `[R,G,B]` waarden die je vervolgens als kleurargument aan een plotfunctie kan meegeven. Bekijk de help van `jet` en `plot`, en maak een plot met mooie kleurtjes door een argument mee te geven aan `plot`.

Sommige plotfuncties (zoals `ezplot`) accepteren geen kleur-argument. Dan nog is het mogelijk op een zelfde manier een kleur toe te kennen met de `set` functie, namelijk als volgt:

```
h = plotfunc(...)
set(h, 'Color', C)
```

Waar `h` een handle naar de plot is, en `C` een kleurnaam is zoals `'y'` of een RGB-triplet.

4.3. **Mandelbrot

De Mandelbrotset staat erom bekend dat het misschien wel de mooiste plaatjes uit de wiskunde genereerd. Hoewel de theorie erachter verre van triviaal is, zijn de plaatjes vrij makkelijk te maken.

De mandelbrotset bestaat uit de punten c in het complexe vlak waarvoor geldt dat bij herhaald toepassen van de functie $f(z) = z^3 + c$, z begrensd blijft.

Om dit te visualiseren gaan we voor een rooster van punten c de volgende functie itereren:

$$z_{n+1} = z_n^2 + c$$

En controleren voor welke n deze functie buiten de cirkel om de oorsprong met straal 2 komt. Merk op dat voor punten in de mandelbrotset dit in principe oneindig is! We moeten dus een maximaal aantal iteraties $Nmax$ implementeren. De waarde waarvoor de iteratie voor het eerst buiten de cirkel met straal 2 komt gaan we plotten.

Hint: gebruik `imagesc()` om de matrix met de waardes van $n(c)$ te plotten.

5. Referenties

[1] D.C. Lay. Linear Algebra and its Applications, third edition, Update. Addison Wesley, U.S.A. 2003.
[2] J. Stewart. Calculus, Early Transcedentals (International Student Edition), sixth edition. Thomson, U.S.A. 2008.