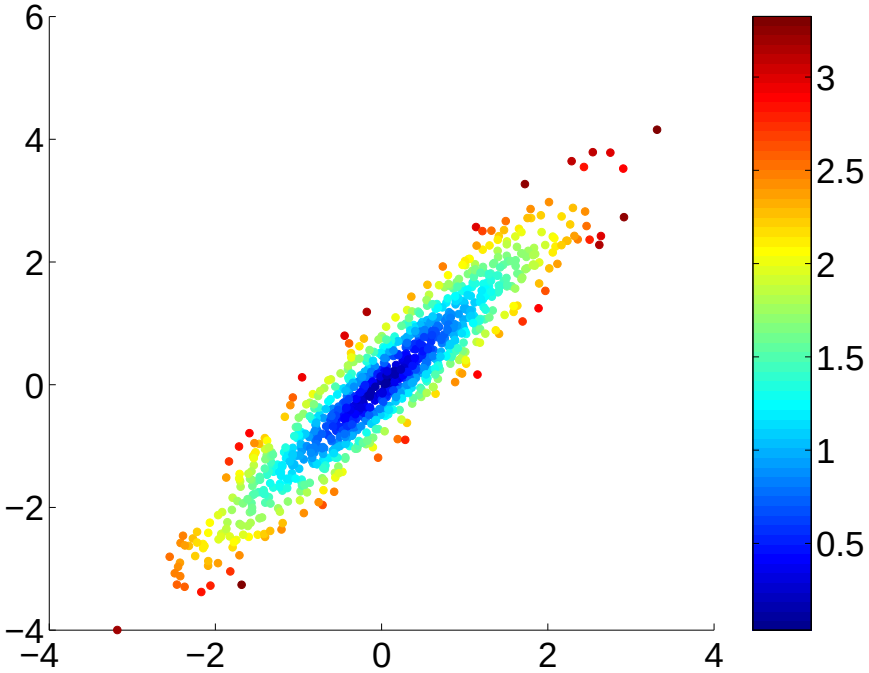
Basics of MATLAB

Jarmo Hurri

What is MATLAB?

- a fast and convenient way for doing numerical computations and simulations (especially prototyping)
- an interactive computing environment
- supports, for example
  - matrix computations
  - graphical display of data
  - execution control (if, for etc.)
  - user-defined functions
  - derived data types (structures, classes)

Example




```
X=rand(2)*randn(2,1000);
C=inv(cov(X'));
XNorms=sqrt(sum (X.*(C*X)));
scatter(X(1,:),X(2,:),20,XNorms,'filled');
colormap jet;colorbar;
print -dpng gaussian-example.png
```

Contents

Getting started

- to run in console window: `matlab -nodesktop`
- to run commands at startup: 1. create directory `$HOME/matlab` 2. in that directory, create file `startup.m` and type your startup commands into that file
- from home: ssh is currently only possible to a few servers, so
  - first login to a server, e.g.,
    `ssh -l <username> melkinkari.cs.helsinki.fi`
    where `<username>` is your username (other servers are `melkki` and `melkinpaasi`)
  - now do `ssh sbz-35`; only if this machine is down, then do `ssh lc-<n>`, where `<n>` is the number of the machine (1–10)
  - but graphics is routed via the first server...

Workspace and variables (1/2)

- note: in MATLAB, the percent-character (%) begins a comment

```
>> a=1 % declare and define new variable
a =
     1
>> x=[1 2 3]; % semicolon - do not print result
>> x
x =
     1     2     3
>> A=[1 2 3;4 5 6;7 8 9] % matrix by rows
A =
     1     2     3
     4     5     6
     7     8     9
```

Workspace and variables (2/2)

```
>> whos % variables in workspace
  Name        Size                    Bytes  Class

  A           3x3                        72  double array
  a           1x1                         8  double array
  ans         1x1                         8  double array
  x           1x3                        24  double array
>> clear a % remove variable a
```

Matrix computations and manipulation (1/6)

```
>> x = [0:pi/4:pi/2] % [first:step:upto], pi constant
x =
         0    0.7854    1.5708
>> y=[0:3] % assumes a step size of 1
y =
     0     1     2     3
>> A % this one we defined above
A =
     1     2     3
     4     5     6
     7     8     9
>> size(A) % returns the dimensions of A
ans =
     3     3
```

Matrix computations and manipulation (2/6)

- operator $*$ denotes *matrix multiplication*
- operator / denotes *right division* ($\mathbf{X}/\mathbf{Y} = \mathbf{X}\mathbf{Y}^{-1}$)

```
>> A*x
??? Error using ==> *
Inner matrix dimensions must agree.
>> A*x' % x' is the transpose of x
ans =
    6.2832
   13.3518
   20.4204
>> x*A*x'
ans =
   42.5627
```

Matrix computations and manipulation (3/6)

- element-wise operations are denoted by .op (e.g., .$*$, ./, .^, for $+$ and - no difference)

```
>>  x.*x
ans =
         0    0.6169    2.4674
>> x*x' % dot product...
ans =
    3.0843
>> sum(x.*x) % ... is the same as sum of x.*x
ans =
    3.0843
```

Matrix computations and manipulation (4/6)

- elements of matrices can be indexed in different ways

```
>> B=A % define B and copy A to it
B =
     1     2     3
     4     5     6
     7     8     9
>> B(1:2,2:3) % rows, cols, index starts from 1
ans =

     2     3
     5     6
```

Matrix computations and manipulation (5/6)

```
>> B(:,1)=ones(3,1) % : means assign to all elements
B =
     1     2     3
     1     5     6
     1     8     9
>> B(3,2:end) = 0 % end denotes last element
B =
     1     2     3
     1     5     6
     1     0     0
```

Matrix computations and manipulation (6/6)

```
>> B>4 % a logical matrix with elements > 4
ans =
     0     0     0
     0     1     1
     0     0     0
>> B(B>4)=-1 % assignment using logical selection
B =
     1     2     3
     1    -1    -1
     1     0     0
>> B(B<0|B>1)=0 % | is logical or, & is and
B =
     1     0     0
     1     0     0
     1     0     0
```

Commands and functions (1/4)

```
>> help ones
 ONES   Ones array.
    ONES(N) is an N-by-N matrix of ones.
    ONES(M,N) or ONES([M,N]) is an M-by-N matrix of ones.
    ONES(M,N,P,...) or ONES([M N P ...]) is an M-by-N-by-P-by-...
    array of ones.
    ONES(SIZE(A)) is the same size as A and all ones.

    See also EYE, ZEROS.
>> help sin
 SIN    Sine.
    SIN(X) is the sine of the elements of X.
>> more on % paging on; no paging with more off
```

Commands and functions (2/4)

```
>> sin(pi/2)
ans =
     1
>> help ans
 ANS Most recent answer.
    ANS is the variable created automatically when express
    are not assigned to anything else. ANSwer.
>> x % was defined above
x =
         0    0.7854    1.5708
>> sin(x) % most functions will work with matrices
ans =
         0    0.7071    1.0000
```

Commands and functions (3/4)

```
>> B=diag([1 2 3]) % a matrix with [1 2 3] on diagonal
B =
     1     0     0
     0     2     0
     0     0     3
>> inv(B) % the inverse of B
ans =
    1.0000         0         0
         0    0.5000         0
         0         0    0.3333
>> diag(inv(B))' % diag on matrix returns diagonal vector
ans =
    1.0000    0.5000    0.3333
```

Commands and functions (4/4)

```
>> eig(B)' % eigenvalues of B
ans =
     1     2     3
>> [E D]=eig(B) % eigenvalue decomposition B = E*D*E'
E =
     1     0     0
     0     1     0
     0     0     1
D =
     1     0     0
     0     2     0
     0     0     3
```

- for more information see

  http://www.mathworks.com/products/matlab/functionlist.html

Statements (execution control)

- most "typical" statements available
- `if (x < 1) error ('xerr'); end`
- `if a > b, disp('a'); else disp('b'); end`
- `if ... elseif ... else ... end`
- `for k = 100:-1:1 disp (k); end`
- `k=100;while (k >= 1) disp (k); k=k-1; end`

Avoiding loops (1/2)

- avoiding loops (`for` etc.) is wise for optimal computation speed
- example: normalize all the columns of matrix `A`

```
>> A=randn(2,3) % a matrix with random Gaussian elements
A =
   -0.4326    0.1253   -1.1465
   -1.6656    0.2877    1.1909
>> norms = sqrt(sum(A.^2)) % norms of colums
norms =
    1.7208    0.3138    1.6531
>> divisor = repmat(norms,[2 1]) % expand norms by replicating
divisor =
    1.7208    0.3138    1.6531
    1.7208    0.3138    1.6531
>> sqrt(sum((A./divisor).^2)) % divisor normalizes
ans =
     1     1     1
```

Avoiding loops (2/2)

- example: evaluate function $f(x, y) = \sqrt{x^2 + y^2}$ at points $x, y \in \{-1, 0, 1\}$

```
>> [x y]=meshgrid([-1:1]) % x and y values at (x,y)-coordinates
x =
    -1     0     1
    -1     0     1
    -1     0     1
y =
    -1    -1    -1
     0     0     0
     1     1     1
>> sqrt(x.^2 + y.^2)
ans =
    1.4142    1.0000    1.4142
    1.0000         0    1.0000
    1.4142    1.0000    1.4142
```

Structures

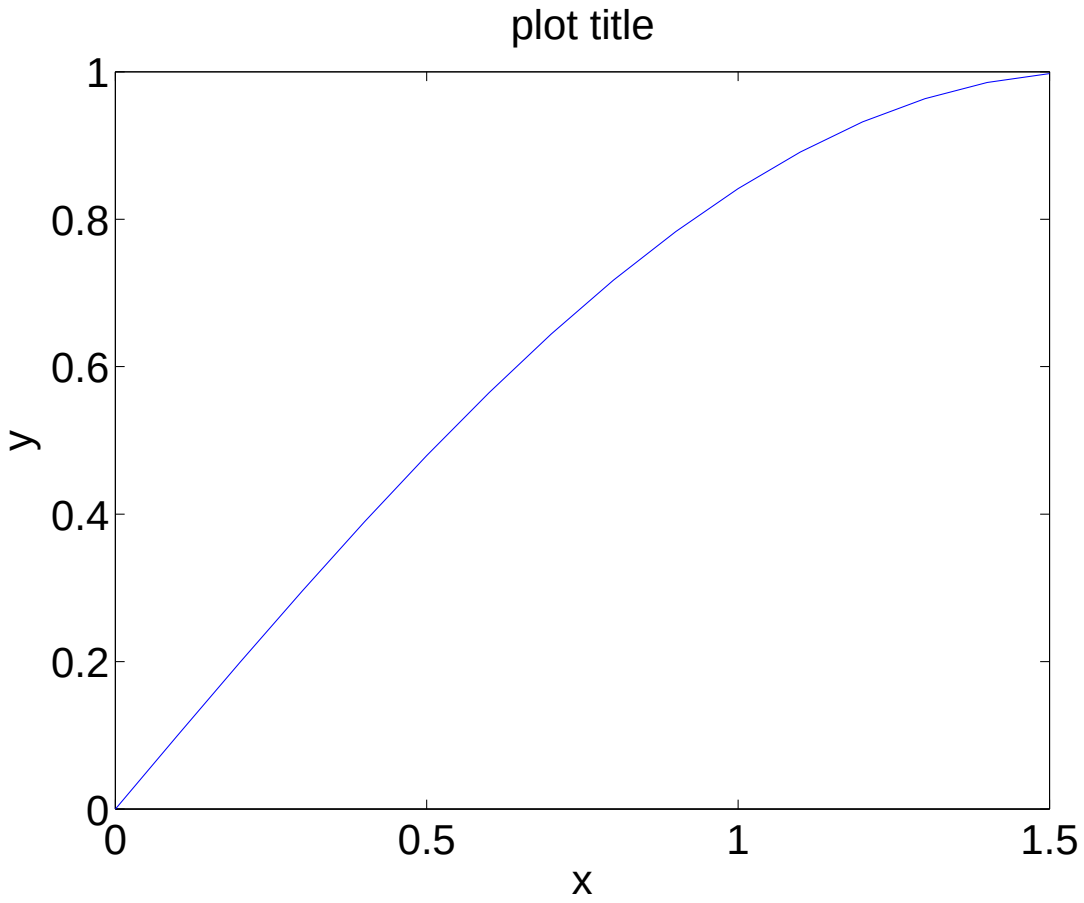
```
>> s.a = 1; % structure s with field a with value 1
>> s.b = 'abc'; % field can be of different data types
>> s
s =

    a: 1
    b: 'abc'
```

Graphics (1/5)

- 2D-plots

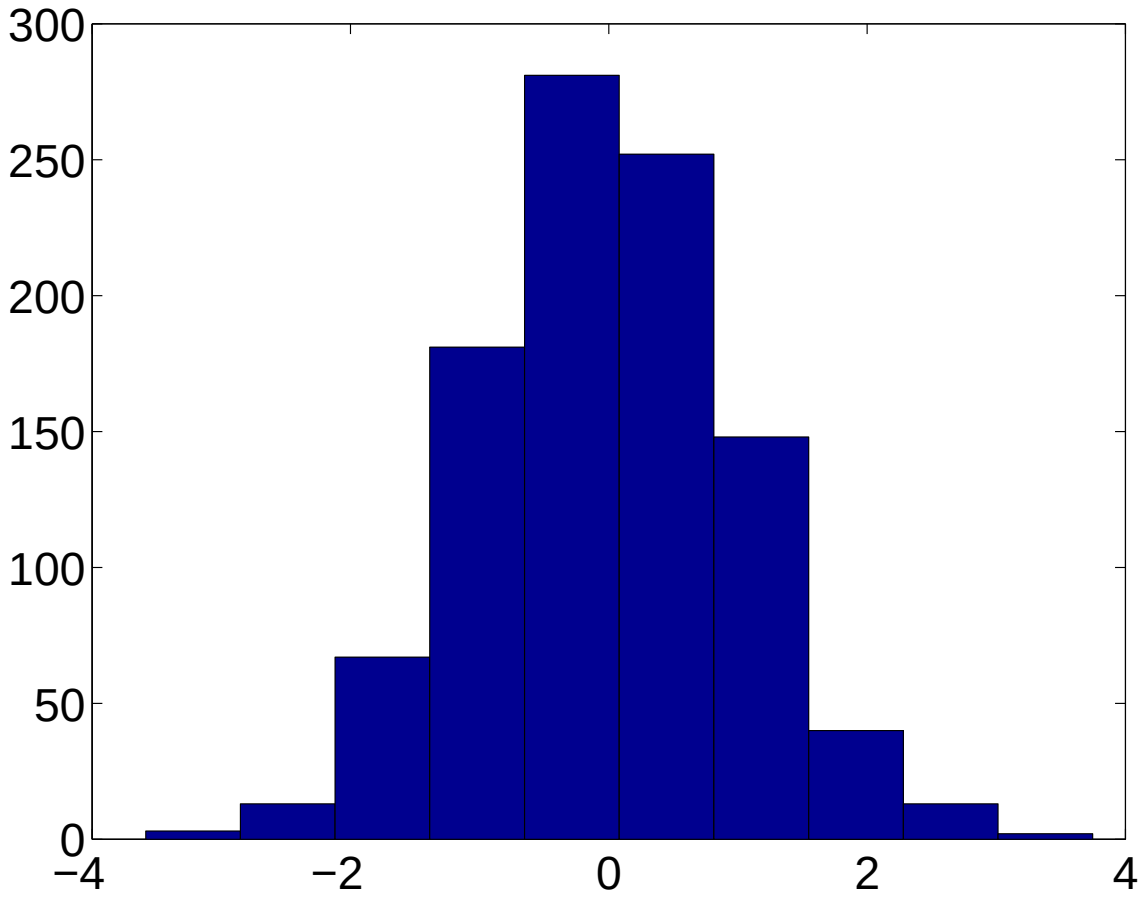
```
>> 'this is a string' % strings are in quotes
ans =
this is a string
>> x=[0:.1:pi/2];
>> plot (x,sin(x)); % x on x-axis, sin(x) on y-axis
>> xlabel 'x'; % label text on x-axis
>> ylabel 'y'; % label text on y-axis
>> title 'plot title'; % title text for the whole plot
>> print -dpng sin.png; % print into a PNG file
```

Graphics (2/5)



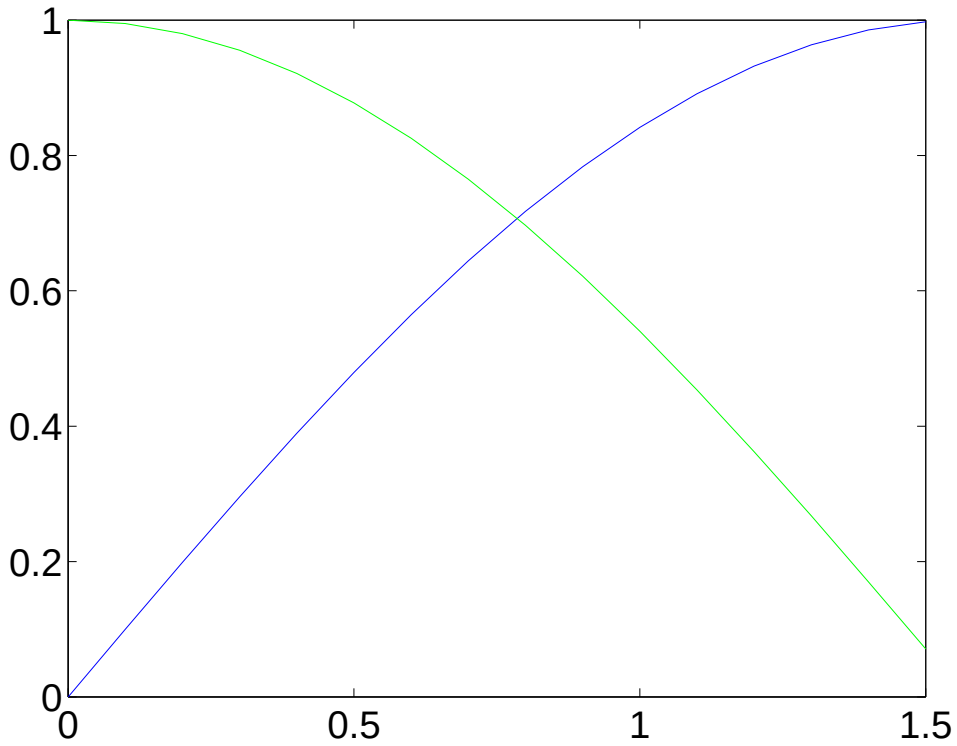
Graphics (3/5)

```
>> hist (randn (1000,1)); % histogram of Gaussian random numbers
```



Graphics (4/5)

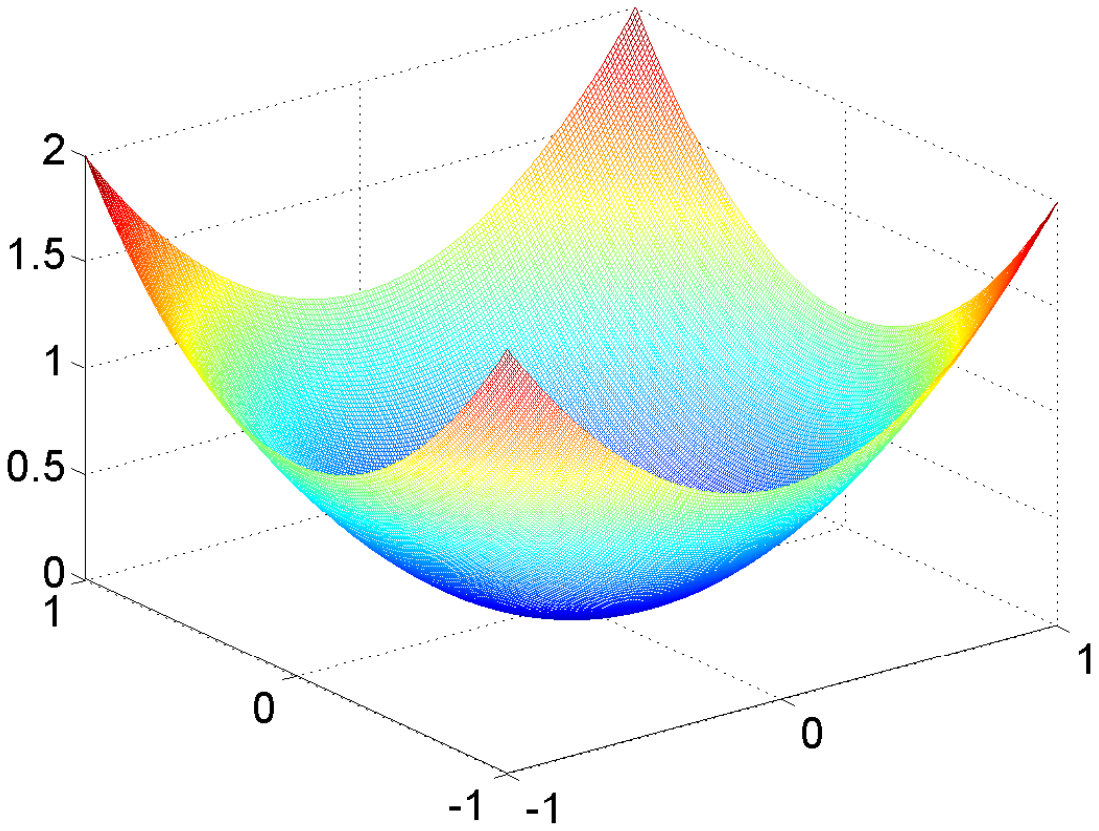
```
>> plot (x,sin(x));
>> hold on; % keep the old plots when plotting
>> plot (x,cos(x),'g'); % second plot in 'g'reen color
>> hold off; % new plots on ``empty paper''
```



Graphics (5/5)

- 3D-plots

```
>> x=[-1:.01:1];
>> [xx yy]=meshgrid(x);
>> mesh (x,x,xx.^2 + yy.^2); % a 3d-plot of f(x,y) = x^2+y^2
```



Basic output and input (1/3)

```
>> disp('Hello!'); % print string (or variable value)
Hello!
>> error('Your mistake'); % print and abort execution
??? Your mistake
>> k=100;
>>fprintf ('k=%d\n', k); % formatted output
k=100
```

Basic output and input (2/3)

```
>> s = sprintf ('file-%04d.dat', k) % format to a string
s =
file-0100.dat
>> fid = fopen (s, 'a') % open file for 'a'ppeding
fid =
     3
>> fprintf (fid, 'data-start'); % output to file
>> fclose (fid) % close file
ans =
     0
```

Basic output and input (3/3)

```
>> clear;
>> v1=randn(2,100);
>> v2=randn(2,100);
>> save vs.mat v1 v2 % save variables to file
>> clear
>> load vs.mat % load variables from file
>> size(v1)
ans =
     2   100
```

User-defined command sequences and functions (1/2)

- a command sequence or function is saved in an `.m`-file
- example command sequence: `plotResult.m`

```
load result.mat
plot (result, 'b');
xlabel ('index');
ylabel ('result');
print -dpng result.png;
```

- all introduced variables are in the workspace!

User-defined command sequences and functions (2/2)

- functions can have local variables: example `normalizeVectors.m`

```
function [newVectors, norms] = normalizeVectors (vectors);
%function [newVectors, norms] = normalizeVectors (vectors);
%
%Normalizes each column vector to norm 1.  Returns the new vectors and the
%norms of the original vectors.
% The first commented section is shown when you type 'help normalizeVectors'

norms = sqrt (sum (vectors .^ 2));
divNorms = norms;
divNorms (divNorms == 0) = 1; % avoid division by zero
newVectors = vectors ./ (repmat (divNorms, [size(vectors,1) 1]));
```

More information

- MATLAB manual on-line

  http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.shtml