Een tweede sessie met Matlab

Koen Engelborghs (K01.13)
Jo Simoens (A04.36)

1 Operaties op matrices en vectoren

Matlab is ontworpen voor operaties op matrices en vectoren. Heel wat bewerkingen waar je in een algemene programmeertaal een lusje voor zou schrijven, kunnen in Matlab met een enkel commando. De efficiëntste manier om met Matlab te werken is dan ook van die functionaliteit gebruik van te maken. Hieronder vind je een aantal voorbeelden die daarbij kunnen helpen.

Zo is er naast de gewone matrixvermenigvuldiging ook een componentsgewijze (puntsgewijze) vermenigvuldiging:

```
>> format compact
>> c = [ 1 2 3 ; 4 5 6 ; 7 8 9 ];
>> d = [ 0 1 4 ; 0 0 2 ; 1 0 1 ];
>> c * d
ans =

     3     1    11
     6     4    32
     9     7    53
>> c .* d
ans =
     0     2    12
     0     0    12
     7     0     9
```

Ook andere bewerkingen kun je componentsgewijs uitvoeren door een punt voor de operator te plakken:

```
>> x = 1:4;
>> w = [ 1 -1 3 2 ];
>> x ./ w
ans =
     1    -2     1     2
>> x .^ w
ans =
    1.0000    0.5000   27.0000   16.0000
```

Elementen in een vector optellen of een cumulatieve som berekenen kan even gevat:

```
>> sum( x .^ 2 )
ans =
    30
>> cumsum( x .^ 2 )
ans =
     1     5    14    30
```

Om slechts enkele elementen te selecteren of de volgorde te veranderen gebruik je een vector van indices:

```
>> y = zeros(4,3);
>> y([2 4], [1 3]) = ones(2,2)
y =
     0     0     0
     1     0     1
     0     0     0
     1     0     1
>> z = ones(1,7);
>> z([2 7 5]) = [-17 20 42]
z =
     1   -17     1     1    42     1    20
>> z([7:-1:1])
ans =
    20     1    42     1     1   -17     1
```

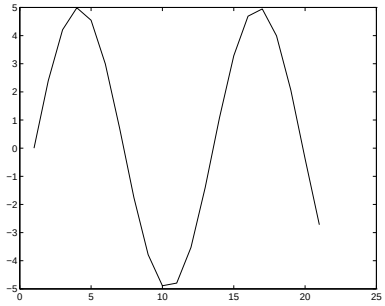
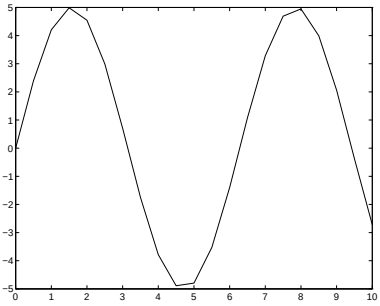
2 Grafieken van 1D data

Een eerste manier om een functie af te beelden is ze te evalueren in een aantal zelfgekozen punten,

```
>> x = 0:0.5:10;
>> a = 5 * sin(x);
>> b = 6 * cos(x);
```

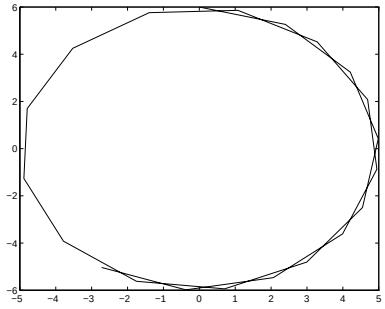
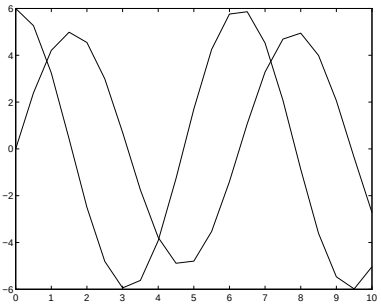
en de vector met functiewaarden uit te zetten. Vergeet niet de juiste abscissen op te geven, anders neemt Matlab de indices in de vector als abscissen:

```
>> figure(1); plot(a);
>> figure(2); plot(x,a);
```

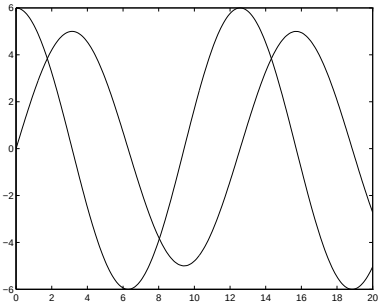
Als meerdere argumenten worden doorgegeven, combineert Matlab de functies in een eenzelfde grafiek:

```
>> figure(1); plot(a,b);
>> figure(2); plot(x,a,x,b);
```

Functies waarvoor een gesloten uitdrukking bestaat, worden indien nodig ook automatisch getekend (let op de komma's en haakjes):

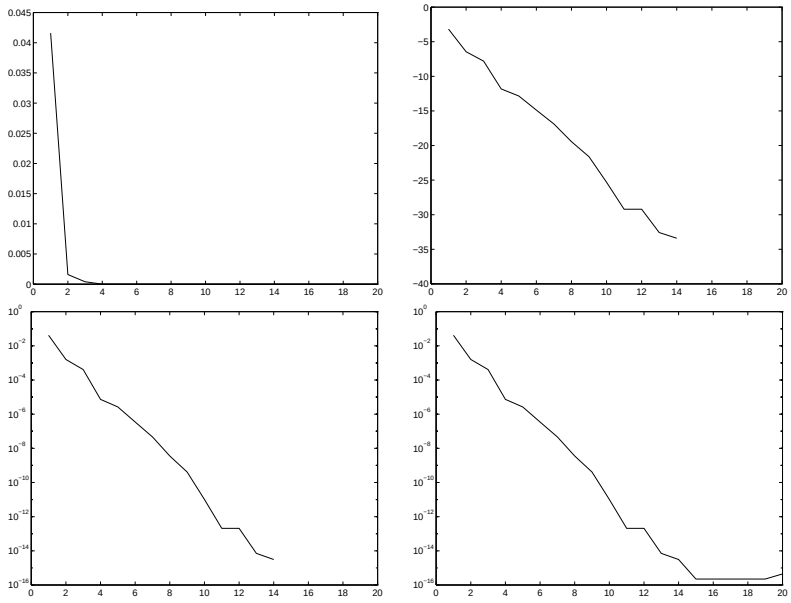
```
>> fplot(
'[5*sin(0.5*x) 6*cos(0.5*x)]',
[0 20]);
```



Het is makkelijker een logaritmische schaal te gebruiken dan het logaritme van een functie te plotten. Als het gaat om een fout — zoals hier — mag je nullen vervangen door de machineprecisie.

```
>> p = round(pi * 10.^(1:20)) ./ (10.^(1:20));
>> fout = abs(pi - p);
>> figure(1); plot(fout);
>> figure(2); plot(log(fout));

Warning: Log of zero
>> figure(3); semilogy(fout);
>> fout = fout + (fout==0)*eps;
>> figure(4); semilogy(fout);
```



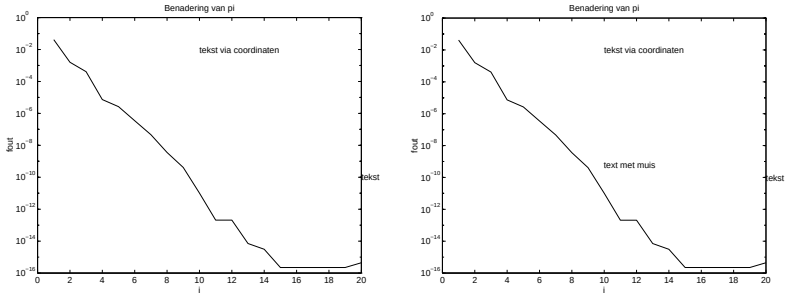
3 Assen en tekst

Er zijn commando's om een figuur van een titel, legendes en losse tekst te voorzien.

```
>> title('Benadering van pi');
>> xlabel('i');
>> ylabel('fout');
>> text(10,1e-2,'tekst via coordinaten');
>> text(20,1e-10,'tekst');
```

De tekst kan ook met de muis op de juiste plek gezet worden:

```
>> gtext('text met muis');
```



Het bereik van de assen wordt ingesteld en uitgelezen met `axis`:

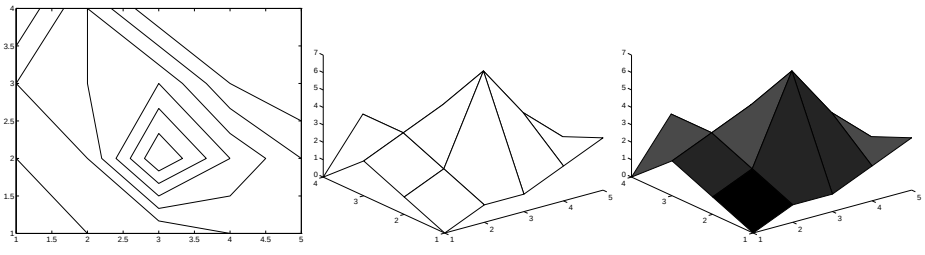
```
>> format short e
>> axis([0 15 1e-7 1e-2])
>> ax = axis
ans =
            0   1.5000e+01   1.0000e-07   1.0000e-02
>> axis('auto')
ans =
            0   2.0000e+01   1.0000e-16   1.0000e+00
```

De verhoudingen van de figuur op het scherm zijn normaliter ongeveer $5:4$, ongeacht het bereik van de assen. Het commando `axis('equal')` zorgt ervoor dat een eenheid op de horizontale en op de verticale as even lang getekend worden. Dit wordt ongedaan gemaakt door `axis('normal')`.

4 Grafieken van 2D data

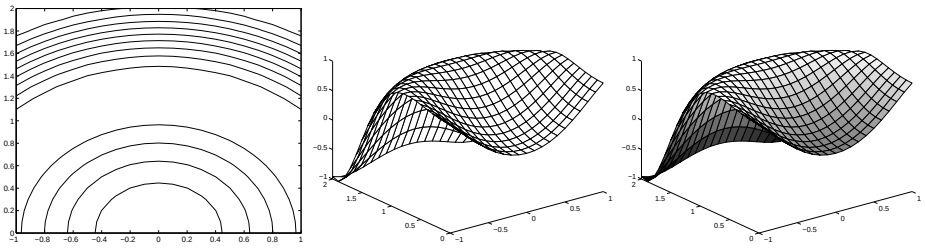
Een matrix van functiewaarden kan rechtstreeks worden geplot. Het tweede argument van `contour` is optioneel en geeft aan op welke niveaus hoogtelijnen getekend moeten worden.

```
>> l = [ 0 1 1 2 3 ; 1 2 7 4 2; 2 3 4 1 0; 0 3 0 0 0 ];
>> figure(1); contour(l,0:10);
>> figure(2); mesh(l);
>> figure(3); surf(l); colormap('gray');
```



Om een continue functie in twee veranderlijken weer te geven moet eerst een matrix met functiewaarden worden gevuld. De functie `meshgrid` kan daarbij handig zijn:

```
>> [x,y] = meshgrid(-1:0.1:1,0:0.1:2);
>> z = sin(x.^2+y.^2);
>> figure(1); contour(x,y,z);
>> figure(2); mesh(x,y,z);
>> figure(3); surf(x,y,z); colormap('gray');
```



5 Werken met figuren

We beginnen met enkele elementaire commando's:

`figure(`n`)` maakt figuur n actief. Als er nog geen figuur bestaat met dat nummer, wordt er een gecreëerd. Het commando `figure` maakt een nieuwe figuur die automatisch een nummer krijgt.

`clf` wist de actieve figuur.

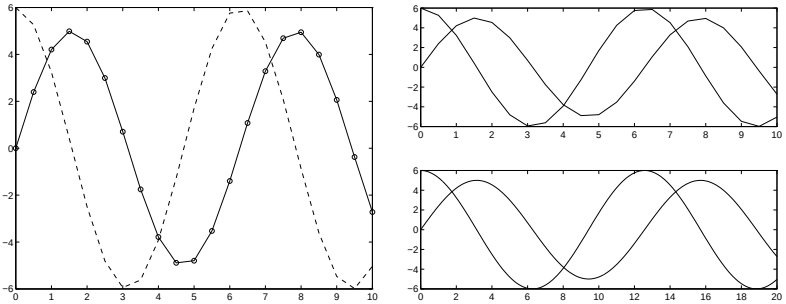
`close` en `close(`n`)` sluiten de actieve c.q. de n-de figuur, en `close all` sluit ze allemaal.

Om in een figuur meerdere grafieken bovenop elkaar te kunnen tekenen, houd je wat al getekend werd vast:

```
>> figure(1);
>> plot(x,a);
>> hold on;
>> plot(x,b,'--');
>> plot(x,a,'o');
>> hold off;
```

In sommige gevallen is het nuttig meerdere grafieken in eenzelfde venster te plaatsen; de eerste twee argumenten van `subplot` geven het aantal rijen en kolommen aan waarin de figuur moet worden verdeeld, het derde argument geeft de positie van de subfiguur aan. Er wordt geteld van boven naar onder en van links naar rechts.

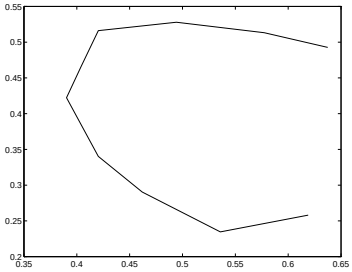
```
>> figure(2);
>> subplot(2,1,1);
>> plot(x,a,x,b);
>> subplot(2,1,2);
>> fplot('[5*sin(0.5*x),6*cos(0.5*x)]',[0 20]);
```



6 In- en uitvoer

De functie `ginput` laat toe een lijst punten in te voeren met de muis. Telkens de gebruiker klikt, wordt opgeslagen wat de coördinaten van de cursor zijn in het huidige assenstelsel en welke muisknop is ingedrukt. De muisknoppen worden genummerd van links naar rechts. De functie keert terug zodra het opgegeven aantal keer werd geklikt.

```
>> clf
>> clear x y;
>> axis([0 1 0 1]);
>> [x,y,button] = ginput(10);
>> plot(x,y);
```



De geproduceerde figuren wil je vast bewaren voor later gebruik. *Encapsulated Postscript* (EPS) is als uitvoerformaat een goede keuze en wordt door de meeste programma's herkend. De Windows-versie van Matlab schrijft ook naar WMF-bestanden. Om de figuur te bewaren precies zoals hij op het scherm staat, moet je hem eerst bevriezen.

```
>> set(gcf,'Resize','off')
>> print figuur.eps -deps
```

Tenslotte kan je ook gegevens naar een tekstbestand schrijven:

```
>> fout = fout'
>> save tabel.txt fout -ascii
```

Het uitgevoerde bestand is leesbaar voor programma's als gnuplot, je favoriete spreadsheet, en natuurlijk Matlab zelf.