can also be used as separate independent software. That software uses the MuPAD language, which has a completely different structure and commands than MATLAB. The Symbolic Math Toolbox is included in the student version of MATLAB. In the standard version, the toolbox is purchased separately. To check if the Symbolic Math Toolbox is installed on a computer, the user can type the command `ver` in the Command Window. In response, MATLAB displays information about the version that is used as well as a list of the toolboxes that are installed.

The starting point for symbolic operations is symbolic objects. Symbolic objects are made of variables and numbers that, when used in mathematical expressions, tell MATLAB to execute the expression symbolically. Typically, the user first defines (creates) the symbolic variables (objects) that are needed, and then uses them to create symbolic expressions that are subsequently used in symbolic operations. If needed, symbolic expressions can be used in numerical operations

The first section in this chapter describes how to define symbolic objects and how to use them to create symbolic expressions. The second section shows how to change the form of existing expressions. Once a symbolic expression has been created, it can be used in mathematical operations. MATLAB has a large selection of functions for this purpose. The next four sections (11.3–11.6) describe how to use MATLAB to solve algebraic equations, to carry out differentiation and integration, and to solve differential equations. Section 11.7 covers plotting symbolic expressions. How to use symbolic expressions in subsequent numerical calculations is explained in the following section.

## 11.1 SYMBOLIC OBJECTS AND SYMBOLIC EXPRESSIONS

A symbolic object can be a variable (without a preassigned numerical value), a number, or an expression made of symbolic variables and numbers. A symbolic expression is a mathematical expression containing one or more symbolic objects. When typed, a symbolic expression may look like a standard numerical expression. However, because the expression contains symbolic objects, it is executed by MATLAB symbolically.

### 11.1.1 Creating Symbolic Objects

Symbolic objects can be variables or numbers. They can be created with the `sym` and/or `syms` commands. A single symbolic object can be created with the `sym` command:

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

where the string, which is the symbolic object, is assigned to a name. The string can be:

- A single letter or a combination of several letters (no spaces). Examples: `'a'`, `'x'`, `'yad'`.

- A combination of letters and digits starting with a letter and with no spaces Examples: `'xh12'`,`'r2d2'`.
- A number. Examples: `'15'`, `'4'`.

In the first two cases (where the string is a single letter, a combination of several letters, or a combination of letters and digits), the symbolic object is a symbolic variable. In this case it is convenient (but not necessary) to give the object the same name as the string. For example, *a*, *bb*, and *x*, can be defined as symbolic variables as follows:

```
>> 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`.

As mentioned, symbolic objects can also be numbers. The numbers don't have to be typed as strings. For example, the `sym` command is used next to create symbolic objects from the numbers 5 and 7 and assign them to the variables *c* and *d*, respectively.

```
>> c=sym(5)
c =
5
>> d=sym(7)
d =
7
```

Create a symbolic object from the number 5 and assign it to `c`.

The display of a symbolic object is not indented.

As shown, when a symbolic object is created and a semicolon is not typed at the end of the command, MATLAB displays the name of the object and the object itself in the next two lines. The display of symbolic objects starts at the beginning of the line and is not indented as is the display of numerical variables. The difference is illustrated below, where a numerical variable is created.

```
>> e=13
e =
    13
```

13 is assigned to e (numerical variable).

The display of the value of a numerical variable is indented.

Several symbolic variables can be created in one command by using the syms command, which has the form:

```
syms variable_name variable_name variable_name
```

The command creates symbolic objects that have the same names as the symbolic variables. For example, the variables *y*, *z*, and *d* can all be created as symbolic variables in one command by typing:

```
>> 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.

When the syms command is executed, the variables it creates are not displayed automatically—even if a semicolon is not typed at the end of the command.

### *11.1.2 Creating Symbolic Expressions*

Symbolic expressions are mathematical expressions written in terms of symbolic variables. Once symbolic variables are created, they can be used for creating symbolic expressions. The symbolic expression is a symbolic object (the display is not indented). The form for creating a symbolic expression is:

```
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);
```

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

```
>> e=b/a+sqrt(2)
```

Create an expression that includes a and b.

```
e =
2^(1/2) + 5/3
```

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

```
>> c=3; d=5;
```

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

```
>> f=d/c+sqrt(2)
```

Create an expression that includes c and d.

```
f =
    3.0809
```

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

```
h =
    3.3333
```

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

```
>> k=sym(5); m=sym(7);    Define k and m as symbolic 5 and 7, respectively.
>> p=k/m+h                       h, k, and m are used in an expression.
p =                       The exact value of h is used in the determination of p.
85/21                     An exact value of p (symbolic object) is displayed.
```

- The `double(S)` command can be used to convert a symbolic expression (object) `S` that is written in an exact form to numerical form. (The name "double" comes from the fact that the command returns a double-precision floating-point number representing the value of `S`.) Two examples are shown. In the first, the `p` from the last example is converted into numerical form. In the second, a symbolic object is created and then converted into numerical form.

```
>> pN=double(p)           p is converted to numerical form (assigned to pN).
pN =
    4.0476
>> y=sym(10)*cos(5*pi/6)                 Create a symbolic expression y.
y =
-5*3^(1/2)                                Exact value of y is displayed.
>> yN=double(y)            y is converted to numerical form (assigned to yN).
yN =
   -8.6603
```

- 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.

- Existing symbolic expressions can be used to create new symbolic expressions. This is done by simply using the name of the existing expression in the new expression. For example:

```
>> 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
>> 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
```

$SA = x + y$

$SB = x - y$

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

### *11.1.3 The* `findsym` *Command and the Default Symbolic Variable*

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 h w y d t         Define x, h, w, y, d, and t as symbolic variables.
>> S=h*x^2+d*y^2+t*w^2      Create a symbolic expression S.
S =
t*w^2 + h*x^2 + d*y^2
>> findsym(S)               Use the findsym(S) command.
ans =                       The symbolic variables are displayed in alphabetical order.
d, h, t, w, x, y
>> findsym(S,5)             Use the findsym(S,n) command (n = 5).
ans =                       Five symbolic variables are displayed in the default order.
x,y,w,t,h
>> findsym(S,1)             Use the findsym(S,n) command with n = 1.
ans =                       The default symbolic variable is displayed.
x
```