INTRODUCTION TO SCIENTIFIC & ENGINEERING COMPUTING
BIL 108E, CRN23320

Assoc. Prof. Dr. Hilmi Berk Çelikoğlu
Dr. S.Gökhan Karaman

Technical University of Istanbul

April 28, 2010

Tentative Course Schedule, CRN 23320

Week	Date	Topics
1	Feb. 10	Introduction to Scientific and Engineering Computing
2	Feb. 17	Introduction to Program Computing Environment
3	Feb. 24	Variables, Operations and Simple Plot
4	Mar. 03	Algorithms and Logic Operators
5	Mar. 10	Flow Control, Errors and Source of Errors
6	Mar. 17	Functions
6	Mar. 20	Exam 1
7	Mar. 24	Arrays
8	Mar. 31	Solving of Simple Equations
9	Apr. 07	Polynomials Examples
10	Apr. 14	Applications of Curve Fitting
11	Apr. 18	Exam 2
11	Apr. 21	Applications of Interpolation
12	Apr. 28	Applications of Numerical Integration
13	May 05	Symbolic Mathematics
14	May 12	Ordinary Differential Equation (ODE) Solutions with Built-in Functions

LECTURE # 12

ORDINARY DIFFERENTIAL EQUATIONS(ODE)

- FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS
- EULER'S METHOD
- RUNGE-KUTTA METHODS
- HIGHER ORDER ORDINARY DIFFERENTIAL EQUATIONS

ORDINARY DIFFERENTIAL EQUATIONS

- A first-order ordinary differential equation is an equation that can be written in the following form,
- $y' = \frac{dy}{dx} = g(x, y)$
- where x is the independent variable and y is a function of x.
- Examples:
  - $y' = g_1(x, y) = 3\,x^2$
  - $y' = g_2(x, y) = -0.131\,y$
  - $y' = g_3(x, y) = 3.4444\,10^{-5} - 0.0015\,y$
  - $y' = g_4(x, y) = 2\,x\,cos^2(y)$
  - $y' = g_5(x, y) = 3\,y + e^{2\,x}$

ORDINARY DIFFERENTIAL EQUATIONS

- A solution to a first-order ODE is a function $y = f(x)$, such that $f'(x) = g(x, y)$.
- Computing the solution to a differential equation involves integration in order to obtain y from y'
- The techniques for solving differential equations are often referred to as techniques for integrating differential equations.
- The solution to a differential equation is generally a family of functions.

ORDINARY DIFFERENTIAL EQUATIONS

- An initial condition or boundary condition is usually needed in order to specify a unique solution.
- Using certain initial conditions, analytical solutions to the given ODE examples are listed below.
  - $y = x^3 - 7.5$
  - $y = 4\,e^{-0.131\,x}$
  - $y = 0.022963 - 0.020763\,e^{-0.0015\,x}$
  - $y = tan^{-1}(x^2 + 1)$
  - $y = 4\,e^{3\,x} - e^{2\,x}$

ODEs

ODEs

- Although an analytical solution to a differential equation is preferred, many differential equations have complicated analytical solutions or no analytical solution at all.
- For these cases, a numerical calculation is needed to solve the differential equation.
- The most common numerical techniques for solving ordinary differential equations are
  - Euler's Method
  - Runge-Kutta Methods.

ORDINARY DIFFERENTIAL EQUATIONS

- Both Euler's method and Runge-Kutta methods approximate a function using its Taylor series expansion.
- A Taylor series is an expansion that can be used to aproximate a function whose derivatives exist on an interval containing a and b.
- The Taylor's series for expansion for $f(b)$ is

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

$$+\frac{(b-a)^n}{n!}\,f^{(n)}(a) + \ldots$$

ORDINARY DIFFERENTIAL EQUATIONS

- A first-order Taylor's series approximation uses the terms involving the function and its first derivative:

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

- A second-order approximation uses the terms involving the function, its first derivative, and its second derivative:

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

- Matlab functions use approximations of order 2, 3, 4 and 5 to approximate the function value $f(b)$.

FIRST ORDER APPROXIMATION(EULER'S METHOD)

- The simplest numerical method for the solution of initial value problems is Euler's method.
- It uses a fixed step size h and generates the approximate solution.
- A first order Runge-Kutta integration equation is the following:

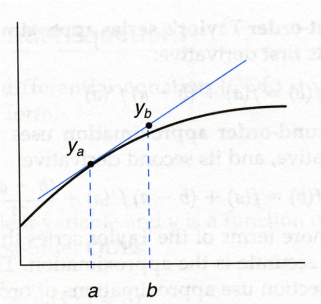
$$y_b = y_a + h\, y'_a$$

- The differential equation is used to compute the value of y'_a

FIRST ORDER APPROXIMATION(EULER'S METHOD)

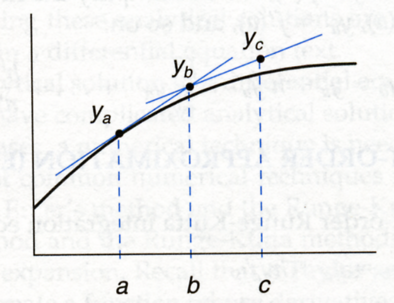
- Once we have determined the value of y_b, we can estimate the next value of the function $f(c)$ using the following:

$$y_c = y_b + h\, y_b'$$



FIRST ORDER APPROXIMATION(EULER'S METHOD)

- This equation uses the tangent line at y_b to estimate y_c.



FIRST ORDER APPROXIMATION(EULER'S METHOD)

- An initial value is needed to start the process.
- This method is also called initial-value solutions or boundary-value solutions.

FIRST ORDER APPROXIMATION(EULER'S METHOD)

- The first-order Runge-Kutta Method is simple to apply.
- If the step size is large or if the slope of the function changes rapidly, it may not be very accurate.
- Higher-order Runge-Kutta integration method is used to approximate the unknown function.
- Fourth-order Runge-Kutta integration equation uses first, second, 3^{rd} and 4^{th} derivatives to compute the function.

ORDINARY DIFFERENTIAL EQUATIONS

MATLAB FUNCTIONS

- `ode23`, `ode45`
- [x, y]=ode23('function_ name', a, b, initial, tolerance)
- [x, y]=ode45('function_ name', a, b, initial, tolerance)

MATLAB FUNCTIONS

- x and y : the set of coordinates that represent the function $y = f(x)$
- function_ name: defines a function that returns the values of the differential equation $y' = g(x, y)$
- a and b : endpoints of the interval.
- initial : the value of the left endpoint of the interval.

ORDINARY DIFFERENTIAL EQUATIONS

EXAMPLE 1:

- $y' = g_1(x, y) = 3\,x^2$
- interval $[2, 4]$
- $y = f(2) = 0.5$

Answer: $y = x^3 - 7.5$

EXAMPLE 1:

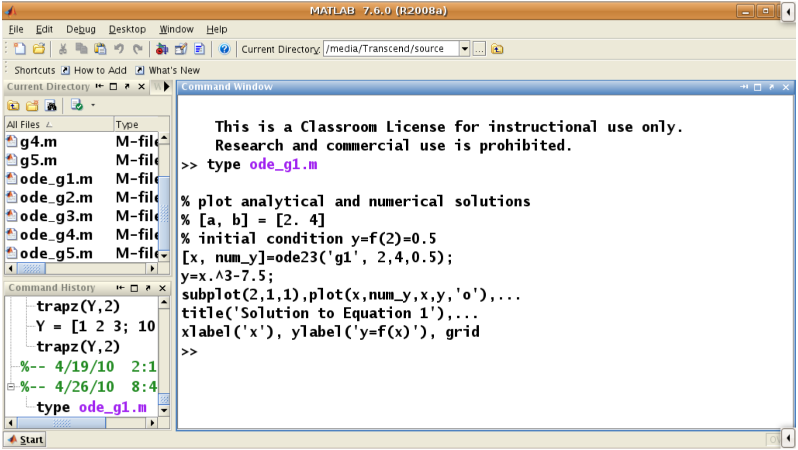
```
function dy=g1(x,y)
% function dy/dx=g1
%
dy=3*x.^2;
```

EXAMPLE 1:

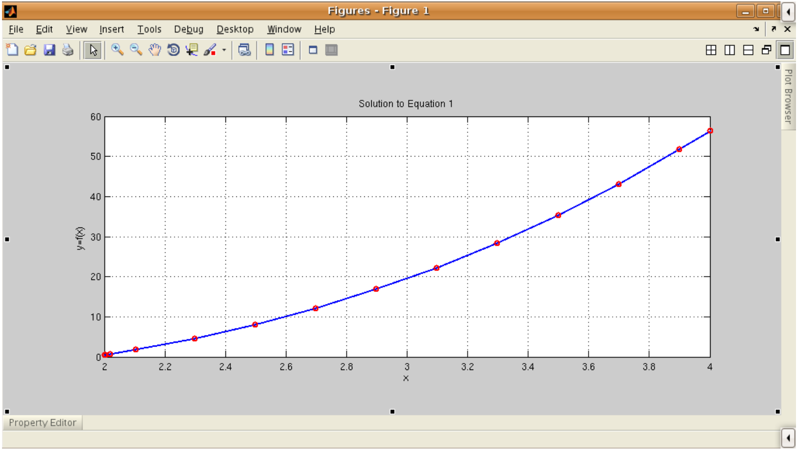
```
% plot analytical and numerical solutions
% [a, b] = [2. 4]
% initial condition y=f(2)=0.5
[x, num_y]=ode23('g1', 2,4,0.5);
y=x.^3-7.5;
subplot(2,1,1),plot(x,num_y,x,y,'o'),...
title('Solution to Equation 1'),...
xlabel('x'), ylabel('y=f(x)'), grid
```

ORDINARY DIFFERENTIAL EQUATIONS

EXAMPLE 1:



EXAMPLE 1:


Figures - Figure 1
File Edit View Insert Tools Debug Desktop Window Help
Solution to Equation 1
y=f(x)
x
0 10 20 30 40 50 60
2 2.2 2.4 2.6 2.8 3 3.2 3.4 3.6 3.8 4
Plot Browser
Property Editor

EXAMPLE 2:

- $y' = g_2(x, y) = -0.131\, y$
- interval $[0, 5]$
- $y = f(0) = 4$

Answer: $y = 4\, e^{-0.131\, x}$

EXAMPLE 2:

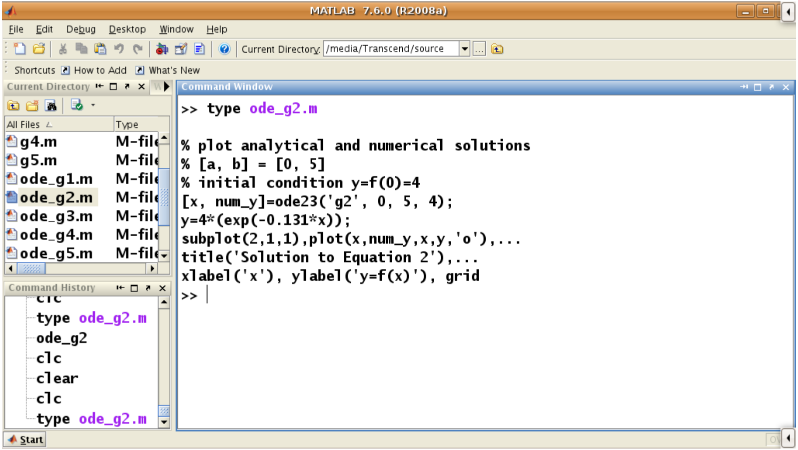
```
function dy=g2(x,y)
% function dy/dx=g2
%
dy=-0.131*y;
```

EXAMPLE 2:

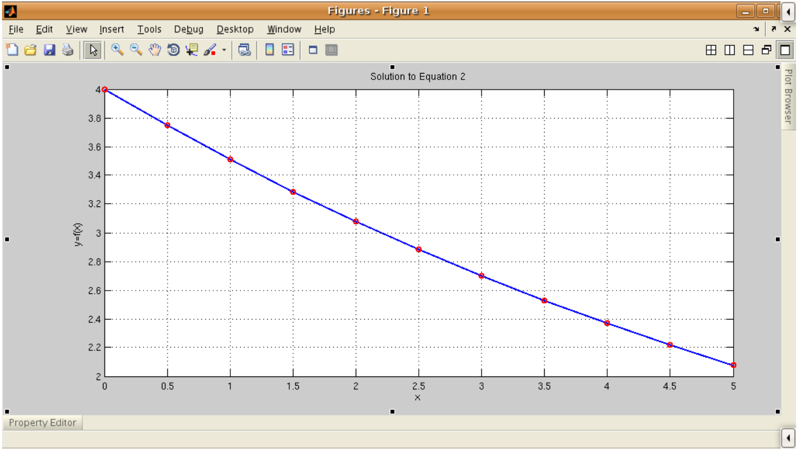
```
% plot analytical and numerical solutions
% [a, b] = [0, 5]
% initial condition y=f(0)=4
[x, num_y]=ode23('g2', 0, 5, 4);
y=4*(exp(-0.131*x));
subplot(2,1,1),plot(x,num_y,x,y,'o'),...
title('Solution to Equation 2'),...
xlabel('x'), ylabel('y=f(x)'), grid
```

ORDINARY DIFFERENTIAL EQUATIONS

EXAMPLE 2:



EXAMPLE 2:


Figures - Figure 1
File Edit View Insert Tools Debug Desktop Window Help
Solution to Equation 2
y=f(x)
x
4
3.8
3.6
3.4
3.2
3
2.8
2.6
2.4
2.2
2
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5
Plot Browser
Property Editor

EXAMPLE 3:

- $y' = g_3(x, y) = 3.4444\,10^{-5} - 0.0015\,y$
- interval $[0, 120]$
- $y = f(0) = 0.0022$

Answer: $y = 0.022963 - 0.020763\,e^{-0.0015\,x}$

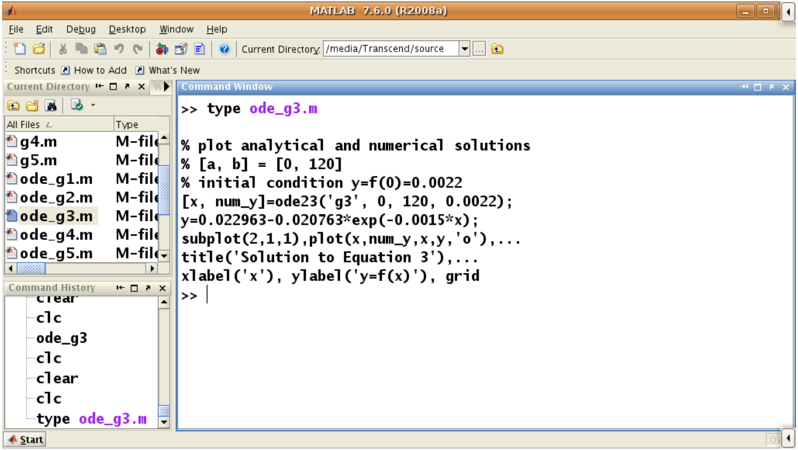
EXAMPLE 3:

```
function dy=g3(x,y)
% function dy/dx=g3
%
dy=3.4444e-05-0.0015*y;
```

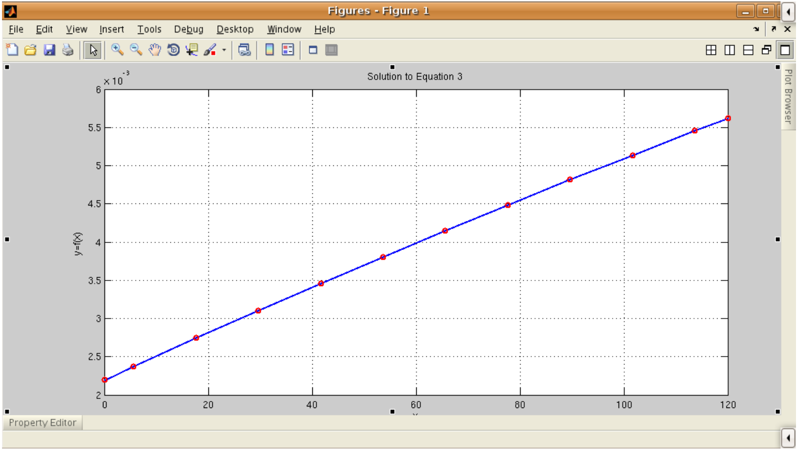
EXAMPLE 3:

```
% plot analytical and numerical solutions
% [a, b] = [0, 120]
% initial condition y=f(0)=0.0022
[x, num_y]=ode23('g3', 0, 120, 0.0022);
y=0.022963-0.020763*exp(-0.0015*x);
subplot(2,1,1),plot(x,num_y,x,y,'o'),...
title('Solution to Equation 3'),...
xlabel('x'), ylabel('y=f(x)'), grid
```

EXAMPLE 3:


MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Shortcuts How to Add What's New
Current Directory
All Files Type
g4.m M-file
g5.m M-file
ode_g1.m M-file
ode_g2.m M-file
ode_g3.m M-file
ode_g4.m M-file
ode_g5.m M-file
Command History
clear
clc
ode_g3
clc
clear
clc
type ode_g3.m
Command Window
>> type ode_g3.m
% plot analytical and numerical solutions
% [a, b] = [0, 120]
% initial condition y=f(0)=0.0022
[x, num_y]=ode23('g3', 0, 120, 0.0022);
y=0.022963-0.020763*exp(-0.0015*x);
subplot(2,1,1),plot(x,num_y,x,y,'o'),...
title('Solution to Equation 3'),...
xlabel('x'), ylabel('y=f(x)'), grid
>>
Start

EXAMPLE 3:


Figures - Figure 1
File Edit View Insert Tools Debug Desktop Window Help
Solution to Equation 3
× 10^-3
y=f(x)
6
5.5
5
4.5
4
3.5
3
2.5
2
0
20
40
60
80
100
120
Plot Browser
Property Editor

EXAMPLE 4:

- $y' = g_4(x, y) = 2\,x\,cos^2(y)$
- interval $[0, 2]$
- $y = f(0) = \frac{\pi}{4}$

Answer: $y = tan^{-1}(x^2 + 1)$

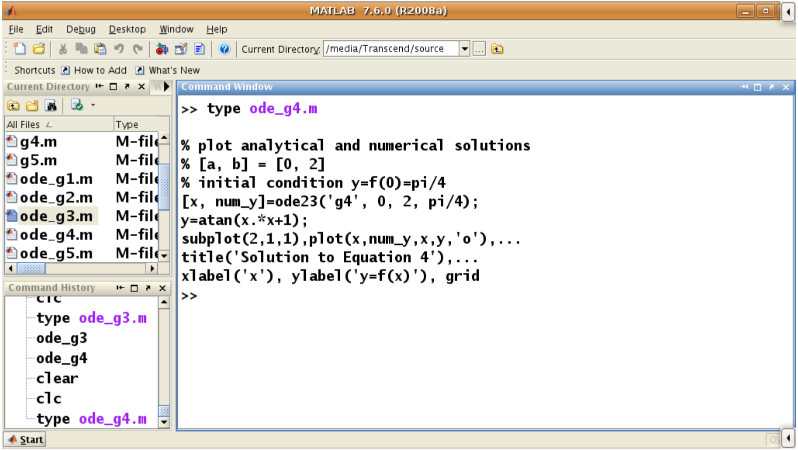
EXAMPLE 4:

```
function dy=g4(x,y)
% function dy/dx=g4
%
dy=2*x.*cos(y).^2;
```

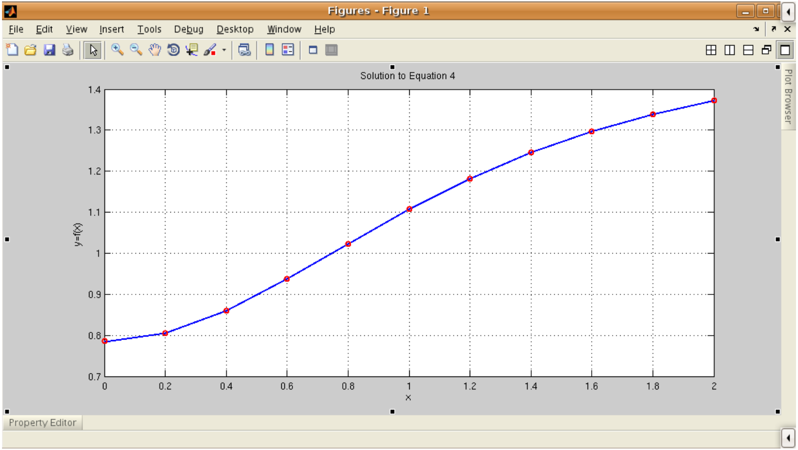
EXAMPLE 4:

```
% plot analytical and numerical solutions
% [a, b] = [0, 2]
% initial condition y=f(0)=pi/4
[x, num_y]=ode23('g4', 0, 2, pi/4);
y=atan(x.*x+1);
subplot(2,1,1),plot(x,num_y,x,y,'o'),...
title('Solution to Equation 4'),...
xlabel('x'), ylabel('y=f(x)'), grid
```

EXAMPLE 4:


MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Shortcuts How to Add What's New
Current Directory
All Files Type
g4.m M-file
g5.m M-file
ode_g1.m M-file
ode_g2.m M-file
ode_g3.m M-file
ode_g4.m M-file
ode_g5.m M-file
Command History
clc
type ode_g3.m
ode_g3
ode_g4
clear
clc
type ode_g4.m
Command Window
>> type ode_g4.m
% plot analytical and numerical solutions
% [a, b] = [0, 2]
% initial condition y=f(0)=pi/4
[x, num_y]=ode23('g4', 0, 2, pi/4);
y=atan(x.*x+1);
subplot(2,1,1),plot(x,num_y,x,y,'o'),...
title('Solution to Equation 4'),...
xlabel('x'), ylabel('y=f(x)'), grid
>>
Start

EXAMPLE 4:


Figures - Figure 1
File Edit View Insert Tools Debug Desktop Window Help
Solution to Equation 4
y=f(x)
x
1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Plot Browser
Property Editor

EXAMPLE 5:

- $y' = g_5(x, y) = 3\,y + e^{2\,x}$
- interval $[0, 3]$
- $y = f(0) = 3$

Answer: $y = 4\,e^{3\,x} - e^{2\,x}$

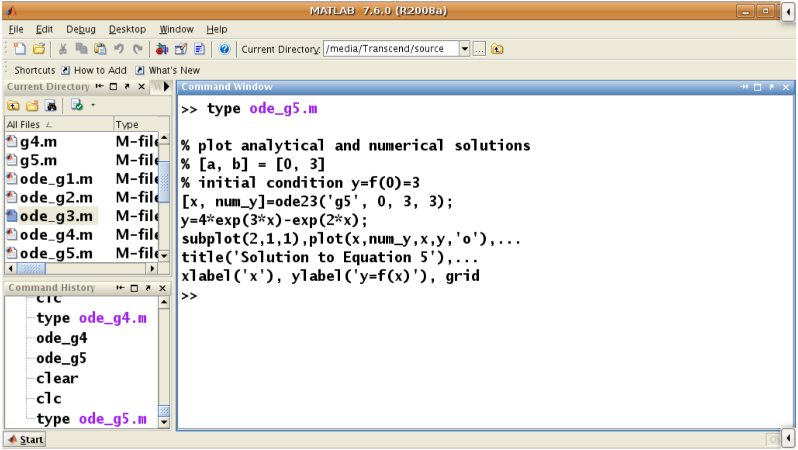
EXAMPLE 5:

```
function dy=g5(x,y)
% function dy/dx=g5
%
dy=3*y+exp(2*x);
```

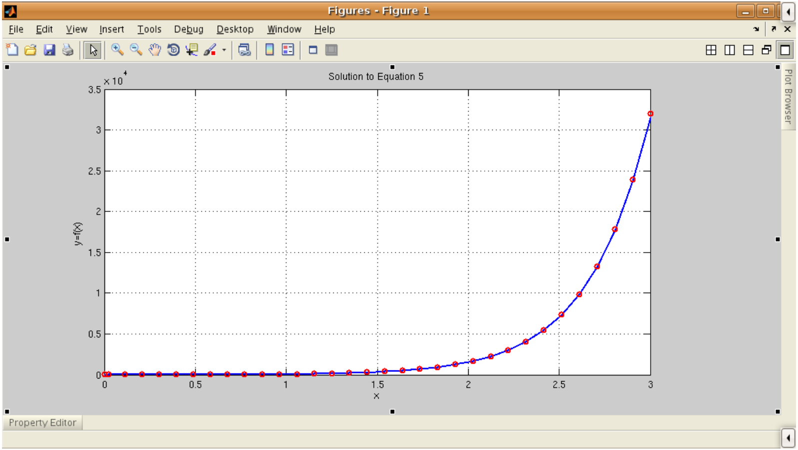
EXAMPLE 5:

```
% plot analytical and numerical solutions
% [a, b] = [0, 3]
% initial condition y=f(0)=3
[x, num_y]=ode23('g5', 0, 3, 3);
y=4*exp(3*x)-exp(2*x);
subplot(2,1,1),plot(x,num_y,x,y,'o'),...
title('Solution to Equation 5'),...
xlabel('x'), ylabel('y=f(x)'), grid
```

EXAMPLE 5:


MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Shortcuts How to Add What's New
Current Directory
All Files Type
g4.m M-file
g5.m M-file
ode_g1.m M-file
ode_g2.m M-file
ode_g3.m M-file
ode_g4.m M-file
ode_g5.m M-file
Command History
clc
type ode_g4.m
ode_g4
ode_g5
clear
clc
type ode_g5.m
Command Window
>> type ode_g5.m
% plot analytical and numerical solutions
% [a, b] = [0, 3]
% initial condition y=f(0)=3
[x, num_y]=ode23('g5', 0, 3, 3);
y=4*exp(3*x)-exp(2*x);
subplot(2,1,1),plot(x,num_y,x,y,'o'),...
title('Solution to Equation 5'),...
xlabel('x'), ylabel('y=f(x)'), grid
>>
Start

EXAMPLE 5:


Figures - Figure 1
File Edit View Insert Tools Debug Desktop Window Help
Solution to Equation 5
× 10^4
3.5
3
2.5
2
1.5
1
0.5
0
y=f(x)
0
0.5
1
1.5
2
2.5
3
x
Plot Browser
Property Editor

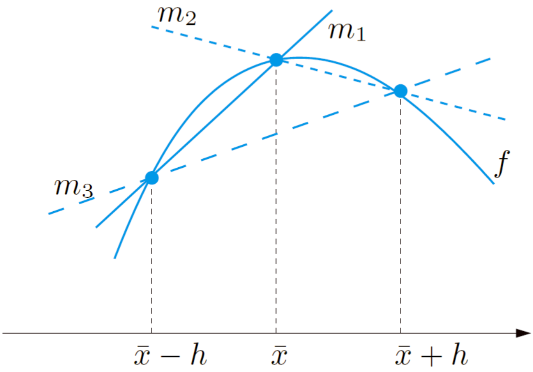
EULER'S METHOD:

- Forward Euler Method
- Backward Euler Method

CRANK-NICOLSON METHOD:

- Adding together the generic steps of the forward and backward Euler methods we find the so-called Crank-Nicolson method.
- $u_{n+1} = u_n + \frac{h}{2}\left(f_n + f_{n+1}\right),\ n = 0, 1, 2, \ldots, N$

CRANK-NICOLSON METHOD:



HIGHER ORDER DIFFERENTIAL EQUATIONS:

- A higher order differential equation can be written as a system of coupled first-order differential equations using a change of variables.
- n^{th} order differential equation:

$$y(n) = g(x, y, y', y'', \ldots, y^{(n-1)})$$

- Define n new unknown functions with these equations
  - $u_1(x) = y^{(n-1)}$
  - $u_2(x) = y^{(n-2)}$
  - ...
  - $u_{n-2}(x) = y''$
  - $u_{n-1}(x) = y'$
  - $u_n(x) = y$

HIGHER ORDER DIFFERENTIAL EQUATIONS:

- The system of n first-order equations is equivalent to the nth order differential equation.
  - $u_1' = y^{(n)} = g(x, u_n, u_{n-1}, \ldots, u_1)$
  - $u_2' = u_1$
  - $\ldots$
  - $u_{n-2}' = u_{n-3}$
  - $u_{n-1}' = u_{n-2}$

EXAMPLE

- $y'' = g(x, y, y') = y'\,(1 - y^2) - y$
- Define two new functions
  - $u_1(x) = y'$
  - $u_2(x) = y$
- Obtain this system of coupled first-order differential equations
  - $u_1' = y'' = g(x, u_2, u_1) = u_1\,(1 - u_2^2) - u_2$
  - $u_2' = u_1$

EXAMPLE

- Matlab `ode` functions return solutions for each of the first order differential equations.
- First we define a function to compute values of the first-order differential equations.
- Then, to solve the system of first-order differential equations over the interval $[0, 20]$ we use initial conditions of $y'(0) = 0.0$ and $y(0) = 0.25$.

EXAMPLE :

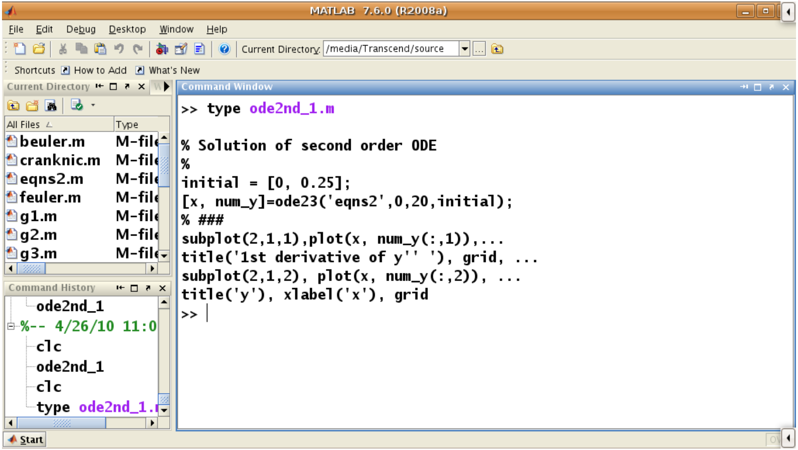
```
function u_prime=eqns2(x,u)
% This function computes values
% for two coupled equations
u_prime(1)=u(1)*(1-u(2)^2)-u(2);
u_prime(2)=u(1);
u_prime=u_prime';
```

EXAMPLE :

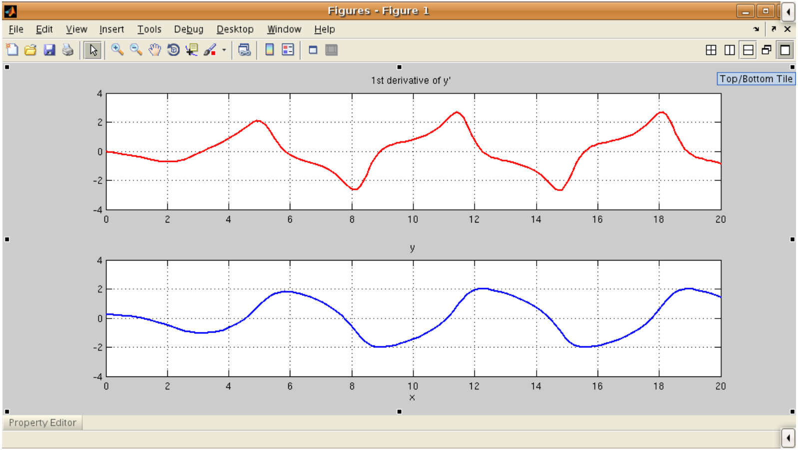
```
% Solution of second order ODE
%
initial = [0, 0.25];
[x, num_y]=ode23('eqns2',0,20,initial);
% ###
subplot(2,1,1),plot(x, num_y(:,1)),...
title('1st derivative of y'' '), grid, ...
subplot(2,1,2), plot(x, num_y(:,2)), ...
title('y'), xlabel('x'), grid
```

ORDINARY DIFFERENTIAL EQUATIONS

EXAMPLE :



EXAMPLE :


Figures - Figure 1
File Edit View Insert Tools Debug Desktop Window Help
Top/Bottom Tile
1st derivative of y'
y
x
Property Editor

MATLAB ODE FUNCTIONS

Solver	problem type	Order of accuracy	When to use
ode45	nonstiff	medium	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.

References

References for Week 10

1. Etter, D.M., Engineering Problem Solving with Matlab, Prentice Hall, 1997.
2. Alfio Quarteroni, Fausto Saleri, Scientific Computing with Matlab and Octave, Springer, 2006.