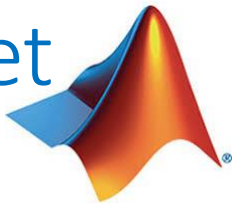
Handleiding wiskunde met MATLAB



Belangrijkste commando's binnen MATLAB

help help <commando> clc clear clear <variabele> who del, backspace, →, ← ↑, ↓ [ctrl + c]	BASISHANDELINGEN MATLAB Help, gebruiksvriendelijke helpfiles met voorbeelden en tutorials geeft de commando's per topic en legt het gebruik van het commando in detail uit wis alle info op het scherm verwijder variabelen uit het geheugen verwijdert een specifieke variabele uit het geheugen geeft een lijst van de gebruikte variabelen navigeren binnen het commando selecteren van een eerder gebruikt commando berekening onderbreken
+ – * / ^ <variabele> = <getal> pi vpa(a) inf *i, j*	INVOER basisbewerkingen tussen getallen toekenning π geeft de numerieke waarde weer van variabele a ∞ imaginaire eenheid, $i^2 = -1 = j^2$
<commando>; ans NaN disp('N=',N)	UITVOER uitvoer onderdrukken laatste niet toegekende uitkomst geen geldig getal als uitvoer toont de waarde van N na 'N='
< <= > >= == ~= & | ~	BEWERKINGEN MET BOOLEANS Relationele bewerkingen Logische bewerkingen : en of niet
laplace(f,t,s) ilaplace(g,s,t)	LAPLACETRANSFORMATIES laplacetransformatie van f inverse laplacetransformatie van g

	MATRICES
$A = [a_{11}\ a_{12}\ \ldots\ a_{1m};\ a_{21}\ a_{22}\ \ldots\ a_{2m};\ \ldots;\ a_{n1}\ a_{n2}\ \ldots\ a_{nm}]$	invoer van een $n \times m$ matrix
$v = [v_1\ v_2\ \ldots\ v_n]$	invoer van een n-dimensionale rijvector
$w = [v_1; v_2; \ldots; v_n]$	invoer van een n-dimensionale kolomvector
v = w'	transponeren van een matrix
+ − .* ./ .^	elementsgewijze bewerkingen tussen matrices
A * B	matrixvermenigvuldiging
A/B = A *B^{-1}	matrixvermenigvuldiging met de inverse
A(i, j)	selecteert het element op positie i, j binnen de matrix
$A(i_1 : i_2, j)$	de elementen op rijen i_1 t.e.m. i_2, in kolom j
$A(i_1 : s : i_2, j)$	de elementen in kolom j van rijen i_1 t.e.m. i_2, om de s stappen
A(:, j)	alle elementen binnen kolom j
eye(n)	genereert de eenheidsmatrix van dimensie n
zeros(n)	de matrix van dimensie n met enkel nullen
ones(n)	de matrix van dimensie n met enkel enen
rand(m, n)	een $m \times n$ matrix met willekeurige elementen tussen 0 en 1
diag(v)	een diagonaalmatrix met de componenten van v
diag(A)	een vector met als componenten de diagonaalelementen van A .
$v_1 : s : v_2$	een vector met getallen van v_1 tot v_2 en stapgrootte s
$linspace(v_1,v_2,n)$	een vector met n equidistante getallen tussen v_1 en v_2
	VEELTERMEN
$p = [a_1\ a_2\ \ldots\ a_{n-1}\ a_n]$	invoeren van een veelterm
conv(p,q)	veeltermvermenigvuldiging
[q, r]=deconv(p1,p2)	Euclidische deling van veelterm p1 door veelterm p2 met q als quotiënt en r als rest
polyder(p)	afleiden van veelterm p
polyval(p,x)	evalueren van veelterm p in x
roots(p)	vinden van nulpunten van een veelterm

1. Basiscommando's

1.1 Invoeren van opdrachten

Wiskundige notatie	Intikken
a + b	`a+b`
a − b	`a-b`
ab	`a*b`
3xy	`3*x*y`
$\frac{a}{b}$	`a/b`
a^b	`a^b`
$\sqrt{x}$	`sqrt(x)`
π	`pi`
e	`exp(1)`
3 − 4i	`3-4*i`
sin x, arctan x, . . .	`sin(x), atan(x), ...`
e^x, ln x	`exp(x), log(x)`
log x (= ^{10}log x)	`log10(x)`
ln x	`ln(x)`
|x|	`abs(x)`
∞	`inf`

(i) Houd er rekening mee dat MATLAB 'case sensitive' is. Dat wil zeggen dat er verschil is tussen hoofd- en kleine letters. Het getal π 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 of RETURN-toets. Het antwoord verschijnt direct op het scherm. Door het commando af te sluiten met een `;` (puntkomma) wordt de uitvoer op het scherm onderdrukt.

1.2 De help-faciliteit

MATLAB beschikt over een uitgebreide help-functie. Met behulp van het commando `help` onderwerp verschijnt er een toelichting bij onderwerp. Bijvoorbeeld:

```
help elfun
```

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

1.3 Rekenen met getallen

MATLAB is in feite een zeer uitgebreide programmeerbare rekenmachine.
Voorbeeld:

```
12/3
  ans = 4
```

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 ongeveer 16 decimalen nauwkeurig (waarvan er standaard vijf op het scherm verschijnen).

Voorbeeld:

```
12/9
   ans = 1.3333
```

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, en verder zijn er de commando's `real`, `imag`, `conj` en `exp` met voor de hand liggende acties.

Standaard geeft MATLAB op het scherm 5–cijferige getallen met een vaste komma. Met behulp van het `format`–commando kun je opgeven welke notatie–vorm je wilt gebruiken. In onderstaande tabel staan verschillende opties:
In de rechterkolom staan de representaties van π, van $1.23456\ 10^{-3}$ en van $1.23456\ 10^{-6}$.

Commando	Resultaat	Voorbeeld
format short	5 cijfers, scaled fixed point	3.1416 0.0012 1.2346e–006
format short e	5 cijfers, floating point	3.1416 1.2346e–003 1.2346e–006
format long	15 cijfers	3.14159265358979 0.00000123456000 1.23456000000000e–006

1.4 Toekenningen

Als een matrix (of getal) meerdere keren gebruikt gaat worden kun je de waarde ervan toekennen aan een variabele. Je hebt hem dan als het ware een naam gegeven, en kunt hem via die naam weer oproepen.

Als voorbeeld berekenen we de eerste vier machten van de matrix $A = \begin{pmatrix} 1 & 2 & 1 \\ 2 & -1 & 2 \\ 1 & 3 & -2 \end{pmatrix}$

```
A = [1, 2, 1; 2, -1, 2; 1, 3, -2]
 A2 = A^2; A3 = A^3; A4 = A^4;
```

Vervolgens kun je de som hiervan berekenen via

```
som = A + A2 + A3 + A4
```

Opmerking: De spaties staan hier slechts voor de leesbaarheid. MATLAB negeert ze.

Opmerking: Als je niet zelf een variabele introduceert om een waarde in op te slaan, kiest MATLAB hiervoor de variable `ans`. Het is echter een goede gewoonte om de resultaten van elke berekening in een eigen variabele op te slaan.

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

`who`	opvragen van alle variabelen met een waarde;
`size(x)`	geeft de afmetingen (aantal rijen, kolommen) van `x`;
`clear x1`	verwijder de variabele x1 ;
`clear all`	verwijder de waarden van alle variabelen.
`clc`	veegt het scherm schoon (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;
```
(hiermee wordt `x` 'leeg' gemaakt)

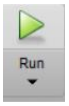
```
functie = x^2 + 4 * sin(x)
```

Het berekenen van de functiewaarde van boven bedoelde functie f in bijvoorbeeld het punt $\pi/3$ kan als volgt gebeuren:

```
x = pi/3;
y = x^2 + 4*sin(x)
```

Opmerking. Met de ↑-toets kun je voorgaande commando's terughalen en, al of niet gewijzigd, opnieuw uit laten voeren door op `Enter` te drukken. (Hiervoor hoeft de cursor niet aan het eind van de regel te staan.)
Met de `Delete`- en de `Back Space`-toets (dat is bij sommige toetsenborden de ← toets boven `Enter`) kun je symbolen wegvegen; met de ← en → rechts van de `Ctrl`-toets kun je binnen een regel van links naar rechts lopen.

Om een stukje code te schrijven met meerdere commando's, kan je een nieuw script openen met 

en het laten uitvoeren met 

1.5 Werken met Matrices

Het basiselement van MATLAB is de matrix. Speciale gevallen hiervan zijn een 1 × 1–matrix, d.w.z. een scalar, en een matrix bestaande uit slechts 1 rij of kolom, d.w.z. een vector. De grenzen van een matrix hoeven niet gedeclareerd te worden. (Ze kunnen opgevraagd worden met het commando `size`.)

Bij het elementsgewijs invoeren van een matrix worden de verschillende rij-elementen gescheiden door spaties of door komma's. De rijen worden gescheiden door punt-komma's, of door naar een nieuwe regel te gaan. De getallen worden omsloten door twee rechte haken, `[` en `]`.

De matrix $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ wordt als volgt ingevoerd:

```
A = [ 1  2  3 ; 4  5  6 ; 7  8  9]
```

of

```
A = [ 1, 2, 3 ; 4, 5, 6 ; 7, 8, 9]
```

Voorbeeld van het invoeren van een kolomvector:

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

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

`A(i,j)`	A_{ij} (het element in de i^e rij, j^e kolom)
`A(:,j)`	j^e kolom van A
`A(i,:)`	i^e rij van A
`A(k:l, m:n)`	deelmatrix van alle elementen A_{ij} $k \le i \le l, m \le j \le n$ (ofwel: k-de t.e.m. l-de rij, m-de t.e.m. n-de kolom)
`v(i:j)`	deelvector $(v_i, v_{i+1}, \ldots, v_j)$ van de vector v

En voor een vierkante matrix:

`diag(A)`	geeft de hoofddiagonaal van A (als kolomvector);
`diag(A,k)`	geeft de k-de nevendiagonaal van A (als $k < 0$, dan wordt degene onder de hoofddiagonaal genomen)

Probeer het een en ander uit:

Voer in de 3 × 4-matrix `A = [1 2 3 4; 5 6 7 8; 1 1 1 1]` in.
Voorspel en controleer wat het resultaat is van de volgende commando's:
(Merk op dat 1 van de commando's een foutmelding zal geven. Welke en waarom?)

- `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)`

Hoewel de laatste regel een voorbeeld is van 'slordig programmeren', kun je er wel duidelijk aan aflezen wat er in feite gebeurt bij een toekenning van het type `A = B`. Eerst wordt de waarde van B uitgerekend, en vervolgens wordt deze waarde (een getal, een vector of een matrix) toegekend aan de variabele A.
Het '='teken betekent hier dus iets heel anders dan 'is gelijk aan'. Je kunt `A = B` dan ook het best lezen als: 'A wordt B'.

Matrix-bewerkingen

`C=A+B`	gewone optelling van matrices
`C=A-B`	gewone verschil van matrices
`C=A*B`	gewone matrixvermenigvuldiging (bij onjuiste afmetingen volgt een foutmelding);
`C=A.*B`	elementsgewijze vermenigvuldiging (*A* en *B* moeten dezelfde afmetingen hebben);
`C=A+k`	bij elk element van *A* wordt het getal *k* opgeteld
`C=A^k`	gewone machtsverheffing (k ∈ Z), bijv. te gebruiken voor berekening van A^{-1};
`C=A.^k`	elementgewijze machtsverheffing
`C=A'`	berekent de getransponeerde A^T
`C=inv(A)`	berekent de inverse van *A*
`d=det(A)`	berekent de determinant van *A*
`d=size(A)`	berekent de afmetingen van *A*
`C=cat(2,A,B)`	2 matrices samenvoegen, kolommen naast elkaar
`C=cat(1,A,B)`	2 matrices samenvoegen, rijen onder elkaar
`C=A\B`	berekent voor een niet-singuliere, vierkante matrix *A* de unieke oplossing van het de vergelijking $AX = B$
`C=B/A`	berekent de unieke oplossing van het de vergelijking $XA = B$
`L=eig(A)`	*L* is een kolomvector met de (mogelijk complexe) eigenwaarden van *A*
`[X,D]=eig(A)`	*D* is een diagonaalmatrix met op de diagonaal de eigenwaarden van *A*, de kolommen van *X* zijn de bijbehorende eigenvectoren;
`A=diag(v)`	geeft diagonaalmatrix met diagonaal *v(1), v(2), ...,v(n)*;
`A=zeros(n)`	*A* wordt de *n*×*n* nulmatrix;
`A=zeros(n,m)`	*A* wordt de *n*×m nulmatrix;
`A=eye(n)`	*A* wordt de *n*×*n* identiteitsmatrix;
`A=ones(n,m)`	*A* wordt de *n*×*m* matrix met louter 1-en;
`A=rand(n,m)`	*A* wordt de *n*×*m* matrix met random getallen;
`x=k:n`	*x* wordt de vector *(k, k+1, k+2, ..., n);*
`x=a:h:b`	*x* wordt de vector *(a, a+h, a+2h, ..., b)*, aangenomen dat *b*–*a* deelbaar is door *h*

1.6 Vectoren

Zowel rij- als kolomvectoren kunnen in MATLAB gebruikt worden. Een rijvector kan je eenvoudig definiëren zoals hieronder:

```
u = [1 2 4]
u  =  1  2  4
```

Deze vector heeft drie componenten met als waarden 1, 2 en 4 respectievelijk. Bij een kolomvector worden de componenten door ; van elkaar gescheiden

```
v = [2;3;6]
v  =  2
      3
      6
```

Vaak wil men vectoren waarvan de componenten opeenvolgende waarden hebben. Deze zijn zeer eenvoudig te definiëren in MATLAB. Het volgende voorbeeld maakt een vector met vijf componenten:

```
w  =  1:5
w  =  1  2  3  4  5
```

De volgende vector heeft ook vijf componenten, maar de waarden hebben een interval van 2 in plaats van 1.

```
w  =  1:2:10
w  =  1  3   5   7   9
```

De componenten zowel als het interval kunnen reële getallen zijn:

```
w  =0.0:0.5:2.0
w  = 0 0.5 1.0 1.5 2.0
```

Individuele componenten van een vector kunnen opgevraagd of veranderd worden door een index te gebruiken.

```
w(2)
 ans = 0.5000
w(3) = 1
```

Het is ook mogelijk een vector te maken die bestaat uit een gedeelte van de componenten van een andere.

```
 w(2:4)
  ans  = 0.5000   1.0000   1.5000
```

Het scalair product van twee vectoren kan berekend worden met `dot`:

$$\sum_{i=1}^{n} u_i v_i$$

```
dot(u,v)
 ans = 32
```

Naast het scalair product (of inproduct) kan je ook het uitproduct van twee vectoren berekenen:

```
A  =  u .* v
A  =  6   18  12
      10  30  20
      14  42  28
```

De componenten van deze matrix A worden gegeven door $A_{ij} = u_i v_j$.

Het vectorieel product kan ook eenvoudig berekend worden met behulp van de functie cross:

```
cross(u,v)
```

```
ans  = 0  2  -1
```

Het aantal componenten van een vector kan bepaald worden met het commando `length`, terwijl de lengte van een vector bekomen wordt met het commando `norm`.

```
length([3 5 7])
ans  = 3
norm([3 5 7])
ans  = 9.1104
```

Er zijn nog een aantal functies die het vermelden waard zijn: `min`, `max`, `sum` en `prod` die respectievelijk de kleinste en grootste component, de som en het product van de vector componenten berekenen.

```
min([3  5  7])
ans  = 3
max([3  5  7])
ans  = 7
sum([3  5  7])
ans  = 15
prod([3 5 7])
ans  = 105
```

Aangezien in MATLAB het gebruik van vectoren centraal staat zijn er operatoren die componentsgewijs werken. Zo is het bijvoorbeeld mogelijk een vector w te bekomen uit de vectoren u en v zo dat $w_i = u_i v_i$ of $w_i = u_i / v_i$, respectievelijk met behulp van de operatoren '.* ' en './'.

```
u = [1  3  5];
v = [2  4  6];
u.*v
ans  =  2          12          30
u./v
ans  =  0.5000     0.7500      0.8333
```

Analoog kan je een vector v bekomen waarvan de componenten die zijn van een vector u verheven tot een zekere macht n. De operator hiervoor is '.^'.

```
[1 3 5].^3
ans  =  1   27  125
```

Merk op: het scalair product '* ' geeft een getal en de componentsgewijze vermenigvuldiging '.* ' geeft een vector.

```
[ 1   2   3 ] * [ 2   3   4 ]
???  Error  using  ==>  *
Inner  matrix  dimensions  must  agree .
[ 1   2   3 ] * [ 2 ;    3 ;   4 ]
ans  = 20
[ 1   2   3 ] . * [ 2   3   4 ]
ans  = 2           6          12
[ 1   2   3 ] . * [ 2 ;    3 ;    4 ]
???  Error  using  ==>   . *
Matrix  dimensions  must  agree .
```

2. Functies

2.1 Gekende functies toegepast op vectoren / matrices

Als je invoert

```
x = 0:0.1:pi; y = sin(x)
```

dan worden er twee variabelen `x` en `y` 'aangemaakt' met waarden
(0, 0.1, 0.2, ... , 3.0, 3.1) resp. (sin(0), sin(0.1), sin(0.2), ... , sin(3.0), sin(3.1)).
MATLAB neemt dus elementsgewijs de sinus van `x`, en slaat de 'functie' als het ware op in een tabel.
De 'elementsgewijze' matrix-operaties `.*` en `.^` zijn met name handig om 'functies' in te voeren:
Voorbeeld:

De functie $f(x) = \frac{1}{3}\,x^2 + x\sin(x)$ op het interval [0, 3] met 'stapgrootte' 0.1.

Ten eerste de *x*-waarden:

```
x = 0:0.1:3.0;
```
(dit geeft een vector van lengte 31)

Nu moeten we voor $k = 1, 2, 3, \ldots, 31$ berekenen $y(k) = f(x(k))$.
Met behulp van de `.*` en de `.^` operator lukt dit als volgt:

```
y = (1/3)*x.^2 + x.*sin(x)
```

2.2 Inline functies

Vaak echter is het nuttig zelf nieuwe functies te definiëren die niet voorzien zijn in MATLAB, hetgeen uiteraard voorzien is.
Deze eerste methode is handig als men functies wil definiëren die men enkel in de huidige sessie zal gebruiken. Voor deze methode zijn de functie-aanroepen ook het meest natuurlijk. Eerst wordt de functie gedefinieerd:

```
g = @(x)  exp(0.5*x.^2)/sqrt(2*pi)
g =  exp(0.5*x*x)/sqrt(2*pi)
```

Het @-teken geeft aan dat hier een functie gedefinieerd wordt, waarbij *x* de naam is van het argument. De functie kan geëvalueerd worden zoals elke gewone MATLAB functie:

```
g(0.0)
  ans  =  0.3989
g(1.0)
  ans  =0.6577
```

Merk op dat de componentsgewijze operator '.^' gebruikt werd in de definitie van 'gaussian'. Hoewel de functie de juiste resultaten zou opleveren met de operator '^' in situaties zoals hierboven (i.e. wanneer het argument een scalaire grootheid is) zou er een foutmelding optreden indien we de functie in vectorcontext zouden gebruiken:

```
g2  =  @(x) exp(0.5*x^2)/ sqrt(2*pi) ;
g2(0.0)
 ans  = 0.3989
g2([0.0 1.0 2.0])
 ??  Error  using  ==>  inline eval
 Error  in   inline expression ==> exp(0.5*x^2)/sqrt(2*pi)
 ???  Error  using  ==>  Matrix must be square .
```

Daarentegen is dit voor de functie gedefinieerd met de operator '.^' geen probleem:

```
g([0 1])
  [0.3989   0.6577]
```

Met behulp van deze methode kan men ook functies met meerdere argumenten definiëren. Men moet eenvoudig alle namen van argumenten opsommen in de definitie:

```
d  =  @(x1,x2) norm(x2-x1)
```

De aanroep is weer eenvoudig:

```
d([0 0] ,[3 4])
 ans  = 5
d([1 2 3],[3 4 5])
 ans  = 3.4641
```

2.3 Maple-files

De tweede manier om functies te maken bestaat erin *een bestand aan te maken* (een zogenaamde MATLAB-file) met daarin de definitie van de functie. Deze methode is vooral handig wanneer men functies wil declareren die men vaak nodig heeft. Hieronder staat de inhoud van de Maple-file '`gaussian.m`' die de functie 'gaussian' definieert:

```
function y = gaussian(x)
  y = exp(0.5*x.^2)/sqrt(2*pi);
```

De eerste regel is de functie declaratie, wat aangegeven wordt door 'function'. De naam van die functie is in dit geval 'gaussian'. De functie heeft een argument met als naam 'x'. Het resultaat van de berekening zal toegekend worden aan de variabele 'y'. De definitie van de functie i.e. hetgeen berekend wordt, staat op de tweede regel. De waarde wordt berekend met de variabele 'x' en toegekend aan de variabele 'y' overeenkomstig de functiedeclaratie. Deze functie kan in MATLAB gebruikt worden zoals een functie die inline gedefinieerd was:

```
gaussian(0.0)
ans  = 0.3989
guassian(1.0)
ans  =0.2420
```

De volgende MATLAB-file '`distance.m`' definieert een functie met twee vectoren als argumenten heeft en de afstand tussen de punten berekent voorgesteld door deze vectoren:

```
function d= distance(x1 ,x2)
     d  =  norm( x2    x1 ) ;
```

De declaratie vermeldt dat de functie met naam 'distance' twee argumenten heeft, namelijk 'x1' en 'x2' en dat het resultaat toegekend zal worden aan de variabele 'd' De aanroep van de functie wordt geïllustreerd in het volgende stukje code waarbij de eerste aanroep vectoren van lengte 2, de tweede die van lengte 3 gebruikt.

```
distance([0 0],[3 4])
 ans  = 5
distance([1 2 3],[3 4 5])
 ans  = 3.4641
```

Merk op dat je slechts 1 functie per MATLAB-file kan definiëren en dat deze dezelfde naam heeft als het bestand. Je kan een functie op deze manier ook enkel definiëren in een bestand, niet in het 'Command Window'.

De variabelen `result` en `x` in de regel `function result = functienaam(x)` zijn matrices. `x` heet de input, `result` de output.
In plaats van een enkele variabele, kun je ook meerdere variabelen als input en/of output nemen. De aanhef wordt dan

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

2.4 Taylorontwikkeling

Met

```
syms x
taylor(f,x,a,'Order',n)
```

kan je een Taylorontwikkeling maken van de functie *f* rond *x=a* waarbij termen meegenomen worden t.e.m. $(x-a)^{n-1}$.
Als je ook de lokale fout wil berekenen voor *x=b* die wordt gemaakt door *f(b)* te benaderen door de functiewaarde van de afgebroken taylorontwikkeling in *x=a*, dan kan dit via:

```
syms xx
f=@(x) …;
t=taylor(f(xx),xx,a,'Order',n);
vpa(f(1)-subs(t,xx,b))
```

Hierbij werd een inline-functie gebruikt voor *f*. Merk op hoe op een verschillende manier functiewaardes worden berekend voor *f* en *t*, aangezien *t* geen inline-functie is. Het commando

```
subs(t,x,a)
```

vervangt de variabele *x* door de waarde *a* in de uitdrukking *t*.

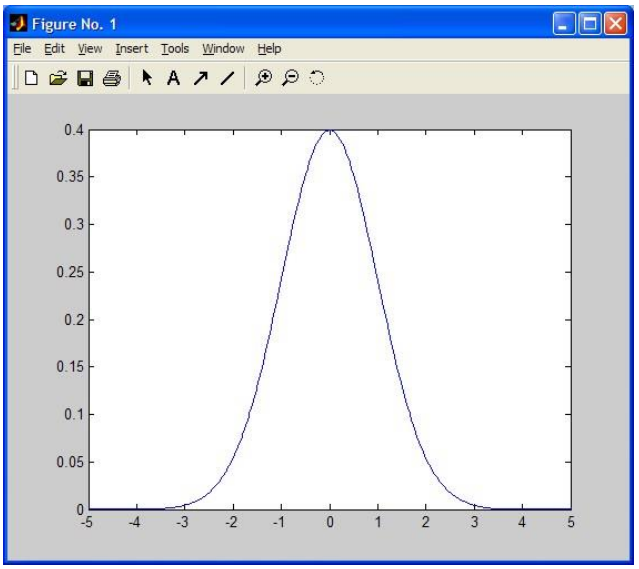
3. Plots

3.1 2D plots

Functie plots

Het is erg eenvoudig plots van functies te maken met behulp van MATLAB. Als voorbeeld wordt de MATLAB functie ‘gaussian’ gebruikt.

```
fplot(@gaussian ,[-5.0,5.0])
```



Figuur 1: MATLAB venster voor figuren

Je kan **fplot** ook gebruiken om symbolische functies mee te tekenen. Vergeet daarbij niet om eerst x als een symbolische variabele vast te leggen.

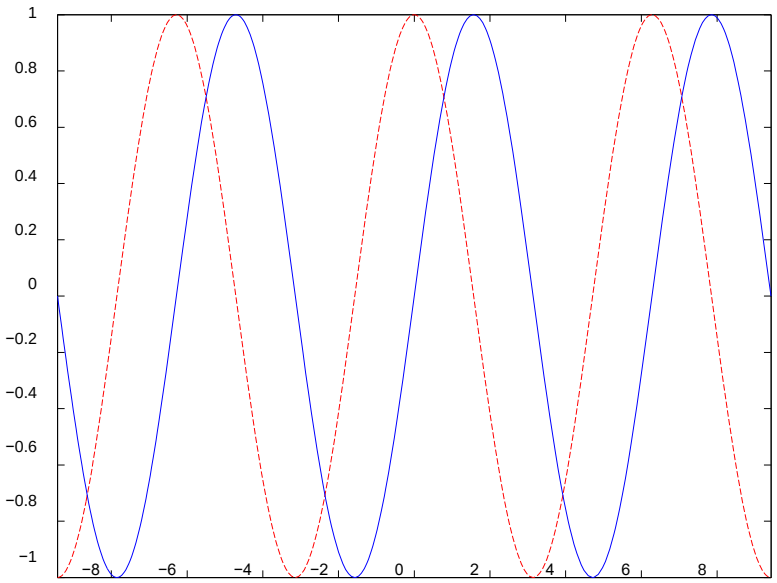
```
syms x
fplot(x^2+2*x-5,[-5,5])
```

Het is eenvoudig om meerdere functies op één figuur te zetten met behulp van de parameter ‘hold’. Als deze de waarde ‘0’ heeft (default) dan overschrijft elk plot commando de vorige uitvoer in het venster. De waarde op ‘on’ zetten daarentegen laat toe in hetzelfde venster te blijven werken.

```
fplot(@sin,[-3*pi,3*pi])
hold on
fplot(@cos,[-3*pi,3*pi],'--r')
hold off
```

Hierbij geeft de string ‘r’ aan dat de plot van de cosinus in het rood ‘r’ en met een gebroken lijn ‘--‘ getoond moet worden. Het resultaat is te zien in Figuur 2. Vergeet niet telkens ‘hold’ weer op ‘off’ te zetten zodra je niet meer in hetzelfde venster wil werken.

De ‘hold’ waarde werkt voor elk soort plot, ook in drie dimensies; een voorbeeld hiervan is te zien in Figuur 8.



Figuur 2: meerdere functies op dezelfde plot: sin x (volle lijn), cos x (gebroken lijn)

Data plots

Vaak wordt de data die gevisualiseerd moet worden niet beschreven door een functie, maar is ze het resultaat van een berekening of een experimentele meting. Ook hiervoor biedt MATLAB erg veel mogelijkheden. Het volgende voorbeeld illustreert hoe x-waarden tegen y-waarden uitgezet kunnen worden.

```
x = 0.0 : 5.0 : 100;
y = sqrt(x) ;
plot(x,y,'-.or')
```

De parameter '-.or' verwijst naar een dash-dot line (-.), cirkelvormige (o) bij de punten en een rode kleur. Andere keuzes zijn:

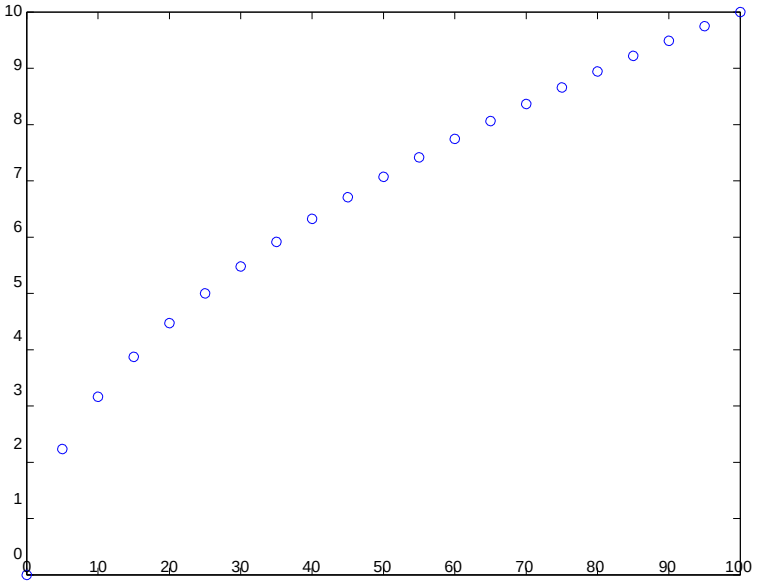
r = Red	'-' = volle lijn	'*' = asterisk als aanduiding van de punten
g = Green	'--' = stippellijn	'+' = plusteken als aanduiding van de punten
b = Blue	'.' = puntlijn	'-.' = punt-streeplijn
k = Black		
y = Yellow		
w = White		

axis equal : dit commando zorgt ervoor dat de indeling van de X-as en Y-as dezelfde zijn.

Met

```
t  =   0 : 0.1 : 10 ;
x = sqrt(t);
y = t.^2;
plot(x,y)
```

wordt een parameterkromme $\begin{cases} x = \sqrt{t} \\ y = t^2 \end{cases}$, $t \in [0,10]$ getekend.



Figuur 3: Data plot

Impliciete plot

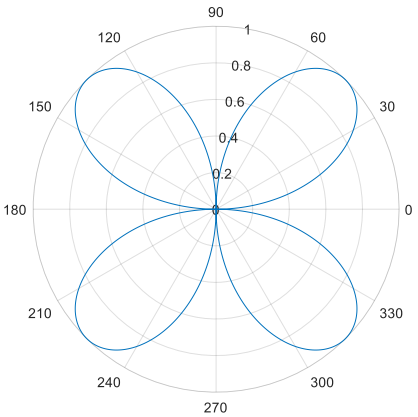
Van een impliciete kromme die symbolisch beschreven wordt, kan ook een voorstelling gemaakt worden.

```
syms x y
eqn = x^2+y^2 == 4;
fimplicit(eqn, [-2,2])
```

Tekenen van poolkrommen

Van een poolkromme kan een voorstelling gemaakt worden via 'polarplot'. Merk op dat MATLAB geen rekening houdt met de voorwaarde dat $r \geq 0$!

```
theta = 0:0.01:2*pi;
rho = sin(2*theta);
polarplot(theta,rho)
```



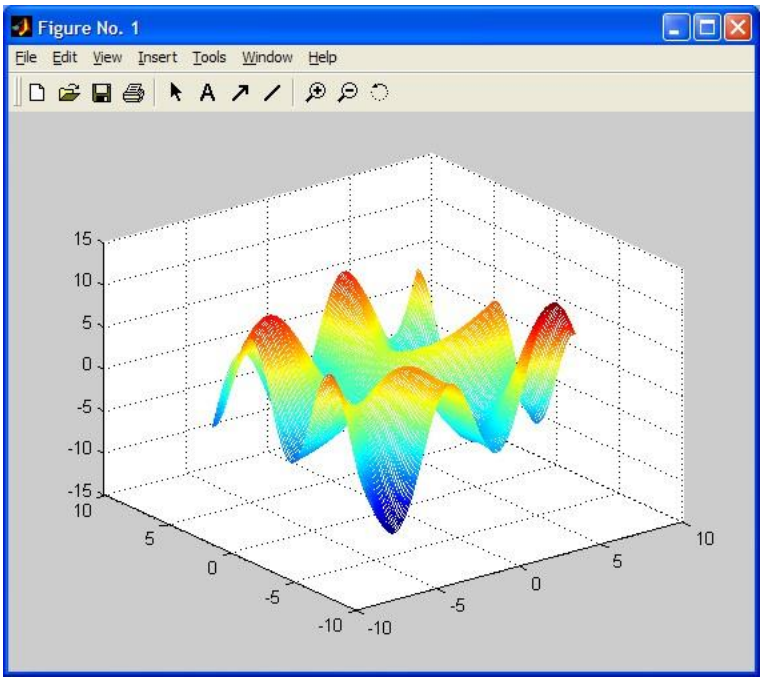
3.2 3D plots

Oppervlakteplots

Beschouw de functie *f(x; y) = y sin x + x cos 2y*, hiervan kunnen we een figuur maken als volgt:

```
V1 = -2*pi : 0.1 : 2*pi;
V2 = -3*pi : 0.1 : 3*pi;
[X,Y] = meshgrid(V1,V2);
Z = sin(X).*Y+X.*cos(Y);
mesh(X,Y,Z)
```

Het resultaat wordt getoond in Figuur 4. De functie 'meshgrid' berekent aan de hand van de vectoren 'V1' en 'V2' twee vierkante matrices 'X' en 'Y' die de componenten van 'V1', respectievelijk 'V2' als rijen dan wel als kolommen bevatten. Met behulp van 'X' en 'Y' kunnen we de vierkante matrix 'Z' berekenen met behulp van componentsgewijs werkende operatoren. Deze matrix bevat nu de waarden f(x, y) en kan getoond worden met behulp van de functie 'mesh'. In de MATLAB GUI is het erg eenvoudig aspecten van de plot aan te passen of deze te roteren in drie dimensies.



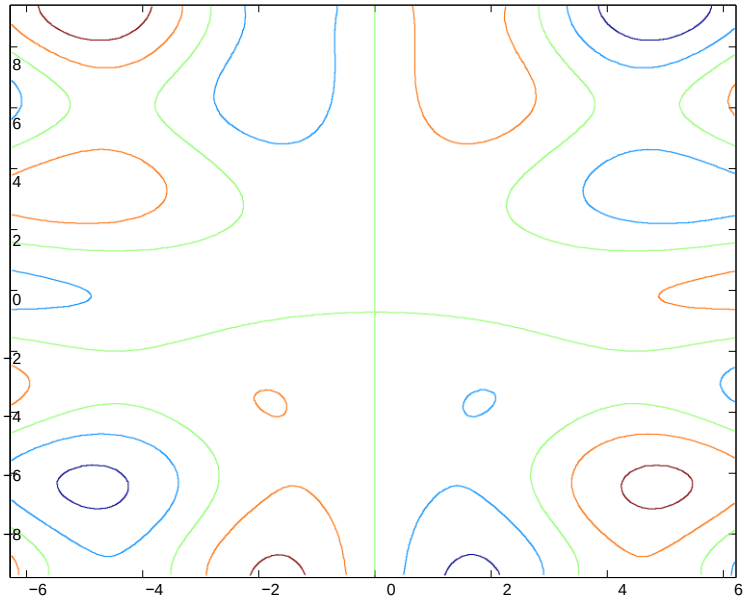
Figuur 4: 3D plot van de functie *f(x; y) = y sin x + x cos 2y*

Contourplot

Een ander type plot dat vaak van pas komt voor drie dimensionale data (en soms gemakkelijker te analyseren is dan een 3D plot) is een contourplot. De onderstaande code genereert het resultaat getoond in Figuur 5.

```
contour(X,Y,Z)
```

De matrices 'X', 'Y' en 'Z' zijn dezelfde die gebruikt werden voor 'mesh'. Het aantal contourlijnen kan meegegeven worden als optionele parameter (bijvoorbeeld contour(X, Y, Z, 20)' zal 20 contourlijnen gebruiken).



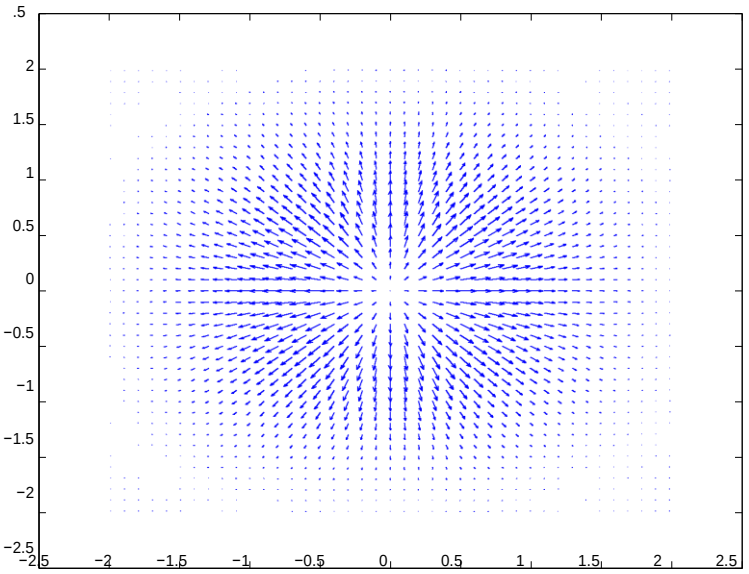
Figuur 5: contourplot van de functie f(x; y) = y sin x + x cos y

Het is ook mogelijk beide types plot te combineren, een voorbeeld hiervan is te zien in Figuur 5 waarbij opnieuw gebruik gemaakt werd van de 'hold' variabele.

Vectorveldplot

MATLAB beschikt over een derde type plot, 'quiver' dat nuttig is voor het visualiseren van vectorvelden. Dit kan geïllustreerd worden aan de hand van het vectorveld: $\vec{F}(x,y) = \left(\overrightarrow{e_x}\sin x + \overrightarrow{e_y}\sin y\right)\exp(-x^2 - y^2)$
Dit veld wordt getoond in Figuur 6.

```
[X, Y] = meshgrid(-2.0 : 0.1 : 2.0) ;
U  =  sin(X).*exp(-(X.^2 + Y.^2)) ;
V  =  sin(Y).*exp(-(X.^2 + Y.^2)) ;
quiver (X, Y, U, V)
```



Figuur 6: vectorveld plot van $\vec{F}(x,y) = (\overrightarrow{e_x}\sin x + \overrightarrow{e_y}\sin y)e^{-x^2-y^2}$

4. Differentiatie, integratie via vectoren, limieten

4.1 Differentiatie

De afgeleide kan men numeriek of symbolisch bekomen.

Numeriek: Wanneer men de afgeleide wil tekenen van de functie ‘sin’ in het interval [0; pi] dan kan men als volgt te werk gaan. Definieer een (groot) aantal x-waarden en bijbehorende functiewaarden:

```
x  =  0.0:0.01:pi;
y  =  sin(x);
```

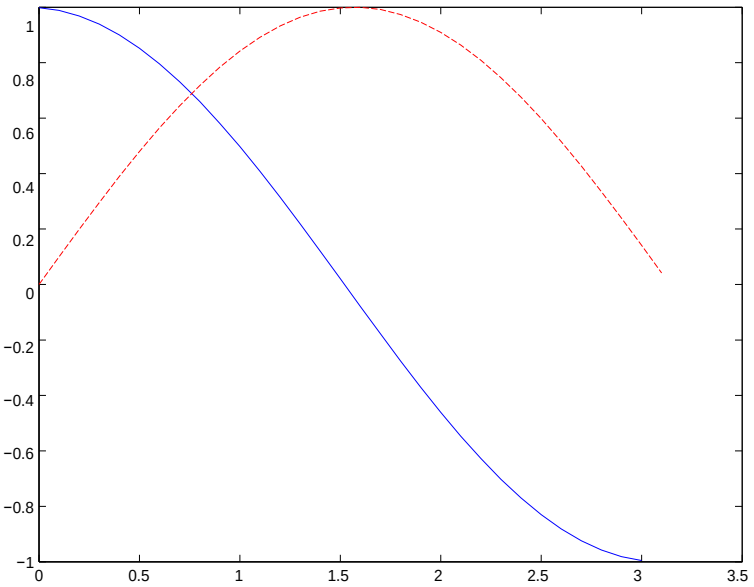
Nu kan men gebruik maken van de MATLAB functie ‘diff’ die het verschil tussen opeenvolgende elementen van een vector berekent:

```
dy  = diff(y)./diff(x);
```

Deze vector ‘dy’ heeft één component minder dan de vector ‘y’ zodat we voor een plot ook enkel de eerste (`length(x)`-1) componenten van de vector ‘x’ nodig hebben:

```
dx  =  x(1:length(x)-1);
```

Figuur 7 toont de plot gemaakt met behulp van ‘plot(dx, dy)’.



Figuur 7: afgeleide functie (volle lijn) van de functie ‘sin’ (gebroken lijn) in het interval [0; 3]

Symbolisch:

gebruik het commando `diff` en vergeet niet je variabele vooraf als symbolisch vast te leggen.

```
syms x
f = sin(x^2)
diff(f)
```

4.2 Integratie

Net zoals bij het afleiden van functies kan de integratie numeriek en symbolisch uitgevoerd worden.

4.2.1 Enkelvoudige integralen

Numeriek kunnen enkel bepaalde integralen kunnen berekend worden. MATLAB heeft een functie `quad` om een bepaalde integraal van een functie te berekenen met behulp van het algoritme van Simpson.

```
quad(@sin,0.0,pi)
 ans  = 2.000
```

Symbolisch heeft MATLAB het commando '**int**' om een (on)bepaalde integraal van een functie te berekenen.

```
syms x
int(sin(x^2), 0, pi)
  ans = (fresnels((2*pi)^(1/2))*(2*pi)^(1/2))/2
```

Numeriek kan dit beter ingeschat worden via:

```
g=@(x) sin(x.^2)
quad(g,0.0,pi)
  ans = 0.7727
```

4.2.2 Meervoudige integralen

Er zijn ook functies voorzien voor het berekenen van dubbele en drievoudige integralen, ‘dblquad’ en ‘triplequad’. ‘dblquad’ wordt als volgt gebruikt:

```
f = @(x,y)(sin(x)*y + x*cos(y))
dblquad(f,0.0,2*pi,0.0,2*pi)
 ans  =  4.7607e-10
```

4.3 Limieten

Limieten van een functie f waarbij x nadert naar a worden berekend via het commando `limit(f,a), limit(f,a,'left')` en `limit(f,a,'right')` waarbij de laatste twee respectievelijk de linker- en rechterlimiet berekenen.

5. Minimalisatie

Een vaak voorkomend probleem is het zoeken van het minimum van een functie. MATLAB maakt dit relatief eenvoudig met behulp van de functies ‘fminbnd’ voor één en ‘fminsearch’ voor meerdere veranderlijken. Hierbij wordt numeriek gewerkt.

5.1 Functies in één veranderlijke

De naam geeft aan dat het minimum van de functie berekend zal worden in een door de gebruiker gespecifieerd interval, met andere woorden, er moet van te voren iets bekend zijn over de aard van de functie als men betrouwbare resultaten wil bekomen. Neem als voorbeeld de onderstaande functie waarvoor we het minimum zoeken in het interval [0; 5]:

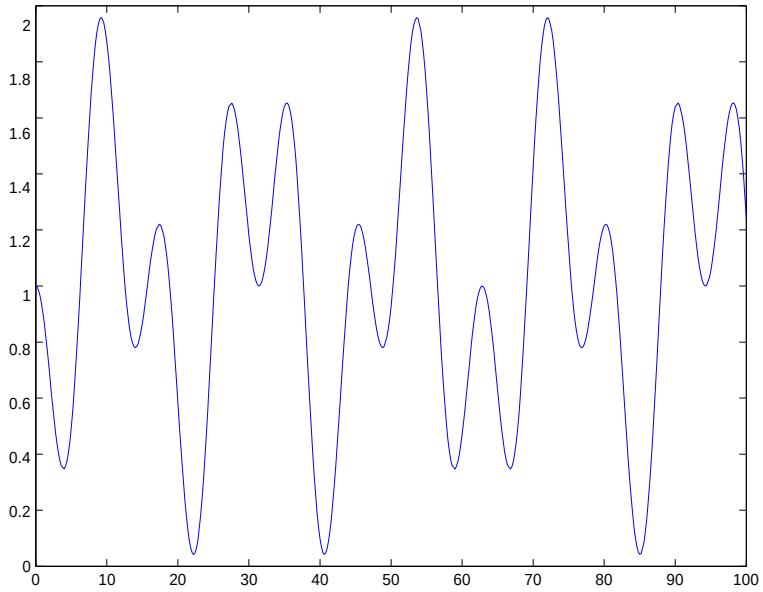
```
f  = @(x) (1-sin(0.2*x).*sin(0.5*x));
```

De positie van het minimum wordt gegeven door:

```
[x0,val] =  fminbnd(f,0.0,5.0)

 x0=3.9079
 Val = 0.3467
```

Hierbij geeft ‘x0’ de positie en ‘val’ de waarde van het minimum van de functie ‘f’. Figuur 8 toont echter dat dit geen globaal minimum is van de functie ‘f’. Een globaal minimum (dat overigens niet uniek is) bevindt zich bij 22.2, hetgeen benadrukt dat functie-onderzoek nuttig is.

```
[x0,val] = fminbnd(f,0.0,50.0)
x0 =22.2117
val =0.0421
```



Figuur 8: De functie *f(x)= 1-sin(0.2 x) sin(0.5 x)* heeft een zowel lokale als globale minima.

5. 2 Functies in meerdere veranderlijken

Optimalisatie van functies in meerdere veranderlijken verloopt zeer analoog, hiervoor is de functie `fminsearch' voorzien. Bijvoorbeeld: $f(x,y) = (x-1)^2 + (y+1)^2 + 2$

```
f  =  @(x)  ((x(1)-1)^2 + (x(2)+1)^2+2) ;
```

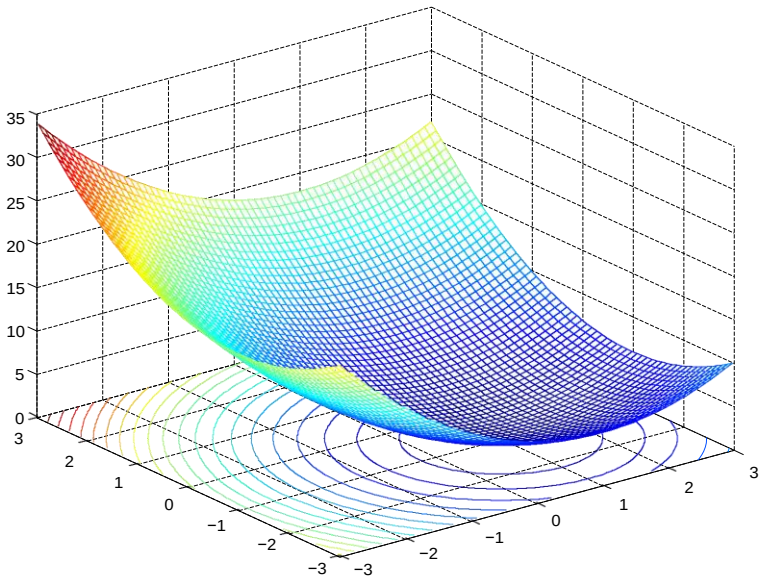
De functie wordt getoond in Figuur 9 met behulp van de volgende commando's:

```
[X,Y]  =  meshgrid(-3.0 : 0.1 : 3.0);
Z  =  (X-1).^2 + (Y+1).^2 + 2;
mesh(X,Y,Z)
hold   on
contour (X, Y,  Z ,20)
hold   off
```

Om het minimum van deze functie te zoeken moeten we een startpunt specifiëren. De keuze hiervan is kritiek indien de functie meerdere lokale minima heeft. Voor dit zeer eenvoudige voorbeeld kunnen we bijvoorbeeld (0, 0) kiezen.

```
[x0,val]= fminsearch(f,[0.0,0.0])
x0  = 1.0000         1.0000
Val = 2.0000
```

Het minimum bevindt zich inderdaad in het punt (1, 1) en heeft de waarde f(1, 1) = 2.



Figuur 9: de functie $f(x, y) = (x -1)^2 + (y + 1)^2 + 2$

6. Vergelijkingen

Een ander veel voorkomend probleem is het oplossen van vergelijkingen. Ook hiervoor levert MATLAB de nodige ondersteuning. Twee soorten problemen worden hier behandeld: (1) het oplossen van een vergelijking in 1 onbekende en (2) het oplossen van stelsels lineaire vergelijkingen.

6. 1 Vergelijkingen in één onbekende

MATLAB heeft geen functies om een vergelijking rechtstreeks op te lossen, maar wel om nulpunten van functies te zoeken. Uiteraard komt dit op hetzelfde neer, immers:

$$f(x) = g(x) \text{ , } F(x) = 0$$

waar $F(x) = f(x)- g(x)$. Beschouw bij wijze van voorbeeld de volgende niet-lineaire vergelijking:

$$\tanh(2x) = x$$

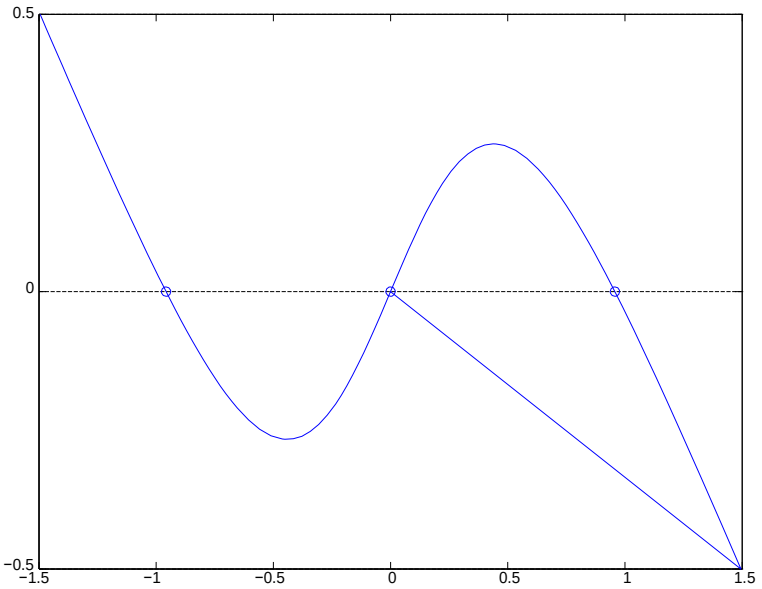
De strikt positieve oplossing kan gevonden worden met behulp van:

```
f = @(x)(tanh(2*x)-x) ;
fzero (f,2.0)
 ans  = 0.9575
```

Hierbij is de waarde 2.0 een startwaarde.

```
fzero (f,0.5)
  ans  = 2.3942 e-21
```

Hoewel 0 een oplossing is van de vergelijking is het niet degene die gevraagd wordt. Figuur 10 toont de drie oplossingen.



Figuur 10: De functie *tanh(2x)- x* heeft drie wortels bij *x=0.96, x* = 0 en *x*=- *0.96.*

6.2 Stelsels van lineaire vergelijkingen

Veelal wordt de oplossing gezocht van een stelsel van *n* lineaire vergelijkingen met *n* onbekenden. Dit is het meest eenvoudige geval, het wordt hier geïllustreerd aan de hand van een voorbeeld. Beschouw het volgende stelsel van vergelijkingen: $\begin{cases} 2\,x_1 + 3\,x_2 = 5 \\ x_1 - x_2 = 0 \end{cases}$

Dit stelsel kan herschreven worden in termen van matrices:

$$A \quad x = b$$

waarbij de matrix A en de vectoren x en b gegeven worden door: $A = \begin{pmatrix} 2 & 3 \\ 1 & -1 \end{pmatrix}, x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, b = \begin{pmatrix} 5 \\ 0 \end{pmatrix}$,

De oplossing van dit stelsel wordt in MATLAB als volgt berekend: de matrix A en de vector b worden gedefinieerd,

```
A  =   [2 3; 1 1]
A  =
      2     3
      1     1
b  =   [ 5 ;  0 ]
b  =
      5
      0
```

en vervolgens wordt de oplossing berekend.

```
x  =  A\b
x  =
  1
  1
```

Het is natuurlijk mogelijk dat de vergelijkingen niet onafhankelijk zijn zoals bijvoorbeeld het volgende stelsel: : $\begin{cases} 2\,x_1 + 3\,x_2 = 5 \\ 2\frac{x_1}{3} + x_2 = 9 \end{cases}$

MATLAB detecteert dat de determinant van de matrix A zeer dicht bij nul ligt en deze A dus singulier is.

```
A  =   [ 2   3 ;   2/3   1 ]
A  =
    2.0000          3.0000
    0.6667          1.0000
    b  =   [ 5 ;    9 ]
b  =
     5
     9
Warning :   Matrix   is  singular  to working   precision.
( Type   " warning   off   MATLAB: singular Matrix"   to
        suppress this   warning . )
ans  =
   Inf
   Inf
```

6.3 Stelsels niet-lineaire vergelijkingen

De functionaliteit die beschreven wordt in deze paragraaf maakt deel uit van de Optimization Toolbox. Dit is een product dat niet standaard deel uitmaakt van MATLAB, zodat het mogelijk niet geïnstalleerd is op de computer waar je mee werkt.

Naast stelsels van lineaire vergelijkingen zijn er ook veel problemen die aanleiding geven tot een stelsel van vergelijkingen die niet lineair zijn. Beschouw bijvoorbeeld het zoeken van de snijpunten van een cirkel en een rechte: de vergelijking van de

cirkel is een tweedegraadsvergelijking, dus niet lineair: : $\begin{cases} x^2 + (y-1)^2 = 2 \\ 3x - y = 1 \end{cases}$

De oplossing van dit stelsel kunnen we bekomen met behulp van de functie 'fsolve', die net zoals 'fzero' de nulpunten zoekt van functies. We definiëren dus eerst een functie die als resultaat een kolomvector heeft met de waarden van de functies:

```
F =@(x)[x(1).^2+(x(2)-1).^2-2;    3*x(1)-x(2)-1];
```

Het argument van deze functie moet een vector zijn met twee componenten, bijvoorbeeld:

```
F([1;1])
ans  =
    -1
     1
```

Analoog aan 'fzero' wordt aan de functie 'fsolve' een kolomvector met startwaarden als parameter gegeven worden.
Het stelsel kan nu opgelost worden met:

```
 fsolve(F,[1;1])
Optimization terminated successfully:
ans  =
    1.0000
    2.0000
```

Het punt (1, 2) is dus een snijpunt van de cirkel en de rechte. Zoals Figuur 11 aangeeft hebben de cirkel en de rechte twee snijpunten, het tweede kunnen we bekomen door andere startwaarden te gebruiken:

```
fsolve(F,[1;-1])
Optimization terminated successfully:
ans  =
     0.200
    -0.400
```

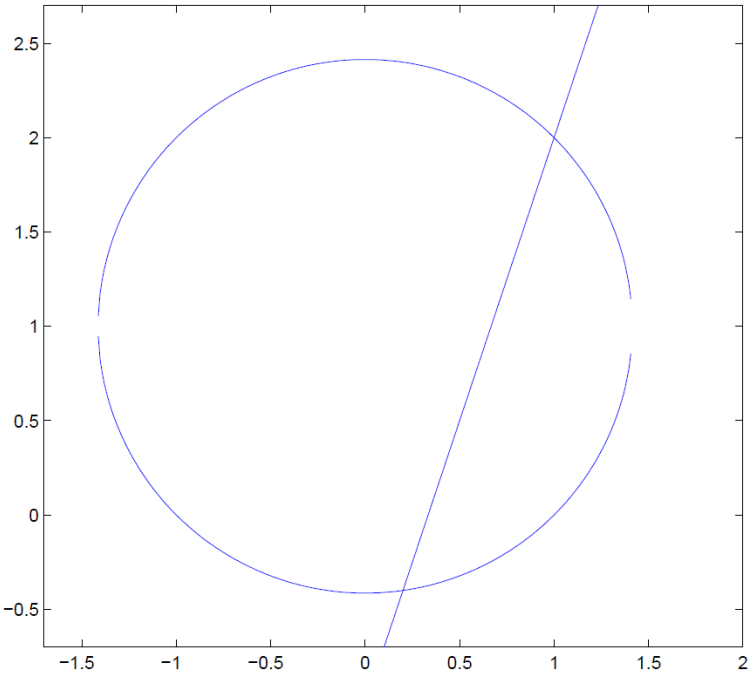
Het tweede snijpunt is dus (0.2, -0.4).

Merk op dat voorzichtigheid geboden is, beschouw de volgende vergelijking die opnieuw een cirkel en een rechte voorstellen: $\begin{cases} x^2 + y^2 = 1 \\ x - y = 3 \end{cases}$

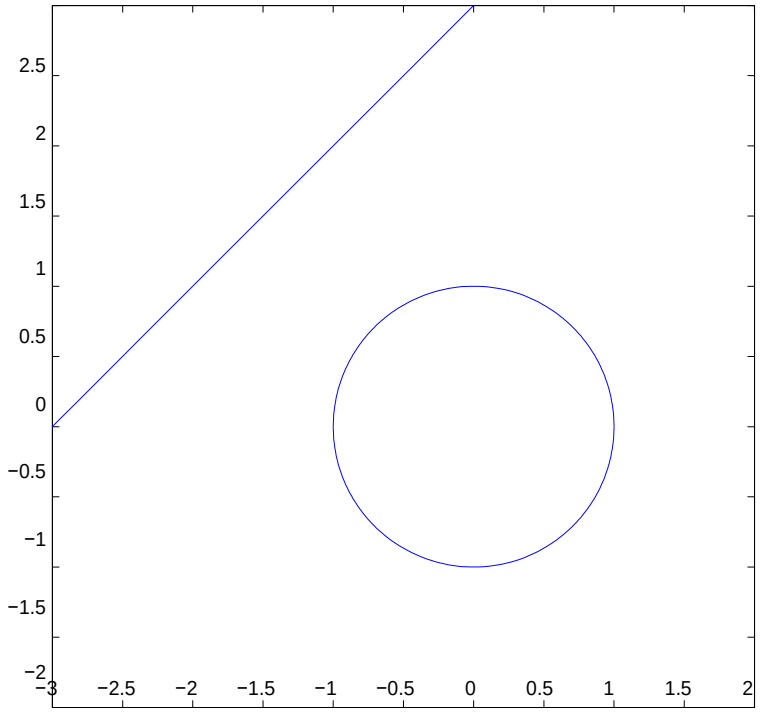
Het is duidelijk uit Figuur 12 dat deze cirkel en rechte geen snijpunt hebben. De 'fsolve' geeft nu de volgende waarschuwing:

```
F = @(x)[(x(1).^2+x(2).^2-1);  (x(1)-x(2)+3)];
options  =  optimset('Display','notify');
fsolve(F,[1;1],options)
Optimizer oot :appears  to   be converging to   a minimum that is not
a Sum of  squares of the function values is > sqrt(options.TolFun) .
Try   again with a new starting point
ans  =
                                    0.9086
                                    0.9086
```

Het resultaat is dus niet betrouwbaar.



Figuur 11: De cirkel en de rechte met respectievelijk vergelijking $x^2 + y^2 = 1$ en $x - y = 3$ hebben twee snijpunten.



Figuur 12: De cirkel en de rechte met respectievelijk vergelijking $x^2 + y^2 = 1$ en $y = x + 3$ hebben geen snijpunten.

Wanneer de vergelijking(en) symbolisch behandeld worden, kunnen we het commando **solve** gebruiken. Vergeet daarbij niet eerst x,… als symbolische variabele vast te leggen.

```
syms x
solve(x^2-5*x+4)==0)

 syms x y z
[x,y,z] = solve(z==x^2, y==-x,-5*x+4*z==1)
```

7. Programmeren in MATLAB

MATLAB beschikt voor een eigen programmeertaal die je kan gebruiken om repetitieve taken eenvoudiger te maken. Ze is echter compleet, met andere woorden, ze is even krachtig als Java of C. Uiteraard is ze vooral en bij uitstek geschikt om problemen op te lossen die te maken hebben met wiskundige modellen, data-analyse, -manipulatie en -visualisatie.

7.1 Voorwaardelijke opdrachten

Beschouw de functie die als resultaat de waarde 1 geeft wanneer haar argument groter is dan 0, de waarde 0 in het andere geval. De MATLAB-file die deze functie implementeert ziet er als volgt uit:

```
function value  =    heaviside(x)
    if x  >  0
       value = 1;
    else
       value = 0
    end
 end
```

Een meer alledaags voorbeeld is een functie die, gegeven een jaartal, de waar 1 geeft wanneer dit een schrikkeljaar is, 0 in het andere geval.

```
function value = leapyear(n)
if   rem(n,100)==0  &  ~rem(n,400)==0
   value = 0 ;
       else if rem(n ,4)  ==  0
          value  =   1 ;
          value  =   0 ;
       else end
       end
```

7.1.1 Relationele operatoren

Naast de evidente relationele operatoren <, <=, > en >= zijn er nog twee operatoren die respectievelijk gelijkheid en ongelijkheid testen: == en ~=.

Merk op dat de toekenningsoperator = en de gelijkheid == een totaal andere rol spelen. De eerste kent de waarde die rechts staat toe aan de variabele links, de tweede vergelijkt de linker- en de rechteroperand.

7.1.2 Logische operatoren

In de eerste test hebben we gebruik gemaakt van twee logische operatoren: & stelt de logische en, ~ de logische niet.
In MATLAB stelt 0 de logische waarde 'vals' voor, alles wat niet gelijk is aan 0 heeft de logische waarde 'waar'.

7.2 Herhalingsopdrachten

7.2.1 for-lussen

Stel bij wijze van voorbeeld dat we een 5×5-matrix A willen definiëren waarvan de componenten gegeven worden door $A_{ij} = 2^i3^j$. Beide onderstaande MATLAB-codes doen dit:

```
for i=1:5
       for j=1:5
       A(i,j) = 2^i*3^j;
     end
   end
A
A  =
       6  18    54   162   486
      12  36   108   324   972
      24  72   216   648  1944
      48 144   432  1296  3888
      96 288   864  2592  7776
```

```
v1  =  (3*ones(1,5)).^(1:5)
v1  =  3    9    27   81    243
v2  =  (2*ones(1,5)).^(1:5)
v2  =  2    4    8   16   32
A  =  v2.'*v1
A  =
             6   18    54   162   486
            12   36   108   324   972
            24   72   216   648  1944
            48  144   432  1296  3888
            96  288   864  2592  7776
```

Dit voorbeeld laat meteen ook zien dat for-lussen genest kunnen worden. De volgende code toont dat we de index 'i' van de for-lus met andere waarden dan 1 kunnen verhogen:

```
for   i = 10:2:1
      sqrt(i)
end
ans  = 3.1623
ans  = 2.8284
ans  = 2.4495
ans  = 2
ans  = 1.4142
```

7.2.2 while-lussen

Het tweede type herhalingsopdracht wordt gestuurd door een voorwaarde. De volgende functie berekent de grootste gemene deler van twee natuurlijke getallen met behulp van de methode van Euclides:

```
function value = ggdi (m,n)
    while  m  ~=  n
         if   m >  n
             m = m-n;
         else
             n = n-m
     end;
     value = m;
end
```

Kom je er tijdens het rekenen achter dat er iets misgaat, (met name bij een while-loop ligt het gevaar van een ‘oneindige’ loop op de loer), dan kun je het rekenproces stopzetten met de toetsen [Ctrl]+C of [Ctrl]+[Break].

Handleiding Statistiek met MATLAB

Bij statistiek moeten we dikwijls werken met grote hoeveelheden data.
Die kunnen we importeren met de knop 'Import Data' in het tabblad HOME.
De data kan in txt-formaat of dat-formaat staan, bijvoorbeeld mijnfile.txt waarin de variabele `x` staat. Vanaf dan kunnen analyses uitgevoerd worden op `x` die wordt aangesproken met `mijnfile.x`

1. Beschrijvende statistiek

mean(x) mean(x,2)	Gemiddelde van de waarden in vector x. Is x een matrix dan is het resultaat een vector met de gemiddeldes per kolom. Met mean(x,2) wordt het gemiddelde per rij berekend.
median(x)	Mediaan van de waarden in vector x
mode(x)	Modus van x (per kolom indien x een matrix is)
iqr(x)	Interquartile Range van de waarden in vector x (=lengte doos van de boxplot)
var(x)	Variantie van de waarden in vector x
std(x)	Standaarddeviatie van de waarden in vector x
range(x)	Interval waarin de waarden in vector x liggen
prctile(x,p)	Percentielen van de waarden in vector x volgens de procenten vermeld in p
boxplot(x)	Boxplot van de waarden in vector x
boxplot([x1,x2])	Boxplot van x1 en van x2 naast elkaar op 1 figuur
hist(x), hist(x,n)	Histogram van de waarden in vector x, n stelt het aantal bins voor
scatter(x,y)	Scatterplot van vectoren x (met x-coördinaten) en y (met y-coördinaten)
randn(n,m)	Genereert een n×m matrix met per kolom een steekproef uit N(0,1)

Voorbeeld: Bereken het gemiddelde van elke rij en kolom van de matrix $A = \begin{pmatrix} 0 & 1 & 1 \\ 2 & 3 & 2 \\ 1 & 3 & 2 \\ 4 & 2 & 2 \end{pmatrix}$

Bereken eveneens de modus van alle waarden uit A (met behulp van een matrixbewerking op A).

```
A = [0 1 1; 2 3 2; 1 3 2; 4 2 2];
a=mean(A)
b=mean(A,2)
c= mode([A(:,1);A(:,2);A(:,3)])
      a = 1.7500    2.2500    1.7500
      b = 0.6667
          2.3333
          2.0000
          2.6667
      c = 2
```

Voorbeeld: Bereken op een efficiënte manier de kwartielen van 2 steekproeven genomen uit een normale verdeling waarvan de eerste bestaat uit 100 waarden uit N(4,1) en de tweede bestaat uit 200 waarden uit N(6,0.5).

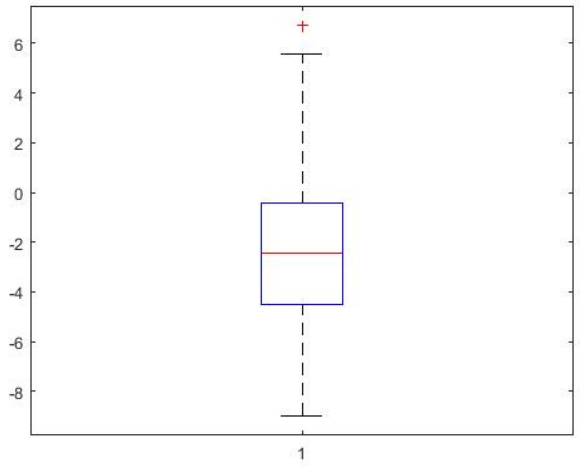
```
x = [normrnd(4,1,1,100) normrnd(6,0.5,1,200)];
p = 100*(0:0.25:1);
y = prctile(x,p);
z = [p; y]
```

Voorbeeld: Maak een steekproef van 10 elementen gelegen tussen 0 en 553.

```
numbers = unidrnd(553,1,10)
      numbers = 293   372   5   213   37   231   380   326   515   468
```

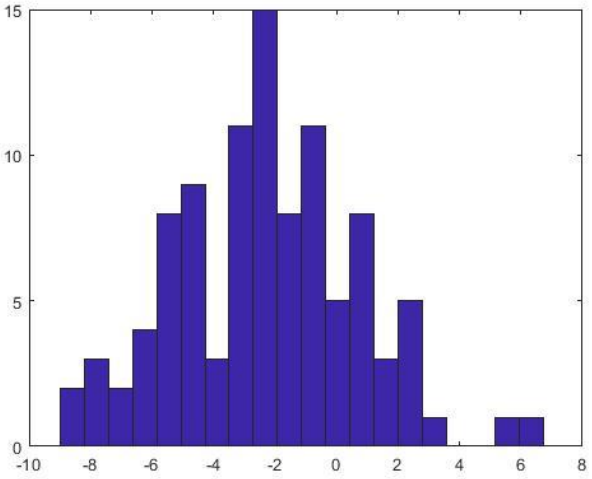
Voorbeeld: Maak een boxplot van steekproef van 100 elementen uit N(-2,3).

```
numbers = normrnd(-2,3,1,100);
boxplot(numbers)
```



Voorbeeld: Maak een histogram met 20 bins van steekproef van 100 elementen uit N(-2,3).

```
numbers = normrnd(-2,3,1,100);
hist(numbers,20)
```



2. Theoretische verdelingsfuncties

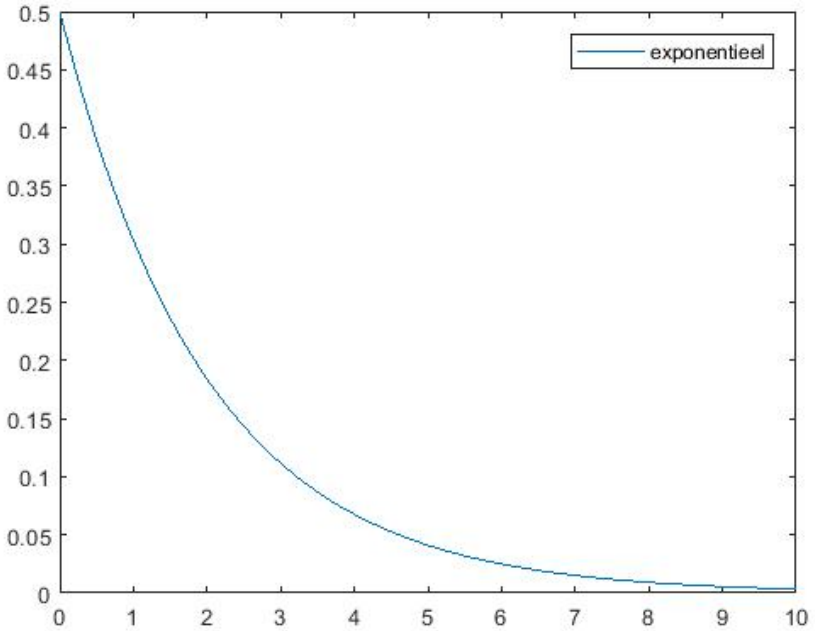
*dist*pdf(x,*par*)	de dichtheidsfunctie (=probability density function) van de distributie *dist* met parameters *par* Bijvoorbeeld: N(μ,σ) → normpdf(x, μ,σ) binomiaal(n,p) → binopdf(x,n,p) poisson(λ)→poisspdf(x,λ) exponentieel(λ)→exppdf(x,λ) t(n d.f.) →tpdf(x,n) χ^2(n d.f.)→chi2pdf(x,n) F(n1,n2 d.f.)→fpdf(x,n1,n2)
*dist*cdf(x,*par*)	de cumulatieve verdelingsfunctie (=cumulative density function) van de distributie *dist* met parameters *par* en grenswaarde x
*dist*inv(x,*par*)	de inverse waarde van de cumulatieve distributie van de distributie *dist* met parameters *par* en bijhorende kans x
*dist*rnd(*par*)	Genereert randomwaarden met *dist* als verdelingsfunctie

Voorbeeld: Vind een interval waarin 95% van de waarden van een N(0,1) verdeling liggen

```
x = norminv([0.025 0.975],0,1)
     x = -1.9600  1.9600
```

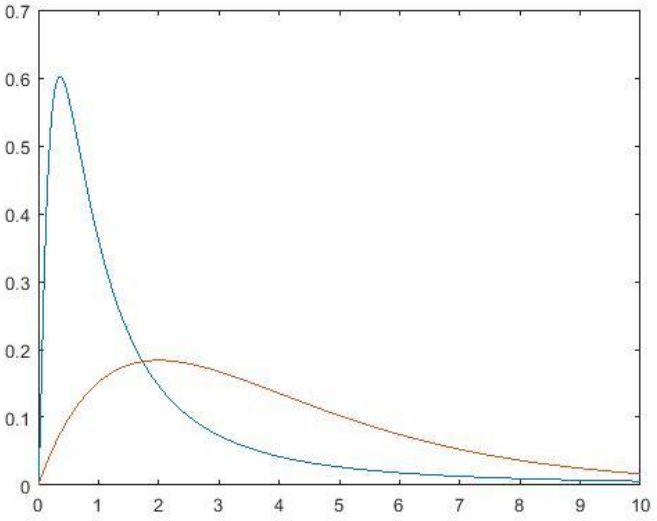
Voorbeeld: Teken de dichtheidsfunctie van exponentiële verdeling met gemiddelde 2. Voeg een legende toe met 'Tools' in het menu van de figuur.

```
x = 0:0.1:10;
y = exppdf(x,2);
plot(x,y)
```



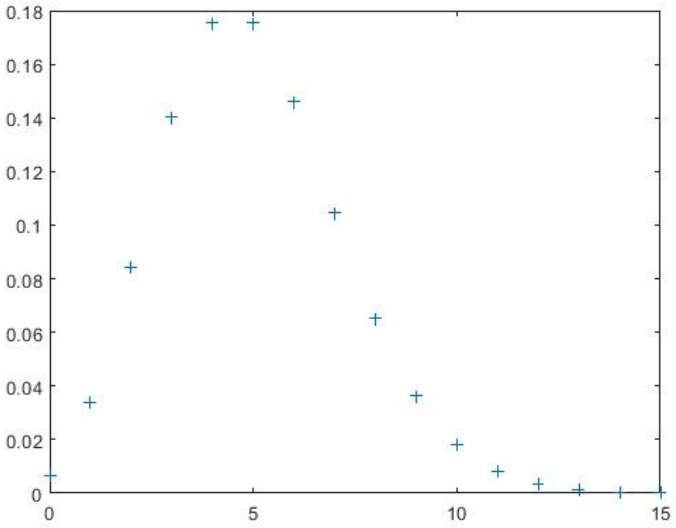
Voorbeeld: Teken de dichtheidsfunctie van F-verdeling met vrijheidsgraden 5 (teller) en 3 (noemer) samen met de dichtheidsfunctie van een χ^2-verdeling met 4 vrijheidsgraden.

```
x = 0:0.01:10;
y = fpdf(x,5,3);
plot(x,y)
hold on
y = chi2pdf(x,4);
plot(x,y)
```



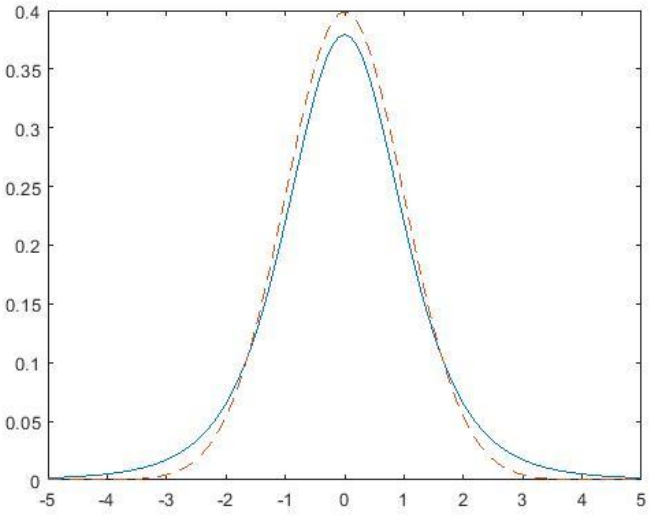
Voorbeeld: Teken de kansfunctie van de poissonverdeling met parameter λ=5 met het '+'-teken als symbool.

```
x = 0:15;
y = poisspdf(x,5);
plot(x,y,'+')
```



Voorbeeld: Maak een tekening waarop de dichtheidsfunctie van een t-distributie met 5 vrijheidsgraden (volle lijn) samen met de dichtheidsfunctie van een genormeerd normale verdeling (stippellijn).

```
x = -5:0.1:5;
y = tpdf(x,5);
z = normpdf(x,0,1);
plot(x,y,'-',x,z,'--')
```



3. Normaliteit testen

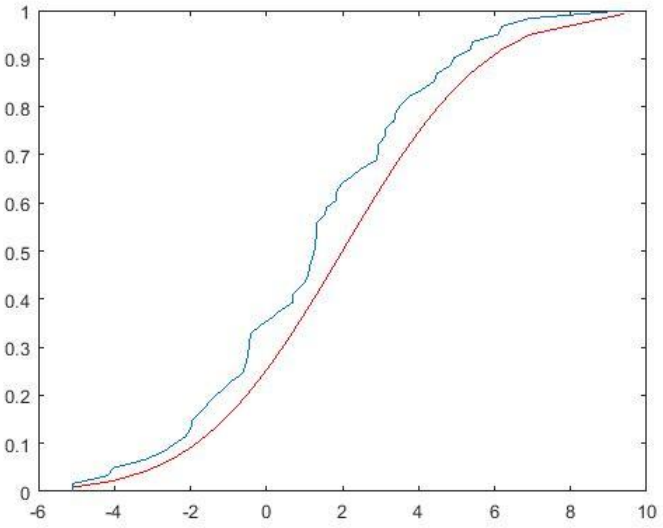
[h,p,D]=kstest(x,*options*)	Kolmogorov-Smirnov test om na te gaan of N(0,1) als aanwezig is (h=0: aanvaarden, h=1: verwerpen) p stelt het significantieniveau voor van deze test D stelt de KS-test veranderlijke voor *options* kan zijn • *'Alpha',0.01* indien afgeweken wordt van 95% betrouwbaarheid • *'CDF',test_cdf* indien afgeweken wordt van N(0,1)
[f,x]=ecdf(y)	f bevat de waarden van de empirische cumulatieve distributiefunctie voor y geëvalueerd voor de waarden uit de vector x
makedist('*dist*','*par*',par_value)	Definieert een distributie van het type *dist* met parameter *par* die de waarde par_value aanneemt.

Voorbeeld: Genereer een dataset van 61 waarnemingen uit N(2,3) voor $x \in [-1,5]$, vermeerderd met een normaal verdeelde ruis met σ=0.1. Maak een tekening van de cumulatieve distributiefunctie verbonden met deze waarden, samen met de cdf van N(2,3) (in rood). Test met een betrouwbaarheid van 99% dat de meetwaarden voorzien van de ruis uit N(2,3) komen.

```
x = -1:0.1:5;
y = normrnd(2,3,1,61);
noise = normrnd(0,0.1,1,61);
```

```
[f,x]=ecdf(y+noise);
plot(x,f);
hold on;
plot(x,normcdf(x,2,3),'r-');
test_cdf = makedist('Normal', 'mu',2, 'sigma',3);
[h,p,D] = kstest(y+noise, 'CDF',test_cdf, 'Alpha',0.01)
```

```
h = 0
p =  0.2709

D = 0.1252
```



4. t-test

[h,p,ci,stats] = ttest(x,a) [h,p,ci] = ttest(x,a,'Alpha',0.01) `[h,p,ci, stats]=ttest(x,a, 'Tail', 'left')`	t-test met H_0:μ=a (h=1: verwerpen, h=0: aanvaarden), ci is het betrouwbaarheidsinterval voor μ stats geeft informatie over de t-test: de toetsingsgrootheid, de vrijheidsgraden en de standaarddeviatie. Als optie kan α vermeld worden indien dit niet 0.05 is. Als optie kan je vermelden of je een linkszijdige t-test wil uitvoeren ('right' voor een rechtszijdige t-test).
[h,p,ci] = ttest2(x1,x2)	t-test met H_0:$\mu_1 = \mu_2$ (h=1: verwerpen, h=0: aanvaarden), ci is het betrouwbaarheidsinterval voor $\mu_1 - \mu_2$
`p = vartestn([x1, x2], 'TestType', 'LeveneAbsolute', 'Display', 'off')`	Levene-test: met H_0: de varianties van steekproeven zijn gelijk. x1 en x2 moeten kolomvectoren zijn.

Voorbeeld: Voer een t-test uit om na te gaan of de gemiddelde prijs gelijk is aan €1.15. Je baseert je op een steekproef met waarden in de kolom `price`.

```
[h,pvalue,ci] = ttest(price,1.15)
```

```
h = 1
pvalue = 4.9517e-04
ci = 1.1675 1.2025
```

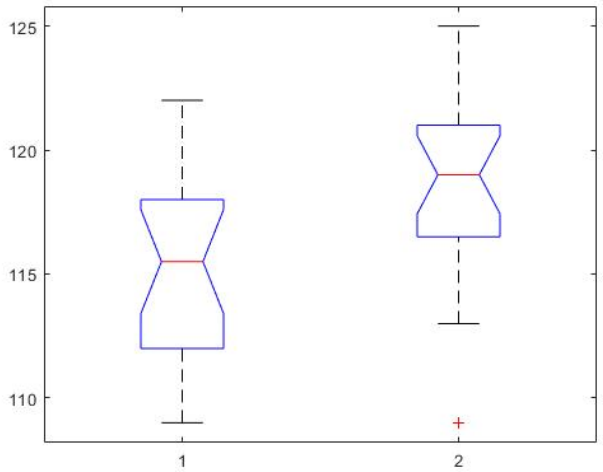
The boolean h is 1, wat betekent dat de nulhypothese die stelt dat μ=0.15 verworpen wordt met significantieniveau 0.05. Dat is ook te zien aan de p-waarde en aan het betrouwbaarheidsinterval dat 1.15 niet bevat.

Voorbeeld: Voer een t-test uit om na te gaan of de gemiddelde prijzen in januari en in februari gelijk zijn. Je baseert je op 2 steekproeven waarvan de kolommen van `prijzen` de waarden bevat. Maak tevens een vergelijkende boxplot.

```
prijzen = [119 118;117 115;115 115;116 122;112 118;121 121;115 120;122 122;
          116 120;118 113;109 120;112 123;119 121;112 109;117 117;113 117;
          114 120;109 116;109 118;118 125];
[h,sig,ci,stats] = ttest2(prijzen(:,1),prijzen(:,2))
```



```
h = 1
sig = 0.0083
ci = -5.7845 -0.9155
stats =
    tstat: -2.7857
       df: 38
       sd: 3.8029
boxplot(prijzen,1)
```



The boolean h is 1, wat betekent dat de nulhypothese die stelt dat $\mu_1 = \mu_2$ verworpen wordt met significantieniveau 0.05. Dat is ook te zien aan de p-waarde die kleiner is dan 0.05 en aan het betrouwbaarheidsinterval voor het verschil van de gemiddeldes, dat 0 niet bevat.

Om na te gaan of de variaties gelijk zijn:

```
p = vartestn([prijzen(:,1),prijzen(:,2)], 'TestType', 'LeveneAbsolute',
'Display', 'off')
```

p = 0.7206

p=0.7206> 0.05 , wat betekent dat de prijzen in januari en februari gelijke variantie hebben met 95% betrouwbaarheid.

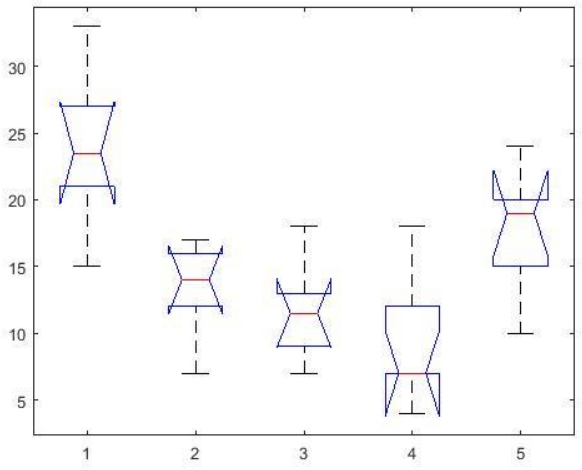
5. One-way anova

[p,table,stats]=anova1(A)	Levert de p-waarde + ANOVA-tabel + vergelijkende boxplot van een ANOVA uitgevoerd op de kolommen van matrix A (Vul aan met NaN indien bij ongebalanceerde steekproeven, m.a.w. als de steekproefgroottes verschillen)
multcompare(stats) multcompare(stats,'Alpha',k)	Levert de p-waardes + vergelijkende plot bij de paarsgewijze vergelijkingen na de ANOVA Als α de waarde k aanneemt i.p.v. de standaardwaarde van 5%

Voorbeeld: Ga na of (en waar) er een verschil is qua gemiddeldes bij de vijf groepen waarvan matrix $M = \begin{pmatrix} 24 & 14 & 11 & 7 & 19 \\ 15 & 7 & 9 & 7 & 24 \\ 21 & 12 & 7 & 4 & 19 \\ 27 & 17 & 13 & 7 & 15 \\ 33 & 14 & 12 & 12 & 10 \\ 23 & 16 & 18 & 18 & 20 \end{pmatrix}$ per kolom een steekproef bevat. Maak een volledig ANOVA.

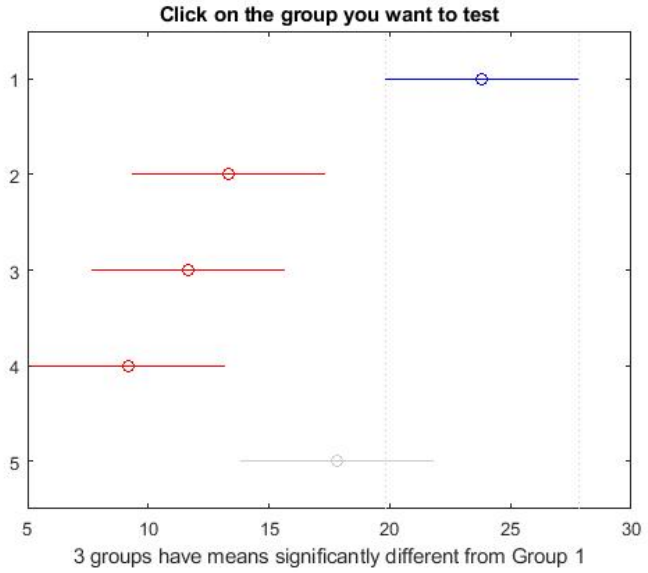
```
M=[24 14 11  7 19 ; 15  7  9  7 24 ; 21 12  7  4 19 ; 27 17 13  7 15 ;
   33 14 12 12 10 ; 23 16 18 18 20];
[p,table,stats] = anova1(M)
multcompare(stats)
```

```
p = 1.1971e-04

Source      SS      df      MS       F      Prob>F
-----------------------------------------------------
Columns     803       4   200.75    9.01   0.0001
Error      557.17    25    22.287
Total     1360.17    29
```




```
ans =

    1.0000    2.0000    2.4953   10.5000   18.5047    0.0059
    1.0000    3.0000    4.1619   12.1667   20.1714    0.0013
    1.0000    4.0000    6.6619   14.6667   22.6714    0.0001
    1.0000    5.0000   -2.0047    6.0000   14.0047    0.2119
    2.0000    3.0000   -6.3381    1.6667    9.6714    0.9719
    2.0000    4.0000   -3.8381    4.1667   12.1714    0.5544
    2.0000    5.0000  -12.5047   -4.5000    3.5047    0.4806
    3.0000    4.0000   -5.5047    2.5000   10.5047    0.8876
    3.0000    5.0000  -14.1714   -6.1667    1.8381    0.1905
    4.0000    5.0000  -16.6714   -8.6667   -0.6619    0.0292
```

De eerste twee kolommen vertellen welke groepen vergeleken worden. De vierde kolom toont het verschil tussen de geschatte groepsgemiddelden. De derde en vijfde kolom tonen de onder- en bovengrens van het 95% betrouwbaarheidsinterval voor het verschil van populatiegemiddelden. De zesde kolom bevat de *p*-waarde voor de nulhypothese die stelt dat de gemiddelde van de 2 groepen gelijk zijn.



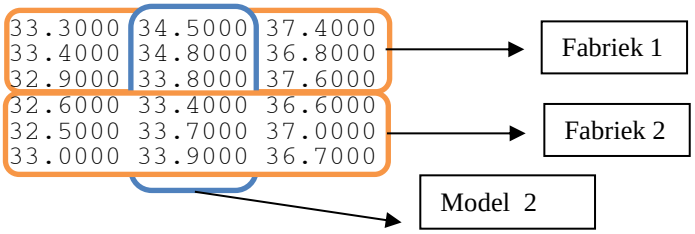
Hier worden 5 groepen vergeleken (te zien op de vertical as). De basisgroep bij deze voorstelling is groep 1. De groepen bij wie het gemiddelde significant afwijkt van dat bij groep 1, zijn rood gekleurd. Je kan de resultaten voor de vergelijkingen met een andere basisgroep bekomen door verticaal een andere groep te selecteren.

6. Two-way anova

[p,table,stats]=anova2(A,*rep*)	Levert de p-waarde + ANOVA-tabel + vergelijkende boxplot van een two-way ANOVA uitgevoerd op matrix A waarvan de kolommen de verschillende niveaus van factor 1 voorstellen en waarvan de rijen de verschilende niveaus van factor 2 voorstellen. Herhalingen worden onder elkaar geplaatst. Het aantal herhalingen kan via een extra optie *rep* meegegeven worden.
multcompare(stats)	Levert de p-waardes + vergelijkende plot bij de paarsgewijze vergelijkingen van de kolommen na de two-way ANOVA
multcompare(stats,'Estimate','row')	Levert de p-waardes + vergelijkende plot bij de paarsgewijze vergelijkingen van de rijen na de two-way ANOVA

Voorbeeld: Onderzoek de invloed van het automodel en van de productiefabriek op de kwaliteitsscore van auto's. De steekproef bestaat uit waarden voor 18 auto's (9 per fabriek en 6 per model) die terug te vinden is in `mileage`. Er is per kolom een model van auto. De eerste 3 rijen verwijzen naar de eerste fabriek en de laatste 3 rijen verwijzen naar de tweede fabriek.

```
load mileage      % of via import data
mileage
      mileage =
```




```
[p,table,stats] = anova2(mileage,3)
```

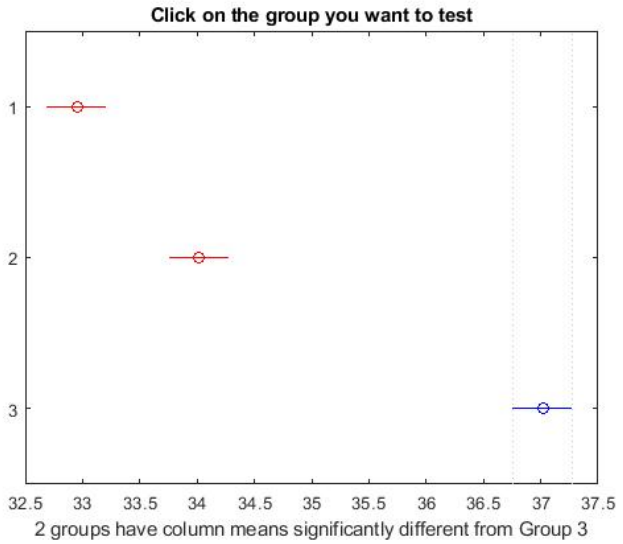
Source	SS	df	MS	F	Prob>F
Columns	53.3511	2	26.6756	234.22	0
Rows	1.445	1	1.445	12.69	0.0039
Interaction	0.04	2	0.02	0.18	0.8411
Error	1.3667	12	0.1139		
Total	56.2028	17			

```
p = 0.0000 0.0039 0.8411
```

Hieruit blijkt dat er geen significant interactie-effect is (95% betrouwbaarheid), maar wel een significant verschil toe te schrijven aan de rijen en aan de kolommen (95% betrouwbaarheid) wegens p-waardes die kleiner zijn dan 0.05. Verdere analyse is nodig.

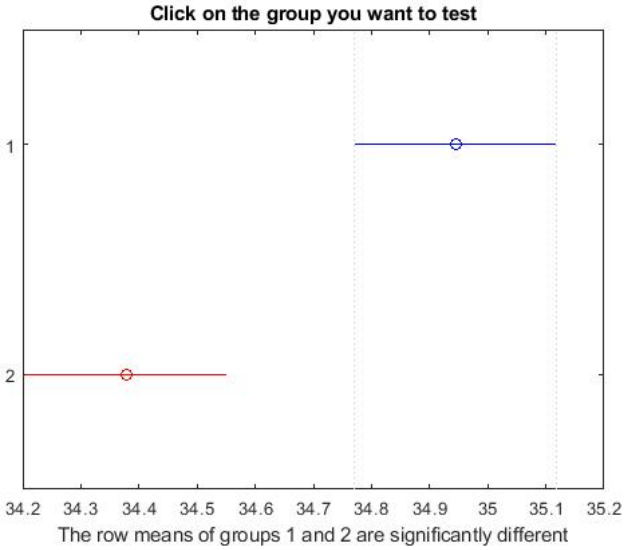
```
multcompare(stats)
```

```
ans =
1.0000    2.0000   -1.5865   -1.0667   -0.5469    0.0004
1.0000    3.0000   -4.5865   -4.0667   -3.5469    0.0000
2.0000    3.0000   -3.5198   -3.0000   -2.4802    0.0000
```




```
multcompare(stats,'Estimate','row')
```

```
ans =
1.0000    2.0000    0.2200    0.5667    0.9133    0.0039
```



Hieruit blijkt welke groepen wel of niet gelijke gemiddeldes hebben (95% betrouwbaarheid). Blijkbaar zijn alle gemiddeldes verschillend bij paarsgewijze vergelijkingen van de automodellen en bij het vergelijken van de fabrieken.

7. Pearson Correlatiecoëfficiënt

[rho,p]= corrcoef(x,y)	Berekent de correlatiecoëfficiënt rho tussen x en y, samen met de p-waarde van de test met nulhypothese dat ρ=0.

Voorbeeld: Beschouw 3 steekproeven van 50 waarnemingen uit een genormeerd normale verdeling. Beschouw tevens een nieuwe steekproef waarbij de i-de waarneming de som is van de i-de waarnemingen uit de 3 andere steekproeven. Bereken de correlatiecoëfficiënten en de p-waardes bij de paarsgewijze vergelijken van deze 4 steekproeven. Verklaar het resultaat en de betekenis van de p-waardes.


```
A = randn(50,3);
A(:,4) = sum(A,2);
[R,P] = corrcoef(A)
```

```
R =
    1.0000    0.1135    0.0879    0.7314
    0.1135    1.0000   -0.1451    0.5082
    0.0879   -0.1451    1.0000    0.5199
    0.7314    0.5082    0.5199    1.0000


P =
    1.0000    0.4325    0.5438    0.0000
    0.4325    1.0000    0.3146    0.0002
    0.5438    0.3146    1.0000    0.0001
    0.0000    0.0002    0.0001    1.0000
```

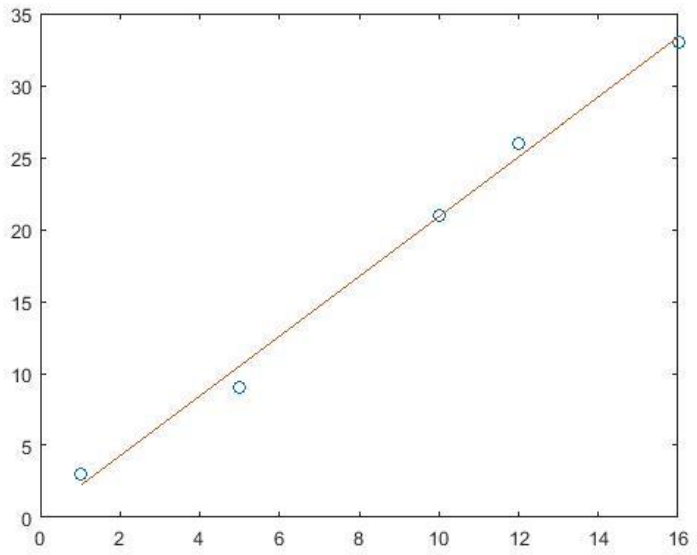
8. Lineaire regressie

polyfit(x,y,k)	Modelleren met een veelterm rond datapunten met een veelterm van graad k (bij *k=1*, mag *k* weggelaten worden)

[b,bint,r,rint,stats] = regress(y,x)	Voert een lineaire meervoudige regressie uit. • b bevat de schatters voor de regressiecoëffciënten • bint bevat de 95% betrouwbaarheidsintervallen voor de schatters uit b • r bevat de residues • rint bevat de 95% betrouwbaarheidsintervallen voor de residues • stats bevat de R^2-waarde, de F-statistiek, de p-waarde en een schatting van de variantiefout
lm=fitlm(x,y) lm=fitlm(x,y,'Intercept',false) anova(lm)	Zoekt het best passende lineaire model op basis van de vectoren x en y (met of zonder intercept) Een anova-tabel ter evaluatie van het bekomen model
[x,y]=meshgrid(xint,yint)	Maakt een 2×2 rooster van coördinaten op basis van de opsommingen xint en yint van resp. x- en y-waarden
scatter3(x,y,z)	Scatterplot in 3 dimensies op basis van de vectoren x, y en z
mesh(x,y,z)	Wireframe in 3 dimensies op basis van de vectoren x, y en z

Voorbeeld: Voer een lineaire regressie uit op de punten (10,21), (16,33), (5,9), (12,26),(1,3) Maak een grafische voorstelling en beoordeel de kwaliteit van de regressierechte.


```
A = [10 21; 16 33; 5 9; 12 26; 1 3];
X = [ones(size(A,1),1) A(:,1)];
y = A(:,2);
[b,bint,r,rint,stats] = regress(y,X);
f = b(1)+b(2)*A(:,1);
plot(A(:,1),y,'o', A(:,1),f,'-')
```



De matrix X heeft een extra kolom met telkens 1 als waarde. Deze is noodzakelijk voor het schatten van het *y*-intercept van het lineair model.

```
b = 0.1153
```

```
    2.0778

bint =
   -3.0678    3.2983
    1.7675    2.3881

r =
    0.1066
   -0.3602
   -1.5043
    0.9510
    0.8069

rint =

   -5.2633    5.4766
   -4.1591    3.4386
   -3.6401    0.6314
   -3.3205    5.2225
   -1.8818    3.4956

stats = 0.9934  454.0096    0.0002    1.3199
```

Het regressiemodel dat wordt voorgesteld is $y = 0.1153 + 2.0778\,x$ en wordt aanvaard als een waardevol regressiemodel met 95% betrouwbaarheid omdat

- R^2=0.9934 wat dicht bij 1igt
- De p-waarde bij de F-teststatistiek slechts 0.0002 is, dus <0.05
- Het 95% betrouwbaarheidsinterval voor b_1 geen nul bevat

Aangezien het 95% betrouwbaarheidsinterval voor b_0 wel de nul bevat, is er nog een verbetering mogelijk met $y = 2.2082\,x$

```
A = [10 21; 16 33; 5 9; 12 26; 1 3];
X = A(:,1);
y = A(:,2);
[b,bint,r,rint,stats] = regress(y,X)
```



```
    b = 2.0875

    bint = 1.9667    2.2082

    stats = 0.9934       NaN       NaN    0.9943
```

Opmerking: Merk op dat de F-waarde niet vermeld wordt bij een model zonder intercept. Die kan wel bekomen worden bij analoge resultaten bekomen met `lm=fitlm(X,Y)` en `lm=fitlm(X,Y,'Intercept',false)` respectievelijk gevolgd door `anova(lm)`.

Voorbeeld: Voer een lineaire regressie uit op data waarbij de te verklaren variabele (IQ van een persoon) terug te vinden is in de derde kolom van de matrix A en de waarden voor de 2 verklarende variabelen (omtrek buik, gewicht van een persoon) in de eerste twee kolommen van A zitten. Maak ook een grafische voorstelling en beoordeel de kwaliteit van de regressie.

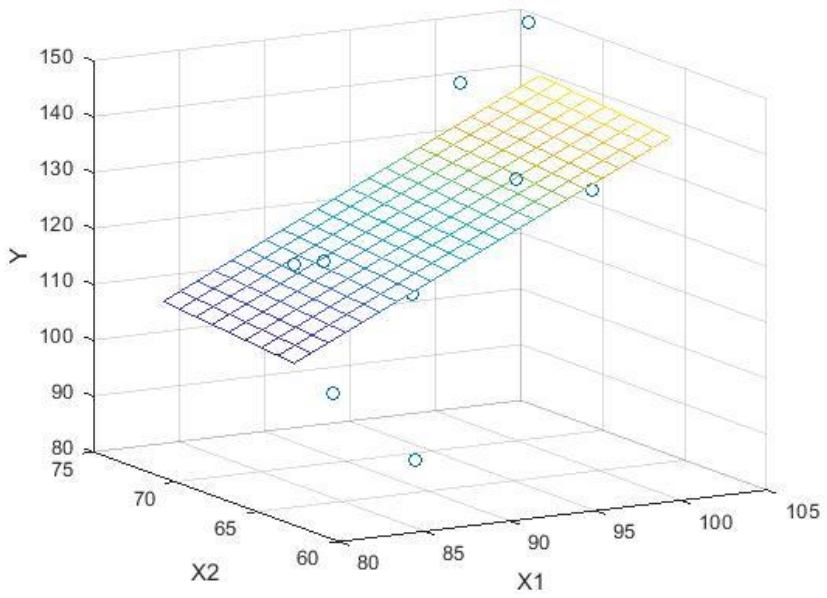
```
A = [81.69 64.5 124;103.84 73.3 150;96.54 68.8 128;95.15 65.0 134;92.88 69.0
110; 99.13 64.5 131;85.43 66.0 98;90.49 66.3 84; 95.55 68.8 147;83.39 64.5
124];
X = [ones(size(A,1),1) A(:,1:2)];
y = A(:,3);
[b,bint,r,rint,stats] = regress(y,X);
scatter3(A(:,1),A(:,2),y)
hold on
x1 = min(A(:,1)):max(A(:,1));
x2 = min(A(:,2)):max(A(:,2));
[x1grid,x2grid] = meshgrid(x1,x2);
yfit = b(1) + b(2)*x1grid + b(3)*x2grid;
mesh(x1grid,x2grid,yfit)
xlabel('X1')
```

```
ylabel('X2')
zlabel('Y')
```

Het *y*-intercept is b(1). De elementen van de vector `stats` zijn de R^2 -waarde, de F-waarde van de nulhypothese die zegt dat er geen aansluitend regressiemodel aanwezig is en de p-value geassocieerd met deze F-waarde.

De slechte kwaliteit van het lineaire regressiemodel blijkt zowel uit de grafische voorstelling als uit de R^2-waarde=28.93% als uit de betrouwbaarheidsintervallen voor de b-coëfficiënten die elk 0 bevatten.

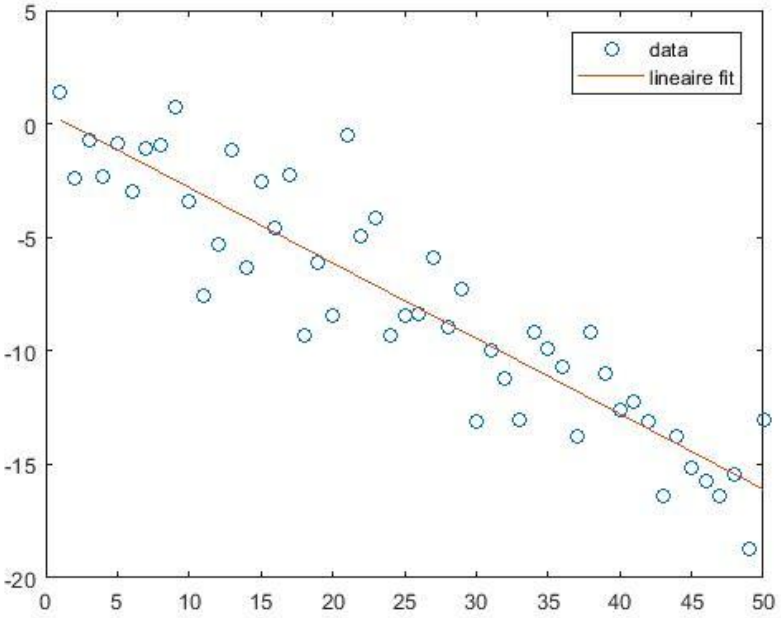
```
stats =    0.2893    1.4245    0.3027  389.0770

bint

 -400.1360  332.7082
   -1.4202    4.3557
   -6.8617    7.4904
```

Voorbeeld: Voer een lineaire regressie uit op datapunten $(i, -0.3\,i + 2\,\varepsilon), 1 \leq i \leq 50$ en ε een random getal uit N(0,1). Maak eveneens een tekening waarbij de datapunten en de regressierechte te zien zijn.

```
x = 1:50;
y = -0.3*x + 2*randn(1,50);
p = polyfit(x,y,1);
f = polyval(p,x);
plot(x,y,'o',x,f,'-')
```

```
legend('data','lineaire fit')
```



9.Contingentieproblemen

[table,chi2,p] = crosstab(x,y)	Genereert een contingentietabel met frequenties.
	De χ^2-test levert een teststatistiek chi2 op en een p-waarde.
	
	Als x en y reeds de frequenties voorstellen i.p.v. de ruwe data, moet dit nog omgezet worden via bijvoorbeeld:
	
	n1 = 51; N1 = 8193; n2 = 74; N2 = 8201;
	x = [repmat('a',N1,1); repmat('b',N2,1)];
	y = [repmat(1,n1,1); repmat(2,N1-n1,1);
	repmat(1,n2,1); repmat(2,N2-n2,1)];

Voorbeeld: Ga de onafhankelijkheid na van 2 variabelen 'land' (1=België, 2=Nederland, 3=Duitsland, 4=Frankrijk) en 'energiebron' (1=wind, 2=zon, 3=fossiel) op basis van een steekproef die je genereert via een discrete random generator. Verwacht je het resultaat?

```
r1 = unidrnd(4,50,1);
r2 = unidrnd(3,50,1);
[table,chi2,p] = crosstab(r1,r2)

        table =
           1     5     5
           4     3     3
           3     4     3
           9     6     4
        chi2 = 5.0625
        p =0.5358
```

Aangezien p>0.05 zal de onafhankelijkheid van land en energiebron aanvaard worden met 95% betrouwbaarheid. Dit is niet te verwonderen met de random oorsprong van de data.
Indien de data bestaat uit absolute frequenties i.p.v. ruwe data, moet eerst nog (artificieel) een data-matrix gemaakt worden waarop de χ^2-test kan worden toegepast.

Voorbeeld: Men wenst te onderzoeken of er een significant verschil is tussen mannen en vrouwen wat betreft hun vervoermiddel om naar het werk te gaan. De aantallen staan genoteerd in volgende tabel. Toets voor α=0.10.

	man	vrouw
Fiets	11	17
Auto	19	10
openbaar vervoer	8	15

```
r1 = [repmat('a',28,1); repmat('b',29,1);repmat('c',23,1)];
r2 = [repmat(1,11,1); repmat(2,17,1);
      repmat(1,19,1); repmat(2,10,1);
      repmat(1,8,1); repmat(2,15,1)];
[table,chi2,p] = crosstab(r1,r2)

  table =
      11    17
      19    10
      8     15
  chi2 =    6.0243
  p =    0.0492
```

Omdat p<0.1 is er een significant verband tussen het vervoermiddel en de gender met 95% betrouwbaarheid.