Computer Application II

Lecture (02-04)

Lecturer. Mohamad Hassan

Polynomial Roots

- We can describe a polynomial in MATLAB with an array whose elements are the polynomial coefficients, *starting with the coefficient of the highest power of x*:
- For example, the polynomial $4x^3 - 8x^2 + 7x - 5$ would be represented by the array [4, -8, 7, -5].
- The roots of the polynomial *f(x)* are the values of x such that *f(x)=0*.
- Polynomial roots can be found with the *roots(a)* function, where *a* is the polynomial coefficient array.

Polynomial Roots

- For example, to find the roots of $x^3 - 7x^2 + 40x - 34$, the session is:

```
>> a=[1, -7, 40, -34];
>> roots(a)
ans =
3.0000 + 5.0000i
3.0000 - 5.0000i
1.0000
```

- The roots are x=1 and x=3+5i and x=3-5i.

Polynomial Roots

- The commands could have been combined into a single command as follows:

```
>> roots([1, -7, 40, -34])
ans =
3.0000 + 5.0000i
3.0000 - 5.0000i
1.0000
```

Polynomial Roots

- The *poly(r)* function computes the coefficients of the polynomial whose roots are specified by the array *r*:
- For example, to find the polynomial whose roots are 1, 3+5i and 3-5i, the session is:

>> r=[1, 3+5i, 3-5i];

>> poly (r)

ans =

1 -7 40 -34

- Thus the polynomial is $x^3 - 7x^2 + 40x - 34$.

Decision-making program in MATLAB

- The usefulness of MATLAB greatly increases with its ability to use decision making functions in its programs.
- These functions enable you to write programs whose operations depend on the results of calculations made by program.
- MATLAB has six relational operators to make comparison between arrays, these operators are shown in table

Decision-making program in MATLAB

Relational operator	Meaning
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
~=	Not equal to

Decision-making program in MATLAB

- The result of a comparison using the relational operators is either a 0 (if the comparison is false), or a 1 (if the comparison is true), and the result can be used as a variable.

Decision-making program in MATLAB

- For example if $x=2$ and $y=5$, typing $z=x<y$ returns the value $z=1$, because x is less than y. typing $u= x==y$ returns the value $u=0$ because x does not equal to y:

```
>> X=2;
>> Y=5;
>> Z=X<Y
Z =
1
>> U=X==Y
U =
0
```

Decision-making program in MATLAB

- The relational operators compare arrays on an element-by-element basis. The arrays must have the same dimension.
- The only exception occurs when we compare an array to a scalar. In that case, all the elements of the array are compared to the scalar.

Decision-making program in MATLAB

- For example, suppose that

x=[6, 3, 9] and y=[14, 2, 9],

then the following MATLAB session shows some examples

Decision-making program in MATLAB

```
>> x=[6,3,9]; y=[14,2,9];
>> z=(x<y)
z
= 1 0 0
>> z=(x>y)
z
= 0 1 0
```

Decision-making program in MATLAB

```
>> z=(x~=y)
z
= 1 1 0
>> z=(x==y)
z
= 0 0 1
>> z=(x>8)
z
= 0 0 1
```

Decision-making program in MATLAB

- We can also use the relational operators to address arrays. For example, with $x=[6, 3, 9, 11]$ and $y=[14, 2, 9, 13]$, typing $z=x(x<y)$ finds all the elements in x that are less than the corresponding elements in y The result is the array $z=[6, 11]$:
-

```
>> x=[6,3,9,11];
>> y=[14,2,9,13];
>> z=x(x<y)
z =
6    11
```

Decision-making program in MATLAB

- We can use the *find* function to create decision-making programs, especially when we combine it with the relational operators.
- The function *find (x)* computes an array containing the indices of the nonzero elements of the array *x*. Consider the session:

```
>> x=[-2,0,4];
>> y=find(x)
y =
1    3
```

Decision-making program in MATLAB

- The result indicates that the first and third elements of x are nonzero.
- The find function returns the indices, not the values.

Decision-making program in MATLAB

- In the following session, note the difference between the result obtained by *x(x<y)* and the result obtained by *find (x<y)*:

```
>> x=[6,3,9,11];
>> y=[14,2,9,13];
>> values=x(x<y)
values =6    11
>> how_many=length(values)
how_many =
2
>> indices=find(x<y)
indices =
1    4
```

Decision-making program in MATLAB

- Thus two values in the array x are less that the corresponding values in the array y.
- They are the first and fourth values, 6 and 11.
- To find out how many, we could also have typed length (indices).

Plotting with MATLAB

- MATLAB contains many powerful functions for easily creating plots of several different types, such as rectilinear, logarithmic, surface, and contour plots.
- The plot appears on the screen is the graphic window, named Figure No. 1.
- The xlabel function places the text in single quotes as label on the horizontal

Plotting with MATLAB

- The ylabel function performs a similar function for the vertical axis.
- Other useful plotting functions are title and gtext.
- These functions place text on the plot.
- Both accept text within parentheses and single quotes, as with the xlabel function.

Plotting with MATLAB

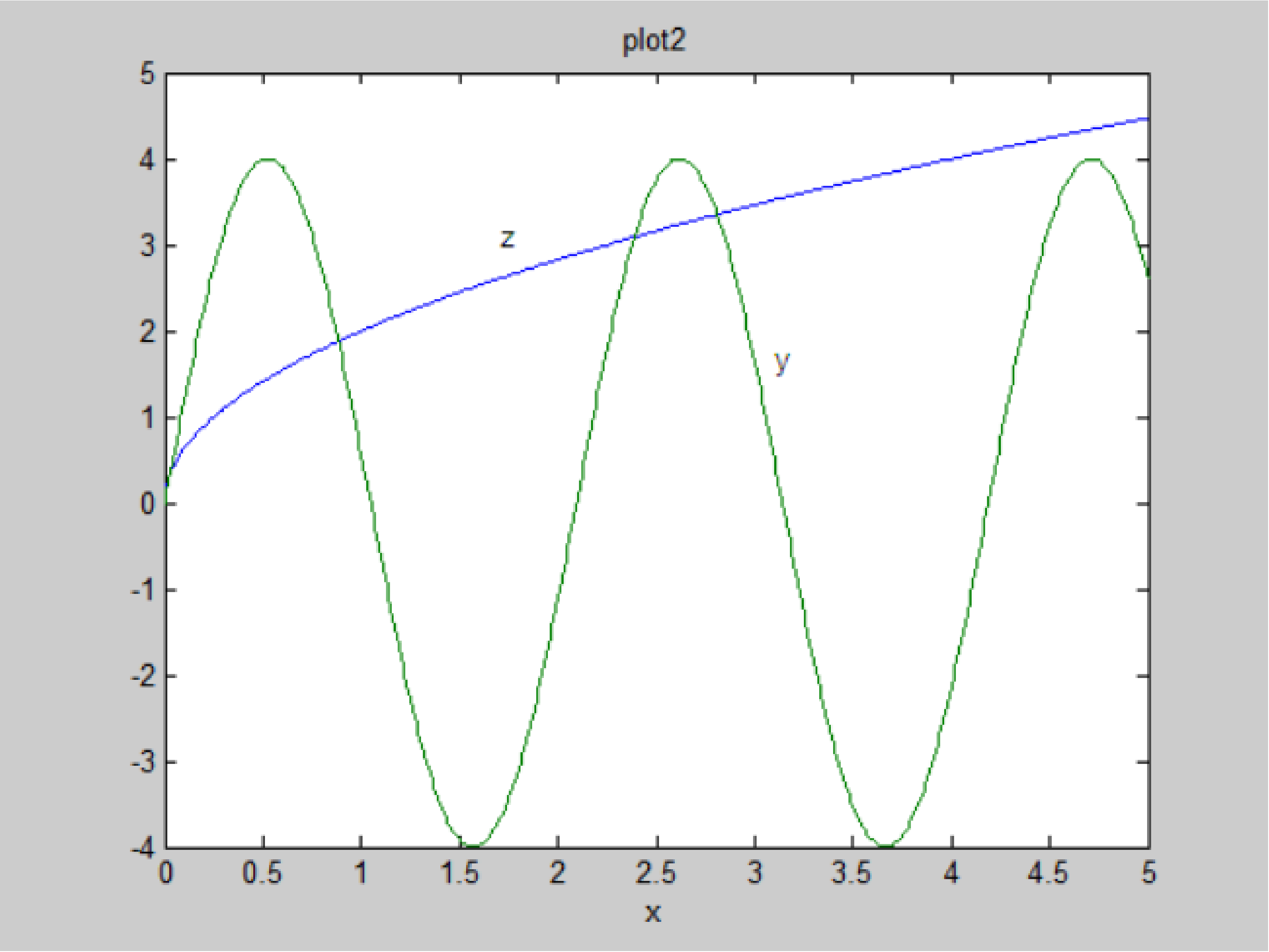
- The title function places the text at the top of the plot; the gtext function places the text at the point on the plot where the cursor is located when you click the left mouse button.
- You can create multiple plots–called overlay plots–by including another set or sets of values in the plot function.

Plotting with MATLAB

- For example to plot the functions $y = 2\sqrt{x}$ and $z = 4\sin 3x$ for $0 \le x \le 5$ on the same plot, the session is:

```
>> x=[0:0.01:5];
>> y=2*sqrt(x);
>> z=4*sin(3*x);
>> plot (x,y,x,z), xlabel('x'), gtext('y'), gtext('z')
```

Plotting with MATLAB



Plotting with MATLAB

- The plotting functions xlabel, ylabel, title and gtext must be placed after the plot function and separated by commas.
- You can also distinguish curves from one another by using different line types for each curve.
- For example to plot z curve using dashed, replace the plot (x, y, x, z) function in above session with plot (x, y,

Plotting with MATLAB

- Sometimes it is useful or necessary to obtain the coordinates of a point on a plot curve. The function ginput can be used for this purpose.
- Place it at the end of all the plot and plot formatting statements, so that the plot will be in its final form. The command is [x,y]=ginput (n) gets n points and returns the x and y coordinates in the vectors x and y, which have the length n

Plotting with MATLAB

- In cases where you are plotting data, as opposed to functions, you should use data markers to plot each data point.
- To mark each point with a plus sign +, the required syntax for the plot function is plot(x, y, ‘+’).

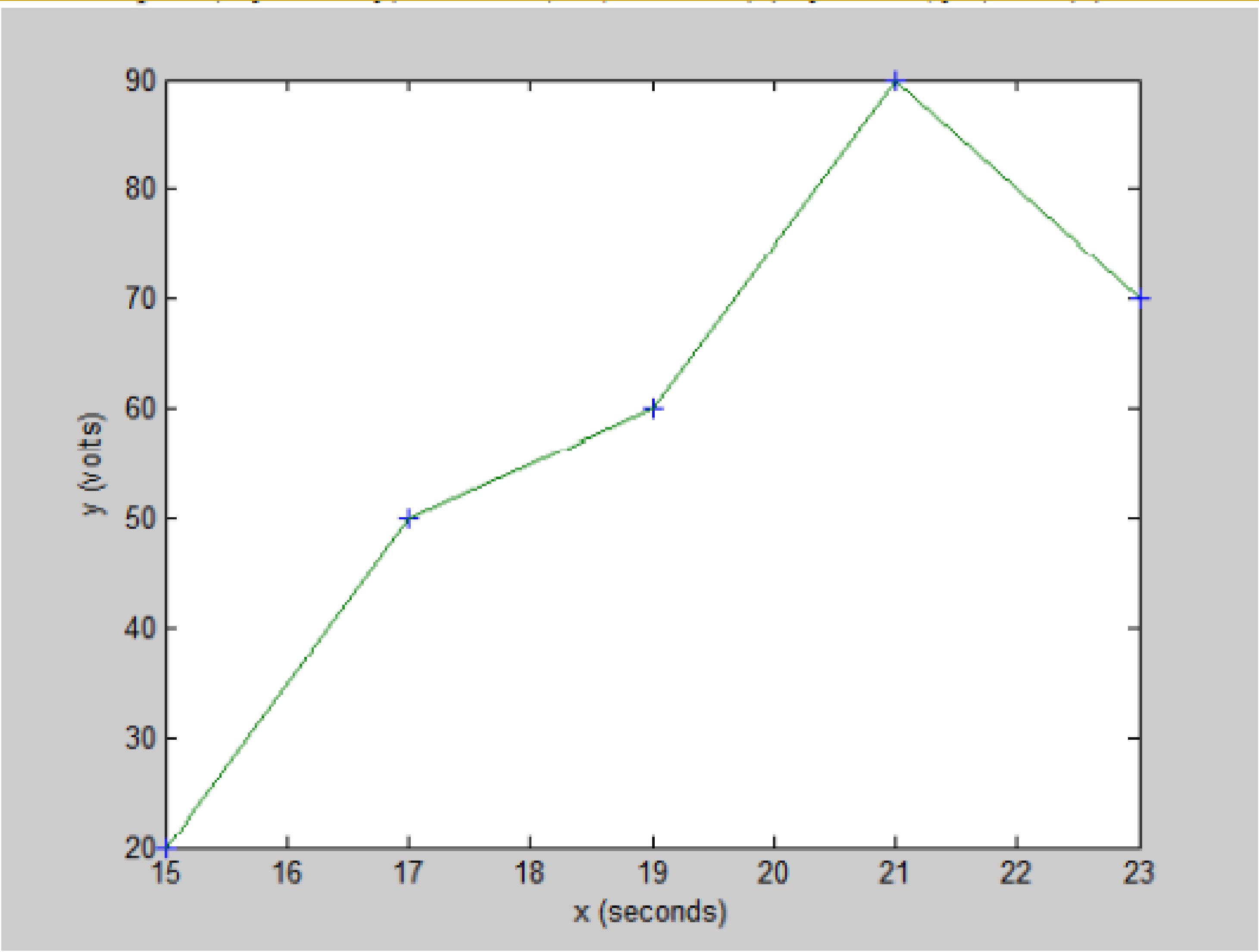
Plotting with MATLAB

- For example, suppose the data for the independent variable is x=[15: 2: 23] with units of seconds, and the dependent variable values are y=[20, 50, 60, 90, 70] with units of volts. To plot the y data with plus sign use the following session:

Plotting with MATLAB

```
>> x=[15:2:23];
>> y=[20, 50, 60, 90, 70];
>> plot(x,y,'+',x,y), xlabel('x (seconds)'),
   ylabel('y (volts)')
```

Plotting with MATLAB



Plotting polynomials

- The polyval(a,x) function evaluates a polynomial at specified values of its independent variable x.
- The polynomial coefficient array is a.

Plotting polynomials

- For example, to evaluate the polynomial $f(x) = 9x^3 - 5x^2 + 3x + 7$ at the points $x = 0, 2, 4, \ldots, 10$, the session is:
- The resulting array f contains 6 values that correspond to *f(0), f(2), f(4), ..., f(10):*

```
>> a=[9,-5,3,7];
>> x=[0:2:10];
>> f=polyval (a,x);
>> f

f =

7     65     515     1789     4319     8537
```

Plotting polynomials

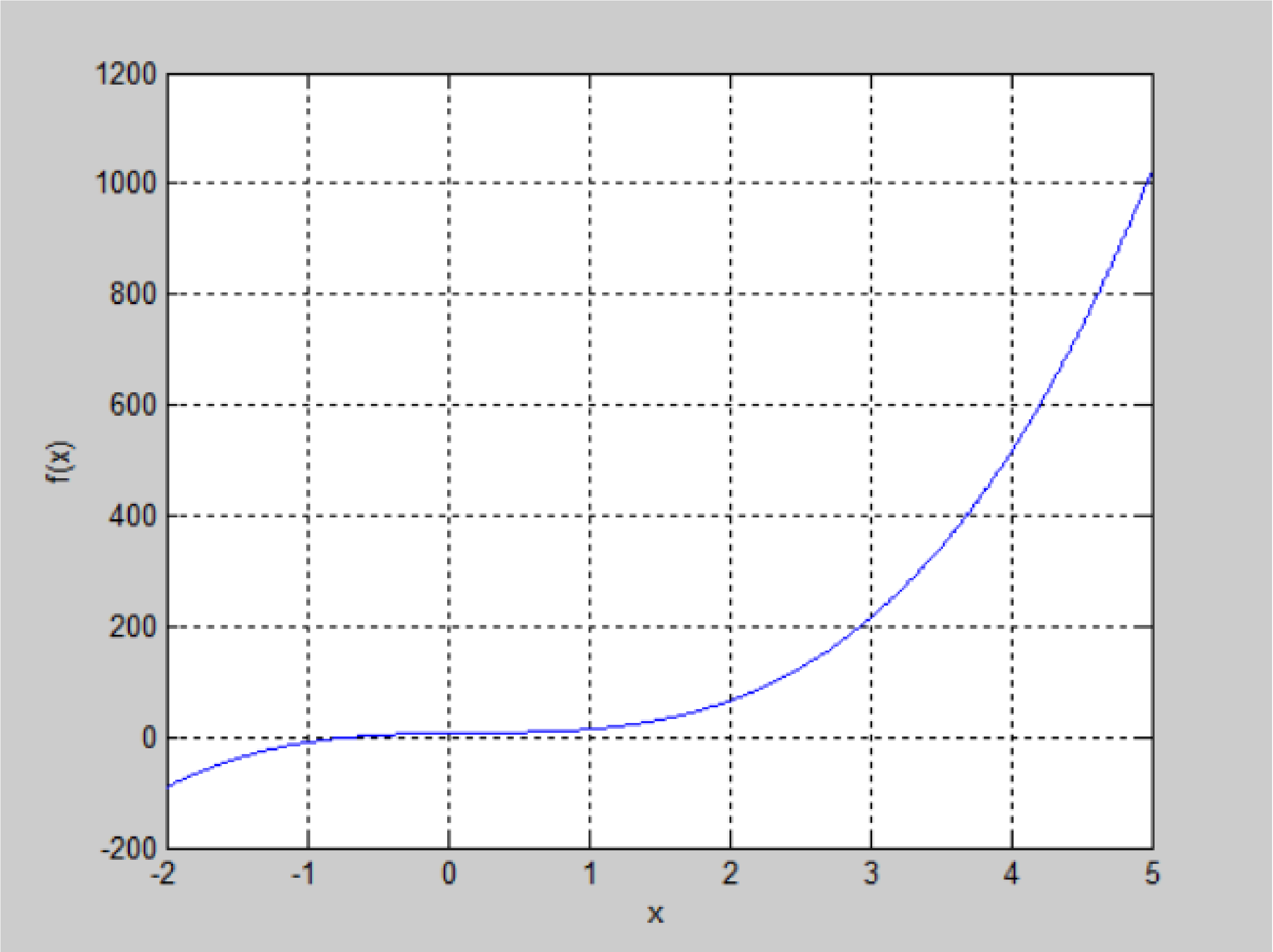
- The polyval function is very useful for plotting polynomials.
- To do this you should define an array that contains many values of the independent variable x in order to obtain a smooth plot.

Plotting polynomials

- For example, to plot the polynomial $f(x) = 9x^3 - 5x^2 + 3x + 7$ for $-2 \le x \le 5$ the session is:
- The *grid* command puts grid lines on the plot.

```
>> a=[9,-5,3,7];
>> x=[-2:0.01:5];
>> f=polyval (a,x);
>> plot(x,f),xlabel('x'),ylabel('f(x)'),grid
```

Plotting polynomials



Plotting with MATLAB

Commands	Description
[x,y] = ginput (n)	Enables the mouse to get n points from a plot, and returns the x and y coordinates in the vectors x and y, which have a length n.
grid	Puts grid lines on the plot.
gtext('text')	Enables placement of text with the mouse.
plot(x,y)	Generates a plot of the array y versus the array x on rectilinear axes.
polyval(a,x)	Evaluates a polynomial at specified values of its independent variable x. the polynomial coefficients of descending powers are stored in the array a. The result is the same size as x.
title('text')	Puts text in a title at the top of the plot.
xlabel('text')	Adds a text label to the horizontal axis .
ylabel('text')	Adds a text label to the vertical axis .

Plotting with MATLAB

A batch distillation process

- Chemical and environmental engineers must sometimes design batch processes for producing or purifying liquids and gases. Applications of such process occur in food and medicine production, and in waste processing and water purification. An example of such process is a system for heating a liquid benzene/toluene solution to distill a pure benzene vapor. A particular batch distillation unit is charged initially with 100 mol of a 60 percent mol benzene /40 percent mol toluene mixture. Let L (mol) be the amount of liquid remaining in the still, and let X (mol B/mol) be the benzene mole fraction in the remaining liquid. Conservation of mass for benzene and toluene can be applied to derive the following relation

Plotting with MATLAB

$$L = 100\left(\frac{X}{0.6}\right)^{0.625}\left(\frac{1-X}{0.4}\right)^{-1.625}$$

Determine what mole fraction of benzene remains when L= 70. Note that it is difficult to solve this equation directly for X. Use a plot of X versus L to solve the problem.

Plotting with MATLAB

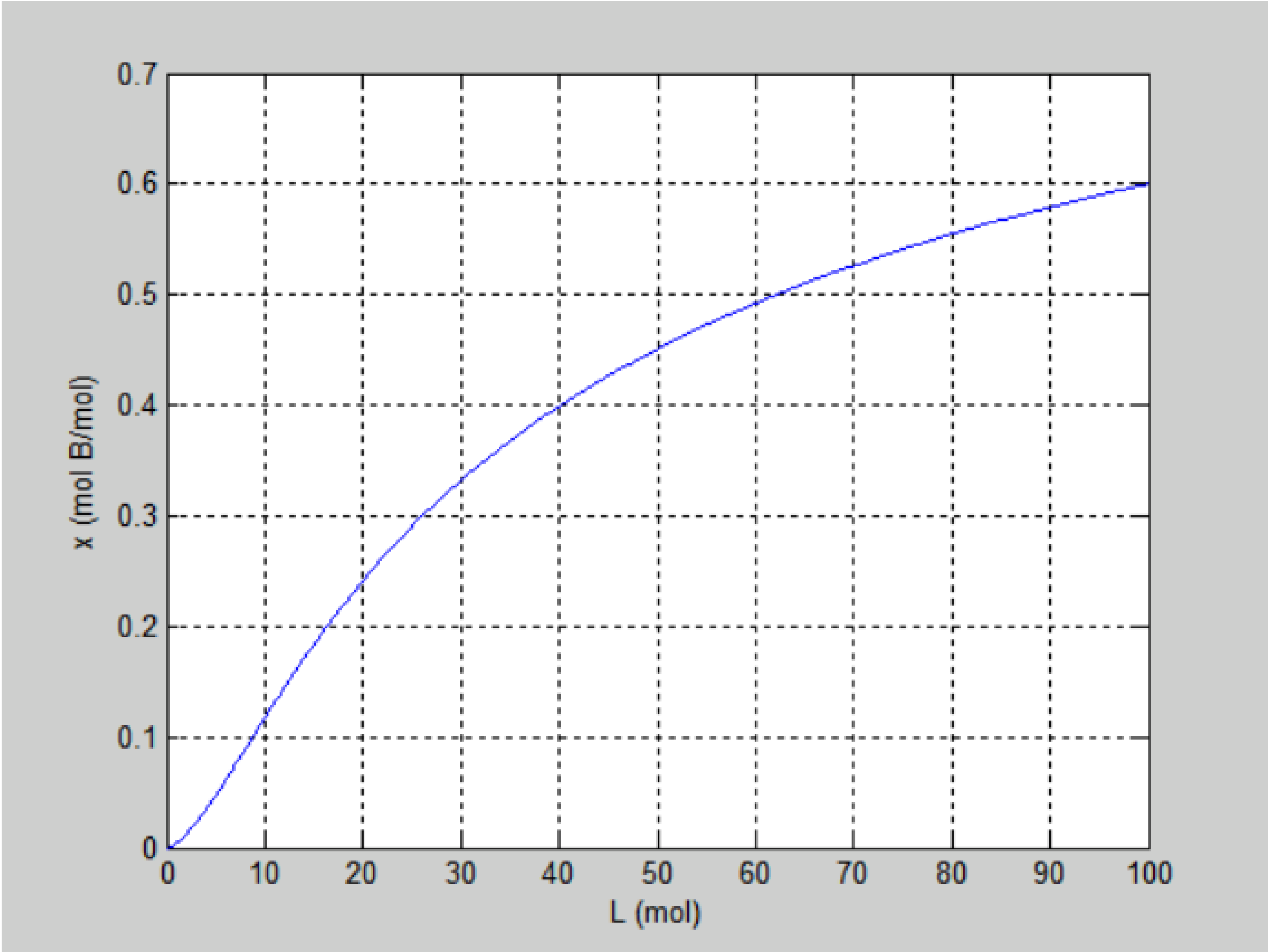
Solution

- The equation involves both array multiplication and array exponentiation. Note that MATLAB enables us to use decimal exponents to evaluate L. It is clear that L must be in the range $0 \le L \le 100$, however, we do not know the range of X, except that $X \ge 0$. Therefore, we must make a few guesses for the range of X, using a session like the following. We find that L>100 if X>0.6, so we choose X=[0:0.001:0.6]. We use the ginput function to find the value of X corresponding to L=70.

Plotting with MATLAB

```
>> x=[0:.001:0.6];
>> L=100*(x/0.6).^(0.625).*((1-x)/0.4).
   ^(-1.625);
>> plot(L,x),grid,xlabel('L (mol)'),ylabel('x
   (mol B/mol)'),...
[L,x]=ginput(1)
L = 69.9309
x = 0.5250
```

Plotting with MATLAB



Linear Algebra equations

- We can use the left division operator (\) in MATLAB to solve sets of linear algebraic equations.
- To solve such set in MATLAB you must create two arrays; A and B.
- The array A has as many rows as there are equations, and as many columns as there are variables. The rows in A must contain the coefficients of x, y, z in that order.
- The array B contains the constants on the right-hand side of the equation; it has one column and as many rows as there are equations.

Linear Algebra equations

Use the left division method to solve the following set

3x-2y-9z=-65

-9x-5y+2z=16

6x+7y+3z=5

Solution

The matrix A is

$$A = \begin{bmatrix} 3 & 2 & -9 \\ -9 & -5 & 2 \\ 6 & 7 & 3 \end{bmatrix}$$

Linear Algebra equations

- We can use MATLAB to check the determinant of A to see whether the problem is singular, the session looks like this:

```
>> det(A)
ans =
288
```

- Because A is not equal to zero, a unique solution exist, it is obtained as follows:

Linear Algebra equations

```
>> A=[3 2 -9; -9 -5 2; 6 7 3];
>> B=[-65; 16; 5];
>> A\B
ans =
2.0000
-4.0000
7.0000
```

Linear Algebra equations

- This answer gives the vector X, which corresponds to the solution x=2, y=-4, z=7. It can be checked by determining whether AX gives the vector B by, typing

```
>> A*ans
ans =
-65.0000
16.0000
5.0000
```

- Which is a vector B. thus the answer is correct.