

MATLAB Introductory courses

Lecture 2

Siamak Ghorbani Faal
sghorbanifaal@wpi.edu



WPI
IT - Academic and Research Computing Center

Course outlines

- Lecture 1: Introduction to MATLAB
- Lecture 2: Using MATLAB
- Lecture 3: Visualization and Specialized Tools

Current lecture outlines

Plotting and Visualization

Functions

Save/Load workspace

Root finding

Function zeros

Polynomial fitting

Interpolation

Numerical integration

Plot

- Plotting: $x = f(t)$
- Define x as a vector
- Compute value of y for each element of x
- Plot these two vectors

$$\begin{array}{l} x = [\boxed{0}, \boxed{0.1}, \boxed{0.2}, \boxed{0.3}, \dots, \boxed{1}] \\ y = [\boxed{f(0)}, \boxed{f(0.1)}, \boxed{f(0.2)}, \boxed{f(0.3)}, \dots, \boxed{f(1)}] \end{array}$$

1 2 3 4 ... 11

Plot

```
>> plot(t, x)
```

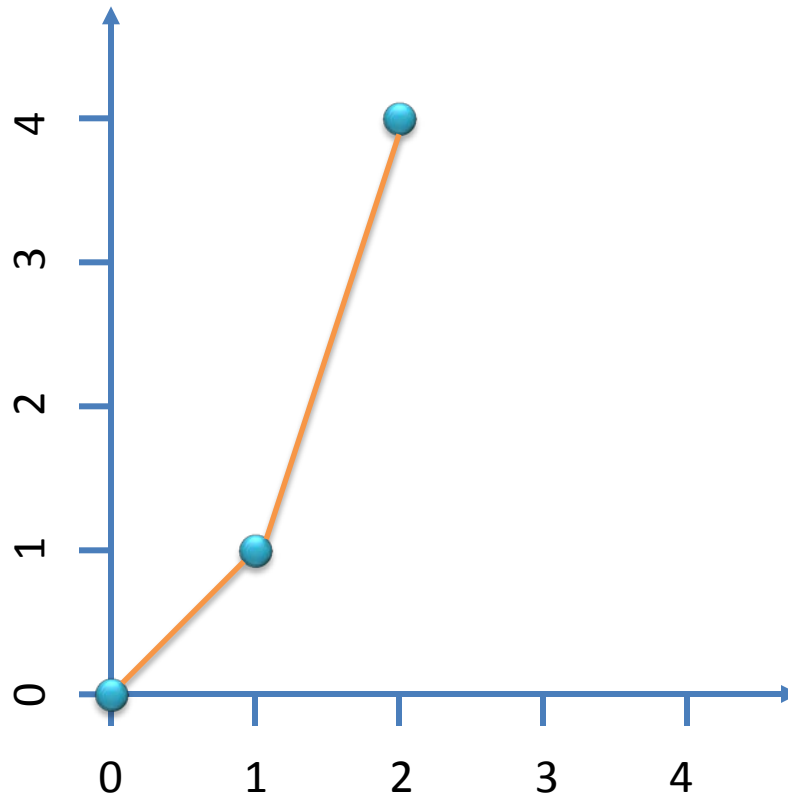
Example:

$t = [0, 1, 2]$

$x = t^2$

$x = [0, 1, 4]$

$points = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 4 \end{bmatrix}$



Plot

```
>> plot(Y)
>> plot(X1,Y1,...,Xn,Yn)
>> plot(X1,Y1,LineSpec,...,Xn,Yn,LineSpec)
>> plot(...,'PropertyName',PropertyValue,...)

>> hold on
>> hold off

>> plotyy
>> subplot

>> figure()
>> f1 = figure();
>> figure(f1);
```

plot_test_1.m

Editor - C:\Users\sghorbanifaal\Documents\MATLAB\L2\plot_test.m

File Edit Text Go Cell Tools Debug Desktop Window Help



+ - 1.0 + ÷ 1.1 × % % !

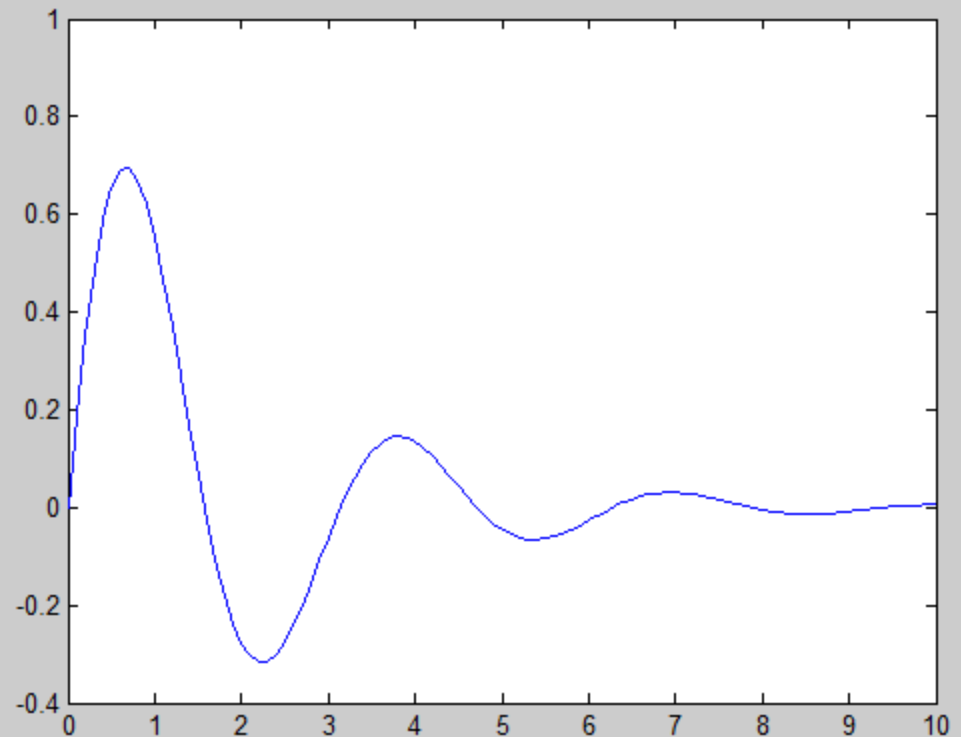
```
1 - clc; clear all; close all;  
2  
3 - t = 0:0.1:10;  
4  
5 - x = exp(-0.5*t).*sin(2*t);  
6  
7  
8 - plot(t,x)
```

Check what happens in each of the following conditions:

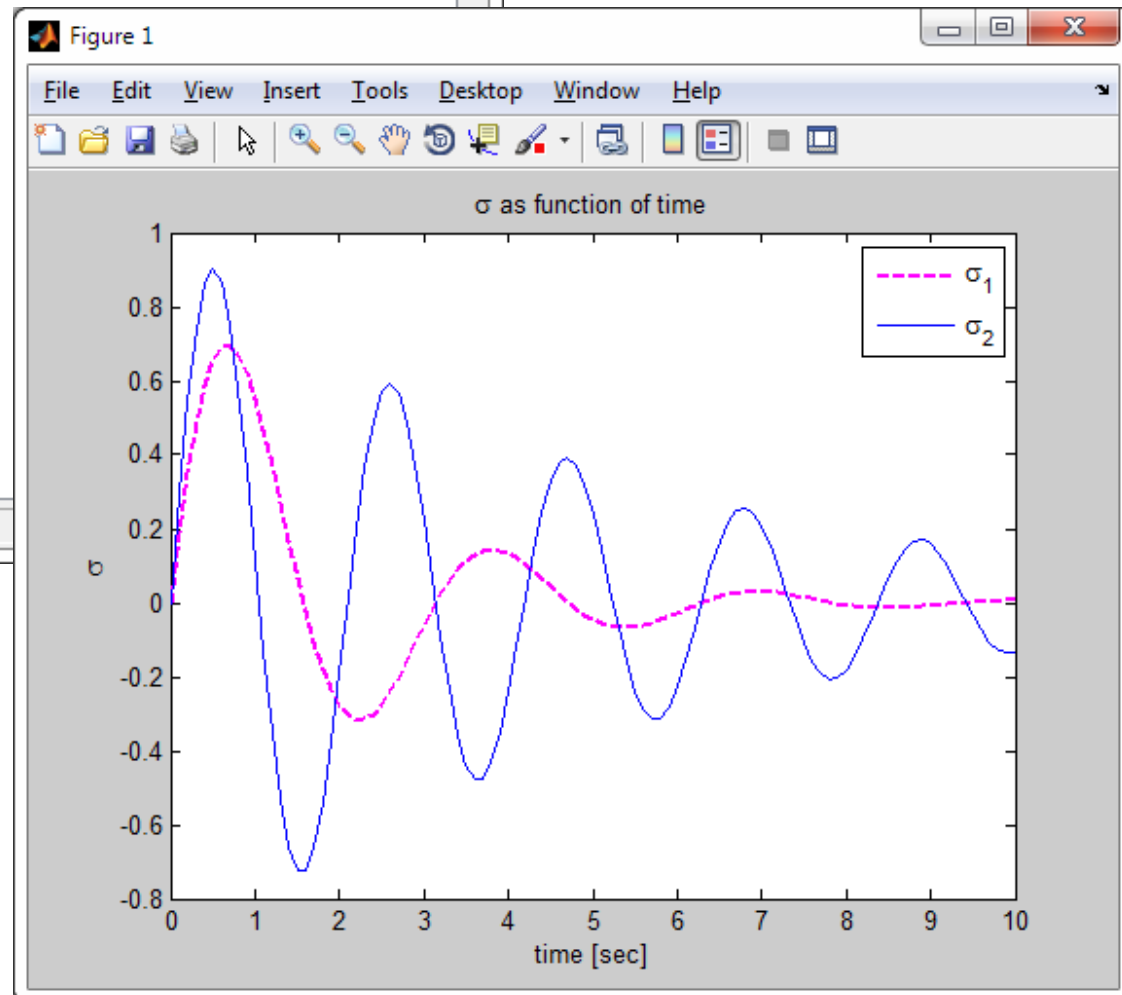
```
>> plot(x)  
>> plot(t)  
>> t = 0:1:10
```

Figure 1

File Edit View Insert Tools Desktop Window Help




```
Editor - C:\Users\sghorbanifaal\Documents\MATLAB\L2\plot_test.m
File Edit Text Go Cell Tools Debug Desktop Window Help
+ - 1.0 + ÷ 1.1 × % % %
1 - clc; clear all; close all;
2
3 - t = 0:0.1:10;
4
5 - x1 = exp(-0.5*t).*sin(2*t);
6 - x2 = exp(-0.2*t).*sin(3*t);
7
8 - plot(t,x1,'--m','LineWidth',2)
9
10 - hold on
11
12 - plot(t,x2,'-b');
13
14 - xlabel('time [sec]');
15 - ylabel('\sigma');
16 - title('\sigma as function of time');
17 - legend('\sigma_1','\sigma_2');
18
script
```



Subplot

- Create axes in tiled positions

```
>> h = subplot(m,n,p)
```

```
>> subplot(m,n,P)
```

```
>> subplot(2, 3, 5)
```

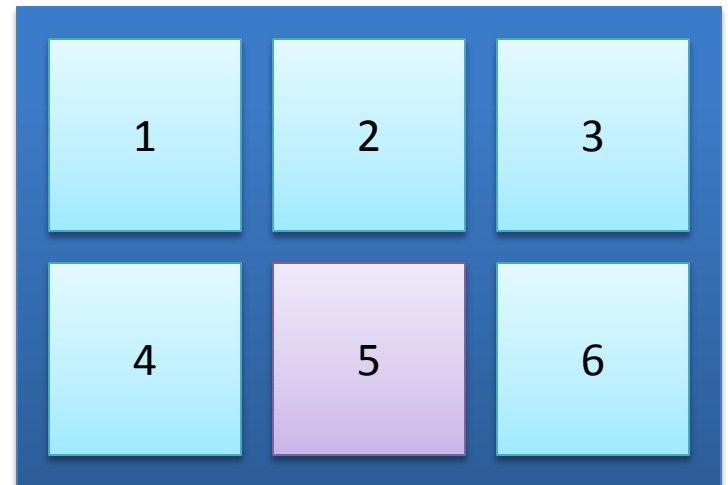
Numbering order:

Left to right

Top to bottom



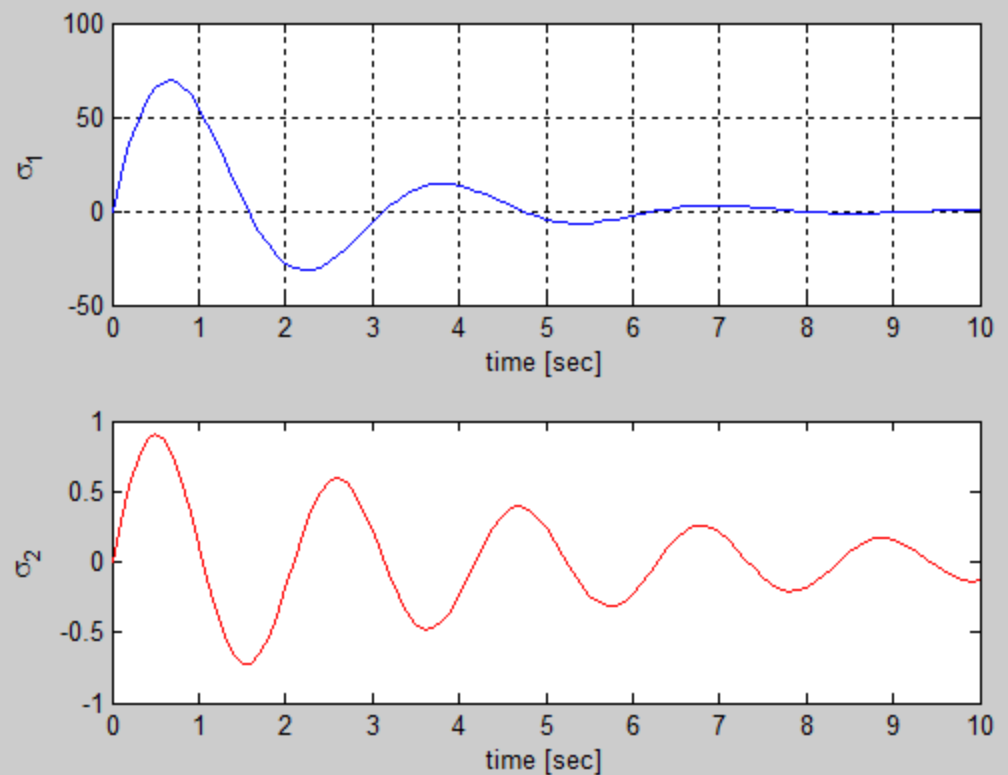
2 rows, 3 columns



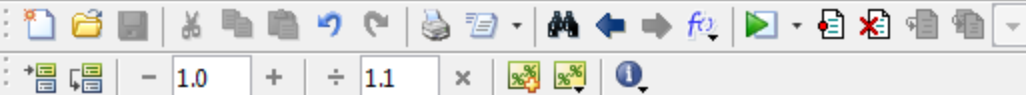
plot_test_5.m

22

23



File Edit Text Go Cell Tools Debug Desktop Window Help

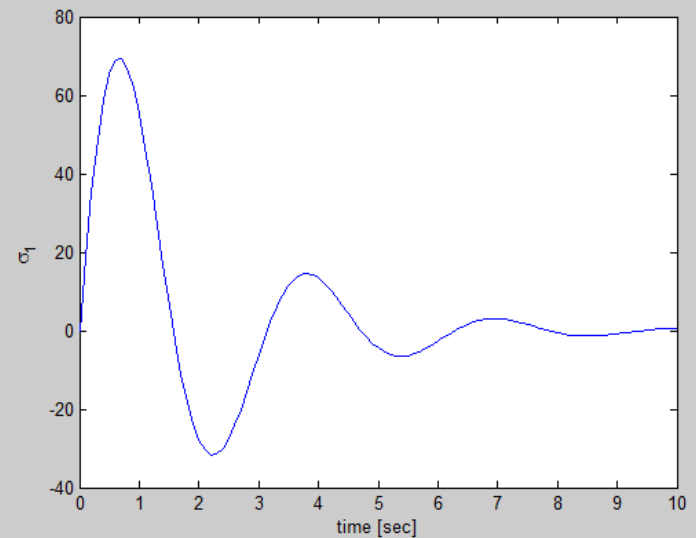

+ 1.0 + ÷ 1.1 × % % ?

```
1 - clc; clear all; close all;  
2  
3 - t = 0:0.1:10;  
4  
5 - x1 = 100*exp(-0.5*t).*sin(2*t);  
6 - x2 = exp(-0.2*t).*sin(3*t);  
7  
8  
9 - f1 = figure();  
10 - plot(t,x1,'b');  
11 - xlabel('time [sec]');  
12 - ylabel('\sigma_1');  
13  
14  
15 - f2 = figure();  
16 - plot(t,x2,'r');  
17 - xlabel('time [sec]');  
18 - ylabel('\sigma_2');  
19  
20  
21 - figure(f2);  
22 - grid on  
23
```

script

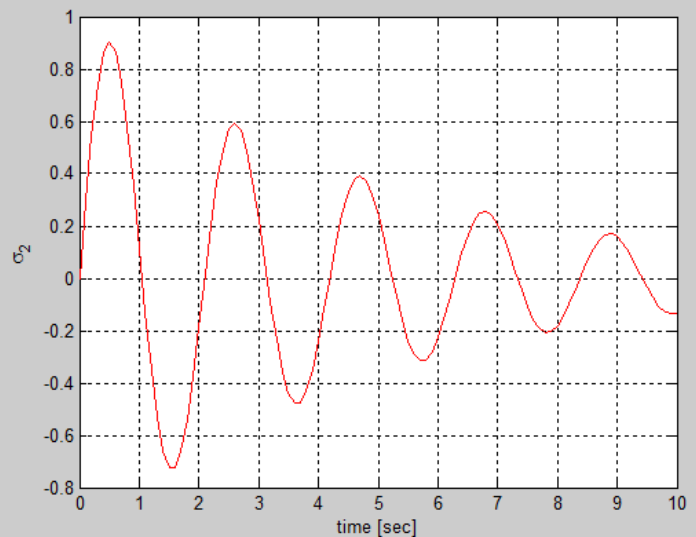
Ln 20 Col

File Edit View Insert Tools Desktop Window Help

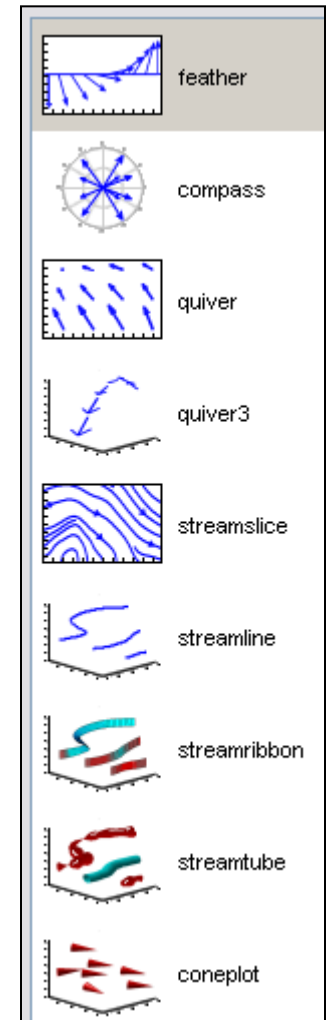
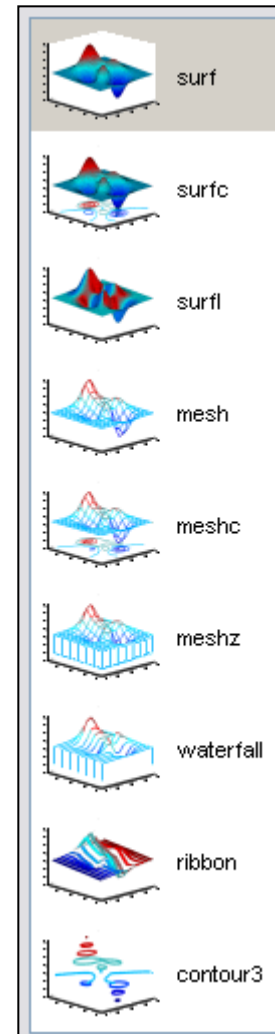
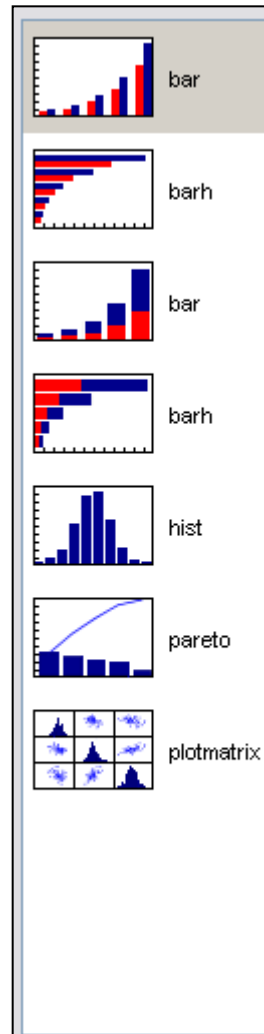
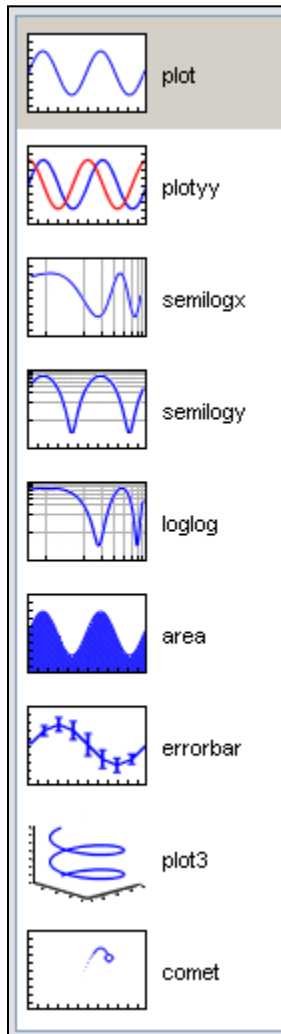


plot_test_6.m

File Edit View Insert Tools Desktop Window Help



Different plots

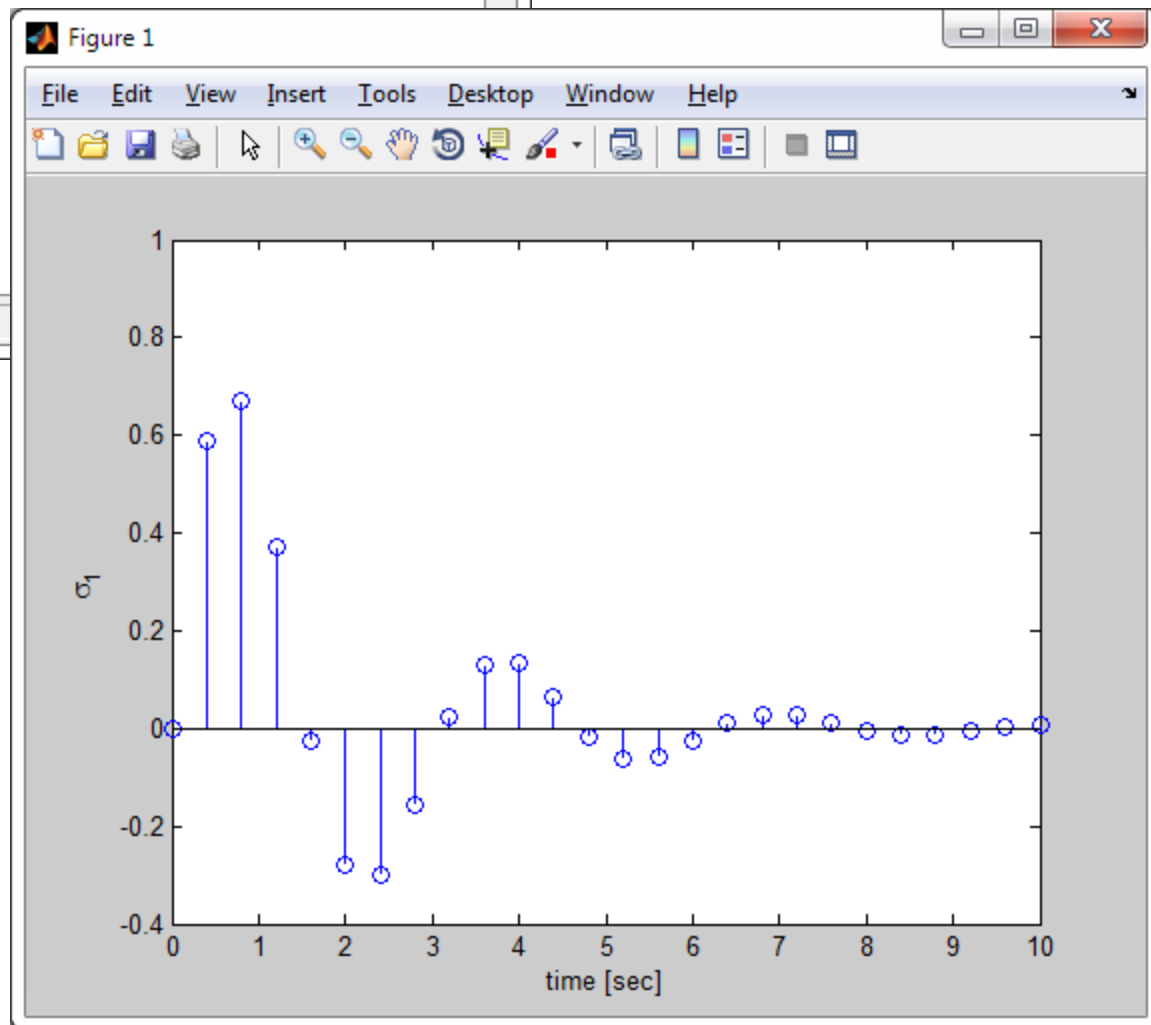


Editor - C:\Users\sghorbanifaal\Documents\MATLAB\L2\plot_test.m

```
File Edit Text Go Cell Tools Debug Desktop Window Help
+ - 1.0 + ÷ 1.1 x
1 - clc; clear all; close all;
2
3 - t = 0:0.4:10;
4
5 - x = exp(-0.5*t).*sin(2*t);
6
7 - stem(t,x,'b');
8
9 - xlabel('time [sec]');
10 - ylabel('\sigma_1');
11
```

script

stem_test.m

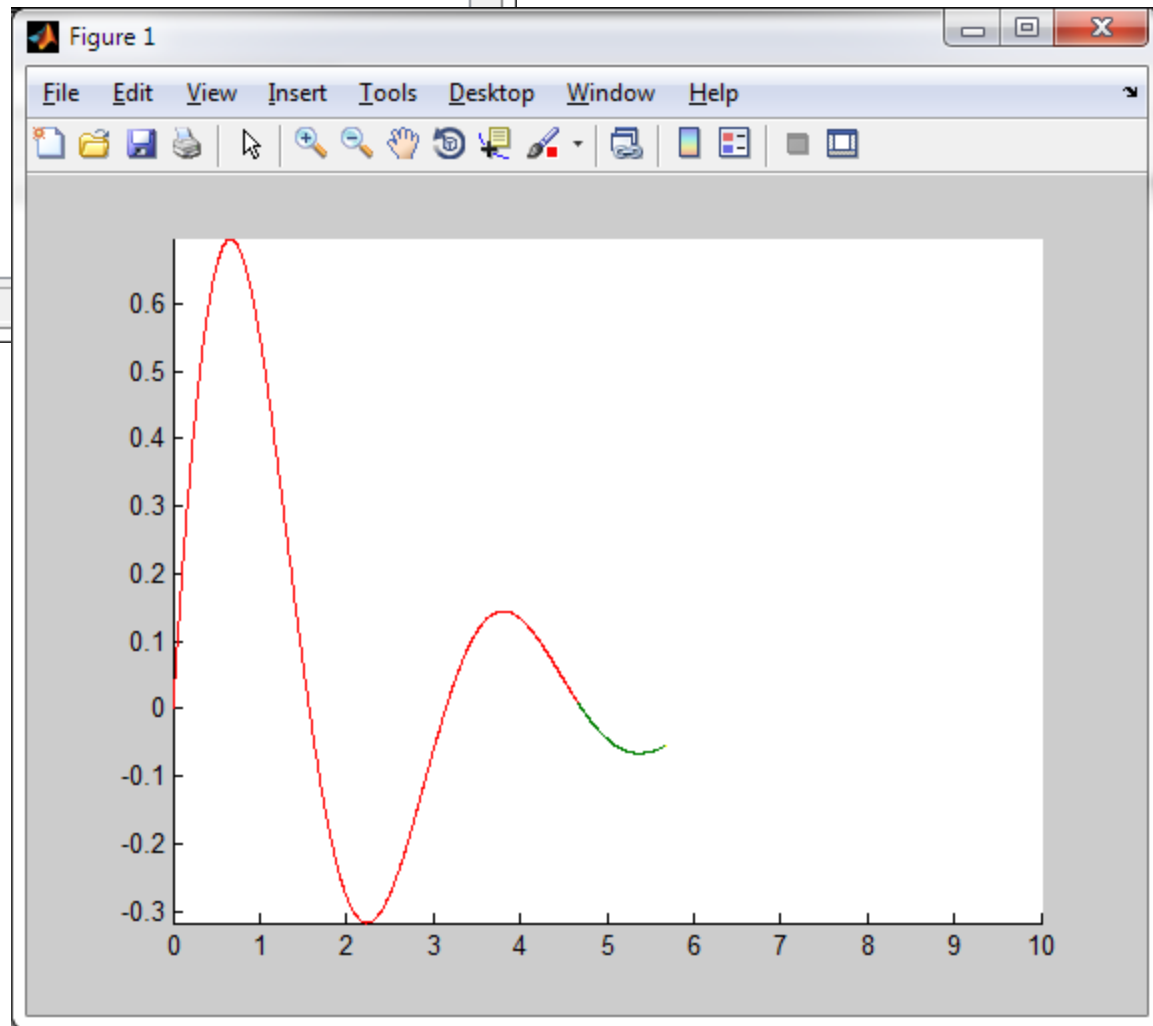


Editor - C:\Users\sghorbanifaal\Documents\MATLAB\L2\plot_test.m

```
File Edit Text Go Cell Tools Debug Desktop Window Help
+ [Icons] - 1.0 + ÷ 1.1 × [Icons]
1 - clc; clear all; close all;
2
3 - t = 0:0.005:10;
4
5 - x = exp(-0.5*t).*sin(2*t);
6
7 - comet(t,x,0.1);
```

script

comet_test.m



plot3

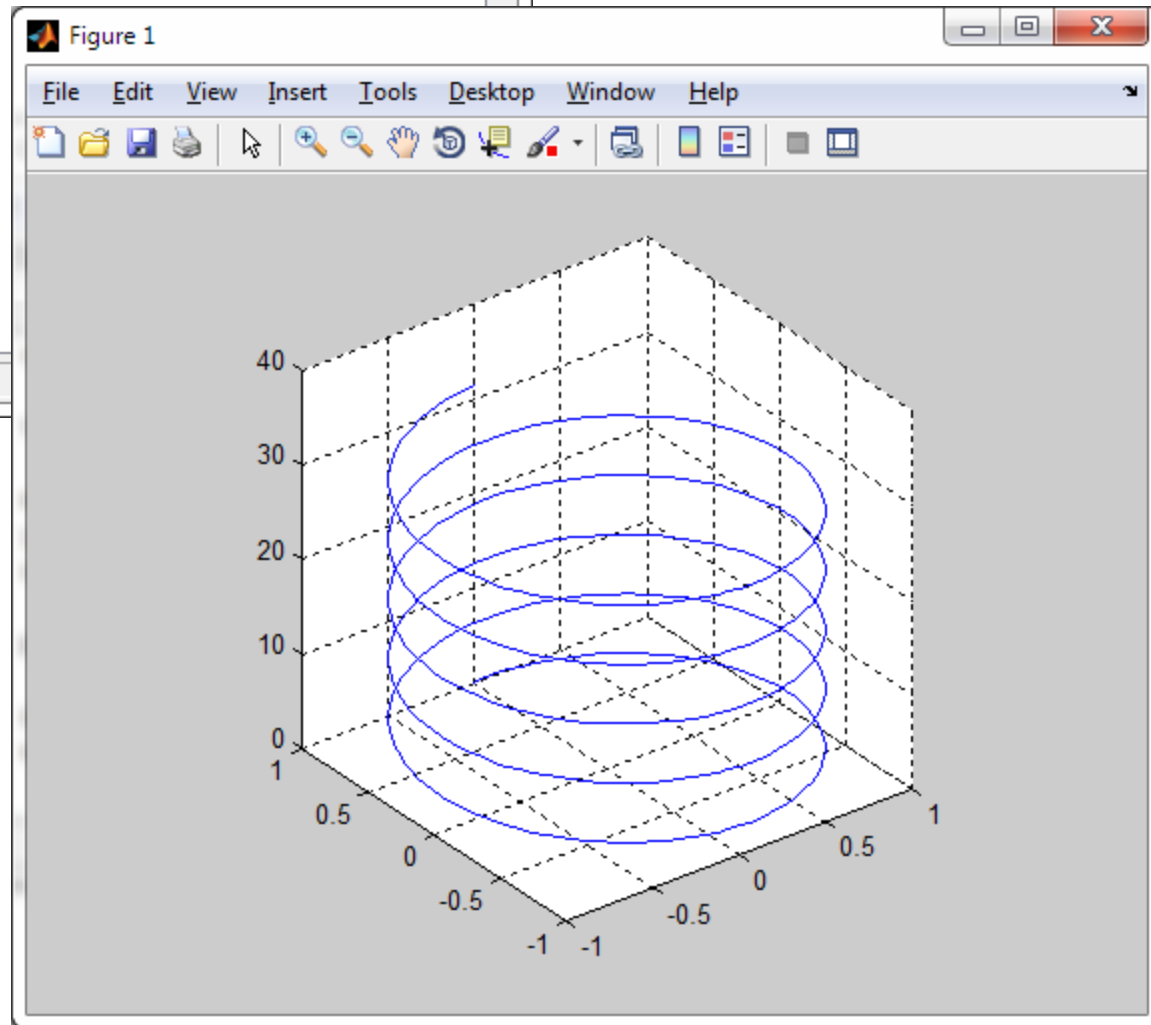
- 3D line plot

```
>> plot3(X1,Y1,Z1,...)
>> plot3(X1,Y1,Z1,LineSpec,...)
>> plot3(...,'PropertyName',PropertyValue,...)
>> h = plot3(...)
```

plot3_test.m

```
Editor - C:\Users\sghorbanifaal\Documents\MATLAB\L2\plot_test.m
File Edit Text Go Cell Tools Debug Desktop Window Help
+ - 1.0 + ÷ 1.1 × % % %
1 - clc; clear all; close all;
2
3 - t = 0:pi/50:10*pi;
4
5 - x = cos(t);
6 - y = sin(t);
7 - z = t;
8
9 - plot3(sin(t),cos(t),t)
10
11 - grid on
12 - axis square
script
```

Check what happens if you
change **square** by **equal**

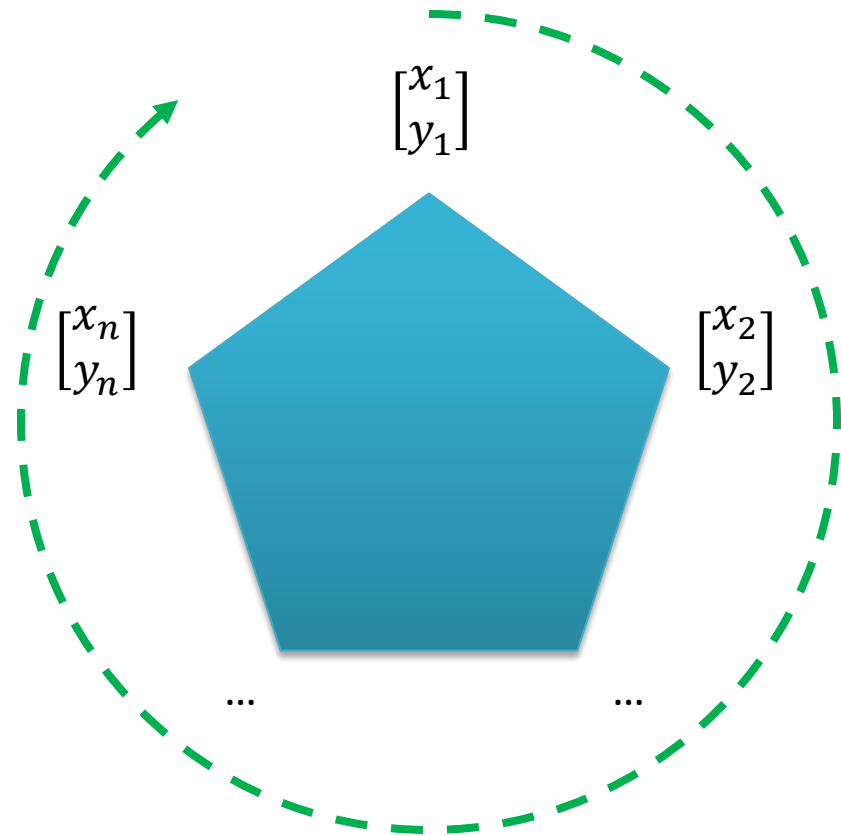


fill & fill3

- fill 2D polygon fill
- fill3 3D polygon fill

```
x = [ x1 , x2 , ... , xn , x1 ]  
y = [ y1 , y2 , ... , yn , y1 ]
```

```
Fill( x , y , color)
```

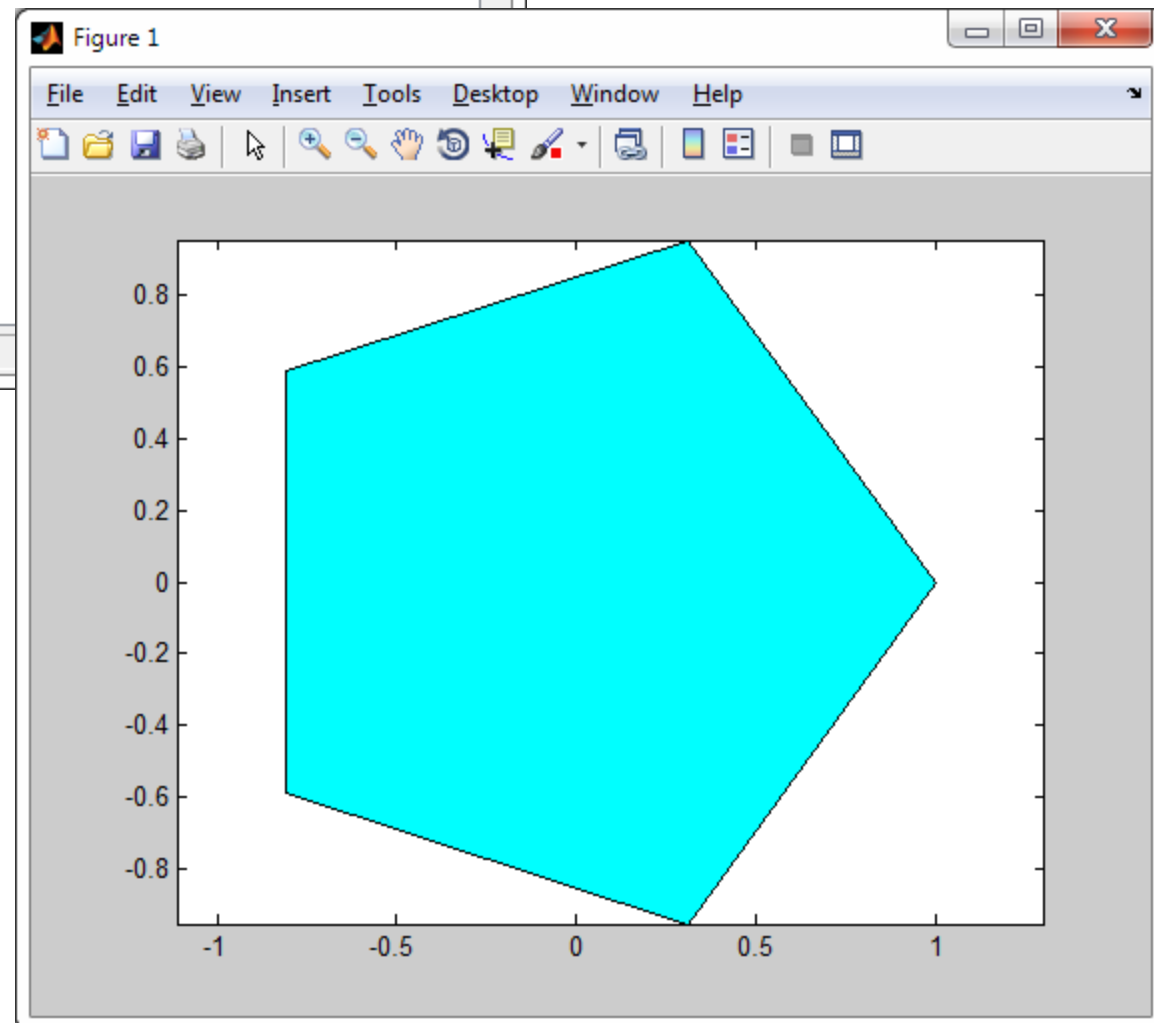


Editor - C:\Users\sghorbanifaa\Documents\MATLAB\L2\plot_test.m

```
File Edit Text Go Cell Tools Debug Desktop Window Help
1 - clc; clear all; close all;
2
3 - t = 0:72:360;
4
5 - x = cosd(t);
6 - y = sind(t);
7
8 - fill(x,y,'c')
9
10 - axis equal
```

script

fill_test.m



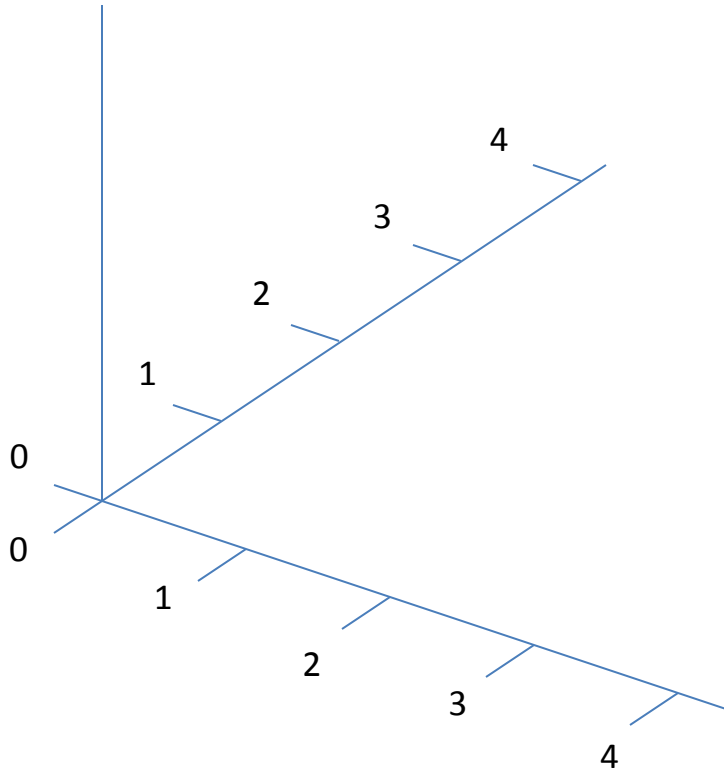
Mesh grid

- Steps required to generate a 3D surface

$$z = f(x, y)$$

$$0 \leq x \leq 4$$

$$0 \leq y \leq 4$$



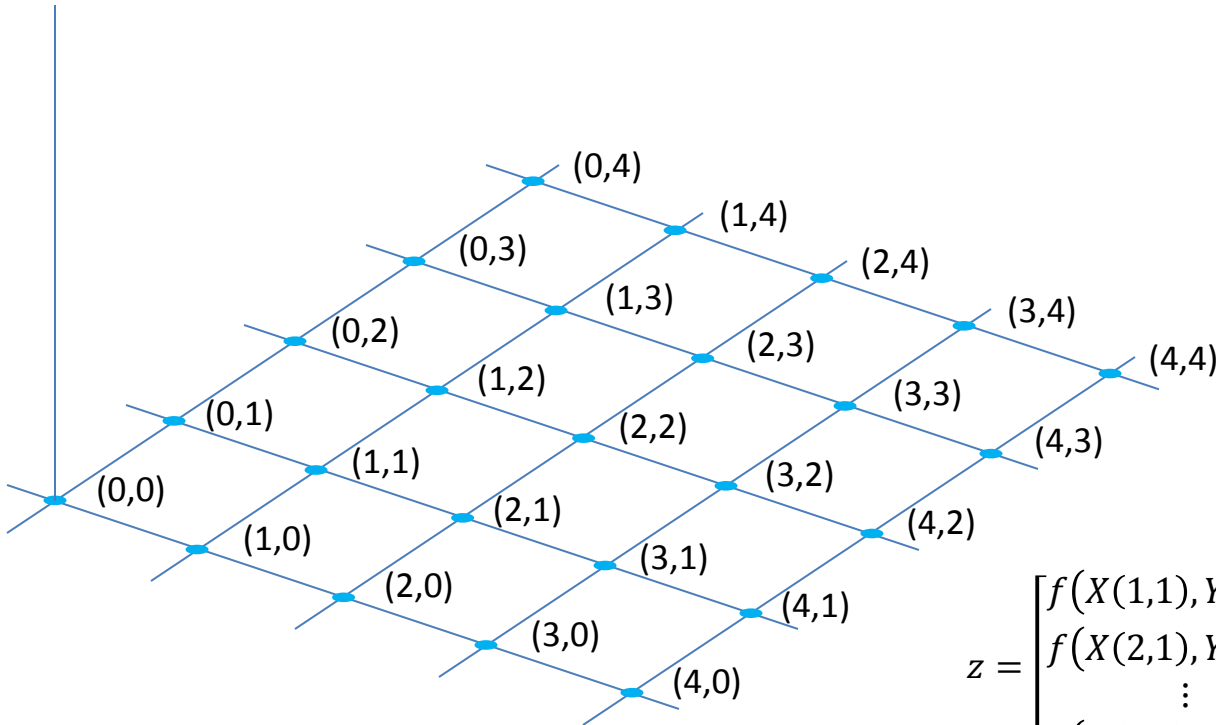
Mesh grid

$$z = f(x, y)$$

$$0 \leq x \leq 4$$

$$0 \leq y \leq 4$$

- Steps required to generate a 3D surface



$$X = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 \end{bmatrix}$$

$$Y = \begin{bmatrix} 4 & 4 & 4 & 4 & 4 \\ 3 & 3 & 3 & 3 & 3 \\ 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$z = \begin{bmatrix} f(X(1,1), Y(1,1)) & \dots & f(X(1,5), Y(1,5)) \\ f(X(2,1), Y(2,1)) & & \\ \vdots & \ddots & \vdots \\ f(X(5,1), Y(5,1)) & & f(X(5,5), Y(5,5)) \end{bmatrix}$$

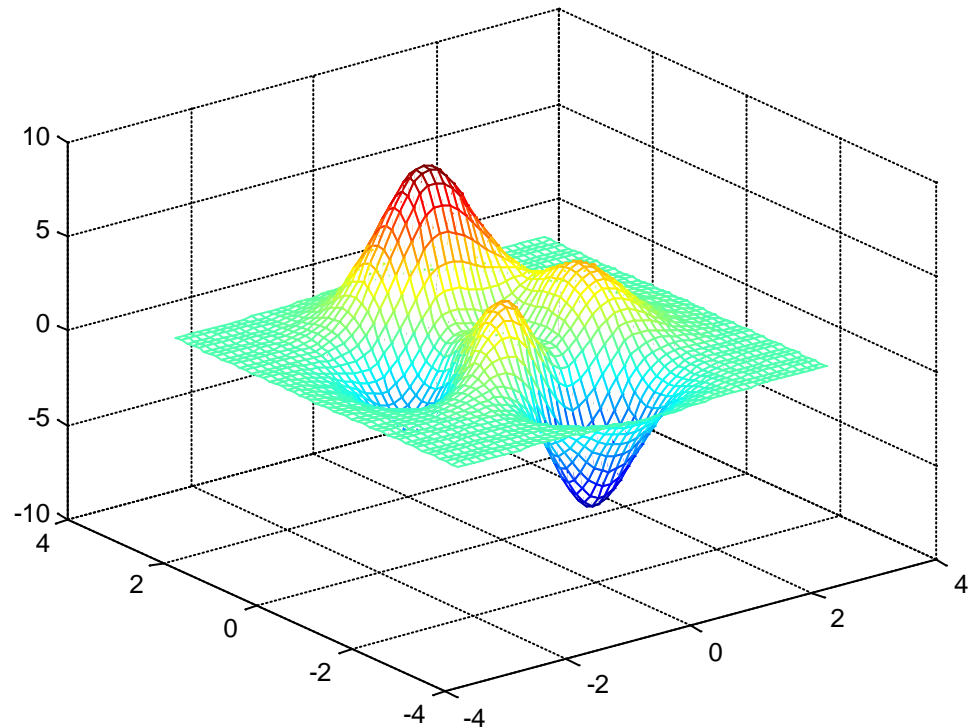
Mesh grid

```
>> x = [0 1 2 3];  
>> y = [0 1 2 3];  
>> [X,Y] = meshgrid(x,y);  
>> z = X.^2 + Y.^2;
```

```
Command Window  
  
>> x = [0 1 2 3];  
>> y = [0 1 2 3];  
>> [X,Y] = meshgrid(x,y)  
  
X =  
  
    0    1    2    3  
    0    1    2    3  
    0    1    2    3  
    0    1    2    3  
  
Y =  
  
    0    0    0    0  
    1    1    1    1  
    2    2    2    2  
    3    3    3    3  
  
>> z = X.^2 + Y.^2  
  
z =  
  
    0    1    4    9  
    1    2    5   10  
    4    5    8   13  
    9   10   13   18  
  
fx >>
```

Mesh

- `mesh(X,Y,Z)` draws a wireframe mesh with color determined by Z, so color is proportional to surface height.



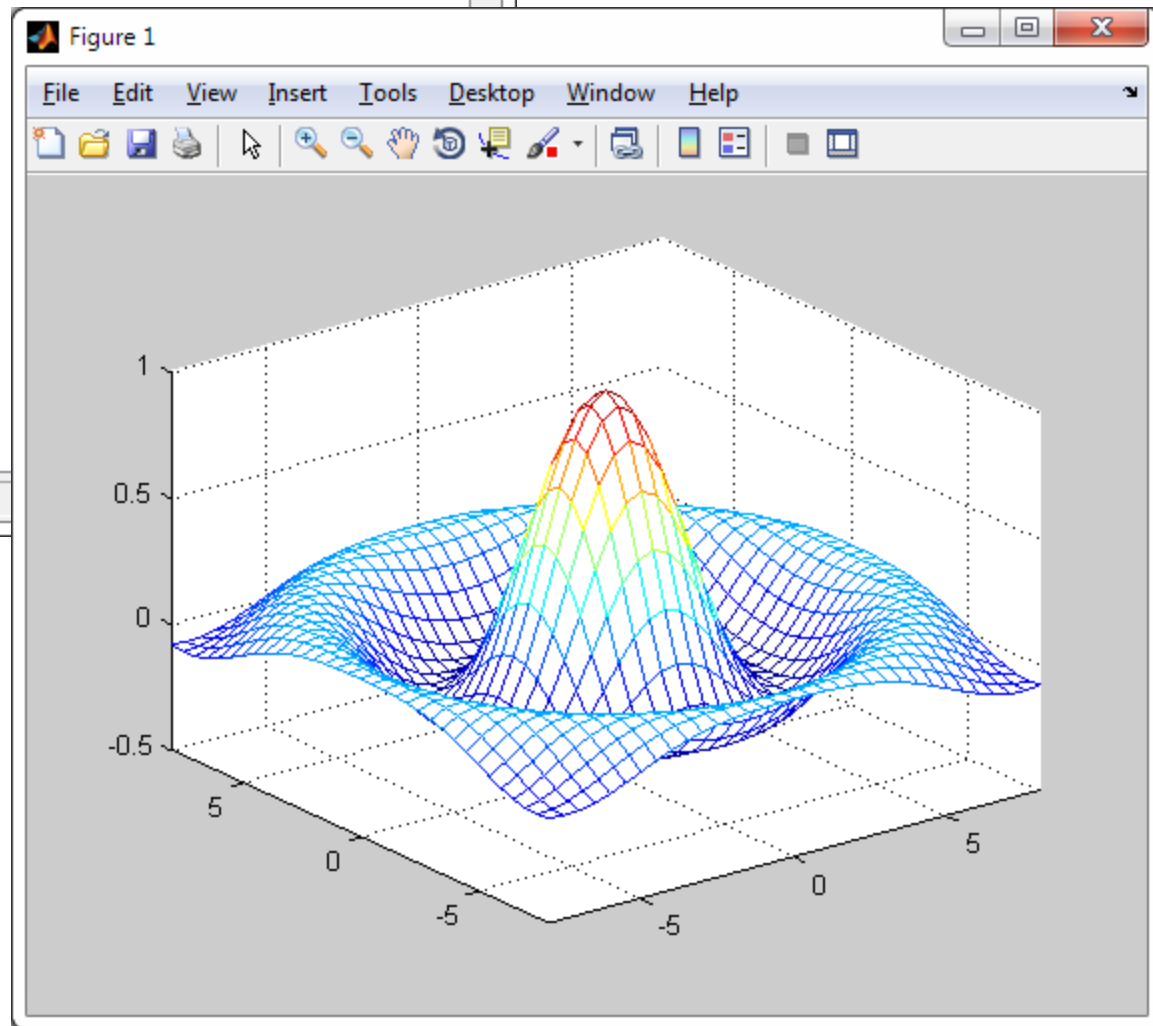
Editor - C:\Users\sghorbanifaal\Documents\MATLAB\L2\plot_test.m*

File Edit Text Go Cell Tools Debug Desktop Window Help

1 - `clc; clear all; close all;`
2 -
3 - `x = -8:0.5:8;`
4 - `y = -8:0.5:8;`
5 -
6 - `[X,Y] = meshgrid(x,y);`
7 -
8 - `R = sqrt(X.^2 + Y.^2) + eps;`
9 -
10 - `Z = sin(R)./R;`
11 -
12 - `mesh(X,Y,Z);`
13 -
14 - `axis([-8 8 -8 8 -0.5 1]);`

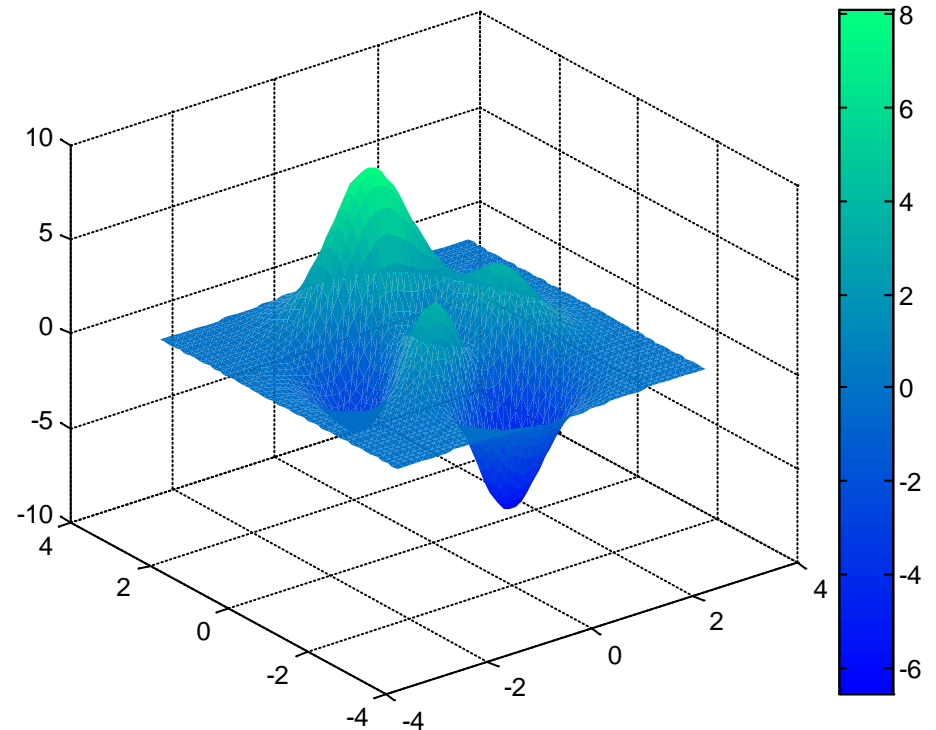
script

mesh_test.m



Surf

- `surf(Z)` creates a three-dimensional shaded surface from the z components in matrix Z , using $x = 1:n$ and $y = 1:m$, where $[m,n] = \text{size}(Z)$.



Editor - C:\Users\sghorbanifaa\Documents\MATLAB\L2\plot_test.m

surf_test.m

File Edit Text Go Cell Tools Debug Desktop Window Help

→                                                                                                                     

```
1 -      clc; clear all; close all;
2 -
3 -      x = -8:0.5:8;
4 -      y = -8:0.5:8;
5 -
6 -      [X,Y] = meshgrid(x,y);
7 -
8 -      R = sqrt(X.^2 + Y.^2) + eps;
9 -
10 -     Z = sin(R)./R;
11 -
12 -     surf(X,Y,Z);
13 -
14 -     axis([-8 8 -8 8 -0.5 1]);
15 -
16 -     shading interp
17 -
18 -     colormap jet
19 -
20 -     colorbar
```

script

```
>> doc shading
>> doc colormap
```


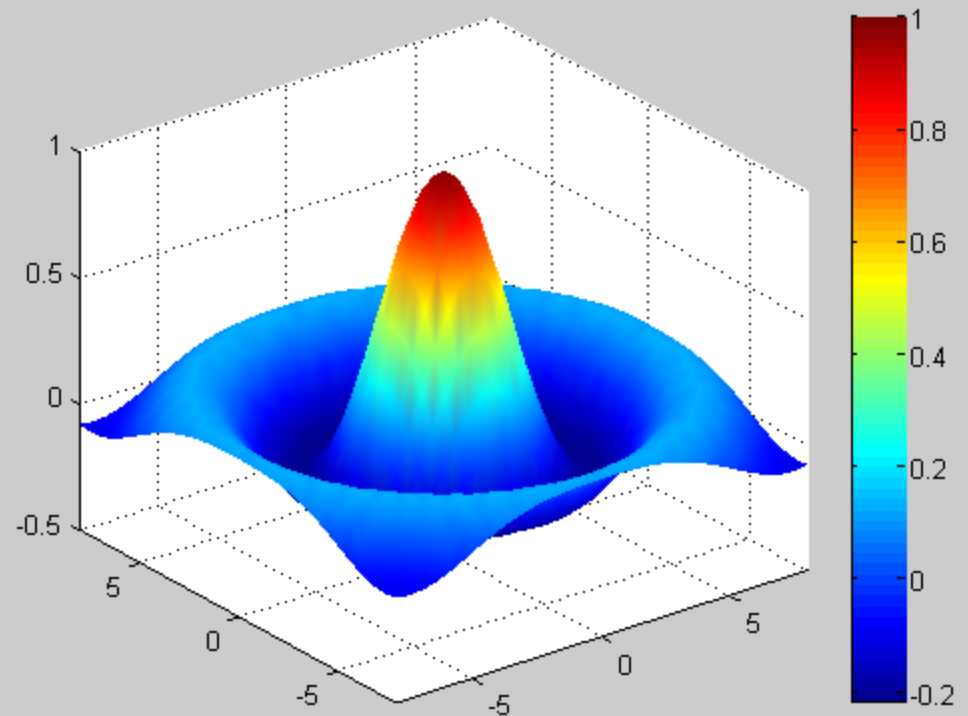


Figure 1

File Edit View Insert Tools Desktop Window Help



Contour

- A contour plot displays isolines of matrix Z .

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

draws a contour plot of matrix Z with n contour levels where n is a scalar.

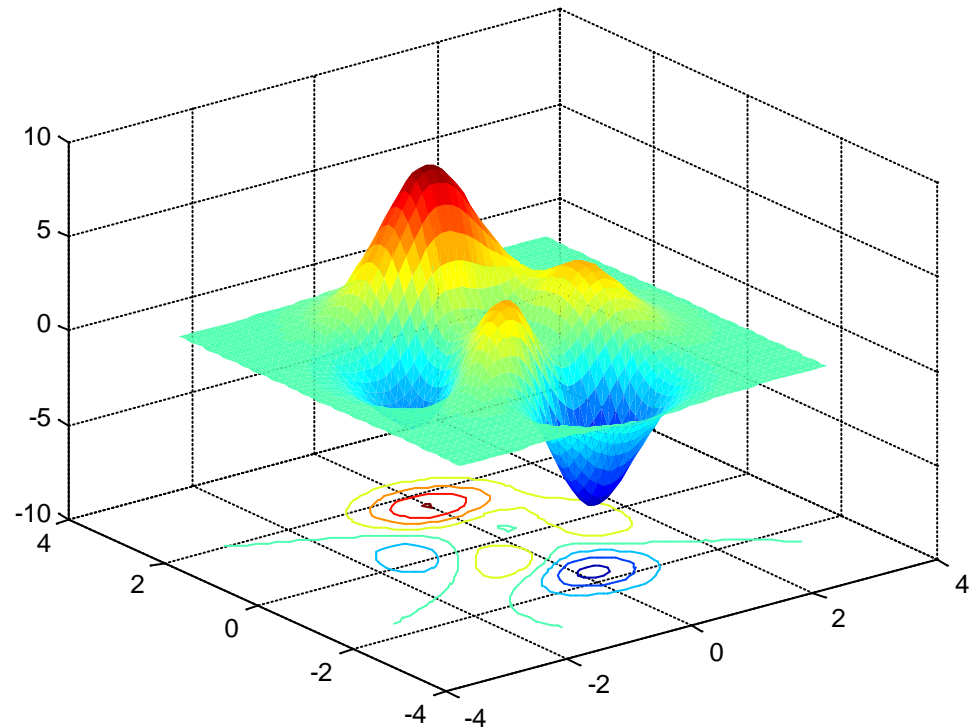
```
>> contour(X,Y,Z,v)
```

draws a contour plot of matrix Z with contour lines at the data values specified in the monotonically increasing vector v . The number of contour levels is equal to $\text{length}(v)$. To draw a single contour of level i , use:

```
contour(Z,[i i]).
```

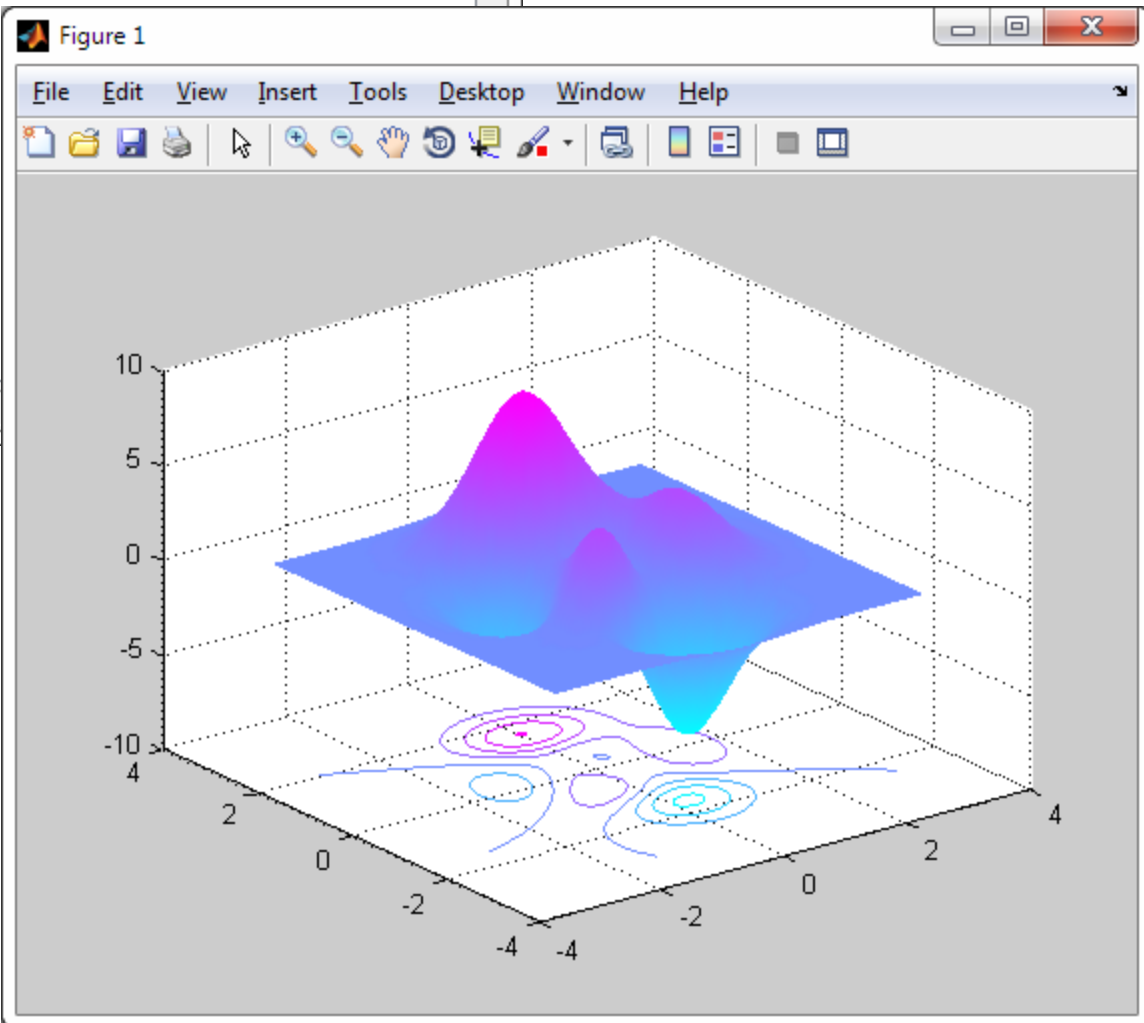
Surfc

- `surfc(Z)` creates a contour plot under the three-dimensional shaded surface from the z components in matrix Z.



surfc_test.m

script

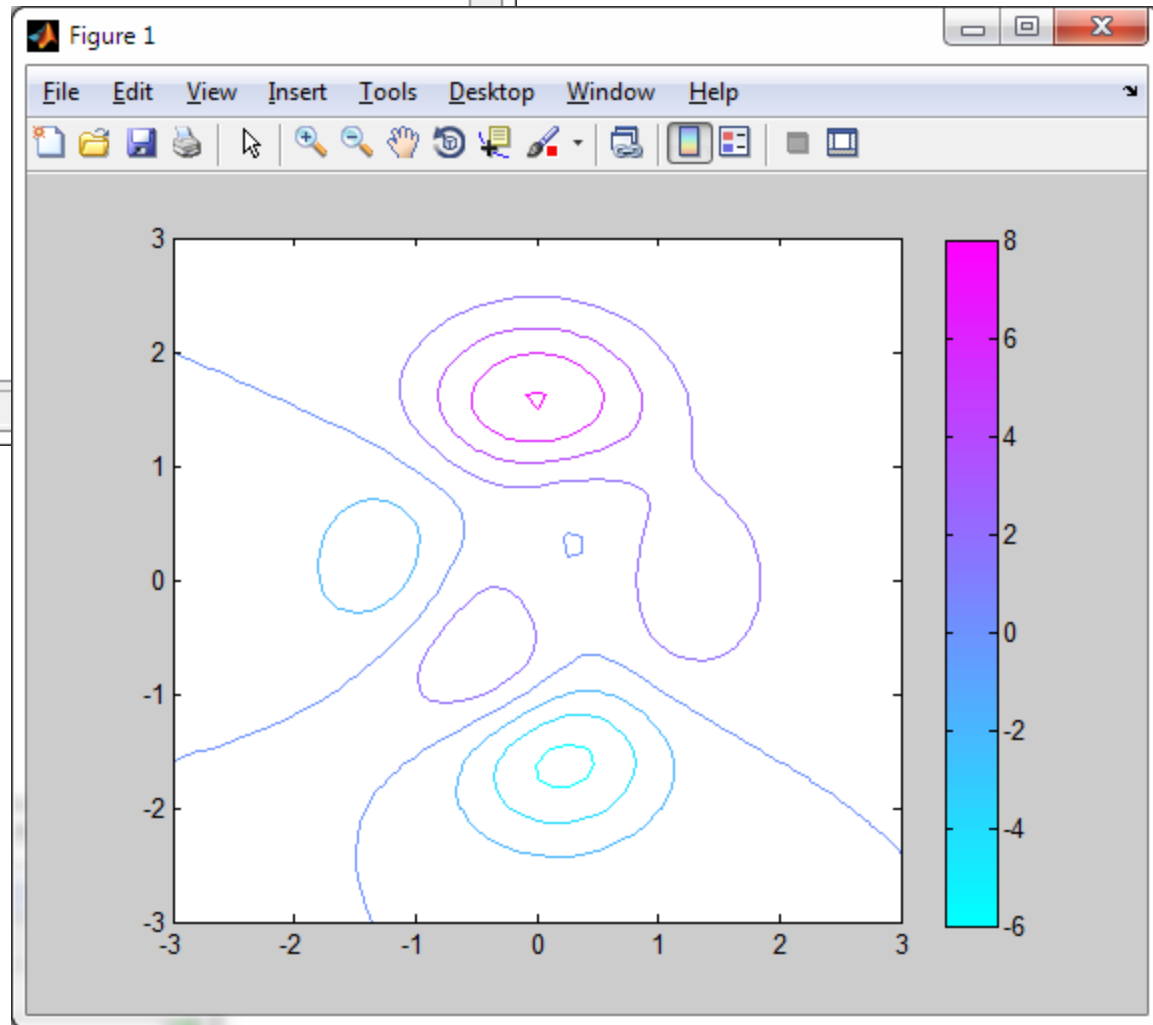


Editor - C:\Users\sghorbanifaal\Documents\MATLAB\L2\plot_test3.m

```
File Edit Text Go Cell Tools Debug Desktop Window Help
+ - 1.0 + ÷ 1.1 × % % %
1 - clc; clear all; close all;
2
3 - [X,Y,Z] = peaks;
4
5 - contour(X,Y,Z)
6
7 - colormap cool
8 - colorbar
```

script

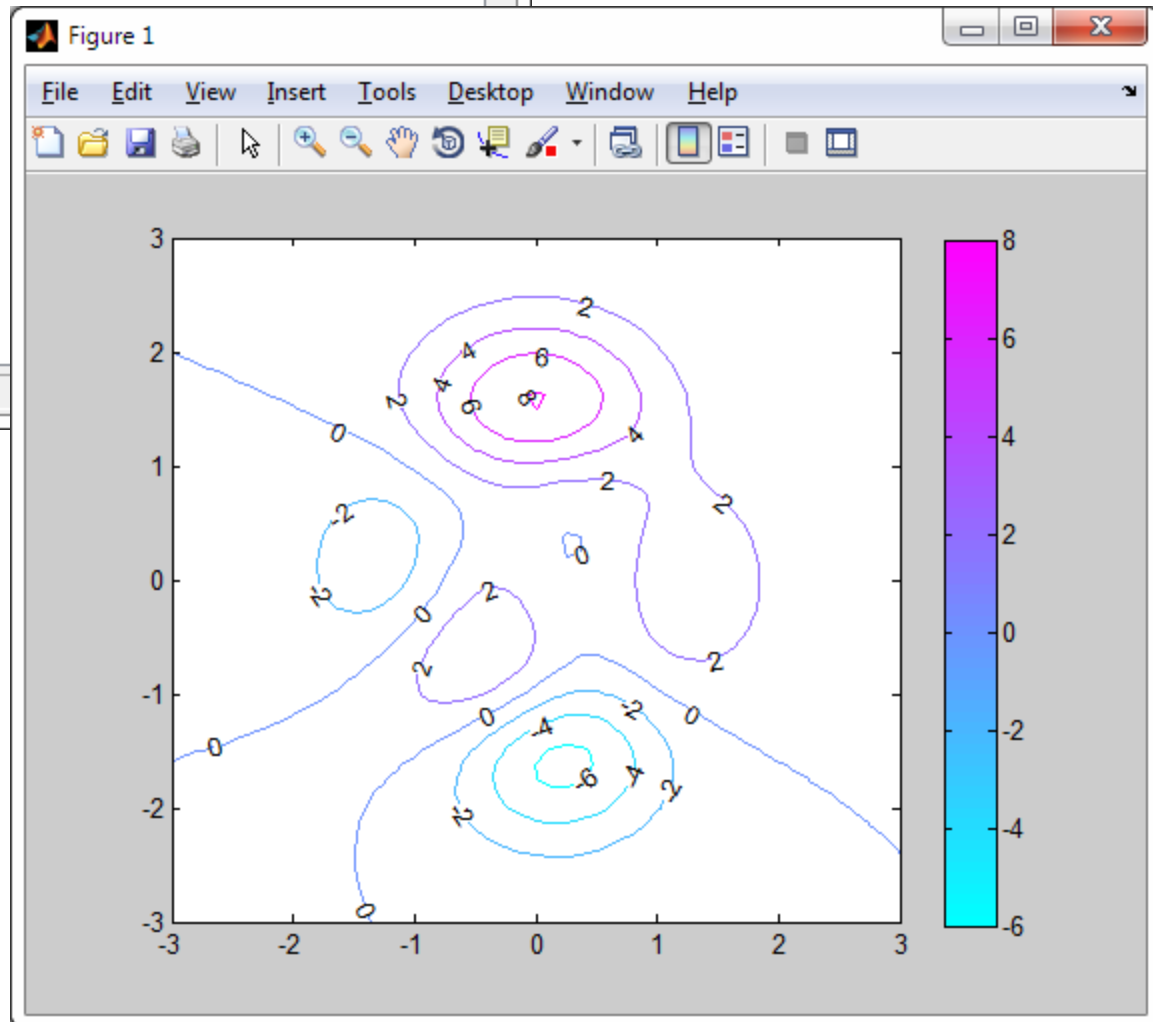
contour_test_1.m



Editor - C:\Users\sghorbanifaal\Documents\MATLAB\L2\plot_test3.m

```
File Edit Text Go Cell Tools Debug Desktop Window Help
+ [Icons] - 1.0 + ÷ 1.1 x [Icons]
1 - clc; clear all; close all;
2
3 - [X,Y,Z] = peaks;
4
5 - [C, h] = contour(X,Y,Z);
6
7 - clabel(C,h);
8
9 - colormap cool
10 - colorbar
script
```

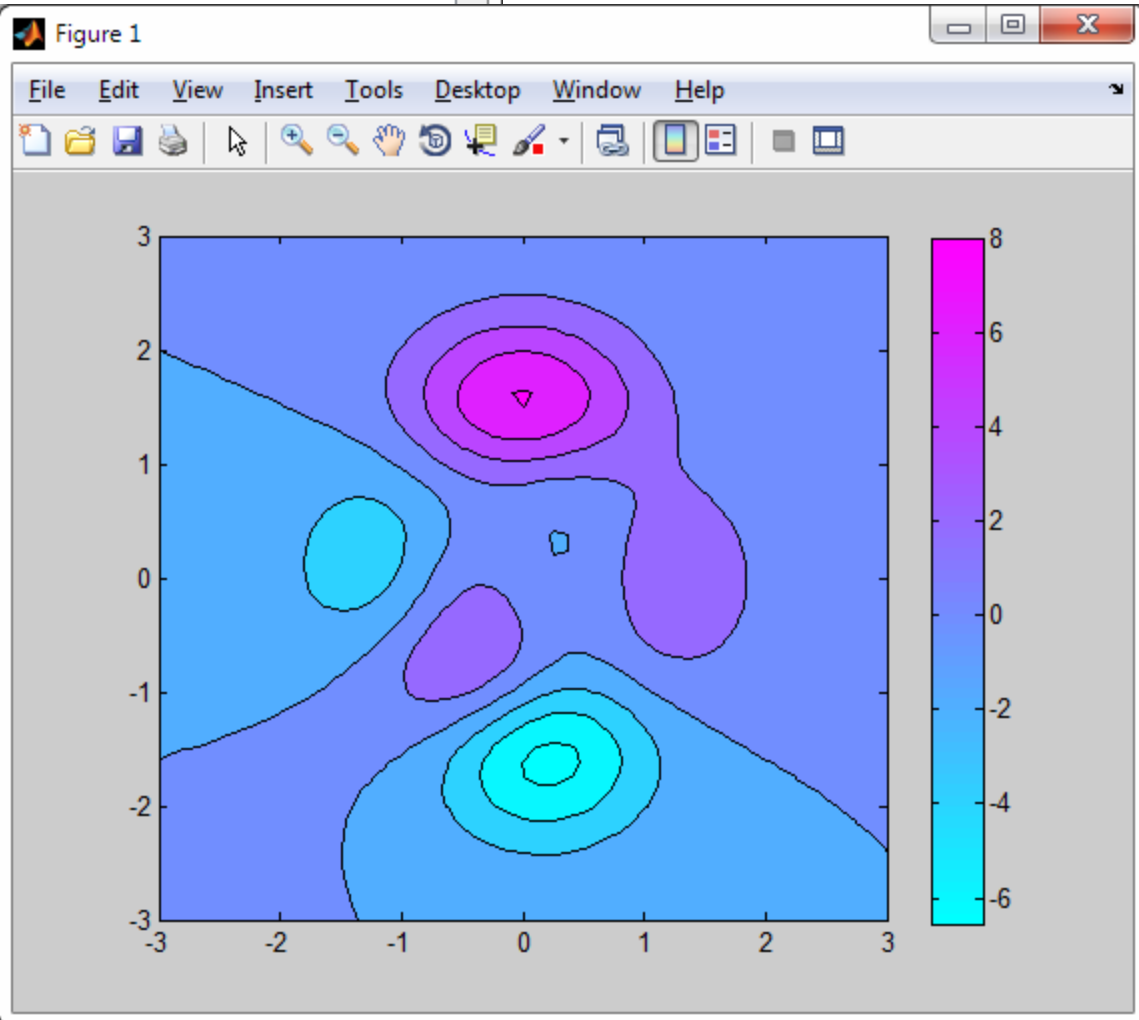
contour_test_2.m



contourf_test.m

```
1 -   clc; clear all; close all;
2
3 -   [X,Y,Z] = peaks;
4
5 -   [C, h] = contourf(X,Y,Z);
6
7 -   colormap cool
8 -   colorbar
```

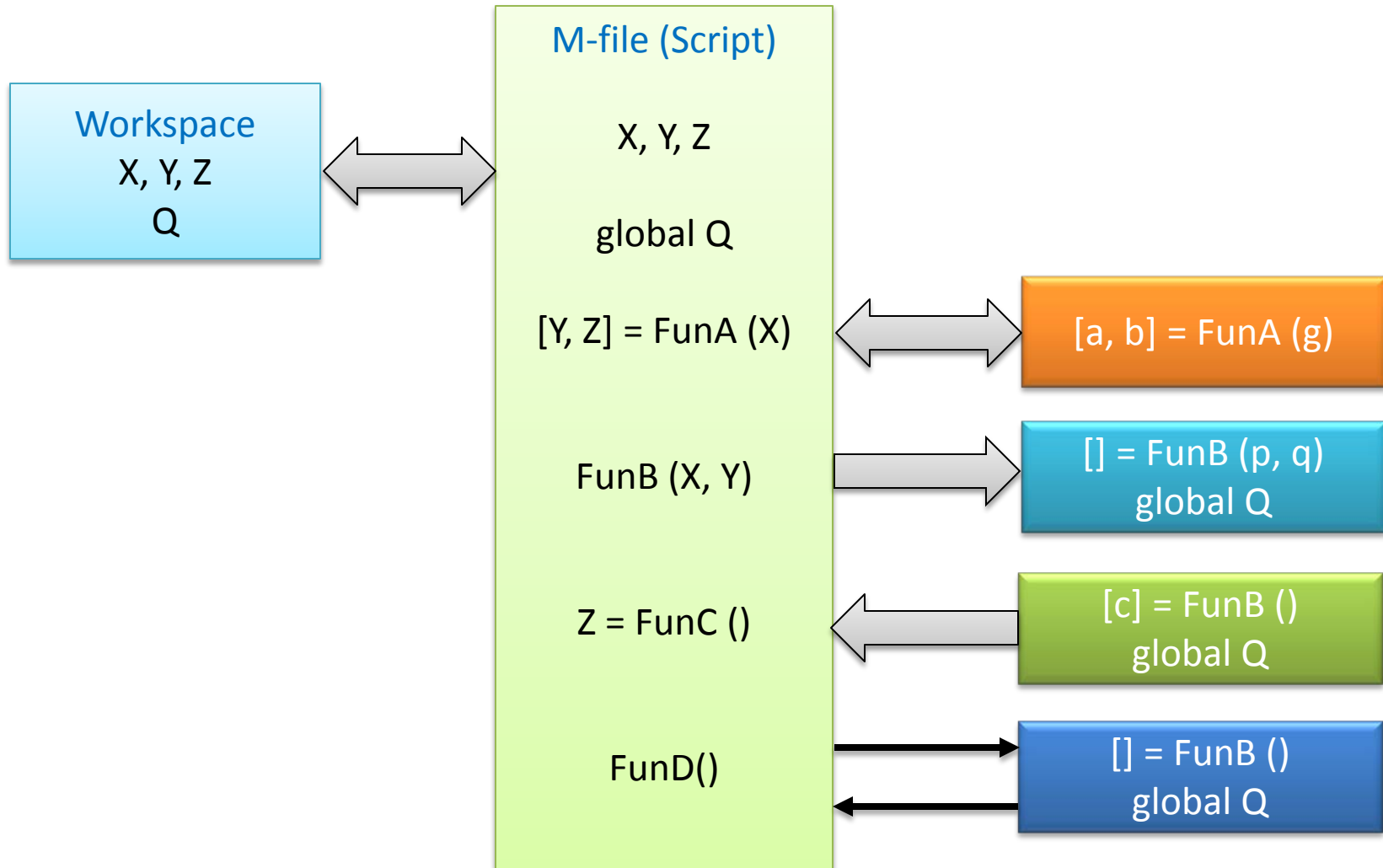
script



Functions

- [Outputs] = FunctionName (Inputs)
- An M-file with an specific header.
- Each M-file function has an area of memory, called the **function workspace**, separate from the MATLAB base **workspace**, in which it operates.

Functions



Editor - C:\Users\sghorbanifaa\Documents\MATLAB\L2\butterfly_fun_call_1...

```
File Edit Text Go Cell Tools Debug Desktop Window Help
[Icons]
- 1.0 + ÷ 1.1 × [Icons]
1 %% Butterfly
2 %% Initializations
3 clc; clear all; close all;
4
5 % Constants
6 a = 1; b = 4; c = 30;
7 A = 2; B = 5;
8
9 N = 90; % Number of rotations
10
11 q = 0:0.01:2*pi*N;
12
13 %% Function call
14 [x,y] = butterfly_fun(q, a, b, c, A, B);
15
16 %% Plot the butterfly image
17 plot(x,y,'b');
18 axis equal
```

butterfly_fun.m × butterfly_fun_call_1.m* ×

script Ln 1

butterfly_fun_call_1.m

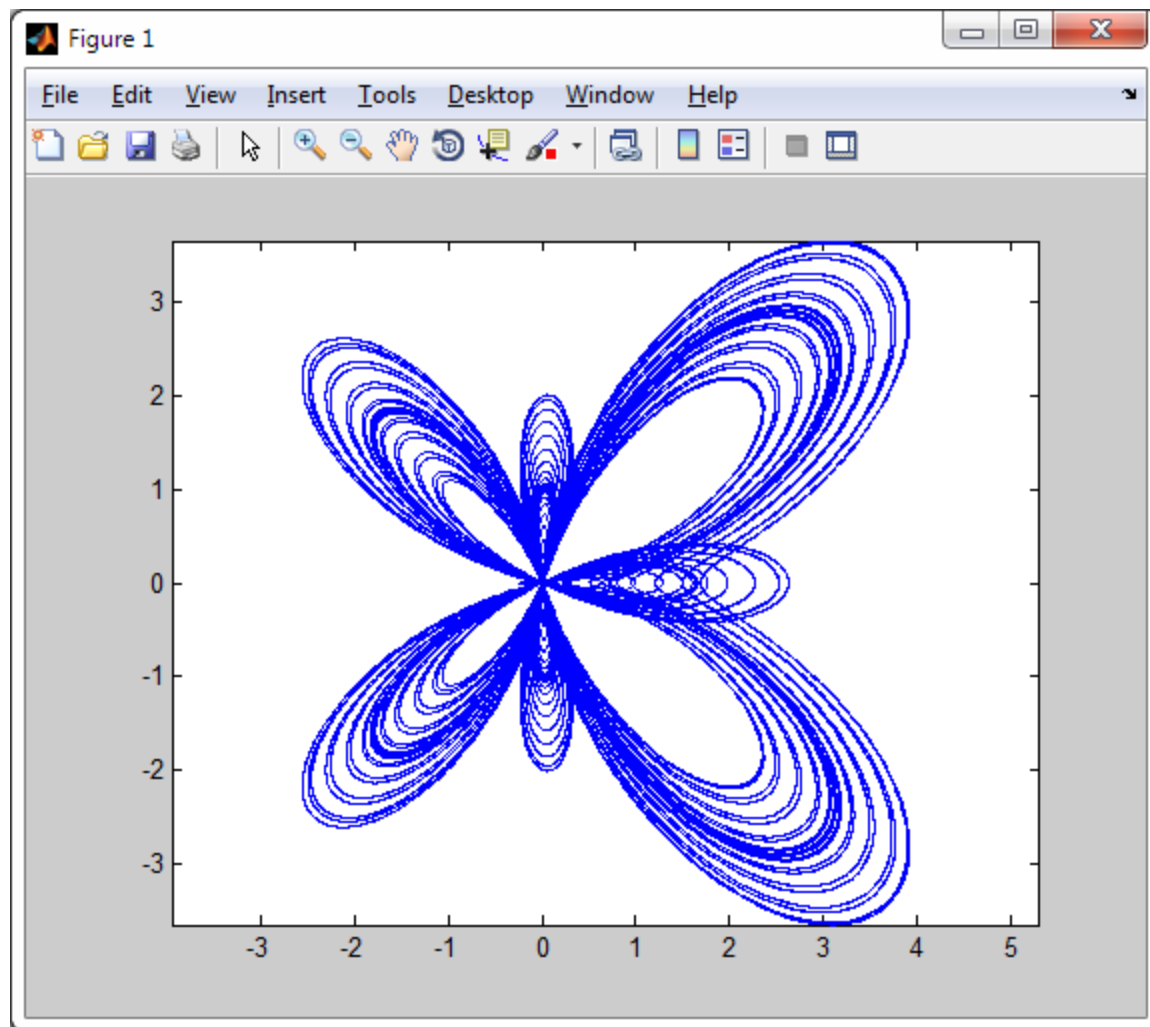
butterfly_fun.m

Editor - C:\Users\sghorbanifaa\Documents\MATLAB\L2\butterfly_fun.m

```
File Edit Text Go Cell Tools Debug Desktop Window Help
[Icons]
- 1.0 + ÷ 1.1 × [Icons]
1 % This function computes r values to draw a butterfly
2 %
3 % Input(s):
4 %         q: Angle
5 %         a, b, c, A, B: constants
6 % Output(s):
7 %         r: radius
8 %
9 % Author: Siamak Faal
10 % Date: Jan, 20, 2013
11 % Worcester Polytechnic Institute
12
13 function [x,y] = butterfly_fun(q, a, b, c, A, B)
14
15     r = exp(cos(a*q)) - A*cos(b*q) + sin(q/c).^B;
16
17     x = r.*cos(q);
18     y = r.*sin(q);
```

butterfly_fun.m × butterfly_fun_call.m ×

butterfly_fun Ln 18 Col 10 OVR



Editor - C:\Users\sghorbanifaal\Documents\MATLAB\L2\bu...

File Edit Text Go Cell Tools Debug Desktop

1 - `clc; clear all; close all;`
2 -
3 - `global a b c A B`
4 - `a = 1; b = 4; c = 30;`
5 - `A = 2; B = 5;`
6 -
7 - `n = 10; % Number of figures`
8 - `N = 90; % Number of rotations`
9 - `q = 0:0.01:2*pi*N;`
10 -
11 - `color = 'rcmbkg';`
12 -
13 - `figure(); hold on;`
14 -
15 - `for i=1:n`
16 - `x0 = 40*rand() - 20;`
17 - `y0 = 40*rand() - 20;`
18 - `q0 = 2*pi*rand();`
19 - `k = round((length(color)-1)*rand()) + 1;`
20 - `[x,y] = butterfly_fun2(q, x0, y0, q0);`
21 - `plot(x,y,color(k));`
22 - `end`
23 - `axis equal`

script Ln 2 Col 1 OVR

butterfly_fun_call_2.m

butterfly_fun2.m

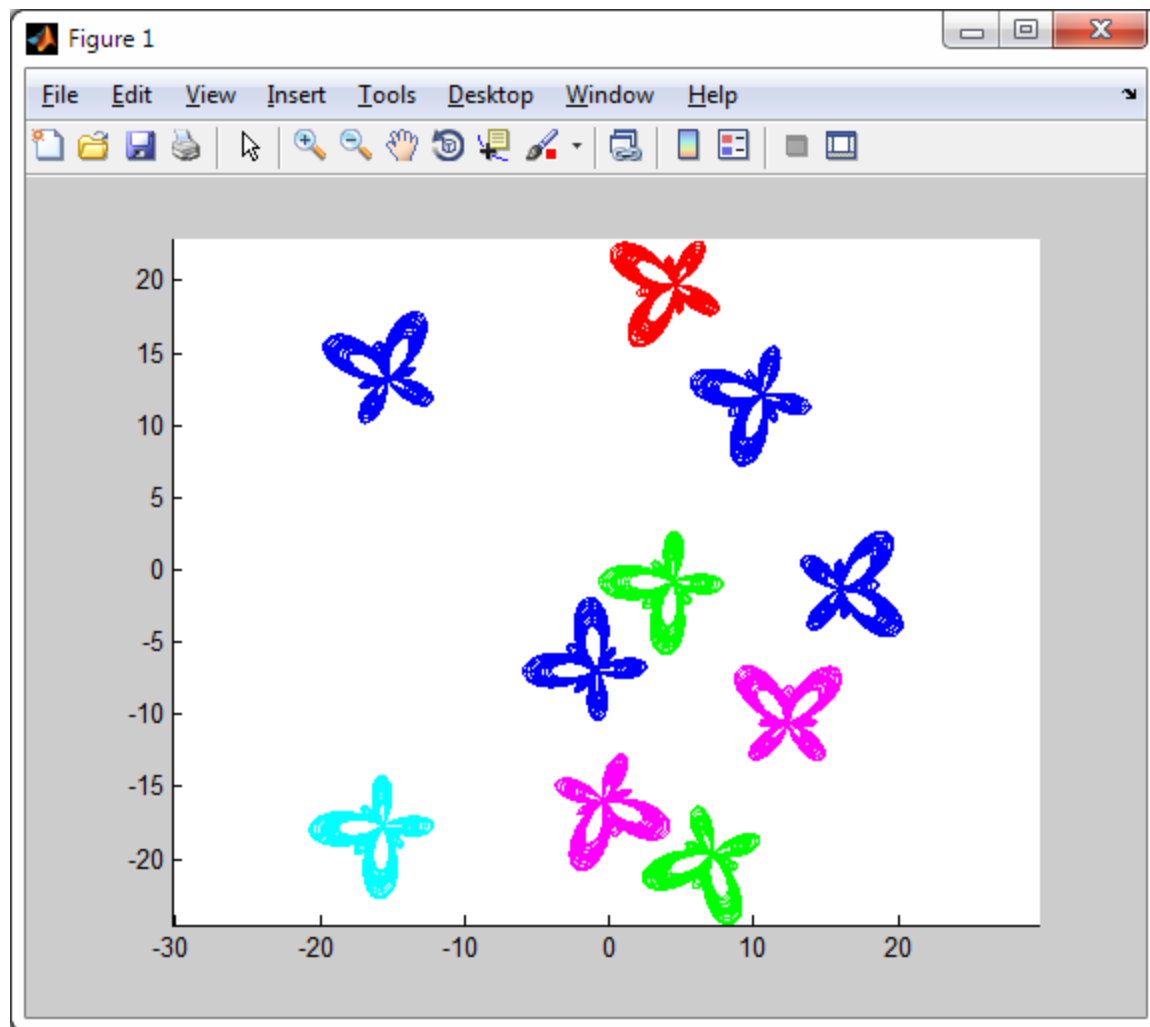
Editor - C:\Users\sghorbanifaal\Documents

File Edit Text Go Cell Tools Debug Desktop

1 - `% This function computes r values to draw a b`
2 - `%`
3 - `% Input(s):`
4 - `% q: Angle`
5 - `% a, b, c, A, B: constants`
6 - `% Output(s):`
7 - `% r: radius`
8 - `%`
9 - `% Author: Siamak Faal`
10 - `% Date: Jan, 20, 2013`
11 - `% Worcester Polytechnic Institute`
12 -
13 - `function [x,y] = butterfly_fun2(q, x0, y0, q0)`
14 -
15 - `global a b c A B`
16 -
17 - `r = exp(cos(a*q)) - A*cos(b*q) + sin(q/c).^B;`
18 -
19 - `x = x0 + r.*cos(q + q0);`
20 - `y = y0 + r.*sin(q + q0);`

butterfly_fun2.m butterfly_fun_call.m

butterfly_fun2 Ln 18 Col 1 OVR



Inline functions

- `inline` construct inline object

- Syntax

```
>> inline(expr)
```

```
>> inline(expr, arg1, arg2, ...)
```

```
>> inline(expr, n)
```

- Example

```
>> f = inline('a^2 + b^2', 'a', 'b')
```

```
>> c = f(3, 4)
```


Anonymous functions

- Syntax

```
>> FunctionName = @(inputs) Expression
```

- Example:

```
>> F = @(a,b) a^2 + b^2
```

```
>> F(3,4)
```

- Example:

```
>> alpha = 0.5;
```

```
>> myFun = @(x) sin(alpha*x)
```

It is possible to use other variables (as constants) in Anonymous functions

Save/Load workspace variables

- **Syntax:**

```
>> save(filename)
```

```
>> save(filename, variables)
```

```
>> S = load(filename)
```

```
>> S = load(filename, variables)
```

Root finding

- Polynomial representation in MATLAB
- $3x^5 + 2x^4 - x^3 + 7x^2 - 5 = 0$
- $3x^5 + 2x^4 - x^3 + 7x^2 + 0x - 5 = 0$
- $p = [3, 2, -1, 7, 0, -5]$

```
>> p = [3, 2, -1, 7, 0, -5];
```

```
>> r = roots(p);
```

Function zeros

- `x = fzero(fun,x0)` tries to find a zero of fun near x_0 , if x_0 is a scalar.

```
>> x = fzero(fun,x0)
```

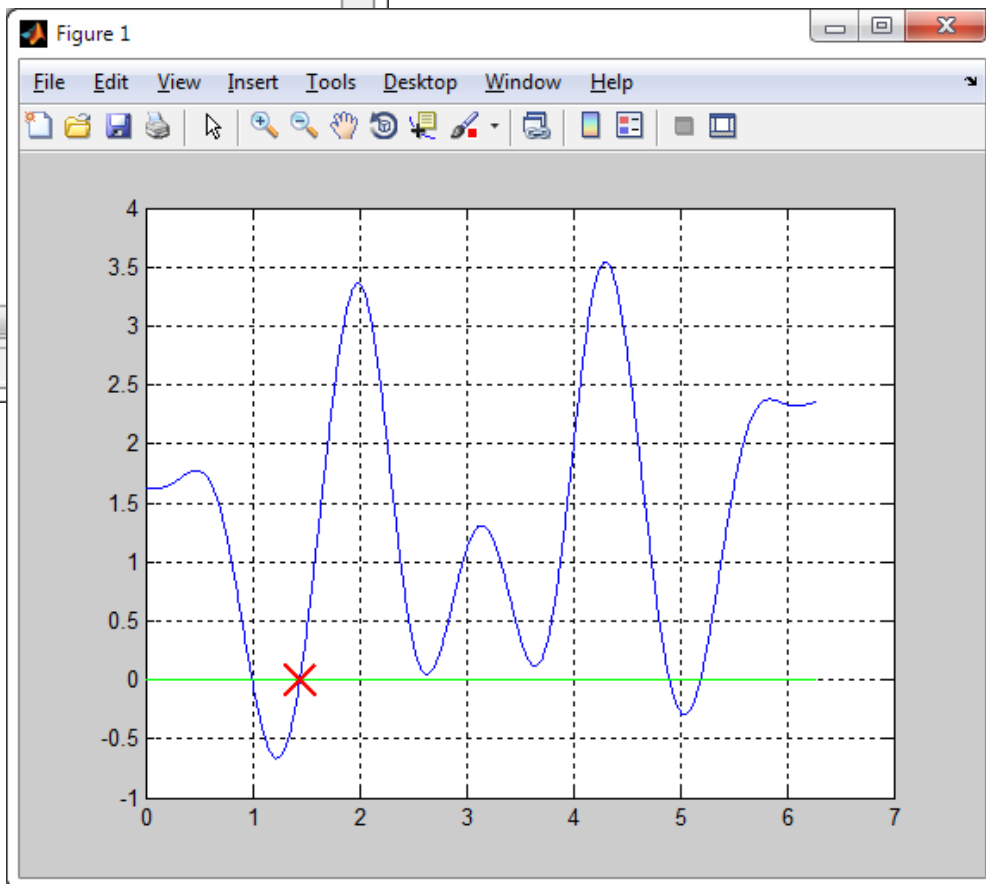
```
>> x = fzero(fun,x0,options)
```

```
>> [x,fval] = fzero(...)
```

```
>> [x,fval,exitflag] = fzero(...)
```

```
>> [x,fval,exitflag,output] = fzero(...)
```

fzero_test.m



Function solve

- `fsolve` finds a root (zero) of a system of nonlinear equations.

```
>> x = fsolve(fun,x0)
```

```
>> x = fsolve(fun,x0,options)
```

```
>> x = fsolve(problem)
```

```
>> [x,fval] = fsolve(fun,x0)
```

```
>> [x,fval,exitflag] = fsolve(...)
```

```
>> [x,fval,exitflag,output] = fsolve(...)
```

```
>> [x,fval,exitflag,output,jacobian] = fsolve(...)
```

Editor - C:\Users\sghorbanifaal\Documents\MATLAB\L2\fsolve_test...

```
File Edit Text Go Cell Tools Debug Desktop Window » » X
+ - 1.0 + ÷ 1.1 x % % %
1 %% Solve the problem
2 clc; clear all; close all;
3
4 x0 = [-2; -2];
5
6 options = optimset('Display','iter');
7 [x,fval] = fsolve(@myFun, x0, options);
8
9 %% Check the result
10 x1 = -4:0.01:3;
11 x2 = -4:0.01:3;
12
13 [X1, X2] = meshgrid(x1,x2);
14
15 F1 = 2*X1 - X2 - exp(-X1);
16 F2 = -X1 + 2*X2 - exp(-X2);
17 P = sqrt(F1.^2 + F2.^2);
18
19 v = 0:5:40;
20 [C,h] = contour(X1,X2,P,v);
21 xlabel('x_1'); ylabel('x_2');
22 clabel(C,h)
23 hold on
24 plot(x(1),x(2),'xr','LineWidth',2,'MarkerSize',20)
```

fsolve_test.m x myFun.m x

script Ln 15 Col 27 OVR

Editor - C:\Users\sghorbanifaal\Documents\M...

```
File Edit Text Go Cell Tools Debug » » X
+ - 1.0 + ÷ 1.1 x % % %
1 function [F] = myFun(x)
2
3 F = [ 2*x(1) - x(2) - exp(-x(1));
4      -x(1) + 2*x(2) - exp(-x(2))];
5
6 end
```

fsolve_test.m x myFun.m* x

n Ln 6 Col 4 OVR

fsolve_test.m

myFun.m

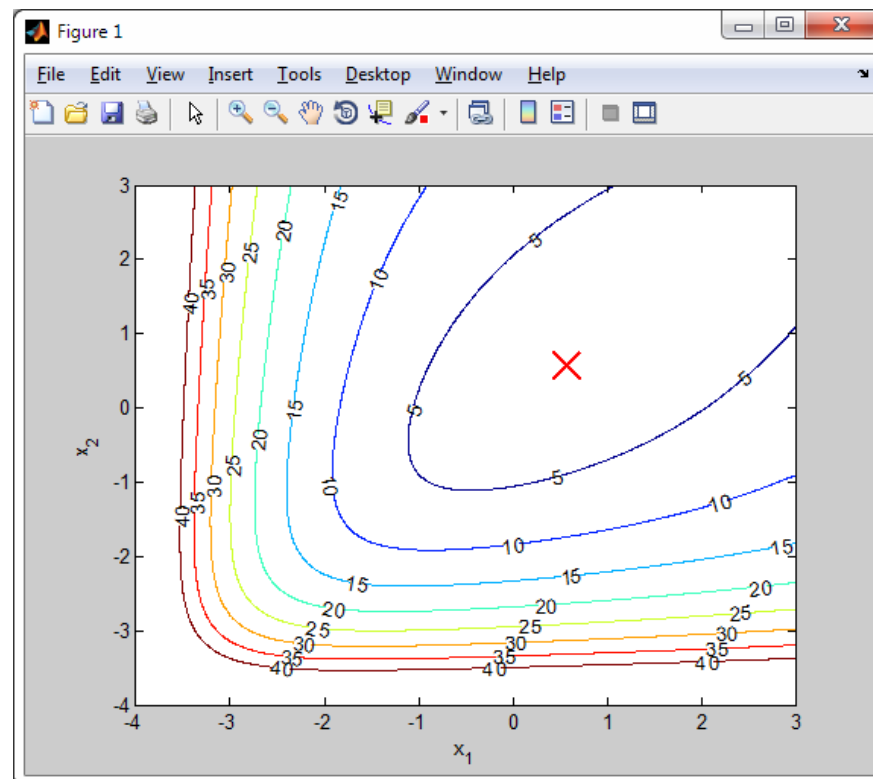
Iteration	Func-count	f(x)	Norm of step	First-order optimality	Trust-region radius
0	3	176.309		78.8	1
1	6	48.7322	1	22.9	1
2	9	11.3497	1	6.66	1
3	12	1.16871	1	1.44	1
4	15	0.00815701	0.573282	0.102	2.5
5	18	4.41277e-07	0.0567828	0.000736	2.5
6	21	1.2978e-15	0.000423838	3.99e-08	2.5

[Equation solved.](#)

fsolve completed because the vector of function values is near zero as measured by the default value of the [function tolerance](#), and the [problem appears regular](#) as measured by the gradient.

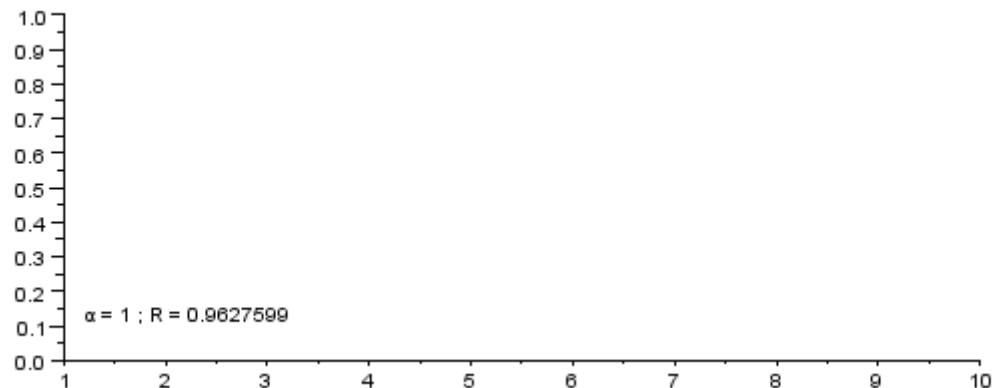
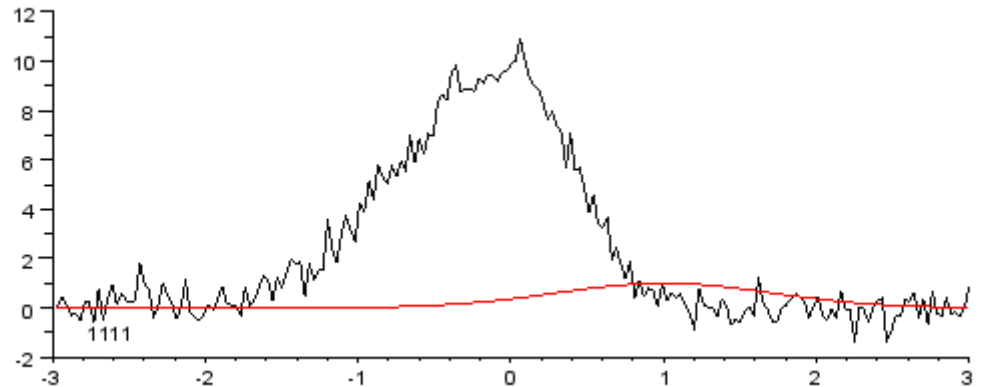
<[stopping criteria details](#)>

f_x >>



Curve fitting

- **Curve fitting** is the process of constructing an specific **curve (mathematical function)**, that has the best fit (minimum error) to a **series of data points**, possibly subject to **constraints**.



Polynomial curve fitting

- `p = polyfit(x,y,n)` finds the coefficients of a polynomial `p(x)` of degree `n` that fits the data, `p(x(i))` to `y(i)`, in a least squares sense. The result `p` is a row vector of length `n+1` containing the polynomial coefficients in `descending` powers.

$$p(x) = p_1x^n + p_2x^{n-1} + \cdots + p_nx + p_{n+1}$$

Editor - C:\Users\sghorbanifaal\Documents\MATLAB\L2\polyfit_test.m

File Edit Text Go Cell Tools Debug Desktop Window Help

+ - 1.0 + ÷ 1.1 × % % %

```
1 %%  
2 clc; clear all; close all;  
3  
4 x = 0:0.01:10;  
5 y = exp(-0.4*x).*sin(1.5*x);  
6  
7 plot(x,y,'b');  
8 hold on;  
9 %%  
10  
11 p = polyfit(x,y,6);  
12  
13 disp(p);  
14 %%  
15  
16 poly_y = polyval(p,x);  
17  
18 plot(x,poly_y,'m');  
19 %%  
20 Error = poly_y - y;  
21 plot(x>Error,:k');  
22 Error_rms = rms(Error);  
23 disp(Error_rms)  
24  
25 legend('Real data','Fitted polynomial','Error');  
26
```

script

Ln 1

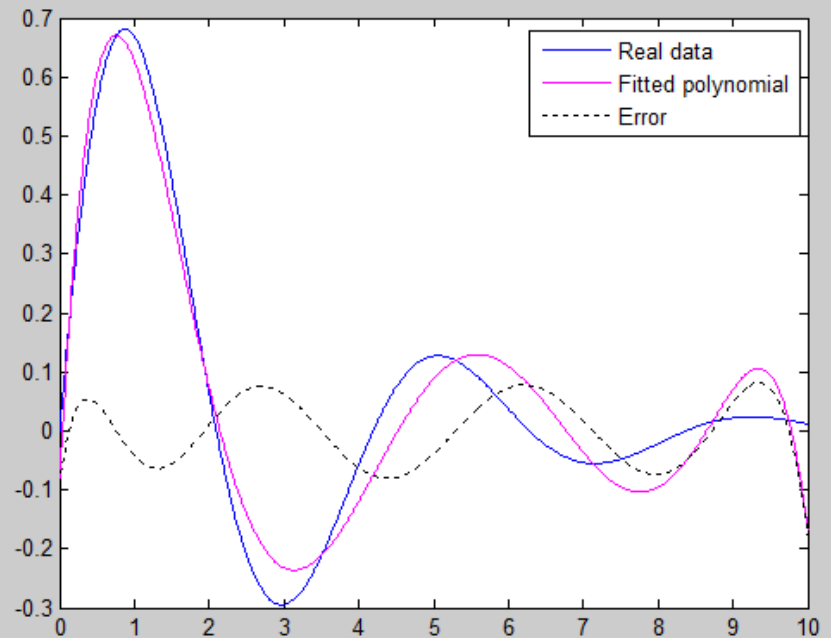
Col 3

OVR

Figure 1

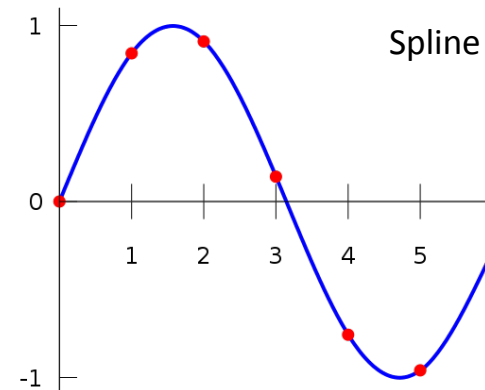
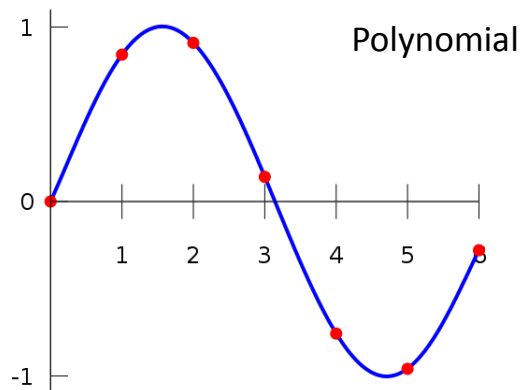
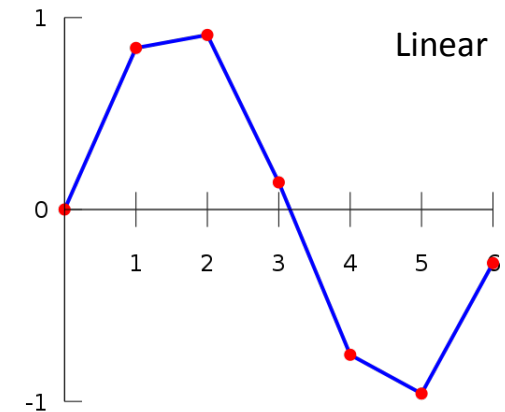
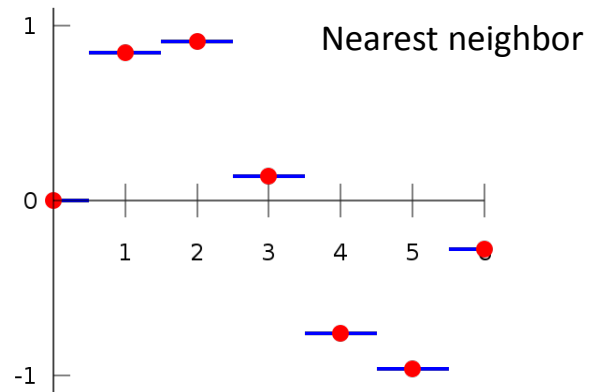
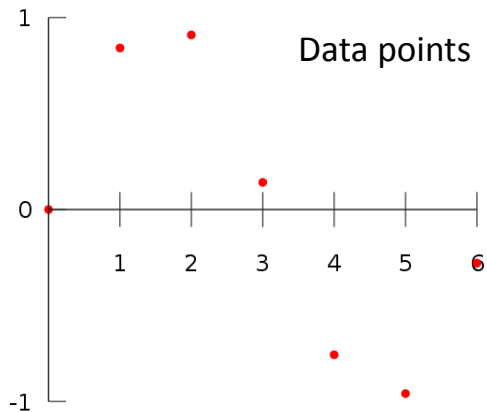
File Edit View Insert Tools Desktop Window Help

+ - 1.0 + ÷ 1.1 × % % %



Interpolation

- **Interpolation** is a method of **constructing new data points** within the range of a discrete set of **known data points**.



Interp1

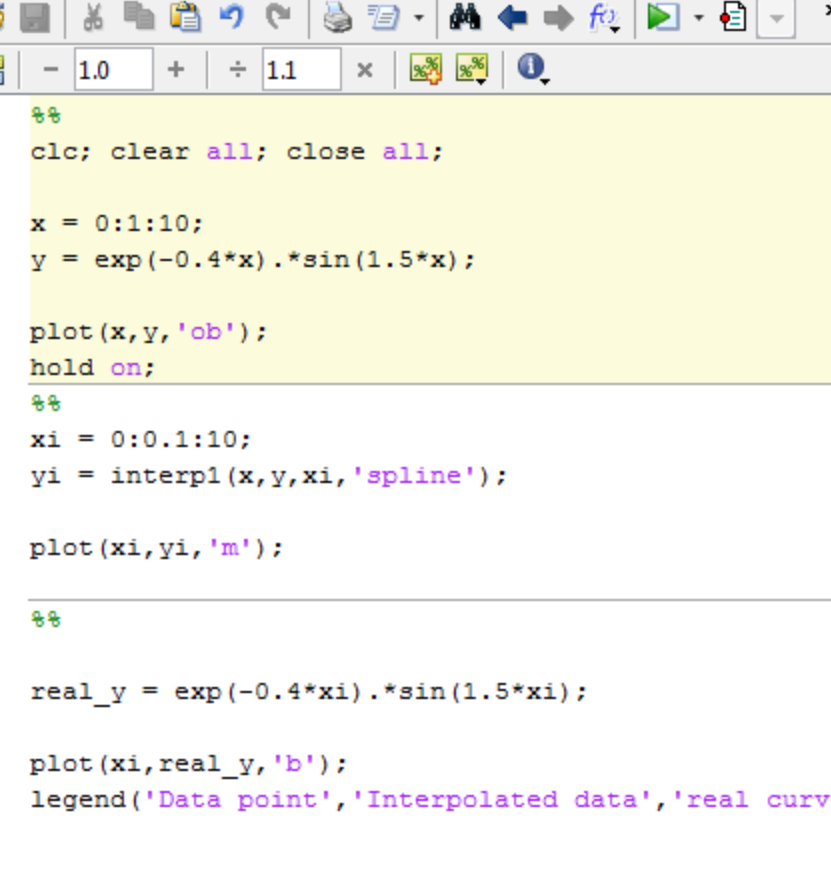
- 1D Interpolation
- Syntax

```
>> yi = interp1(x,Y,xi)
```

```
>> yi = interp1(Y,xi)
```

```
>> yi = interp1(x,Y,xi,method)
```

interp_test.m

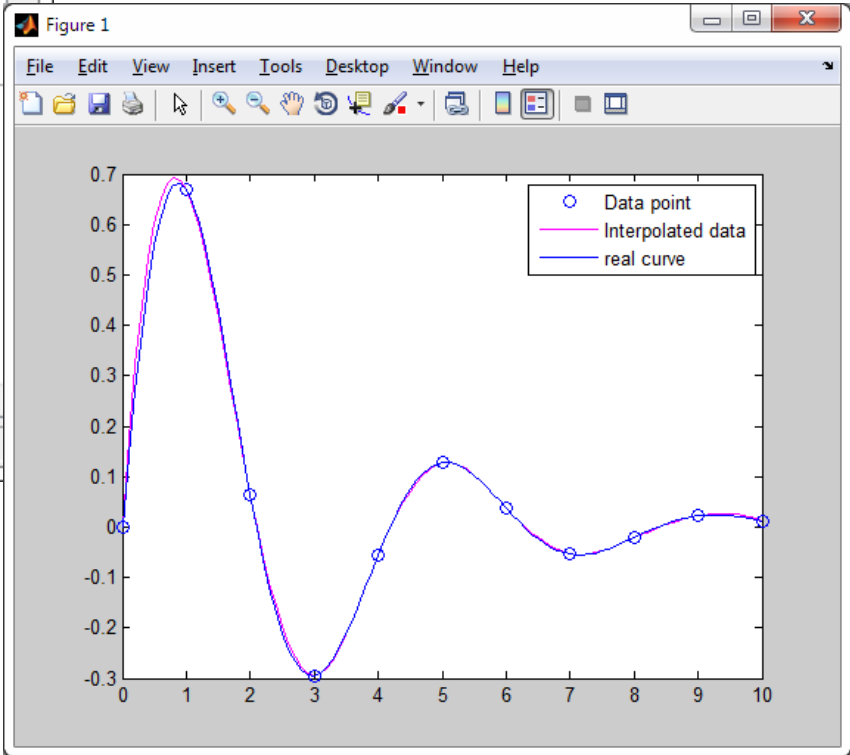


The screenshot shows the MATLAB Editor window with the following script:

```
1 %%  
2 clc; clear all; close all;  
3  
4 x = 0:1:10;  
5 y = exp(-0.4*x).*sin(1.5*x);  
6  
7 plot(x,y,'ob');  
8 hold on;  
9 %%  
10 xi = 0:0.1:10;  
11 yi = interp1(x,y,xi,'spline');  
12  
13 plot(xi,yi,'m');  
14  
15 %%  
16  
17 real_y = exp(-0.4*xi).*sin(1.5*xi);  
18  
19 plot(xi,real_y,'b');  
20 legend('Data point','Interpolated data','real curve');
```

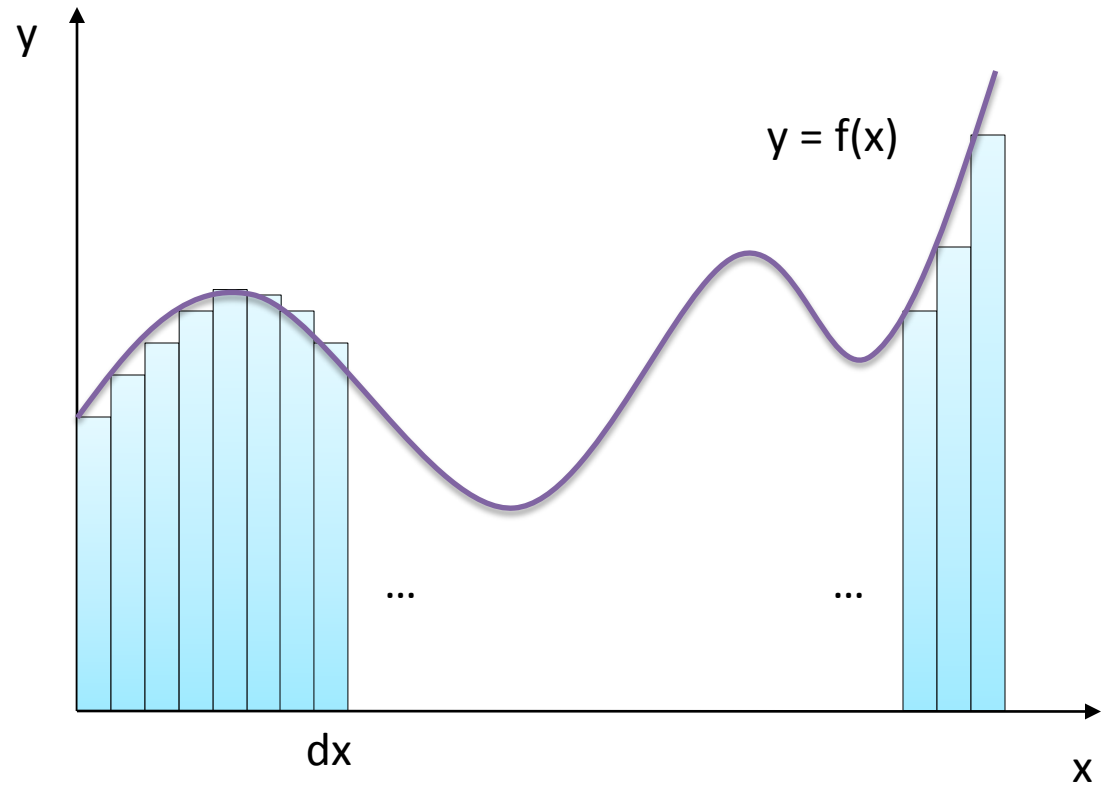
The script performs the following steps:

- Clears the command window, workspace, and all open figures.
- Defines a vector x from 0 to 10 with an interval of 1.
- Calculates the corresponding y values using the function $y = \exp(-0.4x) \cdot \sin(1.5x)$.
- Plots the data points as blue circles ('ob').
- Enables hold mode to overlay additional plots.
- Defines a finer vector xi from 0 to 10 with an interval of 0.1.
- Interpolates the data points to create a smooth curve using the 'spline' method.
- Plots the interpolated data as a magenta line ('m').
- Calculates the real function values for the same xi vector.
- Plots the real function as a blue line ('b').
- Creates a legend with three entries: 'Data point' (blue circles), 'Interpolated data' (magenta line), and 'real curve' (blue line).



Numerical integration

- trapz
- quad



trapz

- Trapezoidal numerical integration
- Syntax

```
>> Z = trapz(Y)
```

```
>> Z = trapz(X,Y)
```

- Example

```
>> x=0:pi/10:pi;
```

```
>> f=sin(x);
```

```
>> q=trapz(x,f)
```

$$\int_0^{\pi} \sin(x) dx = -\cos(\pi) - (-\cos(0)) = 2$$

quad

- Numerically evaluate integral, adaptive Simpson quadrature
- **Quadrature** is a numerical method used to find the area under the graph of a function, that is, to compute a definite integral.
- **`q = quad(fun,a,b)`** tries to approximate the integral of function `fun` from `a` to `b` to **within an error of `1e-6`** using recursive adaptive Simpson quadrature.

- **Example:**

```
>> f=inline('sin(x)', 'x')  
>> [q, fcount] = quad(f, 0, pi)
```