



Evolutionary algorithms

An introduction to MATLAB



What is MATLAB?

- It stands for **MATrix LABoratory**
- It is developed by The Mathworks, Inc.
(<http://www.mathworks.com>)
- It is an interactive, integrated, environment
 - for numerical computations
 - for symbolic computations (via *Maple*)
 - for scientific visualizations
- It is a programming language!



Characteristics of MATLAB

- Programming language based on matrix notation.
 - Slow (compared with fortran or C): it is an interpreted language, i.e. not pre-compiled.
 - Efficient when matrices are involved! Mathworks holds unrevealed algorithms for matrix manipulations...
 - Automatic memory management, i.e., you don't have to declare arrays in advance.
 - Intuitive, easy to use.
 - Compact and handy.
 - Shorter program development time than traditional programming languages such as Fortran and C.
 - Can be converted into C code via MATLAB compiler for better efficiency.
- Many application-specific toolboxes available.



MATLAB Tool-Boxes

- Signal Processing
- Curve-Fitting
- Bioinformatics
- Financial
- Image Processing
- Neural Networks
- Databases
- And many others!



MATLAB Preliminaries

- Latest version is MATLAB 7.0.
- Invoke by typing **matlab** at system prompt:
 - PC's rooms 305-6 – full graphical interface.
 - Sun machines – run X11 terminal for GUI...
- LIACS has a limited number of licenses.



Getting started

Row vector:

```
>> x = [3,6,9]
      x =
           3         6         9
```

Column vector:

```
>> y = 3 * ones(3,1)'
      y =
           3         3         3
```

Elementwise division:

```
>> x ./ y
      ans =
           1         2         3
```



Inner product / outer product

```
>> x=[3,6,9];
```

```
>> y=ones(3,1);
```

```
>>x*y
```

```
ans =
```

```
18
```

```
>>y*x
```

```
ans =
```

```
3    6    9
```

```
3    6    9
```

```
3    6    9
```



And then matrices

Use semicolons to separate the rows:

```
>> A = [1 2 0; 2 5 -1; 4 10 -1]
```

A =

1	2	0
2	5	-1
4	10	-1

And the transpose of A:

```
>> B = A'
```

B =

1	2	4
2	5	10
0	-1	-1



MATLAB and matrices

- MATLAB doesn't require you to deal with matrices as a collection of numbers. MATLAB knows when you are dealing with matrices and adjusts your calculations accordingly.
- Matrix multiplication:
`>> C = A*B;`
- Element-wise multiplication:
`>> D = A.*B;`
- Inverse operator:
`>> E = inv(A);`
- Eigenvalues:
`>> e = eig(A);`



Creating your own functions

You can create your own MATLAB functions by creating a text file with extension '.m'. A simple example:

```
function area = traparea(a,b,h)
% traparea(a,b,h)      Computes the area of a trapezoid
%                       given the dimensions a, b and h,
%                       where a and b are the lengths of
%                       the parallel sides and h is the
%                       distance between these sides
area = 0.5 * (a + b) * h;
```



MATLAB Efficiency

- User-defined MATLAB functions are *interpreted*, not *compiled*. This means that when an **m-file** is executed, each statement is read and then executed, rather than the entire program being parsed and compiled into machine language.
- For this reason, MATLAB programs can be much slower than programs written in a language such as FORTRAN or C.
- In order to get the most out of MATLAB, it is necessary to use built-in functions and operators whenever possible (so that compiled rather than interpreted code is executed).



Matrix Operations versus loops

Compare:

```
>> dx = pi/30;  
>> nx = 1 + 2*pi/dx;  
>> for i = 1:nx  
    x(i) = (i-1)*dx;  
    y(i) = sin(3*x(i));  
end
```

With:

```
>> x = 0:pi/30:2*pi;  
>> y = sin(3*x);
```



Efficiency continued

- Bottom line – try to work as much as possible with matrix notation!
- Examine the time execution of your code:

```
>> Tic;  
    -- Operation --  
>> Toc;
```



Random Distributions

Efficient random number generator:

```
A = rand (4,4); %Uniform [0,1]
```

```
B = 2*rand(2,2) - 1; %Uniform [-1,1]
```

```
C = randn(1,100); %Normal Dist.!
```

What does this piece of code do? What is D...?

```
D = rand(15,1) > 0.5;
```

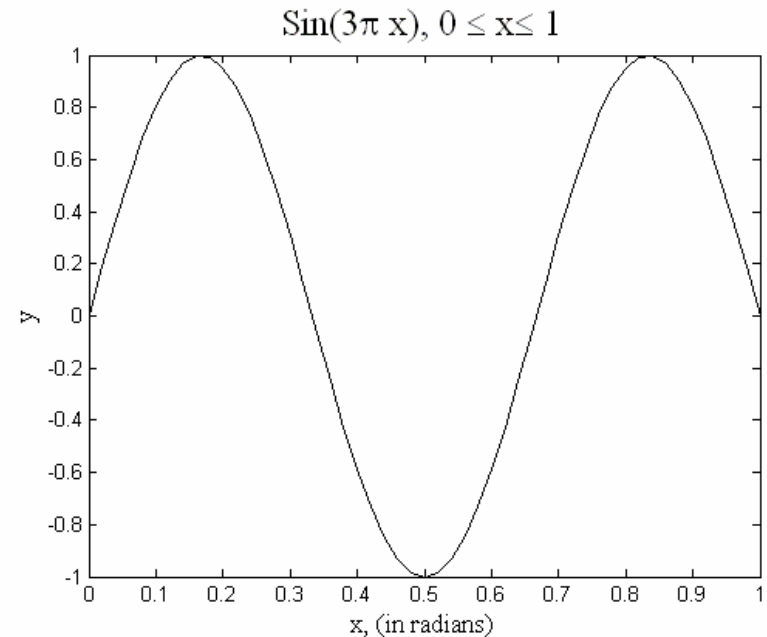


Basic plots

Plot of $f(x) = \sin(3\pi x)$ using 51 points on the interval $[0,1]$:

```
>> x=[0:1/50:1];  
>> y=sin(3*pi*x);  
>> plot(x,y,'k-');
```

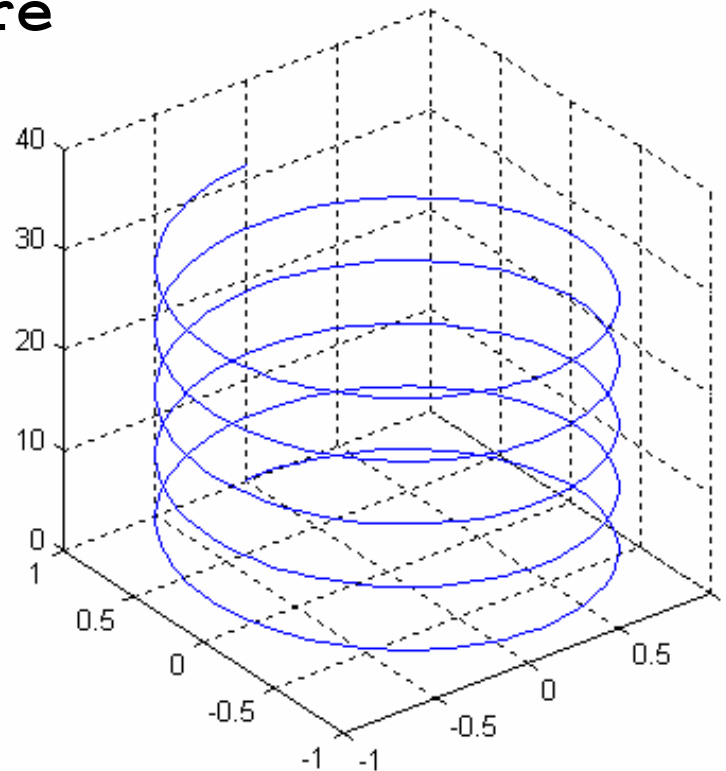
And then, you can fancy it up with a title, an x-axis label, an y-axis label, etc.





3-D Line Plots

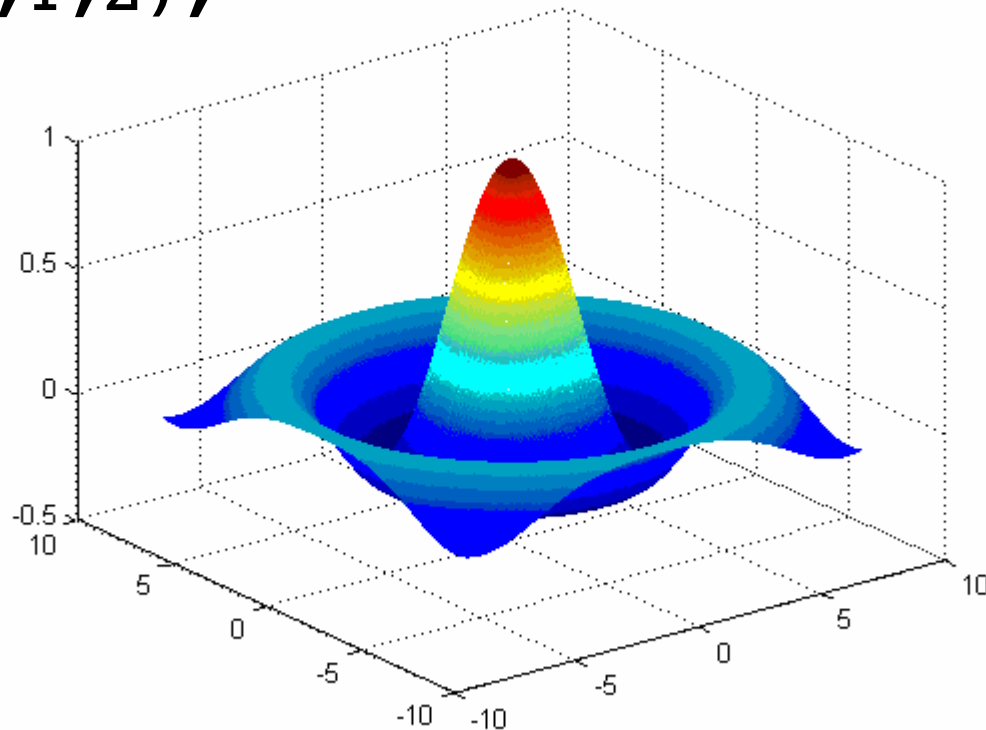
```
>> t = 0:pi/50:10*pi;  
>> plot3(sin(t),cos(t),t);  
>> grid on  
>> axis square
```





3-D Graphics: mesh

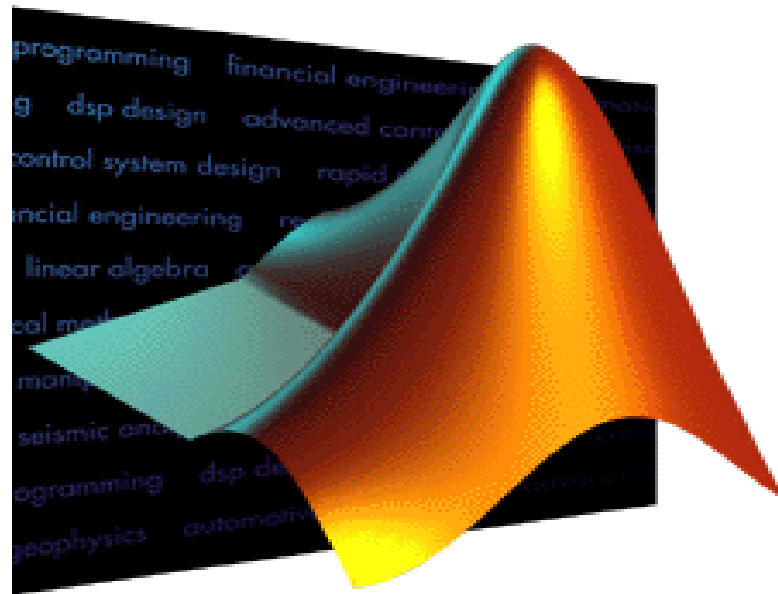
```
>> [X,Y] = meshgrid(-8:0.05:8);  
>> R = sqrt(X.^2 + Y.^2) + eps;  
>> Z = sin(R)./R;  
>> mesh(X,Y,Z);
```





MATLAB HELP!

The most important feature in MATLAB...
the `help` command!



Great documentations, demos etc.! Use it!



Example

- Last year's practical assignment:
The ***Second Harmonic Generation...***



The Model

- Modeling of the *laser field*:

$$E(t) = \int_{-\infty}^{\infty} A(\omega) \cdot e^{i[\omega t + \Phi(\omega)]} d\omega$$

- We interpolate $\Phi(\omega)$, the control (phase) function, in the frequency space (discretization is up to us!).
- The objective function is given by (*maximization*):

$$SHG = \int E^4(t) dt$$



The power of MATLAB

The core of the SHG – implemented only in 2 lines!

```
%Fourier Transform with phase on the Gaussian Ain  
E_t = fftshift(ifft(fftshift(exp(i*phase).*Ain)));  
  
%Integrate the result to yield the SHG  
SHG = sum(abs(E_t).^4);
```

This would take many more lines in C or C++!



Demo

An implementation of an EA for the SHG optimization problem in MATLAB.