Programlama -1

“Matlab Hesaplama”

Dr. Cahit Karakuş, 2020

Programlama Dilini İyi Öğrenebilemek

1- For, IF, While ... Gibi döngüsel işlemler
2- Çizim 2-D, 3-D
3- Dosyadan okuma ve dosyaya yazma
4- Matlab, Python, Java Script, C++

Programlama Dilinde Matematik

1- Değişkenler, Sabitler, katsayılar ve birimler
2- Lineer denklem
3- Vektör, Dizi ve Matris
4- Türev, Limit, Integral
5- Diferansiyel Denklemler
6- FFT, Laplace, Z- dönüşümler
7- Nümerik Analiz (Interpolasyon, Regrasyon)
8- İstatistik ve Olasılık(Bayes, Markov)

Common Powers

Prefix	Symbol	Power of 10	Power of 2	Prefix	Symbol	Power of 10
Kilo	K	1 thousand = 10^3	$2^{10} = 1024$	Milli	m	1 thousandth = 10^{-3}
Mega	M	1 million = 10^6	2^{20}	Micro	μ	1 millionth = 10^{-6}
Giga	G	1 billion = 10^9	2^{30}	Nano	n	1 billionth = 10^{-9}
Tera	T	1 trillion = 10^{12}	2^{40}	Pico	p	1 trillionth = 10^{-12}
Peta	P	1 quadrillion = 10^{15}	2^{50}	Femto	f	1 quadrillionth = 10^{-15}
Exa	E	1 quintillion = 10^{18}	2^{60}	Atto	a	1 quintillionth = 10^{-18}
Zetta	Z	1 sextillion = 10^{21}	2^{70}	Zepto	z	1 sextillionth = 10^{-21}
Yotta	Y	1 septillion = 10^{24}	2^{80}	Yocto	y	1 septillionth = 10^{-24}

Why Matlab?

Matlab is high-performance language for technical computing.

Advantages:

- Easy to use:
  - No dimensioning required.
  - No compiling
  - Good for prototyping, testing
- Functionality is greatly expanded by toolboxes.
- Many fields of application.

Disadvantages:

- Uses much memory (slow for large applications)
- Not good for final software design

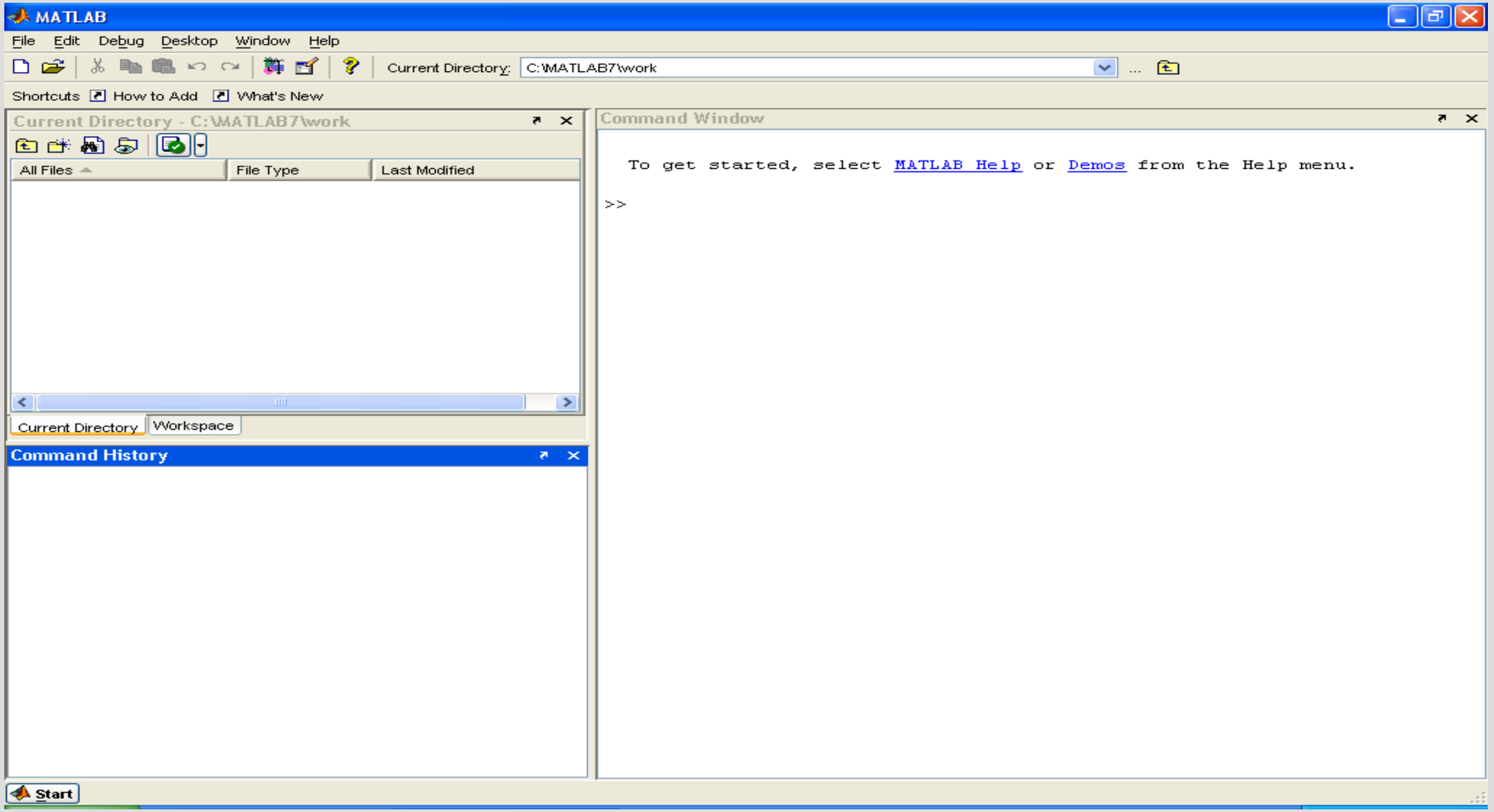
What is matlab?

- Matlab language
  - matrix/array language
  - control flow (if, for, while)
  - functions
- Working environment
  - workspace for computing, importing, and exporting data
  - text editor for creating M-files

Matlab Covers

- Using the development environment.
- Syntax of the language.
- Basic matrix operations.
- Basic graphics features.
- Writing Matlab scripts and functions (m-files)
- The toolboxes including Simulink.

Introduction



How to write an M-file

File → New → M-file

Takes you into the file editor

Enter lines of code (nothing happens)

Save file (we will call ours L2Demo.m)

Exit file

Run file

Edit (*ie* modify) file if necessary

Clear Commands

Clearing the screen: clc

Clearing all variables: clear

Clearing a variable x: clear x

close all: tümünü kapat, tutamaçları gizlenmemiş tüm şekilleri siler..

Matlab’s Workspace

who, **whos** – current workspace vars.

save – save workspace vars to *.mat file.

load – load variables from *.mat file.

clear all – clear workspace vars.

close all – close all figures

clc – clear screen

clf – clear figure

Spacing

- Bir komut satırının farklı bölümleri arasındaki boşluklar, belirli bir komut işlevini veya numarasını bozmamak kaydıyla, “kalabalık” bir görünümden kaçınmak için kullanılabilir.
- Bu nedenle, x=5 , x = 5 ve x = 5 kabul edilebilir.
- Ancak sayı 51 ise 5 ile 1 arasına boşluk koymaya çalışırsanız hata mesajı alırsınız.
- Aynı şekilde bir x1 değişkeni x 1 olarak ifade edilemez.

Basic Commands

% used to denote a comment

; suppresses display of value (when placed at end of a statement)

... continues the statement on next line

eps machine epsilon

inf infinity

NaN not-a number, e.g., 0/0.

Numbers

Numbers

To change format of numbers:

format long, format short, etc.

See "**help format**".

Mathematical functions: **sqrt(x)**, **exp(x)**, **cos(x)**, **sin(x)**, **sum(x)**, etc.

Operations: **+**, **-**, ***, **/**

Constants: **pi**, **exp(1)**, etc.

Sabitler, Değişkenler

F(t)=at^2+bt+c ifadesinde

- Sabitler: a, b, c
- Değişkenler: t, f(t)
- Bağımsız değişken: t
- Bağımlı değişken: f(t)

Numbers

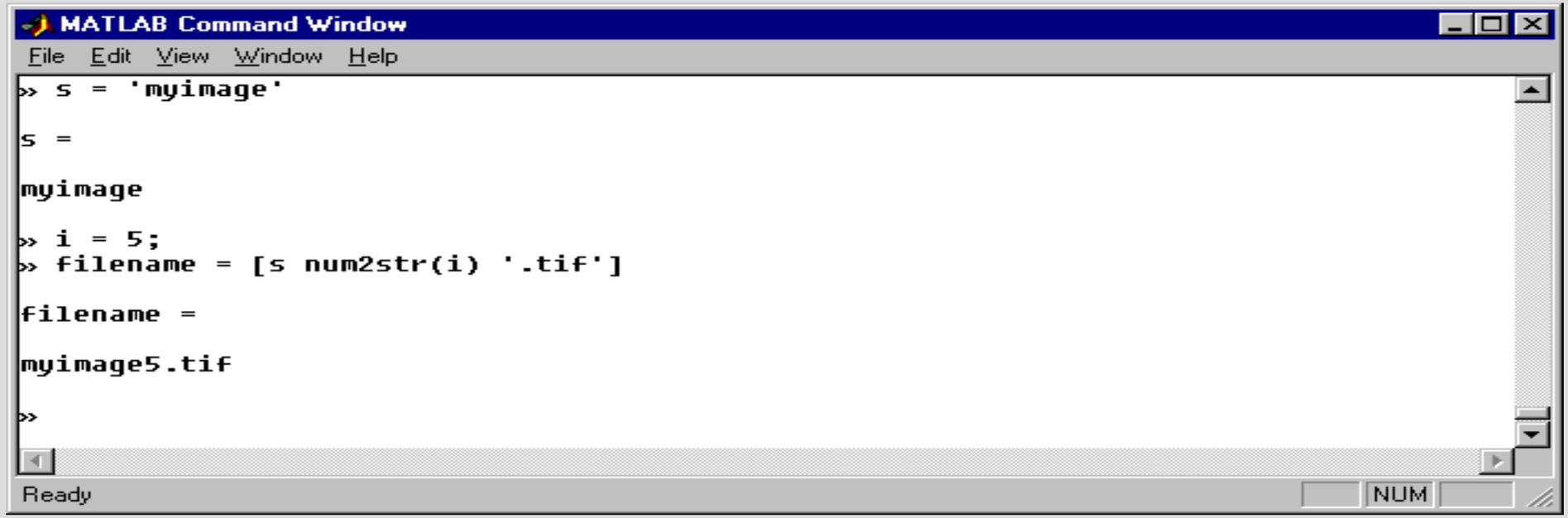
Integer	`-3, 3`
Real	`0.0001, -0.25`
Exponential	`1.60210e-20`
Imaginary	`5 + 2i`

Characters and Text

`s = 'myimage'`

7 character array (string)

`num2str()` - convert a number to string



Basic Arithmetic Operations

Takip eden slaytların birçoğunda, komut penceresinin bir hesap makinesiyle hemen hemen aynı şekilde kullanılabileceğini göstermek için basit sayılar kullanılarak temel aritmetik işlemler kullanılacaktır.

Slaytlarda yer kazanmak için bilgisayar ekranında görünen boş satırların çoğu elenecektir.

Entering Numbers

```
>> 5
ans = 5
>> x = 5
x = 5
>> y = 3
y = 3
```

Addition and Subtraction

```
>> z1 = 5 + 3
z1 = 8
>> z1 = x + y
z1 = 8
>> z2 = y - x
z2 = -2
```

Multiplication

Multiplication is performed by placing an asterisk (*) between the numbers.

```
>> z3 = 5*3
z3 =15

>> z3 = x*y
z3 = 15
```

Division

Division of x by y is performed as follows:

```
 >> z4 = x/y
z4 =    1.6667
```

An alternate form is as follows:

```
>> z4 = y\x
z4 =  1.6667
```

Exponentiation

Exponentiation is performed by using the symbol (^).

```
>> z5 = x^2
z5 = 25
```

Suppressing the Listing of Results

If it is desired to perform a computation and not show it on the screen, the command should be followed by a semicolon (;). For example,

```
>>z6 = y^x;
```

The value is in memory and may be seen by entering the variable name.

```
>> z6
z6 =
   243
```

Entering in Exponential Form

A microwave frequency of 15 GHz (1 GHz = 10^9 Hz) may be entered as

```
>> f = 15e9
f =  1.5000e+010
```

Boltzmann's constant k = 1.38×10^{-23} J/K may be entered as

```
>> k = 1.38e-23
k =  1.3800e-023
```

Square Root

```
>> z7 = sqrt(x)

z7 =

   2.2361
```

Hierarchy

- Parantezler olmadan, aritmetik işlemlerin normal hiyerarşisi üs alma, çarpma, bölme, toplama ve çıkarmadır.
- Sıralamayı değiştirmek için parantezler kullanılabilir. Şüpheye düştüğünüzde, siparişin düzgün bir şekilde gerçekleştirildiğinden emin olmak için parantez eklemek akıllıca olacaktır.
- Parantezlerin parantez içine yerleştirilmesi işlemine iç içe yerleştirme denir.

Consider the algorithm shown below.

1. Add 3 to x.
2. Square the result of step 1.
3. Multiply the result of step 2 by 6.
4. Add 8 to the result of step 3.
5. Take the square root of step 4 and the value obtained is y.

The algorithm may be implemented by MATLAB on a step-by-step basis.

```
u1 = 3 + x ;
u2 = u1^2 ;
u3 = 6*u2 ;
u4 = 8+u3 ;
y = sqrt(u4)
```

Some MATLAB Constants

```
>> pi
ans =
   3.1416
>> eps
ans =
  2.2204e-016
>> sqrt(-1)
ans =
0 + 1.0000i
```

The next slide shows 4 ways to enter an imaginary number.

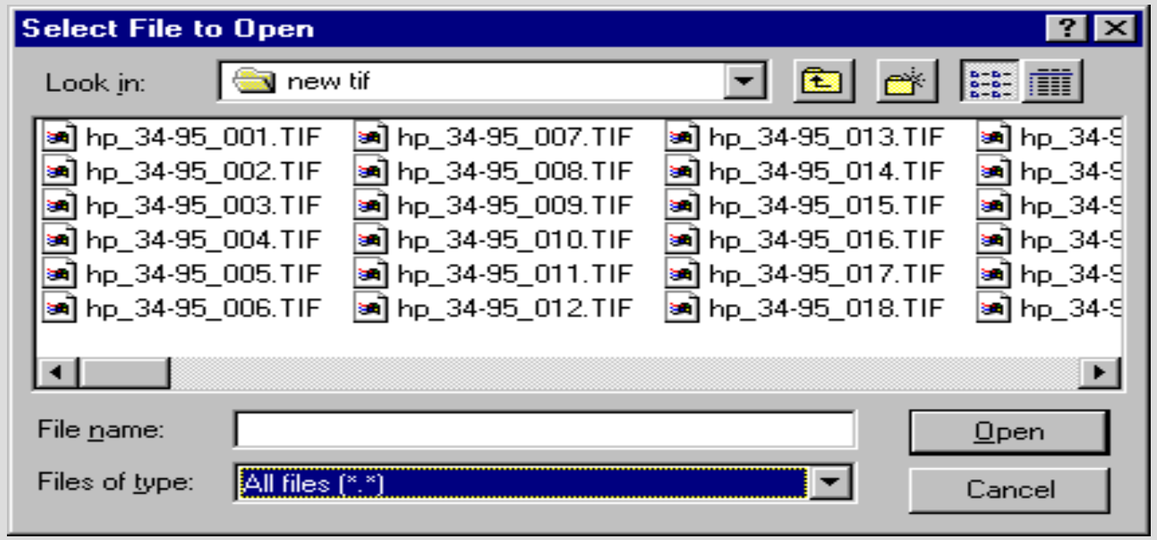
User input

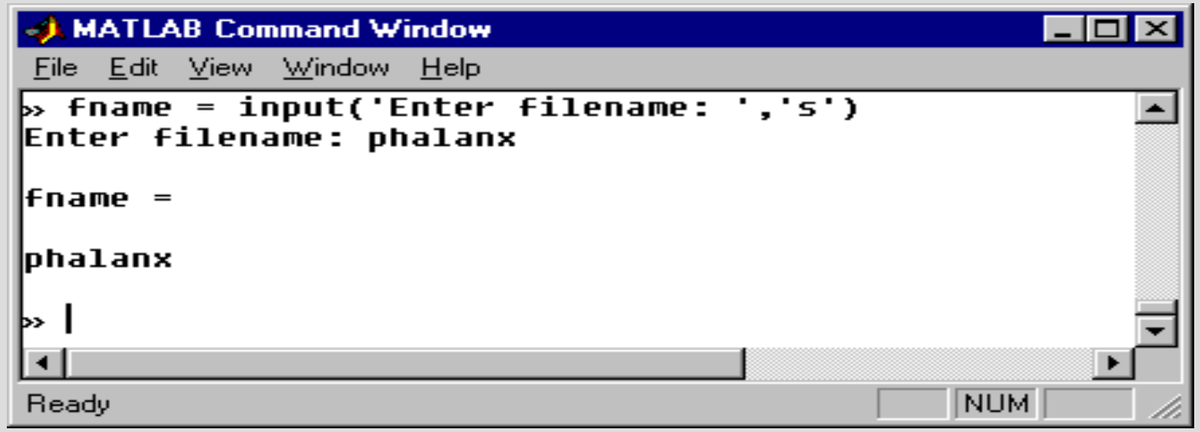
- Keyboard input

```
a_number = input('Enter number');
a_string = input('Enter filename', 's');
```

- Using GUI

```
[filename,path] = uigetfile('*.*');
```





Format Long

Long, fixed-decimal format with 15 digits after the decimal point for double values, and 7 digits after the decimal point for single values. pi=3.141592653589793

longE: Long scientific notation with 15 digits after the decimal point for double values, and 7 digits after the decimal point for single values. 3.141592653589793e+00

Bank: Currency format with 2 digits after the decimal point. 3.14

Rat: Ratio of small integers. 355/113

```
>> 5i
ans =        0 + 5.0000i
>> 5j
ans =        0 + 5.0000i
>> i*5
ans =         0 + 5.0000i
>> j*5
ans =         0 + 5.0000i
```

Division by Zero

- Divison by zero will either yield **Inf** (“infinity”) or **NaN** (“not a number”) depending on the form.
- As examples, if the command **1/0** is entered, the result is **Inf**, and if the command **0/0** is entered, the result is **NaN**.

“Operators”

İşlemler

Aritmetik İşlemler: Toplama, Çıkarma, Çarpma, Bölme, ..

Mantıksal İşlemler: AND, OR, NOT, NAND, NOR, ...

Karşılaştırma: Büyük, küçük, eşit, eşit deüil ...

Veri transferi: Bellek; veri tabanı

I/O: Monitor, Klavye

Operators: Arithmetic

+	Addition
-	Subtraction
*	Multiplication (or .* element-wise)
/	Division
^	Power (or .^ element-wise)
'	Transpose
()	Brackets for evaluation order

Operators: Matris - Vektör

```
A = [ 4   1   3]
A.^2 = [16   1   9]
A.*2 = [ 8   2   6]
```

```
A = [ 4   1   3]
A: bir satır vektörü
A'=[4;1;3]
B=A*A'=[16;1;9]
A'*A=?
```

$$A'*A=\begin{bmatrix} 16 & 4 & 12 \\ 4 & 1 & 3 \\ 12 & 3 & 9 \end{bmatrix}$$

- Skaler çarpım (.), vektör ya da matrisin sonunda nokta var, eleman eleman işlem görür.
- Vektörel çarpım: Satır sütun ile çarpılır; çarpılan elemanlar toplanır. Satırdaki eleman sayısı sütundaki eleman sayısına eşit olmalıdır.

Logical Conditions

==, **<**, **>**, **<=**, **>=**, **~=** (not equal), **~** (not)

& (element-wise logical and), **|** (or)

find('condition') – Return indices of A's elements that satisfies the condition.

Example: **A = [7 6 5; 4 3 2];**

find ('A == 3'); --> returns 5.

Operators: Relational

< Less than

<= Less than or equal to

> Greater than

>= Greater than or equal to

== Equal to

~= Not equal to

Operators: Logical

&	AND
|	OR
~	NOT

Example revisited

$$y = \sum_{i=1}^{n} \frac{1}{\sqrt{i}} = \frac{1}{\sqrt{1}} + \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{3}} + \ldots$$

Can do using MATLAB as a calculator

```
i = 1:10;
term = 1./sqrt(i);
y = sum(term);
```

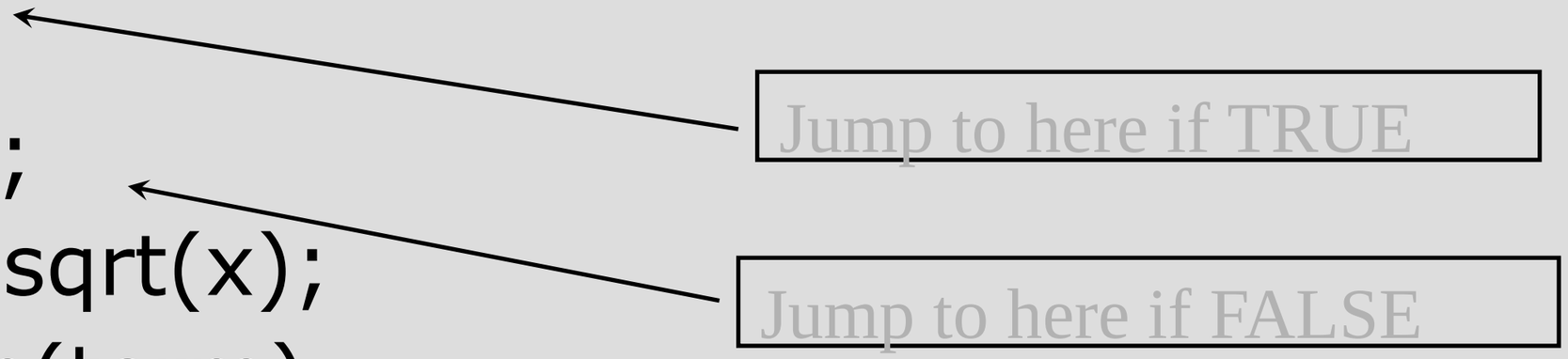
L2Demo Version 1

```
n = input('Enter the upper limit: ');
x = 1:n;          % Matlab is case sensitive
term = sqrt(x);
y = sum(term)
```

What happens if n < 1 ?

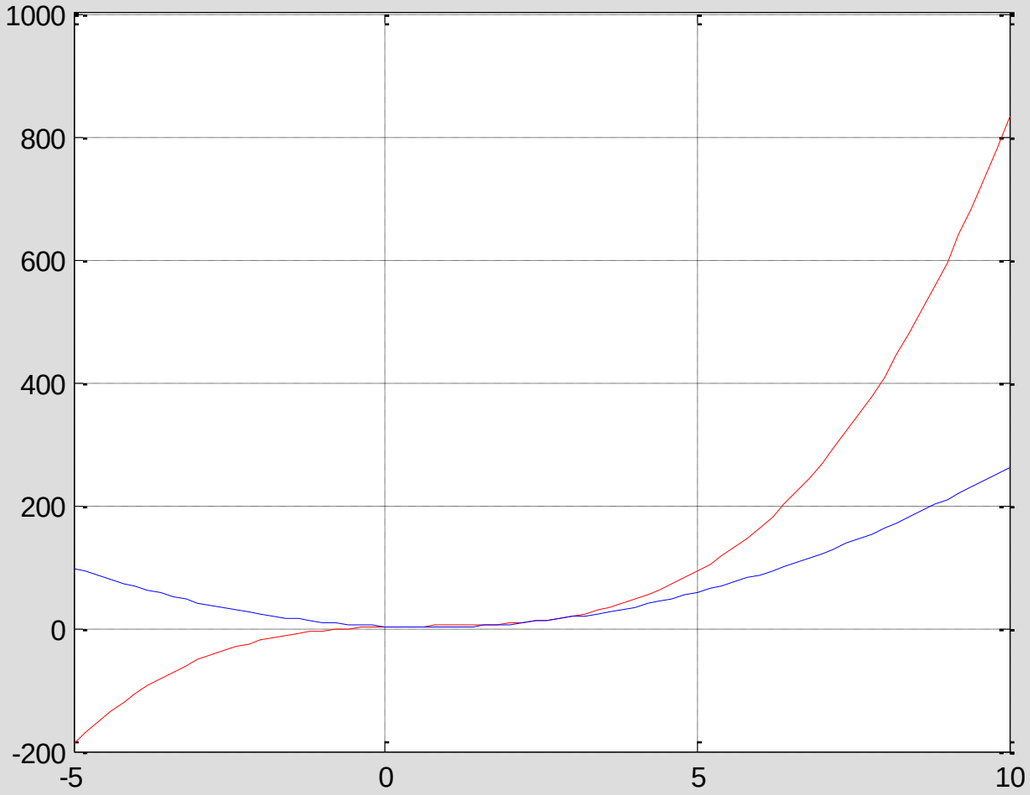
L2Demo Version 2

```
n = input('Enter the upper limit: ');
if n < 1
    disp ('Your answer is meaningless!')
end
x = 1:n;
term = sqrt(x);
y = sum(term)
```



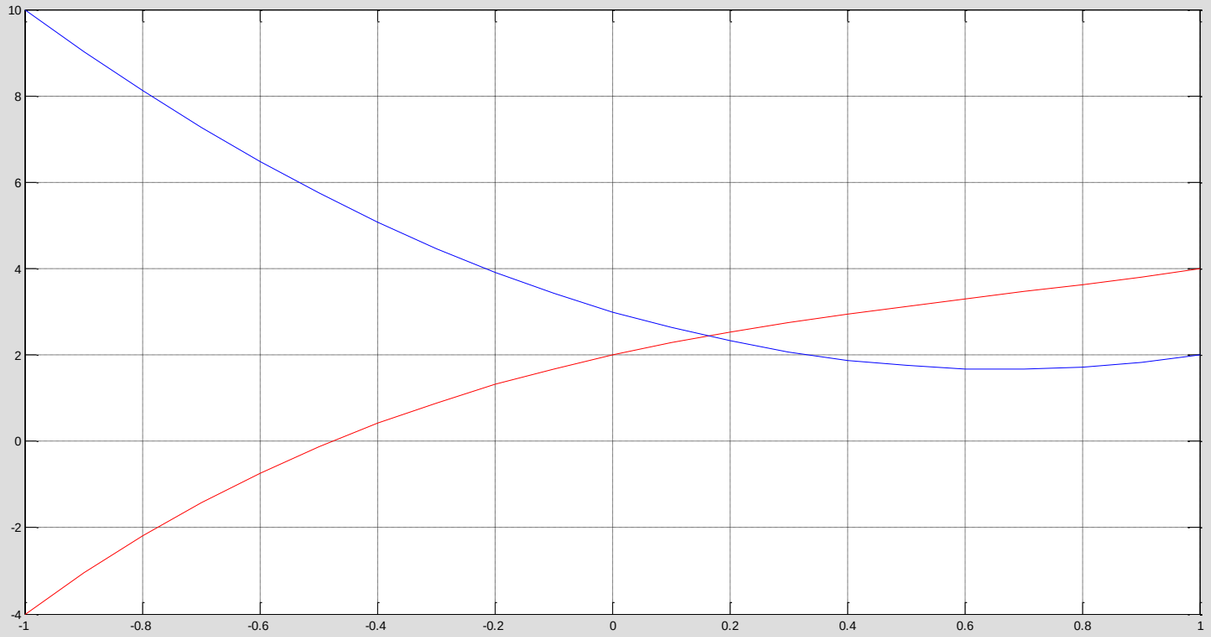
Fonksiyon ve Türevi

```
clear all
close all
x = -5:.2:10;
N = length(x);
for i=1:N
y(i) = x(i)^3 - 2*x(i)^2 + 3*x(i) + 2;
yt(i)=3*x(i)^2 - 4*x(i) + 3;
end
plot(x,y,'r',x,yt,'b')
grid on
```



Fonksiyon ve Türevi: Örnekleme Aralığı

```
clear all
close all
x = -1:.1:1;
N = length(x);
for i=1:N
y(i) = x(i)^3 - 2*x(i)^2 + 3*x(i) + 2;
yt(i)=3*x(i)^2 - 4*x(i) + 3;
end
plot(x,y,'r',x,yt,'b')
grid on
```



Polynomial roots

r = roots(p)
Örnek:
p = [3 -2 -4];
r = roots(p)

Sonuç:
 1.5352
   -0.8685

Flow control

Flow control

if, else, and elseif

switch

while

for

end

if, else, and elseif

```
if karşılaştırma ifadesi
        statements
end
```

```
n = input('Bir sayı giriniz');
if (n<0)
  disp('Pozitif bir sayı giriniz')
elseif (n == 0)
  disp('a is even')
else
  disp('a is odd')
end
```

MATLAB/if,end yapısı

if (eğer) yapısı bir koşulun gerçekleşmesi durumunda bir işlemi yaptırmak için sıklıkla kullanılır. Bu ifade,

```
if koşul
   işlem
end
```

```
if koşul
  işlem
  else
  işlem
  end
```

```
if koşul
işlem
 elseif
 işlem
 else
    işlem
 end
```

biçimlerindedir.

Örnek: Girilen bir sayının negatif olması durumunda, sayıyı doğal logaritmasıyla değiştiren bir kod düşünelim:

```
a=input(' bir sayi giriniz=');
if a<0
   a=log(a);
else
a=a;
end
a
```

"Diğer durumda" anlamındadır: Burada, a>0 koşulunu temsil eder.

Else yapısı kullanılmasaydı

```
a=input(' bir sayi giri
if a<0
   a=log(a);
end
if a>0
   a=a;
```

MATLAB/if,end yapısı

if (eğer) yapısı bir koşulun gerçekleşmesi durumunda bir işlemi yaptırmak için sıklıkla kullanılır. Bu ifade,

```
if koşul
   işlem
end
```

```
if koşul
  işlem
  else
  işlem
  end
```

```
if koşul
işlem
 elseif
 işlem
 else
    işlem
 end
```

biçimlerindedir.

Örnek: Girilen bir sayının negatif olması durumunda, sayıyı poitife çevirip doğal logaritmasını bulan bir kod düşünelim:

```
clear all
close all
a=input(' bir sayi giriniz= ');
    if a<0
      b=log10(abs(a));
    else
    b=log10(a);
    end
    b
```

(If – End)statements

```
clear all
close all

x=12

if (x == 3)
    disp('The value of x is 3.');
  elseif (x == 5)
    disp('The value of x is 5.');
  else
    disp('The value of x is not 3 or 5.');
  end;
```

switch

```
switch expression
  case value1
    statements
  case value2
    statements
  otherwise
    statements
end
```

```
switch (input_num)
  case (-1)
    disp('negative one')
  case (0)
    disp('zero')
  case (1)
    disp('positive one')
  otherwise
    disp('othervalue')
end
```

For Loops

```
clear all
close all
N=5
M=5
for i=1:N          % use for-loops to execute iterations / repetitions
    for j=1:M
        a(i, j) = i + j ;
    end
end
a
```

(For – End) loops

```
x = 0;

for I =1:2:5        % start at 1, increment by 2
    x = x+i;          % end with 5.
 end
```

This computes x = 0+1+3+5=9.

For … end

```
clear all
close all
 b=-5:1:5;
for i=1:11
    a(i)=-5+i-1;
end
a
b
```

while, for

```
while expression
  statements
end
```

```
n = 1;
while (prod(1:n) < 1e10)
  n = n + 1;
end
```

```
for index = start:inc:end
  statements
end
```

```
for i = 2:6
  x(i) = 2*x(i-1);
end
```

```
for i = 1:m
  for j = 1:n
    A(i,j) = 1/(i + j - 1);
  end
end
```

(While End) loops

```
x=7;
   while (x > = 0)
     x = x-2;
   end;
```

This computes x = 7-2-2-2-2 = -1.

(Switch – Eend) statement

```
clear all
close all

face=3
switch face
    case {1}
      disp('Rolled a 1');
    case {2}
      disp('Rolled a 2');
    otherwise
      disp('Rolled a number >= 3');
    end
```

NOTE: Unlike C, ONLY the SWITCH statement between the matching case and the next case, otherwise, or end are executed. *(So breaks are unnecessary.)*

Break statements

break – terminates execution of for and while loops. For nested loops, it exits the innermost loop only.

Vectorization

Efficiency!

Example: Table of logarithms

Find log10 of numbers between 0 and 10.

```
x = 0;
for k = 1:1001
  y(k) = log10(x);
  x = x + 0.01;
end
```

Looped code

```
x = 0:0.01:10;
y = log10(x);
```

Vectorized code

AVOID LOOPS!

Vectorization: Toplama

```
clear all
close all
x=[1 2 3 7 1];
Top=0;
N=length(x);
for i=1:N
    Top=Top+ x(i);
end;
Top
```

Vectorization

- Because Matlab is an interpreted language, i.e., it is not compiled before execution, loops run slowly.
- *Vectorized code runs faster* in Matlab.

```
clear all
close all

x=[1 2 3 7 1]
a=5;
N=length(x);
for i=1:N
    Top(i)=a+ x(i);
end;

Top
```

```
clear all
close all

x=[1 2 3 7 1]
a=5;
Top=a+ x
```

“Calculus Operations”

Calculus Operations

We are now ready to see how calculus operations can be performed with MATLAB. It should be noted that a digital computer operates on a numerical basis with addition, subtraction, multiplication, and division, along with memory storage and logic. True differentiation and integration can never be achieved **exactly** with numerical processes. However, there are two ways that differentiation and integration can be achieved on a practical level with MATLAB.

Two Types of Operations

First, MATLAB can be viewed as a very comprehensive "look-up" table in that numerous derivatives and integrals as character strings can be manipulated with software. The **Symbolic Math Toolbox** permits the exact formulas for derivatives and integrals to be extracted, manipulated, and plotted.

Two Types of Operations (Continued)

Second, MATLAB can be used for calculus with **numerical approximations** to differentiation and integration. While such approximations are not exact, they can be used to provide extremely close approximations to the derivatives and integrals of experimental data, even when closed form solutions are impossible.

Symbolic Variables

A symbolic variable is one that can be manipulated in the same manner as in an equation, and it may or may not ever take on any numerical values. To make **x** and **a** symbolic, the command is

```
>> syms x a
```

Alternately, the symbolic portion of a command may be enclosed with apostrophes ' '.

Symbolic Differentiation

The command is **diff( )** with the function to be differentiated in parentheses. All variables should have been established as symbolic variables or the function should be enclosed by apostrophes.

Use MATLAB to determine derivative of function below.

$$y = 4x^5$$

```
syms x
y = 4*x^5

yt = diff(y)

y =4*x^5
yt =20*x^4
```

Second Approach:

```
yt = diff(4*x^5)

yt =20*x^4
```

Third Approach:

```
y = '4*x^5'

y =4*x^5
yt = diff(y)

yt =20*x^4
```

Use **ezplot** to plot y and yprime.

The simplest form is **ezplot(y)**, but the domain will be from -2π to 2π. The domain can be changed by

ezplot(y, [x1 x2])

In this example,

ezplot(y, [-1 1])

ezplot(yprime,[-1 1])

The plots after labeling are shown on the next two slides.

Figure 8-1. Plot of 5th degree function of Examples 8-1 and 8-2.

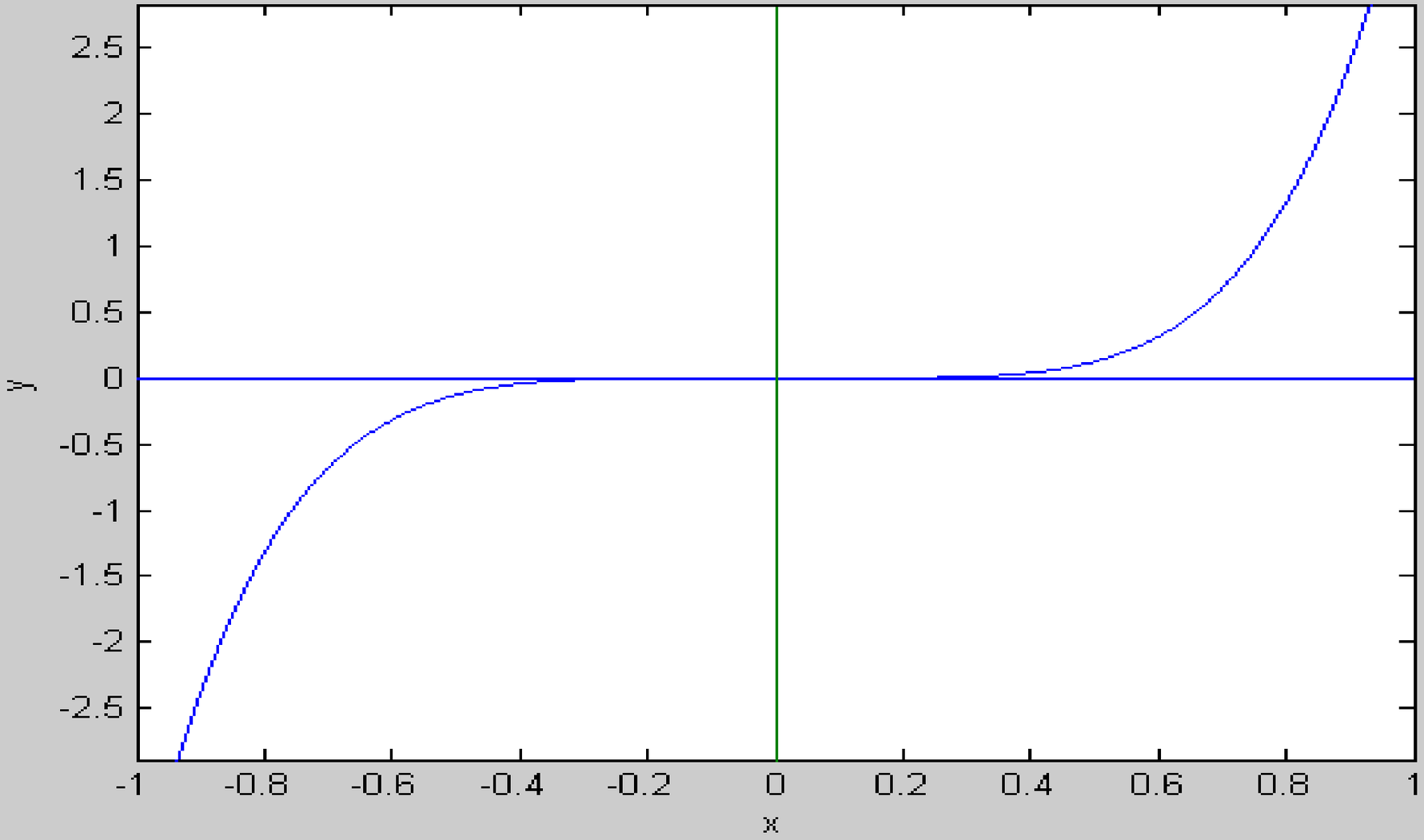
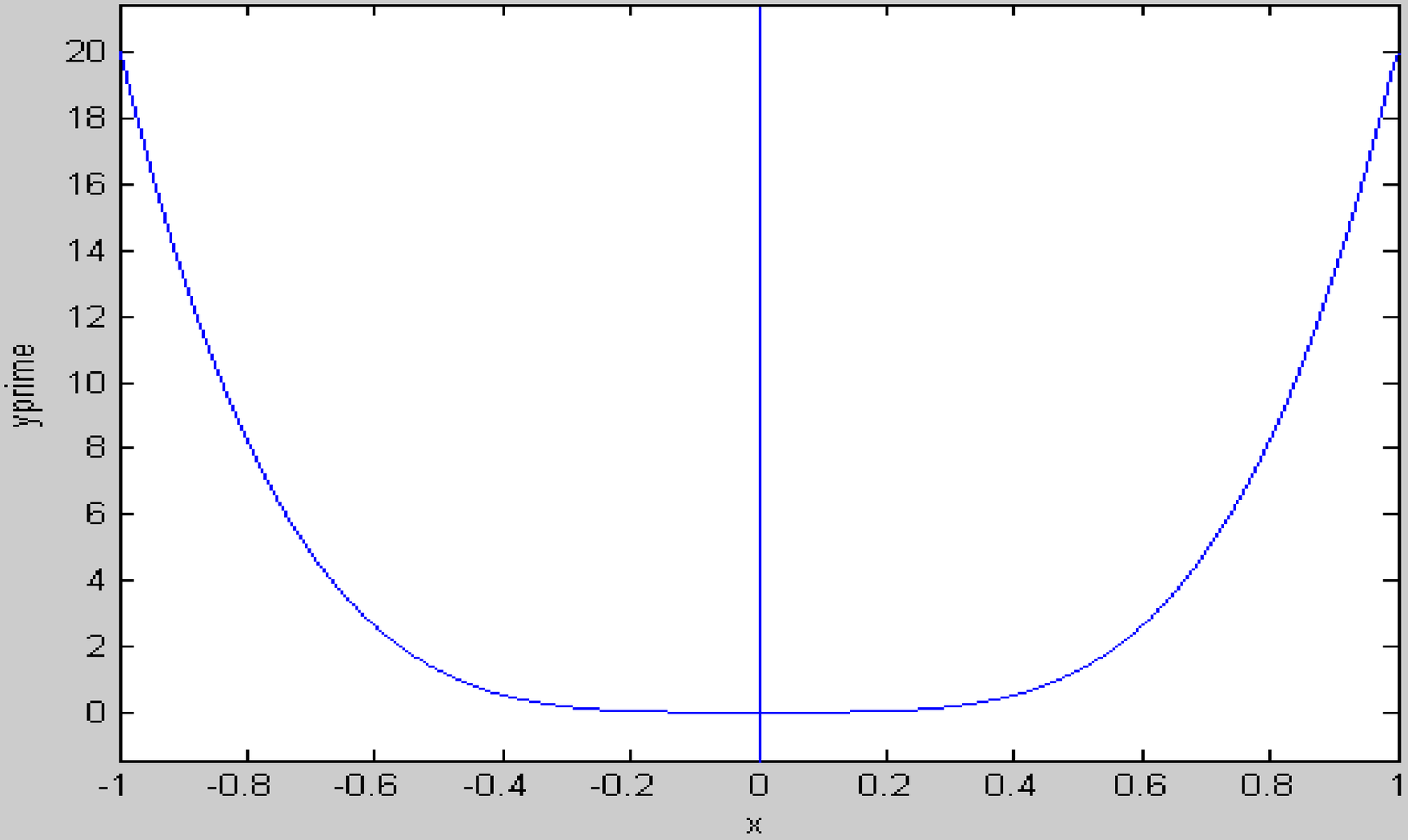


Figure 8-2. Plot of derivative of function of Figure 8-1.



Symbolic Integration

The command is **int( )** with the function to be integrated in parentheses. All variables should have been established as symbolic variables or the function should be enclosed by apostrophes**.**

Indefinite Integral:

>> yint = int(y)

Definite Integral:

>> yint = int(y, a, b)

Use MATLAB to determine integral of function below.

$$z = \int y dx = \int x^2 e^x dx \qquad z(0) = 0$$

```
>> syms x
>> y = x^2*exp(x)
y =
x^2*exp(x)
>> z = int(y)
z =
x^2*exp(x)-2*x*exp(x)+2*exp(x)
```

Continuation.

We require that z(0) = 0, but the function obtained has a value of 2 at x = 0. Thus,

```
>> z = z - 2

z =

x^2*exp(x)-2*x*exp(x)+2*exp(x)-2
```

Plots of y and z are shown on the next two slides.

Figure 8-3. Plot of function of Example 8-3.

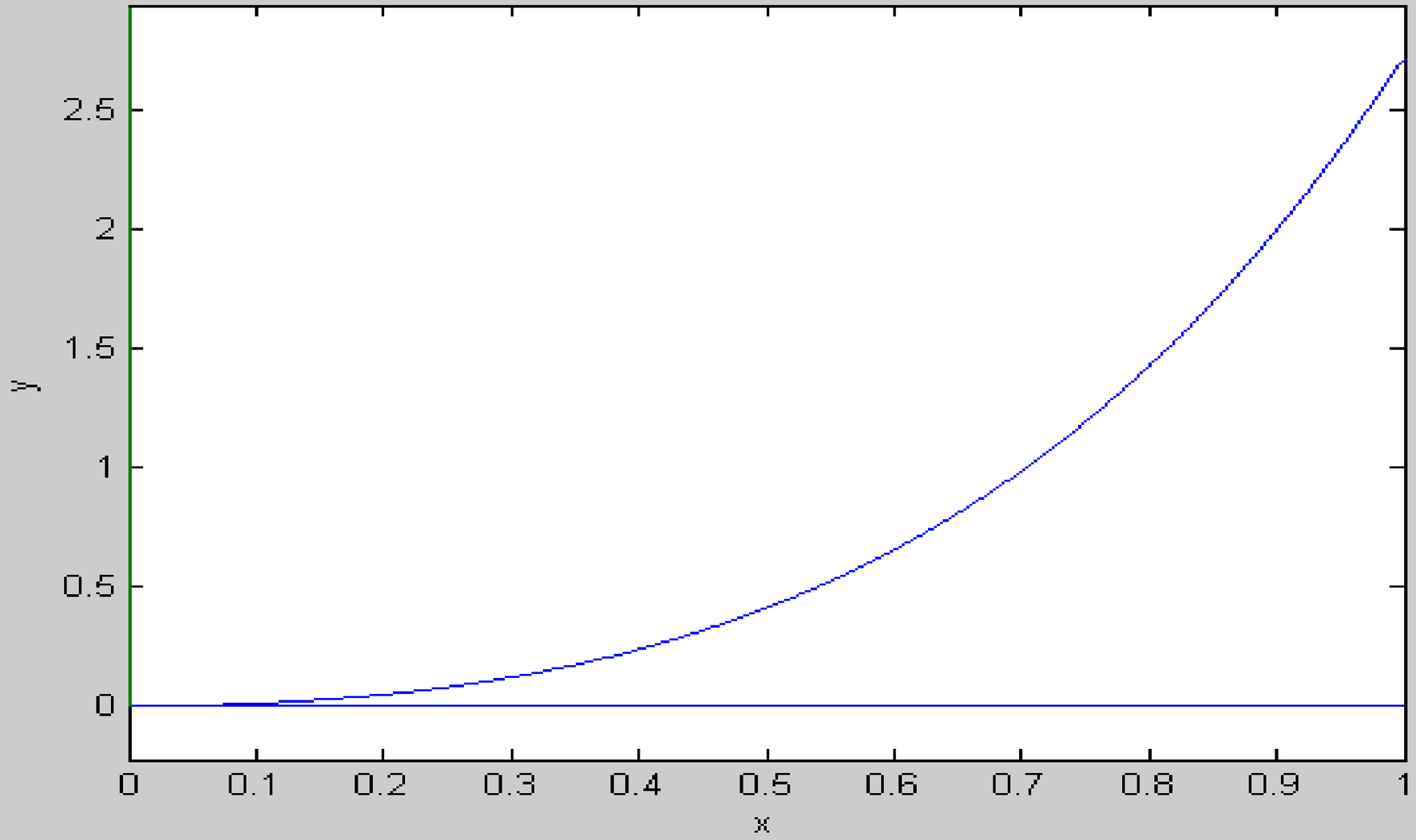
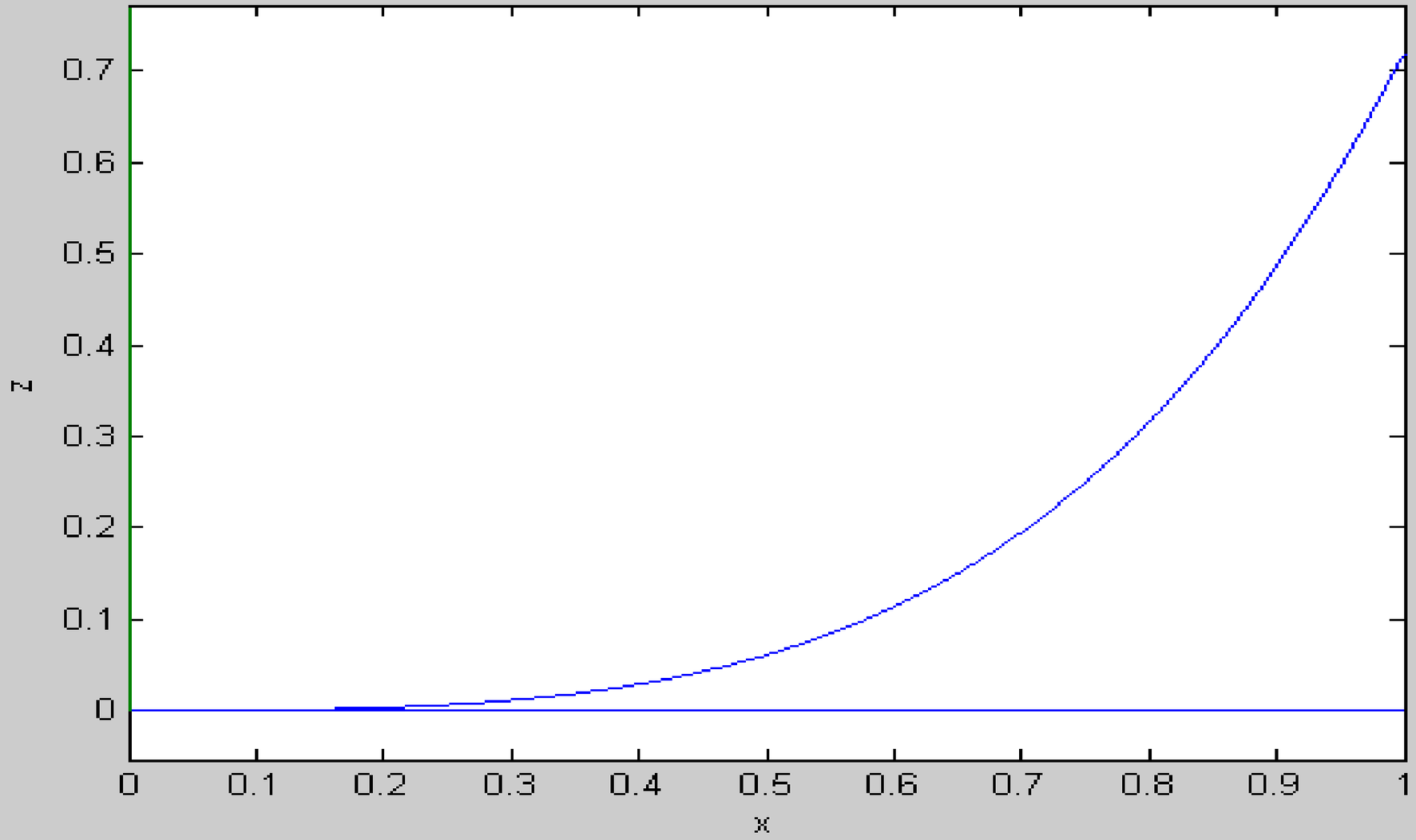


Figure 8-4. Plot of integral of function of Example 8-3.



Numerical Differentiation

Generally, numerical differentiation is more prone to error than numerical integration due to the nature of "sudden changes" in differentiation.

$$\frac{dy}{dx} \approx \frac{\Delta y}{\Delta x}$$

Diff Command for Numerical Data

Assume that *x* and *y* have been defined at *N*+1 points. A vector *u* with *N* points is defined as

$$u_1 = y_2 - y_1$$

$$u_2 = y_3 - y_2$$

$$u_3 = y_4 - y_3$$

$$\cdot \qquad \cdot \qquad \cdot$$

$$u_N = y_{N+1} - y_N$$

Diff Command for Numerical Data. Continuation.

The following command forms u:

```
>> u = diff(y)
```

To approximate derivative,

```
>> yprime = diff(y)/delx
```

Because x and y have one more point than yprime, they can be adjusted.

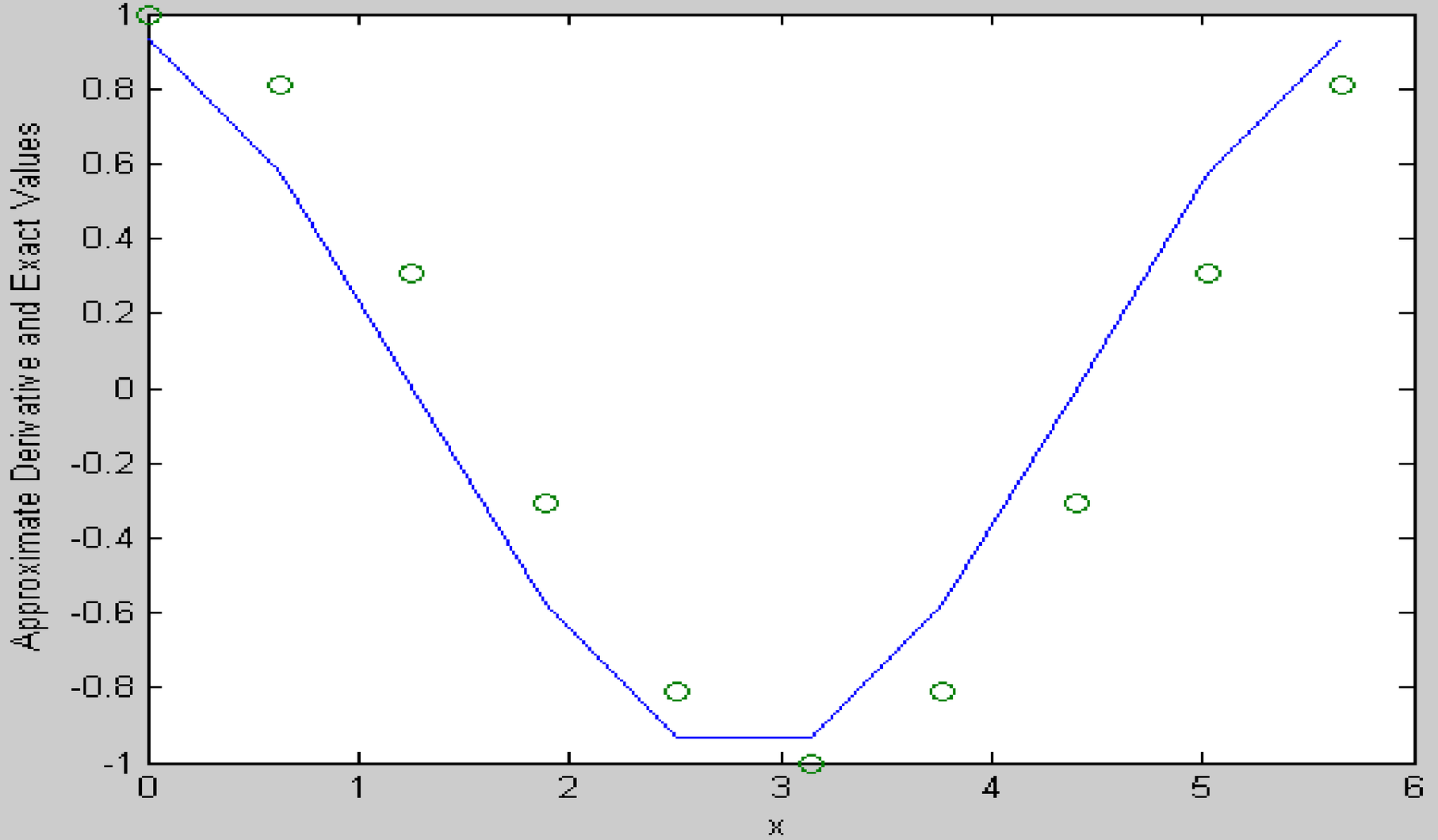
```
>> x = x(1:N)
>> y = y(1:N)
>> plot(x, y, x, yprime)
```

Example 8-5(a) For $y = \sin x$, determine numerical derivative based on 11 points in one cycle.

```
>> x = linspace(0, 2*pi, 11);
>> y = sin(x);
>> delx = 2*pi/10;
>> yprime = diff(y)/delx;
>> x = x(1:10);
>> plot(x, yprime, x, cos(x),'o')
```

The plots are shown on the next slide.The approximation is “crude” as expected.

Figure 8-5. Crude approximation of derivative compared with some exact points.



Example 8-5(b) For $y = \sin x$, determine numerical derivative based on 101 points in one cycle.

```
>> x = linspace(0, 2*pi, 101);
>> y = sin(x);
>> delx = 2*pi/100;
>> yprime = diff(y)/delx;
>> x = x(1:100);
>> plot(x, yprime, x, cos(x),'o')
```

The plots are shown on the next slide.The approximation is much better.

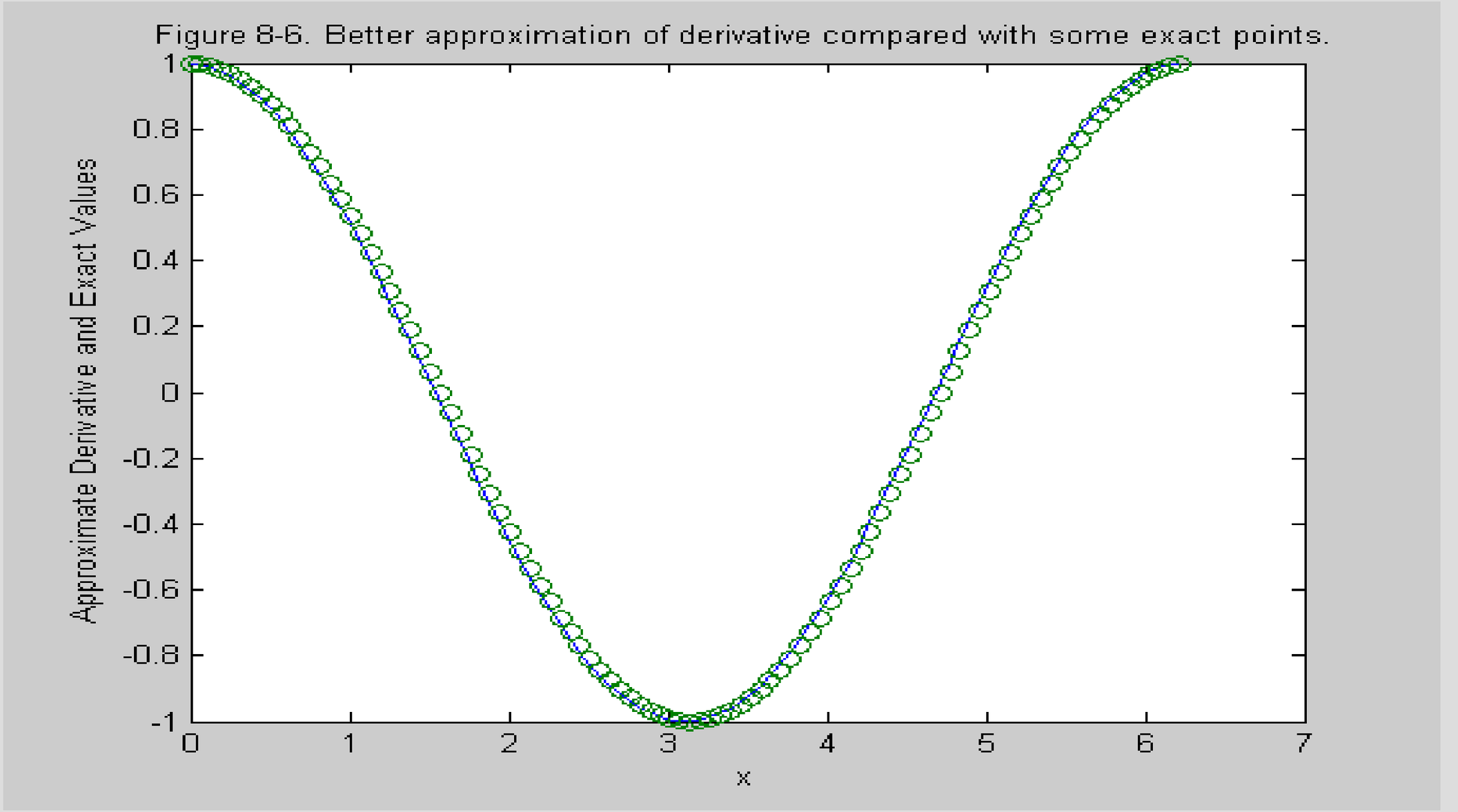


Figure 8-6. Better approximation of derivative compared with some exact points.
Approximate Derivative and Exact Values
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7

Numerical Integration

Two types will be studied:

1. *Zero-order* integration
2. *First-order* integration, which is also known as the *trapezoidal* rule.

Assume vectors *x* and *y* that have each been defined at *N* points.

Zero-Order Integration

$$z_1 = y_1 \Delta x$$

$$z_2 = z_1 + y_2 \Delta x = (y_1 + y_2) \Delta x$$

$$z_3 = z_2 + y_3 \Delta x = (y_1 + y_2 + y_3) \Delta x$$

$$z_k = z_{k-1} + y_k \Delta x = \left(\sum_{n=1}^{k} y_n \right) \Delta x$$

Two Zero-Order MATLAB Commands

The two MATLAB commands for zero-order integration are **sum()** and **cumsum()**.

```
>> area = delx*sum(y)
```

```
>> z = delx*cumsum(y)
```

First-Order Integration

$$z_1 = \left(\frac{0 + y_1}{2} \right) \Delta x$$

$$z_2 = z_1 + \left(\frac{y_1 + y_2}{2} \right) \Delta x$$

$$z_3 = z_2 + \left(\frac{y_2 + y_3}{2} \right) \Delta x$$

$$z_k = z_{k-1} + \left(\frac{y_{k-1} + y_k}{2} \right) \Delta x$$

Two First-Order MATLAB Commands

The two MATLAB commands for first-order integration are **trapz()** and **cumtrapz()**.

```
>> area = delx*trapz(y)
```

```
>> z = delx*cumtrapz(y)
```

Example 8-6(a). Determine exact area *A* for following integral:

$$A = \int_0^2 4x^3 dx$$

$$A = \frac{4x^4}{4} = x^4 \Big]_0^2 = (2)^4 - (0)^4 = 16$$

Example 8-6(b). Determine approximate area A1 with zero-order integration algorithm and step-size of 0.05.

```
>> delx = 0.05;
>> x = 0:delx:2;
>> y = 4*x.^3;
>> A1 = delx*sum(y)
A1 =
   16.8100
```

Example 8-6(c). Determine approximate area A2 with first-order integration algorithm and step-size of 0.05.

```
>> delx = 0.05;
>> x = 0:delx:2;
>> y = 4*x.^3;
>> A2 = delx*trapz(y)
A2 =
   16.0100
```

Example 8-7(a). Determine the exact running integral for the following function:

$$z = \int_0^x \sin x dx$$

$$z = -\cos x \Big]_0^x$$

$$= -\cos x - (-\cos 0)$$

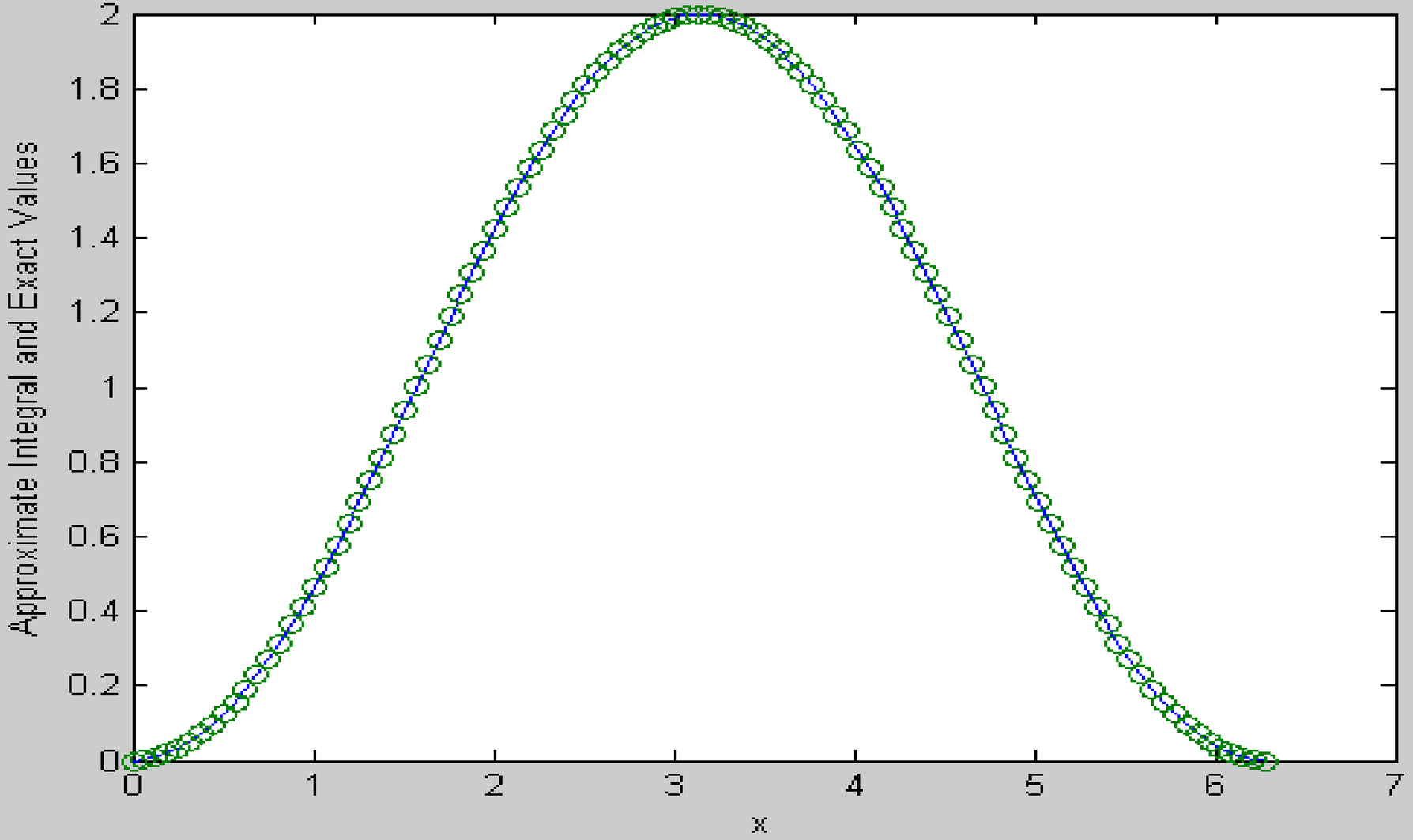
$$= 1 - \cos x$$

Example 8-7(b) and (c). Determine first-order approximation with 100 points per cycle and plot the two functions.

```
>> delx = 2*pi/100;
>> x = 0:delx:2*pi;
>> y = sin(x);
>> z1 = delx*cumtrapz(y);
>> plot(x, z1, x, 1 - cos(x), 'o')
```

The plots are shown on the next slide.

Figure 8-7. First-order approximation compared with exact values in Example 8-7.



Example 8-8

A test is performed on a mechanical part and the acceleration versus time is measured and shown on the next slide. Use MATLAB to determine and plot the velocity and displacement as a function of time.

Acceleration versus Time Data

***t*, s**	**0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
***a*, m/s^2**	**0**	**9.05**	**16.37**	**22.22**	**26.81**	**30.33**	**32.93**	**34.76**	**35.95**	**36.59**
1.0	**1.1**	**1.2**	**1.3**	**1.4**	**1.5**	**1.6**	**1.7**	**1.8**	**1.9**	**2.0**
36.79	**36.62**	**36.14**	**35.43**	**34.52**	**33.47**	**32.30**	**31.06**	**29.75**	**28.42**	**27.07**

MATLAB Analysis

```
>> delt = 0.1;
>> t = 0:delt:2;
>> a = [ the 21 values of a];
>> v = delt*cumtrapz(a);
>> y = delt*cumtrapz(v);
>> plot(t,a,t,v,t,y)
```

Additional labeling was provided and the curves are shown on the next slide.

Figure 8-8. Functions of Example 8-8.