Chapter 3: Branching Statements and Program Design (cont.)

Yunong Zhang (张雨浓)

Email: zhynong@mail.sysu.edu.cn

Switch Construct

```
switch input expression
   case value1
       statement group 1
   case value2
       statement group 2
   case value3
       statement group 3
   …
   otherwise
       statement group n
end
```

Switch Construct (Cont.)

```
switch input expression
   case {value1,value2,value3}
       statement group 1
   case value4
       statement group 2
   case valuei
       statement group i
   ...
   otherwise
       statement group n
end
```

What is the usual meaning/usage of {}?

Switch Construct (Cont.)

```
>> help .
```

```
Operators and special characters.
Arithmetic operators.
  plus       - Plus                            +
  uplus      - Unary plus                      +
  minus      - Minus                           -
  uminus     - Unary minus                     -
  mtimes     - Matrix multiply                 *
  times      - Array multiply                 .*
  mpower     - Matrix power                    ^
  power      - Array power                    .^
  mldivide   - Backslash or left matrix divide  \
  mrdivide   - Slash or right matrix divide     /
  ldivide    - Left array divide              .\
  rdivide    - Right array divide             ./
  kron       - Kronecker tensor product      kron
Relational operators.
  eq         - Equal                          ==
  ne         - Not equal                      ~=
  lt         - Less than                      <
  gt         - Greater than                   >
  le         - Less than or equal             <=
  ge         - Greater than or equal          >=
```

```
Logical operators.
  and        - Logical AND                      &
  or         - Logical OR                       |
  not        - Logical NOT                      ~
  xor        - Logical EXCLUSIVE OR
  any        - True if any element of vector is nonzero
  all        - True if all elements of vector are nonzero
Special characters.
  colon      - Colon                            :
```

Parentheses ( )

Brackets [ ]

Braces { }

```
  punct      - Function handle creation         @
  punct      - Decimal point                    .
  punct      - Structure field access           .
  punct      - Parent directory                 ..
  punct      - Continuation                     ...
  punct      - Separator                        ,
  punct      - Semicolon                        ;
  punct      - Comment                          %
```

```
  punct      - Invoke operating system command   !
  punct      - Assignment                        =
  punct      - Quote                             '
  transpose  - Transpose                         .'
  ctranspose - Complex conjugate transpose       '
  horzcat    - Horizontal concatenation          [,]
  vertcat    - Vertical concatenation            [;]
  subsasgn   - Subscripted assignment            ( ),{ },.
  subsref    - Subscripted reference             ( ),{ },.
Bitwise operators.
  bitand     - Bit-wise AND.
  bitcmp     - Complement bits.
  bitor      - Bit-wise OR.
  bitmax     - Maximum floating point integer.
  bitxor     - Bit-wise XOR.
  bitset     - Set bit.
  bitget     - Get bit.
  bitshift   - Bit-wise shift.
Set operators.
  union      - Set union.
  unique     - Set unique.
  intersect  - Set intersection.
  setdiff    - Set difference.
  setxor     - Set exclusive-or.
  ismember   - True for set member.
```

Switch Example (I)

```
 %var=1:10

var=input(‘enter the data:’);
switch var
   case {1,3,5,7,9}
      disp(‘the variable is odd.’);
  case {2,4,6,8,10}
     disp(‘the variable is even.’);
  otherwise
     disp(‘the variable is out of the range.’);
end
```

Switch Example (I) (Cont.)

```
>>switchexam
     enter the data:3
     the variable is odd.

>>switchexam
     enter the data:2
     the variable is even.
```

Switch Example (II)

cmd9.m

```
response = input(‘Type like, hate, or ok: ’,’s’);
switch response
   case ‘like’
       disp(‘I really like it’);
   case ‘hate’
       disp(‘I do not like it’);
   case ‘ok’
       disp(‘It is ok’);
   otherwise
       disp(‘Your enter is wrong’);
end
```

Switch Example (II) (Cont.)

```
>> cmd9
Type like, hate, or ok: like
I really like it
>> cmd9
Type like, hate, or ok: hate
I do not like it
>> cmd9
Type like, hate, or ok: abc
Your enter is wrong
>> cmd9
Type like, hate, or ok: Like
Your enter is wrong
```

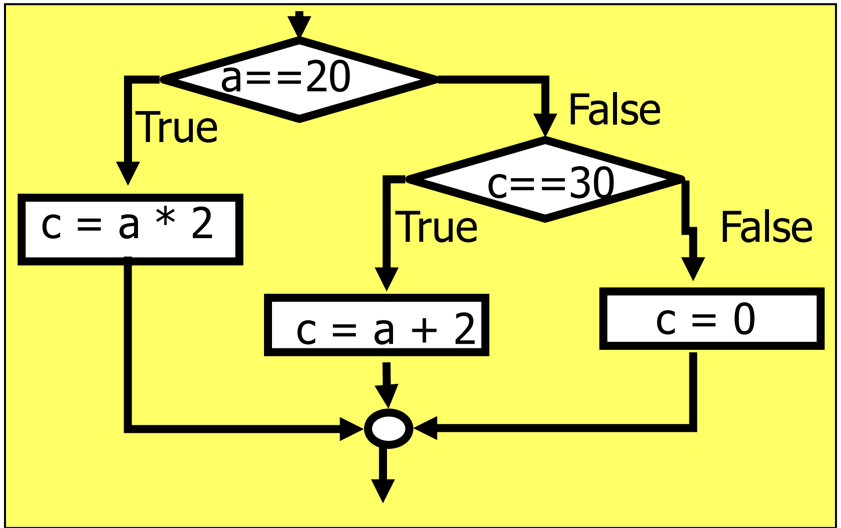
if-else and switch comparison

```
if logical exp1
   statement g1
elseif logical exp2
   statement g2
else
   statement g3
end
```

```
switch input expression
   case value1
       statement g1
   case value2
       statement g2
   otherwise
       statement g3
end
```

if-else and switch comparison (Cont.)

```
if a == 20
   c = a * 2
elseif a == 30
   c = a + 2
else
   c = 0
end
```



```
switch a
   case 20
      c = a * 2
   case 30
      c = a + 2
   otherwise
      c = 0
end
```

Try/Catch Construct

- Normally, a program is aborted when encountering an error.
- An error in the *try* block will lead to the execution of the code in the catch block
- This allows to handle errors without causing the program to stop.

Seen and used very often in JAVA

Try/Catch Construct (Cont.)

```
try
  statement group 1
  statement group 2
  ...
catch
  statement group 3
  statement group 4
  ...
end
```

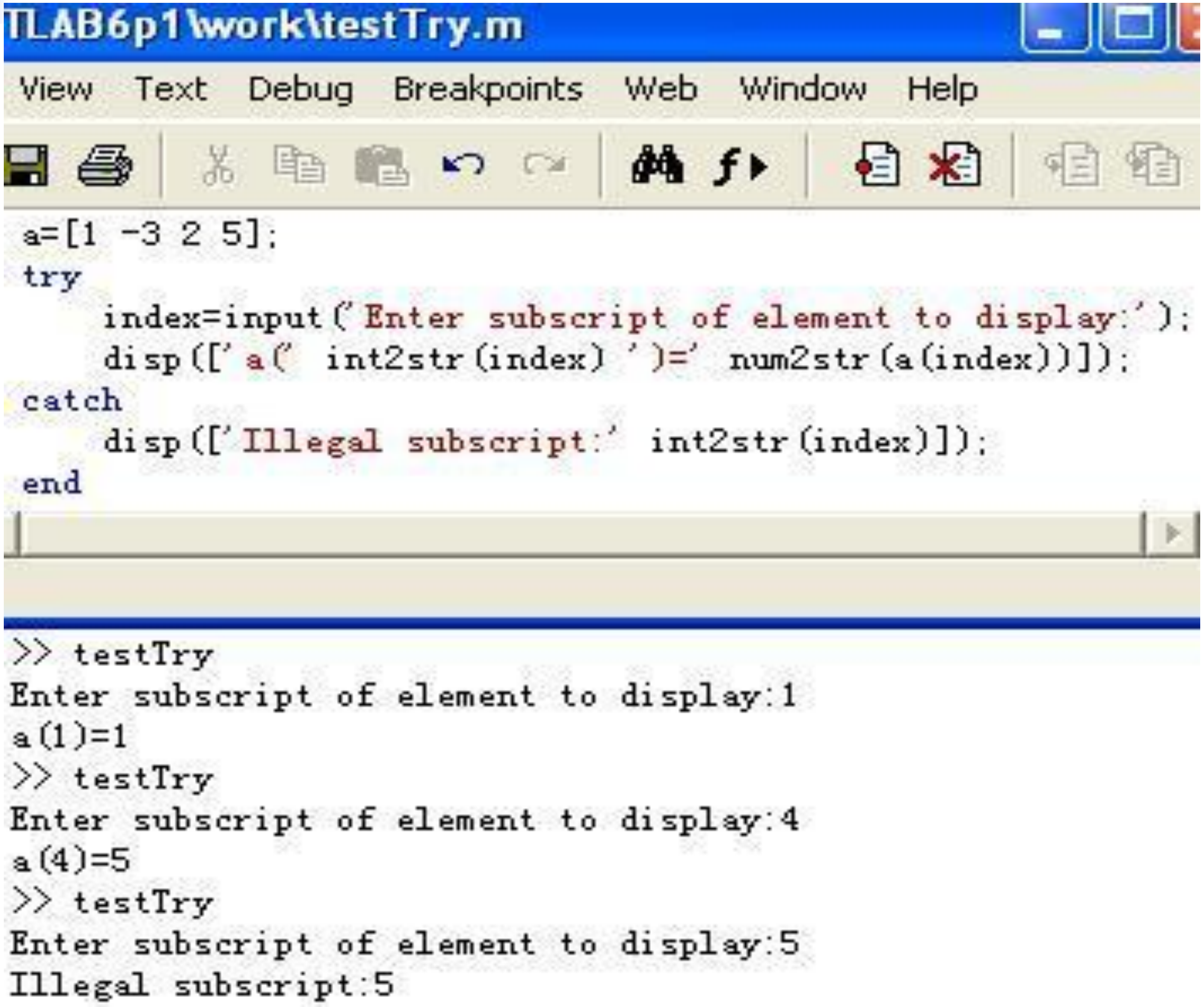
How is the situation in JAVA?

```
Try{
  statement group 1
  statement group 2
  ...
}catch{
  statement group 3
  statement group 4
  ...
}
```

Try/Catch Example

```
a=[1 -3 2 5];
try
    index=input('Enter subscript of element to display:');
    disp(['a(' int2str(index) ')=' num2str(a(index))]);
catch
    disp(['Illegal subscript:' int2str(index)]);
end
```

Try/Catch Example (Cont.)



```
a=[1 -3 2 5];
try
    index=input('Enter subscript of element to display:');
    disp(['a(' int2str(index) ')=' num2str(a(index))]);
catch
    disp(['Illegal subscript:' int2str(index)]);
end
```

```
>> testTry
Enter subscript of element to display:1
a(1)=1
>> testTry
Enter subscript of element to display:4
a(4)=5
>> testTry
Enter subscript of element to display:5
Illegal subscript:5
```

Additional Plotting Features

```
plot(x,y);
```

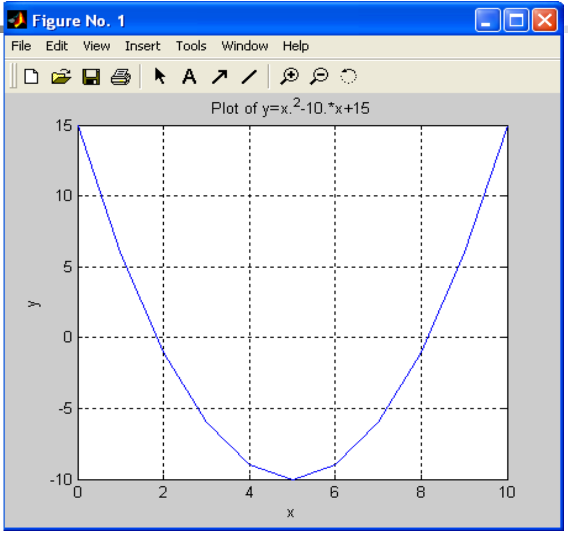
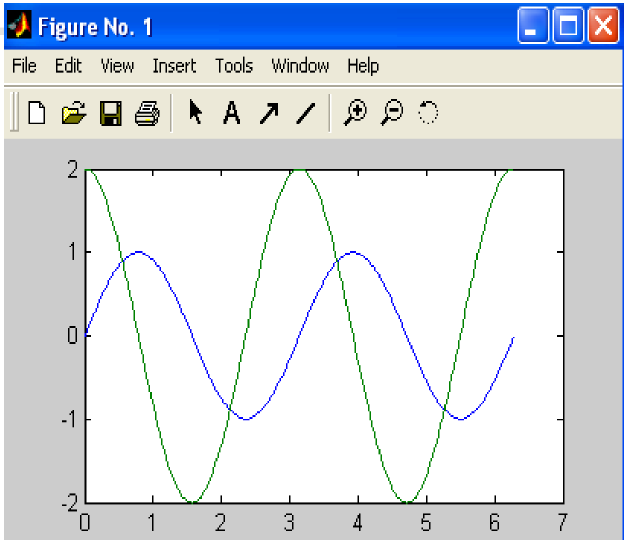
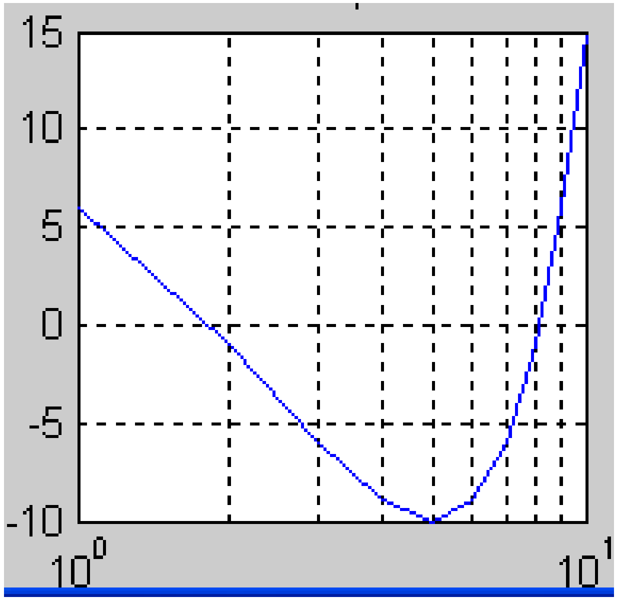


Figure No. 1
File Edit View Insert Tools Window Help
Plot of y=x.^{2}-10.*x+15
15
10
5
0
-5
-10
y
0
2
4
6
8
10
x

```
plot(x,y1,x,y2);
```



```
semilogx(x,y);
```



```
plot(x,y,'r--',x,y,'bo');

=plot(x,y,’r--’);

+  hold on;

+ plot(x,y,’bo’);
```

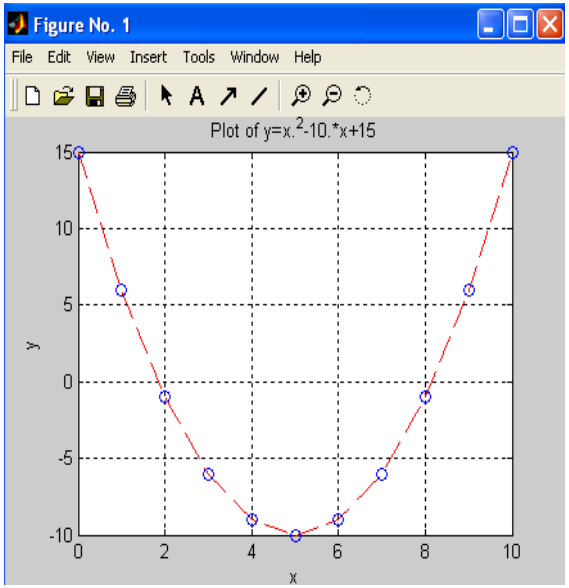
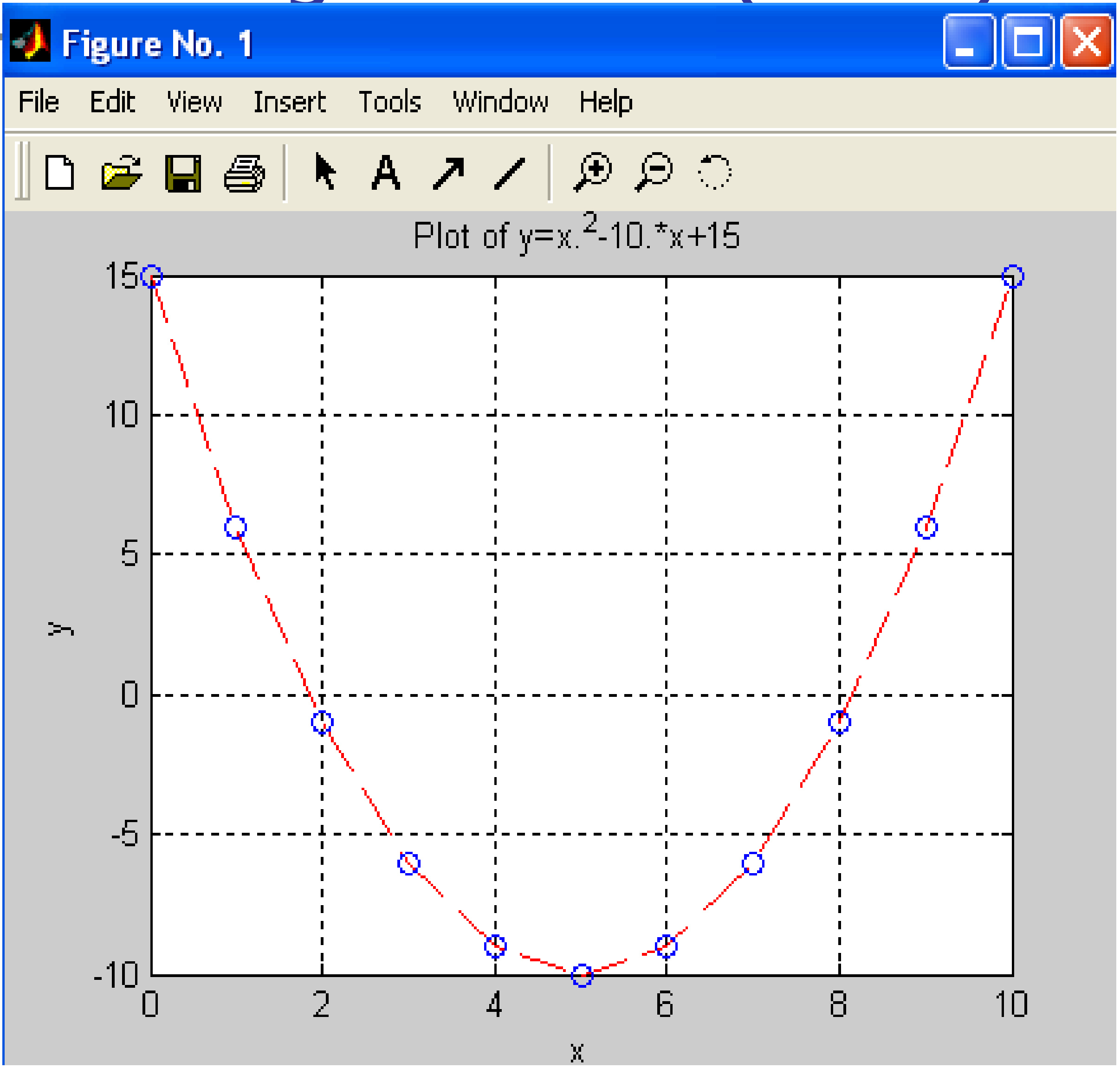


Figure No. 1
File Edit View Insert Tools Window Help
Plot of y=x.^{2}-10.*x+15
15
10
5
0
-5
-10
y
0
2
4
6
8
10
x

Additional Plotting Features (Cont.)

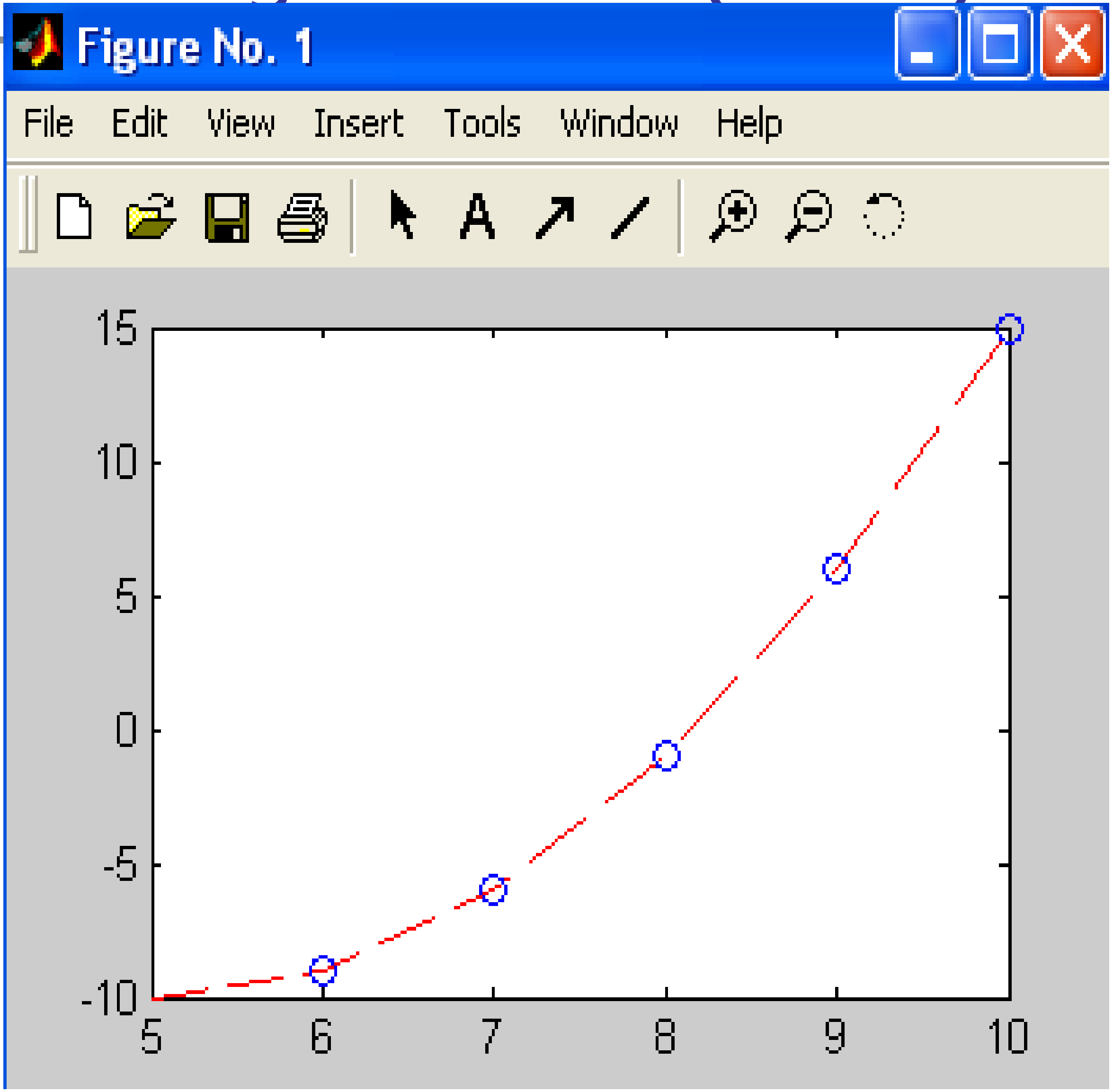
```
x=0:10;
y=x.^2-10.*x+15;
plot(x,y,'r--',x,y,'bo');
```



x: 0~10; y: -10~15

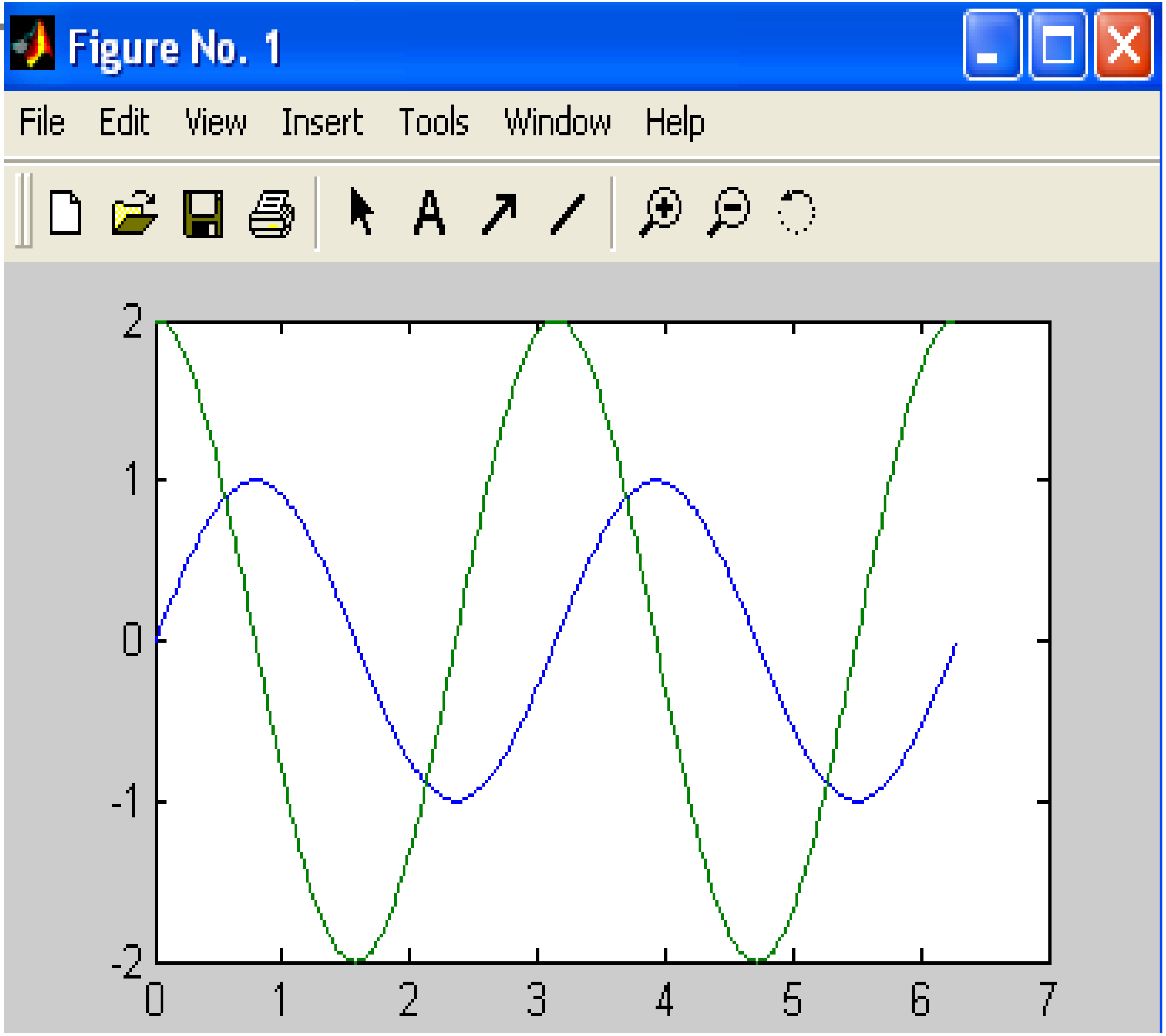
Additional Plotting Features (Cont.)

```
x=0:1:10;
y=x.^2-10.*x+15;
plot(x,y,'r--',x,y,'bo');
axis([5 10 -10 15]);
```



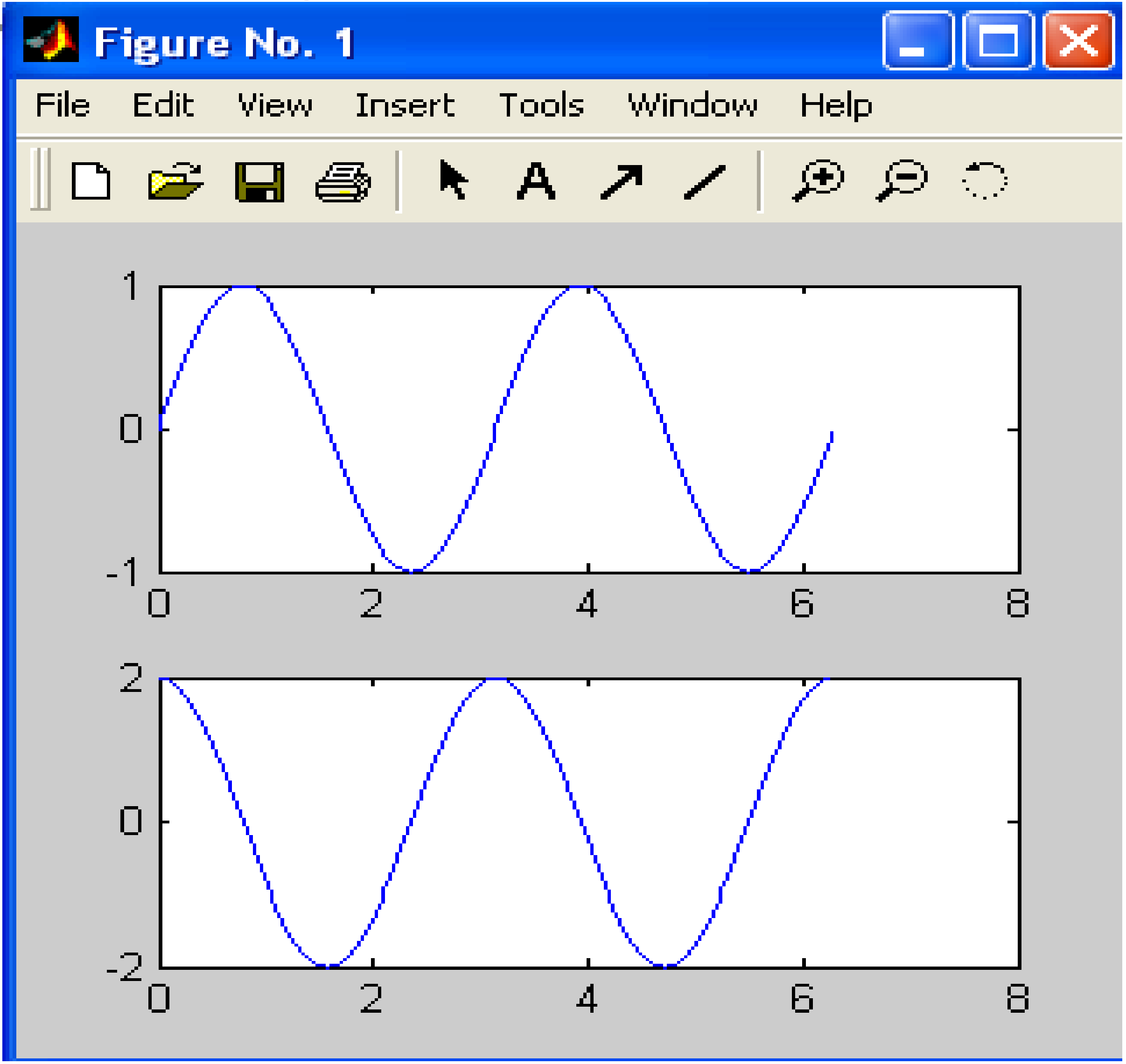
Additional Plotting Features (Cont.)

```
x=0:pi/100:2*pi;
y1=sin(2*x);
y2=2*cos(2*x);
plot(x,y1,x,y2);
```



Additional Plotting Features (Cont.)

```
x=0:pi/100:2*pi;
y1=sin(2*x);
y2=2*cos(2*x);
subplot(2,1,1);
plot(x,y1);
subplot(2,1,2);
plot(x,y2);
```



subplot(m,n,p) creates m*n subplots, with subplot p being the current figure index.

Additional Plotting Features (Cont.)

```
figure(1);
x=0:pi/100:2*pi;
y1=sin(2*x);
plot(x,y1);

figure(2);
y2=2*cos(2*x);
plot(x,y2);
```

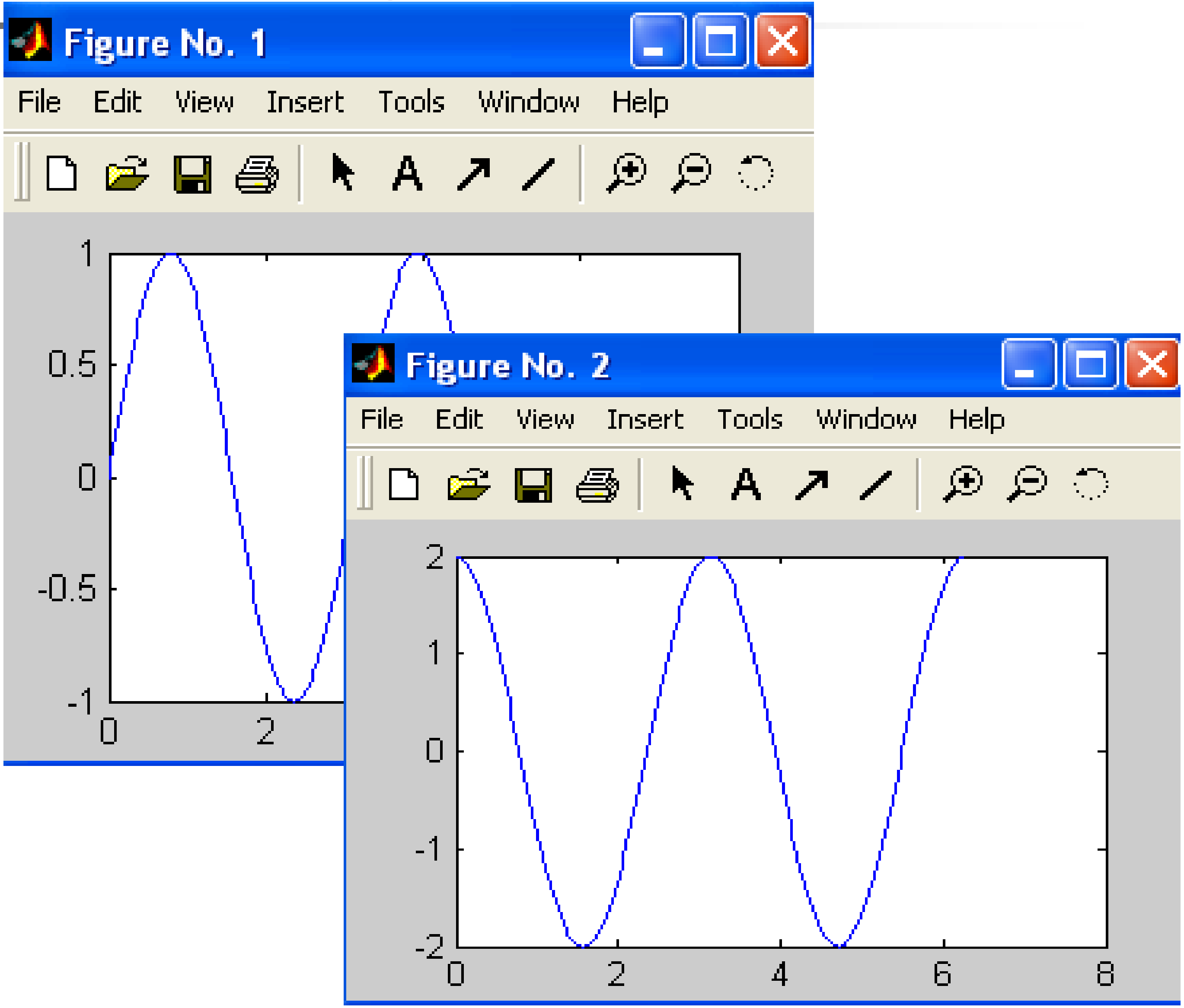
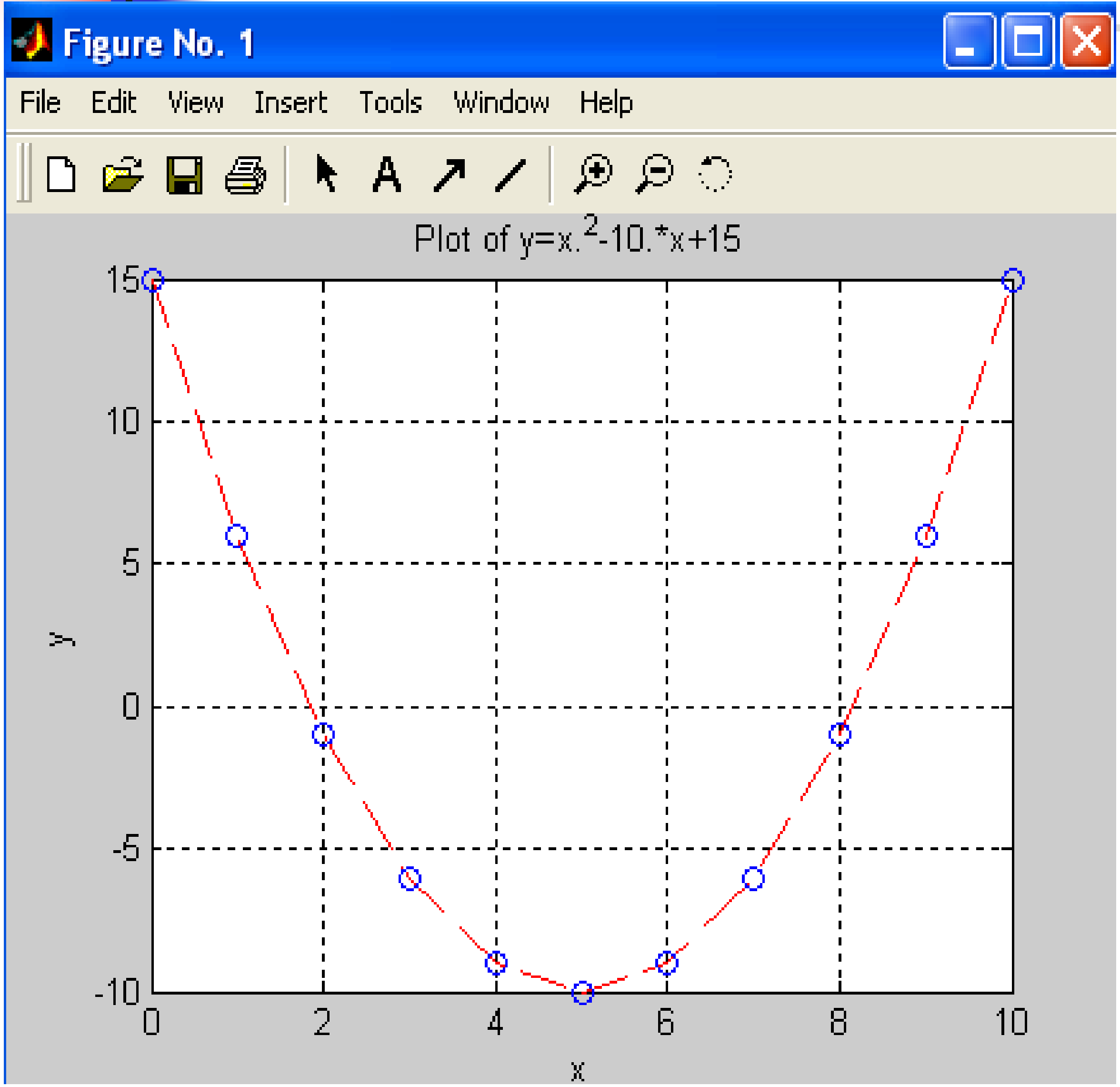


Figure No. 1
File Edit View Insert Tools Window Help
1
0.5
0
-0.5
-1
0
2
Figure No. 2
File Edit View Insert Tools Window Help
2
1
0
-1
-2
0
2
4
6
8

Additional Plotting Features (Cont.)



plot(x,y,'r--',x,y,'bo');

plot(x,y,’Properties’,...)

Additional Plotting Features (Cont.)

MarkerSize:

Specifies the size of the marker in points

MarkerEdgeColor:

Specifies the color of the marker or
the edge color for filled markers

MarkerFaceColor:

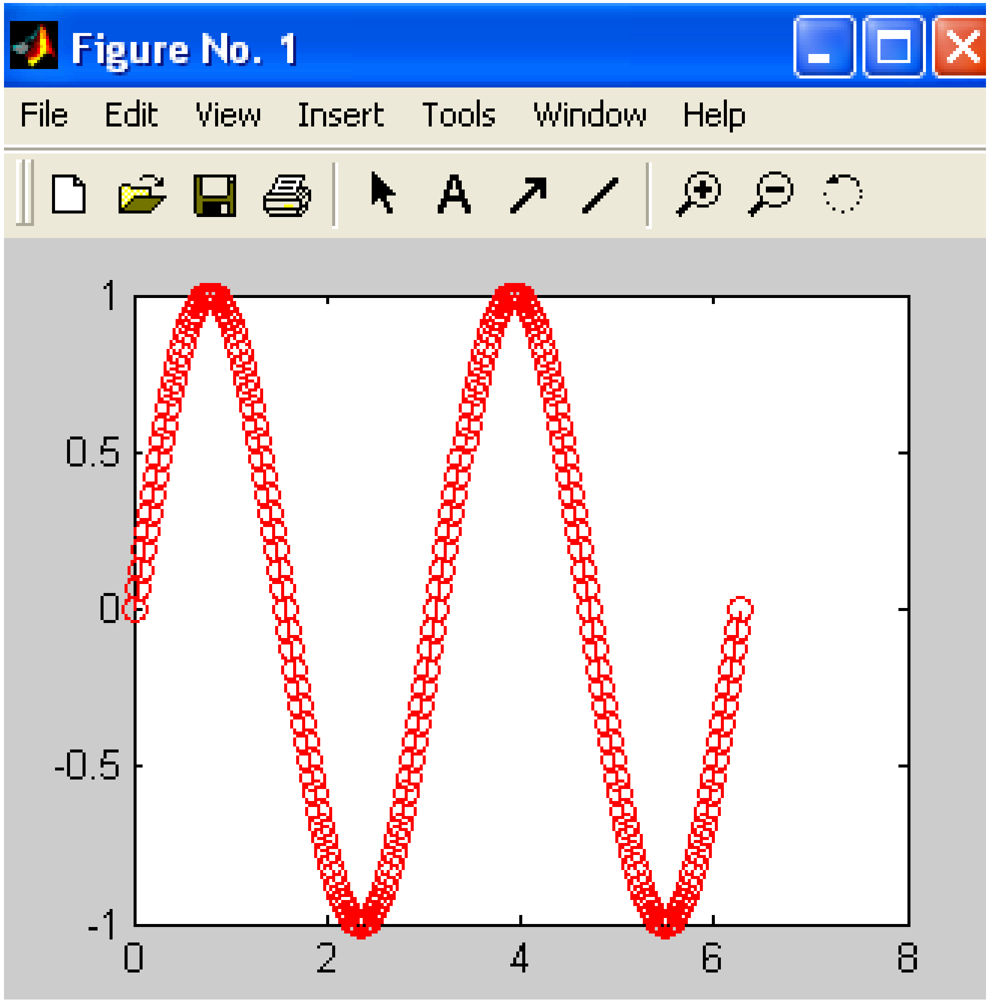
Specifies the color of the face of filled markers

LineWidth:

Specifies the width of each line in points

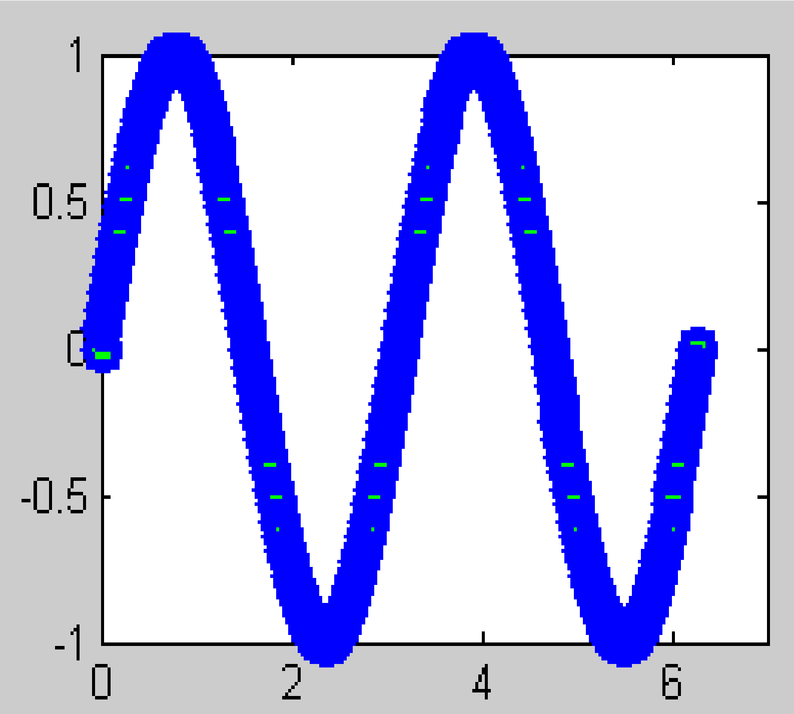
Additional Plotting Features (Cont.)

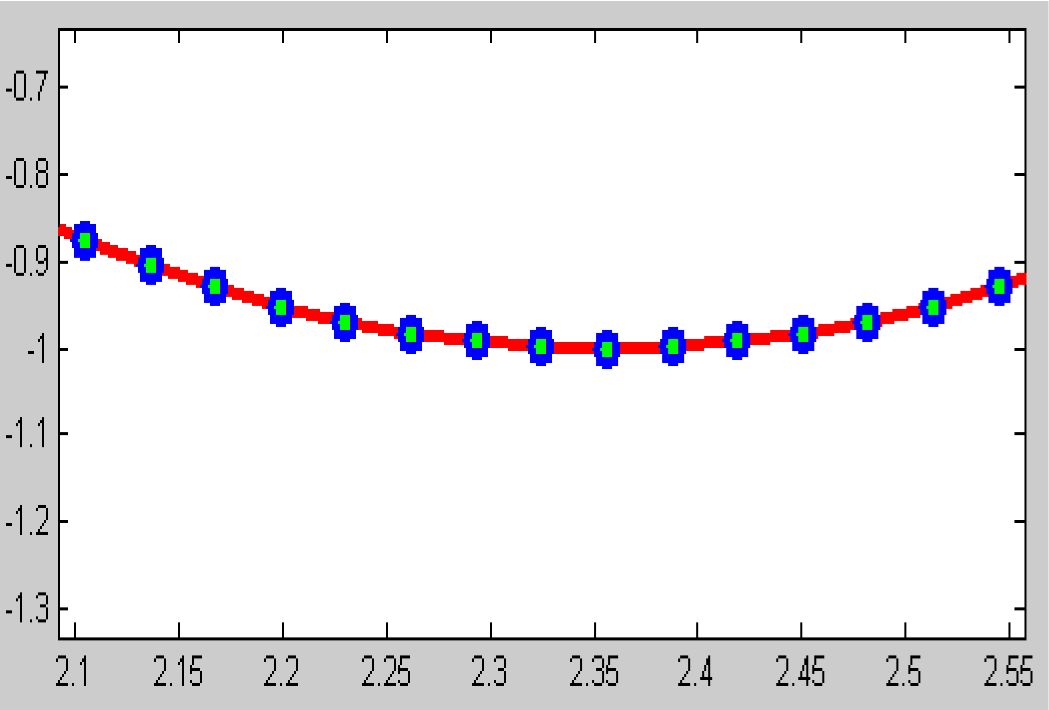
```
x=0:pi/100:2*pi;
y=sin(2*x);
plot(x,y,'-ro');
```

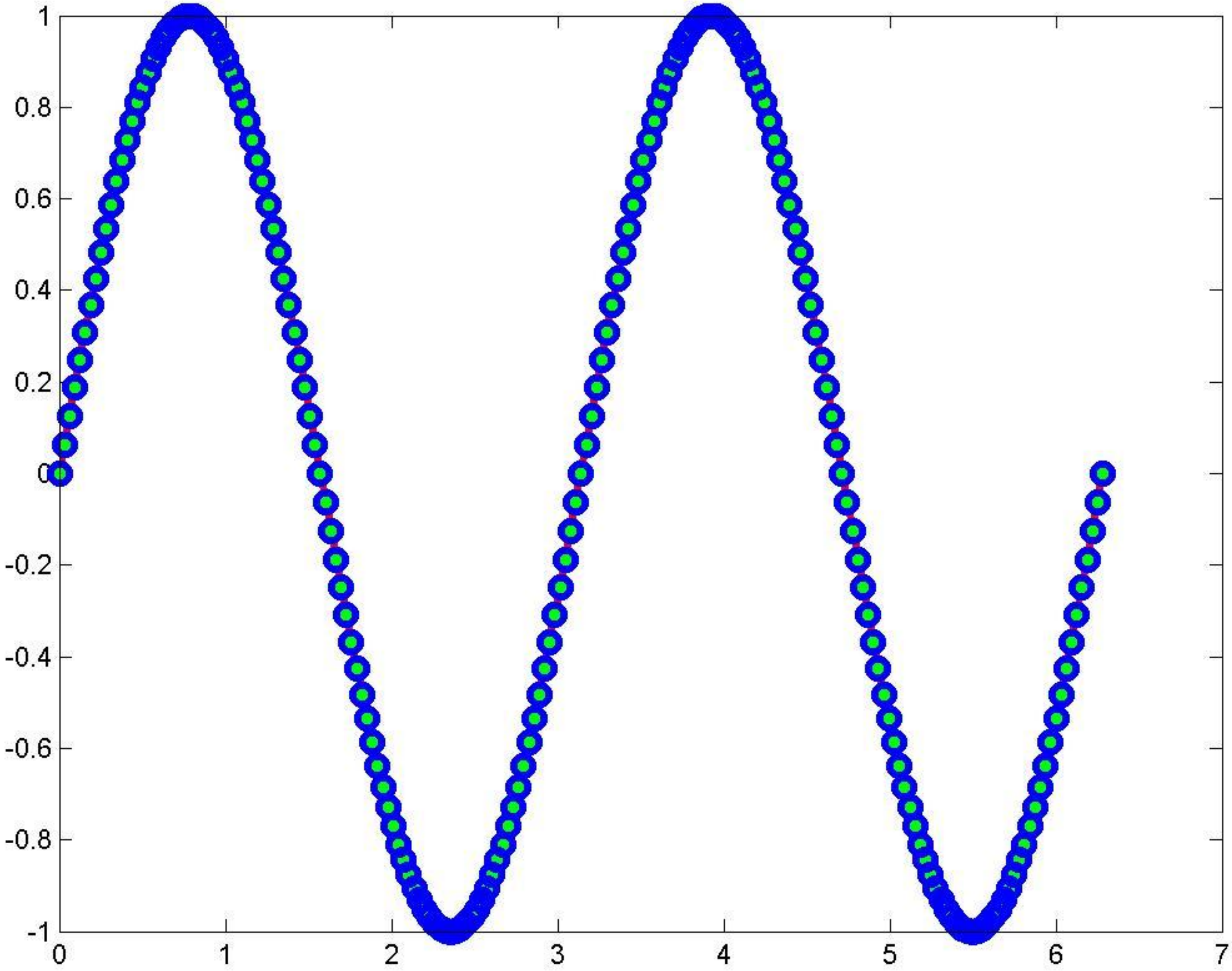


Additional Plotting Features (Cont.)

```
x=0:pi/100:2*pi;
y=sin(2*x);
plot(x,y,‘-ro','LineWidth',3.0,'MarkerSize',8,
'MarkerEdgeColor','b','MarkerFaceColor','g');
```



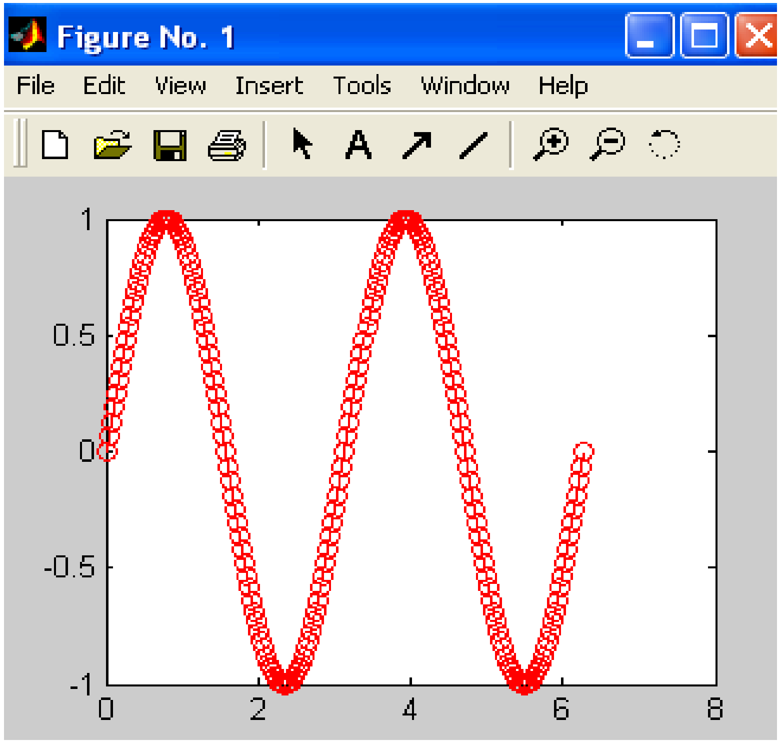


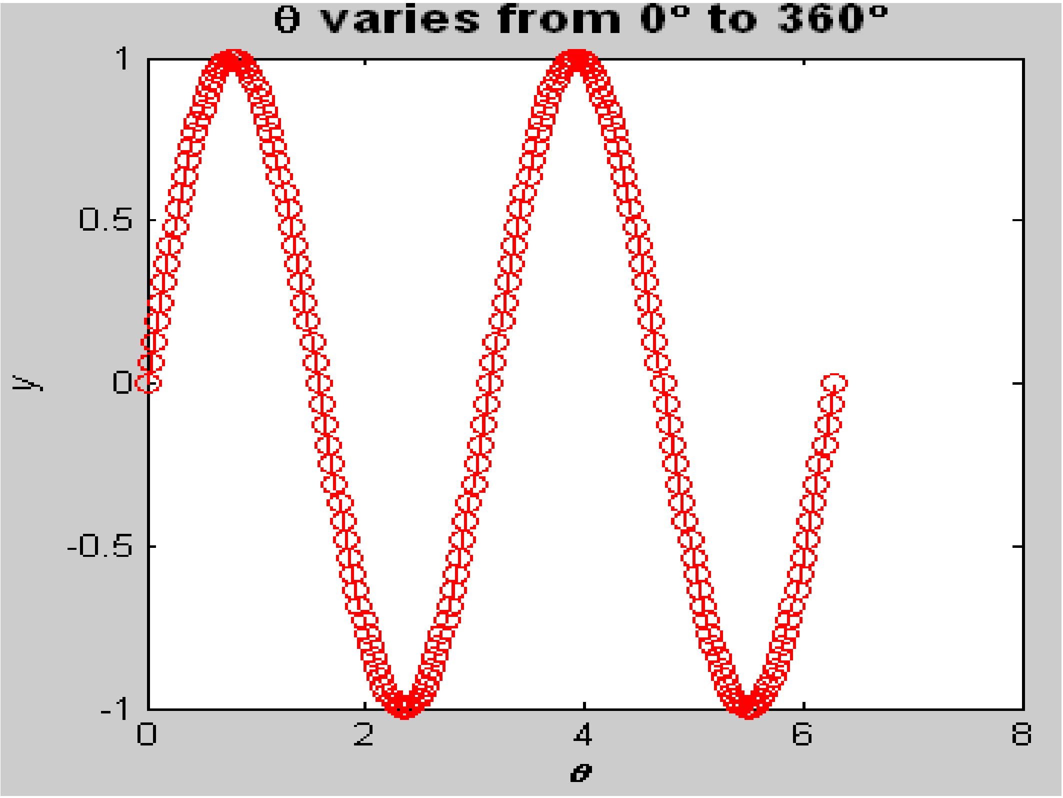

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7

Additional Plotting Features (Cont.)

- Enhanced Control of Text Strings

Text strings (titles, axis, labels, etc.) have the formats such as boldface, italics, or both, as well as special characters (Greek and mathematical symbols)





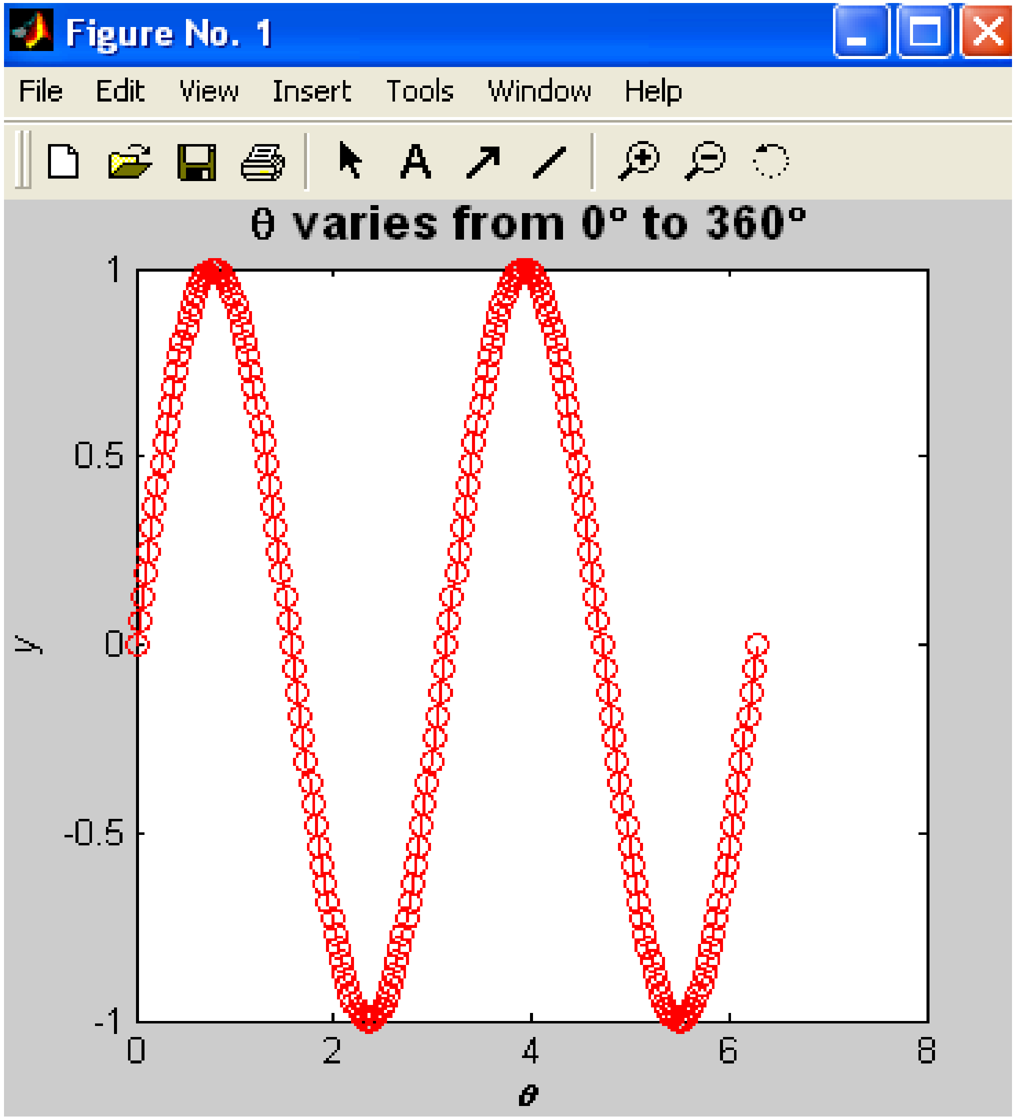
Additional Plotting Features (Cont.)

\bf	Boldface
\it	Italics
\fontname{fontname}	Specify the font name to use
\fontsize{fontsize}	Specify font size
_{***}	The characters inside the braces are subscripts
^{***}	The characters inside the braces are superscripts
\alpha	
\beta	
\gamma	
\tau	
\theta	

A backslash character "\" is used for printing special characters such as \, {, }, _, ^.

Additional Plotting Features (Cont.)

```
x=0:pi/100:2*pi;
y=sin(2*x);
plot(x,y,'-ro');
title('\bf\fontsize{14}\theta varies from 0\circ to 360\circ');
xlabel('\bf\it\theta');
ylabel('\ity');
```



How to generate \dot{\theta}?

Additional Plotting Features (Cont.)

- Polar Plots

expression:
```
polar(theta,r)
```

where *theta* is an array of angles in radians, and *r* is an array of distances.

Additional Plotting Features (Cont.)

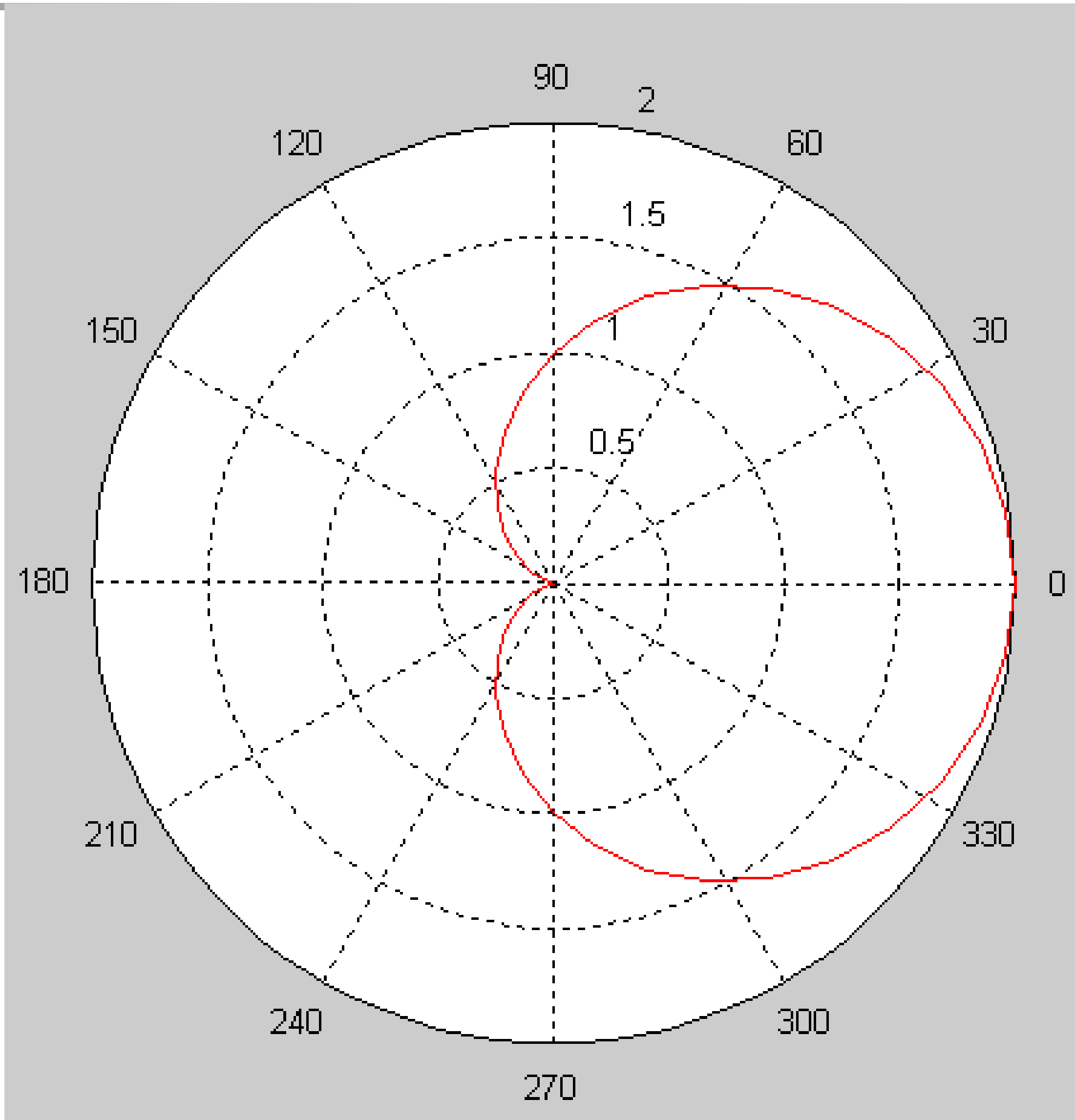
Microphone:

directional
enhancing the signals coming from the singer
suppressing the noise

gain=2g(1+cos(\theta)), with g=0.5

Additional Plotting Features (Cont.)

```
theta=0:pi/20:2*pi;
gain=2*0.5*(1+cos(theta));
polar(theta,gain,'r-');
```



Sincere Thanks!

- Using this group of PPTs, please read

- [1] Yunong Zhang, Weimu Ma, Xiao-Dong Li, Hong-Zhou Tan, Ke Chen, MATLAB Simulink modeling and simulation of LVI-based primal-dual neural network for solving linear and quadratic programs, Neurocomputing 72 (2009) 1679-1687

- [2] Yunong Zhang, Chenfu Yi, Weimu Ma, Simulation and verification of Zhang neural network for online time-varying matrix inversion, Simulation Modelling Practice and Theory 17 (2009) 1603-1617