Chapter 11

Symbolic Math

MATLAB An Introduction With Applications, 6th Edition
Dr. Amos Gilat
The Ohio State University

Slide deck by
Dr. Greg Reese
Miami University

In all previous chapters, MATLAB always used only numbers. The inputs, outputs, results, etc. were numbers. Sometimes it's better to have more general solutions. One type of general solution is to use a symbol that represents any number

Symbolic math

- Can give concise statements of problem solutions
- Provides exact solutions with no numerical errors
- Provides exact solutions when numerical solutions aren't possible or feasible
- Can lead to more understanding of a problem and its solution

Consider the quadratic equation $ax^2 + bx + c = 0$ and its solution

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}, \qquad a \neq 0$$

- Concise – one equation gives us all solutions for the infinite number of values of *a*, *b*, and *c*
- Exact solution - for *a*=9, *b*=0, *c*=-1, exact solution is *x*=±1/3, numerical solution is *x*=0.333333333333333

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}, \qquad a \neq 0$$

- Good even if numerical solutions won't work
  - b=10^{200}, a = 1, c = 1 fails because b^2 out of range
- Symbolic solution shows that the equation always has exactly two roots and that b^2 must be at least $4ac$ for the roots to be real
  - These are harder to learn from purely numerical solutions

Symbolic Toolbox in MATLAB performs symbolic math

- Student version of MATLAB comes with Symbolic Toolbox
- Full version of MATLAB doesn't have Symbolic Toolbox (must buy it)

A *symbolic object* is made of variables and numbers that MATLAB evaluates symbolically, not numerically

- Can be variables or numbers

A *symbolic expression* is a mathematical expression with at least one symbolic object

- Even if a symbolic expression looks like a numeric expression, MATLAB can tell the difference and evaluates it symbolically

A symbolic object can be a variable or a number. Make a symbolic object with the command

```
object_name = sym('string')
```

`'string'` is the name of the symbolic object and can be

- One or more letters (no spaces)
- A combination of letters and digits that starts with a letter
- A number

```
object_name = sym('string')
```

It is common (though not required) to make `object_name` be the same as `'string'`

```
>> a=sym('a')
a =
a
>> bb=sym('bb')
bb =
bb
>> x=sym('x');
>>
```

Create a symbolic object a and assign it to a.

The display of a symbolic object is not indented.

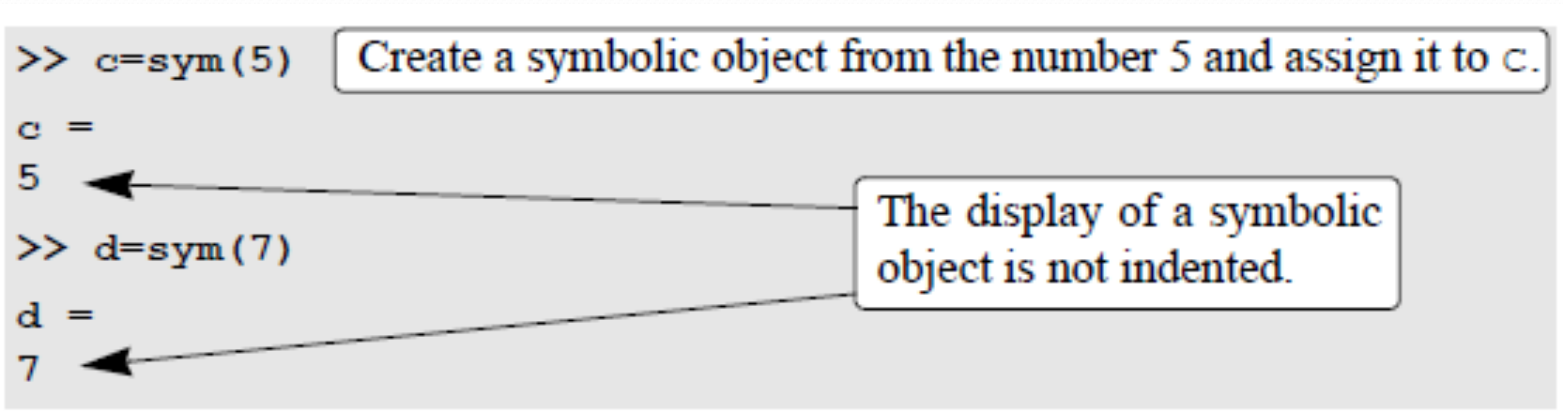
The symbolic variable x is created but not displayed, since a semicolon is typed at the end of the command.

The name of the symbolic object can be different from the name of the variable. For example:

```
>> g=sym('gamma')
g =
gamma
```

The symbolic object is gamma, and the name of the object is g.

When defining a symbolic number, don't need to put the object in quotes



```
syms variable_name1 variable_name2 variable_name3
```

- Creates multiple symbolic objects simultaneously
- Objects have same name as variable names

```
>> syms y z d
>> y
y =
y
```

The variables created by the `syms` command are not displayed automatically. Typing the name of the variable shows that the variable was created,

To create a symbolic expression use

`expression_name = mathematical expression`

A few examples are:

```
>> syms a b c x y
>> f=a*x^2+b*x + c
f =
a*x^2 + b*x + c
```

Define a, b, c, x, and y as symbolic variables.

Create the symbolic expression $ax^2 + bx + c$ and assign it to f.

The display of the symbolic expression is not indented.

When a symbolic expression, which includes mathematical operations that can be executed (addition, subtraction, multiplication, and division), is entered, MATLAB executes the operations as the expression is created. For example:

```
>> g=2*a/3+4*a/7-6.5*x+x/3+4*5/3-1.5
```

$\frac{2a}{3} + \frac{4a}{7} - 6.5x + \frac{x}{3} + 4 \cdot \frac{5}{3} - 1.5$ is entered.

```
g =
(26*a)/21 - (37*x)/6 + 31/6
```

$\frac{26a}{21} - \frac{37x}{6} + \frac{31}{6}$ is displayed.

Notice that all the calculations are carried out exactly, with no numerical approximation. In the last example, $\frac{2a}{3}$ and $\frac{4a}{7}$ were added by MATLAB to give $\frac{26a}{21}$, and $-6.5x + \frac{x}{3}$ was added to $\frac{37x}{6}$. The operations with the terms that contain only numbers in the symbolic expression are carried out exactly. In the last example, $4 \cdot \frac{5}{3} + 1.5$ is replaced by $\frac{31}{6}$.

The difference between exact and approximate calculations is demonstrated in the following example, where the same mathematical operations are carried out—once with symbolic variables and once with numerical variables.

```
>> a=sym(3); b=sym(5);
>> e=b/a+sqrt(2)
e =
2^(1/2) + 5/3
>> c=3; d=5;
>> f=d/c+sqrt(2)
f =
    3.0809
```

Define a and b as symbolic 3 and 5, respectively.

Create an expression that includes a and b.

An exact value of e is displayed as a symbolic object (the display is not indented).

Define c and d as numerical 3 and 5, respectively.

Create an expression that includes c and d.

An approximated value of f is displayed as a number (the display is indented).

An expression that is created can include both symbolic objects and numerical variables. However, if an expression includes a symbolic object (or several), all the mathematical operations will be carried out exactly. For example, if *c* is replaced by *a* in the last expression, the result is exact, as it was in the first example.

```
>> g=d/a+sqrt(2)
g =
2^(1/2) + 5/3
```

Additional facts about symbolic expression and symbolic objects:

- Symbolic expressions can include numerical variables that have been obtained from the execution of numerical expressions. When these variables are inserted in symbolic expressions their exact value is used, even if the variable was displayed before with an approximated value. For example:

```
>> h=10/3
h =
    3.3333
```

h is defined to be 10/3 (a numerical variable).

An approximated value of h (numerical variable) is displayed.

```
>> k=sym(5); m=sym(7);
>> p=k/m+h
p =
85/21
```

Define k and m as symbolic 5 and 7, respectively.

h, k, and m are used in an expression.

The exact value of h is used in the determination of p.
An exact value of p (symbolic object) is displayed.

Use command `double(S)` to convert a symbolic object or expression S to numerical form

```
>> pN=double(p)
pN =
    4.0476
>> y=sym(10)*cos(5*pi/6)
y =
-5*3^(1/2)
>> yN=double(y)
yN =
   -8.6603
```

p is converted to numerical form (assigned to pN).

Create a symbolic expression y.

Exact value of y is displayed.

y is converted to numerical form (assigned to yN).

- A symbolic object that is created can also be a symbolic expression written in terms of variables that were not first created as symbolic objects. For example, the quadratic expression $ax^2 + bx + c$ can be created as a symbolic object named `f` by using the `sym` command:

```
>> f=sym('a*x^2+b*x+c')
f =
a*x^2 + b*x +c
```

It is important to understand that in this case, the variables a, b, c, and x included in the object do not exist individually as independent symbolic objects (the whole expression is one object). This means that it is impossible to perform symbolic math operations associated with the individual variables in the object. For example, it will not be possible to differentiate `f` with respect to x. This is different from the way in which the quadratic expression was created in the first example in this section, where the individual variables are first created as symbolic objects and then used in the quadratic expression.

Can use existing symbolic expressions to create new symbolic expressions – just use name of existing expression in new expression

```
>> syms x y
```
Define x and y as symbolic variables.

```
>> SA=x+y, SB=x-y
```
Create two symbolic expressions SA and SB.

```
SA =
x+y
SB =
x-y
```

$$SA = x + y$$

$$SB = x - y$$

```
>> F=SA^2/SB^3+x^2
```
Create a new symbolic expression F using SA and SB.

```
F =
(x+y)^2/(x-y)^3+x^2
```

$$F = SA^2 / SB^2 + x^2 = \frac{(x+y)^2}{(x-y)^3} + x^2$$

The `findsym` command can be used to find which symbolic variables are present in an existing symbolic expression. The format of the command is:

`findsym(S)` or `findsym(S,n)`

The `findsym(S)` command displays the names of all the symbolic variables (separated by commas) that are in the expression `S` in alphabetical order. The `findsym(S,n)` command displays `n` symbolic variables that are in expression `S` in the default order. For one-letter symbolic variables, the default order starts with x, and followed by letters, according to their closeness to x. If there are two letters equally close to x, the letter that is after x in alphabetical order is first (y before w, and z before v). The default symbolic variable in a symbolic expression is the first variable in the default order. The default symbolic variable in an expression `S` can be identified by typing `findsym(S,1)`. Examples:

```
>> syms x y
```

Define x and y as symbolic variables.

```
>> S=(x^2+x-exp(x))*(x+3)
S =
(x + 3)*(x - exp(x) + x^2)
```

Create the symbolic expression $(x+3)(x-e^x+x^2)$ and assign it to S.

```
>> F = collect(S)
```

Use the `collect` command.

```
F =
x^3+4*x^2+(3-exp(x))*x-3*exp(x)
```

MATLAB returns the expression: $x^3+4x^2+(3-e^x)x-3e^x$.

```
>> T=(2*x^2+y^2)*(x+y^2+3)
T =
(2*x^2+y^2)*(y^2+x+3)
```

Create the symbolic expression T $(2x^2+y^2)(y^2+x+3)$.

```
>> G=collect(T)
```

Use the `collect(T)` command.

MATLAB returns the expression $x^3+(2y^2+6)x^2+y^2x+y^2(y^2+3)$.

```
G =
2*x^3+(2*y^2+6)*x^2+y^2*x+y^2*(y^2+3)
>> H=collect(T,y)
```

Use the `collect(T,y)` command.

```
H =
y^4+(2*x^2+x+3)*y^2+2*x^2*(x+3)
```

MATLAB returns the expression $y^4+(2x^2+x+3)y^2+2x^2(x+3)$.

Symbolic expressions often not in simplest or preferred form. Can change form by

- Collecting terms with the same power
- Expanding products
- Factoring out common multipliers
- Using mathematical and trigonometric identities
- Many other ways

The collect command:

```
collect(S)   collect(S,variable_name)
```

- Gathers terms in expression that have the variable with the same power.
- In new expression, terms ordered in decreasing order of power
- `collect(S)` form works best when expression has only one symbolic variable

`collect(S)  collect(S,variable_name)`

- If expression has more than one variable MATLAB will collect the terms of one variable first, then of a second variable, etc.
  - Order of variables determined by MATLAB
  - User can specify the first variable by using the form `collect(S, variable_name)`

11.2.1 The `collect`, `expand`, and `factor` Commands

```
>> syms x y
```
Define x and y as symbolic variables.

```
>> S=(x^2+x-exp(x))*(x+3)
S =
(x + 3)*(x - exp(x) + x^2)
```
Create the symbolic expression $(x+3)(x-e^x+x^2)$ and assign it to S.

```
>> F = collect(S)
```
Use the `collect` command.

```
F =
x^3+4*x^2+(3-exp(x))*x-3*exp(x)
```
MATLAB returns the expression: $x^3+4x^2+(3-e^x)x-3e^x$.

```
>> T=(2*x^2+y^2)*(x+y^2+3)
T =
(2*x^2+y^2)*(y^2+x+3)
```
Create the symbolic expression T $(2x^2+y^2)(y^2+x+3)$.

```
>> G=collect(T)
```
Use the `collect(T)` command.

MATLAB returns the expression $x^3+(2y^2+6)x^2+y^2x+y^2(y^2+3)$.

```
G =
2*x^3+(2*y^2+6)*x^2+y^2*x+y^2*(y^2+3)
>> H=collect(T,y)
```
Use the `collect(T,y)` command.

```
H =
y^4+(2*x^2+x+3)*y^2+2*x^2*(x+3)
```
MATLAB returns the expression $y^4+(2x^2+x+3)y^2+2x^2(x+3)$.

Note that when `collect(T)` is used, the reformatted expression is written in order of decreasing powers of x, but when `collect(T,y)` is used, the reformatted expression is written in order of decreasing powers of y.

The expand command:

```
expand(S)
```

expand command works in two ways

1. It carries out products of terms that include summation (at least one of the terms)
2. It uses trigonometric identities and exponential and logarithmic laws to expand corresponding terms that include summation

```
>> syms a x y
>> S=(x+5)*(x-a)*(x+4)
S =
-(a-x)*(x+4)*(x+5)
>> T=expand(S)
T =
20*x-20*a-9*a*x-a*x^2+9*x^2+x^3
>> expand(sin(x-y))

ans =
cos(y)*sin(x)-cos(x)*sin(y)
```

Define a, x, and y as symbolic variables.

Create the symbolic expression $-(a-x)(x+4)(x+5)$ and assign it to S.

Use the expand command.

MATLAB returns the expression $20x-20a-9ax-ax^2+9x^2+x^3$.

Use the expand command to expand $\sin(x-y)$.

MATLAB uses trig identity for the expansion.

The factor command:

`factor(S)`

`factor` changes an expression that is a polynomial to be a product of polynomials of a lower degree

```
>> syms x
>> S=x^3+4*x^2-11*x-30
S =
x^3+4*x^2-11*x-30
>> factor(S)
ans =
(x+5)*(x-3)*(x+2)
```

Define x as a symbolic variable.

Create the symbolic expression $x^3+4x^2-11x-30$ and assign it to S.

Use the `factor` command.

MATLAB returns the expression $(x+5)(x-3)(x+2)$.

The `simplify` command:

The `simplify` command is a tool for simplifying the form of an expression. The `simplify` command uses mathematical operations (addition, multiplication, rules of fractions, powers, logarithms, etc.) and functional and trigonometric identities to generate a simpler form of the expression. The format of the `simplify` command is:

```
simplify(S)
```

where either `S` is the name of the existing expression to be simplified, or an expression to be simplified can be typed in for `S`.

Two examples are:

```
>> syms x y
```
Define x and y as symbolic variables.

```
>> S=(x^2+5*x+6)/(x+2)
S =
(x^2+5*x+6)/(x+2)
```
Create the symbolic expression $(x^2+5x+6)/(x+2)$, and assign it to `S`.

```
>> SA = simplify(S)
SA =
x+3
```
Use the `simplify` command to simplify `S`. MATLAB simplifies the expression to $x+3$.

```
>> simplify((x+y)/(1/x+1/y))
ans =
x*y
```
Simplify $(x+y)/(\frac{1}{x}+\frac{1}{y})$. MATLAB simplifies the expression to xy).

`pretty(S)` displays a symbolic expression in a form similar to the way you write the expression mathematically.

Example:

```
>> syms a b c x
>> S=sqrt(a*x^2 + b*x + c)
S =
(a*x^2+b*x+c)^(1/2)
>> pretty(S)
                    2           1/2
                (a x  + b x + c)
```

Define a, b, c, and x as symbolic variables.

Create the symbolic expression $\sqrt{ax^2 + bx + c}$, and assign it to

The `pretty` command displays the expression in a math format.

MATLAB can symbolically solve linear equations with one or more unknowns

Solving a single equation:

An algebraic equation can have one or several symbolic variables. If the equation has one variable, the solution is numerical. If the equation has several symbolic variables, a solution can be obtained for any of the variables in terms of the others. The solution is obtained by using the `solve` command, which has the form

```
h = solve(eq)
```
or
```
h = solve(eq,var)
```

- The argument `eq` can be the name of a previously created symbolic expression, or an expression that is typed in. When a previously created symbolic expression `S` is entered for `eq`, or when an expression that does not contain the = sign is typed in for `eq`, MATLAB solves the equation `eq` = 0.
- An equation of the form $f(x) = g(x)$ can be solved by typing the equation (including the = sign) as a string for `eq`.
- If the equation to be solved has more than one variable, the `solve(eq)` command solves for the default symbolic variable (see Section 11.1.3). A solution for any of the variables can be obtained with the `solve(eq,var)` command by typing the variable name for `var`.
- If the user types `solve(eq)`, the solution is assigned to the variable `ans`.
- If the equation has more than one solution, the output `h` is a symbolic column vector with a solution at each element. The elements of the vector are symbolic objects. When an array of symbolic objects is displayed, each row is enclosed with square brackets (see the following examples).

```
>> syms a b x y z
```
Define a, b, x, y, and z as symbolic variables.

```
>> h=solve(exp(2*z)-5)
h =
log(5)/2
```
Use the solve command to solve $e^{2z} - 5 = 0$.

The solution is assigned to h.

```
>> S=x^2-x-6
S =
x^2-x-6
```
Create the symbolic expression $x^2 - 6 - x$, and assign it to S.

```
>> k=solve(S)
k =
 -2
  3
```
Use the solve(S) command to solve $x^2 - 6 - x$.

The equation has two solutions. They are assigned to k, which is a column vector with symbolic

```
>> solve('cos(2*y)+3*sin(y)=2')
ans =
     pi/2
     pi/6
 (5*pi)/6
```
Use the solve command to solve $\cos(2y) + 3\sin(y) = 2$. (The equation is typed as a string in the command.)

The solution is assigned to ans.

```
>> T= a*x^2+5*b*x+20
T =
a*x^2+5*b*x+20
```
Create the symbolic expression $ax^2 + 5bx + 20$, and assign it to T.

```
>> solve(T)
ans =
 -(5*b+5^(1/2)*(5*b^2-16*a)^(1/2))/(2*a)
 -(5*b-5^(1/2)*(5*b^2-16*a)^(1/2))/(2*a)
```
Use the solve(S) command to solve $T = 0$.

The equation $T = 0$ is solved for the variable x, which is the default variable.

```
>> M = solve(T,a)
M =
-(5*b*x+20)/x^2
```
Use the solve(eq,var) command to solve $T = 0$.

The equation $T = 0$ is solved for the variable a.

Solving a system of equations:

Use `solve` to solve a system of equations

- If number of equations same as number of variables, solution is numerical
- If number of variables greater than number of equations, solution is symbolic for desired variables in terms of other variables
- If system has one solution, each solution variable has one numerical value
- If system has multiple solutions, each variable can have several values

```
output = solve(eq1,eq2,....,eqn)
```

or

```
output = solve(eq1,eq2,...,eqn,var1,var2,...,varn)
```

- The arguments `eq1,eq2,...,eqn` are the equations to be solved. Each argument can be a name of a previously created symbolic expression, or an expression that is typed in as a string. When a previously created symbolic expression `S` is entered, the equation is `S = 0`. When a string that does not contain the = sign is typed in, the equation is `expression = 0`. An equation that contains the = sign must be typed as a string.
- In the first format, if the number of equations *n* is equal to the number of variables in the equations, MATLAB gives a numerical solution for all the variables. If the number of variables is greater than the number of equations *n*, MATLAB gives a solution for *n* variables in terms of the rest of the variables. The variables for which solutions are obtained are chosen by MATLAB according to the default order (Section 11.1.3).
- When the number of variables is greater than the number of equations *n*, the user can select the variables for which the system is solved. This is done by using the second format of the `solve` command and entering the names of the variables `var1,var2,...,varn`.

Output from `solve` can have two different forms – a cell array or a structure

- *cell array* – an array in which each of the elements can be an array
  - Arrays in elements can be different dimensions
- *structure* – an array in which the elements have text names. Elements are called *fields*

When a cell array is used in the output of the `solve` command, the command has the following form (in the case of a system of three equations):

```
[varA, varB, varC] = solve(eq1,eq2,eq3)
```

- Once the command is executed, the solution is assigned to the variables `varA`, `varB`, and `varC`, and the variables are displayed with their assigned solution. Each of the variables will have one or several values (in a column vector) depending on whether the system of equations has one or several solutions.
- The user can select any names for `varA`, `varB`, and `varC`. MATLAB assigns the solution for the variables in the equations in alphabetical order. For example, if the variables for which the equations are solved are x, u, and t, the solution for t is assigned to `varA`, the solution for u is assigned to `varB`, and the solution for x is assigned to `varC`.

The following examples show how the `solve` command is used for the case where a cell array is used in the output:

```
>> syms x y t
>> S=10*x+12*y+16*t;
>> [xt yt]=solve(S, '5*x-y=13*t')
xt =
2*t
yt =
-3*t
```

Define x, y, and t as symbolic variables.

Assign to S the expression $10x + 12y + 16t$.

Use the `solve` command to solve the system: $10x + 12y + 16t = 0$, $5x - y = 13t$

Output in a cell array with two cells named `xt` and `yt`.

The solutions for x and y are assigned to `xt` and `yt`, respectively.

In the example above, notice that the system of two equations is solved by MATLAB for x and y in terms of t, since x and y are the first two variables in the default order. The system, however, can be solved for different variables. As an example, the system is solved next for y and t in terms of x (using the second form of the `solve` command:

```
>> [tx yx]=solve(S,'5*x-y=13*t',y,t)
tx =
x/2
yx =
-(3*x)/2
```

The variables for which the system is solved (y and t) are entered.

The solutions for the variables for which the system is solved are assigned in alphabetical order. The first cell has the solution for t, and the second cell has the solution for y.

When a structure is used in the output of the `solve` command, the command has the form (in the case of a system of three equations)

```
AN = solve(eq1,eq2,eq3)
```

- `AN` is the name of the structure.
- Once the command is executed the solution is assigned to `AN`. MATLAB displays the name of the structure and the names of the fields of the structure, which are the names of the variables for which the equations are solved. The size and the type of each field is displayed next to the field name. The content of each field, which is the solution for the variable, is not displayed.
- To display the content of a field (the solution for the variable), the user has to type the address of the field. The form for typing the address is: `structure_name.field_name` (see example below).

As an illustration the system of equations solved in the last example is solved again using a structure for the output.

```
>> syms x y t
>> S=10*x+12*y+16*t;
>> AN=solve(S,'5*x-y=13*t')
AN =
   x: [1x1 sym]
   y: [1x1 sym]
>> AN.x
ans =
2*t
>> AN.y
ans =
-3*t
```

Use the `solve` command to solve the system: $10x + 12y + 16t = 0$, $5x - y = 13t$

MATLAB displays the name of the structure `AN` and the names of its fields `x` and `y` (size and type), which are the names of the variables for which the equations are solved.

Type the address of the field `x`.

The content of the field (the solution for x) is displayed.

Type the address of the field `y`.

The content of the field (the solution for y) is displayed.

To do symbolic differentiation, use

`diff(S)` or `diff(S,var)`

- `S` is name of existing symbolic expression, or can type in an expression
- For `diff(S)`
  - If only one variable in `S`, command differentiates with respect to that variable
  - If multiple variables in `S`, command differentiates with respect to first variable in default order
- `diff(S,var)` differentiates with respect to `var`

To get the *n*th derivative, use

`diff(S,n)` or `diff(S,var,n)`

- e.g., `diff(S,2)` is second derivative, `diff(S,3)` is third derivative, etc.

```
>> syms x y t
```
Define x, y, and t as symbolic variables.

```
>> S=exp(x^4);
```
Assign to S the expression e^{x^4}.

```
>> diff(S)
```
Use the diff(S) command to differentiate S.

```
ans =
4*x^3*exp(x^4)
```
The answer $4x^3e^{x^4}$ is displayed.

```
>> diff((1-4*x)^3)
```
Use the diff(S) command to differentiate $(1-4x)^3$.

```
ans =
-12*(1-4*x)^2
```
The answer $-12(1-4x)^3$ is displayed.

```
>> R=5*y^2*cos(3*t);
```
Assign to R the expression $5y^2\cos(3t)$.

```
>> diff(R)
```
Use the diff(R) command to differentiate R.

```
ans =
10*y*cos(3*t)
```
MATLAB differentiates R with respect to y (default symbolic variable); the answer $10y\cos(3t)$ is displayed.

```
>> diff(R,t)
```
Use the diff(R,t) command to differentiate R w.r.t. t.

```
ans =
-15*y^2*sin(3*t)
```
The answer $-15y^2\sin(3t)$ is displayed.

```
>> diff(S,2)
```
Use diff(S,2) command to obtain the second derivative of S.

```
ans =
12*x^2*exp(x^4)+16*x^6*exp(x^4)
```
The answer $12x^2e^{x^4} + 16x^6e^{x^4}$ is displayed.

Use `int(S)` or `int(S,var)` to compute a symbolic definite or indefinite integral

- Either `S` can be the name of a previously created symbolic expression, or an expression can be typed in for `S`.
- In the `int(S)` command, if the expression contains one symbolic variable, the integration is carried out with respect to that variable. If the expression contains more than one variable, the integration is carried out with respect to the default symbolic variable (Section 11.1.3).
- In the `int(S,var)` command, which is used for integration of expressions with several symbolic variables, the integration is carried out with respect to the variable `var`.

Examples

```
>> syms x y t
```
Define x, y, and t as symbolic variables.

```
>> S=2*cos(x)-6*x;
```
Assign to S the expression $2\cos x - 6x$.

```
>> int(S)
```
Use the `int(S)` command to integrate S.

```
ans =
2*sin(x)-3*x^2
```
The answer $2\sin x - 3x^2$ is displayed.

```
>> int(x*sin(x))
```
Use the `int(S)` command to integrate $x\sin(x)$.

```
ans =
sin(x)-x*cos(x)
```
The answer $\sin x - x\cos x$ is displayed.

```
>>R=5*y^2*cos(4*t);
```
Assign to R the expression $5y^2\cos(4t)$.

```
>> int(R)
```
Use the `int(R)` command to integrate R.

```
ans =
(5*y^3*cos(4*t))/3
```
MATLAB integrates R with respect to y (default symbolic variable); the answer $5y^3\cos(4t)/3$ is displayed.

```
>> int(R,t)
```
Use the `int(R,t)` command to integrate R w.r.t. t.

```
ans =
(5*y^2*sin(4*t))/4
```
The answer $5y^2\cos(4t)/4$ is displayed.

For definite integration the form of the command is:

`int(S,a,b)` or `int(S,var,a,b)`

- `a` and `b` are the limits of integration. The limits can be numbers or symbolic variables. $(\sin y - 5y^2)dy$

For example, determination of the definite integral $\int_0^{\pi} (\sin y - 5y^2)dy$ with MATLAB is:

```
>> syms y
>> int(sin(y)-5*y^2,0,pi)
ans =
2 - (5*pi^3)/3
```

- It is possible also to use the `int` command by typing the expression to be integrated as a string without having the variables in the expression first created as symbolic objects. However, the variables in the integrated expression do not exist as independent symbolic objects.
- Integration can sometimes be a difficult task. A closed-form answer may not exist, or if it exists, MATLAB might not be able to find it. When that happens MATLAB returns `int(S)` and the message `Explicit integral could not be found.`

A first-order ordinary, differential equation (ODE) contains the first derivative of the dependent variable

- For example, $\frac{dy}{dt} = f(t, y)$, where t is the independent variable and $y=y(t)$ is the dependent variable

A second-order ODE contains the second derivative (and possibly also the first), i.e., $\frac{dy}{dt} = f(t, y, \frac{dy}{dt})$

A *solution* is a function *y(t)* that satisfies the differential equation

- A *general solution* contains constants whose values are unknown
- In a *particular solution* the values of the constants are set by initial conditions

Use `dsolve` to get a general or particular solution to an ODE

General solution:

For obtaining a general solution, the `dsolve` command has the form:

`dsolve('eq')` or `dsolve('eq','var')`

- `eq` is the equation to be solved. It has to be typed as a string (even if the variables are symbolic objects).
- The variables in the equation don't have to first be created as symbolic objects. (If they have not been created, then, in the solution the variables will not be symbolic objects.)
- Any letter (lowercase or uppercase), except `D` can be used for the dependent variable.
- In the `dsolve('eq')` command the independent variable is assumed by MATLAB to be `t` (default).
- In the `dsolve('eq','var')` command the user defines the independent variable by typing it for `var` (as a string).
- In specifying the equation the letter `D` denotes differentiation. If y is the dependent variable and t is the independent variable, `Dy` stands for $\frac{dy}{dt}$. For example, the equation $\frac{dy}{dt} + 3y = 100$ is typed in as `'Dy + 3*y = 100'`.
- A second derivative is typed as `D2`, third derivative as `D3`, and so on. For example, the equation $\frac{d^2y}{dt^2} + 3\frac{dy}{dt} + 5y = \sin(t)$ is typed in as: `'D2y + 3*Dy + 5*y = sin(t)'`.
- The variables in the ODE equation that is typed in the `dsolve` command do not have to be previously created symbolic variables.
- In the solution MATLAB uses `C1`, `C2`, `C3`, and so on, for the constants of integration.

For example, a general solution of the first-order ODE $\frac{dy}{dt} = 4t + 2y$ is obtained by:

```
>> dsolve('Dy=4*t+2*y')

ans =
C1*exp(2*t) - 2*t - 1
```

The answer $y = C_1 e^{2t} - 2t - 1$ is displayed.

A general solution of the second-order ODE $\frac{d^2x}{dt^2} + 2\frac{dx}{dt} + x = 0$ is obtained by:

```
>> dsolve('D2x+2*Dx+x=0')

ans =
C1/exp(t)+(C2*t)/exp(t)
```

The answer $x = C_1 e^{-t} + C_2 t e^{-t}$ is displayed.

The following examples illustrate the solution of differential equations that contain symbolic variables in addition to the independent and dependent variables.

The following examples illustrate the solution of differential equations that contain symbolic variables in addition to the independent and dependent variables.

```
>> dsolve('Ds=a*x^2')

ans =
a*t*x^2 + C1
```

The independent variable is t (default). MATLAB solves the equation $\frac{ds}{dt} = ax^2$.

The solution $s = ax^2t + C_1$ is displayed.

```
>> dsolve('Ds=a*x^2','x')

ans =
(a*x^3)/3 + C1
```

The independent variable is defined to be x. MATLAB solves the equation $\frac{ds}{dx} = ax^2$.

The solution $s = \frac{1}{3}ax^3 + C_1$ is displayed.

```
>> dsolve('Ds=a*x^2','a')

ans =
(a^2*x^2)/2 + C2
```

The independent variable is defined to be a. MATLAB solves the equation $\frac{ds}{da} = ax^2$.

The solution $s = \frac{1}{2}a^2x^2 + C_1$ is displayed.

Particular solution:

A particular solution of an ODE can be obtained if boundary (or initial) conditions are specified. A first-order equation requires one condition, a second-order equation requires two conditions, and so on. For obtaining a particular solution, the `dsolve` command has the form

First-order ODE:

```
dsolve('eq','cond1','var')
```

Higher-order ODE:

```
dsolve('eq','cond1','cond2',....,'var')
```

- For solving equations of higher order, additional boundary conditions have to be entered in the command. If the number of conditions is less than the order of the equation, MATLAB returns a solution that includes constants of integration (`C1`, `C2`, `C3`, and so on).
- The boundary conditions are typed in as strings in the following:

Math form	MATLAB form
$y(a) = A$	`'y(a)=A'`
$y'(a) = A$	`'Dy(a)=A'`
$y''(a) = A$	`'D2y(a)=A'`

- The argument `'var'` is optional and is used to define the independent variable in the equation. If none is entered, the default is t.

For example, the first-order ODE $\frac{dy}{dt} + 4y = 60$, with the initial condition $y(0) = 5$ is solved with MATLAB by:

```
>> dsolve('Dy+4*y=60','y(0)=5')

ans =
15 - 10/exp(4*t)
```

The answer $y = 15 - 10 / e^{4t}$ is displayed.

The second-order ODE $\frac{d^2y}{dt^2} - 2\frac{dy}{dt} + 2y = 0$, $y(0) = 1$, $\left.\frac{dy}{dt}\right|_{t=0} = 0$, can be solved with MATLAB by:

```
>> dsolve('D2y-2*Dy+2*y=0','y(0)=1','Dy(0)=0')

ans =
exp(t)*cos(t)-exp(t)*sin(t)
>> factor(ans)
ans =
exp(t)*(cos(t)-sin(t))
```

The answer $y = e^t \cos t - e^t \sin t$ is displayed.

The answer can be simplified with the `factor` command.

The simplified answer $y = e^t (\cos t - \sin t)$ is displayed.

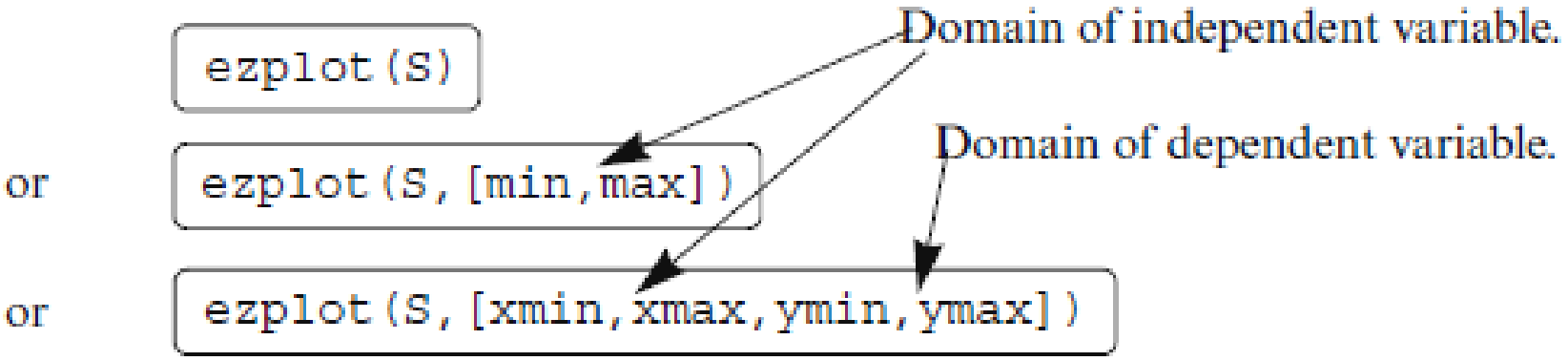
Additional examples of solving differential equations are shown in Sample Problem 11-5.

If MATLAB cannot find a solution, it returns an empty symbolic object and the message `Warning: explicit solution could not be found.`

Can easily plot symbolic expressions with `ezplot`

- If symbolic expression `S` contains one symbolic variable `var`, `ezplot` plots `S(var)` vs `var`
- If it contains two symbolic variables, `ezplot` graphs `S(var1,var2)=0` in the plane

To plot a symbolic expression `S` that contains one or two variables, the `ezplot` command is:



- `S` is the symbolic expression to be plotted. It can be the name of a previously created symbolic expression, or an expression can be typed in for `S`.
- It is also possible to type the expression to be plotted as a string without having the variables in the expression first created as symbolic objects.
- If `S` has one symbolic variable, a plot of $S(var)$ versus (var) is created, with the values of var (the independent variable) on the abscissa (horizontal axis), and the values of $S(var)$ on the ordinate (vertical axis).

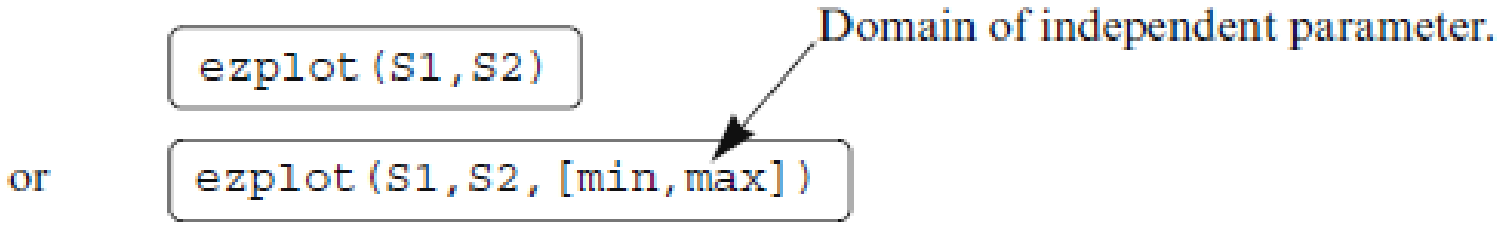
- If the symbolic expression `S` has two symbolic variables, `var1` and `var2`, the expression is assumed to be a function with the form $S(var1, var2) = 0$. MATLAB creates a plot of one variable versus the other variable. The variable that is first in alphabetic order is taken to be the independent variable. For example, if the variables in S are x and y, then x is the independent variable and is plotted on the abscissa and y is the dependent variable plotted on the ordinate. If the variables in S are u and v, then u is the independent variable and v is the dependent variable.

- In the `ezplot(S)` command, if `S` has one variable ($S(var)$), the plot is over the domain $-2\pi \leq var \leq 2\pi$ (default domain) and the range is selected by MATLAB. If `S` has two variables ($S(var1, var2)$), the plot is over $-2\pi \leq var1 \leq 2\pi$ and $-2\pi \leq var2 \leq 2\pi$.

- In the `ezplot(S,[min,max])` command the domain for the independent variable is defined by `min` and `max`:—$min \leq var \leq max$—and the range is selected by MATLAB.

- In the `ezplot(S,[xmin,xmax,ymin,ymax])` command the domain for the independent variable is defined by `xmin` and `xmax`, and the domain of the dependent variable is defined by `ymin` and `ymax`.

Can also use `ezplot` to plot a 2-D parametric curve, e.g., $x=x(t)$ and $y=y(t)$



- `S1` and `S2` are symbolic expressions containing the same single symbolic variable, which is the independent parameter. `S1` and `S2` can be the names of previously created symbolic expressions, or expressions can be typed in.
- The command creates a plot of $S2(var)$ versus $S1(var)$. The symbolic expression that is typed first in the command (`S1` in the definition above) is used for the horizontal axis, and the expression that is typed second (`S2` in the definition above) is used for the vertical axis.
- In the `ezplot(S1,S2)` command the domain of the independent variable is $0 < var < 2\pi$ (default domain).
- In the `ezplot(S1,S2,[min,max])` command the domain for the independent variable is defined by `min` and `max`: $min < var < max$.

Additional comments:

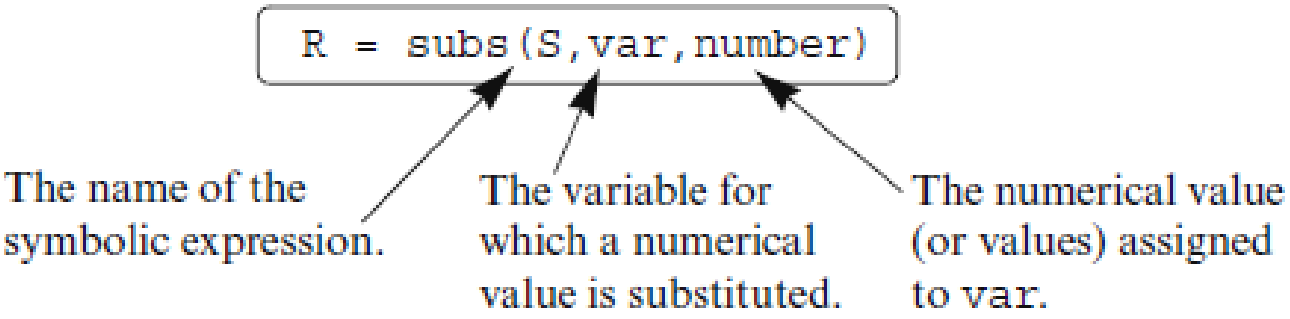
Once you create plot with `ezplot`, can format same way as plots made with `plot`

- When plot symbolic expression with `ezplot` it automatically displays expression at top of plot

Table 11-1 in book shows several examples of symbolic-expression plots

After manipulating symbolic expression, often need to evaluate it numerically. Can do this with the MATLAB command `subs`, which substitutes numerical values for symbolic ones

Substituting a numerical value for one symbolic variable:



- `number` can be one number (a scalar), or an array with many elements (a vector or a matrix).
- The value of `S` is calculated for each value of `number` and the result is assigned to `R`, which will have the same size as `number` (scalar, vector, or matrix).
- If `S` has one variable, the output `R` is numerical. If `S` has several variables and a numerical value is substituted for only one of them, the output `R` is a symbolic expression.

An example with an expression that includes one symbolic variable is:

```
>> syms x
>> S=0.8*x^3+4*exp(0.5*x)
S =
4*exp(x/2) + (4*x^3)/5
>> SD=diff(S)
SD =
2*exp(x/2)+(12*x^2)/5)
>> subs(SD, x, 2)
ans =
   15.0366
>> SDU=subs(SD, x, [2:0.5:4])

SDU =
   15.0366   21.9807   30.5634   40.9092   53.1781
```

Define x as a symbolic variable.

Assign to S the expression $0.8x^3 + 4e^{(0.5x)}$.

Use the diff(S) command to differentiate S.

The answer $2e^{x/2} + 12x^2/5$ is assigned to SD.

Use the subs command to substitute $x = 2$ in SD.

The value of SD is displayed.

Use the subs command to substitute $x = [2, 2.5, 3, 3.5, 4]$ (vector) in SD.

The values of SD (assigned to SDU) for each value of x are displayed in a vector.

In the last example, notice that when the numerical value of the symbolic expression is calculated, the answer is numerical (the display is indented). An example of substituting numerical values for one symbolic variable in an expression that has several symbolic variables is:

```
>> syms a g t v
>> Y=v^2*exp(a*t)/g
Y =
v^2*exp(a*t)/g
>> subs(Y,t,2)
ans =
v^2*exp(2*a)/g
>> Yt=subs(Y,t,[2:4])

Yt =
[ v^2*exp(2*a)/g, v^2*exp(3*a)/g, v^2*exp(4*a)/g]
```

Define a, g, t, and v as symbolic variables.

Create the symbolic expression $v^2 e^{at} / g$ and assign it to Y.

Use the subs command to substitute $t = 2$ in SD.

The answer $v^2 e^{2a} / g$ is displayed.

Use the subs command to substitute $t = [2, 3, 4]$ (vector) in Y.

The answer is a vector with elements of symbolic expressions for each value of t.

Substituting a numerical value for two or more symbolic variables:

Can use `subs` to substitute for two or more symbolic variables. For example, for two symbolic variables

```
R = subs(S,{var1,var2},{number1,number2})
```

The name of the symbolic expression.

The variables for which numerical values are substituted.

The numerical value (or values) assigned to `var1` and `var2`.

- The variables `var1` and `var2` are the variables in the expression `S` for which the numerical values are substituted. The variables are typed as a cell array (inside curly braces { }). A cell array is an array of cells where each cell can be an array of numbers or text.
- The numbers `number1`, `number2` substituted for the variables are also typed as a cell array (inside curly braces { }). The numbers can be scalars, vectors, or matrices. The first cell in the numbers cell array (`number1`) is substituted for the variable that is in the first cell of the variable cell array (`var1`), and so on.
- If all the numbers that are substituted for variables are scalars, the outcome will be one number or one expression (if some of the variables are still symbolic).
- If, for at least one variable, the substituted numbers are an array, the mathematical operations are executed element-by-element and the outcome is an array of numbers or expressions. It should be emphasized that the calculations are performed element-by-element even though the expression `S` is not typed in the element-by-element notation. This also means that all the arrays substituted for different variables must be of the same size.
- It is possible to substitute arrays (of the same size) for some of the variables and scalars for other variables. In this case, in order to carry out element-by-element operations, MATLAB expands the scalars (array of `1`s times the scalar) to produce an array result.

```
>> syms a b c e x
```
Define a, b, c, e, and x as symbolic variables.

```
>> S=a*x^e+b*x+c

S =
a*x^e+b*x+c
```
Create the symbolic expression $ax^e + bx + c$ and assigned it to S.

```
>> subs(S,{a,b,c,e,x},{5,4,-20,2,3})
```
Substitute in S scalars for all the symbolic variables.

Cell array. Cell array.

```
ans =
    37
```
The value of S is displayed.

```
>> T=subs(S,{a,b,c},{6,5,7})

T =
5*x+ 6*x^e+7
```
Substitute in S scalars for the symbolic variables a, b, and c.

The result is an expression with the variables x and

```
>> R=subs(S,{b,c,e},{[2 4 6],9,[1 3 5]})

R =
[  2*x+a*x+9, a*x^3+4*x+9, a*x^5+6*x+9]
```
Substitute in S a scalar for c, and vectors for b and e.

The result is a vector of symbolic expressions.

```
>> W=subs(S,{a,b,c,e,x},{[4 2 0],[2 4 6],[2 2 2],[1 3 5],[3 2 1]})
```
Substitute in S vectors for all the variables.

```
W =
    20    26     8
```
The result is a vector of numerical values.

Can also substitute numerical values into a symbolic expression by first setting symbolic variables in the expression to numerical values and then calling `subs`

```
R = subs(S)
```

- Note that once you redefine symbolic variables to numerical variables, you can't use them as symbolic variables any more

```
>> syms A c m x y
>> S=A*cos(m*x)+c*y
S =
c*y+A*cos(m*x)
>> A=10; m=0.5; c=3;
>> subs(S)
ans =
3*y + 10*cos(x/2)
>> x=linspace(0,2*pi,4);
>> T = subs(S)
T =
[ 3*y+10, 3*y+5, 3*y-5, 3*y-10]
```

Define A, c, m, x, and y as symbolic variables

Create the symbolic expression $A\cos(mx) + cy$ and assign it to S.

Assign numerical values to variables A, m, and c.

Use the subs command with the expression S.

The numerical values of variables A, m, and c are substituted in S.

Assign numerical values (vector) to variable x.

Use the subs command with the expression S.

The numerical values of variables A, m, c, and x are substituted. The result is a vector of symbolic expressions.