UMEÅ UNIVERSITET
Datavetenskap
Marie Nordström/Per Lindström

March 2003

MATLAB

Introduction

MATLAB is an integrated development environment, with matrices as the basic data structure. You can do advanced computations and visualize the result in a simple way. In MATLAB you have the possibility to work directly in the workspace and use Matlab as an advanced calculator but it is also possible to program in Matlab using functions and command files (.m files).

Purpose

The following exercises aim at making you acquainted with Matlab's syntax and a little about the graphical possibilities.

Hints

- Use the built in functions. The command `help` in the command window or as one of possible menus or by typing `helpbrowser` that gives a more detailed description in "web looking" format.
- `who` shows the current variables
- `format compact` gives less empty lines when printing
- Previously used commands are stacked and can be reused by the arrow keys ↑ and ↓.
- Remenber that operators are normally defined as matrices operators. E.g. A*B where A and B are matrices will compute the matris product AB provided the dimensions of A and B match (A m*n and B n*p results in a matrix of order m*p). The operators may be used element-wise by preceeding the operator with a dot as in A.*C where the matices A and C should have the same dimensions. If A and C are of order m*n the reulting matix D=A.*C is an m*n matrix with elements $D_{ij} = A_{ij}*C_{ij}$ for i=1,2,,,m and j=1,2,,,,n. For multiplication, division and exponentiation we have .*, ./, .^
- Dimensions are important! Every operation must be correct concerning dimensions. Scalars are normally exceptions, as in mathematics. E.g. A=2*B is ok since every element in B is multiplied with 2.

Before starting with the exercises below you should

- login on the Windows system
- identfy yourself to Home since that is needed if you would like to print on the printer in a PC-lab
- create a directory matlab in your home directory on the Unix-system (peppar). E.g. create matalb as a subdirectory under ~username/edu/
- start Matlab by clicking on the icon  MATLAB 6.1.lnk
- set the current directory (in Matlab) to ~username/edu/matlab by browsing

Basic commands

(Pages refer to *Användarhandledning för MATLAB 6*)

*******Concentrate on exercises written in English!******

1. Create the matrices (p.26-27)

$$A=\begin{bmatrix} 1 & 4 & 0 \\ 6 & 3 & 1 \\ -1 & 0 & -1 \end{bmatrix},\quad B=\begin{bmatrix} 3 & 1 & -4 \\ -1 & 5 & 19 \\ 2 & -6 & -5 \end{bmatrix} \text{ and } v=\begin{bmatrix} -1 \\ 3 \\ 0 \end{bmatrix}$$

2. Compute the products `2A`, `AB` och `A.*B` and note the differences (p. 48,52)

3. Compute B-transpose, B^T (p.51)

4. Compute the solution of $Ax = v$ without using the inverse A^{-1} (p.121)

5. Put v as the second column in A (p.74)

6. Put v as the third row in B (Note: to make v a row vector you have to transpose it) (p.74)

7. Create a row vector x containing $0, \pi/6, 2\pi/6, \ldots 2\pi$. Check the size of x with the command `size` (p. 30,72)

8. Create a column vector `t` with 200 elements between 0 och 4π (p.72,73)

9. What is the difference of `length` and `size`?

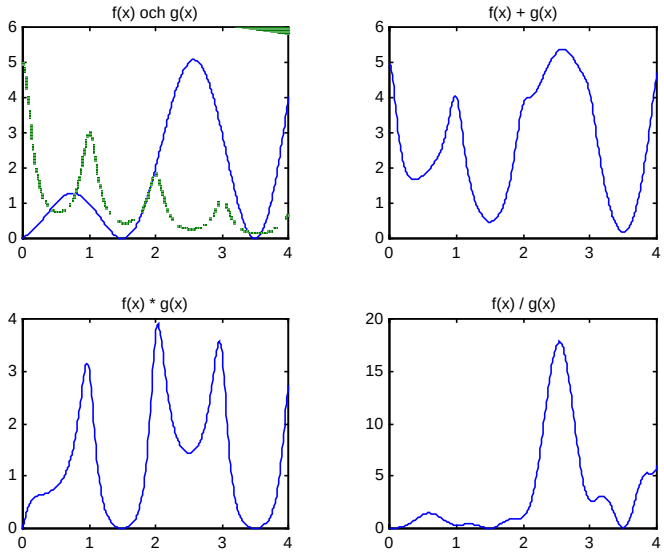
10. Create the matrix $D=\begin{bmatrix} 11 & 9 & 9 \\ 9 & 11 & 9 \\ 9 & 9 & 11 \end{bmatrix}$ without typing in the elements one by one (sec. 4.1).

Hint: use `ones()` and `diag()`.

Exercises (with permission of Gerd Eriksson NADA,KTH)

1. Function curves

Plot curves for the functions $f(x) = x(1+\sin \pi x)$ and $f(x) = \dfrac{5e^{-x/2}}{3-2\cos 2\pi x}$ in the interval $0 \le x \le 4$. Experiment to see what is a suitable distance between points on the x-axes. Use different plot windows and plot both curves in the same plot window. Also, plot by using `subplot` curves for $f(x)+g(x)$, $f(x)\cdot g(x)$ and $f(x)/g(x)$.



Useful MATLAB -commands

☐ `: (op)` ☐ `plot` ☐ `subplot` ☐ `title`

☐ `./ (op)` ☐ `xlabel` `ylabel`

2. One more function

Plot the curve $f(x) = \dfrac{10}{\sqrt{1+x^2}} + \dfrac{e^{x/2}}{\sqrt{2}+\sin \pi x} + \dfrac{4}{x-5}$ in the interval $-2 \le x \le 4$, with a suitable step between x-points such that the curve dose not not look edgy.

Useful MATLAB -commands

☐ `.^ (op)` ☐ `exp` ☐ `grid`

3. Circles

Circles can be plotted by using the parameter form of a circle: $x = x_c + R\cos\varphi$, $y = y_c + R\sin\varphi$ where the angle runs from 0 to 2π. Plot a circle with radius 3 and the origin at $(0,1.42)$. Use the stepsize $2\pi/60$ for the angle. Mark the mid-point. Use the command `axis equal` to make the circle look like a circle.

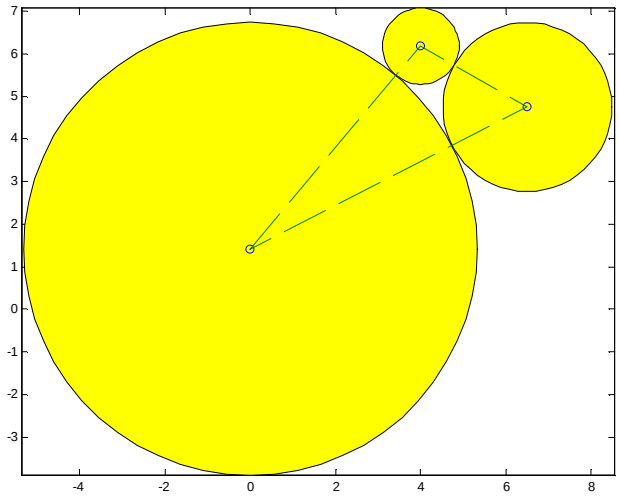
Useful MATLAB -commands

☐ `pi` ☐ `axis equal` ☐ `plot(x,y,'o')`

4. Three touching circles

Let three touching circles have the center at the points (x_1,y_1), (x_2,y_2) and (x_3,y_3) respectively.

`x=[0 4 6.5]; y=[1.42 6.18 4.75];` Compute the radius of each circle such that they touch each other and plot the triangle and the circles. Hint : the relations $r_1 + r_2 = s_1$, $r_2 + r_3 = s_2$ and $r_3 + r_1 = s_3$ between the radii and the sides of the triangle are easiest written on matrix-vector form as a system of linear equations Ar=s. Solve with `r=A\s`. (only ones



and zeros in the matrix A).

Useful MATLAB -commands

☐ `\ (op)`	☐ `clear`	☐ `clf`	☐ `hold on`	☐ `diff`
☐ `^. (op)`	☐ `sqrt`	☐ `for`	☐ `fill`	

5. Darts

Plot ten concentric circles with radius 1,2, ..., 10. Fill the innermost with red. Put point marks on the dart board with the `text`-function. Generate ten normal distributed darts with standard deviation 5 in x-direction and 4 in y-direction:

```
for pil=1:10
     plot(5*randn,4*randn,'*')
     pause(0.7)
end
```

the `pause`-statement gives 0.7 seconds delay between each dart. The flashing in the plot window at each dart is due to the erasing of the current plot before the new one is drawn and replaced by a new figure. You can get rid of the flashing by using doubble-buffering. Use

```
set(gcf,'DoubleBuffer','on')
```

Redesign your program such that the total points after ten darts is printed on the screen. The experience of the darter can be simulated by changing the standard deviation used in the program.

Useful MATLAB -commands

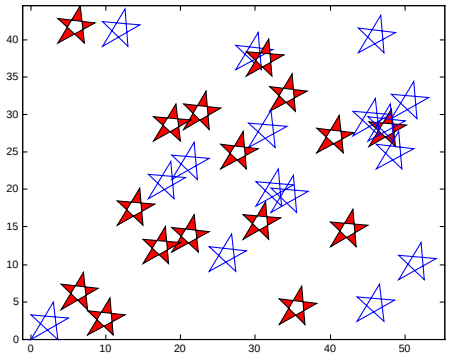
☐ `pause`	☐ `for`	☐ `randn`	☐ `gcf`

☐ `set(gcf` ☐ `if - else` ☐ `axis off` ☐ `text`

6. Five pointed stars

Five pointed stars can be drawn without lifting your pencil. Draw such a one in Matlab. Let one point be in the origin and the rest on suitable coordinates. Also, draw a five pointed star that is moved four units in both x- and y-direcdtion. Use `fill()` instead of `plot()` for that one.

Make a starry sky by generating 15 random filled and unfilled stars respectively evenly distributed in $0 \le x \le 60$ and $0 \le y \le 45$



Useful MATLAB -commands

☐ `pause`

7. Klicka in punkter

Modifiera uppgiften med tre cirklar så att användaren får ange tre önskade hörnpunkter till en triangel. Detta skall ske genom att användaren klickar i grafikfönstret.
Börja med ett axis-kommando med lämpliga värden för x- och y-axlarna, t.ex.

```
clear, clf, axis([0 10 0 8]), hold on
```

Klickning av tre punkter kan göras med `[x,y]=ginput(3); plot(x,y,'o')`
Nackdelen är att punkterna inte markeras förrän alla tre matats in. Pröva!
Det är bättre att klicka in en punkt i taget, markera den och uppdatera x och y vektorerna.

```
x=[]; y=[];
for i=1:3
  [xp,yp]=ginput(1); plot(xp,yp,'*'), x=[x; xp]; y=[y;
yp];
end
```

Användbara MATLAB -kommandon

☐ `ginput` ☐ `axis` ☐ `disp` ☐ `clf`

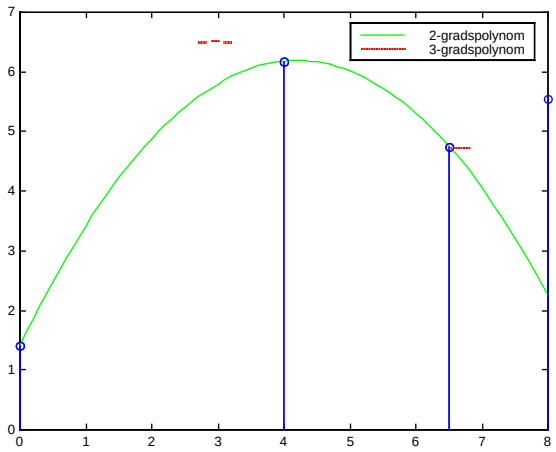
8. Polynom through given points

From mathematics it is know that a unique polynomial of degree 3 that passes through 4 given points can be determined.
The simplest way of representing the polynomial is $P(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3$
Four points are given $(x_i, y_i) \quad i = 1,..,4$. Since the polynomial should interpolate at each given point we demand $P(x_i) = y_i$ which leads to a system of equations $Ac = y$ where the coefficients c_0, c_1, c_2, c_3 are unknown.

The system matrix can be written `A=[ones(size(x)) x x.^2 x.^3]` where `x` is a column vector containing the x-values.

a) We have only 3 given points, the points that make up the corners in the triangle from problem 4. Compute and plot the unique parabola that interpolates the given points.
b) Add the point (8, 5.55) and compute the third degree polynomial that passes through the four given points. Matk the given points in the plots too. Try to use `stem(x,y)`.

Useful MATLAB –commands



☐ `ones` ☐ `stem` ☐ `xlabel` ☐ `legend` ☐ `figure(2)`

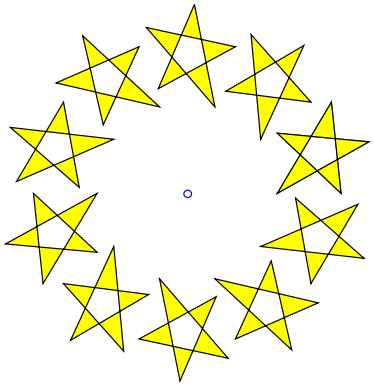
9. Tvisted figures

By using a transformation matrix $S = \begin{pmatrix} \cos(v) & -\sin(v) \\ \sin(v) & \cos(v) \end{pmatrix}$ it is easy to rotate figures..

In the folliwing we start with the five pointed star from problem 5 and moves it 5 units to the right and rotates around the origin.

.

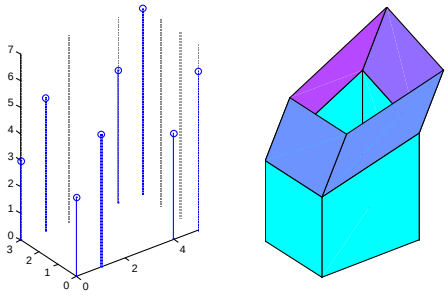
```
% femuddsnurr
clear, clf
x=5+[0 5 0 3.5 3 0];
y=[0 2.8 3.3 0 5 0];
plot(0,0,'o', x,y)
axis equal, hold on,axis off
n=10; v=2*pi/n;
```



```
S=[cos(v) -sin(v);
sin(v) cos(v)];
for k=1:n
      P=S*[x;y];
      x=P(1,:);
      y=P(2,:);
      fill(x,y,'y')
end
```

10. Sned låda, 3D-ritning

Den sneda lådan i den högra figuren är konstruerad av stolparna till vänster som ritas med `stem3(x,y,z)`. Rektangeln i botten är fyra enheter i x-led och tre enheter i y-led. De heldragna stolphöjderna är `[3 5 5 3]`. De streckade stolparna har förflyttats en enhet i x-led och höjderna ökats med två enheter. Rita figuren med `surf`-kommandot.

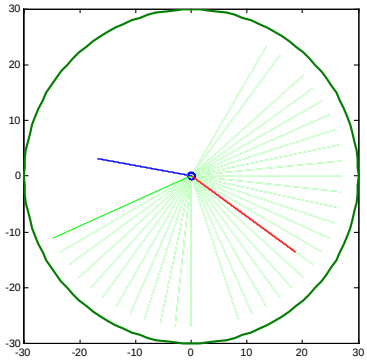


Användbara MATLAB -kommandon

☐ `stem3` ☐ `surf` ☐ `colormap` ☐ `rotate3d`

11. Klocka

Konstruera en klocka med cirkel-formad urtavla och färgglada tim-, minut- och sekundvisare.



Du behöver inte sätta ut markeringar på urtavlan. Rita om visarna varje hel sekund, utnyttja `drawnow` efter `plot`-satsen. MATLAB har en funktion clock som retrurnerar en vektor med följande innehåll `[year month day hour minute seconds]`, där de fem första är heltal och den sista, `seconds` , har några få decimaler. Detta kan man lösa med `fix(clock)`. För att slippa flimmer kan du utnyttja samma teknik som i pilkastningsuppgiften.
Tänk på att låta timvisaren följa minutvisaren så att den inte "skuttar" fram en timme i taget.

Användbara MATLAB -kommandon

☐ `clock` ☐ `fix` ☐ `plot(..,..,'Linewidth',2)` ☐ `plot(..,..,'b')`

12. Banana function

The banana function is famous in optimization because of its flat vally where some optimization methods make very little progres on its way towards the minimum of the function. One of the demos in Matlab demonstrates how different optimization methods converges fast or slow towards the minimum value of the banana function at (1,1).

The banan function is defined as $f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$

a) plot the surface of f(x) in the range $-1.5<x_1<1.5$ and $-0.5<x_2<1.5$

b) make a contour plot in the same region as in a) above. It is a bit tricky to "see the vally". Experiment with how many "isobars" that should be printed by contour().

c) Imagine an optimization method has started in Xstart=(-1.3, 1) and proceeds to successive points (x_i, y_i) on its way towards the minimum (1,1). Illustrate the "way" this optimization method has taken by plotting lines from Xstart to the end of the arrays x and y where the way is described by the points in x and y. x=[-1.3, -1.35, -1.1, -0.6, -0.2, 0.3, 0.5, 0.75,1] y=[1, 1.2, 0.9, 0.5, -0.1, 0.05, 0.5, 0.8, 1]