Oefeningen Numerieke Wiskunde

Oefenzitting 1: Een eerste kennismaking met MATLAB

1 Inleiding

MATLAB is een pakket dat voornamelijk interactief wordt gebruikt en dat vooral geschikt is om te werken met matrices. Vele functies uit welgekende pakketten voor lineaire algebra (LAPACK, ...) worden in MATLAB aangeboden. Zo zijn er bv. commando's beschikbaar voor het berekenen van de eigenwaarden en eigenvectoren van een matrix of voor de LU-decompositie van een matrix.

Deze tekst is een inleiding voor MATLAB. Eerst worden de basiscommando's overlopen. Sectie 3 geeft dan een redelijk uitgebreid overzicht van alle commando's die volgens ons gekend moeten zijn om enigszins vlot met het pakket te kunnen werken. Sectie 4 overloopt de belangrijkste program flow constructies en in Sectie 5 wordt het schrijven van MATLAB functies en scripts besproken. Sectie 6 geeft tenslotte een overzicht van de belangrijkste commando's om figuren te maken.

Het is uiteraard niet de bedoeling om onmiddellijk alle commando's in deze tekst van buiten te kennen. We hopen echter dat de lezer een idee krijgt van wat er allemaal kan en dit document later kan gebruiken om de belangrijke commando's op te zoeken.

MATLAB bevat een uitgebreide helpfunctie en het is een goede gewoonte om deze steeds te raadplegen bij het gebruik van nieuwe commando's. Door het intypen van

```
>> doc
```

verschijnt het helpvenster, dat ook een ingebouwde zoekfunctie heeft. Om te weten te komen wat de functie exp precies doet, typt men

```
>> doc exp
```

Je kan in plaats van doc ook het commando

```
>> help exp
```

gebruiken. De documentatie verschijnt hierdoor in de Command Window waarin je aan het werken bent, wat vaak handiger is. We verwijzen verder ook naar de help functie online, op `http://www.mathworks.nl/help/matlab/`.

2 Basiscommando's

Bij het werken met MATLAB krijg je op het scherm telkens een prompt. Dit is: >> . Hierachter kan je de gewenste bevelen intikken. Als je op de enter-toets drukt, wordt het bevel uitgevoerd. Merk op dat je enkel achtereenvolgende commando's kan ingeven. Het is niet mogelijk om bijvoorbeeld met de muis op een eerder uitgevoerd commando te klikken

om dit te wijzigen. Er bestaat wel de mogelijkheid om met de pijltjes toetsen te navigeren doorheen de commando's die je eerder ingegeven hebt. Zo kan je eenvoudig een commando opnieuw onder de cursor halen en eventueel aanpassen.

2.1 Het invoeren van variabelen

Getallen

```
< naam-van-de-variabele > = < waarde >
```

Voorbeeld:

```
>> x = 0.65
```

heeft als resultaat dat MATLAB aan de variabele x de waarde 0.65 toekent. MATLAB antwoordt :

```
x =

    0.6500
```

Vectoren

Vectoren worden aangemaakt door getallen gescheiden door een spatie, een komma of een puntkomma te plaatsen tussen vierkante haken. Als de getallen gescheiden worden door een puntkomma, is de vector een kolomvector, anders is het een rijvector.

```
>> v = [1 8 6 7]

v =
    1   8   6   7

>> v = [1; 8 ;6 ;7]

v =
    1
    8
    6
    7
```

Vectoren die bestaan uit equidistante elementen kan je aanmaken door gebruik te maken van de dubbelpuntoperator. Voorbeeld:

```
a:b:c
```

maakt een rijvector aan met als elementen:

```
a, a+b, a+2b, ..., c
```

Om een kolomvector te maken gebruik je de transpose operator '

```
(a:b:c)'
```

Matrices

Matrices worden rij per rij gedefinieerd. Een vierkante haak [maakt MATLAB duidelijk dat er een matrix volgt. Een matrix wordt afgesloten met]. Je kan een matrix op 2 manieren invoeren. Voorbeeld: Je wil

$$A = \begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{pmatrix}$$

dan tik je ofwel

```
>> A = [ 0 1 2 ; 3 4 5 ; 6 7 8 ]
```

ofwel

```
>> A = [ 0 1 2
         3 4 5
         6 7 8 ]
```

In beide gevallen antwoordt MATLAB :

```
A =

     0     1     2
     3     4     5
     6     7     8
```

De rijen worden dus gescheiden door een ; of door een enter, de verschillende elementen van een rij worden gescheiden door een spatie of door een komma.

2.2 De uitvoer

De inhoud van een variabele kan je opvragen door gewoon de naam van de variabele in te tikken, en dan op de enter-toets te drukken.

```
>> x
```

MATLAB antwoordt :

```
x =

    0.6500
```

Dit is het standaard uitvoerformaat van MATLAB. Wetenschappelijke notatie verkrijg je met het bevel:

```
>> format short e
```

Wil je meer cijfers van je getal zien, dan kan dat met de bevelen

```
>> format long
```

of

```
>> format long e
```

Na het intikken van dit laatste bevel, beantwoordt MATLAB

```
>> x
```

met

```
x =

     6.500000000000000e-01
```

Meerdere opties wat uitvoer van getallen betreft, kan je opvragen met het commando

```
>> doc format
```

Merk op dat MATLAB steeds uitvoer geeft als een commando wordt ingetypt. Om dit te vermijden (wat bijvoorbeeld handig is als men grote matrices als uitvoer verwacht) kan je achter het commando een ; plaatsen.

2.3 Selecteren van elementen

Je kan elementen uit een vector selecteren door gebruik te maken van ronde haakjes:

`v(i)`	geeft het i-de element van een rij- of kolom-vector terug.
`v(end)`	geeft het laatste element van een rij- of kolom-vector terug.
`v([1,3,4])`	geeft het eerste, derde en vierde element van de vector v terug

Je kan matrix-elementen als volgt selecteren :

`A(i,j)`	geeft het element op de i-de rij en de j-de kolom van A.
`A(i,:)`	geeft de hele i-de rij van A.
`A(:,j)`	geeft de hele j-de kolom van A.
`A(i:j,k:l)`	geeft de deelmatrix A(i ... j , k ... l).
`A(end,end)`	geeft het laatste element in de matrix.

Voorbeeld:

```
>> M = A(1:2,2:3)

M =

     1     2
     4     5
```

2.4 Rekenkundige operatoren

De volgende rekenkundige bewerkingen kunnen zowel voor scalars als voor vectoren en matrices gebruikt worden:

`+` optelling
`-` aftrekking
`*` vermenigvuldiging
`/` rechtse deling (`a/b` is theoretisch equivalent met `a*inv(b)`)
`\` linkse deling (`a\b` is theoretisch equivalent met `inv(a)*b`)
`^` machtsverheffing

Voorbeeld:

```
>> N = A(1:2,1:2);
>> P = N * M
P =
     4     5
    19    26
```

berekent het matrixproduct van de matrices `N` en `M`. Uiteraard moeten deze matrices de juiste dimensies hebben. Om de dimensie van de matrix `M` op te vragen kan men het commando

```
>> size(M)
```

gebruiken.

De volgende rekenkundige bewerkingen kunnen zowel voor vectoren als voor matrices gebruikt worden:

`.*` elementsgewijze vermenigvuldiging
`./` elementsgewijze deling
`.^` elementsgewijze machtsverheffing

Voorbeeld:

```
>> v = [ 9 5 4 ];
>> w = [ 6 7 0 ];
>> u = v .* w

u =
    54    35     0
```

Als je het resultaat van een bewerking niet toekent aan een variabele, wordt hiervoor automatisch de variabele ans gebruikt.

```
>> 2^5

ans =

    32

>> ans - 5

ans =

    27
```

2.5 Relationele en logische operatoren

Relationele operatoren		Logische operatoren	
==	gelijkheid	&	elementsgewijze AND
~=	ongelijkheid	|	elementsgewijze OR
<	kleiner dan	~	elementsgewijze NOT
<=	kleiner dan of gelijk	&&	"Short-Circuit" AND
>	groter dan	||	"Short-Circuit" OR
>=	groter dan of gelijk		

Een overzicht van deze operatoren krijg je met

```
>> help relop
```

Het resultaat van een logische uitdrukking is 1 als ze waar is en 0 als ze onwaar is.
Voorbeeld:

```
>> 8 == ( 2^3 )

ans =
     1

>> 2 > 6

ans =
     0
```

Merk op dat ronde haakjes kunnen worden gebruikt om de volgorde van operatoren op te leggen. Strikt genomen zijn deze in bovenstaand voorbeeld overbodig omdat de machtsverheffing een hogere prioriteit heeft dan de gelijkheid. Meer informatie over de volgorde van operatoren kan men bekomen op `http://www.mathworks.nl/help/matlab/matlab_prog/operator-precedence.html`

Je kan relationele operatoren ook toepassen op vectoren of matrices. Het resultaat is een logische array. De logische operatoren &, | en ~ kunnen toegepast worden op deze logische arrays.
Voorbeeld:

```
>> [1 2 3 4 3 5] <= 3

ans =
     1     1     1     0     1     0

>> [1 2 3 4 3 5] <= 3 & [1 2 3 4 3 5] >= 2

ans =
     0     1     1     0     1     0
```

De logische operatoren && en || zijn vooral van belang bij `while` lussen en `if else` structuren, zie Sectie 4.

2.6 Selecteren van element met logische arrays

Naast het selecteren van elementen in vectoren en matrices door exacte indices op te geven zoals beschreven in Sectie 2.3, kan je ook logische indices opgeven. Hiermee kan je bijvoorbeeld alle elementen van een matrix nul maken die aan een bepaalde voorwaarde voldoen:

```
>> A = [1 2 3; 4 5 6]

A =
     1     2     3
     4     5     6

>> A( A<=3 | A==6 ) = 0

A =
     0     0     0
     4     5     0
```

Met de functie `find` kan je dan de indices opvragen van elementen die aan een bepaalde voorwaarde voldoen. Merk op dat je eigenlijk een logische array aan `find` geeft.

```
>> a = [1 3 4 -1 5 -2]

a =
     1     3     4    -1     5    -2

>> find(a<0)

ans =
     4     6
```

Je kan dus ook elementen selecteren m.b.v. `find`

```
>> A(find( A<=3 | A==6 )) = 0
```

maar dit is eigenlijk een omweg.

2.7 Oefeningen

Oef 1. Maak een rijvector v met 100 elementen van de vorm: $1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots$

Oef 2. Maak een kolomvector met als elementen $1024, 512, \ldots, 16, 8, 4, 2, 1$

Oef 3. Maak voor de volgende functies f de vector $[f(0), f(0.1), f(0.2), \ldots, f(2)]$.

(a) $f(x) = x^2 + 2x + 1$.

(b) $f(x) = \dfrac{x + 3^x}{x^{1/2} + x^{-2}}$

Oef 4. Maak twee matrices

$$G = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix} \text{ en } H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \end{pmatrix}$$

(a) Bereken het verschil tussen G en H.

(b) Vermenigvuldig G met H elementsgewijs.

(c) Vermenigvuldig G met H^T .

(d) Bewaar de eerste drie kolommen van de matrix G in G_3 en bereken G_3^2.

Oef 5. Gegeven $x = \begin{bmatrix} 3 & 15 & 9 & 12 & -1 & 0 & -12 & 9 & -6 & 1 \end{bmatrix}$, voer het commando uit dat

(a) de negatieve waarden van x gelijk stelt aan 0;

(b) de waarden van x groter of gelijk aan 9 stockeert in de vector y.

(c) het element in x na het element 12 gelijk stelt aan 100 (gebruik makend van `find`).

3 Functies in MATLAB

In wat volgt worden verschillende functies overlopen die nuttig zijn voor het vak Numerieke Wiskunde. We proberen een redelijk volledig overzicht te geven.

Let op: je gebruikt best geen functienamen als naam voor je variabelen, want dan werkt de functie niet meer. Zie oefening 6.

3.1 Elementaire functies in MATLAB

Het argument geef je in tussen ronde haakjes. Voorbeeld:

```
>> log(1)

ans =
     0
```

Een overzicht vindt je met

```
>> doc elfun
```

Belangrijke functies zijn

```
sqrt, log, log10, exp, sin, cos, tan, asin, sinh, asinh, ...
abs, round, ceil, floor, sign, rem ...
```

3.2 Algemene functies in MATLAB

Resultaten van berekeningen in MATLAB worden vaak elders opnieuw gebruikt. De waarden van variabelen kunnen bewaard worden in een .mat-bestand. Dit gebeurt met het commando

```
>> save filename
```

waardoor alle gebruikte variabelen worden bewaard in het bestand met naam 'filename.mat' De extensie .mat hoeft niet opgegeven te worden. Hoeven er maar een aantal variabelen te worden opgeslagen, dan dienen die gespecificeerd te worden na de bestandsnaam. Het opgeslagen bestand kan in MATLAB worden ingelezen met de opdracht

```
>> load filename
```

waarbij de extensie .mat mag worden weggelaten. De variabelen met hun waarden kunnen nu weer gebruikt worden. **Let op:** alle variabelen in de workspace die dezelfde naam hebben worden overschreven.

Volgende tabel geeft een overzicht van verschillende algemene MATLAB-functies:

`who`/`whos`	geeft een lijst van de variabelen die je gebruikt.
`clear`	geeft de gebruikte geheugenruimte terug vrij.
`clc`	wist de Command Window.
`save`	bewaart de variabelen van de huidige MATLAB-sessie in een .mat-bestand.
`load`	leest de variabelen uit een .mat-bestand terug in zodat deze terug de waarden hebben van net voor de laatste save.
`pause`	laat Matlab wachten op de gebruiker
`path`	zet hierin de padnamen van de directories waar MATLAB naar functies (.m-bestanden) moet zoeken.
`pwd`	geeft de huidige map in een string.
`disp`	geeft variabele of string weer in Command Window
`fprintf`	schrijft geformatteerde string naar bestand of Command Window
`tic`/`toc`	meet de uitvoeringstijd van de commando's tussen `tic` en `toc`

De functie `fprintf` is zeer handig voor het printen van output naar de Command Window. Deze output is een geformatteerde string. Voorbeeld:

```
>> fprintf('Dit is iteratie %i: x = %0.3f, fout = %0.3e \n', ...
      3,0.12345,0.00098765)
Dit is iteratie 3: x = 0.123, fout = 9.877e-04
```

De variabelen (of in dit geval getallen) die worden meegegeven worden geformatteerd naar:

`%i`	een geheel getal
`%0.3f`	een komma getal met 3 cijfers na de komma
`%0.3e`	een getal in wetenschappelijke notatie met 3 cijfers na de komma

Merk ook de drie puntjes `...` op. Deze dienen om een regel te splitsen over meerdere lijnen.

3.3 Voorgedefinieerde variabelen

`pi`	het getal pi.
`Inf`	oneindig
`NaN`	Not a Number, om resultaten als $\frac{0}{0}$ en $\frac{\infty}{\infty}$ voor te stellen

3.4 Functies voor vectoren en/of matrices

Handige tools

`length`	geeft het aantal elementen van een vector terug.
`size`	geeft het aantal rijen en kolommen van een matrix terug.
`zeros(m,n)`	geeft een matrix van dimensie m x n met allemaal nullen.
`ones(m,n)`	geeft een matrix van dimensie m x n met allemaal eentjes.
`eye(n)`	geeft de eenheidsmatrix terug van dimensie n.
`rand(m,n)`	geeft een (m x n)-matrix met randomgetallen, uniform verdeeld tussen 0 en 1.
`linspace(a,b,N)`	geeft een rijvector van N equidistante punten tussen a en b terug
`reshape(A,m,n)`	hervormt de matrix A naar dimensie $m \times n$, kolom per kolom.
`find`	zoekt indices van niet-nul elementen
`fliplr(A)`	draait de matrix `A` om in de links/rechts richting. (noot: dit is equivalent met `A(:,end:-1:1)`)
`flipud(A)`	draait de matrix A om in de onder/boven richting.
`diag(A)`	als A een vector: diagonaalmatrix met de elementen van A op de diagonaal. als A een matrix: vector met de diagonaalelementen van A.
`tril(A)`	geeft het onderdriehoeksdeel terug van A.
`triu(A)`	geeft het bovendriehoeksdeel terug van A.
`A'`	geeft de getransponeerde van de matrix A.
`A(:)`	zet alle kolommen van A onder elkaar in een vector.
`sort`	sorteert de elementen van een vector of matrix.
`sortrows`	sorteert de elementen van een matrix.
`sum`	sommeert de elementen van een matrix volgens rijen of kolommen
`prod`	vermenigvuldigt de elementen van een matrix volgens rijen of kolommen
`max` en `min`	geeft maximum/mininum terug en ook de bijhorende index

Numerieke algoritmen

`inv`	berekent de inverse matrix (vierkante matrix !).
`det`	berekent de determinant (vierkante matrix !).
`cond`	berekent het conditiegetal van een matrix.
`rank`	bepaalt de rang van een matrix.
`norm`	berekent de norm van een matrix of van een vector.
`A\b`	berekent de oplossing van het stelsel $Ax = b$ als de matrix A vierkant is (gebruikt verschillende algoritmen naargelang het type matrix A, voor een algemene volle matrix is dit Gauss eliminatie)
`lu`	geeft de factoren terug na Gauss-eliminatie (met optimale rij-pivotering).
`chol`	berekent de Cholesky-factorizatie.
`qr`	berekent de QR factorizatie.
`svd`	berekent de singuliere waarden ontbinding.
`eig`	berekent eigenwaarden en eigenvectoren.
`polyval`	evalueert een veelterm.
`poly`	berekent veelterm met gegeven nulpunten
`roots`	berekent nulpunten van een veelterm

Voorbeelden

Na het intikken van

```
>> [L,U] = lu(A)
```

geeft MATLAB twee matrices L en U terug; L is een benedendriehoeksmatrix (op een permutatie van de rijen na) met 1-tjes op de diagonaal en U is een bovendriehoeksmatrix. Om een stelsel Ax = b op te lossen, gebruik je de MATLAB-deling

```
>> x = A \ b
```

Hierbij wordt impliciet ook Gauss-eliminatie met optimale rij-pivotering gebruikt. Als A geen vierkante matrix is (dit betekent dat het aantal vergelijkingen verschilt van het aantal onbekenden), dan krijg je de kleinste kwadraten oplossing.

```
>> [V,D] = eig(A)
```

geeft een diagonaal matrix D met de eigenwaarden en een volle matrix V waarvan de kolommen de eigenvectoren zijn zodat A V = V D. (Hier moet A een vierkante matrix zijn!)

Merk op dat `x = eig(A)` een andere output geeft dan `[V,D] = eig(A)`. Je moet dus expliciet het aantal output argumenten meegeven dat je wil terugkrijgen.

3.5 Oefeningen

Oef 6. Deze oefening laat zien waarom je best geen functienamen gebruikt als naam voor je variabelen. Voer de volgende opdrachten uit in de onderstaande volgorde:

(a) `>> sqrt`

(b) `>> sqrt(2)`

(c) `>> a = sqrt(2)`

(d) `>> sqrt = sqrt(2)`

(e) `>> sqrt(1)`

(f) `>> sqrt(4)`

(g) `>> clear sqrt`

(h) `>> sqrt(4)`

Op welke lijnen (a t/m h) wordt `sqrt` als een functie gezien en op welke als een variabele? Waarom krijg je bij (e) geen foutmelding en bij (f) wel?

Oef 7. Bereken de volgende uitdrukkingen met MATLAB:

- $\sin(\frac{\pi}{4})$
- e^2
- $\frac{\tan(x)}{\sqrt{1+x^2}}$, met $x = 4$

Oef 8. Voer de twee opdrachten

```
>> f = 2
>> g = 3 + f
```

uit. Bekijk de variabelen in het geheugen met `whos`. Sla de variabelen f en g op in het bestand `probeer.mat`. Wis alle variabelen met `clear` en controleer met `whos`. Laad de file `probeer.mat`. Kijk of de variabelen f en g bekend zijn. Waar is het bestand `probeer.mat` terecht gekomen?

Oef 9. Evaluatie van de opdracht pause(30) heeft als effect dat MATLAB 30 seconden pauze houdt. Voer deze opdracht uit en pas dan de toetscombinatie ctrl+C toe. Wat is het effect?

Oef 10. Maak met behulp van `rand` een random 10 bij 10 matrix R en bereken zijn inverse. Controleer of hun product gelijk is aan de eenheidsmatrix.

Oef 11. Gebruik het commando `diag` om een diagonaalmatrix met de elementen $10, 20, 30, \ldots 80$ aan te maken. Bereken daarna de determinant van deze matrix en controleer dat deze gelijk is aan het product van de elementen op de diagonaal. (Zie de functies `prod` en `det` .)

4 Constructies: if, for, while, switch

Deze constructies kunnen zowel interactief in de Command Window als in een .m-bestand (zie Sectie 5) worden gebruikt.

4.1 if - elseif - else - end

Syntaxis:

```
if condition
  statements;
elseif condition
  statements;
else
  statements;
end
```

Voorbeeld:

```
if i<=n && fout > 1E-6
  i=j+1;
  j=i/2;
elseif i == j+1
  j=0;
else
  j=2*i;
end
```

Het voorbeeld slaat op weinig. De eerste conditie is wel een goed voorbeeld van de operator `&&`. De regels erna worden enkel uitgevoerd als beide condities waar zijn. Indien echter de eerste conditie `i<=n` onwaar is, dan wordt onmiddellijk naar de `elseif` overgegaan.

4.2 for - end

Syntaxis:

```
for variable = vector
  statements;
end
```

Voorbeeld:

```
x=0;
for i=1:5
   x=x+i;
end
```

Het is mogelijk om met het commando `break` bij het voldaan zijn van een bepaalde conditie de lus te verlaten:

```
x=0;
for i=1:5
   x=x+i;
   if abs(x) < 10
      break
   end
end
```

4.3 while - end

Syntaxis:

```
while condition
  statements;
end
```

Voorbeeld:

```
while x <= 5
  x=x+1;
  y=y+x;
end
```

Ook hier kan je gebruik maken van `break`.

4.4 switch - case - otherwise - end

Syntaxis:

```
switch variable
  case value1
    statements;
  case value2
   statements;
 ...
  case valueN
   statements;
 otherwise
   statements;
end
```

Voorbeeld:

```
switch a
  case 0
    x = A\b;
  case 1
    x = eig(A);
  case 2
    x = sum(A,1);
 otherwise
   disp('Error')
end
```

5 Zelf functies en scripts schrijven

5.1 Functies

In MATLAB kan je zelf ook functies schrijven om complexe berekeningen uit te voeren. Zodoende kan je het pakket naar je eigen behoefte verder uitbouwen. Een dergelijke functie wordt meestal met de MATLAB editor geschreven en opgeslagen (in ASCII-formaat) in een bestand met extensie '.m'. De eerste regel van een functie bevat bijvoorbeeld:

```
function [x,y] = naam(a,b,c)
```

Als je dit bestand opslaat met de naam naam.m, kan je deze functie in MATLAB oproepen als:

```
>> [x,y] = naam(a,b,c)
```

Als je

```
>> x = naam(a,b,c)
```

doet, dan krijg je standaard enkel de eerste output variabele terug.

De parameters a, b, c zijn de gegevens die aan de functie worden doorgegeven. Het resultaat moet binnen de functie worden toegekend aan de variabelen x en y. Uiteraard moeten de namen van de variabelen en parameters binnen het .m-bestand en bij het oproepen van de functie niet hetzelfde zijn. Let op dat de **uitvoerparameters tussen vierkante haken** staan en de **invoervariabelen tussen ronde haken**. Als er één of geen enkele uitvoerparameter is, kan je de vierkante haken weglaten. De ronde haken kan je weglaten als er geen invoer nodig is.

Na deze functiehoofding voeg je best wat commentaar toe over het doel van de functie, en bijvoorbeeld de volgorde waarin je de argumenten moet geven. Dit kan je doen door je regel te laten beginnen met een `%` -teken. Deze eerste commentaarlijnen verschijnen als antwoord op

```
>> help naam
```

Je kan uiteraard ook op andere plaatsen in je functie commentaar schrijven.

Vergeet niet om achter elk commando een ; te schrijven, anders krijg je bij het uitvoeren van de functie alle tussenresultaten op je scherm. Om de functie te verlaten voor de laatste opdracht is uitgevoerd, bv. bij het slagen van een bepaalde test, gebruik je `return`.

5.2 Scripts

Indien je een functie wenst te schrijven zonder invoer en uitvoerparameters, dan hoef je de eerste regel

```
>> function [x,y] = naam(a,b,c)
```

niet toe te voegen aan je bestand. Het resulterende bestand is nu geen functie maar een script dat je kan uitvoeren door in de editor op de run knop te duwen of door de naam van het bestand in te geven in het Command Window. Het voordeel aan een script is dat alle variabelen die gebruikt zijn na het uitvoeren van het script nog steeds in je workspace zitten. (Je kan dit checken met `whos`.)

6 Grafieken

`figure`	open een nieuwe figuur
`plot(x,y)`	plot vector `y` t.o.v. vector `x`
`semilogx(x,y)`	zoals `plot`, maar met een logaritmische schaal op de x-as
`semilogy(x,y)`	zoals `plot`, maar met een logaritmische schaal op de y-as
`loglog(x,y)`	zoals `plot`, maar met een logaritmische schaal op de x-as én de y-as
`hold on`	houdt huidige plot en schaal van de assen (lineair, logaritmisch, ...) vast zodat nieuwe plots worden toegevoegd i.p.v. de oude te overschrijven
`hold all`	zoals `hold on`, maar elke nieuwe plot is in een andere kleur
`hold off`	doet `hold on` teniet
`xlabel`	voeg label toe aan x-as
`ylabel`	voeg label toe aan y-as
`title`	voeg titel toe
`legend`	voeg legende toe
`subplot`	toon meerdere plots naast en/of boven elkaar in dezelfde figuur
`clf`	wis de plot op de huidige figuur

Voorbeelden

Plot de exponentiële functie in $[-1, 1]$:

```
>> x = linspace(-1,1,100);
>> figure
>> plot(x,exp(x))
>> xlabel('x')
>> ylabel('y')
>> title('exponential function')
```

Plot een sinus en een cosinus in een figuur met 'handle' 3 en voeg een legende toe:

```
>> x = linspace(-2*pi,2*pi,100);
>> figure(3)
>> plot(x,sin(x))
>> hold all
>> plot(x,cos(x))
>> legend('sin','cos')
```

Plot 2^{-x} in $x = 0, 1, 2, \ldots, 30$ met rode sterretjes en met logaritmisch geschaalde y-as in figuur 3. Als figuur 3 al bestond, wis dan eerst alle plots:

```
>> x = 0:30;
>> figure(3),clf
>> semilogy(x,2.^(-x),'*r')
```

6.1 Oefeningen

Oef 12. Maak een script `oefening_logplots.m`. Plot hierin de functies $f(x) = 2^x$ voor $x \in [0, 30]$ en $g(x) = x^3$ voor $x \in [0, 100]$. Doe dit in verschillende figuren, waarbij je de commando's `plot`, `loglog` en `semilogy` vergelijkt. Wat valt je op?

Oef 13. Schrijf een script `oefening_legende.m` waarin de functies sin(t) en cos(t) getekend worden op het interval $0 \leq t \leq 2$ in een figuur. Zorg ervoor dat de sinus in het rood en de cosinus in het groen getekend wordt. Benoem de assen en voorzie een legende.

Oef 14. Maak een script `oefening_subplots.m`. Maak een figuur met twee subplots boven elkaar. In de eerste subplot teken je de functie $f(x) = \dfrac{\sin(x) + 2x}{1 + x^2}$ op het interval $[0, 2\pi]$, in de tweede subplot plot je een random vector met lengte 1000.

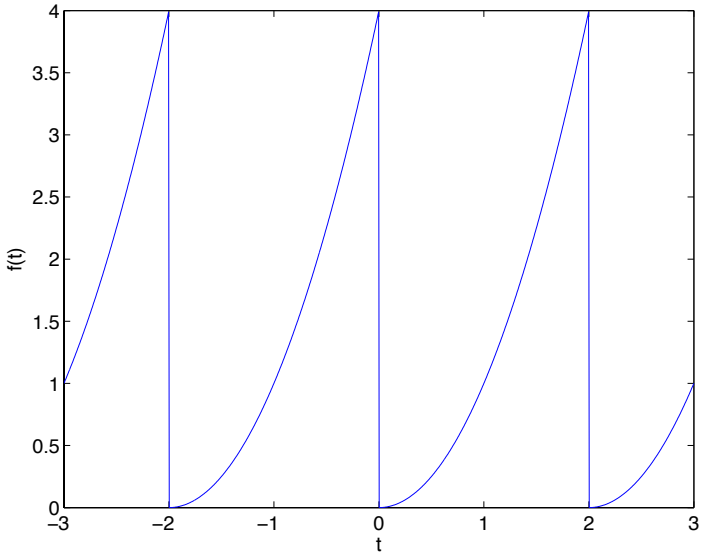
Oef 15. De periodieke functie f met periode 2 is gegeven door:

$$f(t) = \begin{cases} t^2 & , \quad 0 \leq t \leq 2 \\ f(t-2) & , \quad 2 \leq t \\ f(t+2) & , \quad t < 0 \end{cases}$$

Definieer deze functie in MATLAB. Zorg ervoor dat de functie ook voor vectoren werkt. (Hint: er zijn minstens drie mogelijke implementaties. De snelste implementatie maakt gebruik van `rem`.) Ga na dat je functie werkt door de commando's

```
>> t = linspace(-3,3,1000);
>> figure
>> plot(t,f(t))
```

uit te voeren en te vergelijken met Figuur 1.



Figuur 1: $f(t)$ van Oefening 15