BIL 108

Introduction to the Scientific and Engineering Computing with MATLAB

Lecture 11

F. Aylin Konuklar   Lale T. Ergene

Three Dimensional Plots

Learning Objectives

- **Understand the anatomy of a 3D plot**
- **Basics of constructing plots in 3D**

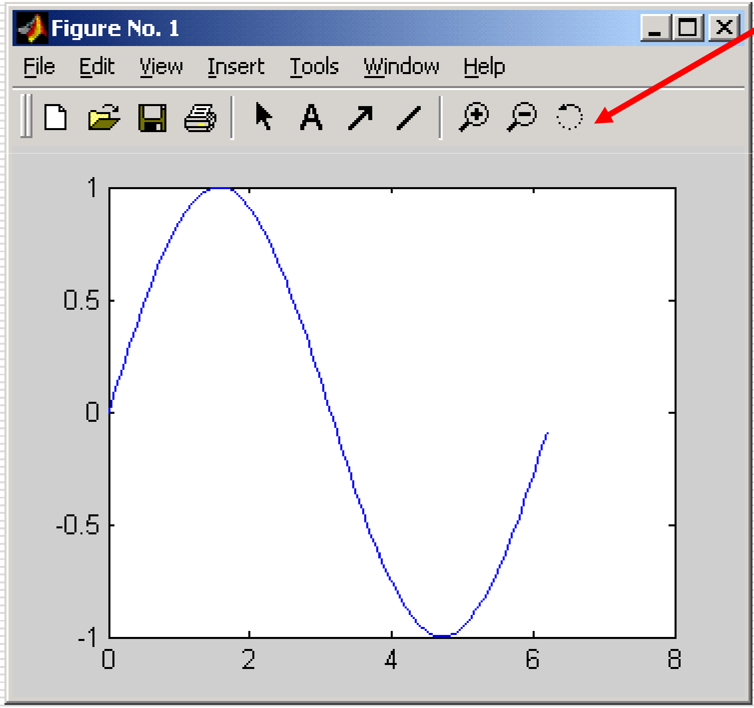
Topics

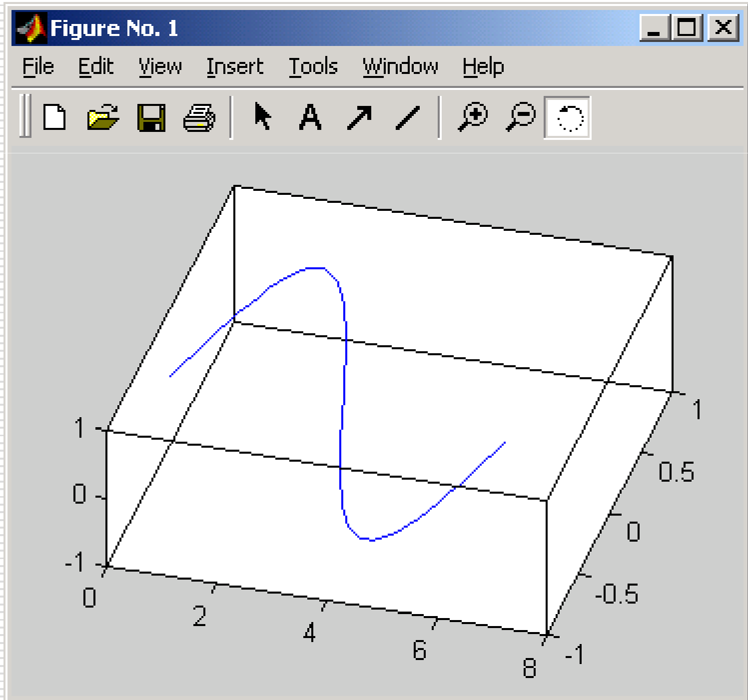
- **Plotting in 3D: major differences**
- **The `meshgrid()` function to create plaids**
- **3D mesh plots**
- **3D surface plots**
- **Contour plots**
- **Use of color to add a 4th**

2D Plots vs 3D Line Plots

- Actually, every 2D plot is simply a 3D plot without the 3rd dimension being specified.

Rotate 3D button: experiment with how it works…





3D Surface Plots

- **It is often desirable to plot functions of the form: z=f(x,y)** ***for each (x,y), we can compute a value for z this defines a surface in 3D space***

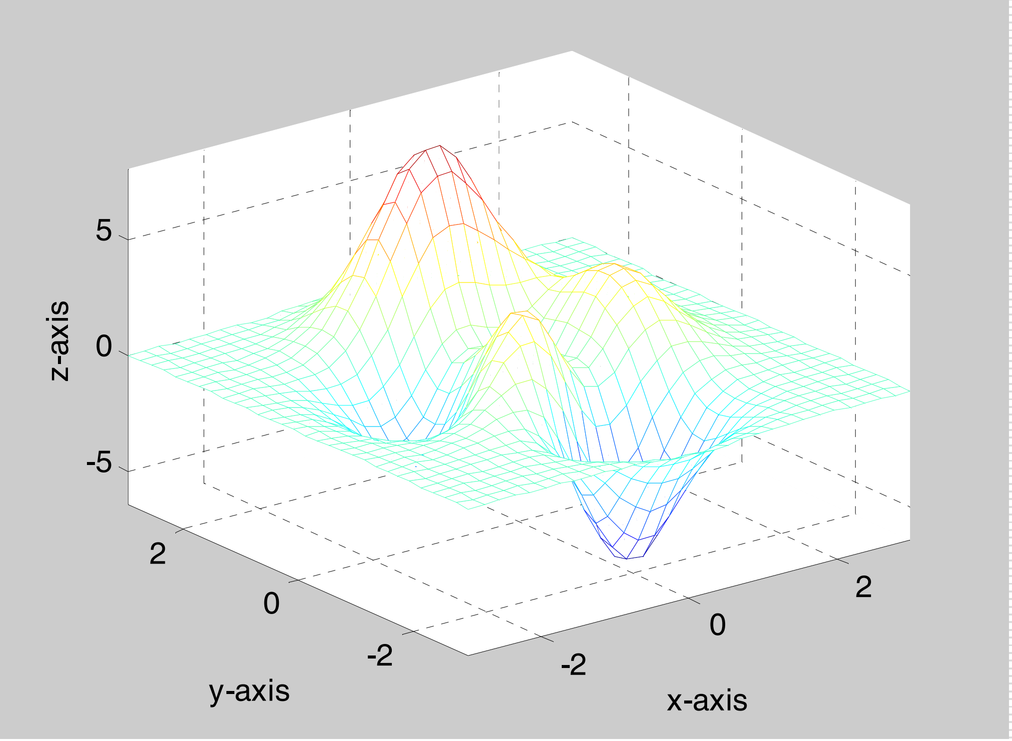
```
x=-1:0.1:3;
y=1:0.1:4;
[X,Y]=meshgrid(x,y);
Z=X.*Y.^2./(X.^2+Y.^2);
surf(X,Y,Z)
mesh(X,Y,Z)
```

We'll use `peaks( )` to create a z=f(x,y) function that is interesting and shows off the 3D plotting

Note: you should check `help peaks` and `help mesh` and also the textbook for further details on these functions

```
>> [x,y,z]=peaks(30);
>> mesh(x,y,z)
>> axis tight
>> xlabel('x-axis')
>> ylabel('y-axis')
>> zlabel('z-axis')
```

Suggestion:
Try using **`hidden off`** and **`hidden on`** to see what happens.

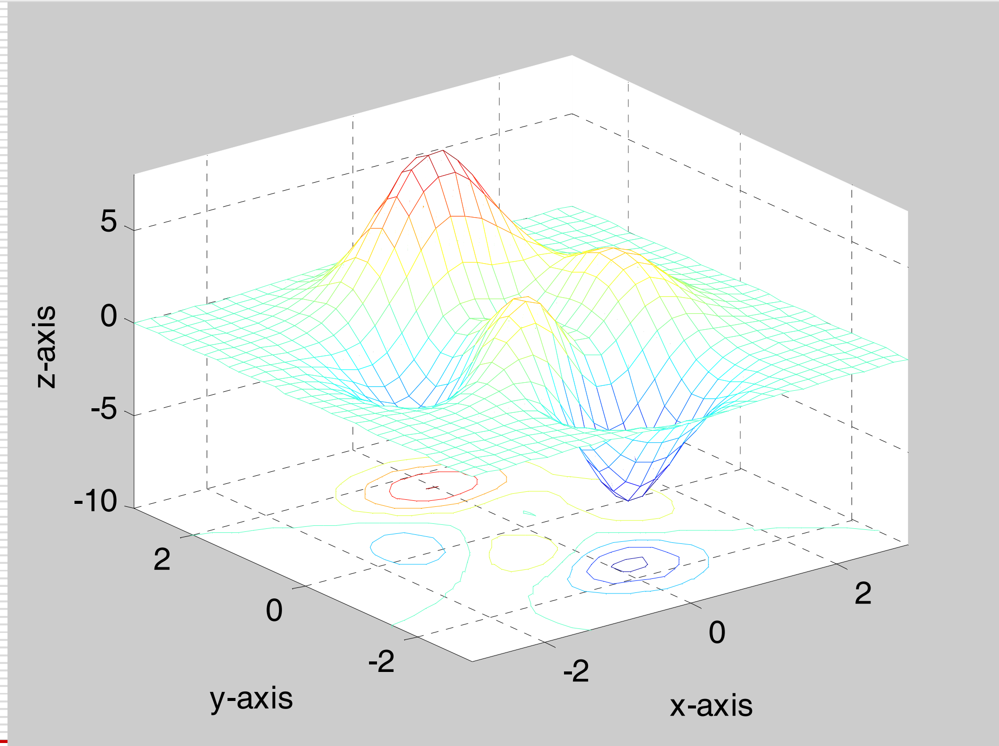


Exploring meshc Plots

meshc( ) adds a contour plot directly below the mesh

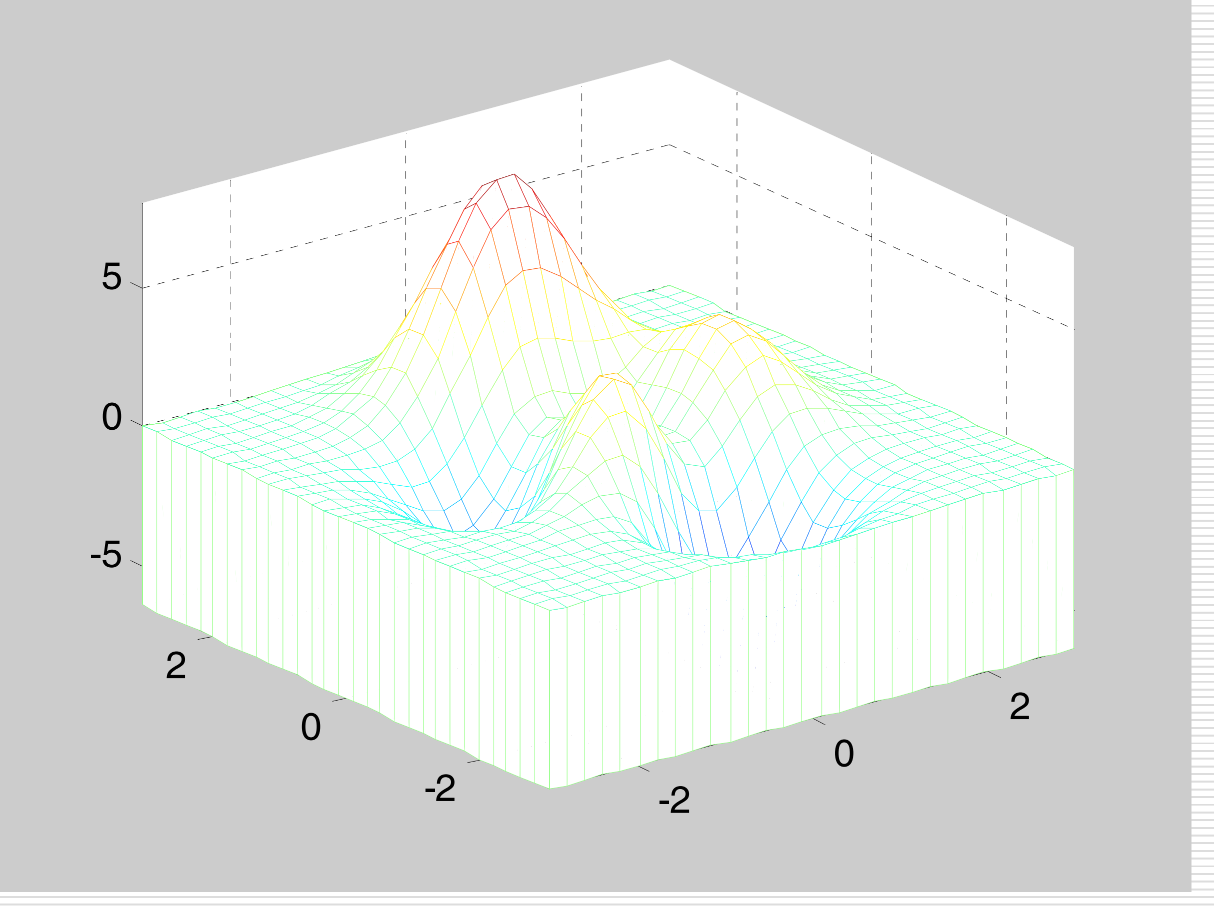
- ***helps visualize the contours***
- ***can locate the peaks and dips***

```
>> [x,y,z]=peaks(30);
>> meshc(x,y,z)
>> axis tight
>> xlabel('x-axis')
>> ylabel('y-axis')
>> zlabel('z-axis')
```



Exploring **meshz** Plots

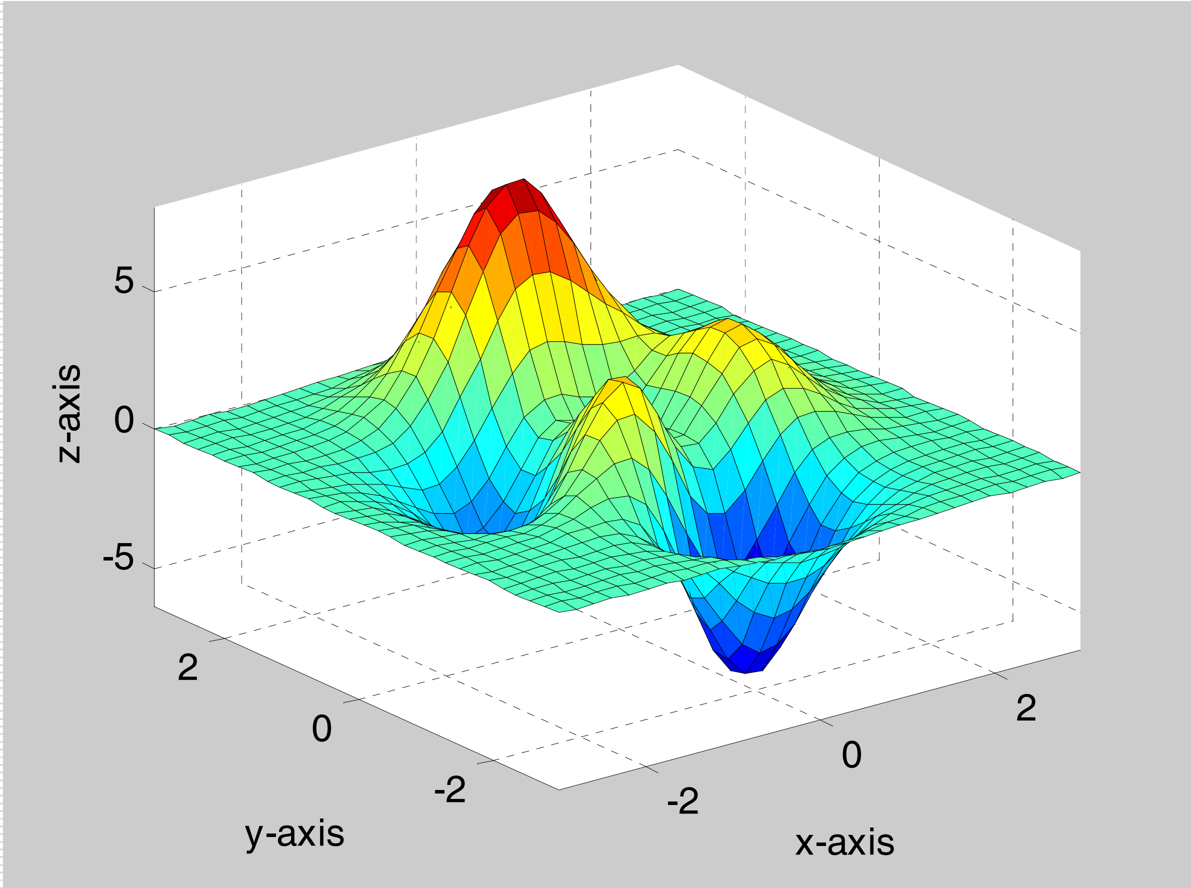
```
>> [x,y,z]=peaks(30);
>> meshz(x,y,z)
>> axis tight
>> xlabel('x-axis')
>> ylabel('y-axis')
>> zlabel('z-axis')
```



Explore the `surf( )` Function

- The basic function uses the default `shading faceted` and this shows the mesh:

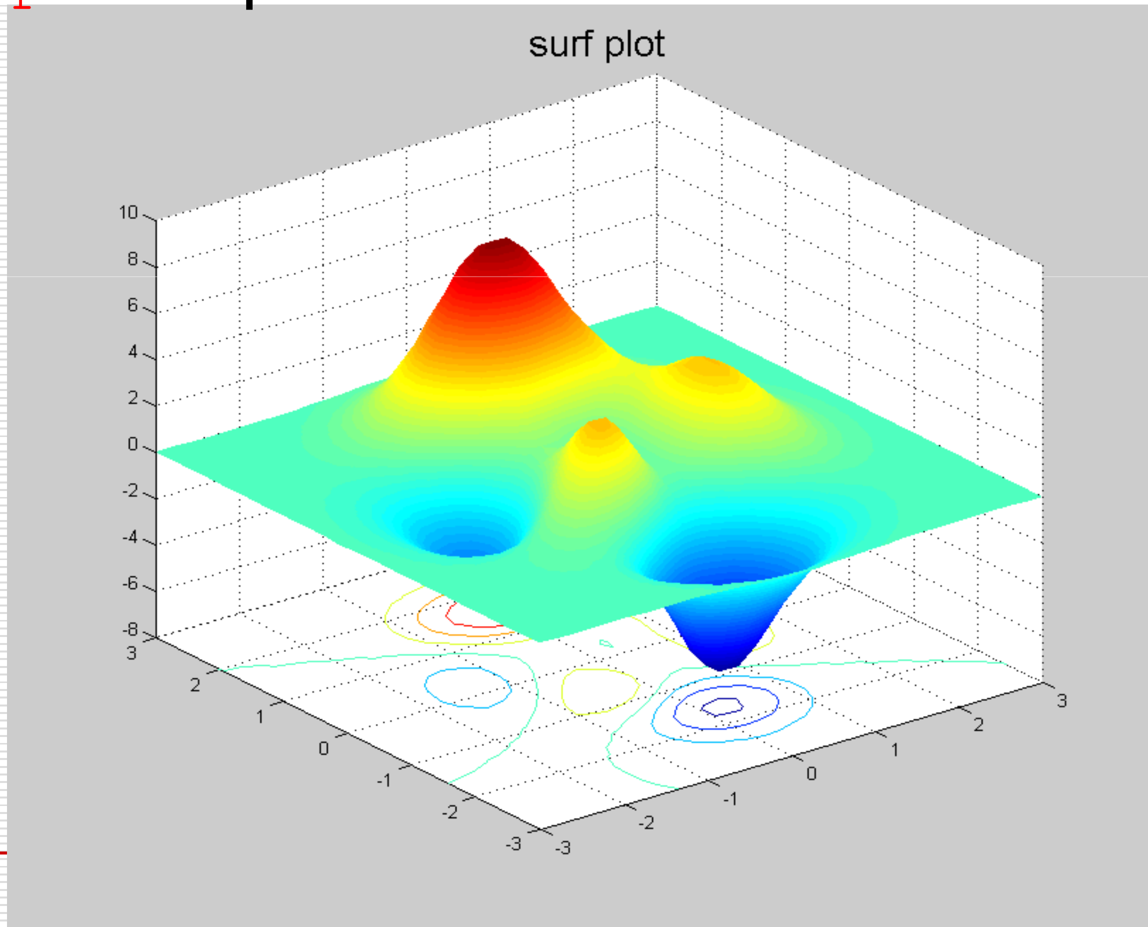
```
>> [x,y,z]=peaks(30);
>> surf(x,y,z)
>> axis tight
>> xlabel('x-axis')
>> ylabel('y-axis')
>> zlabel('z-axis')
```



- surfc acts much like meshc with a contour plot drawn below the surface

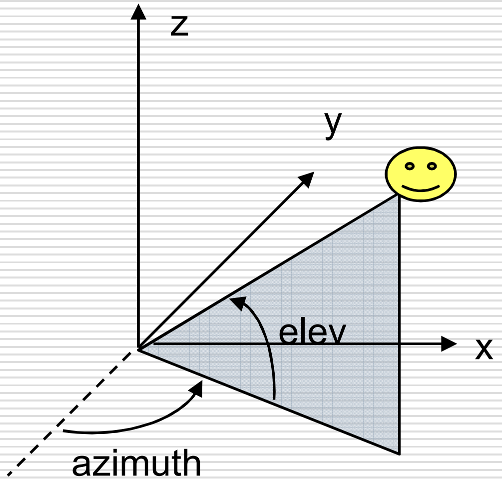
- shading interp interpolates color over each facet

```
>> [x,y,z]=peaks(30);
>> surfc(x,y,z)
>> shading interp
>> axis tight
>> xlabel('x-axis')
>> ylabel('y-axis')
>> zlabel('z-axis')
```

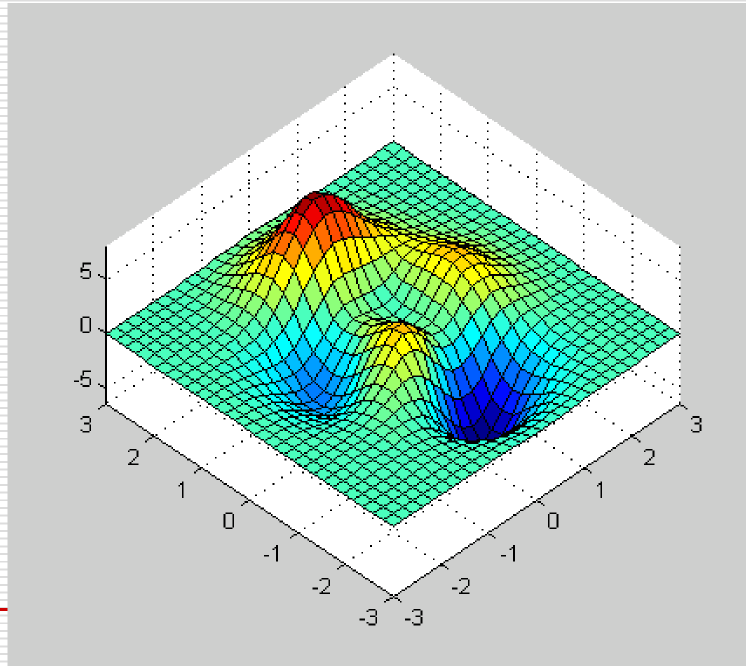


Changing the Viewing Direction

- You can change the orientation of the object
  - Viewing direction: *view(az,el)* or you can use the rotate3d button on the view toolbar on the figure window menu
  - Camera direction: this is best controlled from the camera toolbar on the figure window menu




```
» view(-45,60)
```



ODE

- Only limited number of ODE has an analytical solution
- Numerical Methods MATLAB built-in functions
- FIRST ORDER ODE

$$y' = \frac{dy}{dx} = f(x, y)$$

X independent variable
Y dependent variable (function of x)

- An initial or boundary condition is needed in order to specify a unique solution.
- **STEPS solving a single first order ODE**..

$$\frac{dy}{dt} = f(t, y)$$

- Let us take the independent variable as t(time)

$$t_o \le t \le t_f$$

- STEP 1. Write the problem in a standard form

With $y=y_0$ and $t=t_o$

- These three info is required to solve an ODE
- EX:

$$\frac{dy}{dt} = \frac{t^3 - 2y}{t}$$

$$1 \le t \le 3$$

y=4.2 at t=1

initial value of y

STEP 2

- Create a function file
  - Create a user defined fnc. that calculates dy/dt for a given values of t and y.

  - function dydt=ODEexp1(t,y)
  - dydt=(t^3-2*y)/t;

STEP3. Select a method of solution

- Both Euler's and Runge Kutta Methods approximate a fnc. Using its Taylor series expansion.
- TAYLOR SERIES is an expansion that can be used to approximate a fnc. whose derivatives exist on an interval containing a and b. The Taylor series expansion for f(b) is

$$f(b) = f(a) + (b-a)f'(a) + \frac{(b-a)^2}{2!}f''(a) + \ldots\ldots + \frac{(b-a)^n}{n!} + \ldots.$$

- A first order Taylor Series Approx.

$$f(b) = f(a) + (b-a)f'(a)$$

- 1st order Runge-Kutta (Euler's Method)

$$y_b = y_a + hy'_a$$

- This equation estimates the function value y_b using a straight line tangent to the function at y_a
- Once we have determined the value of y_b, we can estimate the next value of the function f(c) using the following

$$y_c = y_b + hy'_b$$

- Initial Value Solutions : an initial value is needed to solve them.

MATLAB’s ODE Solvers

MATLAB has a wide variety of ODE solvers which can vary in how they solve and what ODE’s they are certified to solve. Some are as follows:

- Ode45, ode23, ode113, ode15s, ode23s, ode23t, ode23tb

MATLAB ODE SOLVERS

Solver	Problem Type	Order of Accuracy	When to Use
`ode45`	Nonstiff	Medium	Most of the time. This should be the first solver you try.
`ode23`	Nonstiff	Low	If using crude error tolerances or solving moderately stiff problems.
`ode113`	Nonstiff	Low to high	If using stringent error tolerances or solving a computationally intensive ODE file.
`ode15s`	Stiff	Low to medium	If `ode45` is slow because the problem is stiff.
`ode23s`	Stiff	Low	If using crude error tolerances to solve stiff systems and the mass matrix is constant.
`ode23t`	Moderately Stiff	Low	If the problem is only moderately stiff and you need a solution without numerical damping.
`ode23tb`	Stiff	Low	If using crude error tolerances to solve stiff systems.

Stiff problem: includes fast and slowly changing components and require small time steps in their solution.

Step4. Solve the ODE

- [t,y]=solver_name('ODEFUNC',tspan,y0)
- Go back to the example:

$$\frac{dy}{dt} = \frac{t^3 - 2y}{t}$$

$$1 \le t \le 3$$

y=4.2 at t=1

initial value of y

- [t y]=ode45('ODEexp1',[1:0.5:3],4.2)
- plot(t,y)

How to use MATLAB TO Solve ODE

You need to write two MATLAB files:

1. The main program which contains the known numerical constants and calls the solver.
2. A function file which contains the ODE in the form MATLAB requires.

Solving a sample 1st Order ODE

Here's an example. Let's say we have the ODE and initial condition:

$$\dot{y} + 2ty = 0$$

$$y(0) = 3$$

I want to plot this from 0 to 5 seconds.

The Function File

The contents of the function file are as follows. When I save the file it needs to be called yprime.m

```
function dy = yprime(t,y)
dy = -2*t*y;
```

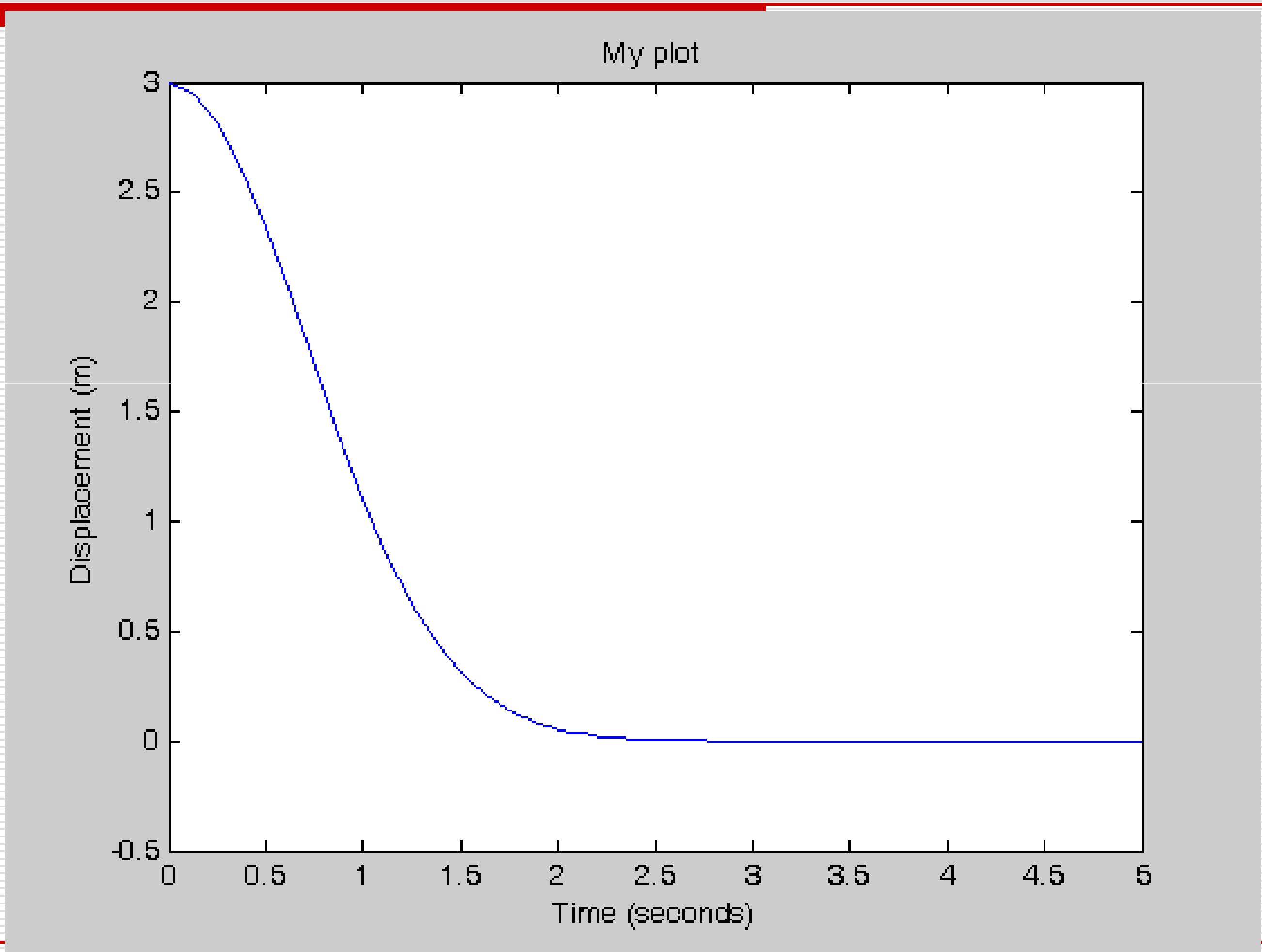
Choose your ODE solver

You should use the help to pick which ODE solver you would like to use. Read this carefully so you know what the solver needs in order to do it’s job.   In this example we will use ODE45.

The Main program

```
clear all
close all
clc
tspan = [0 5];
icond = 3;
[T,Y] = ode45('yprime',tspan,icond);
plot(T,Y),xlabel('Time (seconds)'),ylabel('Displacement
    (m)'),...
  Title('My plot')
```

The results


My plot
Displacement (m)
3
2.5
2
1.5
1
0.5
0
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Time (seconds)

Examples using Symbolic Matlab

- int : integration
- int(S)
- int(S,a,b)
  - eg1. syms x n

    int(x^n,x)

  - eg2. a=0; b=pi/2;

    int (cos(2*x),0,pi/2)

Cont…

- eg. syms x k
- a=sym('k!');

  symsum((x^k)/a, k,0,inf)

Differentiation (diff)

- diff (s)
- diff(s,’v’) differentiate wrt v
- diff(s,’v’, n) differentiate n times

- eg. syms a x
- f=(a^2)*(x^3)
- diff(f)
- diff(f,a)
- diff(f,2)

2nd order diff –nth order diff

- Eq.

```
>>syms x a b c d
>>fx=a+b*x+c*x^2+d*x^3
>>dfx2=diff(fx,2)
>>dfx3=diff(fx,3)
```

dsolve

$$\frac{dy}{dx} = x + y$$

In order to solve such an equation in Matlab,

>>dsolve('Dy=y+x',x)