CHAPTER 8

Symbolic Computation

8.1 Introduction

Until now, we have been dealing with numeric computation where variables store numeric values. In Chapter 7, you learned that numerical computation involves working with *approximate* solutions. On the other hand, an analytical solution is not an approximation since one uses *symbols* rather than numbers. MATLAB provides the means to perform symbolic computations, too.

8.2 Defining a Symbolic Variable

The keyword `syms` is used to define single or multiple *symbolic variable(s)*. The key feature of a symbolic variable is that it just stores a symbol to perform symbolic calculations.

```
1  >> syms x y z
2  >>
```

After executing the command, inspect the Workspace window (see Figure 8-1) and note that three new variables—`x`, `y`, and `z`—have been created.

S. Nagar, *Introduction to MATLAB for Engineers and Scientists*,
https://doi.org/10.1007/978-1-4842-3189-0_8

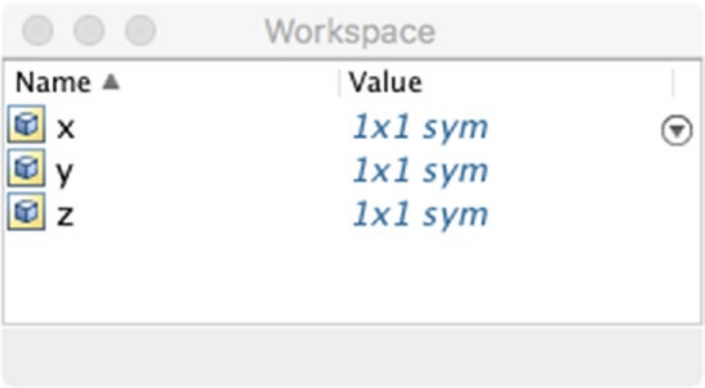

Figure 8-1. *New symbolic variables appearing in the workspace*

8.3 Defining a Symbolic Equation

Once the variables have been defined, you can define an equation:

$$z = x^2 + y \qquad \text{(Equation 8-1)}$$

using these variables as follows:

```
1  >> z = x^2+y
2  z =
3  x^2 + y
```

In the present example, z was predefined as a symbolic variable. The output variable is created by MATLAB and becomes a symbolic variable by default. Its inputs have been defined as symbolic variables. For example, suppose you want to define this equation:

$$a = x^3 + y^2 + z \qquad \text{(Equation 8-2)}$$

This results in the creation of a new symbolic variable (which can be verified by checking the Workspace window). The following MATLAB code performs this task:

```
>> syms x y z
>> a = x^3+y^2+z
a =
x^3 + y^2 + z
```

8.4 Performing Symbolic Computations

Symbolic computations are same as what we are used to doing by hand on paper. You define a variable and use mathematical rules of algebra as well as calculus to perform calculations. For example, two roots (r_1 and r_2) of a quadratic equation:

$$y = ax^2 + bx + c \qquad \text{(Equation 8-3)}$$

can be written as follows:

$$r_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad \text{(Equation 8-4)}$$

$$r_1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \qquad \text{(Equation 8-5)}$$

This can be performed using the following MATLAB code:

```
>> syms a b c x
>> y = a*(x^2)+(b*x)+c
y =
a*x^2 + b*x + c
>> solve(y)
ans =
-(b + (b^2 - 4*a*c)^(1/2))/(2*a)
-(b - (b^2 - 4*a*c)^(1/2))/(2*a)
```

Similarly, a symbolic mathematical expression can be integrated and differentiated as follows:

```
>> syms a b c x
>> y = a*(x^2)+(b*x)+c
y =
a*x^2 + b*x + c
>> int(y)
ans =
(a*x^3)/3 + (b*x^2)/2 + c*x
>> diff(y)
ans =
b + 2*a*x
```

This can be verified using paper-based calculation by hand, where we know the following:

$$y = ax^2 + bx + c \qquad \text{(Equation 8-6)}$$

$$\int y = \frac{ax^3}{3} + \frac{bx^2}{2} + cx \qquad \text{(Equation 8-7)}$$

$$\frac{dy}{dx} = 2ax + b \qquad \text{(Equation 8-8)}$$

8.4.1 Arithmetic Expressions

Simple arithmetic expressions can be dealt with using symbols. For example, two polynomials can be used to define a new polynomial.

```
>> syms x y z
>> a1 = x^2+2*y+z
a1 =
x^2 + 2*y + z
```

```
>> a2 = x^(-2)-2*y+3*z
a2 =
3*z - 2*y + 1/x^2
>> a3 = a1/a2
a3 =
(x^2 + 2*y + z)/(3*z - 2*y + 1/x^2)
>> a4 = a1*a2
a4 =
(x^2 + 2*y + z)*(3*z - 2*y+1/x^2)
```

8.4.2 Trigonometric Expressions

Trigonometric variables defined using symbolic variables can also be used in mathematical calculations, as follows:

$$y = \sin(x) \qquad \text{(Equation 8-9)}$$

$$\frac{dy}{dx} = -\cos(x) \qquad \text{(Equation 8-10)}$$

```
>> syms a b c x
>> y = sin(x)
y =
sin(x)
>> int(y)
ans =
-cos(x)
```

Even more complicated calculations can be performed by a click of a button.

```
>> z = cos(x^(1/2)) - (sin(y))^(1/3)
z =
cos(x^(1/2)) - sin(y)^(1/3)
```

```
>> int(z)
ans =
2*cos(x^(1/2)) – x*sin(y)^(1/3) + 2*x^(1/2)*sin(x^(1/2))
>> diff(z)
ans =
–sin(x^(1/2))/(2*x^(1/2))
```

8.4.3 Expanding and Factorizing an Expression

The `expand()` function can be used to write equations with individual terms of expanded polynomials. The most important use of `expand()` is the application of the distributivity law to rewrite products of sums as sums of products. If `f` represents a symbolic expression, then `expand(f)` is calculated using the following set of rules:

- $x^{a+b} = x^a \times x^b$
- $(xy)^b = x^b \times y^b \forall x, y \geq 0, b \in I$
- $\left(x^a\right)^b = x^{a+b}$

It is also important to note that the `expand()` function will work recursively on the subexpressions of a given expression.

```
>> syms x y z
>> a1 = x^2+2*y+z
a1 =
x^2 + 2*y + z
>> a2 = x^(–2)–2*y+3*z
a2 =
3*z – 2*y + 1/x^2
>> a3 = a1/a2
a3 =
(x^2 + 2*y + z)/(3*z – 2*y + 1/x^2)
```

```
>> a4 = a1*a2
a4 =
(x^2 + 2*y + z)*(3*z - 2*y + 1/x^2)
>> a5 = expand(a3)
a5 =
(2*y)/(3*z - 2*y + 1/x^2) + z/(3*z - 2*y + 1/x^2) + x^2/
(3*z - 2*y + 1/x^2)
>> a6 = expand(a4)
a6 =
4*y*z + (2*y)/x^2 - 2*x^2*y + z/x^2 + 3*x^2*z - 4*y^2 +
3*z^2 + 1
```

The function named `factor` produces factors of an expression such that multiplying all factors results in the final expression. Let's try to factorize the values stored in symbolic variable `a5` and `a6`.

```
>> a7 = factor(a5)
a7 =
[-1,x,x, x^2 + 2*y + z, -1/(3*x^2*z - 2*x^2*y + 1)]
>> a8 = factor(a5)
a8 =
[-1,x,x, x^2 + 2*y + z, -1/(3*x^2*z - 2*x^2*y + 1)]
```

The factors are present as elements of an array, which can be accessed using their index. This comes in handy when extracting a factor and its usage in mathematical analysis.

```
>> a7[2] = x
>> a7[4] = x^2 + 2*y + z
>> a8[3:5] = [x, x^2 + 2*y + z, -1/(3*x^2*z - 2*x^2*y + 1)]
```

In the previous example, `a7[2]` extracts the second element of variable `a7`, `a7[4]` extracts the fourth element of variable `a7`, and `a8[3:5]` extracts all elements from the third to fifth element and stores them as a list of symbolic expressions.

When an expression is written as a power of another expression, `expand()` works just like mathematical rules. For example, consider the case when an expression:

$$a = xy^{(z+y)}$$

is defined. Its expansion is given as:

$$x \times y^{y} \times y^{z}$$

Each term is clearly a factor of the expression. This can be verified with the following MATLAB code:

```
>> syms x y z
>> a = x*y^(z+y)
a =
x*y^(y + z)
>> b = expand(a)
b =
x*y^y*y^z
>> c = factor(b)
c =
[x,y^y,y^z]
```

When an expression is powered by another expression, the `expand()` function works recursively.

```
>> a=((x+y)^(x+z+2))
a =
(x + y)^(x + z + 2)
>> expand(a)
```

```
ans =
x^2*(x + y)^x*(x + y)^z + y^2*(x + y)^x*(x + y)^z + 2*x*y*
(x + y)^x*
(x + y)^z
```

It can be used to check out trigonometric identities:

```
>> expand(sin(x+y))
ans =
cos(x)*sin(y) + cos(y)*sin(x)
>> expand(cos(x+y))
ans =
cos(x)*cos(y) - sin(x)*sin(y)
>> expand(tan(x+y))
ans =
-(tan(x) + tan(y))/(tan(x)*tan(y) - 1)
>> expand(sec(x+y))
ans =
1/(cos(x)*cos(y) - sin(x)*sin(y))
>> a = cosh(x+y)
a =
cosh(x+y)
>> expand(a)
ans =
cosh(x)*cosh(y) + sinh(x)*sinh(y)
>> a = cosh(2*x)
a =
cosh(2*x)
>> expand(a)
ans =
2*cosh(x)^2 - 1
```

```
>> a = coth(x+y)
a =
coth(x + y)
>> expand(a)
ans =
(coth(x)*coth(y) + 1)/(coth(x) + coth(y))
```

8.5 Summary

This chapter illustrated the usage of symbols to solve mathematical equations. Symbolic computation proves useful when error-prone numerical computing is not acceptable, but it has its limits. A limited set of built-in functions must be appended by user-defined functions, and this requires experience with writing MATLAB packages. But it is definitely worth exploring.

This book has illustrated the use of MATLAB as a tool for efficient scientific computing. It first illustrated the basic usage using single-line commands and then illustrated writing multi-line commands as an `.m` file. Arrays for the fundamental blocks of scientific computing and thus matrix-based calculations can be performed using arrays. Plotting graphs is simplified to the extent that even a beginner can easily plot a equation to visualize a graph. Using loops and functions, programs can be made modular and information flow can be controlled in an efficient fashion. You also saw some basic examples of numerical computing. The book should enable any beginner to enter the world of scientific computing with ease. Its widely popular usage has rightly coined the phrase "MATLAB is the language of engineering".