1 Expressions, Variables and Built-in Functions

Mathematica is an expression evaluator. You input an expression and *Mathematica* outputs its evaluation. This Chapter addresses the basics of how you form expressions to communicate with *Mathematica*.

1.1 Expressions

As we begin, we assume you've been able to successfully start up *Mathematica* and that a new window has been opened on your computer screen. You're now ready to start typing.

1.1.1 Simple Expressions

You can enter numerical expressions quite easily into *Mathematica* and it will respond (*you type the first line and either press the* ***Enter*** *key on the numeric keypad, or* ***Shift-Return*** *on the keyboard, to make Mathematica respond with what you see on the second line*):

```
(3+4)(4-8)
```

-28

Addition and subtraction are written in the obvious way. Parentheses are used for grouping. Multiplication of terms is denoted by writing the terms or factors next to each other, as above. However, we may often prefer to use the explicit multiplication operator, the asterisk (*****), when writing an expression such as the previous one:

```
(3+4)*(4-8)
```

-28

Division is indicated by using the slash (**/**) character, so that the previous expression divided by 5 would be written as:

```
(3+4)*(4-8)/5
```

$$-\frac{28}{5}$$

Notice that *Mathematica* prints the answer as a simple fraction (in reduced terms), the same way that you might write it on an algebra test. In general, *Mathematica* always represents its evaluations as exactly as it can, unlike the way in which most calculators would.

Mathematica knows the meaning of parentheses **(** and **)** in evaluating expressions, just as they've been used algebraically for years. On the other hand, you may have also used other grouping symbols such as curly braces **{** and **}** or square brackets [and]. But in *Mathematica*, curly braces and square brackets have an entirely different meaning than parentheses and *cannot* be used to group expressions.

Exponentiation is a common operation for numbers and expressions, and *Mathematica* uses the caret symbol ("**^**") to represent it. For example, to compute 42 raised to the twenty seventh power, 42^{27}, you write:

```
42^27
```

672559700084060999217109284563873855680020 48

You now already see that *Mathematica* can easily handle very large numbers, something that handheld graphing calculators cannot do.

A final computation involves factorials. Recall that if n is a positive integer, the symbol

$n!$ represents the product of all positive integers less than or equal to n. In other words, we usually write $n! = n(n-1)(n-2)\ \ldots\ (3)(2)(1)$. *Mathematica* knows factorial notation, and is even able to compute very large values, such as:

```
32!
```

103331479663861449296666513375232000000000

1.1.2 In and Out Labeling

Mathematica numbers every input it receives and every output it produces starting with 1, from the beginning of a session. What you will see "on-screen" for some of the previous inputs and outputs will usually appear in a form such as that shown below. (The **In**[ ] and **Out**[ ] labels are supplied automatically by *Mathematica*; you do not enter them yourself.)

```
In[1] := (3+4)*(4-8)
```

```
Out[1] := −28
```

```
In[2] := (3+4)*(4-8)/5
```

$$\texttt{Out[2] :=} -\frac{28}{5}$$

```
In[3] := 42^27
```

```
Out[3] := 6725597000840609992171092845638738556800 2048
```

```
In[4] := 32!
```

```
Out[4] := 103331479663861449296666513375232000000000
```

From this point on in the text, we'll avoid the In- and Out-notation. We'll emphasize the start of an input with a leading filled triangle ("▶") and the start of an output (that is not otherwise a graphic) with a non-filled triangle ("▷"). Examples:

```
▶ (3+4)*(4-8)
```

▷ −28

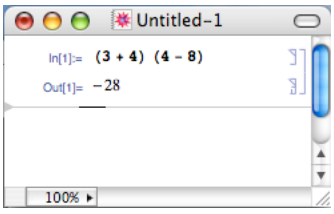
```
▶ (3+4)*(4-8)/5
```

$$\triangleright\ -\frac{28}{5}$$

1.1.3 Notebooks and Cells

The open window in which you have been working interactively is called a *notebook*, reminiscent of the way in which many students have done mathematics work over the years. Notebooks are the computer documents that you'll eventually print and/or save on your computer system. Notebooks can later be retrieved and edited, just as word processing or spreadsheet documents are managed on a computer.

What you saw on-screen for the first input you entered and the first output you received should have looked something like the following.



You'll notice brackets appearing at the right edge of the window. These delineate notebook *cells*, areas of the window which hold certain types of information. The **(3+4)(4-8)** that you typed was collected into an *input cell* by *Mathematica* and bracketed; the output −28 was collected and placed into an *output cell* and bracketed; and both cells were then bracketed together to indicate their in and out pairing.

As you start to work with *Mathematica*, you'll only be typing or editing in input cells, and you'll almost never alter the contents of an output cell.

You'll also see a horizontal, flashing cursor on a line across the window, which indicates that *Mathematica* awaits insertion of a new cell following the output cell. Typing anything will, by default, create a new input cell in which your typing will appear.

This organization of a notebook as a collection of cells is extremely useful in creating structured documents that, as you will see, eventually contain chapter and section headings, paragraphs that mix text, graphics, and mathematics, and *Mathematica* input and output. You'll learn more about this in later work.

1.1.4 *Mathematica* Watches Your Input

Have you noticed, yet? As you entered expressions into *Mathematica*, you'll see characters colored differently. Not only will the coloring change as you type, but you'll also see some flashing going on. So what's happening?

It's simple. *Mathematica* gives you clues as you type as to whether what you've got so far makes any sense, even before you try to evaluate it.

For example, in entering the expression **(3+4)(4-8)**, you'll see the following sequence:

- Type "**(**". *Mathematica* color-codes the left parenthesis in red. This indicates that *Mathematica* is now waiting for a matching right parenthesis.
- Type "**3**". By itself, the 3 looks OK and it appears in black.
- Type "**+**". The plus sign is suspicious if left alone and so is also colored red. Shouldn't something else follow?
- Type "**4**". Better! The $3 + 4$ now makes sense, although we're still waiting on that left parenthesis, aren't we?
- Type "**)**". *Mathematica* flashes back on the left parenthesis to show you what matched the right parenthesis. All is good now; everything's set in black.
- Type "**(**". *Mathematica* color-codes the left parenthesis in red and we're now waiting once again for a matching right parenthesis.
- And so it continues . . . until you've completed the expression, at which time all will be colored black.

All in all, this is a good thing. When an expression appears completely in black, at least it seems to make sense. When characters appear in red, something's amiss. (You'll see other colors come and go in the future as well.) Always watch the coloring!

You'll also see *Mathematica* take some proactive measures as you type. For example, if you type the characters "2" and "3" with a space between (2 – space – 3), *Mathematica* will reformat your input in-place when you type the 3 as

► 2 × 3

This happens because spaces in *Mathematica* implicitly mean multiplication. Here, *Mathematica* is assisting you so that you know what you've entered is not the number 23, but rather the product of 2 and 3.

It's possible to control *Mathematica*'s color coding and oversight rules more precisely, but

the default behavior should be quite acceptable for our purposes.

1.1.5 Operator Precedence and Parentheses

Mathematica regards some operators as having more importance than others in evaluating expressions. For example, the following expression produces what may be a non-intended result:

▶ `3+4*5`

▷ 23

Indeed, the multiplication of 4 with 5 is performed before the addition with 3, yielding an expression that is not the same as:

▶ `(3+4)*5`

▷ 35

This should be expected, because mathematicians have been writing expressions such as $\sin(3 + 2x)$ and $1 + x^2$ for years, without confusing them to be $\sin(5x)$ and $(1 + x)^2$, respectively. It will be important to use parentheses in writing expressions for *Mathematica* for clarity.

The evaluation of 3+4*5 as 3+(4*5) reflects the notion that multiplication has a higher priority or precedence than addition during evaluation. Similar rules of precedence must be established for expressions which may contain various mixtures of additions, subtractions, multiplications, divisions and exponentiations. Among the operators mentioned so far,

- the exponentiation operator has the highest precedence, or importance – for example, $1 + x^2$ is not $(1 + x)^2$;
- multiplication and division have lower precedence than exponentiation, neither has precedence over the other, but both have precedence over addition and subtraction – for example, $\sin(3 + 2x)$ is not $\sin(5x)$;
- addition and subtraction have the lowest precedence, but neither has precedence over the other; and
- operators of equal precedence are generally evaluated left-to-right, except for multiple exponentiation operators, which are evaluated right-to-left.

However, the most important idea to remember involving the precedence of operators is that the explicit use of parentheses always overrides any implicit priority or precedence of evaluation. We strongly recommend that you use parentheses in forming expressions to avoid any possible confusion, rather than to rely on built-in rules of operator precedence.

1.1.6 Calculator-Style Values

As seen above, *Mathematica* always produces an exact (symbolic) result for expressions whenever it can, rather than a numerical value as would be given by an electronic calculator. To convert an exact, symbolic expression into an approximate numerical value, the operator **N** is used:

▶ `N[(3+4)*(4-8)/5]`

▷ -5.6

The *Mathematica* operator **N** (the capital letter is significant) gives the numerical value of an exact expression. The square brackets are the notation for "applying" the operator **N** to the expression inside the square brackets.

The numerical value of any expression must be understood to be only an approximate

value, useful to some number of decimal places or significant digits. For example, if you enter a simple fraction such as $\frac{109}{98}$, you get an exact result:

```
▶ 109/98
```

$$\triangleright \frac{109}{98}$$

Evaluating **N** for this fraction produces a result whose (display) precision is given to 6 significant digits (this is not the same as requiring that the result be correct to 6 decimal places after the decimal point):

```
▶ N[109/98]
```

▷ 1.11224

You can specify that more decimal digits be shown as an approximation for an exact value – say 40 – by supplying a second argument for **N** (a comma separates the arguments):

```
▶ N[109/98,40]
```

▷ 1.112244897959183673469387755102040816327

When approximate numerical values must be written using "many" digit positions to either the right or left of the decimal point, *Mathematica* will begin to use standard, scientific notation to display approximate results, such as with the following value:

```
▶ N[1234567890]
```

▷ 1.23457×10^9

Approximate numerical values will be used for any expression that involves either the result of **N**, or any numbers explicitly written with a decimal point. For example, the fraction above is treated differently and automatically converted to an approximate numerical value if it is written with terms involving decimal points:

```
▶ 109.0/98.0
```

▷ 1.11224

Similarly, if the following value is entered directly using a decimal point, it is considered to be an approximate, numerical value and is reported using scientific notation:

```
▶ 0.000003492836
```

▷ 3.49284×10^{-6}

1.2 Formulas using Variables, Expressions, and Assignments

We start this section with a simple geometry formula.

> Given a triangle with sides of lengths a, b, and c, Heron's formula gives the area of the triangle by the formula
>
> $$A = \sqrt{s(s-a)(s-b)(s-c)},$$
>
> where s is the *semi-perimeter* s of the triangle, or one-half the perimeter of the triangle, i.e., $s = (a+b+c)/2$.

We will use variables that closely match the formula to show how to compute the area of a triangles whose sides have lengths 3, 4 and 6 as follows. Indeed, following the formula closely, we see that the variables a, b, and c in the formula should take on the values of 3, 4, and 6, respectively. In *Mathematica* we say to do exactly this:

```
▶ a = 3
```

▷ 3

```
▶ b = 4
```

▷ 4

```
▶ c = 6
```

▷ 6

Each of the three input cells above introduce a name into *Mathematica*'s vocabulary, and each name is being *assigned* a value. Each of the inputs is called *an assignment statement.* It has an equal sign ("=") in the middle as part of the syntax

a name of our choosing = an expression

Mathematica figures out the value of what's on the right side of the equal sign and then gives the result the name that appears on the left side of the equal sign. It also reports the value computed.

Essentially, we could paraphrase the sequence above by saying:

- "Let **a** be a name for the value 3" – and *Mathematica* reported "OK, the value is 3"
- "Let **b** be a name for the value 4" – and *Mathematica* reported "OK, the value is 4"
- "Let **c** be a name for the value 6" – and *Mathematica* reported "OK, the value is 6"

Mathematica will now remember these values whenever you use one of the names **a**, **b**, or **c** in an expression. For example, if you forget what **a** represents, you can ask for its value with

```
▶ a
```

▷ 3

Or if you wanted to know what the perimeter of the triangle was, you could evaluate

```
▶ a+b+c
```

▷ 13

For the area computation, we must next find the semi perimeter. We not only compute this with an expression, but we use an assignment statement to give the result a name so *Mathematica* can use it later.

```
▶ s = (a+b+c)/2
```

▷ $\frac{13}{2}$

Once again, we could paraphrase the assignment statement above by saying: "Let **s** be a name for the value of the expression (**a**+**b**+**c**)/2," and because each of **a**, **b**, and **c** has a value, this expression is really just $(3 + 4 + 6)/2$, which simplifies as 13/2.

Finally, it's time to get the area $\sqrt{s(s-a)(s-b)(s-c)}$ and we do this just by forming this expression in an input cell and evaluating it. (Note: The expression uses *Mathematica*'s square root function, named **Sqrt**. The argument of the function must appear within square brackets [and].)

```
▶ Sqrt[s(s-a)(s-b)(s-c)]
```

▷ $\frac{\sqrt{455}}{4}$

This final input could be phrased as "What is the value of the expression **Sqrt[s(s-a)(s-b)(s-c)]**," which *Mathematica* works backwards to be $\sqrt{\frac{13}{2}(\frac{13}{2}-3)(\frac{13}{2}-4)(\frac{13}{2}-6)}$, then it does the arithmetic and simplifies the result.

Question: Suppose we wanted to use the value $\frac{\sqrt{455}}{4}$ later. How could we ask *Mathematica* to remember it? The answer is simple: give the result a name.

But what name would you use? If you use **a** as a name for the value of the area, then *Mathematica* will forget about **a** originally representing a side length of 3 (a name in *Mathematica* can only have one value!).

Remember that the mathematical formula was written as $A = \sqrt{s(s-a)(s-b)(s-c)}$ and, in mathematics, we know that A and a mean different things. The same is true in *Mathematica*: the names **a** and **A** mean different things (names are said to be case-sensitive in *Mathematica*). We can use the name **A** for the value of the area.

One possibility then – which as we'll advise below is definitely not quite the right thing to do – would be to input the following (perhaps with a copy-and-paste of the expression from the last output cell so it will look nicer on screen):

```
▶ A = Sqrt[455]/4
```

Now *Mathematica* would know that whenever you used the name **A** in an expression, its value would be $\frac{\sqrt{455}}{4}$. However, you should never do anything like this, because as a general principle:

> *Never take an output and reenter it in a new input cell. Whenever you compute something and you expect to use the result of the computation again, give a name to the result with an assignment statement when you do the computation.*

What we should have done was not simply compute the area above and observe the result, but we should have used an assignment statement to give the result of the computation a name when we computed it. We should now go back, edit, and re-evaluate the input cell with the area computation:

```
▶ A = Sqrt[s(s-a)(s-b)(s-c)]
```

▷ $\frac{\sqrt{455}}{4}$

1.2.1 Variable Naming Rules

Can there be variable names in *Mathematica* other than single letters that are either lowercase or uppercase? Yes – otherwise, the range of variable names would be very limited. Variable names can be longer than one character and the characters are not restricted to being only letters. The rules are that variable names must:

- start with a letter, and
- can only be made up of letters and digits and a handful of other, special characters (which we will not show here).

Examples of variable names we could have used in the area computation above include: **area**, **side1**, **side2**, **side3** and **semiPerimeter**. And, as we mentioned above, *all names are case sensitive*. Thus **a** and **A** mean different things, as do **semiPerimeter** and **semiperimeter**.

1.2.2 Variable Naming Convention

One observation here is that there are literally thousands of names that *Mathematica* knows already, such as **Sqrt**, the name of the square root function. As a result, you'd probably not want to use a variable name yourself called **Sqrt**, because it would conflict with what *Mathematica* expects it to mean. And how would you know if you accidentally defined something *Mathematica* already knows about?

To solve this problem of your own variable names possibly getting mixed up with *Mathematica* names, the designers of *Mathematica* made sure that every name that *Mathematica* already knows *begins with a capital letter*. As a result, we strongly recommend that you use the following convention (as we throughout this text) when you choose variable names, to ensure that the names you define will not conflict with *Mathematica*'s pre-defined names

> *Every variable name you define should start with a lowercase letter.*

Thus, in Heron's Formula, we would never use **A** for the area on our own; we'd prefer to use, say, the name **area**. It just turns out that *Mathematica* does not use the name **A** itself, so we were lucky. (We could have made it to **B**, in fact, with no trouble, but *Mathematica* already has its own **C**, **D**, and **E**, and you already know about **N**.)

1.2.3 Review of the Area Computation

To summarize, here's the sequence of statements we recommend to compute the area of the triangle with side lengths 3, 4, and 6 using Heron's Formula.

```
▶ a = 3
```

▷ 3

```
▶ b = 4
```

▷ 4

```
▶ c = 6
```

▷ 6

```
▶ s = (a+b+c)/2
```

▷ $\frac{13}{2}$

```
▶ area = Sqrt[s(s-a)(s-b)(s-c)]
```

▷ $\frac{\sqrt{455}}{4}$

1.2.4 Input Conveniences

Suppose now we wanted to compute the area of a triangle having sides 4, 6, and 9. The easiest way is to redo the entire sequence of assignments used above. However, as a convenience and to conserve some space, we'll start by entering new values for the variables **a**, **b**, and **c** in the same input cell as follows:

```
▶ a=4; b=6; c=9;
```

Notice that no output cell was generated here, and that's because of the semicolon *at the end*. The other two semicolons serve to *separate* the three assignments. Indeed, semicolons are used in *Mathematica* for exactly these two purposes:

- they separate expressions and

- they suppress output when placed at the end of an input cell.

At this point, you might be tempted to think that the area is

```
▶ area = Sqrt[s(s-a)(s-b)(s-c)]
```

$$\triangleright \frac{5i\sqrt{13}}{4}$$

Clearly, this is not the right answer, since it is a complex number! The problem, of course, is that the **s** in the computation still represents the value 13/2. This is an important observation the first time you make it: when we defined **s** to be a name for the value of the expression (**a**+**b**+**c**)/2, its value was computed using the values of **a**, **b**, and **c** at the time, and not the new values we've assigned to these variable.

Indeed, the semi perimeter **s** must be redefined before a new computation of the area can be made. To conserve space, we'll again use a semicolon (the actual value of **s** is not necessarily of interest in this case and need not be printed), and then place the expression for the area on a second line of the *same input cell* (you *press the* ***Return*** *key* to advance to a new line within an input cell).

```
▶ s = (a+b+c)/2; (* computes new semi perimeter *)
  area = Sqrt[s(s-a)(s-b)(s-c)]
```

$$\triangleright \frac{\sqrt{1463}}{4}$$

Note that we've added an English phrase out the right of the first input line above, surrounded by the two-character combinations of **(*** and ***)**. This tells *Mathematica* that the phrase "*computes a new semi perimeter*" is a comment meant only as a helpful reminder for us of what we're computing, and that *Mathematica* should ignore it. We often add comments to *Mathematica* input cells whenever we must keep clear as to what we're computing.

The numerical approximation of this last area calculation, to the default precision of 6 significant digits, is given by:

```
▶ N[area]
```

▷ 9.5623

A quicker way to see the same result is is use the **N** symbol in what's called *post-fix syntax*, where is follows the argument instead of preceding it with matching square brackets:

```
▶ area//N
```

▷ 9.5623

1.2.5 Delayed Assignments (Advanced)

We'll now very subtlety rewrite the computation of semi perimeter and area above as follows:

```
▶ s := (a+b+c)/2; (* computes new semi perimeter *)
  area := Sqrt[s(s-a)(s-b)(s-c)]
```

What has changed? Notice that no output was generated, and it's easy to guess it probably has something to do with the use of the colon-equal combination (":=") instead of the equal sign alone. You'd be right, of course, but let's see what we've gained by doing this.

First, the values of **s** and **area** are still correct, should we ask for them:

▶ `s`

▷ $\frac{19}{2}$

▶ `area`

▷ $\frac{\sqrt{1463}}{4}$

But now, let's change the sides of the triangle to, say, 10, 13, and 19, with

▶ `a=10; b=13; c=19;`

Without re-inputting what you might think are formulas needed to redefine both **s** and **area**, we have:

▶ `s`

▷ 21

▶ `area`

▷ $4\sqrt{231}$

Why does this happen? This second method of defining both **s** and **area** is called a *delayed assignment* because of the use of the colon-equal syntax (":="). In short, it means that *Mathematica* will only figure out what **s** and **area** are when you ask for their values, and not when you define them.

Since **s** is defined in terms of **a**, **b**, and **c**, the value of **s** is determined only when you ask for it, based on the values of **a**, **b**, and **c** at the time you ask for it.

The same is true for the value of **area**: when you ask for its value, whatever are the values of **a**, **b**, **c**, and **s** at that time are used; and since **s** is determined based by the values of **a**, **b**, and **c**, it means that we have a whole new area computation anytime we need it just by changing the values of **a**, **b**, and **c**.

Indeed, a new area computation with side lengths 5, 7, and 11 looks like this:

▶ `a = 5; b = 7; c = 11; area (* note: no semicolon at end *)`

▷ $\frac{3\sqrt{299}}{4}$

So ... which should you use: delayed assignments (with ":=") or immediate assignments (with "=")? For most situations, unless there is a specific reason to take advantage of a delayed assignment, *you will want to use immediate assignment.* Using delayed assignment is certainly an important aspect of *Mathematica* syntax, but using it without a specific purpose tends to hide the dependencies of one variable upon others.

Nevertheless, when we meet up with functions shortly, you will see the benefits of delayed assignment.

1.3 Symbolic Formulas and Expressions

Mathematica can not only work with expressions that use variables and evaluate as numbers, but it can work with expressions in purely symbolic terms.

For example, we have the quadratic formula, where given a generic quadratic equation $ax^2 + bx + c = 0$, with $a \neq 0$, the roots of this equation are given by the Quadratic Formula to be

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Suppose we were to form this expression in *Mathematica* for the root that uses the plus sign (*see the subsection that follows about how you can use the Basic Math Assistant to enter this more naturally and much more cleanly*):

```
► (-b+Sqrt[b^2-4*a*c])/(2*a)
```

$$\triangleright \ \frac{1}{10}\left(-7+3i\sqrt{19}\right)$$

Wait! We were expecting just an expression with the variable names **a**, **b**, and **c**, but we got a number (and a complex one at that). The reason is simple: when we were last using **a**, **b**, and **c**, we had assigned values to these variables, and so the expression above was evaluated with those values. That's certainly not what we wanted.

To make *Mathematica* essentially forget about whatever it's been doing before with these variables, you use the **Clear** command, just before forming the expression. We prefer to keep everything in the same cell as the expression (this keeps them together both logically and physically); we'll define expressions for each of the roots separately (on separate lines within the same input cell); and, as you should guess, we'll give each expression a name (here: **root1** and **root2**) because we'll be using each below.

```
► Clear[a,b,c,root1,root2]
  root1 = (-b+Sqrt[b^2-4*a*c])/(2*a)
  root2 = (-b-Sqrt[b^2-4*a*c])/(2*a)
```

$$\triangleright \ \frac{\sqrt{b^2-4ac}-b}{2a}$$

$$\triangleright \ \frac{-\sqrt{b^2-4ac}-b}{2a}$$

Don't be confused about the fact that two outputs appeared as the result of a single input cell (complete input expressions entered on separate lines are suitably separated and each generates an output). Also, don't be confused about the fact that the output came out in a different order – it is correct as it stands, and the output format often depends on personal settings you've made within *Mathematica*.

The important observation, however, is that we wish to treat **a**, **b**, and **c** as symbolic quantities on their own. The **Clear** command guarantees that they are exactly that and have no meaning already assigned to them.[1]

> *Forgetting to appropriately **Clear** variable names is, by far, the number one source of problems encountered by students learning Mathematica.*

1.3.1 Substitution

Now we'll use the definition of **root1** to find the larger root of the equation $2x^2+5x-6=0$, you need only evaluate the **root1** expression by (the unfortunately-named operation of) "plugging in the values of a, b and c." You might think we should use assignments to set $a=2$, $b=5$, and $c=-6$, but that's not the best approach – doing so would remove the intended symbolic usage of **root1** and **root2**.

Yet this type of evaluation of an expression for a particular choice of parameters is such a common operation that *Mathematica* supplies a *substitution mechanism* that avoids the need to explicitly define variables. In syntax, you write:

[1] Including **root1** and **root2** in the **Clear** command would probably be omitted in practice, since the assignment statements that define them will replace previously assigned values or expressions. However, should there be a user-defined function named **root1** or **root2**, its definition would now be compromised – better safe than sorry, here.

```
▶ root1 /. { a→2, b→5, c→-6 }
```

$$\triangleright \frac{1}{4}\left(\sqrt{73}-5\right)$$

The compound symbol "**/.**" is the substitution symbol, made from the slash and period characters typed in sequence. Here, it means to evaluate the **root1** expression after replacing **a** with 2, **b** with 5, and **c** with −6, respectively.

The syntax defining the substitutions to be made requires that you specify a variable name, enter an "arrow" that is made up of the minus sign ("−") and greater than sign (">") together in sequence, and a value. (*You'll notice that Mathematica will rewrite the arrow much more nicely after you type it!*) These three *substitution rules* are separated by commas and enclosed in curly braces **{** and **}**.

The smaller root of the quadratic $2x^2 + 5x - 6 = 0$ could be written

```
▶ root1 /. { a→2, b→5, c→-6 }
```

$$\triangleright \frac{1}{4}\left(-5-\sqrt{73}\right)$$

It's easy to confuse the substitution mechanism with the notion of an assignment statement. Because we've used substitution, the variables **a**, **b**, and **c** have not been assigned any value:

```
▶ a
  b
  c
```

▷ a

▷ b

▷ c

As a result, we've therefore protected the symbolic integrity of the expressions named **root1** and **root2**:

```
▶ root1
```

$$\triangleright \frac{\sqrt{b^2-4ac}-b}{2a}$$

```
▶ root2
```

$$\triangleright \frac{-\sqrt{b^2-4ac}-b}{2a}$$

1.3.2 2-D Input Templates

Since expressions such as the one above that defines **root1** are becoming a little more complicated, now is a good time to introduce you to the Basic Math Assistant (in the Palettes Menu) for Typesetting. (See Figure 1.) It can help you form expressions in two-dimensions more easily using the mouse. This will let you see fractions, square roots, and exponents more clearly.

To define **root1** as the expression $\dfrac{-b+\sqrt{b^2-4ac}}{2a}$ in a new input cell (and assuming that we've executed a **Clear[a,b,c]**), we proceed as follows. (*Note how the input expression nicely reformats as you type.*)

- Type "**root1:=**".
- Click the fraction button . You'll see a template created in the input cell with the

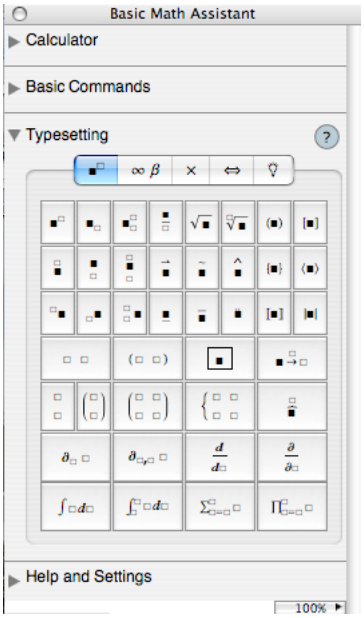


Figure 1: The Typesetting Palette of the Basic Math Assistant

cursor waiting for you to enter the numerator.

- Type "**-b+**". It will appear in the numerator of the fraction.
- Click the square root button $\boxed{\sqrt{\blacksquare}}$. The cursor is now located under the radical sign.
- Click the exponent button $\boxed{\blacksquare^{\blacksquare}}$. The cursor is now positioned in the base box of the exponent template.
- Type "**b**".
- Press the **Tab** key. This moves the cursor up to the exponent box.
- Type "**2**".
- Press the **right arrow** key to move to the right of the b^2 expression
- Type "**-4ac**".
- Press the **Tab** key. This should move the cursor to the denominator.
- Type "**2a**".
- With the expression now constructed, press the **Enter key** on the numeric keypad, or **Shift-Return** on the keyboard, to evaluate the expression.

Any expression you form using buttons and templates in a palette can be edited directly, should you make a mistake, just by clicking in the input expression at the proper level and editing as usual (although it may take a little practice to properly place the cursor in the exponent, for example). The Tab and arrow keys can also be used to navigate the expression.

Should you always use typesetting help from the Basic Math Assistant? We recommend that you do, since it's less likely that you'll make a syntax mistake, because you can see exactly what the expression is as you type. It may take a little thought however, to figure out the right sequence of buttons to click.

Further, by clicking on the button bar along the top of the Typesetting palette, more templates and buttons will be revealed, making it easy to enter common symbols such as the natural exponential base e, the complex constant i, the infinity symbol ∞, and Greek letters such as α. You'll find this very convenient when the time comes to use such symbols.

Finally, many of the templates and symbols available in the Typesetting palette have keyboard equivalents. For example, the square root template can be chosen by typing **Control-2**, and the fraction template can be chosen by typing **Control-/**. The α character can be typed directly using the sequence **Escape-a-Escape**. (Helpful aid: If you hover the mouse over any button in a palette, its tool tip will display the associated keystroke sequence.)

Given all these variations of how you can enter input, it's up to you to find how you can work with *Mathematica* most easily. Experiment!

Note. For this text, we will display input almost all the time without using a typesetting palette or control key sequence alternative. This is easier for preparing this publication, and the result is always more clear in print when you see the keystrokes. However, as a new *Mathematica* user, we *strongly recommend* that you use the typesetting palettes for entering expressions, primarily because you'll be able to see the expression on-screen more clearly and avoid mistakes cause by precedence rules.

1.4 Predefined Constants

Some common values mathematicians use are the transcendental numbers π and e. *Mathematica* already knows these constants under the names **Pi** and **E**, respectively.[2] It's easy to find the first fifty significant digits of π and e:

```
▶ N[Pi,50]
```

▷ 3.14159265358979323846264338327950288419716939937510

```
▶ N[E,50]
```

▷ 2.7182818284590452353602874713526624977572470937

There are other, predefined values in *Mathematica* (e.g., the Catalan and Fibonacci numbers) and all of them begin with an upper-case letter. The following table lists some commonly-used constants in *Mathematica*.

Constant	Value	Explanation	*Mathematica*
π	3.1415926...	Ratio of a circle's circumference to its diameter	**Pi**
e	2.71828...	Natural Exponential	**E**
i	$\sqrt{-1}$	Imaginary Number	**I**
$\frac{\pi}{180}$	.0174532	Degree to radian conversion multiplier	**Degree**
∞	∞	(positive) infinity	**Infinity**
$\frac{1+\sqrt{5}}{2}$	1.61803	Golden Ratio derived from Fibonacci Sequence	**GoldenRatio**

1.5 Built-in Functions

You've already seen the use of the **N** and **Sqrt** commands, or equivalently, functions. And just as any calculator has several dedicated buttons for commonly-used functions, *Mathematica* has an extensive list of built-in functions, including not only **N** (for numeric approximation) and **Sqrt** (for square root), but also **Log** (for the natural logarithm), **Sin**

[2] Each of these symbols is available in a concise, printed format in the Basic Math Assistant.

(for sine), and **ArcTan** (for arctangent). All *Mathematica* function names start with an upper-case letter and all use square brackets to designate their argument(s). For example,

▶ `Sin[Pi]` `(* sin π = 0 *)`

▷ 0

▶ `ArcTan[1]` `(* tan(π/4) = 1, so tan⁻¹(1) = π/4 *)`

▷ $\frac{\pi}{4}$

The following table lists some of the most commonly encountered functions.

Function(s)	Sample(s)	*Mathematica* Name(s)
natural logarithm	$\ln(x)$	**Log**
exponential	e^x	**Exp**
absolute value	$\|x\|$	**Abs**
square root	$\sqrt{x}$	**Sqrt**
trigonometric	$\sin(x)$, $\cos(x)$, etc.	**Sin**, **Cos**, **Tan**, **Cot**, **Sec**, **Csc**
inverse trigonometric	$\sin^{-1}(x)$, $\cos^{-1}(x)$, etc.	**ArcSin**, **ArcCos**, **ArcTan**, **ArcCot**, **ArcSec**, **ArcCsc**
hyperbolic	$\sinh(x)$, $\cosh(x)$, etc.	**Sinh**, **Cosh**, **Tanh**, **Coth**, **Sech**, **Csch**
inverse hyperbolic	$\sinh^{-1}(x)$, $\cosh^{-1}(x)$, etc.	**ArcSinh**, **ArcCosh**, **ArcTanh**, **ArcCoth**, **ArcSech**, **ArcCsch**

Among those listed above, note that the logarithm function can have a *second* argument.[3] To compute a logarithm to a base other than the standard base e, say to compute $\log_2(8)$, you write (putting the base of 2 first, and then the argument 8):

▶ `Log[2]//N` `(* ln 2 ≈ 0.693147 *)`

▷ 0.693147

▶ `Log[2,8]` `(* log₂(8) = 3 since 2³ = 8 *)`

▷ 3

We'll work with functions in more detail in the next Chapter, and we will show how you can define your own functions.

1.6 Final Details

1.6.1 Complex Numbers and Surd

Mathematica treats all expressions that you use over the complex numbers (we've seen at least one mention of this previously). In many cases, this is the mathematically correct thing to do – but when you're using the software mostly for Calculus or simple statistics, you may sometimes find that this strategy gets in your way.

[3] If more than one argument is required for the function, the arguments are separated by a comma (or commas).

Consider trying to find the cube root of -1, written as $\sqrt[3]{-1} = (-1)^{1/3}$. Symbolically, we enter:

```
▶ cuberoot = (-1)^(1/3)
```

▷ $\sqrt[3]{-1}$

It probably surprises you to see that *Mathematica* has not simplified this expression – but it is exact as it stands without simplification. Importantly, the result does have the property that its cube gives the value -1:

```
▶ cuberoot^3
```

▷ -1

Numerically, though, the value of **cuberoot** is not the -1 you expect:

```
▶ N[cuberoot]
```

▷ $0.5 + 0.866025i$

Mathematica has reported a numerical approximation for $\sqrt[3]{-1}$ that is symbolically $\frac{1}{2} + \frac{\sqrt{3}}{2}i$. If you understand complex numbers, you should check (*by hand!*) that $\left(\frac{1}{2} + \frac{\sqrt{3}}{2}i\right)^3 = -1$. In fact, there are two other complex numbers whose cube is -1, namely -1 or $\frac{1}{2} - \frac{\sqrt{3}}{2}i$, and this is why *Mathematica* was unwilling to simplify $\sqrt[3]{-1}$ because it would have to choose one of the three values.

When we ask for a numeric approximation, however, *Mathematica* is forced to choose one of the roots, and chooses the one that is sometimes called the principal cube root of -1.

Version 9 of *Mathematica* has introduced a new built-in function named **Surd** specifically to handle the problem of interpreting the n^{th} root of an expression (presumably a numeric expression) to be a real number. Thus we have

```
▶ Surd[ -1, 3]   (* the 3 is for the third root *)
```

▷ -1

Similarly, to handle the more general case of fractional exponents correctly for real numbers, the default behavior of *Mathematica* reports

```
▶ (-1)^(3/5) // N
```

▷ $-0.309017 + 0.951057i$

However, using Surd requires an integer root, and since $(-1)^{3/5} = \left((-1)^3\right)^{1/5}$ represents the fifth root of $(-1)^3$, we have

```
▶ Surd[ (-1)^3, 5]   (* the 5 is for the third root *)
```

▷ -1

1.6.2 When *Mathematica* Complains About Expressions

When you enter long expressions for *Mathematica* to evaluate, you'll certainly make (hopefully not too many) mistakes. It's easy to leave off a parenthesis, or perhaps add an extra one, especially when expressions get complicated.

If an expression is evaluated by *Mathematica* that doesn't make syntactic sense, *Mathematica* will already have alerted you with its coloring of the input expression, as we discussed earlier. If you hadn't noticed the syntax problem and evaluated the input, *Mathematica*'s response will be to highlight the offending text and highlight the input cell's bracket with a

little [+] to click on, so you can see an error message.

▶ `3*(4-5))+6`

▷ Syntax::bktmop: Expression "3*(4-5))" has no opening "(".

Here, the input expression's error has been correctly identified as having an extra right parenthesis in it with no matching left parenthesis. Parsing of the expression had to be terminated because not enough left-parentheses (looking left-to-right) had been scanned so that the second right parenthesis would be meaningful when it was encountered.

For other types of errors beyond the simple syntactic mismatching of parentheses, more cryptic messages may be issued, such as this:

▶ `5/0`

▷ Power::infy: Infinite expression $\frac{1}{0}$ encountered.
ComplexInfinity

Here, the expression we entered made syntactic sense – at least in form – but we tried to execute a division by zero. This may not be obvious from a result such as ComplexInfinity (which makes perfect sense if you understand the complex number system), so you must be prepared. *Mathematica*, or for that matter, any piece of software, will not be able to correctly detect or report all your errors in a way you'll understand, since it has no way of figuring out exactly what you're thinking.

You'll not understand every error message you see. Try not to get too frustrated while you're learning *Mathematica* and getting used to its environment.

1.6.3 Brackets and Curly Braces

One of the most basic of all *Mathematica* syntax constructions involves the use of curly braces **{** and **}**. We've seen only one instance of this so far when we introduced the notion of a *substitution list*, such as **{ x→3, y→4 }**. (This particular substitution list has two expressions, each being a substitution rule, and the rules are separated by a comma and enclosed in the curly braces.)

Many inputs to *Mathematica* that we'll use in the future will require list syntax, and you'll begin to see that outputs received from *Mathematica* will be lists, or even lists of lists. Carefully inspect any syntax that uses curly braces **{** and **}** and be sure you can identify the items that make up the list.

However, despite our limited use of lists so far, it's important to know that

> *Curly braces* **{** *and* **}** *can only be used with lists, and can never be used in place of parentheses to group terms in an expression.*

We'll see many more examples of lists shortly.

Similarly, you've seen the use of built-in functions such a N and Sqrt and Sin and ArcTan and Log – and with each appearance, you've see square brackets [and] used. For example (using the newfound list syntax mentioned above!):

▶ `{Log[E], Cos[Pi], Sqrt[9]}`

▷ $\{1, -1, 3\}$

A similar warning must be added here about when you can use square brackets [and]:

> *Square brackets* [*and*] *can only be used with functions to identify the arguments*

(inputs) of those functions, and can never be used in place of parentheses to group terms in an expression.

1.6.4 Approximate versus Exact

A basic principle of *Mathematica* is that every numerical expression is managed by default as an *exact* quantity, unless you ask for its *approximate numerical* value or introduce an approximate numerical into the expression.

Consider just the difference between the exact number π and its approximate numerical value. Evaluation of the sine function at π certainly produces the expected, exact result of zero:

```
x = Pi; (* x references an exact quantity *)
Sin[x]
```

▷ 0

However, if we replace π by its approximate numerical value, the result is no longer zero:

```
x = Pi//N; (* x references an approximate numerical value *)
Sin[x]
```

▷ 1.22465×10^{-16}

Because **x** does not represent an exact quantity, *Mathematica* cannot produce an exact result. Of course, the result is very close to zero (the result is 0.00000000000000122465), functioning now as an approximate numerical value for zero, and this may still be acceptable in many practical situations.

The same situation results whenever approximate numerical values are introduced into a computation. For example, it is the case that *Mathematica* treats the quotient $\frac{275}{55} = 5$ exactly, and hence we have

```
x = 275/55 Pi; (* x references the exact quantity 5π *)
Sin[x]
```

▷ 0

However, should the expression defining **x** contain *any* term that represents an approximate numerical value – such as a decimal number with a decimal point – everything changes. In particular, the division $\frac{2.75}{0.55} = 5.0$ contains decimal points and is thus treated as having an approximate numerical value

```
x = 2.75/0.55 Pi; (* x references an approximate numerical value *)
Sin[x]
```

▷ 6.12323×10^{-16}

Neither of the outputs above would significantly impact a numerical computation; but it's not very difficult to see how the use or (abuse) of approximate numerical values can degrade a computation. Be sure you understand the difference in these two computations:

```
x = (123456789^3)*Pi; (* 123456789^3 × π is an exact quantity *)
Sin[x]
```

▷ 0

```
x = (123456789^3)*Pi//N; (* x has an approximate numerical value *)
Sin[x]
```

▷ −0.955171

The last result is far from correct – we perhaps expect it to be 0 exactly – yet the value produced is meaningless and in no way could be considered anywhere even "close" to 0, especially since values of the sine function are confined to the interval $[-1, 1]$.

This same type of problem appears on a calculator, which can represent values only to so many digits; and when computation after computation is made with quantities having limited precision, the result can be meaningless. The subject of analyzing how loss of precision affects computations is beyond what we can describe here, so we state:

> *Unless you have a reason to do otherwise, always form expressions in Mathematica using exact terms, and avoid the use of N or decimal points within expressions.*

1.6.5 List of Symbols

The following table contains most of the symbols we introduced in this Chapter.

Symbol(s)	Use or Meaning
+	Addition
−	Subtraction
* (or a space)	Multiplication
/	Division
^	Exponentiation
!	Factorial
=	Assignment
:=	Delayed Assignment
%	The previous result
;	Separation & Output Suppression
/.	Substitution
(and)	Used for grouping terms in an expression
{ and }	Delimiters for (substitution) lists
(* and *)	Delimiters for comments
[and]	Delimiters for function arguments
,	Separator within (substitution) lists
->	Substitution rule
?	Information (or Help) Operator

1.7 Exercises

1. Order the following five values from smallest to largest:

$$7.149\pi,\ \pi^e,\ e^\pi,\ \left(\frac{1+\sqrt{5}}{2}\right)^{11\pi/5},\ (2+e)^{\pi-1}$$

2. Enter each of the following constant expressions in *Mathematica*, and give it a name (e.g., **expr**) using the equal sign. After verifying that *Mathematica* responds with the correct expression symbolically, use the syntax "**N[expr]**" to evaluate the expression numerically.

(a) π^{2+e}

(b) $\dfrac{\sqrt{e-1}}{\pi-2}$

(c) $\dfrac{\sin(1)-6}{5-\ln 2}$

(d) $\left|\sinh\left(\dfrac{\pi-5}{\log_3 37}\right)\right|$

(e) $\dfrac{\tanh^{-1}(1/2)}{2-\tanh(3)}$

(f) $e^{\sqrt{5}}$

(g) $\sqrt[3]{53}+e^{\pi^2-1}$

3. Enter each of the following symbolic expressions in *Mathematica*, and give the expression a name (e.g., **expr**) using the equal sign. After verifying that *Mathematica* responds with the correct expression symbolically, use the substitution syntax "**expr /. { x→ }**" to evaluate it symbolically at the indicated value, and then execute "**N[%]**" to evaluate the result numerically.

(a) $\sqrt{\dfrac{e^{\sin(1+x)}}{1+\cos x}}$, when $x=0$.

(b) e^{x^3}, when $x=2$.

(c) $\dfrac{1+\sqrt{16-x^2}}{2x}$, when $x=3$.

(d) $|4\cos x+\pi|$, when $x=\pi$.

4. Suppose that ℓ is the line through the points (x_1,y_1) and (x_2,y_2).

(a) The distance from (x_1,y_1) to (x_2,y_2) is given by

$$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$$

Form the expression symbolically, and evaluate it using substitution when $(x_1,y_1)=(1,2)$ and $(x_2,y_2)=(4,6)$.

(b) The slope of the line ℓ is given by $\dfrac{y_2-y_1}{x_2-x_1}$. Form the expression symbolically, and name the expression slope. Evaluate slope using substitution when $(x_1,y_1)=(1,2)$ and $(x_2,y_2)=(4,6)$.

(c) Form the expression $-\frac{4}{3}x+y-\frac{2}{3}$ symbolically. Show that zero is obtained when substituting $(x,y)=(1,2)$ and $(x,y)=(4,6)$ (hence the equation for the line through these two points must be $-\frac{4}{3}x+y-\frac{2}{3}$).

(d) The distance from a line with equation $ax+by+c=0$ to a point (p,q) not on the line is given by the expression $\dfrac{|ap+bq+c|}{\sqrt{a^2+b^2}}$. Form this expression symbolically, and evaluate it exactly and numerically for the line of part **c** when $(p,q)=(\pi+2,e-1)$.

5. If you finance a loan with a total amount of *loan*, at an annual interest rate of *interest*, for a period of *months*, then the monthly payment on the loan is given by the formula:

$$payment=(loan)\frac{1-r}{(1-r^{months})\cdot r},\ \text{where } r=\frac{12}{12+interest}$$

(a) Form symbolic expressions for the terms r (in terms of interest) and payment (in terms of *loan*, r and *months*) using delayed assignment, and evaluate each symbolically to check that they are entered correctly.

(b) Using these formulae, suppose a new automobile is financed with a loan of $21,000 over a five year (60 month) period at 9% interest. Find the monthly payment for the loan.

(c) Using these formulas, consider the purchase of a home costing $270,000 for which the buyer will make a down payment of $35,000, and finance the loan for 30 years.

For each of the interest rates 5.0%, 5.5%, 6.0%, 6.5%, and 7.0%, find the amount of interest paid on the loan over its lifetime.

6. Find the area of the triangle that has vertices at the points $(0, 0)$, $(5, 1)$ and $(3, 2)$.

7. Find the area of the quadrilateral that has vertices at the points $(0, 0)$, $(6, -1)$, $(7, 5)$ and $(-2, 4)$. (Suggestion: break the quadrilateral into two triangles.)

8. Stirling's formula provides an approximation for $n!$, in the form

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

Formulate this expression symbolically, and evaluate it numerically using substitution, for values of $n = 20$, 40, 80 and 160. Compare your results with 20!, 40!, 80! and 160! Are the values close?

9. Evaluate the quantity $\dfrac{\left|\sqrt{2\pi n}\left(\frac{n}{e}\right)^n - n!\right|}{n!}$ for each of the values in the previous exercise. Why it is appropriate to say that "Stirling's Formula approximates $n!$ with an error on the order of $\frac{1}{n}$."?

10. Find the smallest positive integer n that satisfies $n! > n^{10} + n^6 + 10$.

11. The expression $4\left(\tan^{-1}(1/4) + \tan^{-1}(3/5)\right)$ is a well-known mathematical constant. Which constant is it?

12. We propose that $n(\sqrt[n]{n} - 1)$ is a good approximation for $\ln(n)$. Evaluate each expression for $n = 20$, 40, 80 and 160, and decide whether this is true. Comment on what is you think is meant by the phrase "a good approximation".

13. Find the integral power of 3 that is closest to each of one million, one billion and one trillion.

2 Functions

We learned in the previous Chapter that *Mathematica* already has many built-in functions (e.g., the trigonometric, logarithmic, exponential, and hyperbolic functions). Many others are already defined as well for specialized applications (e.g., the Bessel functions), although we'll see only a few in the sequel. In this Chapter, we introduce the syntax you need to define your own functions.

2.1 Defining Functions

A function f of a single, real variable is defined by stating its *rule of assignment* for each element in its *domain*.

Example. The function that you normally write as $f(x) = x^2$ is the function which, for each real number x, assigns the square of the number. The *domain* of this function is all real numbers (these are numbers which can be *input* to the function) and the rule of assignment is "square the input" (which determines the *output* from a given input).

Notice that the role of the symbol x in the formula-like definition $f(x) = x^2$ above is that of a place-holder; you could equally well write $f(t) = t^2$ to define exactly the same function.

In *Mathematica*, you define the square function above with the following input. (*Note. You'll receive no output response for this, and you'll also see a new coloring scheme as each of* ***f*** *and* ***x*** *are typed.*)

```
▶ Clear[f] (* be sure f conflicts with no other names *)
  f[x_] := x^2 (* use the Basic Math Assistant to typeset x^2 *)
```

In the syntax of the definition above, notice that we've once again wrapped in the use of **Clear** with the definition we're making for f.

> *Use of* ***Clear*** *was emphasized in the previous Chapter, and it becomes all the more important to include it when defining functions.*

Next you see square brackets [and], which designate exactly that **f** will represent a function. What appears between the square brackets define what the inputs to **f** will look like – in this case, that there will be a single input (which we probably intend to be a number, but it can more generally be any single expression).

The symbol combination **x_** appearing on the left side of the expression is typed as the two-character sequence "**x**" and then the underscore character "**_**". This syntax says exactly that "**x**" will be only a place-holder to represent the single variable of the function in the rule of assignment that is to follow, and guarantees that it will not conflict with any other symbol **x** *Mathematica* might know about at the time of definition.[4]

The use of *Mathematica*'s delayed assignment operator "**:=**" in the definition means that only the *symbolic expression* **x^2** that appears on its right side is to be associated with the symbol **f[x_]**, not that the *value* of the expression **x^2** is to be assigned to the symbol **f[x_]**. In short, the syntax describes the rule of assignment as "square the input," without confusing x to mean the value of some other expression we might have at the time.[5]

[4] The expression **x_** used in this definition forms what is known as a *replacement pattern* or *pattern variable* in *Mathematica*.

[5] The subtlety of this statement – and how using "**:=**" differs from what you think should probably be just "**=**" – appeared briefly in the previous Chapter and will become more clear in the future. But it is almost always the case that you will use the "**:=**" combination when defining functions, unless you have specific reasons not to use it.

If you want to see whether *Mathematica*'s recorded your definition of f correctly, the best way to ask is with

```
▶ ?f
```

```
▷ Global`f
  f[x_] := x^2
```

This is somewhat of a new statement for us – we use the question mark **?** to ask *Mathematica* what it knows about the name that follows the question mark. We see above that *Mathematica* reports back the definition, and includes more info about the fact the **f** is known in its global context. *Until you gain proficiency with Mathematica and need to know more about contexts, we'll simply state that every name you introduce in Mathematica will live in the global context, and you should otherwise ignore the context qualifier.*

A simple evaluation of the mathematical expression $f(3) = 3^2 = 9$ is given with:

```
▶ f[3]   (* functions use [ and ] and not parentheses! *)
```

```
▷ 9
```

Here, *Mathematica* has evaluated the expression **f[3]** by substituting for it the expression **3^2**, producing the simplification of 9. Similarly, evaluation of $f(-4) = (-4)^2 = 16$ is reproduced by *Mathematica* as:

```
▶ f[-4]
```

```
▷ 16
```

Again, *Mathematica* evaluated the expression **f[-4]** by substituting for it the expression $(-4)^2$, producing the simplification of 16. Indeed, in the future, any expression that *Mathematica* evaluates of the form "**f**[*something*]" will be replaced by the expression "*something*^**2**" and then evaluated.

This principle applies even for symbolic expressions and other variables. If a variable **a** has already been defined, then evaluating f at **a** produces the correct result:

```
▶ a = 5;
  f[a]
```

```
▷ 25
```

In this case, the expression **f[a]** is replaced by the expression **f[5]**, which in turn is replaced by the expression 5^2, yielding the result 25. Symbolically, this sequence of replacements could be recorded by the sequence

$$f[a] \to f[5] \to 5^2 \to 25$$

If **a** has not yet been previously defined to have a particular value as a variable, but only represents an arbitrary symbol, evaluation of **f[a]** produces a correct symbolic result:

```
▶ Clear[a]; f[a]
```

```
▷ a^2
```

In fact, any expression passed as the argument to f will produce a proper evaluation:

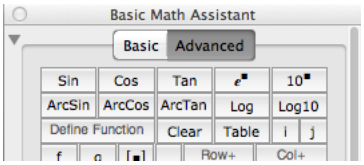
```
▶ Clear[b,c]; f[b-c]
```

```
▷ (b - c)^2
```

2.1.1 Using the Function Template

The syntax of function definition is perhaps the most important notational aspect of using *Mathematica* to do mathematics, yet more errors in *Mathematica* are probably introduced because of incorrect use of the syntax.

For this reason, the Advanced tab of the Basic Math Assistant's calculator provides an automatic template that you can fill in with the proper syntax. Look first for the button marked "Define Function" (third row at the left).



Clicking the Define Function button will introduce the following arrangement into the current input cell, creating a new input cell if necessary.

name [*var* _] := *expr*

Start by typing **f** (which appears immediately in the *name box* of the template).

f [*var* _] := *expr*

Now press the **tab key** to advance to the *var box* of the template and type the variable **x**.

f[**x**_] := *expr*

Pressing the **tab key** one more time will move you to the *expr box* of the template, where you can now enter the rule of assignment for the function (*not shown here*).

When you use the template, notice that you do not have to enter any of the underscore character ("_"), the square braces [and], or the colon-equal (":="). You can focus on

- picking a name for the function
- pressing the tab key and choosing a name for the variable
- and pressing the tab key and entering the expression for the rule of assignment.

2.1.2 Composition of Functions

Mathematically, if f and g are (suitable) functions, we can form new functions h and k by defining $h(x) = f(g(x))$ and $k(x) = g(f(x))$, called the **compositions** of f and g. Understanding compositions is an important aspect in Calculus and, the good news is that *Mathematica* follows a natural syntax.

Example. If $f(x) = 2x + 5$ and $g(x) = 3x$, then we have

$$f(g(x)) = f(3x) = 2(3x) + 5 = 6x + 5$$

and

$$g(f(x)) = g(2x + 5) = 3(2x + 5) = 6x + 15$$

Not surprisingly, these two composite functions are different, since *order matters in a composition*. Indeed, the operation of $f(g(x))$ of "multiply by 3" and then "multiply the result by 2 and add 5" does not result in the same computation of $g(f(x))$, which is "multiply by 2 and add 5" and then "multiply the result by 3."

In *Mathematica* we have what should be the obvious syntax for composition

```
► Clear[f,g]   (* we always precede with a Clear *)
  f[x_] := 2x+5   (* and we keep all together in one input cell *)
  g[x_] := 3x
```

```
► Clear[x]   (* we're thinking symbolically in x *)
  f[g[x]]
```

▷ $6x + 5$

```
► g[f[x]] // Expand   (* expands the result, else we get 3(2x+5) *)
```

▷ $6x + 15$

2.1.3 More than One Variable

Beginning students in mathematics often think of functions only in the form $f(x)$, but most practical uses of functions involve more than a single variable.

Example. If the automobile has traveled m miles, in the span of t minutes, then it has traveled for $\frac{t}{60}$ hours, and its average speed in miles per hour will given by the expression $\frac{m}{(t/60)} = \frac{60m}{t}$. It is sensible to define the average speed function "$f(m,t) = \frac{60m}{t}$."

More complex examples might be encountered in modeling economic situations. The growth rate of a national economy might depend upon several factors: the unemployment rate u, worker productivity p, trade deficit t, and prevailing interest rate i. Such a function would be modeled in the form $f(u,p,t,i)$, where the computation of the economic growth would depend on and be determined by these factors.

In *Mathematica*, the "speed" function depending upon mileage covered and time taken described above would be given by:

```
► Clear[speed];
  speed[m_,t_] := m/(t/60)
```

The function above has two arguments, each of which appears as a pattern variable on the left inside the square brackets, and they are separated by a comma. The definition of the distance computation in terms m and t appears on the right side of the delayed assignment **:=**.

For example, if a distance of 45 miles is traveled in 30 minutes, the average speed in miles per hour is given by:

```
► speed[45,30]
```

▷ 90

Traveling 60 miles in 50 minutes produces an average speed in miles per hour of:

```
► speed[60,50]
```

▷ 72

Example. A function that gives one of the (assumed) real roots of a quadratic polynomial $p(x) = ax^2 + bx + c$ can be defined in terms of the coefficients a, b, and c. Indeed, it makes sense to think of a root function "$f(a,b,c)$," whose value is given by the quadratic formula

$\dfrac{-b+\sqrt{b^2-4ac}}{2a}$:

```
▶ Clear[f]
  f[a_,b_,c_] := (-b+Sqrt[b^2-4a c])/(2a) (* Basic Math Assistant used *)
```

For example, to compute one root of the polynomial $x^2 - 3x - 10$, you can evaluate:

```
▶ f[1,-3,-10]
```

▷ 5

2.1.4 Piecewise Defined Functions

Not every useful function can be easily defined by a single formula or closed expression. The simplest example of such a function is a step function such as:

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

A function of this type is said to be *defined piecewise*, in that the domain of the function (here, all real numbers) is split into disjoint *pieces* or *branches*, and a separate rule of assignment is used for each branch.

We can define f in *Mathematica* using the **Piecewise** command with the syntax:

```
▶ Clear[f]
  f[x_] := Piecewise[ { {1,x>=0} , {0,x<0} } ]
```

In this form of the **Piecewise** command, we see the usual square brackets [and] on the right, and only a single argument that is a *list* of the branch definitions enclosed by curly braces **{** and **}**. Inside these curly braces, we see two additional pairs of curly braces, one for each branch of the function, separated by commas. Each of the pairs that defines a branch consists of a *value* for the function and the *logical condition* that must be satisfied to use this branch.

We must digress (only quickly) to show how one forms logical conditions used in defining the branches. The ≥ sign is entered by typing the two-character sequence >= (which you'll see *Mathematica* reformat as ≥ as you type). The following table shows how other logical conditions should be entered.

Logical Connective	Meaning	In *Mathematica*
$<$	Less than	<
$\leq$	Less than or Equal To	<=
$>$	Greater Than	>
$\geq$	Greater Than or Equal To	>=
$=$	Equal To	==
$\neq$	Not Equal To	!=
and	And	&&
or	Or	||

The definition of the function f could also have been written using the alternate syntax

```
▶ Clear[f]
  f[x_] := Piecewise[{ {1,x>=0} }, 0 ]
```

Here, after entering just a single condition that defines the first branch inside curly braces as before, we add a second argument following a comma to define whatever will be the default value of the function, i.e., the value to be used if none of the conditions defining the branches is satisfied. That is, we have the definition

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

More branches can, of course, be included when defining a piecewise function. Consider, for example, the following

$$h(x) = \begin{cases} 1 - x, & \text{if } x < 0 \\ x^2, & \text{if } 0 \leq x \leq 1 \\ x + 2, & \text{if } x > 1 \end{cases}$$

The corresponding input for *Mathematica* would be

```
▶ h[x_] := Piecewise[{ {1-x,x<0}, {x^2,0<=x<=1}, {x+2,x>1} }]
```

(Note that *Mathematica* even makes sense of the compound condition for the specification of h in the interval $[0, 1]$.) Because the function definition has started to become syntactically complex, it will help you if you visually format a **Piecewise** expression by explicitly pairing the conditions and values, one to a line, such as with:

```
▶ h[x_] := Piecewise[{
        {1-x,x<0},
        {x^2,0<=x<=1},
        {x+2,x>1}
        }]
```

2.1.5 Recursively Defined Functions (Advanced)

A recursive function is one that is defined in terms of itself.

Example. The most common, recursive function with which you're already familiar is the factorial function, defined for every non-negative integer n to be

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \times (n-1)! & \text{if } n > 0 \end{cases}$$

Thus, 0! is 1, 1! is $1 \times 0! = 1 \times 1 = 1$, and even at this level, the computation of 2! according to the definition becomes cumbersome, since

$$2! = 2 \times 1! = 2 \times (1 \times 0!) = 2 \times (1 \times 1) = 2 \times 1 = 2$$

And on it goes.

Example. Another commonly-known example is the definition of the Fibonacci numbers $c_0, c_1, c_2, \ldots$. Here, we define, for each non-negative index n, the value

$$c_n = \begin{cases} 1 & \text{if } n = 0, 1 \\ c_{n-1} + c_{n-2} & \text{if } n \geq 2 \end{cases}$$

The first three Fibonacci numbers are $c_0 = 1$, $c_1 = 1$, $c_2 = c_1 + c_0 = 1 + 1 = 2$, and these are the start of the sequence $1, 1, 2, 3, 5, 8, 13, 21, \ldots$, where each term is the sum of two previous terms.

Mathematica can handle each of these definitions easily. Indeed, the factorial function could be written as

```
▶ Clear[f]
  f[0] = 1;
  f[n_] := n * f[n-1]
```

The Fibonacci numbers could be written as

```
▶ Clear[c]
  c[0] = 1; c[1] = 1;
  c[n_] := c[n-1] + c[n-2]
```

We won't go any farther into recursively-defined functions at this point, since, as simple as they may seem, they require more attention to detail than directly computable functions such as $f(x) = x^2$. For example, the following input would drive *Mathematica* crazy (and do you see why?)

```
▶ f[-1]
```

Nevertheless, should it become convenient to use a recursive definition in the future, we'll not shy away from it.

2.1.6 Modules (Advanced)

Function definitions may sometimes require more than a direct, one-line calculation. This may be the case either because of the need to use intermediate calculations, or even the complexity of the function definition itself.

A simple example of the complexity problem is the calculation of the area of a triangle using Heron's formula. Clearly, it would be convenient to write an area function of three variables in the form:

```
▶ area[a_,b_,c_] := an expression to be evaluated
```

However, the area calculation was conveniently broken down into the two-step process of first computing the semi-perimeter $s = (a + b + c)/2$, in terms of the three sides a, b, and c of a triangle; and then forming the expression $\sqrt{s(s-a)(s-b)(s-c)}$ to give its area. Essentially, two separate evaluations must be performed, with the second of these giving the value of the area function.

You could, of course, write out the expression under the radical in this calculation by eliminating the semi-perimeter s with

$$\sqrt{\left(\frac{a+b+c}{2}\right)\left(\frac{a+b+c}{2}-a\right)\left(\frac{a+b+c}{2}-b\right)\left(\frac{a+b+c}{2}-c\right)}$$

This would be cumbersome at best to enter in *Mathematica*, since the necessary syntax would require many characters written over multiple lines, and is likely to have mistyped characters or even mismatched parentheses.

Additionally, the simplicity of the area calculation as $\sqrt{s(s-a)(s-b)(s-c)}$, where $s = (a + b + c)/2$, has been lost.

To handle this situation, we will use the **Module** operator. It allows us to use intermediate variables and results (such as the computation of the semi-perimeter s) and collect them together to produce a single result.

The definition of the area function, using the **Module** syntax, will be written as follows:

```
▶ Clear[area]
  area[a_,b_,c_] := Module[
      {s},
      s = (a+b+c)/2;
      Sqrt[s(s-a)(s-b)(s-c)]
    ]
```

We start to unravel the syntax above as follows. At the outer level, the result of the area calculation shown after the **:=** has the form

```
▶ Module[ { variable } , statement ; statement ]
```

The curly braces that appear first contain the name(s) of any intermediate variable(s) used in the calculation. The **area** function above will use the intermediate variable **s** to compute the semiperimeter. Importantly, though, the symbol **s** will be distinct from any other symbol **s** that might be known by *Mathematica*. Further, once the area computation is completed, the symbol **s** used here will simply no longer exist.

Following the curly braces is a comma, and following the comma are the two statements that in sequence perform the area calculation. These are separated by a semicolon (and note in particular that no semicolon follows the last statement). The value of the last computation made becomes the value of the function.

2.2 Final Details

2.2.1 Lists and Curly Braces Revisited

You've already seen situations where *Mathematica* syntax requires curly braces **{** and **}**. Examples would include

- A substitution list such as **{ x→3, y→4 }** that we used in the previous Chapter
- A branch of a piecewise-defined function such as **{x+1, x>0}**
- A list of branches of a piecewise-defined function such as **{** *branch #1*, *branch #2*, ... **}**
- The first argument for a **Module** that defines intermediate variables such as **{s}**

In each case, one or more items were contained inside the curly braces, with multiple items being separated by commas. This is one of the most basic of all *Mathematica* syntax constructions and is called a *list*.

Many inputs to *Mathematica* require list syntax and outputs received from *Mathematica* often are in the form of a list. For example, the graphing operators you'll see in the next Chapters use list syntax quite a bit, so be prepared. Carefully inspect any syntax that uses curly braces **{** and **}** and be sure you can identify the items that make up the list.

Finally, the *only* use of curly braces in *Mathematica* is for *lists*. Curly braces are never used for grouping terms in expressions. In the sequel, we'll describe list syntax intricacies only as needed; Appendix A has more complete information on list structures.

2.2.2 Complex Numbers

As we mentioned in the previous Chapter, *Mathematica* works in the complex number system, while most beginning users of *Mathematica* assume that everything in life is a real number. As a result, you'll occasionally think that a function definition must be incorrect when, in fact, the definition is correct in the complex sense – it just appears to not work in the real number system.

Example. If we define $f(x) = x^{2/3}$, then mathematically we have $f(-8) = 4$, since $(-8)^{2/3} = \sqrt[3]{(-8)^2} = \sqrt[3]{64} = 4$.

In *Mathematica*, the function definition below initially appears to work.

```
▶ Clear[f]
  f[x_] := x^(2/3) (* Basic Math Assistant used *)
▶ f[8]  (* 8 squared is 64, cube root is 4 *)
```

▷ 4

But we next see

```
▶ f[-8]  (* -8 squared is 64, cube root is 4 *)
```

▷ $4(-1)^{2/3}$

The natural response is the think that maybe there's something wrong with the function definition – but there's nothing wrong with it at all *in the complex sense* (review the section on complex numbers in the previous Chapter on this point about the meaning of $(-1)^{2/3}$).

Rather, if you're interested in the *real-valued* function $f(x) = x^{2/3}$, or any function whose definition involves the use of a *rational exponent*, remember to incorporate the **Surd** function in its definition. The proper definition in this case should be

```
▶ Clear[f]
  f[x_] := Surd[x^2,3]
```

2.3 Exercises

1. The area of a circle is given by the formula $A = \pi r^2$, where r is the radius of the circle. Define the area function in *Mathematica*, and demonstrate its use to compute the areas of the circles whose radii are 4, 9, and 3.14.

2. Define $f(x) = 3x + 19$ and $g(x) = \frac{1}{3}(x - 19)$. Use *Mathematica* to show (algebraically) that these are inverse functions – i.e., that $f(g(x)) = x$ and $g(f(x)) = x$.

3. Let m and b be constants, with $m \neq 0$.[6] Define $f(x) = mx + b$ and $g(x) = \frac{1}{m}(x - b)$. Use *Mathematica* to show (algebraically) that these are inverse functions – i.e., that $f(g(x)) = x$ and $g(f(x)) = x$.

4. An ellipse with a semi-major axis of length a and a semi-minor axis of length b has a perimeter of approximately

$$f(a, b) = 2\pi\sqrt{\frac{a^2 + b^2}{2}}$$

Define this function in *Mathematica*, and use it to approximate the perimeter of an ellipse having a major axis of length 5 and a minor axis of length 4.

5. Consider a right, circular cone with base radius r and height h. Define each of the following functions in *Mathematica*:

(a) The volume function $V = f(r, h) = \frac{1}{3}\pi r^2 h$, and
(b) The surface area function $S = g(r, h) = \pi r^2 + \pi r\sqrt{r^2 + h^2}$.

Evaluate each of these quantities for the circular cone having radius 4 and height 11.

[6] You should not try to tell *Mathematica* explicitly to "assume that m is not zero" to do this computation. Just do the computation. Yes, we mathematicians worry about the statement that follows when $m = 0$ when we're writing mathematics; but *Mathematica* will continue to carry out computations for you and then let you worry on your own about special cases such as $m = 0$.

6. If a triangle has sides of length a, b and c, the radius of its circumscribed circle (the circle through its three vertices) is $r = f(a,b,c) = \dfrac{abc}{4k}$, where k is the area of the triangle (the area can be computed using Heron's formula).

Write a definition of this radius function in *Mathematica*, and use it to find the radius of the circumscribed circle for a triangle having sides 4, 7 and 10. (Suggestion: use a **Module** construction.)

7. The length of the parabolic arc from $(0,0)$ to (x,y) along the parabola $y^2 = x$ is given by the arc length function f valid for points $(x,y) = (x,\sqrt{x})$ on the curve in the first quadrant by writing

$$f(x,y) = \frac{u + \dfrac{y^2 \ln\left(\dfrac{2x+u}{y}\right)}{2x}}{2} \text{ where } u = \sqrt{4x^2 + y^2}$$

(a) Write a function of two variables $f(x,y)$ that computes the length of the arc from $(0,0)$ to (x,y), according to the formula above. Use a **Module** construct in your definition.

(b) Use the function you defined in part **a** to evaluate the arc length at the points $(4,2)$ and $(9,3)$.

8. Bus fares on the local transit system are determined according to the rider's age. Children 6 years of age and less ride for free; senior citizens (age 65 or higher) ride for 10 cents, and all others ride for 50 cents. Write a *Mathematica* function to represent this fare structure, and test its value for riders 5, 36, and 71 years old.

9. The function

$$g(x) = \begin{cases} 2x, & 0 \le x \le \frac{1}{2} \\ 2x - 1, & \frac{1}{2} \le x \le 1 \end{cases}$$

defined on $[0,1]$, is called the *baker's transformation*, because of the following interpretation. Consider a gob of dough of length 1 that is to be kneaded in a certain fashion. If a point of the dough is a distance x from the end, with $0 < x < 1$, let $g(x)$ represents its position after kneading the dough once.

Define the function g, and determine (using repeated evaluations of the function g) the sequence of positions attained by the point of dough starting at $x = \frac{1}{10}$ through several transitions. Is there a pattern?

3 Graphs of Functions

This Chapter shows how to plot the graphs of functions in *Mathematica*.

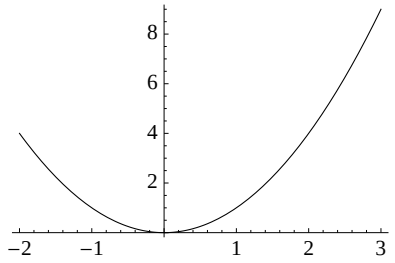
3.1 Drawing the Graph of a Function with Plot

Consider beginning with the definition of a familiar function, such as:

```
▶ f[x_] := x^2
```

To plot the graph of a function such as $f(x) = x^2$ above, we use *Mathematica*'s **Plot** operator. To see the graph over, say, the interval $[-2, 3]$, use the following:

```
▶ Plot[f[x], {x,-2,3}]
```

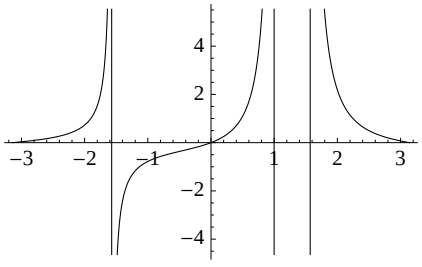


The first argument for **Plot** is the function $f(x)$ to be plotted. If you prefer, you can enter the expression for f directly as the first argument, without the need to separately define is as a named function. However, you'll almost always find it more convenient to define the function on its own, if only to keep the syntax a little cleaner.

The second argument for **Plot** is formed as a *list*, consisting of items separated by commas and enclosed in curly braces **{** and **}**. The first item of the list is the name of the independent variable name of the function or expression. The second and third items are the left- and right-endpoints of the interval, respectively, over which the graph is to be drawn.

Mathematica can plot many types of functions, and generally render them intelligently – even when the functions are discontinuous or behave wildly, such as in the case of the following expression that has infinite or non-existent limits at $-\pi/2$, 1 and $\pi/2$ in the interval $[-\pi, \pi]$:

```
▶ Plot[Tan[x]/(1-x), {x,-Pi,Pi}]
```



The vertical lines are, of course, to be interpreted as asymptotes; they do not represent values of the tangent function.

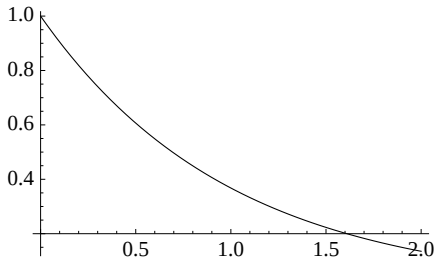
Should *Mathematica* not draw a suitable graph for you at any time, you have the ability to specify a number of options following the interval specification to adjust the display. We now consider a few such adjustments.

3.1.1 PlotRange

The **PlotRange** option allows you to manually adjust the vertical range that appears in a graph produced by **Plot**.

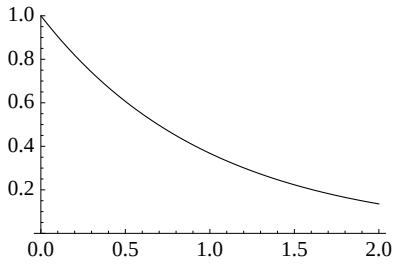
For example, when plotting the graph of $f(x) = e^{-x}$ on $0 \le x \le 2$, we have

```
▶ Plot[Exp[-x], {x,0,2}]
```



If you're not watching, you'd think that this exponential graph cuts through the x-axis somewhere around $x = 1.6$, and that certainly is not the case! The problem is that the y-range shown does not include 0. We correct for this by explicitly setting the y-range of the graph to be $0 \le y \le 1$ by adding the **PlotRange** option:

```
▶ Plot[Exp[-x], {x,0,2}, PlotRange→{0,1}]
```

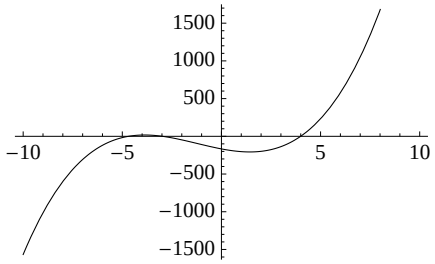


Notice that we've extended the syntax with a comma after the interval specification, then the option **PlotRange** is followed by an "arrow," made by typing the minus sign (**-**) and the greater-than sign (>) in sequence. (*The two characters you type to produce the arrow will be nicely converted into the single arrow character by Mathematica.*) Next comes a *list* of the y-values 0 and 1, separated by a comma and enclosed in curly braces **{** and **}**.

In the example above, we specified the **PlotRange** because *Mathematica*'s graph did not include enough of the y-range. There may be times when you'll want to specify the **PlotRange** should *Mathematica* show too much of the y-range.

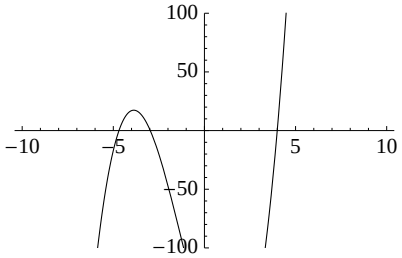
For example, consider trying to locate the zeroes of the polynomial $p(x) = 3x^3 + 11x^2 - 50x - 167$

```
▶ Clear[p]
  p[x_] := 3 x^3 + 11 x^2 - 50 x - 167
  Plot[p[x], {x,-10,10}]
```



We suspect that this polynomial has three zeroes, but because of the large range of y-values shown, we don't have enough detail to see what's happening near $x = -5$. Thus, we'll trim down the y-range.

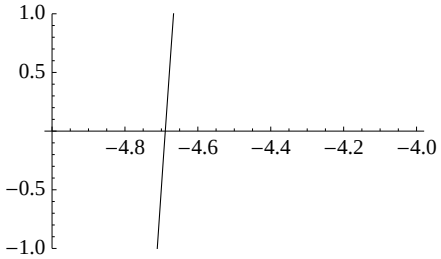
► `Plot[p[x], {x,-10,10}, PlotRange→{-100,100}]`



Now we can see that there really are three zeroes, near -5, -3, and 4. With a little more work, further restricting the x- and y-ranges as needed, we can get better approximations for each of the three zeroes just from the graph. This is often called *zooming in* on the graph.

For example, if you want more information about the zero between $x = -5$ and $x = -4$, you could try

► `Plot[p[x], {x,-5,-4}, PlotRange→{-1,1}]`



There's enough visual evidence in the graph above to guess the zero to be $x \approx -4.69$.

3.1.2 AspectRatio

You'll notice that all of the graphics so far have the same heights and widths, with the height being about 62% of the width. This is *Mathematica*'s default *aspect ratio* – the ratio of a graphic's height to its width.

This ratio comes to us from the early days of mathematics. It is the value $\frac{\sqrt{5}-1}{2} \approx 0.618034$, thought by the Greeks to be the most aesthetically-pleasing ratio of height to

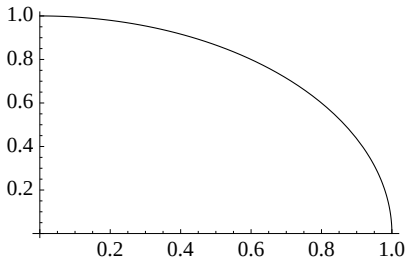
width. Numerically, the *Mathematica* constant **GoldenRatio** is the value $\frac{\sqrt{5}+1}{2}$, and the default aspect ratio for *Mathematica* graphic output is its reciprocal **1/GoldenRatio**, or

$$\frac{1}{(\sqrt{5}+1)/2} = \frac{\sqrt{5}-1}{2}.$$

You can manually control the aspect ratio of a graph produced by **Plot** by adding the **AspectRatio** option. This is especially important whenever you want to see true proportions in a graph.

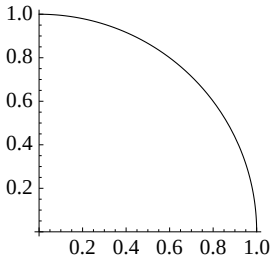
For example, the portion of the unit circle $x^2 + y^2 = 1$ that lies in the first quadrant can be seen with

```
► Plot[Sqrt[1-x^2], {x,0,1}]
```



Clearly, this does not look anything like a portion of a circle, due to its presentation with the default aspect ratio. To correct for this, either of the following inputs can be used

```
► Plot[Sqrt[1-x^2], {x,0,1}, AspectRatio→1]
    (* or *)
  Plot[Sqrt[1-x^2], {x,0,1}, AspectRatio→Automatic]
```



The first input forces the ratio of the graphic's height to width to be 1, which is correct for this specific graphic. How is the value of 1 determined? The x and y-intervals shown are both $0 \le x \le 1$, each of which has length one, and therefore have a ratio of 1 (y-length to x-length).

The second input, using the setting **AspectRatio→Automatic**, lets *Mathematica* make the computation of the aspect ratio for you so that units in the x-direction are just as long as units in the y-direction[7]. Setting **AspectRatio→Automatic** is always preferred when you want circles to appear as circles and right angles to appear as right angles.

[7]There's also one subtlety taken care of by using **AspectRatio→Automatic**. Any additional space in the graphic taken up by labels or other graphic elements that you might add is properly accounted for.

On the other hand, setting **AspectRatio→Automatic** will produce a poor result when the x- and y-ranges do not have comparable lengths. For the graphic below, the length of the y-range is less than 0.4% the length of the x-range.

```
► Plot[Sqrt[x]/10, {x,0,1000},
  AspectRatio→Automatic]
```

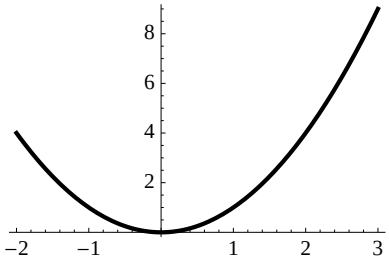
3.1.3 PlotStyle

You should think that *Mathematica* sketches graphs with some sort of "pen in hand." Pens can have different drawing characteristics such as the color of the ink used, thickness of the point, or even (especially in the case of worn pens) how solid an image might be produced. Such characteristics of the pen "strokes" used to draw the graph of a function or expression may be adjusted by specifying a value or collection of values for the **PlotStyle** option.

3.1.3.1 Thickness

You might wish to have the graph stand out more from the axes by drawing it with a thicker pen. The most convenient way to do this is with

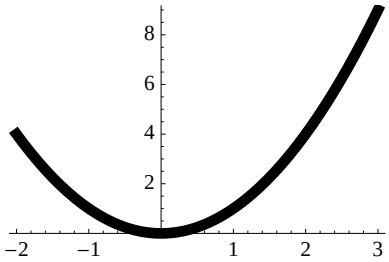
```
► Plot[x^2, {x,-2,3}, PlotStyle→Thick]
```



Mathematica supplies the predefined constants **Thick** (as used above, with the graph appearing thicker than the axes) and **Thin** (which would make the graph thinner than the axes) for convenience. You'll probably use one or the other in most instances.

More generally, you can control the thickness of a curve very precisely using the **AbsoluteThickness** directive. Its argument defines the width of the curve in terms of printer's points, with 1 point being approximately $\frac{1}{72}$ of an inch. So, for example, to have the graph drawn with a 5 point width, use

```
► Plot[x^2, {x,-2,3},
    PlotStyle→AbsoluteThickness[5]]
```



The following table shows how **Thick** and **Thin** compare with the default in terms of **AbsoluteThickness**.

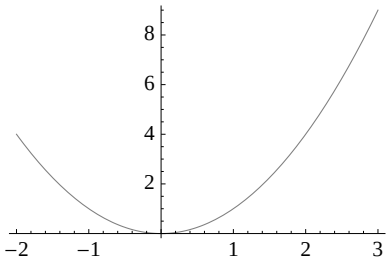
Abbreviation	Specification
Thin	**AbsoluteThickness[0.25]**
(Default)	**AbsoluteThickness[0.5]**
Thick	**AbsoluteThickness[2]**

3.1.3.2 Color and Gray Scale

A second, useful alteration of the way in which a curve is drawn is to use either color or shades of gray (in case you're working with a black and white printer). In fact, if you've been watching, *Mathematica* has already been drawing curves in blue for you (although the graphs in this text have been altered for black and white printing).

For example, to have the graph of $y = x^2$ above sketched in gray, you supply a **GrayLevel** directive for **PlotStyle**.

```
▶ Plot[x^2, {x,-2,3}, PlotStyle→GrayLevel[0.5]]
```



The argument of **GrayLevel** should be a number between 0 and 1. A **GrayLevel** of 0.0 plots the curve in black. Values larger than zero use less black and more white. A **GrayLevel** of 1.0 plots the curve in white. The 0.5 value used above draws the curve in a 50% gray level.

Some useful abbreviations that you can use (instead of specifying the gray level directly) are shown in the table below.

Abbreviation	Specification
LightGray	**GrayLevel[0.85]**
Lighter[Gray]	**GrayLevel[2/3]**
Gray	**GrayLevel[0.50]**
Darker[Gray]	**GrayLevel[1/3]**

As for colors, they're also pretty easy to handle. *Mathematica* has already predefined several commonly used colors such as **Blue**, **Red**, **Yellow**, and **Green**. So we could instead use the following to have a curve drawn in red in its default thickness (although the graph will not be shown here).

```
▶ Plot[x^2, {x,-2,3}, PlotStyle→Red]
```

The complete list of named colors and some of their variations can be found in the online Documentation Center by searching for Colors.

Finally, you can choose any color of the rainbow by using the **RGBColor** directive. This allows you to specify the exact levels of red, green, and blue to be used in creating the color.

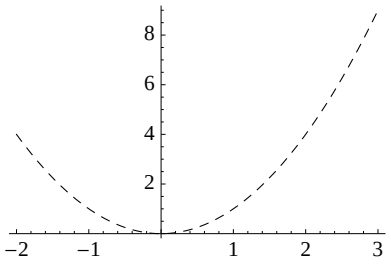
For example, the color yellow is obtained by mixing red and green without blue, thus **RGBColor[1,1,0]** generates a bright yellow. Brown is obtained by mixing red, green, and blue in the combination of 60% red, 40% green, and 20% blue, thus **RGBColor[0.6,0.4,0.2]** generates brown.

A fourth argument for **RGBColor** allows you to specify the *opacity* of the color, the degree to which the color is transparent (and thus the degree to which portions of a graphic behind the color are visible.) We'll not say anything further on opacity, except to say that you should check the online examples, starting with **?RGBColor**.

3.1.3.3 Dashed and Dotted Curves

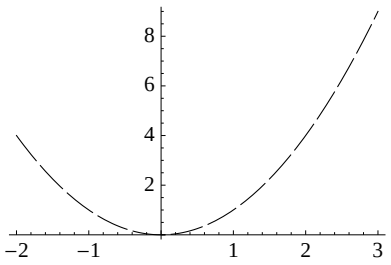
Curves can be drawn using a dashed line, rather than solid line – as if the pen you're drawing with might be running out of ink. For example, to have the graph sketched with a simple dashing, you can use the **Dashed** value for **PlotStyle**.

```
▶ Plot[x^2, {x,-2,3}, PlotStyle→Dashed]
```



The **Dashed** value is a convenient abbreviation that's sufficient for most situations. More generally you can control dashing more precisely with one of **Dashing** or **AbsoluteDashing**. For example, to draw the dashes 15 points in length, but leave 3 points of space between successive dashes, you'd use

```
▶ Plot[x^2, {x,-2,3},
    PlotStyle→AbsoluteDashing[{15,3}]]
```



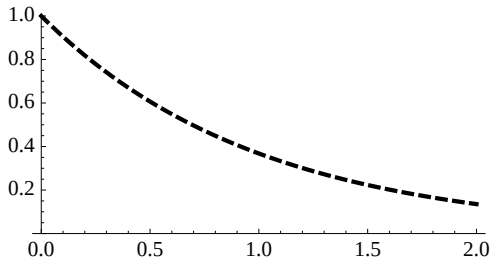
The companion **Dashing** directive follows a similar syntax, but its measurements are specified as percentages of horizontal width rather than absolute widths. **Dashed** is an abbreviation (approximately) for **Dashing[{0.025,0.025}]**.

Finally, **Dotted** can be specified as a value for **PlotStyle**. It equates (approximately) to **Dashing[{0,0.025}]** and produces exactly what you'd expect for output (however, we'll not demonstrate that here).

3.1.4 Multiple Options, Multiple Styles

It is possible to specify multiple options for the **Plot** command, as well as multiple pen characteristics for the **PlotStyle** option itself for a single curve, as the following example shows.

```
► Plot[Exp[-x], {x, 0, 2},
    PlotRange → {0, 1},
    AspectRatio → Automatic,
    PlotStyle → {Black, Dashed, Thick}]
```



Here, you see that

(a) The options **PlotRange**, **AspectRatio**, and **PlotStyle** simply are appended after the interval specification and separated by commas (they can appear in any order);
(b) We've formatted the input so that each of the options for **Plot** appears on a new line to make the input easier to read;
(c) The three items for the **PlotStyle** option have been formed into a list, separated by commas and enclosed by curly braces **{** and **}** (the order of the items within this list is not important).

As input now becomes more complicated, you'll find it a good strategy to separate options out one to a line, as we did above. The input is easier to read and, more importantly, easier to edit.

One additional modification we'd recommend for the input above is to separate out the styles under a separate name as follows (the output is the same as what's above, of course)

```
► styles = {Black, Dashed, Thick};
  Plot[Exp[-x], {x, 0, 2},
    PlotRange → {0, 1},
    AspectRatio → Automatic,
    PlotStyle → styles]
```

3.1.5 Other Plot Options

PlotRange, **AspectRatio**, and **PlotStyle** are just three of almost 60 different options allowed with the **Plot** operator (as of version 7 of *Mathematica*). The following table includes a few other easy-to-use (and understand) options you may want to know about.

Option	Sample Usage
Axes	**Axes→False**
AxesLabel	**AxesLabel→{"x","y"}**
AxesOrigin	**AxesOrigin→{0,0}**
Filling	**Filling→Axis**
Frame	**Frame→True**
GridLines	**GridLines→Automatic**
PlotLabel	**PlotLabel→**"The Graph of the Sine Function"
Ticks	**Ticks→None**

3.2 Getting Help for Plot (and other commands)

Since the basic **Plot** command is used so often and has so many possible options, you should know that there are three primary ways in which you can get some help in putting together exactly the right Plot command. You'll also find that many of these methods extend to other command areas in *Mathematica*.

3.2.1 The ? Command

For any *Mathematica* command for which you know its name, use the question mark operator to ask for information about the operator.

For example, to get help for the **Plot** operator, the following sequence is how you start:

```
▶ ?Plot
```

Plot[f, {x, x_{min}, x_{max}}] generates a plot of f as a function of x from x_{min} to x_{max}.
Plot[{f_1, f_2, ...}, {x, x_{min}, x_{max}}] plots several functions f_i. ≫

In addition to a short summary of the operator's format and arguments, you can click on the >> link at the end of the summary to open up a help page on the **Plot** operator (*not shown*).

Of note is that the help page not only has basic information, but also

- You can copy and paste any of the examples shown on the Help page to your notebook – and, in fact, you don't even have to do a copy and paste, since you can execute any example shown right on the help page.
- Each help page is broken into a number of sections that can be shown or hidden. Opening up the Details and Options area of the page will show more specific information about the many options available for use with the operator.
- Opening up the See Also section of a help page will provide links to related commands, often giving you clues in case you're looking for a more appropriate command to use for a particular situation instead of, say, **Plot**.
- Opening up the Tutorials sections will lead to very useful tutorial information.

➽ You can get help using the question mark ("**?**") on *any* command.

But what happens if you don't remember (or haven't yet heard of) the name of a command? The answer is pretty easy, because the **?** command allows the use of the asterisk as a wildcard character to represent any string of zero or more characters.

For example, suppose you need some help factoring an expression (we'll learn about this operation in the next Chapter), and you're guessing that *Mathematica* probably knows about factoring. Remembering that names in *Mathematica* begin with a capital letter, use

▶ `?Factor*`

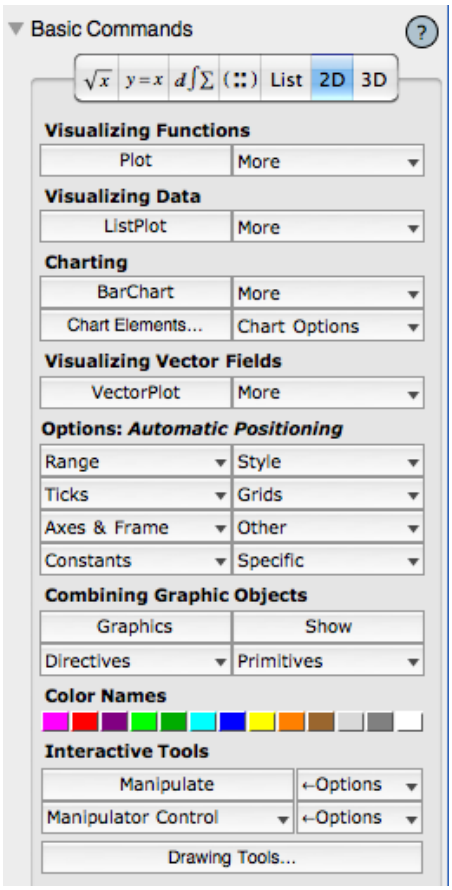
and *Mathematica* will return a list of commands that *begin with* the word **Factor**, whereas

▶ `?*Factor*`

will return a list of commands that have the word **Factor** anywhere in their name.

3.2.2 The Basic Commands Palette

We've already mentioned the use of a palette. *Mathematica* contains palettes for many different areas, including one specifically to help you in forming proper **Plot** syntax. Open up the Basic Commands palette and click the **2D** button.



When you click on the Plot button, a template for the **Plot** operator is added to your notebook:

Plot[*function* , { *var* , *min* , *max* }]

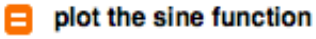
The "function" area of the template is highlighted, so just start typing the function or expression. Hit the Tab key to move to the next placeholder in the template for the variable, and so on.

By clicking on buttons such as Range, Style, Ticks, and so forth in the Basic Commands palette, additional option templates will be added to the Plot command you're building. (*We'll not show that here; this is best seen interactively.*)

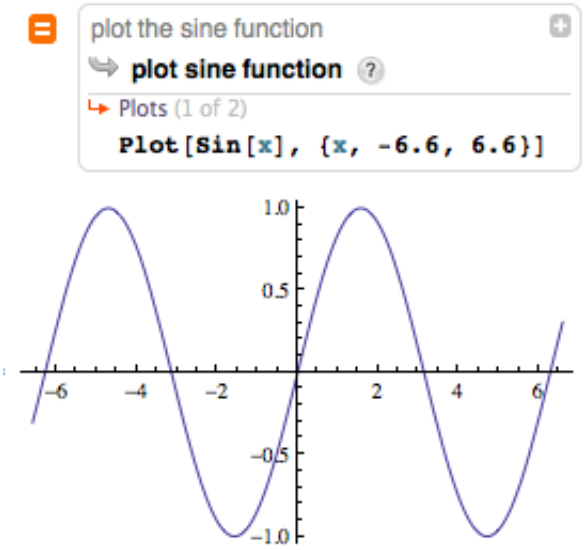
3.2.3 The = Operator

Version 8 of *Mathematica* introduced the capability to enter commands in a less-structured, natural language format. As long as you are *connected to the internet* and begin an input cell with the = sign, you can type commands almost in English and *Mathematica* will return what best fits your language.

For example, if you wanted to begin working with a plot the sine function, you **begin** an input cell with the = operator and then just type anything that's reasonably close to being interpreted as a plot request, such as



Mathematica will respond with the syntax for a basic **Plot** command for the sine function, having chosen a suitable interval to get you started (in this case, one that's slightly longer than $[-2\pi, 2\pi]$).



Clicking on the + button above will reveal more information and possible alternatives – that will not be demonstrated here.

Once again, use of the = operator can be used at any time, in any context, and you don't have to be very precise in your choice of language or syntax to get started. This is, perhaps, the most powerful feature added in Version 8 of *Mathematica*.

3.3 Graphing Several Functions

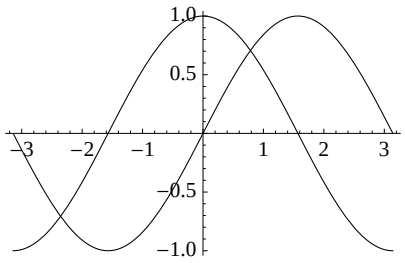
We often place several curves on the same graph and *Mathematica* supports this easily. Sometimes, the graphs are drawn over the same interval, so only one **Plot** is needed. In other situations, the graphs are drawn using completely different x- and y-ranges using separate **Plot**s and then are combined using the **Show** operator.

3.3.1 Multiple Functions in the Same Plot

The first argument of the **Plot** operator can be either a single function (or expression), as we've seen already, or a *list* of functions (and/or expressions). The individual functions and expressions are enclosed in curly braces **{** and **}** and separated by commas. The use of a list allows us to graph more than one function or expression over the same interval.

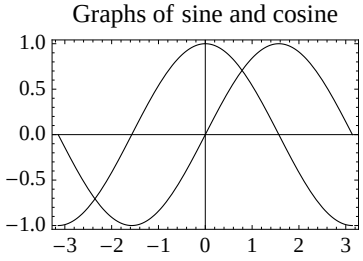
For example, to see the sine and cosine curves on the same set of axes, we use:

```
► Plot[ {Sin[x],Cos[x]}, {x,-Pi,Pi}]
```



Options that affect the graphic *as a whole* can be specified just as before. To add a title to the graphic and place a frame around it, we would use

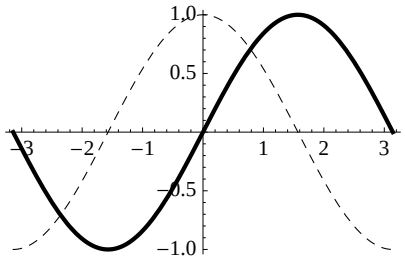
```
► Plot[ {Sin[x],Cos[x]}, {x,-Pi,Pi}, Frame→True,
    PlotLabel→¨Graphs of sine and cosine¨]
```



However, specifying values for the **PlotStyle** option becomes more involved, since *Mathematica* extends the syntax of **PlotStyle** to allow different style characteristics to be applied to each curve separately. (There are some other options where this is also the case, although we'll only demonstrate the technique for **PlotStyle**.)

As a first demonstration, we'll draw the sine curve with a thicker pen and the cosine curve dashed.

```
► Plot[ {Sin[x],Cos[x]}, {x,-Pi,Pi},
    PlotStyle→{Thick, Dashed}]
```



Notice what's happened. When we drew a single curve earlier, both the **Thick** and **Dashed** directives would have been applied to the curve. Here, two curves are drawn, with **Thick** applied to the first curve and **Dashed** applied to the second curve.

So the meaning of the value for **PlotStyle** has changed. A list of two elements is specified (since there are two curves), separated by a comma and enclosed in curly braces **{** and **}**. The first element of the list is associated with the first curve, and the second element of the list is associated with the second curve.

But then how do we specify more than one directive for either one of the curves? The answer: collect those directives into a *list*. Here's how the curves would be drawn with the sine thick and red, and the cosine dashed in green (although the output will not be shown).

```
► Plot[{Sin[x], Cos[x]}, {x, -Pi, Pi},
    PlotStyle -> {{Thick,Red}, {Dashed,Green}}]
```

Be sure you recognize the structure of the specification of the **PlotStyle**. It is a *list* of two items, each of which is itself a list. Make sense?

With the complexity that you now see involved in writing a single **Plot** input, we offer the following technique, one that we suggested earlier. If the **PlotStyle** specification starts to become unwieldy, define the styles separately as follows.

```
► style1 = {Thick,Red};
  style2 = {Dashed,Green};
  Plot[{Sin[x], Cos[x]}, {x, -Pi, Pi},
    PlotStyle -> {style1, style2}]
```

3.3.2 Combining Multiple Plots

Graphics produced in *Mathematica* can be assigned to variables, just as values and expressions can be assigned. Using the **Show** operator, those graphics can later be combined into a single graphic.

For example, the graphs of sine and cosine produced above could be constructed separately and assigned to variables and then merged into a single graphic using **Show**, as in the following sequence.

```
► graph1 = Plot[Sin[x], {x, -Pi, Pi},
      PlotStyle→{Thick,Red}];
  graph2 = Plot[Cos[x], {x, -Pi, Pi},
      PlotStyle→{Dashed,Green}];
  Show[graph1,graph2]
```

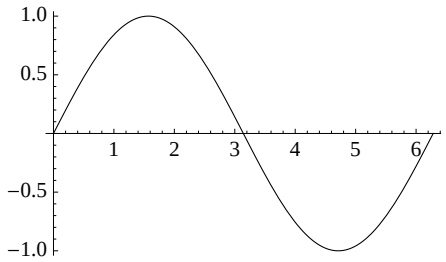
The sequence above is a blueprint for how to construct complicated graphics: draw them in sections and combine them with **Show**.

However, when graphics have associated options that might possibly conflict when the

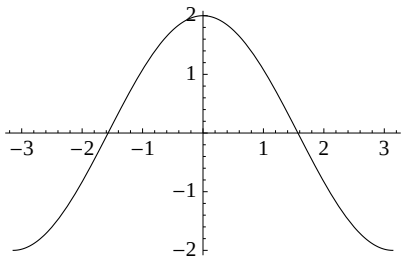
graphics are combined, **Show** uses the options specified for the *first* of its arguments as the primary control of its output.

You can see an example of this behavior in the following sequence, where the sine and cosine graphs are sketched over different intervals.

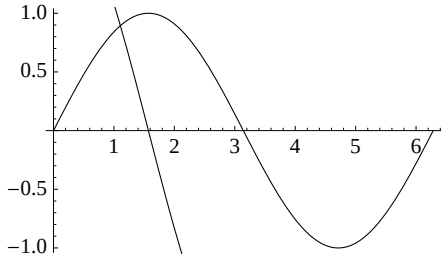
```
► graph1 = Plot[Sin[x],{x,0,2Pi}];
```




```
► graph2 = Plot[2Cos[x],{x,-Pi,Pi}];
```



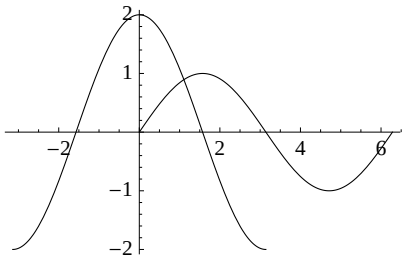

```
► Show[graph1,graph2]
```



The problem above is that **graph1** had an associated **PlotRange** of $0 \leq x \leq 2\pi$ and $-1 \leq y \leq 1$. This was used as the **PlotRange** for the combined graphic produced by **Show**.

We can correct for this by explicitly specifying the **PlotRange** to be used by **Show**.

```
► Show[graph1,graph2, PlotRange→{{-Pi,2Pi},{-2,2}}]
```



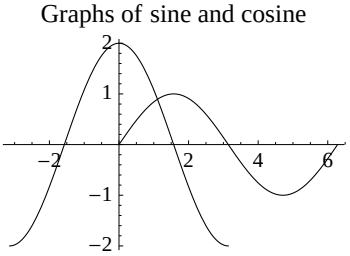
Once again, you see syntax complexity creeping into the discussion. Here, the **PlotRange** has been specified using a list of two items, one for the x-range, and one for the y-range. Each of these is itself a list of **{-Pi,2Pi}** and **{-2,2}**, respectively.

As a convenience, *Mathematica* lets you specify the value **Automatic** for a **PlotRange**. Doing so with **Show** forces *Mathematica* to enlarge the range in each direction to include the ranges of each individual graphic.

Finally, other options you'd use with **Plot** can be included with **Show** directly, either to enhance the output, or to override options associated with the individual plots.

For example, if the last graph above was acceptable, but an appropriate label for the graphic was not generated for either of the component graphics, it can now be added by specifying a **PlotLabel** option with **Show**.

```
► Show[graph1,graph2, PlotRange→{{-Pi,2Pi},{-2,2}},
            PlotLabel→"Graphs of sine and cosine"]
```



3.4 Exercises

1. Two options sometimes used with **Plot** are **Ticks** and **GridLines**. Find out about these two options and their default values using the information operator **?**, and experiment with their effect on **Plot** by producing a few simple plots.

2. Consider plotting the following sixth-degree polynomial p over the interval $-5 \le x \le 10$.

$$p(x) = 106.04 - 298.75x + 347.83x^2 - 214.16x^3 + 73.52x^4 - 13.34x^5 + x^6$$

True or False: for $0 \le x \le 4$, $p(x) = 0$. What explanation can you give for your observation?

3. Plot the graph of the function below over the interval $[-3, 4]$ using the **Piecewise** operator.

$$f(x) = \begin{cases} x^4 + x + 1, & x \le -1 \\ 2x + e, & -1 < x < 2 \\ \cos\frac{\pi x}{2} + 5x, & 2 \le x \end{cases}$$

Are there any unexpected features in the graph?

4. By properly restricting the **PlotRange** and the interval over which **Plot** is used, estimate all zeroes of the following function over the interval $-\pi \le x \le \pi$.

$$f(x) = \frac{\pi}{10} + \sin x + \sin(3x) + \sin(4x)$$

Be sure to check your answers numerically – i.e., is $f(x) \approx 0$ for each value x you find?

5. By properly restricting the **PlotRange** and the interval over which **Plot** is used, estimate all zeroes of $f(x) = \sin(2x)e^{\cos(3x)} - 0.1$ in $[0, 8]$. Be sure to check your answers numerically – i.e., is $f(x) \approx 0$ for each value x you find?

6. By properly restricting the **PlotRange** and the interval over which **Plot** is used, estimate the maximum and minimum values of each of the following functions, on the indicated interval, to about three decimal places:

(a) $f(x) = \dfrac{(x^5 - 4)\sin x^2}{1 + x^4}$, on the interval $[0, 2]$.
(b) $f(x) = \dfrac{\pi}{10} + \sin x + \sin(3x) + \sin(4x)$ on $[-\pi, \pi]$.
(c) $f(x) = \sin(2x)e^{\cos(3x)} - 0.1$ on $[0, 8]$.

7. Use the **Plot** operator to help determine whether each of the following statements is true. Explain why you think your answer is correct.

(a) $r(x) = \dfrac{x - 5}{x^2 + 2x}$ is increasing for $x > 0$.
(b) $f(x) = 64x^4 - 16x^3 + x^2$ is increasing on $[0, 5]$.

8. Define the function $f(x) = x^2 - 3x + 2$. Plot the graphs of $f(x)$, $f(x - 1)$ and $f(x - 2)$ over the interval $[-1, 7]$ on the same set of axes, using different shading and dashing for each of the graphs. What can be concluded about the relationship of the graph of $f(x)$ and $f(x - a)$, for any value of a?

9. Define the function $f(x) = 2x^2 - 4x - 5$. Plot the graphs of $f(x)$ and $|f(x)|$ over the interval $[-4, 4]$ on the same set of axes, using different shading and dashing for each of the graphs. What can be concluded about the relationship of the graph of $f(x)$ and $|f(x)|$ in general?

10. Define the function $f(x) = x^3 - 2x$. Plot the graphs of $f(x)$, $f(x) + 2$ and $f(x) + 4$ over the interval $[-4, 4]$ on the same set of axes, using different shading and dashing for each of the graphs. What can be concluded about the relationship of the graph of $f(x)$ and $f(x) + a$ in general, for any value of a?

11. Define the function $f(x) = \sin x$. Plot the graphs of $f(x)$, $f(2x)$ and $f(3x)$ over the interval $[-\pi, \pi]$ on the same set of axes, using different shading and dashing for each of the graphs. What can be concluded about the relationship of the graph of $\sin(x)$ and $\sin(ax)$ in general, for any value of a?

12. Define the function $f(x) = \ln x$. Plot the graphs of $f(x)$, $f(2x)$ and $f(3x)$ over the interval $[1, 10]$ on the same set of axes, using different shading and dashing for each of the graphs. What can be concluded about the relationship of the graph of $\ln(x)$ and $\ln(ax)$ in general, for any value of a?

13. Sketch the graphs of $y = \sinh(x)$, $y = \cosh(x)$ and $y = \frac{1}{2}e^x$ on the same set of axes, over the interval $[-4, 4]$, using different shading and dashing. Which curves are above which

other curves? Are any of the curves asymptotic to any others? If so, explain why. (You may have to refresh your memory about the definition of $y = \sinh(x)$ and $y = \cosh(x)$.)

14. Graph the rational function $r(x) = \dfrac{x^3 - x^2}{5x + x^2 - 2x^3}$ with a solid pen, together with the line $y = -1/2$ over the interval $[-10, 10]$, using dashing. Is there an asymptotic relationship between the curves?

15. The improper rational function $r(x) = \dfrac{x^3 - x^2}{x^2 + x - 7}$ may be written $r(x) = x - 2 + \dfrac{9x - 14}{x^2 + x - 7}$, after division of the polynomials is carried out. Hence, the line $y = x - 2$ is an oblique asymptote of the graph of the function r. Graph r and this line over the interval $[-10, 10]$, using dashing for the line, to demonstrate the asymptotic relationship.

16. Graph the functions

$$y = \ln x \text{ and } y = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \frac{x^6}{720}$$

on the same set of axes over the interval $[1, 7]$, using different shading and dashing. Is there any symmetry in the graphs with respect to the axes? Do the graphs have any symmetry with the line $y = x$?

17. Graph (both of) the composition(s) of the functions given in the previous problem, over the interval $[1, 7]$. What do you observe?

18. Use a graph to estimate the smallest positive root of the equation $\tan(x) = x$. (A value x_0 is a root of this equation if the curves $y = \tan(x)$ and $y = x$ intersect at x_0.) Be sure to check the result you find numerically.

19. Use a graph to estimate a root of $\cosh(x) - x = 1 + \cos(x)$. Be sure to check the result you find numerically.

20. Use a graph to estimate all solutions to the equation $x^3 - x = \sin(x) - 1$. Be sure to check your proposed solutions by evaluating the equation at each potential solution.

21. Use a graph to estimate all solutions to the equation $x = 4\sin(x) + 1$. Be sure to check your results by evaluating the equation at each potential solution.

22. Use a graph to estimate all solutions to the equation $x = 4\tan^{-1}(x)$. Be sure to check your results by evaluating the equation at each potential solution.

23. Use a graph to estimate all intersections of the graphs of the functions $f(x) = \frac{x}{3}$ and $g(x) = \ln(x^4 + \frac{1}{3})$. Be sure to check your proposed solutions by evaluating the equation at each intersection you find.

24. Use a graph to estimate all intersections of the graphs of the functions $f(x) = \sin(3x)$ and $g(x) = \frac{x^2}{22}$. Be sure to check your proposed solutions by evaluating the equation at each intersection you find.

25. Use a graph to estimate the solution to the inequality $\left|\dfrac{2x - 5}{x^7 - 5x^3 - 2}\right| \leq 1$ by plotting the functions $f(x) = \dfrac{2x - 5}{x^7 - 5x^3 - 2}$, $g(x) = -1$ and $h(x) = 1$ on the same set of axes and zooming for intersections.

4 Graphs of Equations and Parametric Curves

Not every graph we draw is that of a function. In the most general case, we'll be handed a simple equation in two variables (say, x and y) and asked to plot its graph (i.e., the collection of all points having coordinates (x, y) where x and y satisfy the equation).

In more interesting cases, the curve might be parameterized in a form $(x(t), y(t))$ for a parameter t in some interval, or even be given in polar coordinates where $x(t) = r(t)\cos\theta$ and $y(t) = r(t)\sin\theta$, for θ restricted to some interval.

The following sections address each of these cases.

4.1 Graphs of Arbitrary Equations

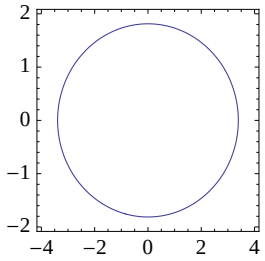
Not every equation involving variables x and y can be written in the simple form $y = f(x)$, for some function f, and thus we cannot use the **Plot** operator to graph it. Thus we will rely on the higher level **ContourPlot** operator to handle the task of graphing arbitrary equations.[8]

4.1.1 ContourPlot and One Equation

The **ContourPlot** operator is used to graph an equation (or equations) involving two variables (say, x and y) that lies within a given rectangle (say, $a \le x \le b$ and $c \le y \le d$).

For example, to see the ellipse $2x^2 + 7y^2 = 23$, we will use

```
► ContourPlot[2x^2+7y^2==23, {x,-4,4}, {y,-2,2}]
```



The syntax for this use of **ContourPlot** has three arguments: an equation involving two variables, and the specifications of suitable intervals for each of the variables over which the equation is to be considered.

The first of these above is easy to understand: it's just the equation of interest. However, you have to use the *double* equal sign == to enter the equation. (*A common mistake is to use a single equal sign, so watch carefully for this.*)

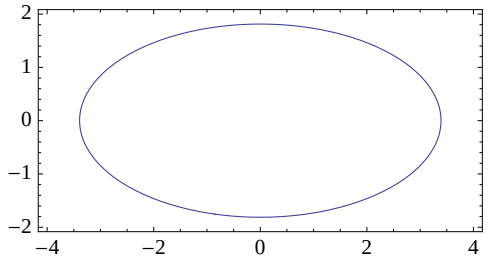
The second and third arguments are the interval specifications for $-4 \le x \le 4$ and $-2 \le y \le 2$, in the usual form. How did we know that these were appropriate choices? Apart from some combination perhaps of guessing and experience, we can see that the variables must satisfy the inequalities $x^2 \le 23/2 = 11.5$ and $y^2 \le 23/7 \approx 3.3$ and then we rounded up a little after taking square roots.

As you probably should guess from the graphic, **ContourPlot** sets **AspectRatio→1** for its output by default. Indeed, the ellipse $2x^2 + 7y^2 = 23$ should have its longer axis parallel

[8] Be aware, though, that **ContourPlot** is a more general tool for working with level curves of two-variable functions, and what we show here demonstrates only a portion of the full capability of **ContourPlot**.

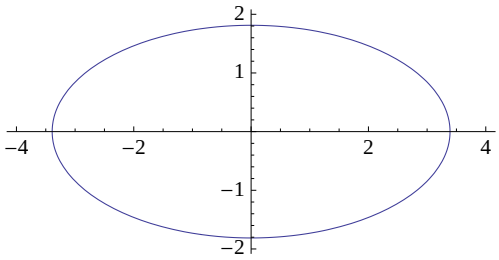
to the x-axis and its shorter axis parallel to the y-axis. To get a truer representation for this graph, we'd want to set the **AspectRatio** ourself.

```
► ContourPlot[2x^2+7y^2==23, {x,-4,4}, {y,-2,2},
    AspectRatio→Automatic]
```



You'll also notice that the graphic produce above is framed and no axes have been drawn. If you'd prefer to see the picture with a more common presentation of the axes, you can use the following to turn off the frame and to turn on the axes.

```
► ContourPlot[2x^2+5y^2==23, {x,-4,4}, {y,-2,2},
    AspectRatio→Automatic,
    Frame→False, Axes→True]
```

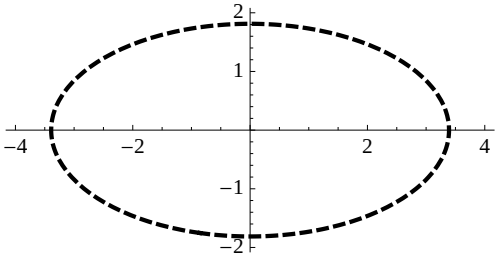


There's a cute feature associated with **ContourPlot** that's worth mentioning here. For the graphic above (and the two that preceded it), you'll see a tooltip on the mouse as you move over the curve showing the equation of the curve. Neat!

4.1.1.1 Controlling the Style

Some options you can specify for **Plot** can be used with **ContourPlot**. However, there is one important difference: the drawing characteristics of a curve are now controlled using the **ContourStyle** option. So to add some style to the graph above, you'd use

```
► ContourPlot[2x^2+5y^2==23, {x,-4,4}, {y,-2,2},
    AspectRatio→Automatic,
    Frame→False, Axes→True,
    ContourStyle→{Magenta,Thick,Dashed}]
```

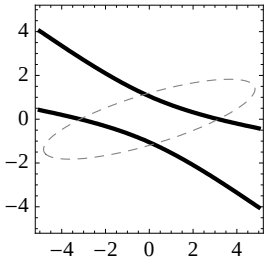


Notice that the actual list of styles you supply follows the same syntax you learned earlier for **PlotStyle** with the **Plot** command.

4.1.1.2 Multiple Equations

ContourPlot may be used to plot the graph of more than one equation at the same time (inside a common rectangle). For example, the curves $x^2 + 8xy + 9y^2 = 10$ and $x^2 - 4xy + 7y^2 = 10$ define a hyperbola and ellipse, respectively, and can be seen with

```
► ContourPlot[{x^2+8x*y+9y^2==10,x^2-4x*y+7y^2==10},
      {x,-5,5},{y,-5,5},
      ContourStyle→ {{Thick}, {Gray, Dashed}}]
```



You once again see list syntax used to define the two equations above for **ContourPlot**. They must be separated by a comma and enclosed in curly braces **{** and **}** to form the first argument of **ContourPlot**.

The usual range of options for **ContourPlot** can be specified when more than one curve is drawn, although as you see above, the specification for **ContourStyle** requires the usual syntactic attention to detail, since it is a list of two items, each of which is itself a list.

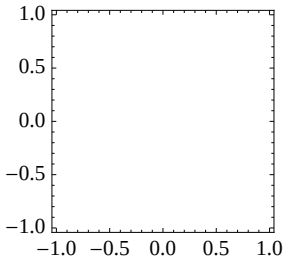
4.1.1.3 ContourPlot Issues

ContourPlot is a fairly robust command, so there are rarely technical issues that arise when you use it as we've described here. However, you do need to think ahead and be careful when using **ContourPlot** for two reasons:

- the equation you're attempting to plot may not have *any* solutions (e.g., $x^2+y^2 = -1$); or more generally,
- the equation you're attempting to plot does not have any solutions *within the enclosing rectangle* you're specifying (e.g., $(x-3)^2 + y^2 = 1$ has no solutions if you restrict $-1 \le x \le 1$).

Attempting to do either of the above does *not* generate an error message. Rather, all you'll see is an *empty graphic* for the rectangle you specified.

```
► ContourPlot[(x-3)^2+y^2==1, {x,-1,1}, {y,-1,1}]
```



Beware the dreaded empty graphic! It is likely not an error of **ContourPlot**, but rather an error on your part either in specifying the equation correctly and restricting the plot to a section of the plane in which the equation actually has solutions.

4.2 Parametric Plots

A curve is said to be defined *parametrically* if it has the form $\{(x(t), y(t)) : a \leq t \leq b\}$, where x and y are functions of a variable t, and a and b are constants. *Mathematica* supports working with such curves.

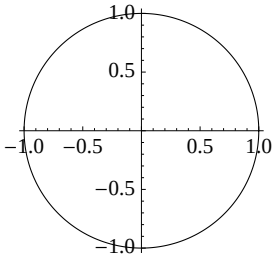
4.2.1 Plotting One Parametric Curve

A standard example of a curve that is defined parametrically is the unit circle, which can be described as the set of points

$$\{(\cos t, \sin t) : 0 \leq t \leq 2\pi\}$$

You can see its graph with the following input.

```
▶ ParametricPlot[{Cos[t],Sin[t]}, {t,0,2Pi}]
```



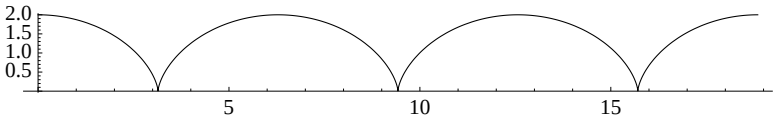
Notice that the first argument for **ParametricPlot** contains the parametric expressions for the x- and y-coordinates separated by a comma and enclosed in curly braces **{** and **}**. The second argument that names the independent variable and the range over which it varies has a familiar form, just as we've seen previously with the **Plot** operator.

A more interesting parametric curve is the *cycloid*, which is formed as the path through which a fixed point on a circle travels as the circle is "rolled" along the x-axis. Parametrically, it is usually described by the equations

$$x(t) = t + \sin t \text{ and } y(t) = 1 + \cos t$$

with the selection of the range of the parameter t determining which section of the curve is being investigated. Using this description directly over the interval $0 \leq t \leq 6\pi$, *Mathematica* produces

```
▶ ParametricPlot[{t+Sin[t],1+Cos[t]}, {t,0,6Pi}]
```

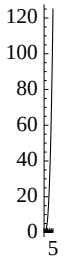


4.2.1.1 The AspectRatio Option

By default, the **ParametricPlot** operator produces a graphic with its **AspectRatio** set to **Automatic**. Thus we really do see a circle above in the first graphic and we see axes for the cycloid drawn with equal length units.

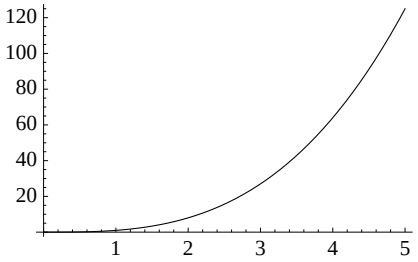
However, be forewarned that **AspectRatio→Automatic** may not always be the best choice and that you may have to control the aspect ratio directly. For example, the parametric curve $x(t) = t$ and $y(t) = t^3$, for $0 \leq t \leq 5$ will not display well with **AspectRatio→Automatic** because the x-coordinates vary from 0 to 5, while the y-coordinates vary from 0 to 125.

```
► ParametricPlot[{t,t^3}, {t,0,5}]
```



Reexecuting this with either of **AspectRatio→1** or **AspectRatio→1/GoldenRatio** will quickly get the graphic above under control.

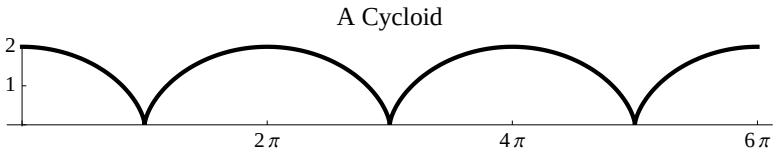
```
► ParametricPlot[{t,t^3}, {t,0,5},
      AspectRatio→Automatic]
```



4.2.1.2 Other ParametricPlot Options

Most options that can be specified with **Plot** can be used with **ParametricPlot** as well. For example, for the cycloid drawn above, it is easy to add a label, specify which tick marks to place on the axes, and make the graph appear think and in green.

```
► ParametricPlot[{t+Sin[t],1+Cos[t]}, {t,0,6Pi},
      PlotLabel→¨A Cycloid¨,
      PlotStyle→{Green,Thick},
      Ticks→{{0,2Pi,4Pi,6Pi}, {0,1,2}}]
```



A complete list of the more than 50 options that can be used with **ParametricPlot** is available in the online help. They can also be seen quickly with

```
► Options[ParametricPlot]
```

4.2.1.3 Multiple Curves in One ParametricPlot

You can easily plot more than one parametric curve using a single **ParametricPlot**, just as you can plot more than one function using **Plot**. The only requirement is that the parametric interval be the same.

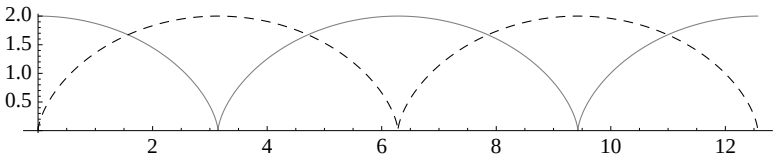
Consider sketching two cycloids at the same time, with the second "out of phase" from the first by π. That is, consider sketching the two cycloids:

$$\begin{cases} x_1(t) = t + \sin t \\ y_1(t) = 1 + \cos t \end{cases} \quad \text{and} \quad \begin{cases} x_2(t) = t + \sin(t+\pi) \\ y_2(t) = 1 + \cos(t+\pi) \end{cases}$$

The effect is much the same as rolling one ball along the x-axis, tracking the points that are initially at the "top" and the "bottom" of the ball.

The following illustrates the motion of the two points, with the point at the top represented by the gray track and the point at the bottom represented by the red, dashed track.

```
► x1[t_]:=t+Sin[t]
  y1[t_]:=1+Cos[t]
  x2[t_]:=t+Sin[t+Pi]
  y2[t_]:=1+Cos[t+Pi]
  ParametricPlot[{{x1[t],y1[t]},{x2[t],y2[t]}},
      {t,0,4Pi},
      PlotStyle→{{Gray},{Red,Dashed}}]
```



Notice first that we gave separate definitions to the parametric functions x_1, y_1, x_2, and y_2 that define each of the curves above. This keeps the syntax manageable and shows more clearly the syntax of the first argument for **ParametricPlot**.

Once again, we see list syntax being used throughout the input. The first argument is a list of two items, separated by a comma and enclosed in curly braces **{** and **}**. Each of these items defines a curve and is itself a list of two parametric functions separated by a comma and enclosed in curly braces **{** and **}**.

The syntax of the values supplied to the **PlotStyle** option is the same as what we saw earlier for multiple curves in a single **Plot**. Here, since two parametric curves are plotted,

the value for **PlotStyle** is a list of two items, separated by a comma and enclosed in curly braces **{** and **}**. Each of the items is itself a list of style directives separately associated with the curves (only one item appears in the first list, while two items appear in the second).

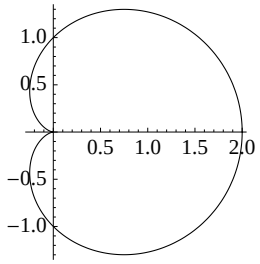
4.3 Polar Plots

A special class of curves that can be defined parametrically are those given in polar coordinates (r, θ), where $|r|$ represents the distance of a point P from the origin, and θ the angle formed by the x-axis and the line segment from the origin to P (as long as $r \neq 0$).

Usually, such curves appear in the form $r = f(\theta)$, over a given range of the variable θ. Points on such curves all appear in the form $\{(r\cos\theta, r\sin\theta) : \theta_1 \leq \theta \leq \theta_2\}$, and so these curves may be graphed using the **ParametricPlot** operator.

For example, the graph of a simple *cardioid* $r = 1 + \cos\theta$ is easily defined and plotted. (Here, we make use of the Typesetting palette to enter a Greek θ as the variable and a Greek π for the *Mathematica* constant **Pi**)

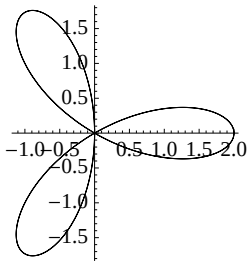
```
► Clear[f]
  f[θ_] := 1+Cos[θ]
  ParametricPlot[{f[θ]Cos[θ],f[θ]Sin[θ]}, {θ,0,2π}]
```



In fact, because such a curve is so common, *Mathematica* supplies the **PolarPlot** operator as a convenient way to use with the **ParamtericPlot** operator, without the need to write the curve explicitly in the form **{f[θ]Cos[θ],f[θ]Sin[θ]}**.

Indeed, consider plotting the three-leaf rose given by the equation $f(\theta) = 2\cos(3\theta)$.

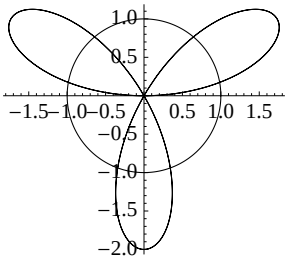
```
► Clear[f]
  f[θ_] := 2*Cos[3*θ]
  PolarPlot[f[θ],{θ,0,2π}]
```



Note that **AspectRatio→Automatic** is the default for the **PolarPlot** operator (as was the case for **ParametricPlot**). It may be necessary to explicitly specify, say, one of **AspectRatio→1** or **AspectRatio→1/GoldenRatio** to better present an unruly plot.

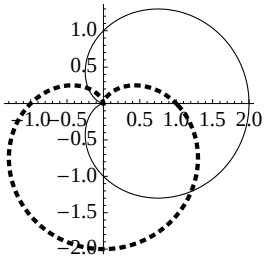
The **PolarPlot** operator is also capable of plotting multiple curves in the form $r = f(\theta)$ on the same set of axes, such as the three-leaf rose $r = 2\sin(3\theta)$ and the circle $r = 1$.

```
► PolarPlot[{2Sin[3θ],1},{θ,0,2π}]
```



Additionally, the full range of **PlotStyle** options may be specified with **PolarPlot**, as is the case with the following two cardioids, one of which is simply a rotation of the other:

```
► Clear[f,g]
  f[θ_]:=1+Cos[θ]
  g[θ_]:=1-Sin[θ]
  PolarPlot[{f[θ],g[θ]},{θ,0,2π},
      PlotStyle→{{Red}, {Thick, Dotted, Green}}]
```



It appears the curves intersect in three points. Two of the angles at which the intersections occur are somewhat obvious from the diagram, namely at $\theta = -\pi/4$ and $\theta = 3\pi/4$. This can be verified easily (recall that the *Mathematica* test of equality is the *double* equal sign ==)

```
► f[3π/4]==g[3π/4]
```

▷ True

```
► f[-π/4]==g[-π/4]
```

▷ True

The origin (pole) is also a point of intersection of the curves, but for different values of the angle θ, as we can see with

```
► f[π/2]==g[0]
```

▷ True

4.4 Combining Multiple Graphics

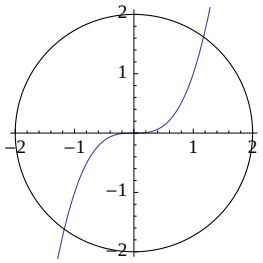
Any combination of function graphs created with **Plot**, graphs of equations created with **ContourPlot**, parametric plots created with **ParametricPlot**, and polar curves created

with **PolarPlot** separately can be combined using the **Show** command. The method is the same we demonstrated with **Plot** earlier: create the graphics separately, assign the results (the graphics) to variables, then use **Show** (supplying option adjustments as needed) to combine them in a single graphic.

In particular, you are not restricted to two graphics created by the *same* operator; *any two graphics* can be combined.

For example, to investigate where the circle $x^2 + y^2 = 4$ intersects the cubic $y = x^3$, we could use the following graphic (and then zoom in, of course).

```
graph1 = Plot[x^3,{x,-2,2}];
graph2 = ParametricPlot[{2Cos[t],2Sin[t]}, {t,0,2Pi}];
Show[graph1, graph2, PlotRange→{-2,2}, AspectRatio→Automatic]
```



4.5 Exercises

1. Use **ContourPlot** to help identify the curve given by the equation $x^2+y^2-2x+4y = -1$. Find parametric equations for this curve, and check your result with **ParametricPlot**.

2. Use **ContourPlot** to sketch the curve given by the equation $x^3 + y^3 = 6xy$ (this is often called the *Folium of Descartes*). Estimate as best you can the point "on the loop" that's farthest from the origin.

3. Consider the curve given parametrically by the equations

$$x(t) = \frac{3t}{t^3+1} \text{ and } y(t) = \frac{3t^2}{t^3+1}$$

for $t \neq -1$. Sketch this curve. Indicate the order in which the curve is traced. Does the curve look familiar? (*Hint: the previous problem!*)

4. Consider the curve given parametrically by $x(t) = \cot(t)$ and $y(t) = \sin(t)\cos(t)$, for $0 < t < \pi$. Sketch the curve. Compare the curve with the graph of the function $y = \dfrac{x}{1+x^2}$. What can you conclude? Why?

5. Sketch the polar curves $r = a + 2\sin\theta$, for values of $a = \frac{1}{2}, 1, \frac{3}{2}, 2, \frac{5}{2}$, on the same set of axes, using different shadings or dashings to distinguish the curves. What can be concluded about the character of the curve in terms of a?

6. Sketch the graphs of the polar equation $f(\theta) = \sin(k\theta)$, for a few positive, integral values of k. What is the relationship between k and the number of leaves in the graph?

7. Sketch the graphs of the polar equations $f(\theta) = \sin(k\theta)$ and $f(\theta) = \cos(k\theta)$ on the same set of axes, for a few positive, integral values of k. What is the relationship between the graphs?

8. Determine the relationship between the positive integer k and the number of leaves on the graph of $f(\theta) = \cos\theta - \cos(k\theta)$.

9. (Conic Sections) You may remember that the graph of any equation of the form $Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$ is a conic section, which will be a (possibly degenerate) circle, ellipse, or hyperbola. For such an equation, the quantity $B^2 - 4AC$ determines the character of the graph (barring degeneracy) to be:

- a parabola if $B^2 - 4AC = 0$;
- an ellipse if $B^2 - 4AC < 0$; or
- a hyperbola if $B^2 - 4AC > 0$.

You've probably worked algebraically with such equations in the case when $B = 0$. Unfortunately, the study of the case for $B \neq 0$ involves a rotation of the conic section, and this computation is often omitted standard coursework (although it appears in almost all Calculus books).

Use the information and discussion above to first identify the graph of each of the following equations in terms of its coefficients, and then produce its graph.

(a) $x^2 - 7xy + 5y^2 + 2x + 3y - 9 = 0$
(b) $4x^2 + 2xy + 3y^2 + x - 5y - 17 = 0$
(c) $2x^2 + 4xy + 8y^2 + 3x - y + 5 = 0$
(d) $3x^2 - xy - 1 = 0$

10. (Conic Sections in Polar Coordinates). The graph of any polar equation in the form $r = \dfrac{ed}{1 + e\cos(\theta - \theta_0)}$, for constants $d, e > 0$ and θ_0, is a conic section. If $0 < e < 1$, the graph is an ellipse, if $e = 1$, the graph is a parabola, and for $e > 1$, the graph is a hyperbola. Identify each of the following according to this result, and graph:

(a) $r = \dfrac{2}{1 - \cos\theta}$
(b) $r = \dfrac{10}{2 + \cos\theta}$
(c) $r = \dfrac{3}{1 + 2\cos\theta}$

11. Estimate the number of times that the curve $r = \theta$ intersects the curve $r = 9\sin(3\theta)$. In general, whats the character of the polar curve $r = \theta$?

12. Estimate the number of times that the curve $r = \sqrt{\theta}$ intersects the curve $r = 3\sin(2\theta)$.

13. Sketch the curve defined by the equation $x^4 + x^2y - y^6 + 5 = 0$.

14. In how many points do the ellipses

$$\frac{(x-1)^2}{2^2} + \frac{y^2}{1^2} = 1 \text{ and } \frac{(x-2)^2}{1^2} + \frac{(y-2)^2}{2^2} = 1$$

intersect? Identify the coordinates of the intersection points as best as you can graphically.

5 Computer Algebra

Mathematica has the capability to do most of the algebraic and symbolic manipulations with which you're already familiar, including factoring, recognizing identities, canceling common factors in quotients, and solving equations. This Chapter introduces most of the commands you'll use to do symbolic manipulation.

5.1 Working With Expressions

In this section, we'll see how *Mathematica* commands can be used to manipulate expressions. We'll discuss each of these commands according to the type of expression with which you might be working, be it polynomial, rational, trigonometric, exponential, of logarithmic.

5.1.1 Polynomials

Polynomials of a single variable are expressions of the form

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

where x is the independent variable in the expression, n is a nonnegative integer, and a_n, a_{n-1}, …, a_0 are constants.

Polynomials are expanded by hand with the "FOIL" method that students know, and factored using ad-hoc techniques that require some combination of cleverness and possibly guessing. *Mathematica* does each of these very well.

5.1.1.1 Expand and Factor

Consider entering the following polynomial in its factored form:

▶ `(x-2)(x-3)(x+1)^2`

▷ $(x-3)(x-2)(x+1)^2$

The input is not processed in any way, except that *Mathematica* has rearranged the order of the factors. The **Expand** command forces the expression to be multiplied out fully.

▶ `Expand[%]` `(* % represents the last output *)`[9]

▷ $x^4 - 3x^3 - 3x^2 + 7x + 6$

The inverse operation of factoring a polynomial – forming it into a product of lower degree factors – is performed as best as possible using the **Factor** command

▶ `Factor[%]`

▷ $(x-3)(x-2)(x+1)^2$

By default, *Mathematica*'s **Factor** command only attempts to factor polynomials having (real) integer coefficients into terms that also have only integer coefficients. Thus neither of the following accomplishes much.

▶ `Factor[x^2-5]` `(* factoring requires` $\sqrt{5}$ `*)`

▷ $x^2 - 5$

▶ `Factor[x^2+1]` `(* factoring requires` i `*)`

▷ $x^2 + 1$

[9] We usually discourage the use of the % sign in notebooks, but because we demonstrate so many short sequences, one after the other, we'll break our own rules when it's convenient to do so – as it is now.

If the coefficients of a polynomial are rational, as is the case with

$$p(x) = \left(3x + \frac{1}{2}\right)\left(x - \frac{4}{5}\right)\left(4x - \frac{1}{3}\right)$$

Expand produces an answer with integer coefficients and a lead, rational coefficient $\frac{1}{30}(360x^3 - 258x^2 - 29x + 4)$, and then conveniently multiplies out the result.

► `Expand[(3x+1/2)(x-4/5)(4x-1/3)]`

▷ $12x^3 - \frac{43x^2}{5} - \frac{29x}{30} + \frac{2}{15}$

This result can be combined into factors, all having integer coefficients

► `Factor[%]`

▷ $\frac{1}{30}(5x - 4)(6x + 1)(12x - 1)$

Factor will still work when polynomials have arbitrary real coefficients, such as π or e

► `Expand[(x-Pi/2)(x+E/5)^3]`

▷ $x^4 - \frac{\pi x^3}{2} + \frac{3ex^3}{5} - \frac{3}{10}e\pi x^2 + \frac{3e^2x^2}{25} - \frac{3}{50}e^2\pi x + \frac{e^3x}{125} - \frac{e^3\pi}{250}$

► `Factor[%]`

▷ $-\frac{1}{250}(\pi - 2x)(5x + e)^3$

Expand and **Factor** can be used when coefficients are approximate values.

► `Expand[(2.23 x+1.5) (5.07x-7.36)]`

▷ $11.3061x^2 - 8.8078x - 11.04$

► `Factor[%]`

▷ $11.3061(x - 1.45168)(x + 0.672646)$

The last expressions above involve approximate numbers, and thus we should take all the algebra above as *only approximations*. In fact, although $x = 7.36/5.07$ must clearly be a root of the original expression $(2.23x+1.5)(5.07x-7.36)$, neither this nor its expansion and subsequent factoring will evaluate *exactly* as zero when $x = 7.36/5.07$.

Finally, **Factor** *will* use complex numbers if at least one of the coefficients of the polynomial is complex. For example, the polynomial $p(x) = (5 - 3i) + (-4 + i)x + (1 - i)x^2$ has complex coefficients and can be easily factored.

► `Factor[(5-3I)+(-4+I)x+(1-I)x^2]`

▷ $((1 + i) - ix)((1 + i)x + (1 - 4i))$

5.1.1.2 The Extension Option for Factor

As we've seen, **Factor** will not produce an output with complex coefficients unless the polynomial has at least one complex coefficient. Thus

► `Factor[x^2+1]`

▷ $x^2 + 1$

However, we can force **Factor** to consider factoring an expression over the complex numbers by specifying the **Extension** option. (The name comes from the mathematical notion

that we must extend or enlarge the real numbers to include the imaginary number $i = \mathbf{I}$ in order to find factors.)

▶ `Factor[x^2+1, Extension→{I}]`

▷ $(x - i)(x + i)$

The **Extension** option can be used quite cleverly, in fact, if you know the form of the factors. For example, neither $x^2 + 5$ nor $x^2 - 5$ can be factored without help, since the first relies on the irrational number $\sqrt{5}$ and the second must have available both $\sqrt{5}$ and the complex $i = \mathbf{I}$.

▶ `Factor[x^2-5, Extension→{Sqrt[5]}]`

▷ $-\left(\sqrt{5} - x\right)\left(x + \sqrt{5}\right)$

▶ `Factor[x^2+5, Extension→{I,Sqrt[5]}]`

▷ $\left(\sqrt{5} - ix\right)\left(\sqrt{5} + ix\right)$

5.1.1.3 Simplify and FullSimplify

A command related to **Factor** and **Expand** is **Simplify**. Given an expression, **Simplify** performs a number of possible operations involving **Factor** and **Expand**, attempting to produce what you'd probably consider the simplest way of rewriting the expression.

For example, given the polynomial $x^2-2x+1 = (x-1)^2$, *Mathematica* chooses to simplify it with the shorter, factored expression

▶ `Simplify[x^2-2x+1]`

▷ $(x - 1)^2$

On the other hand, while the polynomial x^3+2x^2-2x-1 factors as $(x-1)(x^2+3x+1)$, *Mathematica* chooses its expression as a polynomial to be simpler, primarily because the polynomial has only four terms, while the factored form has five!

▶ `Simplify[x^3+2x^2-2x-1]`

▷ $x^3 + 2x^2 - 2x - 1$

A close cousin of **Simplify** is **FullSimplify**. It attempts a somewhat wider range of possible simplifications and *may* produce a different and perhaps more useful result.

▶ `FullSimplify[x^3+2x^2-2x-1]`

▷ $(x - 1)(x(x + 3) + 1)$

It's not clear that the result above is either helpful of preferred. However, whenever **Simplify** does not seem to do much to help you, keep in mind that **FullSimplify** is ready and waiting to step in to give it a try.

5.1.2 Rational Functions

Rational functions have the form $\dfrac{p(x)}{q(x)}$, where p and q are polynomials. Rational functions not only have zeroes (when $p(x) = 0$), but some real values may not be in the domain (when $q(x) = 0$).

5.1.2.1 Expand and Factor

Expand can be used for rational functions. It produces a result in which every term of the polynomial p appears divided by q

```
► Expand[(1+2x+x^3)/(1+x)^2]
```

$$\triangleright \frac{x^3}{(x+1)^2} + \frac{2x}{(x+1)^2} + \frac{1}{(x+1)^2}$$

The associated command **ExpandAll** does a little more, expanding the denominator as well.

```
► ExpandAll[(1+2x+x^3)/(1+x)^2]
```

$$\triangleright \frac{2x}{x^2+2x+1} + \frac{1}{x^2+2x+1} + \frac{x^3}{x^2+2x+1}$$

If **Factor** is applied to a rational function, it will factor both the numerator and denominator

```
► Factor[(x^3-1)/(x^2-4)]
```

$$\triangleright \frac{(x-1)\left(x^2+x+1\right)}{(x-2)(x+2)}$$

Should there be common factors between the numerator and denominator, **Factor** will usually remove them in the process

```
► Factor[(x^3 - 1)/(x^2 - 1)]
```

$$\triangleright \frac{x^2+x+1}{x+1}$$

5.1.2.2 Together and Apart

We often combine algebraic expressions involving quotients over a common denominator. The **Together** command performs this function.

```
► Together[x/(x^2-4) - 5/(2x+3)]
```

$$\triangleright \frac{-3x^2+3x+20}{(2x+3)\,(x^2-4)}$$

The reverse process, of decomposing a rational function into what is called its *partial fractions expansion*, is accomplished by the **Apart** command.

```
► Apart[%]
```

$$\triangleright \frac{1}{2(x+2)} - \frac{5}{2x+3} + \frac{1}{2(x-2)}$$

Apart is also useful if you wish to force division of an improper rational function (where the degree of numerator is at least as large at the degree of the denominator).

```
► Apart[(2x^4-5x+2)/(3x-1)]
```

$$\triangleright \frac{2x^3}{3} + \frac{2x^2}{9} + \frac{2x}{27} + \frac{29}{81(3x-1)} - \frac{133}{81}$$

5.1.2.3 Numerators and Denominators

Mathematica provides commands to work separately with the numerator and denominator of a rational function (or, any quotient expression, for that matter). The list is short and, curiously, self-explanatory as to what each of these commands does: **Numerator**, **Denominator**, **ExpandNumerator**, and **ExpandDenominator**.

For example, we would have

```
► Numerator[(1+2x+x^3)/(1+x)^2]
```

▷ $x^3 + 2x + 1$

5.1.3 Trigonometric Functions

The six, standard trigonometric functions are sine, cosine, tangent, cotangent, secant, and cosecant. The commands **Simplify**, **Expand**, and **Factor** can certainly be used with trigonometric functions.

However, rewriting trigonometric expressions "correctly" remains a mysterious goal for students, due to the fact that each of the last four of these functions is defined in terms of the first two, and the first two are related by the identity $\sin^2(x) + \cos^2(x) = 1$.

In short, there may be no best way to rewrite trigonometric expressions unless there's some obvious identity that appears.

Let's see how *Mathematica* does.

5.1.3.1 Simplify

The **Simplify** command will, by default, use common trigonometric identities such as $\sin^2 x + \cos^2 x = 1$.

▶ `Simplify[Sin[x]^2+Cos[x]^2]`

▷ 1

Simplify will use other known identities, such as the double angle formula $\sin(2x) = 2\sin(x)\cos(x)$.

▶ `Simplify[2Sin[x] Cos[x]]`

▷ $\sin(2x)$

On the other hand, **Simplify** always prefers a short output, so we have the following, even though $\sin(x)\cos(x) = \frac{1}{2}\sin(2x)$.

▶ `Simplify[Sin[x] Cos[x]]`

▷ $\sin(x)\cos(x)$

There are some variations on simplification when working with trigonometric functions. One command that works with the output above is **TrigReduce**, which prefers replacing products of $\sin(x)$ and $\cos(x)$ by expressions involving $\sin(2x)$ and the like.

▶ `TrigReduce[Sin[x] Cos[x]]`

▷ $\frac{1}{2}\sin(2x)$

The online help provides other possibilities for alternate simplifications, one of which involves changing **Simplify**'s default setting of **Trig→True** to **Trig→False**, although we'll not demonstrate that here.

5.1.3.2 Expand and Factor

The commands **Expand** and **Factor** usually do not rely on identities and double-angle formulas and may seem ineffective when you first try to use them.

▶ `Factor[Sin[2x]Cos[3x]-Sin[x]Cos[4x]]`

▶ $\sin(2x)\cos(3x) - \sin(x)\cos(4x)$

▶ `Expand[Sin[x]^2+Sin[2x]]`

▷ $\sin^2(x) + \sin(2x)$

But you can force **Factor** and **Simplify** to be more creative with trigonometric expressions by specifying the option **Trig→True**.

▶ `Factor[Sin[2x]Cos[3x]-Sin[x]Cos[4x],Trig→True]`

▶ $\sin(x)(\cos(x) - \sin(x))(\sin(x) + \cos(x))$

▶ `Expand[Sin[x]^2+Sin[2x],Trig→True]`

▷ $\dfrac{\sin^2(x)}{2} - \dfrac{1}{2}\cos^2(x) + 2\sin(x)\cos(x) + \dfrac{1}{2}$

In each of the output expressions above, you'll notice that the output terms are presented only in terms of $\sin(x)$ and $\cos(x)$.

5.2 Computer Algebra Technicalities

When you're working with real numbers, there are definition issues that get in the way of simplifications and algebra. And, as we mentioned several times already, *Mathematica* considers its expressions over the complex numbers, where more interesting algebraic identities exist.

Either of these concerns can sometimes get in the way of what you think *Mathematica* should be doing. You may need to know about either a special command to handle the situation, or a special option that could be enforced to control specific behavior.

5.2.1 Powers and Roots

When x is a nonnegative real number, we know that $\sqrt{x^2} = x$, because the regular rules of exponentiation assert that the order in which we square and take the square root doesn't matter. Often, we'd write either

$$\sqrt{x^2} = (x^2)^{1/2} = x^{2\cdot\frac{1}{2}} = x^1 = x$$

or

$$\sqrt{x^2} = \left(\sqrt{x}\right)^2 = (x^{1/2})^2 = x^{\frac{1}{2}\cdot 2} = x^1 = x$$

However, for a negative number, this interchange of operations would be disastrous (e.g., the first sequence above makes sense for any x, but the second sequence makes no sense if x is negative). This is the reason that *Mathematica* will not simplify $\sqrt{x^2}$ as x

▶ `Simplify[Sqrt[x^2]]`

▷ $\sqrt{x^2}$

One way to make *Mathematica* reduce $\sqrt{x^2}$ as x is to use the command **PowerExpand**, which essentially tells *Mathematica* that it's OK to play games with the exponents as we did above, forgetting about any possible domain issues. Thus

▶ `PowerExpand[Sqrt[x^2]]`

▷ x

5.2.2 Assumptions

There is a preferred method that you can use to force *Mathematica* to simplify $\sqrt{x^2}$ as x, while staying more faithful to the underlying mathematics. We will use the **Assumptions** option of **Simplify** to alert *Mathematica* that we wish to assume that x is (real and) nonnegative.

▶ `Simplify[Sqrt[x^2],Assumptions→{x≥0}]`

▷ x

A similar situation occurs in trying to simplify $\ln(x^2)$ as $2\ln(x)$, since this is true only if x is (real and) positive. As a result, *Mathematica* will not simplify $\ln(x^2)$ (remember below that *Mathematica* uses the name **Log** for the natural logarithm).

```
► Simplify[Log[x^2]]
```

▷ $\log\left(x^2\right)$

```
► Simplify[Log[x^2],Assumptions→{x>0}]
```

▷ $2\log(x)$

As you see above, the format of an assumption is that of a logical condition. Conditions can be quite detailed or state very specific domain specification such as "is a real number" or "is an integer."

For example, if x is any *real* number, we can write $\sqrt{x^2} = |x|$.

```
► Simplify[Sqrt[x^2],Assumptions→{x∈Reals}]
```

▷ $|x|$

(The symbol $\in$ above, used mathematically to denote the phrase "is an element of," was entered using one of the Typesetting palettes. Alternatively, you could write **Element[x,Reals]** for the assumption.)

Similarly, if x is a complex number written in the form $x = a + bi$ with a and b real, then we still have $\sqrt{x^2} = x$ as long as the real part of x is positive (if $a > 0$).

```
► Simplify[Sqrt[x^2],Assumptions→{Re[x]>0}]
```

▷ x

Assumptions are a very powerful tool in *Mathematica*. They're especially useful for integration purposes. Be sure you take a look at the online documentation. Start with **?Assumptions**.

5.3 Solving Equations Symbolically

Mathematica's capability to solve equations is quite impressive, whether we're interested in solving a *single* equation of a *single* variable (using either numeric or symbolic terms) or a *system* of equations in any *number* of variables (also using either numeric or symbolic terms). Not surprisingly, the basic command to use is **Solve**.

5.3.1 Syntax of the Solve Command

First we consider a simple problem, solving a linear equation of a single variable. Here we "solve for x in the equation $2x + 5 = 9$."

Mathematica's **Solve** command handles this type of problem directly.

```
► Clear[x]
  Solve[2x+5==9,x]
```

▷ {{x→2}}

The syntax of the **Solve** *input* is to first enter the equation to solve, and then enter the variable for which to solve. (The variable can be omitted if only one variable appears in the equation.) Two key syntax points to keep in mind here are

- The equation is entered using the *double* equal sign ==. If you use a single equal

sign (which means immediate assignment), you'll either get an error or a result that is meaningless to you (although perfectly sensible from *Mathematica*'s point of view).
- The variable of the equation must not have any previous meaning, so it's recommended that you use **Clear** before using **Solve**.

As for the *output* of **Solve**, we can certainly guess that $x = 2$ is the solution of $2x+5 = 9$, but what's with all the syntax?

First, the result of **Solve** will always be a *list*. The equation $2x + 5 = 9$ has only one solution, but other equations may have more than one solution and a list is a convenient way to report all of them. That explains the outer level of the curly braces **{** and **}**.

The list above has only one item in it, **{x→2}**. This itself is a list containing just one substitution rule, **x→2**. The reason list syntax is used here is to allow for more than one substitution rule, to handle situations where more than one variable appears in the solution (you'll see this shortly when we discuss solutions to system of equations). That explains this inner level of curly braces **{** and **}**.

Finally, why is the result reported as the substitution rule **x→2** and not simply as "**2**"? The substitution rule syntax will make it easy to evaluate expressions in the future using substitution ("**/.**"); and if the equation has more than one variable, how would you know which solution belonged to which variable?

For example, we would check that we really do have a solution to the equation with

```
▶ 2x+5==9 /. {x→2}
```

▷ True

5.3.2 Handling the Result of Solve

We'll use *Mathematica* syntax to keep track of solutions for us (so we can use them later) by assigning it/them to a variable.

```
▶ Clear[x]
  solutions = Solve[2x+5==9,x]
```

▷ {{x→2}}

Since **solutions** is a list (of only one element), we need a way to extract one (the first) of its solutions. To do this, we use *list subscript notation* to pick out an item by its position in the list.

```
▶ solutions[[1]]   (* gets the first solution *)
```

▷ {x→2}

The double square brackets [[and]] are list specific syntax and denote an element of a list by its position. It's now easy to evaluate the equation with the (only) result found by **Solve**.

```
▶ 2x+5==9 /. solutions[[1]]
```

▷ True

What if you want to "get the number 2," and not a substitution rule for it? That's easy, but it uses a bit of syntax trickery. You form the expression x and you substitute 2 into x in this expression.

```
▶ Clear[x]   (* x must be symbolic to use this *)
  x /. solutions[[1]]
```

▷ 2

This is sometimes called the "dummy variable" trick and, despite its being a syntactic curiosity for most students, it's exactly what we need.

5.3.3 Multiple Solutions

The discussion above now extends easily to solving equations having more than one solution. Consider the quadratic equation $x^2 - 3x + 1 = 0$, where its roots are found with

```
► solutions = Solve[x^2-3x+1==0,x]
```

▷ $\left\{\left\{x \to \frac{1}{2}\left(3 - \sqrt{5}\right)\right\}, \left\{x \to \frac{1}{2}\left(3 + \sqrt{5}\right)\right\}\right\}$

The output above is a list of *two* items (since there are *two* solutions), each of which is itself a list of *one* item, namely a substitution rule for a solution.

```
► solutions[[1]]
```

▷ $\left\{x \to \frac{1}{2}\left(3 - \sqrt{5}\right)\right\}$

```
► solutions[[2]]
```

▷ $\left\{x \to \frac{1}{2}\left(3 + \sqrt{5}\right)\right\}$

To see the solutions directly, we must use the dummy variable trick.

```
► x /. solutions[[1]]
```

▷ $\frac{1}{2}\left(3 - \sqrt{5}\right)$

```
► x /. solutions[[2]]
```

▷ $\frac{1}{2}\left(3 + \sqrt{5}\right)$

Mathematica will report repeated roots of a polynomial as repeated substitution rules.

```
► Solve[x^3-2x^2+x==0,x]
```

▷ $\{\{x \to 0\}, \{x \to 1\}, \{x \to 1\}\}$

Finally, you should yet again be aware that *Mathematica* freely uses complex numbers, so don't be surprised by

```
► Solve[x^2-3x+5==0,x]
```

▷ $\left\{\left\{x \to \frac{1}{2}\left(3 - i\sqrt{11}\right)\right\}, \left\{x \to \frac{1}{2}\left(3 + i\sqrt{11}\right)\right\}\right\}$

5.3.4 Symbolic Solutions

Equations can certainly have symbolic parameters, such as we see in solving the equation $2x + 5 = a$ for x, where a is an unknown constant.

```
► Clear[a,x]
  Solve[2x+5==a,x]
```

▷ $\left\{\left\{x \to \frac{a-5}{2}\right\}\right\}$

We could go even a little farther. If $y = f(x) = mx + b$ is a linear function, then we can solve this equation for y in terms of x with

```
► Clear[m,x,b]
  Solve[y==m*x+b,x]   (* notice the explicit use of * here *)
```

▷ $\left\{\left\{x \to \dfrac{y-b}{m}\right\}\right\}$

Mathematica certainly knows the quadratic formula in terms of symbolic coefficients a, b, and c.

```
► Clear[a,b,c,x]
  Solve[a*x^2+b*x+c==0,x]
```

▷ $\left\{\left\{x \to \dfrac{-\sqrt{b^2-4ac}-b}{2a}\right\}, \left\{x \to \dfrac{\sqrt{b^2-4ac}-b}{2a}\right\}\right\}$

If necessary, *Mathematica* can produce (symbolic) solutions to polynomial equations up through degree 4, although the symbolic solutions of a cubic $ax^3 + bx^2 + cx + d = 0$ will easily fill up your computer screen.

5.3.5 Solve and High Degree Polynomials

For polynomial equations involving a single variable of degree 5 or higher, **Solve** will only succeed if the polynomial can be factored into lower-degree polynomials, each having degree four or less.

For example, the polynomial $p(x) = x^5 - x^4 - 1$ factors easily:

```
► Factor[x^5-x^4-1]
```

▷ $\left(x^2 - x + 1\right)\left(x^3 - x - 1\right)$

Consequently, explicitly solving the equation $x^5 - x^4 - 1 = 0$ is possible in this case since we can solve both of the equations $q(x) = x^2 - x + 1 = 0$ and $r(x) = x^3 - x - 1 = 0$. Thus **Solve** successfully finds all five solutions of the equation $x^5 - x^4 - 1 = 0$ (we show only the fifth solution symbolically, but we also can see numeric approximations for all of the solutions using **N**).

```
► solutions = Solve[x^5-x^4-1==0,x];
  solutions[[5]]
```

▷ $\left\{x \to -\frac{1}{6}\left(1 - i\sqrt{3}\right)\sqrt[3]{\frac{27}{2} - \frac{3\sqrt{69}}{2}} - \frac{\left(1+i\sqrt{3}\right)\sqrt[3]{\frac{1}{2}\left(9+\sqrt{69}\right)}}{2\,3^{2/3}}\right\}$

```
► N[solutions]
```

▷ $\{\{x \to 0.5 + 0.866025i\}, \{x \to 0.5 - 0.866025i\},$
$\{x \to 1.32472\}, \{x \to -0.662359 + 0.56228i\},$
$\{x \to -0.662359 - 0.56228i\}\}$

If a polynomial of degree 5 or higher does not factor into lower degree polynomials (each of degree four or less), **Solve** will fail. Its output may be syntactically frightening (and we only show the first few terms of the output below)

```
► solutions = Solve[x^5-x^3-1==0,x]
```

▷ $\left\{\left\{x \to \text{Root}\left[\#1^5 - \#1^3 - 1\&, 1\right]\right\}, \ldots\right\}$

Each of the "roots" above is written using the **Root** command (usually, you will not actually use the **Root** command yourself). The first that is shown represents "the first root of the equation $x^5 - x^3 - 1$," with the **#1** notation representing the (first) variable. There is no closed form expression for it.

Nevertheless, you can still see numerical approximations for the roots

```
► N[solutions]
```

▷ $\{\{x \to 1.23651\}, \{x \to -0.959048 - 0.428366i\},$
$\{x \to -0.959048 + 0.428366i\}, \{x \to 0.340795 - 0.785423i\},$
$\{x \to 0.340795 + 0.785423i\}\}$

5.3.6 Solve and Non-Polynomial Equations

Not every equation of a single variable involves only polynomial (or, for that matter, rational) functions. **Solve** does a reasonable job on many algebraic equations, such as $\frac{\sqrt{x}}{x^2+1} = 12$, which has two complex solutions (although we'll not show the symbolic solutions, which fill up a full page of output).

```
► solutions = Solve[Sqrt[x]/(1+x^2)==12,x];
  N[solutions]
```

▷ $\{\{x \to 0.0294564 - 0.97053i\}, \{x \to 0.0294564 + 0.97053i\}\}$

However, **Solve** has significant problems with equations involving transcendental functions. Even simple equations such as the following have no symbolic "solution," and your input is returned unevaluated.

```
► Solve[Sin[x]==x,x]
```

▷ Solve::tdep: The equations appear to involve the variables
to be solved for in an essentially non-algebraic way.
Solve[Sin[x] == x, x]

This equation $\sin(x) = x$ has a single solution of $x = 0$, yet to find a *symbolic* solution, our only hope is perhaps to apply the inverse sine function $\sin^{-1}$ to both sides of the equation, producing the equation $\sin^{-1}(\sin(x)) = \sin^{-1}(x)$, or $x = \sin^{-1}(x)$. This equation is not significantly different from the first equation, and there's little hope of proceeding on the basis of algebraic canceling or applying identities.

On the other hand, if a is a symbolic quantity, we have the following

```
► Solve[Sin[x]==a,x]
```

▷ Solve::ifun: Warning: Inverse functions are being used
by Solve, so some solutions may not be found ...
$\{\{x \to \sin^{-1}(a)\}\}$

Here again, *Mathematica* applied the inverse sine function to both sides of the equation $\sin(x) = a$ to produce $x = \sin^{-1}(a)$. This is a correct solution, as long as you're looking for a solution satisfying $-\frac{\pi}{2} \le x \le \frac{\pi}{2}$ (this is the substance of the warning received above). Otherwise, a numeric method (discussed later in the text) is likely our best option to find a solution of interest.

5.3.7 Solving Systems of Equations

Consider solving two equations in two unknowns, such as

$$\begin{cases} 3x + 8y = 5 \\ 5x + 2y = 7 \end{cases}$$

The **Solve** command can handle such systems of equations using an extended syntax. The individual equations are input as items of a *list*, and the variables in which the system is to be solved also are input as items of a *list*.

```
► Clear[x,y]
  Solve[{3x+8y==5,5x+2y==7},{x,y}]
```

▷ $\left\{\left\{x \to \frac{23}{17}, y \to \frac{2}{17}\right\}\right\}$

Solutions can be generated in purely symbolic terms, so that the same system in which the constants 5 and 7 are replaced by symbolic constants a and b can be solved

$$\begin{cases} 3x + 8y = a \\ 5x + 2y = b \end{cases}$$

```
► Clear[x,y,a,b]
  Solve[{3x+8y==a,5x+2y==b},{x,y}]
```

▷ $\left\{\left\{x \to \frac{1}{17}(4b - a), y \to \frac{1}{34}(5a - 3b)\right\}\right\}$

The equation or system of equations specified for **Solve** is not required to be linear. For example, the following system has two real and two complex solutions. However, we'll display them only numerically (the symbolic display is quite large, if not interesting on its own).

$$\begin{cases} 3x - y^2 = 4 \\ 2x^2 + y = 9 \end{cases}$$

```
► solutions = Solve[{3x-y^2==4,2x^2+y==9}, {x,y}];
  N[solutions,5]
```

▷ $\{\{x \to -2.1343 - 0.3784i, y \to 0.17570 - 3.2301i\},$
$\{x \to -2.1343 + 0.3784i, y \to 0.17570 + 3.2301i\},$
$\{x \to 1.9539, y \to 1.3645\}, \{x \to 2.3147, y \to -1.7159\}\}$

5.3.7.1 Extracting Solutions of Systems

The same dummy variable trick we demonstrated earlier can be used to extract solution values. For example, with the system used to open this section

$$\begin{cases} 3x + 8y = 5 \\ 5x + 2y = 7 \end{cases}$$

we have

```
► Clear[x,y]
  solutions = Solve[{3x+8y==5,5x+2y==7},{x,y}];
  x /. solutions[[1]]
```

▷ $\frac{23}{17}$

```
► y /. solutions[[1]]
```

▷ $\frac{2}{17}$

5.3.7.2 Existence of Solutions

Remember that some systems of equations will have no solution, such as the inconsistent system with equations $x + y = 1$ and $x + y = 2$.

```
► Solve[{x+y==1,x+y==2},{x,y}]
```

▷ {}

A list of possible solutions has been returned as expected, but it has no elements. Always watch for empty returns, signifying that the system has no solutions.

Also, watch for possible dependencies among equations. For example, we have

```
► Solve[{x+y==1,2x+2y==2},{x,y}]
```

▷ Solve::svars: Equations may not give solutions
for all ¨solve¨ variables.
$\{\{x \to 1 - y\}\}$

Since the second equation $2x + 2y = 2$ above is a simple multiple of the first equation $x + y = 1$, we've actually asked **Solve** to find a solution to the *single* equation $x + y = 1$. The message received and the result reported show that the y variable has been treated as a symbolic parameter and the equation has been solved for x in terms of y.

However, there will be times when it is desired to find a solution to a system of n equations in m variables with $n < m$. The example above is exactly such a situation where we ask to solve (what is essentially) one equation in the two unknowns x and y, and *Mathematica* selected y to be a parameter to describe the solution.

If you use **Solve** and specify fewer variables to solve for than are present in the equations, you are indicated that the variables you do not explicitly name to solve for to be treated as parameters in the solution, such as in

```
► Solve[{x+y+z==1,x+2y+3z==3},{x,y}] (* x and y named, z symbolic *)
```

▷ $\{\{x \to z - 1, y \to -2(z - 1)\}\}$

One possible application of this result is that the intersection of the two planes $x+y+z = 1$ and $x + 2y + 3z = 3$ in three-space consists of a line parameterized as the set of points $(z - 1, -2z + 2, z)$, for z real.

5.4 Solving Equations Numerically

In general, **Solve** is a limited command, used best with low-degree polynomials, rational functions, and some algebraic functions. Each maintenance update or new version of *Mathematica* seems to extend its capability; but the general problem of finding complete, symbolic solutions to equations remains a difficult one.

Numerical techniques have been developed over the course of literally hundreds of years of mathematics to find *approximate* solutions to (specific types of) equations. The subject of *Numerical Analysis* has a deep history and is well worth studying on its own. *Mathematica* implements many of its techniques, although they are disguised in just two powerful commands that we now describe.

5.4.1 NSolve

NSolve is the command you should use when solving one or more *polynomial* equations and you're satisfied with approximate, numerical solutions (rather than exact solutions).

The input syntax and output format of **NSolve** is exactly the same as **Solve**. Unlike **Solve**, however, there is no practical degree restriction for the equation to solve.

```
► NSolve[x^5-x^3-1==0,x]
```

▷ $\{\{x \to -0.959048 - 0.428366i\}, \{x \to -0.959048 + 0.428366i\},$
$\{x \to 0.340795 - 0.785423i\}, \{x \to 0.340795 + 0.785423i\},$
$\{x \to 1.23651\}\}$

NSolve can also be used to find solutions to systems of equations, with exactly the syntax of **Solve** for such systems. This will not be shown here, however.

5.4.2 FindRoot

The **FindRoot** command is an extremely powerful tool used to find a solution of an equation, and thou should expect to use it often when working problems in *Mathematica*.

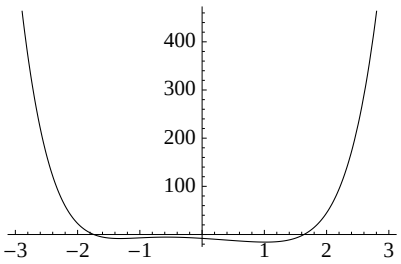
Unlike **Solve** and **NSolve**, however, **FindRoot** does not find "all solutions" to an equation; rather, it must be instructed to look for a *single solution* to an equation near where a solution is known (or is virtually certain) to exist.

More importantly, however, **FindRoot** works *only* with approximate numerical values and cannot find solutions of equations symbolically. While this may seem to be a limitation, the fact is that finding a symbolic solution to an equation may not only be difficult but may simply be impossible; and in practice, this is more than sufficient.

Example. We wish to find the *zeroes* or *roots* of the function $f(x) = x^6 + 5x^3 - 8e^x$. That is, we wish to find all solutions to the equation $f(x) = 0$.

We begin by examining a simple graph of f to get a sense of where on the number line f might have a zero.

```
▶ Clear[f]
  f[x_] := x^6 + 5x^3 - 8Exp[x]
  Plot[f[x],{x,-3,3}]
```



In rather gross terms, we can see that f crosses the axis near $x = 1.6$ and again near $x = -1.8$. We'll first ask **FindRoot** to look for an approximation for the zero near $x = 1.6$ with

```
▶ solution = FindRoot[f[x]==0, {x,1.6}]
```

▷ {x→1.63614}

We should now believe that the number 1.63614 provides a suitable numerical approximation of a zero of f. (We'll address the issue of working with more digits of precision shortly).

Notice that the output of **FindRoot** is a list containing a single substitution rule of $x \to 1.63614$. Thus we can see if we really have found a zero for f by simply using this substitution rule

```
▶ f[x] /. solution
```

▷ -1.77636×10^{-14}

Notice we did not get an exact zero. **FindRoot** only works with approximate numerical values. However, we can accept that the approximate numerical value found by **FindRoot**

of $x \approx 1.63614$ provides a suitable estimate of the zero of f near 1.6 – and we can see that f is essentially zero at this value ($f(x) \approx -0.0000000000000177636$). Although the result is not exactly zero, we're quite satisfied.

To "find" the second root we see in the graph near $x = -1.8$, we have:

```
► solution = FindRoot[f[x]==0, {x,-1.8}]
```

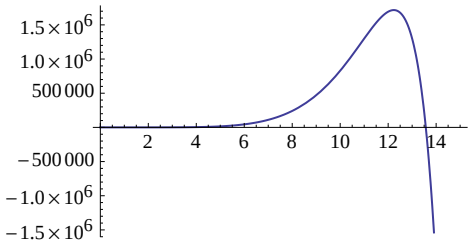
▷ {x→−1.73985}

Have we completed the problem – to find all zeroes of the function f? On the one hand, the best (and, frankly, only) option we have is to provide numerical approximations of the roots, and with that in mind, it *appears* we've found the zeroes.

But how do you know that we've found *all* the zeroes? In fact, there is a third zero for f, and you'll need to apply your mathematical knowledge of the growth of the individual terms x^6 and e^x to recognize that the third root of f will lie farther to the *right* of the origin. Do you know why?

Once you determine that the third root lies to the right of the origin, then you have to come up with a rough approximation of it. The following graph – after some experimentation was done – provides the information we need

```
► Plot[f[x],{x,0,15}]
```



The third zero is near 13.5, thus we can use

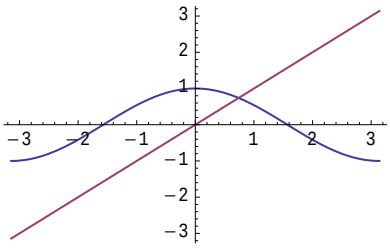
```
► FindRoot[f[x]==0, {x,13.5}]
```

▷ {x→13.5695}

Example. Find all solutions of the equation $\cos x = x$.

Solving the equation $\cos x = x$ is the equivalent to finding where the graphs of $f(x) = \cos x$ and $g(x) = x$ intersect. We'll provide a quick sketch near the origin.

```
► Plot[{Cos[x],x},{x,-π,π}]
```



We can see that the graphs intersect somewhere between perhaps 0.5 and 1.0 – and because these are such simple functions that we so well understand, we know that this is the *only* intersection of the graphs.

To solve the equation $\cos x = x$, then, we use **FindRoot**, asking it to investigate the area nearby our choice of $x = 0.75$, subsequently finding that a solution to the equation exists at $x \approx 0.739085$.

```
▶ FindRoot[Cos[x]==x, {x,0.75}]
▷ {x→0.739085}
```

5.4.2.1 Troubleshooting FindRoot

To properly apply **FindRoot**, you must first have some knowledge of where solutions to equations are likely to be found. Graphing, together with some mathematical understanding of the situation, provides a simple technique to help you identify *where* solutions might be found.

At this point, we'll defer any extended discussion about the inner workings and technical issues related to **FindRoot**. For most of what you'll see in the sequel, you already have enough to work with. However, you should have already suspected that the following are the main issues in properly using **FindRoot**.

(a) How *sensitive* is **FindRoot** to your choice of a guess as to where a root is located? The answer in general is complicated, unfortunately, but "the closer you can locate a solution to get started, the better."
(b) How *accurate* is the answer produced by **FindRoot**? In general, it's pretty good, but what's pretty good for a Calculus student may not be good enough in an application. If you need more information about precision concerns, you should look into the options **AccuracyGoal**, **PrecisionGoal**, and **WorkingPrecision** that allow you more control over your concerns.
(c) Will **FindRoot** always find the root *nearest* to where you start looking? The answer is, unfortunately, no. If **FindRoot** succeeds, you've found a solution – just possibly not the one you thought you were looking for.
(d) Can **FindRoot** fail? Yes, unfortunately, it can, and sometimes in spectacular fashion.

Other than checking on-line documentation for **FindRoot**, you'll only best appreciate its strengths and weaknesses once you become familiar with Newton's Method in your first Calculus course.

5.4.2.2 FindRoot and Systems of Equations

FindRoot may also be used to find solutions to systems of equations. Consider finding a solution to the simultaneous system:

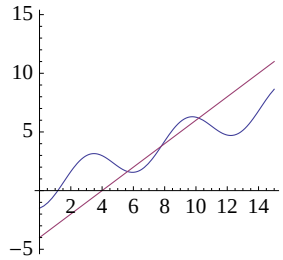
$$\begin{cases} x = 2y + 3\cos x \\ y = x - 4 \end{cases}$$

There is a simple way to solve this particular system: replace y in the first equation with the expression $x - 4$ (from second equation), then solve numerically for x. Using the result, you then determine the value of y. For demonstration purposes, however, suppose we don't immediately see this.

The two equations above can each be understood to define curves in the plane (see Section 4.1). Finding a solution to this system is equivalent to locating points at which the curves intersect.

These two curves are easily drawn with **ContourPlot** (although our choices of ranges for the variables x and y required some experimentation before settling on the rectangle on $0 \le x \le 15$, $-5 \le y \le 15$).

```
► ContourPlot[{x==2y+3Cos[x], y==x-4},
  {x,0,15}, {y,-5,15},
  Frame→False, Axes→True]
```



The curves appear to intersect in three points, near $(6, 2)$, $(8, 4)$ and $(10, 6)$.

In this situation, **FindRoot** implements a two-variable generalization of its one-variable methods. The syntax necessary requires that the equations be specified as a *list*, and that nearby values of likely solutions be specified for of each of the variables.

Using these three estimates for intersections, approximate solutions to the system may be found:

```
► FindRoot[{x==2y+3Cos[x],y==x-4},{x,6},{y,2}]
```

▷ $\{x \to 5.6256, y \to 1.6256\}$

```
► FindRoot[{x==2y+3Cos[x],y==x-4},{x,8},{y,4}]
```

▷ $\{x \to 7.78087, y \to 3.78087\}$

```
► FindRoot[{x==2y+3Cos[x],y==x-4},{x,10},{y,6}]
```

▷ $\{x \to 10.1814, y \to 6.18143\}$

5.5 Exercises

1. Solve each of the following equations or systems of equations.

(a) $x^3 + 2x = 17$

(b) $\begin{cases} 9x + 2y = 17 \\ 5x - 7y = 23 \end{cases}$

(c) $\begin{cases} 3x + \pi y = a \\ 2x - 5y = b \end{cases}$, for constants a and b

(d) $\begin{cases} 3x^2 y + xy = 3 \\ xy^3 - 18y = 9 \end{cases}$

2. Find all common factors of the polynomials $p(x) = x^5 - x^4 - 81x + 81$ and $q(x) = x^5 - 3x^4 - x + 3$.

3. Use **Factor** to verify that

$$\begin{aligned} x^2 - 1 &= (x-1)(x+1) \\ x^3 - 1 &= (x-1)(x^2+x+1) \\ x^4 - 1 &= (x-1)(x+1)(x^2+1) \\ x^6 - 1 &= (x-1)(x+1)(x^2-x+1)(x^2+x+1) \end{aligned}$$

Notice that all coefficients in the fully-factored forms above are ± 1.

Choose any ten values of a positive integer $10 \leq n \leq 100$ at random, and factor $x^n - 1$, for each of your choices. Do you observe that all coefficients of the polynomials in the fully-factored forms are ± 1? What might you conjecture from these observations?

4. Find a polynomial of the form $p(x) = ax^3 + bx^2 + cx + d$ whose graph contains the points $(-1, 1)$, $(0, 3)$, $(3, -4)$ and $(5, 4)$, by using the Solve command. Graph the result over the interval $[-2, 6]$ to see that the right polynomial has been obtained.

5. Any circle in the plane will have an equation of the form $x^2 + y^2 + ax + by + c = 0$, where a, b and c are constants. Find the equation of the circle that goes through the points $(2.3, 3.4)$, $(3.7, 5.8)$ and $(6.1, -1.2)$.

6. Let r be a fixed, positive constant.

(a) Find the equation of the parabola which has vertex at $(0, -2r)$, and goes through the points $(r, 0)$ and $(-r, 0)$. Write your answer in the form $p(x) = ax^2 + bx + c$.

(b) Find the point in the first quadrant where the parabola of part (a) intersects the ellipse having vertices at $(\pm 2r, 0)$ and $(0, \pm r)$.

(c) Check your answers to parts (a) and (b) both numerically and graphically when $r = 1$.

6 Calculus Applications (I)

So far, we've presented the basic tools of *Mathematica* that we use: functions, graphs, and computer algebra. Beginning with this chapter, we'll focus more on mathematical topics. New *Mathematica* commands and syntax will certainly be needed, but will now more naturally arise according to the mathematical subject areas we discuss.

Our choice of topics in this chapter come from the beginning notions of calculus.

6.1 Inverse Functions

Inverse functions are important components of the Calculus toolbox. The natural logarithm is often *defined* as the inverse of the natural exponential, and the inverse trigonometric functions are indispensable tools when it comes to integration.

6.1.1 What Are Inverse Functions?

Definition. *Given a function f defined on a domain D, we say that f has an inverse function if there is a function g defined on the range of f so that $g(f(x)) = x$, for all $x \in D$. Notationally, we write $g = f^{-1}$, or that $g(x) = f^{-1}(x)$, and we call g the inverse of f.*

Two simple examples will help explain the definition.

(a) The linear function $f(x) = 3x + 5$ is defined for all real x. Its range is $\mathbb{R}$, the set of all real numbers. If we define g on $\mathbb{R}$ by setting $g(x) = (x - 5)/3$, we can see that $g(f(x)) = x$, for all real x. The function g is then the inverse function for f.

(b) The natural exponential function $f(x) = e^x$ is defined for all real x and has range $\{x \mid x > 0\}$. We see that the inverse of f is $g(x) = \ln(x)$, defined for all $x > 0$, since $g(f(x)) = \ln(e^x) = x$ for all real x.

Not every function has an inverse. The usual example is to consider $f(x) = x^2$ defined on the domain $D = \mathbb{R}$, the set of all real numbers. Since $f(1) = f(-1) = 1$, there cannot be a function g such that $g(f(x)) = x$. Indeed, if such a g existed, we'd have to have both $1 = g(f(1)) = g(1)$ and $-1 = g(f(-1)) = g(1)$, which is senseless if g is to be a function.

What types of functions f have inverses? The answer is well-known: the function f must be *one-to-one* on its domain D, that is, if whenever we select two values x_1 and x_2 in D and $f(x_1) = f(x_2)$, then it must be the case that $x_1 = x_2$. This technical condition simply states that if we write $y = f(x)$, then *every y in the range of f must arise from a unique x in D.*

Functions that are either strictly increasing or strictly decreasing on their domain necessarily have inverses. Thus, simply examining the graph of f may often be enough to convince you that a function has an inverse.

Note that if f has an inverse function g, then we certainly have $g(f(x)) = x$ for all x in the domain of g. But consider any y in the range of f. Then $x = g(y)$ has the property that $y = f(x) = f(g(y))$, and so the composition $f(g(y)) = y$. Therefore, f is the inverse function of g and we see that *inverse functions must always occur in pairs.* In other words, if $g = f^{-1}$, then $f = g^{-1}$.

6.1.2 Finding Formulas for Inverse Functions

We turn our attention to using *Mathematica* (if possible) to find a formula for an inverse function, should one exist.

If f has an inverse function, it *algebraically* means that the equation $y = f(x)$ can be

solved for x in terms of y. For the function $y = f(x) = 3x + 5$, we solve for x in terms of y as $x = (y - 5)/3$, and so the inverse function is $g(y) = (y - 5)/3$ – although we usually switch the roles of the variables at this point, writing $y = (x - 5)/3$ or $g(x) = (x - 5)/3$.

Thus we can use *Mathematica* to find inverse functions algebraically by using the **Solve** command, *at least in the simple cases where we expect that it will succeed.*

Consider finding an inverse for the linear function $f(x) = 3x + 5$.

```
▶ Clear[f]
  f[x_] := 3x+5
```

To solve the equation $y = 3x+5$ in terms of x is an easy exercise for the **Solve** command.

```
▶ Clear[x,y]
  Solve[y==f[x],x]
```

▷ $\left\{\left\{x \to \dfrac{y-5}{3}\right\}\right\}$

We use this response to define a function g as a function of a variable y using this solution.

```
▶ Clear[g]
  g[y_] := (y-5)/3
```

We can verify that $g(f(x)) = x = f(g(x))$, for all real x, convincing us that f and g really are inverse functions.

```
▶ f[g[x]]
```

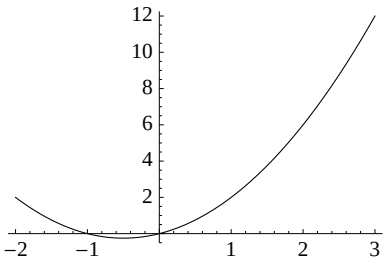
▷ x

```
▶ g[f[x]]
```

▷ x

As a second example, consider the function $f(x) = x^2 + x$. As a function defined on all of $\mathbb{R}$, this *does not* have an inverse function, as we can see from its graph on an interval near to the origin.

```
▶ f[x_] := x^2+x
  Plot[f[x],{x,-2,3}]
```



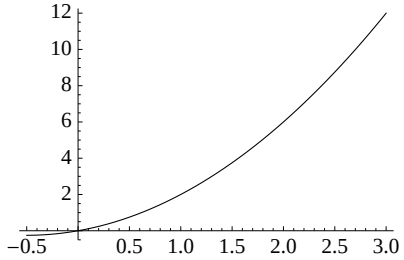
What feature of the graph suggests that there is no inverse for f? The graph shows specifically that $f(-2) = f(1) = 2$, so f is not one-to-one. Said geometrically, the horizontal line $y = 2$ cuts the graph of f in more than one point.

More generally, if any horizontal line intersects the graph of f in more than one point, f cannot have an inverse. Most texts call this *the horizontal line test.*

Nevertheless, the graph of this parabola suggests that the function *will* have an inverse if restricted to either the intervals $(-\infty, -1/2]$ or $[-1/2, \infty)$, since the function will be strictly decreasing or strictly increasing, respectively, on these intervals.

Consider restricting to the interval $[-1/2, \infty)$ and plotting, to demonstrate that the function is increasing:

```
► Plot[f[x],{x,-1/2,3}]
```



Notice that since $f(-1/2) = -1/4$, the range of f with this domain restriction will be exactly $\{y \mid y \geq -\frac{1}{4}\}$. Thus the inverse we're hoping to find will have this set as its domain.

To find the inverse, simply solve for $y = f(x) = x^2 + x$ in terms of x

```
► solutions = Solve[y==f[x],x]
```

▷ $\left\{\left\{x \to \frac{1}{2}\left(-\sqrt{4y+1} - 1\right)\right\}, \left\{x \to \frac{1}{2}\left(\sqrt{4y+1} - 1\right)\right\}\right\}$

Since this discussion of an inverse is now restricted to the right half of the parabola with values of $x \geq -1/2$, we'll use the *second* of the solutions returned by **Solve** to define an inverse function, since its values will always satisfy $x \geq -\frac{1}{2}$ when $y \geq -\frac{1}{4}$. We'll use the dummy variable trick to define g.[10]

```
► g[y_] = x /. solutions[[2]]
```

▷ $\frac{1}{2}\left(\sqrt{4y+1} - 1\right)$

Of course, the function g is defined only for values $y \geq -1/4$. To see, then, that f and g are inverse functions, you only need to compute $f(g(x))$ and $g(f(x))$

```
► Simplify[f[g[x]]]
```

▷ x

```
► Simplify[g[f[x]]]
```

▷ $\frac{1}{2}\left(\sqrt{(2x+1)^2} - 1\right)$

The first of these results is as expected, but the second is not. *Mathematica* cannot **Simplify** this last expression, since the value of $\sqrt{(1+2x)^2}$ is not *symbolically* the same as $1 + 2x$ unless $1 + 2x \geq 0$. But in this case we specifically do require $x \geq -1/2$. Thus we can legitimately use **PowerExpand** to collapse exponents to be $(1 + 2x)^{2(1/2)} = 1 + 2x$

```
► PowerExpand[%]
```

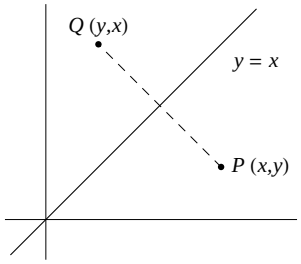
▷ x

We're now convinced that $f^{-1}(y) = g(y) = \frac{1}{2}\left(\sqrt{4y+1} - 1\right)$.

[10] We must use immediate assignment (=) here, rather than delayed assignment (:=). We see the result of the definition to confirm what we've done, and the definition of g will not change later should the variable **solutions** change.

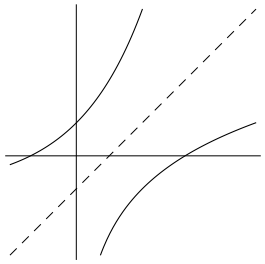
6.1.3 The Geometry of Inverse Functions

There is a simple and pleasing relationship between the graph of a function and that of its inverse (when one exists). A point $P = (x, y)$ will be on the graph of f if and only if the point $Q = (y, x)$ is on the graph of f^{-1}. But as the diagram below suggests, the line segment between the points P and Q will then have the line $y = x$ as its perpendicular bisector. Thus the graph of f^{-1} will always be the *symmetric image* of the graph of f in the line $y = x$.



First, we'll check this relationship in the graphs of the natural exponential and logarithm functions. However, getting a pleasing yet accurate picture requires making separate plots of the two functions (over appropriate intervals) and the line $y = x$ and then combining them.

```
► graph1 = Plot[Exp[x], {x,-1,1}];
  graph2 = Plot[Log[x], {x,1/E,E}]
  graph3 = Plot[x, {x,-1,E},
      PlotStyle→{Thin, Black, Dashed}]
  Show[graph1, graph2, graph3,
      AspectRatio→Automatic,
      PlotRange→{{-1,E},All}, Ticks→None]
```



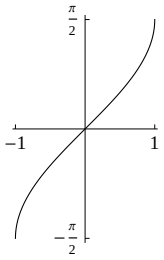
Now, consider using this graphic relationship in reverse. That is, should a function f have an inverse, we can see the graph of its inverse just by plotting a graph with the coordinates reversed. There's no need to work through algebra to find a formula for the inverse function; rather, we'll simply sketch the parametric curve $\{(f(x), x) \mid a \le x \le b\}$.

As an example, the function $f(x) = \sin(x)$ is strictly increasing on $-\frac{\pi}{2} \le x \le \frac{\pi}{2}$, thus we can see its inverse (i.e., the inverse sine function) with the following which reverses the x and y coordinates.[11]

[11] If you'd like something more graphically-driven than using a **ParametricPlot**, the following sequence more directly "flips the graph over the line $y = x$." See if you understand why!

```
Rotate[ImageReflect[Plot[Sin[x], {x,−π/2,π/2}, AspectRatio→Automatic], Left→Right], 90 Degree]
```

```
► ParametricPlot[{Sin[x],x}, {x,-Pi/2,Pi/2},
      Ticks→{{-1,1},{-Pi/2,Pi/2}}]
```



Be sure you catch the subtlety of what's been done above. Even though we don't think of the inverse sine function to be given by a formula, we know it exists, we can see its graph, and we know several its values.

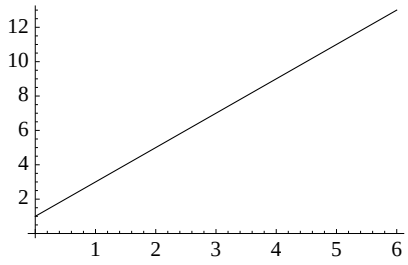
6.2 Limits

The notion of a limit is central to calculus. The definitions of the derivative, the integral, and many notions involving convergence (e.g., series) depend on it.

6.2.1 The Limit Concept

We begin by considering the function $f(x) = \dfrac{2x^2 - 5x - 3}{x - 3}$, a rational function that is defined for all $x \neq 3$. Many students immediately think "asymptote," but this is certainly not the case.

```
► Clear[f]
  f[x_]:= (2x^2-5x-3)/(x-3)
  Plot[f[x], {x,0,6}]
```



Other than not being defined at $x = 3$, this function is quite well-behaved around $x = 3$ graphically. The same is true numerically.

To see this, we'll make a short *list* of values of f near (and including!) $x = 3$ in a single input, with the items separated by commas and enclosed in curly braces **{** and **}**.

```
► {f[2.998],f[2.999],f[3],f[3.001],f[3.002]}
```

▷ {6.996, 6.998, Indeterminate, 7.002, 7.004}

The output above comes with warnings from *Mathematica* about the lack of meaning of the term $f(3)$ we included; but otherwise, the values of f near $x = 3$ all appear to be about

7. If f *were* to be defined at $x = 3$, and if we wanted the graph to *nicely* resemble the graph we drew above, then we'd clearly *want* to have $f(3) = 7$.

We summarize the statements above in writing $\lim_{x\to 3} f(x) = 7$. More needs to be said, of course, to properly *define* the meaning of a limit, but a numerical approach is usually the first approach taken towards understanding the *concept* of a limit.

Fortunately, our calculus books will add all the necessary detail about the precise meaning of writing $\lim_{x\to 3} f(x) = 7$, and we'll not replicate that discussion here.

6.2.2 The Limit command

Mathematica's **Limit** command performs a symbolic computation of the limit of an expression that depends on a single variable. For example, to evaluate $\lim_{x\to 3} \dfrac{2x^2 - 5x - 3}{x - 3}$ discussed above, we'd use

```
Clear[f]
f[x_]:= (2x^2 - 5x - 3)/(x - 3)
Clear[x]   (* x must be symbolic for Limit to to work! *)
Limit[f[x],x→3]
```

▷ 7

The point at which the limit is computed is specified using the familiar "arrow syntax," made up of a minus sign and a greater-than sign. This nicely replicates the mathematical notation "$x \to 3$" we see in writing $\lim_{x\to 3} \dfrac{2x^2 - 5x - 3}{x - 3}$.

Another common example (one that is fundamental to understanding both limits and working with trigonometric functions) is that $\lim_{x\to 0} \dfrac{\sin x}{x} = 1$

```
Limit[Sin[x]/x,x→0]
```

▷ 1

Limit is capable of handling most computations which would ordinarily involve advanced techniques such as L'Hôpital's Rule. Indeed, we'd use this rule to compute mathematically that:

$$\lim_{x\to 0} \frac{e^x - 1 - x}{x^2} = \lim_{x\to 0} \frac{e^x - 1}{2x} = \lim_{x\to 0} \frac{e^x}{2} = \frac{1}{2}$$

Mathematica handles this calculation directly:

```
Limit[(e^x - 1 - x)/x^2, x→0]
```

▷ $\dfrac{1}{2}$

Limit will properly report infinite limits, as in

```
Limit[1/x^2, x→0]
```

▷ ∞

When a limit does not exist, or if *Mathematica* is unable to determine whether a limit exists – perhaps because you've supplied an expression that is symbolically ambiguous – you'll likely have your input returned as output.

```
Clear[f,x]
```

```
Limit[f[x], x→0]
```

▷ $\lim_{x\to 0} f(x)$

However, you need to be aware of an important implementation note concerning the **Limit** command.

➽ *Just because Mathematica reports a limit, it doesn't necessarily mean that the limit exists. Indeed, the* ***Limit*** *command reports the right-hand limit of an expression.*

Example. A standard result with which you're familiar is that $\lim_{x\to 0} \dfrac{|x|}{x}$ does not exist. However, *Mathematica* reports that

```
► Limit[Abs[x]/x, x→0]
```

▷ 1

In fact, *Mathematica* reports that $\lim_{x\to 0^+} \dfrac{|x|}{x} = 1$, which is correct. Don't be tricked – you'll need to now examine the left-hand limit at 0.

To compute a left-hand limit, you add the **Direction** option to the **Limit** command and specify its value to be +1. Thus the left-hand limit $\lim_{x\to 0^-} \dfrac{|x|}{x} = 1$ is computed with

```
► Limit[Abs[x]/x,x→0,Direction→+1]
```

▷ −1

It is only now, when we compare this result to the **Limit** computation above (where we did not specify the **Direction**), that we conclude that $\lim_{x\to 0} \dfrac{|x|}{x}$ does not exist (and, more specifically, that it has a jump discontinuity, if you're familiar with the term).

One-sided limit computations arise all the time with rational functions, for which we often suspect the presence of vertical asymptotes in a graph. For example, $y = \dfrac{1-x}{x-3}$ has a vertical asymptote at $x = 3$, and by hand we should certainly be able recognize the left- and right-hand limits

$$\lim_{x\to 3^-} \frac{1-x}{x-3} = +\infty \text{ and } \lim_{x\to 3^+} \frac{1-x}{x-3} = -\infty$$

In *Mathematica*, the computation looks like this

```
► Limit[(1 - x)/(x - 3), x→3, Direction→+1] (* left-hand limit *)
```

▷ ∞

```
► Limit[(1 - x)/(x - 3), x→3, Direction→-1] (* right-hand limit *)
```

▷ −∞

The **Direction** option should be +1 for left-hand limits, and using −1, the default value, can certainly be explicitly specified for a right-hand limit.

➽ The +1 is used to denote the computation of the limit as the value of the variable *increases*, coming from the left; −1 is used to denote the computation of the limit as the value of the variable *decreases*, coming from the right. *Don't*

confuse the sign of the ***Direction*** *specifier with the mathematical* $\pm$ *sign used to denote left- and right-hand limits.*

6.2.2.1 Limits at Infinity

Limits at infinity such as $\lim_{x\to\pm\infty} \dfrac{x}{\sqrt{x^2+1}} = \pm 1$ can be computed in *Mathematica*

```
Clear[f]
f[x_] := x/Sqrt[x^2+1]
Limit[f[x],x→Infinity]
```

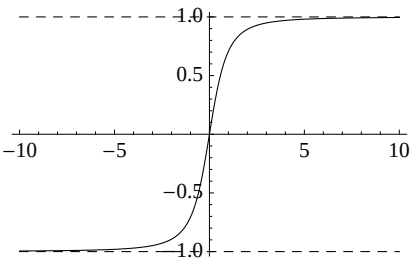
▷ 1

```
Limit[f[x],x→-Infinity]
```

▷ −1

We can now conclude that the function $f(x) = \dfrac{x}{\sqrt{x^2+1}}$ has a horizontal asymptote at $y = 1$, as $x \to +\infty$, and at $y = -1$, as $x \to -\infty$. Indeed, consider:

```
Plot[{-1,1,x/Sqrt[x^2+1]},{x,-10,10},
     PlotStyle→{{Dashed},{Dashed},{Black}}]
```



6.2.2.2 Symbolic Limits

Limits can be found even when expressions depend on symbolic parameters. For example, *Mathematica* can make the specific computation that $\lim_{x\to 0}(3x+1)^{1/(2x)} = e^{3/2}$

```
Limit[(3x+1)^(1/(2x)), x→0]
```

▷ $e^{3/2}$

Impressively, *Mathematica* can make the more general, symbolic computation that $\lim_{x\to 0}(ax+1)^{1/(bx)} = e^{a/b}$, for any symbolic constants a and b

```
Clear[a,b]
Limit[(a*x+1)^(1/(b*x)), x→0]
```

▷ $e^{a/b}$

Similarly, if a and b are constants, we would algebraically make this lengthy calculation of a limit at infinity.

$$\begin{aligned}
&\lim_{x\to\infty}(\sqrt{x^2+ax+b}-x) = \\
&\qquad \lim_{x\to\infty}(\sqrt{x^2+ax+b}-\sqrt{x^2})\frac{\sqrt{x^2+ax+b}+\sqrt{x^2}}{\sqrt{x^2+ax+b}+\sqrt{x^2}} \\
&\qquad = \lim_{x\to\infty}\frac{ax+b}{\sqrt{x^2+ax+b}+\sqrt{x^2}}
\end{aligned}$$

$$= \lim_{x\to\infty} \frac{a + b/x}{\sqrt{1 + a/x + b/x^2} + \sqrt{1}}$$
$$= \frac{a}{2}$$

The **Limit** command reproduces this result (despite issuing warnings that it is evaluating an indeterminate expression of the form $\infty - \infty$):

```
▶ Limit[Sqrt[x^2+a*x+b]-x,x→Infinity]
```

▷ Infinity::indet: Indeterminate expression
-Infinity + Infinity encountered.

$$\frac{a}{2}$$

6.3 Continuity

For a function f to be continuous at a point $x = a$, we require that both $\lim_{x\to a} f(x)$ and $f(a)$ exist and that they be equal. We usually condense the language by simply writing that $\lim_{x\to a} f(x) = f(a)$.

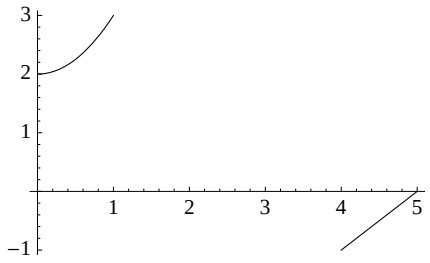
Geometrically, a function continuous at a point $x = a$ has a graph without any holes, skips, or jumps near $x = a$. Most functions we work with in practice (polynomials, rational functions, trigonometric, exponential, and logarithmic functions) are continuous wherever they are defined.

We provide only one example here specifically related to continuity, using *Mathematica* to determine the constants a and b so that the following function becomes continuous on $\mathbb{R}$.

$$f(x) = \begin{cases} x^2 + 2 & x < 1 \\ ax + b & 1 \le x \le 4 \\ x - 5 & x > 4 \end{cases}$$

Should we attempt to plot the graph of f *as defined*, we of course see nothing appearing over the interval $[1, 4]$ since f does not yet have *numerical* values there.

```
▶ Clear[f,a,b]
  f[x_] := Piecewise[{
      {x^2+2, x<1},
      {a*x+b, 1<=x<=4},
      {x-5, x>4}
    }]
  Plot[f[x], {x,0,5}]
```



Geometrically, the values of a and b must be chosen to "connect the dots," since the graph of f will be linear on $[1, 4]$. Analytically, the problem is one of arranging for continuity by

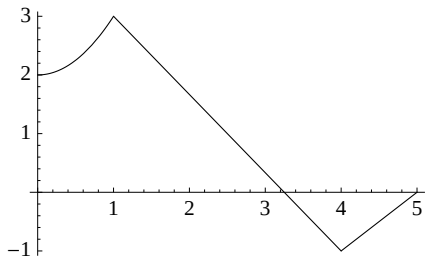
selecting a and b so that $\lim_{x\to 1^-} f(x) = f(1) = a + b$ and $\lim_{x\to 4^+} f(x) = f(4) = 4a + b$. Thus we have

```
▶ lim1 = Limit[f[x], x→1, Direction→+1];
  lim4 = Limit[f[x], x→4, Direction→-1];
  solutions=Solve[{lim1==f[1],lim4==f[4]}, {a,b}]
```

▷ $\left\{\left\{a \to -\frac{4}{3}, b \to \frac{13}{3}\right\}\right\}$

Let's be sure we have the correct solutions. We'll now define a and b using this result (using, once again, the dummy variable trick) and see what the graph looks like.

```
▶ a = a /. solutions[[1]];
  b = b /. solutions[[1]];
  Plot[f[x], {x,0,5}]
```



Mission accomplished!

6.4 Exercises

1. Find the inverse function for each of the following functions, and verify that the graphs of the function and its inverse are symmetric in the line $y = x$.

(a) $f(x) = x^3 + 2x$

(b) $f(x) = \dfrac{7x + \pi}{x - 1}$

2. A function of the form $f(x) = \dfrac{ax + b}{cx + d}$, for constants a, b, c, and d, is called a *linear, fractional transformation*. Show that f^{-1} is also a linear, fractional transformation, and determine its form in terms of the constants a, b, c, and d.

3. For the two functions $y = \tanh(x)$ and $y = \dfrac{1}{2}\ln\left(\dfrac{1+x}{1-x}\right)$

(a) Verify that neither of the compositions of the two functions can be easily simplified to show algebraically that they are inverse functions.

(b) Nevertheless, visually verify that they are inverse functions on $(-1, 1)$ by plotting them together with the line $y = x$.

(c) Plot the composition of the two functions, in each of the two orders, on $(-1, 1)$ and compare the results with the graph of $y = x$. Is there now sufficient evidence to believe that these are inverse functions?

4. Given the two functions $f(x) = \sin(x)$ and

$$g(x) = x + \frac{x^3}{6} + \frac{3x^5}{40} + \frac{5x^7}{112} + \frac{35x^9}{1152} + \frac{63x^{11}}{2816}$$

plot their compositions $f(g(x))$ and $g(f(x))$ over the interval $[-1, 1]$, and compare with the graph of $y = x$. Is it reasonable to believe that f and g are inverse functions? Why or why not?

5. Find the smallest value of the constant a for which the function $f(x) = x^4 - x^2$ has an inverse on $[a, \infty)$ (it suffices to choose a so that f is strictly increasing on $[a, \infty)$ – this can be determined graphically). Find an inverse g for f on this interval, and demonstrate that f and g are inverse functions. Be specific as to the domain of the function g that you find.

6. Use the **Limit** command to evaluate $\lim_{x\to 0} \sin(\frac{1}{x})$. Explain what happens. Consider sketching the graph of $f(x) = \sin(\frac{1}{x})$ near zero to help you explain.

7. Repeat the previous problem for $\lim_{x\to 0} x \sin(\frac{1}{x})$.

8. Evaluate $\lim_{x\to 0} \dfrac{\sin(ax)}{bx}$, where a and b are constants with $b \neq 0$.

9. Evaluate $\lim_{x\to 0} \dfrac{\sqrt{x} - \sqrt{a}}{x - a}$, where a is a positive constant.

10. Let $r(x) = \dfrac{x^3 - x^2}{x^2 + x - 7}$.

(a) Verify that $\lim_{x\to\infty} (r(x) - (x - 2)) = 0$.

(b) This limit calculation suggests that the line $y = x - 2$ is an oblique asymptote of the graph of the function r. Graph r and this line over the interval $[-10, 10]$ to demonstrate the asymptotic relationship.

11. Let $f(x) = \sqrt{4x^2 + 3x + 2}$.

(a) Compute $\lim_{x\to\infty} (f(x) - 2x)$.

(b) This result of part (a) suggests that the graph of $y = f(x)$ has an oblique asymptote of the form $y = ax + b$, for suitable constants a and b. What are a and b?

(c) Graph f and this line over the interval $[-10, 10]$ to demonstrate the asymptotic relationship.

12. Evaluate $\lim_{x\to\infty} (\sqrt[3]{x^3 + ax^2 + bx + c} - x)$ in terms of the coefficients a, b, and c.

13. For $x \neq 0$, define $f(x) = \dfrac{e^{x^3} - 1}{x \sin x^2}$. What value should be assigned to $f(0)$ so that f will become continuous at $x = 0$?

14. Repeat the previous problem for $f(x) = \dfrac{x \sin x^3}{\cos x^2 - e^{x^4}}$.

15. Suppose f is the following function, where a and b are arbitrary constants.

$$f(x) = \begin{cases} x^4 + x + 1, & x \leqslant -1 \\ ax + b, & -1 < x < 2 \\ \cos\frac{\pi x}{2} + 5x, & 2 \leqslant x \end{cases}$$

(a) Find constants a and b so that the following function becomes continuous for all real x. (Note: f will be continuous at -1 if $f(-1) = \lim_{x\to -1^+} f(x)$, and continuous at 2 if $f(2) = \lim_{x\to 2^-} f(x)$.)

(b) Enter f in *Mathematica* using a **Piecewise** definition and **Plot** the resulting function. Is the function continuous for all real x?

7 Calculus Applications (II)

This chapter continues discussion of Calculus-based concepts in *Mathematica*. We focus our attention here on the concept of the derivative.

7.1 The Derivative

Recall that the derivative of a function f at a point $x = a$ is given by the following expression, whenever it exists.

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

When $f'(a)$ exists, the graph of $y = f(x)$ has a tangent line at the point $(a, f(a))$ and the slope of this tangent line is exactly $f'(a)$. If $y = f(x)$ is a *position function* – a function that gives the position of an object traveling linearly along a number line at time x – then $f'(a)$ gives the instantaneous velocity of the object at time $x = a$.

7.1.1 Calculating the Derivative by Definition

To find the derivative of a function f at a point $x = a$ using *algebraic methods*, you first calculate the difference quotient

$$\frac{f(a+h) - f(a)}{h} \qquad \text{for } h \neq 0$$

and then evaluate the limit of this expression as h approaches 0.

In the case that the expression above simplifies nicely, evaluating the limit can be as easy as evaluating the simplified expression with $h = 0$. We can certainly do this directly in *Mathematica*, using its computer algebra capability.

To compute the derivative of the function $f(x) = x^3$, we begin with

```
▶ Clear[f]
  f[x_] := x^3
```

We form the difference quotient $\frac{f(a+h)-f(a)}{h}$, at an arbitrary point named a

```
▶ Clear[a,h]
  (f[a+h]-f[a])/h
```

$$\triangleright \frac{(a+h)^3 - a^3}{h}$$

Only a little work is required to obtain a better-looking expression when $h \neq 0$.

```
▶ Simplify[%]
```

$$\triangleright 3a^2 + 3ah + h^2$$

Lastly, the simplified difference quotient is given by a polynomial in the variable h and its limit is computed by taking h *to be* 0 using a standard variable substitution rule (polynomials are continuous throughout their domain)

```
▶ % /. {h→0}
```

$$\triangleright 3a^2$$

We conclude that $f'(a) = 3a^2$, as it should be.

In the early days of calculus development, before rigorous notions of limits were available, Fermat argued that you should compute the derivative exactly in this manner, by first

computing the difference quotient, and then simply taking h to be 0 (after appropriate simplification of the quotient). *Mathematica* had no trouble with that procedure as we saw above.

On the other hand, relying on algebraic simplification will not carry very far beyond polynomials and rational functions. Indeed, as we saw earlier, working algebraically with the transcendental functions is particularly tricky.

For example, if $f(x) = \sin(x)$, the procedure above starts out with

```
Clear[f]
f[x_] := Sin[x]
(f[a+h]-f[a])/h
```

$$\triangleright \frac{\sin(a+h) - \sin(a)}{h}$$

Working with this expression algebraically is difficult. We could try a few things

```
TrigExpand[%]
```

$$\triangleright -\frac{\sin(a)}{h} + \frac{\sin(a)\cos(h)}{h} + \frac{\cos(a)\sin(h)}{h}$$

```
Together[%]
```

$$\triangleright \frac{\sin(a)\cos(h) + \cos(a)\sin(h) - \sin(a)}{h}$$

But it doesn't look as if we're getting close to any expression where you could simply take h to be zero. Thus Fermat's idea won't work here.

Instead, we'll need the full power of the **Limit** operator with the definition of derivative to obtain $f'(a) = \cos(a)$.

```
Clear[a]
Limit[(f[a+h]-f[a])/h, h→0]
```

$\triangleright \cos(a)$

7.1.2 f' Notation and the D Operator

The derivative computation is already implemented directly in *Mathematica*, so there's no need to explicitly carry out the limit definition.

If f is a function, *Mathematica* will understand the symbol f' as the derivative of f. You enter the prime as the single quote character (`'`). For example

```
Clear[f]
f[x_] := x^4
f'[x]
```

$\triangleright 4x^3$

Higher-order derivatives follow with the usual notation

```
f''[x]
```

$\triangleright 12x^2$

```
f'''[a]
```

$\triangleright 24a$

An alternative operator-based notation for the derivative is to use the **D** operator.

```
▶ D[f[x],x] (* first derivative *)
```

▷ $4x^3$

To calculate higher-order derivatives, either of the following syntax formats can be used, say, for the third derivative.

```
▶ D[f[x],x,x,x] (* third derivative *)
  (* or *)
  D[f[x],{x,3}]
```

▷ $24x$

Technically, the operator **D** computes only the *partial derivative* of an expression with respect to the stated variable. That is, it treats all symbols other than the given variable to be constant with respect to the variable. For example

```
▶ Clear[a,b,c]
  D[a*x^2+b*x+c,x] (* deriv of f(x) = ax^2 + bx + c *)
```

▷ $2ax + b$

The operator **D** recognizes the usual collection of derivative computation rules. For example, the computation of derivatives of sums, products and compositions of functions are known symbolically

```
▶ Clear[f,g]
  D[f[x]+g[x],x]
```

▷ $f'(x) + g'(x)$

```
▶ D[f[x]g[x],x]
```

▷ $g(x)f'(x) + f(x)g'(x)$

```
▶ D[f[g[x]],x]
```

▷ $f'(g(x))g'(x)$

Finally, *Mathematica* may not always produce exactly the derivative formula you'd expect. An excellent example is the case of the "usual" derivative formula for the inverse secant function, defined for $|x| \geq 1$ (and where the value of $\sec^{-1}(x)$ lies in $[0, \frac{\pi}{2}) \cup (\frac{\pi}{2}, \pi]$)

$$\frac{d}{dx}\sec^{-1}(x) = \frac{1}{|x|\sqrt{x^2-1}}$$

In this case, *Mathematica* computes

```
▶ D[ArcSec[x],x]
```

▷ $\dfrac{1}{\sqrt{1-\frac{1}{x^2}}\ x^2}$

Of course, this is a proper result, since

$$\frac{1}{x^2\sqrt{1-1/x^2}} = \frac{1}{|x|^2\sqrt{1-1/x^2}} = \frac{1}{|x|\sqrt{x^2(1-1/x^2)}} = \frac{1}{|x|\sqrt{x^2-1}}$$

The output format chosen by *Mathematica* for the derivative cleverly avoids the use of the absolute value function, which should help if you hope to make any algebraic use of the result.

7.1.3 Technical Note: Defining Functions with D

To define a function in *Mathematica*, we've recommended using the delayed assignment mechanism (**:=**). However, if the right side of a function definition depends on **D** (or several other *Mathematica* operators), we must adopt a slightly different route.

The simple example is when we might define a function g to be the derivative of a function f that's already in use in your *Mathematica* session. For demonstration, we'll use $f(x) = x^2$

```
▶ Clear[f,g]
  f[x_]:= x^2
  g[x_] := D[ f[x],x ] (* we THINK g(x) = 2x *)
```

Unfortunately, the function g is not properly defined

```
▶ g[2] (* we THINK this is 2(2) = 4 *)
```

▷ General::ivar: 2 is not a valid variable

What's happened above is that the expression **g[2]** has been transformed into the expression **D[f[2],2]**, or **D[4,2]** *well before* the derivative is computed. This last expression makes no sense since, as the message states, the second argument of **D** must be a variable and not a number such as 2.

The problem is easily (and perhaps best) solved by using the prime syntax instead of the **D** operator. Indeed, we have

```
▶ Clear[f,g]
  f[x_]:= x^2
  g[x_] := f'[x] (* this REALLY IS g(x) = 2x *)
  g[2]
```

▷ 4

Should there come a situation where the prime syntax cannot easily be used (e.g., the function g above might be the derivative of some expression found after a sequence of calculations), the alternative solution is to use *immediate assignment* to define g, forcing the derivative computation to be made when the definition is executed.

Thus the following sequence *does not work*

```
▶ Clear[g]
  g[x_] := D[x^3+5x,x] (* NOT! *)
```

but the following sequence *does work*, because it uses the immediate assignment operator (=)

```
▶ Clear[g]
  g[x_] = D[x^3+5x,x] (* g(x) IS 3x^2 + 5 *)
```

▷ $3x^2 + 5$

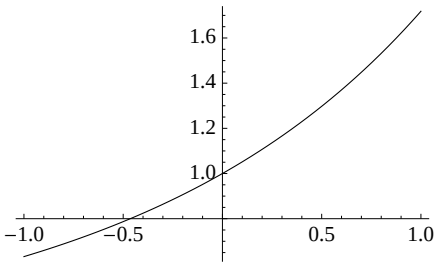
7.1.4 Derivatives and Removable Discontinuities

Finding the derivative of a function f at a point $x = a$ where f has a removable discontinuity usually requires a direct application of the definition of the derivative.

For example, given the function $f(x) = \dfrac{e^x - 1}{x}$, we're aware that the function as defined is discontinuous at the origin (the formula is not defined for $x = 0$). However, the discontinuity is removable as you can see from its graph

```
▶ Clear[f]
```

```
f[x_] := (Exp[x]-1)/x
Plot[f[x], {x,-1,1}]
```



The graph agrees with the computation that $\lim_{x\to 0} \dfrac{e^x - 1}{x} = 1$, and thus the definition of f can be extended at the origin by setting $f(0)$ to be this limit. In *Mathematica*, we'll thus set

```
► f[0] = Limit[f[x], x→0]
```

▷ 1

To understand the extended *Mathematica* definition of f, we examine

```
► ?f
```

▷ Global‘f
f[0]=1
f[x_]:=(Exp[x]-1)/x

With f now properly defined at the origin to be a continuous function, we can determine that it is, in fact, differentiable at the origin with $f'(0) = \frac{1}{2}$ by executing the definition of the derivative directly.

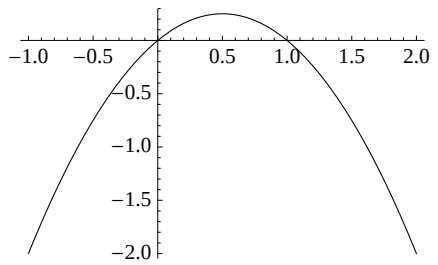
```
► Clear[h]
  Limit[(f[0+h]-f[0])/h, h→0]
```

▷ $\dfrac{1}{2}$

7.2 Tangent Lines

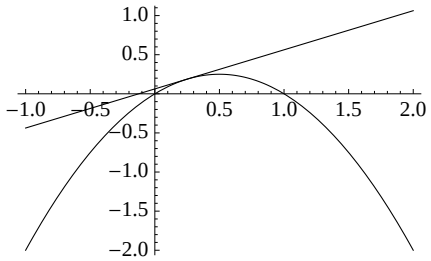
Mathematica easily plots tangent lines to the graphs of functions. Consider first the graph of $f(x) = x - x^2$. The graph of f is a parabola that opens downward.

```
► Clear[f]
  f[x_] := x-x^2
  Plot[f[x],{x,-1,2}]
```



At any point $(a, f(a))$, the tangent line at that point on the graph has equation $y = f(a) + f'(a)(x - a)$. This line can be specified directly in the **Plot** operator at any point $x = a$. To see the graph above together with its tangent line at $a = \frac{1}{4}$, we use

```
► a = 1/4;
  Plot[{f[x],f[a]+f'[a](x-a)},{x,-1,2}]
```

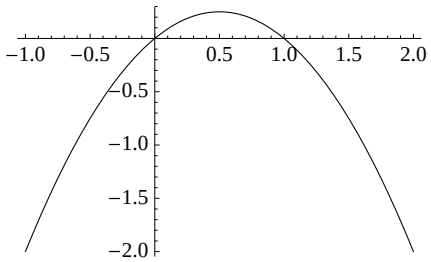


7.2.1 Special Feature: Manipulate and Symbolic Parameters

Students are often confused when we use symbolic terms in expressions and other inputs. The sequence above provides a good example. Indeed, we used a general formula to describe the tangent line's equation in the plot, but to make it effective, the symbol **a** had to have a value before executing the **Plot**.

Suppose we attempted the same sequence, but without having assigned a value to the variable **a**

```
► Clear[f,a]
  f[x_] := x-x^2
  Plot[{f[x],f[a]+f'[a](x-a)},{x,-1,2}]
```



Only the graph of f appeared above and none of the tangent line with equation $y = f(a) + f'(a)(x - a)$ has been drawn, and for an obvious reason. Since **a** has no numeric value, no numeric value can be found for any y-coordinate $f(a) + f'(a)(x - a)$, and so no points can be plotted for the line.

However, suppose we were interested in generally examining the behavior of the tangent line at *any* value of a, say, as a varies over the interval $-1 \le a \le 2$?

One option is to draw lots of graphs using different values of a. That's possible, but tedious (one graph for each value of a) and does not have the same comparative effect as what we now demonstrate.

The **Manipulate** operator is a powerful operator whose output allows you to interactively display a wide range of plots depending on a symbolic parameter such as a. This is exactly what we need right here, right now.

We begin with these definitions

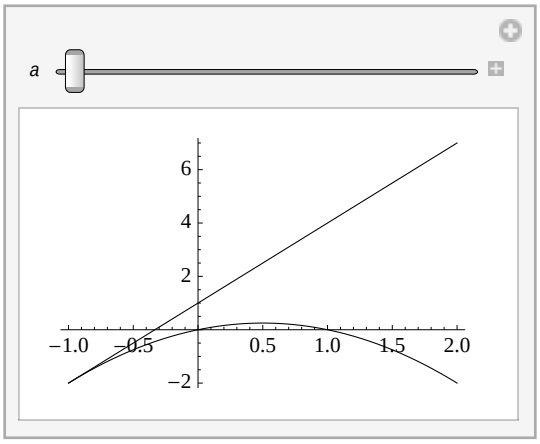
```
Clear[f,a]
f[x_] := x-x^2
```

and instead of attempting to "plot" the graph of f together with its symbolic tangent line

```
Plot[{f[x],f[a]+f'[a](x-a)},{x,-1,2}]
```

we instead hand over this entire **Plot** command to **Manipulate**, specifying that we're interested in a range of the symbolic parameter a from, say, $-1 \leq a \leq 2$.

```
Manipulate[
  Plot[{f[x],f[a]+f'[a](x-a)},{x,-1,2}],
  {a,-1,2} ]
```



You'll be presented with a small window in which you are originally looking at the graph of f and the tangent line at $a = -1$. A slider appears at the top of the window and, by dragging it to the right, the position of the tangent line will vary in real time over the interval $-1 \leq a \leq 2$.

Further, if you click on the little $\boxed{+}$ sign that appears at the right end of the slider, you'll see standard VCR-like controls allowing you to "see the movie," and there will also be a box where you can enter the value of a directly. Neat!

Manipulate is a very capable graphics operator. This use – literally manipulating a complete **Plot** command that depends on a symbolic parameter – only scratches the surface of its possibilities. More will be seen later. But three comments should be added here to help you work with **Manipulate**.

First, you'll notice that the graphic will "jump around" a little as you drag the slider to control the value of **a**. The reason for this is that the **Plot** output uses a different plot range for each value of a.

You can eliminate the "jumping around" problem by specifying a common **PlotRange** directly in the **Plot** command. Restricting the graphic to the range of $-3 \le y \le 3$ seems to work pretty well (although we'll not show the result).

```
▶ Manipulate[
    Plot[{f[x],f[a]+f'[a](x-a)},
         {x,-1,2}, PlotRange→{-3,3}],
    {a,-1,2} ]
```

Second, pay special attention to syntax. As used above, **Manipulate** has two arguments, a complete **Plot** command that starts with the characters **Plot**[and ends with the character], and a range specification for the symbolic parameter. These are separated by a comma.

Notice that we took the time above to enter the complete **Manipulate** expression over multiple lines so that we can more easily identify these two parameters. This helps make it more clear as to what elements of the input belong within the **Plot** command.

Finally, if you're having trouble using **Manipulate**, be sure to test the **Plot** command on its own (with some fixed value of the parameter) to see that it works correctly. **Manipulate** won't do the right thing if the **Plot** command doesn't work on its own!

7.2.2 Technical Note. D and Plot

Whenever possible, we recommend that you use the f' syntax for a derivative if it is used in conjunction with **Plot**. Use of **D** with **Plot** unfortunately requires some care. This is due more an oddity of the **Plot** operator than anything else, and we'll now take the opportunity to discuss the issue briefly.

It would seem that the following sequence should make sense, but it does not. Only error messages and an empty graph are produced.

```
▶ f[x_] := x^2
  Plot[ D[f[x],x], {x,-2,2} ] (* plots y = 2x ? *)
```

The **Plot** operator treats its first argument (the one that defines the function or functions to be plotted) much differently than do most *Mathematica* operators. To execute the **Plot**, *Mathematica* begins by substituting several x-values from the interval $-2 \le x \le 2$ into the expression **D[f[x],x]** and then evaluating the expression. This means that *Mathematica* attempts to evaluate expressions such as **D[f[-2],-2]**, **D[f[-1],-1]**, **D[f[0],0]**, **D[f[1],1]**, and **D[f[2],2]**.

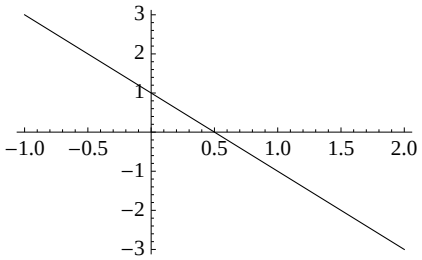
Mathematica has simply "plugged in" values into the expression **D[f[x],x]** *before* computing the derivative and tried to make sense out of it. But none of these expressions makes any sense! (This situation is not unrelated to the syntax issues that were noted in section 7.1.3.)

In order to force *Mathematica* to evaluate the derivative *before* **Plot** gets hold of it, you should use one of the following with the **Evaluate** operator

```
▶ Plot[ Evaluate[D[f[x],x]], {x,-2,2} ]
  (* or *)
  Plot[ D[f[x],x]//Evaluate, {x,-2,2} ]
```

The input above is then forced to be treated as the equivalent of the following, intended input and you'll get a meaningful plot

```
▶ Plot[ 2x, {x,-2,2} ]
```



7.3 Optimization

Many useful applications of differential Calculus involve computation of the extreme (maximum and minimum) values of a function. The following is usually presented as the basis for such optimizations:

Theorem. *Let f be continuous on $[a, b]$. Then f will attain both a maximum and minimum value on $[a, b]$. Further, if f attains an extreme value at some c in (a, b), then either f is not differentiable at c, or else $f'(c) = 0$.*

Consequently, the standard Calculus operation of maximizing or minimizing a continuous function f over a closed interval $[a, b]$ reduces to a simple three-step method:

(a) locate the critical numbers of f on $[a, b]$ – i.e., those points where f' is undefined or zero;
(b) evaluate f at each of these critical numbers; and
(c) evaluate f at the endpoints a and b.

Necessarily, the extreme values of f will be found from among the values identified in steps (b) and (c).

Example. Find the extreme values of $f(x) = x^2 - x$ on the interval $[-1, 3]$.

Solution. First we define the function, and locate its critical numbers (this function is differentiable everywhere):

```
Clear[f];
f[x_] := x^2 - x
solution = Solve[f'[x]==0,x]
```

▷ $\{\{x \to \frac{1}{2}\}\}$

The only critical number of f occurs at $x = 1/2$. We must then compare the value of f at this critical number, with the values of f at the endpoints $x = -1$ and $x = 3$:

```
f[x] /. solution
```

▷ $\{x \to -\frac{1}{4}\}$

```
f[-1]
```

▷ 2

```
f[3]
```

▷ 6

It follows that f attains a maximum value of 6, at $x = 3$; and a minimum value of $-1/4$, at $x = 1/2$.

If we hadn't specified the interval $[-1, 3]$, and were we only interested in the *relative*

extrema of f, then f has only the one critical number at $1/2$. Since f is twice differentiable, we could determine that f has a relative minimum at $1/2$ by using the Second Derivative Test. Indeed, f has a relative minimum at $1/2$ since $f''(1/2) > 0$.

```
► f''[1/2]
```

▷ 2

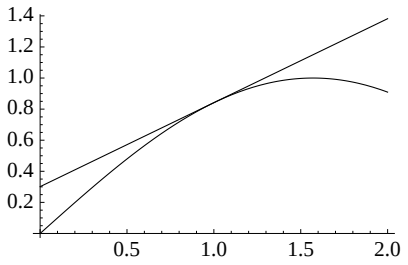
In general, the processes above are important, but the computer algebra steps demonstrated above will probably fail. Computing the critical numbers of the function f relies on the use of **Solve**, so only a limited collection of functions f will be amenable to the process. More robust, numerical techniques for optimization will be addressed later in the text.

7.4 Local Linearity and Differentiability

A key concept for a function f that is differentiable at a point $x = a$ is *local linearity* at the point $x = a$. Visually, this means that the graph of f and its tangent line at $x = a$ are, essentially, indistinguishable when viewed near $x = a$. We demonstrate.

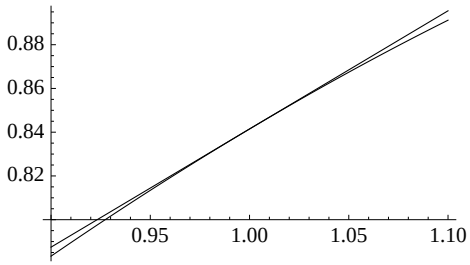
Consider the graph of $y = f(x) = \sin(x)$ near the point $a = 1$, together with its tangent line.

```
► Clear[f]
  f[x_] := Sin[x]
  a = 1;
  Plot[{f[x],f[a]+f'[a](x-a)},{x,0,2}]
```



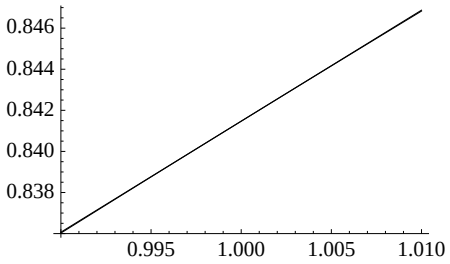
If we zoom in on the graph near the point $(a, f(a)) = (1, \sin(1))$ by shrinking the x-interval around $a = 1$, we see that the graph and the tangent line are quite close

```
► a = 1; (* just a reminder that a = 1 here! *)
  Plot[{f[x],f[a]+f'[a](x-a)},{x,0.9,1.1}]
```



We'll shrink once more, to the interval $a \pm 0.01 = 1 \pm 0.01$. But to more easily manage the syntax while shrinking the interval around $a = 1$, we introduce the following sequence, where the variable **h** controls the distance to each side of the point $x = a$ in the graph.

```
► a = 1;
  h = 0.01; (* we graph over a − h ≤ x ≤ a + h *)
  Plot[{f[x],f[a]+f'[a](x-a)},{x,a-h,a+h}]
```



So indeed, the graph of $f(x) = \sin(x)$ is essentially a line near $a = 1$ when the view is localized.

7.4.1 A Numerical Consequence of Local Linearity

Given the sequence above, we can conclude that when f is differentiable at a point $x = a$, the slope of the tangent line at $x = a$ and the slope of *what appears to be* the linear graph of f over an interval $a - h \le x \le a + h$ should be about the same.

Numerically, this means that the value of $f'(a)$ is approximately given by the average rate of change of f over the interval $a - h \le x \le a + h$, which is just

$$\frac{f(a+h) - f(a-h)}{(a+h) - (a-h)} = \frac{f(a+h) - f(a-h)}{2h}$$

For the sine function of the discussion above, we compute the average rate of change shown in the last graph above (where $f(x) = \sin(x)$, $a = 1$ and $h = 0.01$).

```
► (f[a+h]-f[a-h])/(2h)
```

▷ 0.540293

In comparison, we see that the exact value of $f'(a) = f'(1)$ is

```
► f'[a] //N
```

▷ 0.540302

The values above differ by very little (by no more than 0.00001) and, as a general conclusion, a numerical approximation for the the derivative at a point $x = a$ can be given as

$$f'(a) \approx \frac{f(a+h) - f(a-h)}{2h} \quad \text{when } h \text{ is close to } 0$$

Most graphing calculators you've used provide a numerical derivative function that employs *exactly* the estimate above, usually with $h = 0.001$ by default. For example, in *Mathematica* we estimate the derivative of the sine function at $a = 1$ with

```
► Clear[f]
  f[x_] := Sin[x]
  a = 1;
  h = 1/1000; (* an exact 0.001 *)
  estimate = (f[a+h]-f[a-h])/(2h);
  N[estimate,10]
```

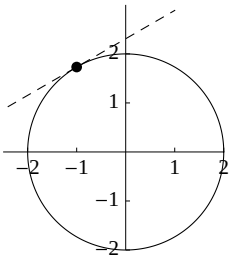
▷ 0.5403022158

On the TI-83 and TI-84 calculator, you'll find this to be *exactly* the output obtained from the command `nDeriv(sin(X),X,1)`.

7.5 Implicit Differentiation

Recall that curves in the plane are sometimes defined only by an equation. Working with a curve as the graph of a function is only possible if the curve passes the *vertical line test*; and even if it does pass the test, algebraically finding a formula for the function may not be either easy or even possible.

Thus consider the question of finding the slope of the line tangent to the curve $x^2+y^2=4$ at the point $(-1,\sqrt{3})$, as shown below.



A standard Calculus technique called *implicit differentiation* treats (say) the y in the equation $x^2+y^2=4$ as function of the variable x, and treats the point $(-1,\sqrt{3})$ as a point on its graph.

We may not know a *formula* for f (well, we really can find a formula, but assume for the moment that we cannot), but we have an *equation* that f satisfies, namely

$$x^2+f(x)^2=4$$

The calculus now proceeds to differentiate both sides of the equation and solve the resulting equation for $f'(x)$. Using the Leibnitz notation, we have

$$\begin{aligned} x^2+y^2&=4\\ 2x+2y\cdot\frac{dy}{dx}&=0\\ \frac{dy}{dx}&=-\frac{x}{y}\end{aligned}$$

The tangent slope at $(-1,\sqrt{3})$ is then

$$\left.\frac{dy}{dx}\right|_{(-1,\sqrt{3})}=-\frac{(-1)}{\sqrt{3}}=\frac{1}{\sqrt{3}}$$

The total differential operator **Dt** is used to carry out this computation in *Mathematica*. After specifying the variable of the equation to be **x**, **Dt** treats all remaining symbols in the equation as functions of **x** (of course, constants such as the 4 have zero derivative)

```
▶ Clear[x, y]
  Dt[x^2+y^2==4, x]
```

$$\triangleright\ 2y\frac{dy}{dx} + 2x = 0$$

The result is an equation involving the derivative $\dfrac{dy}{dx}$, written as **Dt[y,x]** in *Mathematica*, and we can solve for it explicitly (notice here how the **Solve** operator can be used to solve equations for a complete expression such as **Dt[y,x]**):

```
► solution = Solve[%, Dt[y,x]]
```

$$\triangleright\ \left\{\left\{\frac{dy}{dx} \to -\frac{x}{y}\right\}\right\}$$

To isolate the expression $-\dfrac{x}{y}$ so that we can work with it, we use a variation on the now familiar dummy variable trick, but with the expression **Dt[y,x]**

```
► expression = Dt[y,x] /. solution[[1]]
```

$$\triangleright\ -\frac{x}{y}$$

Finally we can evaluate this expression at $(-1, \sqrt{3})$ to compute the tangent slope $\left.\dfrac{dy}{dx}\right|_{(-1,\sqrt{3})}$

```
► expression /. {x→-1,y→Sqrt[3]}
```

$$\triangleright\ \frac{1}{\sqrt{3}}$$

Calculus texts will demonstrate the method of implicit differentiation in some detail, as well as discuss the *existence* of implicit functions and the *validity* of the technique. Certainly, as we see above, the *mechanics* of the method are easily replicated in *Mathematica*.

7.6 Exercises

1. Compute the derivatives of each of the functions in the following classes of functions, and compare the results with the standard entries in a Calculus text:

(a) trigonometric
(b) inverse trigonometric
(c) hyperbolic
(d) inverse hyperbolic

2. Verify symbolically that **D** follows the standard quotient rule for derivatives.

3. If f and g are differentiable functions, the product rule states that

$$\frac{d}{dx}\left(f(x)g(x)\right) = f(x)g'(x) + f(x)g'(x)$$

Continuing to the second order, the product rule expands as

$$\frac{d^2}{dx^2}\left(f(x)g(x)\right) = f(x)g''(x) + 2f'(x)g'(x)f''(x)g(x)$$

There's a familiar pattern at work here for derivatives of products that looks much like the pattern seen with binomial expansions.

Use *Mathematica* to determine the exact form of the pattern by checking **D[f[x]g[x],{x,n}]**, for a few values of n. What do you observe? Have you discovered Leibnitz' Rule?

4. For each of the following, zoom in on the graph around the indicated point $x = a$ until the graph appears to be a line, then compare the value of $f'(a)$ with the average value of f over the x-interval in the graph. Comment on how close the two values are (e.g., to 3 decimal places past the decimal point, or perhaps more?)

(a) $f(x) = (x-4)^2(x-2)^2(x+1)^2$ at $a = 1$ and $a = 3$

(b) $f(x) = \dfrac{\sin(x)}{1+x^2}$ at $a = 0.5$ and $a = 3$

(c) $f(x) = \tan^{-1}(x)$ at $a = 1$ and $a = 2$

5. Explain what the problem is with carrying out a local linearity investigation of the function $f(x) = |x|$ at the point $x = 0$. *Use Mathematica to support your explanation.*

6. Each of the following functions has a removable discontinuity at $x = 0$, and when suitably defined at $x = 0$, is differentiable there. For each,

- use the **Limit** operator to determine how $f(0)$ should be defined to remove the discontinuity;
- extend the definition of f in *Mathematica* to have this value;
- use both local linearity (in the graph) and the definition of the derivative to find the value of $f'(0)$.

Please be sure that both numerical estimates you find for $f'(0)$ (one from the graph, one from the **Limit** computation) agree to 3 or more decimal places past the decimal point.

(a) $f(x) = x^2 \sin(\frac{1}{x})$

(b) $f(x) = \dfrac{\ln(x+1)}{x}$

(c) $f(x) = \dfrac{e^x - 1 - x}{x^2}$

(d) $f(x) = \dfrac{e^{x^3} - 1}{x \sin x^2}$

7. Try repeating the previous problem for $f(x) = \dfrac{x \sin x^3}{\cos x^2 - e^{x^4}}$.

8. For the curve $x^2 - 3x^3y^4 + 5y^6 = 5$, find:

(a) the equation of the line tangent to the curve at the point $(0, 1)$.

(b) the value of the second derivative $\left.\dfrac{d^2y}{dx^2}\right|_{(0,1)}$ (differentiate implicitly a second time, and substitute the value of the first derivative obtained in part (a)).

9. Repeat the previous problem for the curve

$$x^2 \sin(xy) + (1+y)^3 \tan^{-1}(x+y) = \frac{\pi}{4}$$

but considered at the point $(1, 0)$.

10. Suppose $y = p(x) = ax^2 + bx + c$ is the parabola that passes through the points $(2, 7)$, $(6, 3)$ and $(10, 14)$. Determine the coordinates of the vertex of the parabola. (Suggestion: $p'(x) = 0$ at the vertex.

8 Integral Calculus

In this Chapter, we discuss – *in very general terms* – how *Mathematica* handles integration.

8.1 The Integrate command and Antiderivatives

The **Integrate** command provides symbolic antidifferentiation capabilities in *Mathematica*. For example, mathematically we would write:

$$\int x^2 - x^3 + \cos(x) - \ln(x)\, dx = \frac{x^3}{3} - \frac{x^4}{4} + \sin(x) - (x\ln(x) - x) + C$$

In *Mathematica*, we would make this computation as follows:

```
Clear[f]
f[x_]:= x^2 - x^3 + Cos[x] - Log[x]
answer = Integrate[f[x],x]
```

$$\triangleright\ -\frac{x^4}{4} + \frac{x^3}{3} + x - x\log(x) + \sin(x)$$

Note that the **Integrate** command has two arguments – the expression to integrate, and the variable of integration. Note also that *Mathematica*'s *log* function represents the *natural logarithm*, and that the terms of the antiderivative generated may not necessarily appear in the anticipated order. Finally, the result above is only one of many antiderivatives; the usual arbitrary constant of integration is omitted from the result of **Integrate**.

We can check that the antiderivative above is correct by computing its derivative:

```
D[answer,x]
```

$$\triangleright\ -x^3 + x^2 - \log(x) + \cos(x)$$

Note. You can (and may prefer to) use the Basic Commands integration template from the Basic Math Assistant palette to enter an integration expression. You'd first be given an integral template, then you can enter the integrand, press the Tab key, and enter the variable of integration. The sequence would look like this:

$$\int \textit{expr}\ d\,\textit{var} \quad \rightarrow \quad \int \texttt{x\^{}2 - x\^{}3 + Cos[x] - Log[x]}\ d\,\textit{var} \quad \rightarrow \quad \int \texttt{x\^{}2 - x\^{}3 + Cos[x] - Log[x]}\ dx$$

However, while the expression *appears* to look right, when you evaluate the expression you'll receive an error message! Unlike mathematical syntax in which the integral sign precedes the integrand and the dx marks the end of the integrand, the order of precedence in the *Mathematica* expression above is different, with only the **x^2** term recognized as belonging to the integral (and no dx term following). For precedence reason, the proper entry should include parentheses.

$$\int \texttt{(x\^{}2 - x\^{}3 + Cos[x] - Log[x])}\ dx$$

The **Integrate** command is, of course, essentially the inverse of the (partial) differential command **D**. In particular, this means that **Integrate** assumes all symbols other than the specified variable of integration are constant with respect to the variable, as in the following expression, where **a** is treated as a constant:

```
Clear[a]
Integrate[ a*x^2, x ]
```

▷ $\frac{ax^3}{3}$

Integrate usually provides antiderivatives without regard to special cases. You're expected to recognize when special cases are needed, as in:

```
▶ Clear[n]
  Integrate[x^n,x]
```

▷ $\frac{x^{n+1}}{n+1}$

For the special case when $n = -1$, you must explicitly evaluate:

```
▶ Integrate[x^(-1),x]
```

▷ $\log(x)$

8.1.1 How *Mathematica* Integrates

The techniques implemented by *Mathematica* in finding symbolic antiderivatives are based on the ability to decompose the integrand into a specific combination of elementary functions, and then to integrate the result using integral table lookups, (trigonometric) identities and highly specialized logic. The following examples show why the way that you've been trained to see an integral may not be exactly the way that *Mathematica* sees it.

Example. To evaluate $\int \frac{dx}{4+9x^2}$ by hand, you would typically use the substitution $x = (2/3)\tan(u)$. In this case, $dx = (2/3)\sec^2(u)du$, so we write out

$$\int \frac{dx}{4+9x^2} = \int \frac{(2/3)\sec^2(u)du}{4+9\left((2/3)\tan(u)\right)^2} = \frac{2}{3}\left(\frac{1}{4}\right)\int \frac{\sec^2(u)du}{1+\tan^2(u)}$$
$$= \frac{1}{6}\int du = \frac{1}{6}u + C = \frac{1}{6}\tan^{-1}\left(\frac{3x}{2}\right) + C$$

(The last equality was generated from the original substitution by writing u in terms of x as $u = \tan^{-1}(3x/2)$.)

More generally, following the same substitution technique, we see that for any constants a and b,

$$\int \frac{dx}{a^2+b^2x^2} = \frac{1}{ab}\tan^{-1}\left(\frac{bx}{a}\right) + C$$

and this antiderivative formula appears in most integral tables. *Mathematica* has access to a fairly large integral table, and uses exactly this entry to evaluate the integral:

```
▶ Integrate[1/(4+9x^2),x]
```

▷ $\frac{1}{6}\tan^{-1}\left(\frac{3x}{2}\right)$

In this situation, *Mathematica* has looked up the answer directly, and not done the substitution that you might have carried out by hand. Indeed, *Mathematica* already knows the table formula in general terms:

```
▶ Integrate[1/(a^2 + b^2 9x^2),x]
```

▷ $\frac{\tan^{-1}\left(\frac{bx}{a}\right)}{ab}$

Forms given in integral tables may not always be so easily matched, as in the case of the following example where a student would have to complete a square.

Example. To evaluate $\displaystyle\int \frac{dx}{\sqrt{4x^2-12x+15}}$, we complete the square under the radical so that

$$\int \frac{dx}{\sqrt{4x^2-12x+15}} = \int \frac{dx}{\sqrt{(2x-3)^2+6}}$$

and then we perform a simple substitution of $u = 2x-3$ to obtain

$$\int \frac{dx}{\sqrt{(2x-3)^2+6}} = \frac{1}{2}\int \frac{du}{\sqrt{u^2+(\sqrt{6})^2}}$$

This form matches the standard integral table entry

$$\int \frac{dx}{\sqrt{x^2+a^2}} = \ln|x+\sqrt{x^2+a^2}| + C$$

and thus is easily resolved by writing

$$\frac{1}{2}\int \frac{du}{\sqrt{u^2+(\sqrt{6})^2}} = \frac{1}{2}\ln\left|u+\sqrt{u^2+(\sqrt{6})^2}\right| + C$$

With a final substitution for the variable u, we can write:

$$\begin{aligned}\int \frac{dx}{\sqrt{4x^2-12x+15}} &= \frac{1}{2}\ln\left|(2x-3)+\sqrt{(\sqrt{6})^2+(2x-3)^2}\right| + C\\ &= \frac{1}{2}\ln\left|(2x-3)+\sqrt{4x^2-12x+15}\right| + C\end{aligned}$$

How *Mathematica* evaluates this particular integral will be discussed shortly. However, it's reasonable to assume that either the mechanics of completing a square will be performed easily by *Mathematica* as above; or (better) that a new integration table entry could be formed as a result of the general case of completing the square in an integral such as $\displaystyle\int \frac{dx}{\sqrt{ax^2+bx+c}}$.

Example. To evaluate $\displaystyle\int \sin^5 x\,dx$ by hand, you can twice replace $\sin^2 x$ by $1-\cos^2 x$, expand the integrand and integrate the result. Alternatively, most students appeal to the reduction formula

$$\int \sin^n x\,dx = -\frac{1}{n}\sin^{n-1} x\cos x + \frac{n-1}{n}\int \sin^{n-2} x\,dx$$

which is valid for integers $n \geq 2$. Repeated application generates the sequence:

$$\begin{aligned}\int \sin^5 x\,dx &= -\frac{1}{5}\sin^4 x\cos x + \frac{4}{5}\int \sin^3 x\,dx\\ &= -\frac{1}{5}\sin^4 x\cos x + \frac{4}{5}\left(-\frac{1}{3}\sin^2 x\cos x + \frac{2}{3}\int \sin x\,dx\right)\\ &= -\frac{1}{5}\sin^4 x\cos x - \frac{4}{15}\sin^2 x\cos x + \frac{8}{15}\int \sin x\,dx\end{aligned}$$

$$= -\frac{1}{5}\sin^4 x\cos x - \frac{4}{15}\sin^2 x\cos x - \frac{8}{15}\cos x + C$$

Mathematica, however, does not actually use the reduction formula or make any substitutions using trigonometric identities:

```
► Integrate[Sin[x]^5,x]
```

▷ $-\frac{5\cos(x)}{8} + \frac{5}{48}\cos(3x) - \frac{1}{80}\cos(5x)$

However, this *is indeed* a proper antiderivative:

```
► D[%,x]//Simplify
```

▷ $\sin^5(x)$

What's happened is that *Mathematica* rewrote the integrand before integrating, generating integrals that are quite simple to do. In fact, this is what was integrated

```
► TrigReduce[Sin[x]^5]
```

▷ $\frac{1}{16}(10\sin(x) - 5\sin(3x) + \sin(5x))$

8.2 When Things Go Wrong with Integration

Blindly using **Integrate** can lead to cases either where the command simply fails, or produces a result that is unfamiliar or unexpected. This section notes a few of these situations.

8.2.1 Unfriendly or Unfamiliar Results

When *Mathematica* computes an antiderivative, there's a high probability of receiving extremely unfriendly solutions. Integral tables are often confounded by the presence of the inverse trigonometric or inverse hyperbolic functions.

Example. *Mathematica* does not go through the process of completing squares in the same way you do. While you might guess that $\int \frac{dx}{\sqrt{4x^2-12x+15}}$ will be recognized as $\int \frac{dx}{\sqrt{(2x-3)^2+6}}$, and matched against the proper table entry from that point (as was done earlier) to produce $\frac{1}{2}\ln\left|2x-3+\sqrt{4x^2-12x+15}\right| + C$, *Mathematica* produces a perhaps unfriendly result:

```
► answer = Integrate[1/Sqrt[4x^2-12x+15],x]
```

▷ $\frac{1}{2}\sinh^{-1}\left(\frac{2x-3}{\sqrt{6}}\right)$

This is a proper antiderivative, as can be checked by evaluating:

```
► D[answer,x]//Simplify
```

▷ $\frac{1}{\sqrt{4x^2-12x+15}}$

Indeed, the inverse hyperbolic sine function satisfies a simple identity with the logarithm function: $\sinh^{-1}(x) = \ln(x+\sqrt{x^2+1})$. The result given above is simply

$$\frac{1}{2}\ln\left|2x-3+\sqrt{4x^2-12x+15}\right| = \frac{1}{2}\ln\left|\left(\frac{2x-3}{\sqrt{6}}\right)+\sqrt{\left(\frac{2x-3}{\sqrt{6}}\right)^2+1}\right| + \frac{1}{2}\ln\sqrt{6}$$

$$= \frac{1}{2}\sinh^{-1}\left(\frac{2x-3}{\sqrt{6}}\right) + \frac{1}{2}\ln\sqrt{6}$$

So the answer is correct as it stands, up to a constant of integration, despite its appearance involving a function often not studied in much detail in a Calculus course.

In fact, this particular integral is most likely matched directly against the general formula

$$\int \frac{dx}{\sqrt{ax^2+bx+c}} = \frac{1}{\sqrt{a}}\ln(2ax+b+2\sqrt{a}\sqrt{ax^2+bx+c}) + C$$

which is the result if you formally carry out completion of the square. For the given coefficients, the result is then simplified using the inverse hyperbolic sine identity above. Indeed:

```
► Integrate[1/Sqrt[a*x^2+b*x+c],x]
```

$$\triangleright \frac{\log\left(2\sqrt{a}\sqrt{ax^2+bx+c}+2ax+b\right)}{\sqrt{a}}$$

Example. A similar situation arises in evaluating $\displaystyle\int \frac{dx}{x\sqrt{ax+b}}$, for which most standard Calculus texts provide the table entry:

$$\int \frac{dx}{x\sqrt{ax+b}} = \frac{1}{\sqrt{b}}\ln\left|\frac{\sqrt{ax+b}-\sqrt{b}}{\sqrt{ax+b}+\sqrt{b}}\right| + C \quad \text{when } b>0$$

```
► result = Integrate[1/(x Sqrt[a*x + b]),x]
```

$$\triangleright -\frac{2\tanh^{-1}\left(\frac{\sqrt{ax+b}}{\sqrt{b}}\right)}{\sqrt{b}}$$

Once again, *Mathematica* has used a simple relationship between the (real-valued) inverse hyperbolic tangent and the natural logarithm function:

$$\tanh^{-1}x = \frac{1}{2}\ln\frac{1+x}{1-x} \quad \text{for } -1<x<1$$

(we've seen this identity several times before!). So the answer above can be made massaged a little with the following statements that replace the inverse hyperbolic tangent function with the function $f(x) = \dfrac{1}{2}\ln\dfrac{1+x}{1-x}$:

```
► Clear[f]
  f[x_] := 1/2 Log[(1+x)/(1-x)]
  result /. ArcTanh→f // Simplify
```

$$\triangleright -\frac{\log\left(\frac{\sqrt{ax+b}+\sqrt{b}}{\sqrt{b}-\sqrt{ax+b}}\right)}{\sqrt{b}}$$

You'll be more comfortable with this answer, which is almost exactly the standard table lookup result you'd expect.

Of course, if $b<0$, *Mathematica*'s original answer using the inverse hyperbolic tangent function is still correct when interpreted with a complex-valued square root. It reduces to $\dfrac{2}{\sqrt{-b}}\tan^{-1}\sqrt{\dfrac{ax+b}{-b}} + C$ in this case, and relies on complex-valued identities involving the inverse tangent and inverse hyperbolic tangent. This will not be shown here.

8.2.2 Imaginary Results

Complications may further be introduced because *Mathematica* uses relationships among the trigonometric and exponential functions in the sense of complex numbers, variables, and functions. Consequently, computing a real integral may actually generate a complex-valued function (or so it might seem). A typical example comes in the case of $\int \frac{dx}{\sin x - \cos x}$:

```
► Clear[f]
  f[x_]:=1/(Sin[x]-Cos[x])
  answer = Integrate[f[x],x]
```

$$\triangleright\ (-1+i)\sqrt[4]{-1}\tanh^{-1}\left(\frac{\tan\left(\frac{x}{2}\right)+1}{\sqrt{2}}\right)$$

Note the presence of the complex number i in the result. More importantly, even if the result is meaningful to you, computing its derivative and manipulating it to reproduce the integrand is a non-trivial task.

Mathematically, the trigonometric functions and exponential functions are intimately related as *complex-valued* functions, and *Mathematica* often takes advantage of such relationships to produce complex-valued solutions (which may, ultimately, yield real values anyway). Two such identities that may be familiar to students of Complex Variables are: $e^{i\theta} = \cos\theta + i\sin\theta$, and $\tan^{-1}(ix) = i\tanh^{-1}(x)$.

If you're familiar with complex variables, you'll recognize that one reason *Mathematica* cannot simplify the result to a real-valued expression is the presence of the term $(-1)^{1/4}$. As a complex-valued expression, this could represent any of the four fourth-roots of -1. The principal fourth-root will be $\frac{1+i}{\sqrt{2}}$, and *Mathematica* can be forced to substitute this as the value of $(-1)^{1/4}$ with (assuming the expression above is the most recent evaluation):

```
► answer /. (-1)^(1/4)→(1+I)/Sqrt[2]
```

$$\triangleright\ -\sqrt{2}\tanh^{-1}\left(\frac{\tan\left(\frac{x}{2}\right)+1}{\sqrt{2}}\right)$$

In other words, once we say how to simplify $(-1)^{1/4}$, *Mathematica* is able to proceed with other simplifications in which the complex constants are algebraically absorbed.

8.2.3 Special Functions in Integration

In general, *Mathematica* can find antiderivatives for polynomials, many rational functions, trigonometric functions, and the like. Some integrations may require extra effort, as seen above. However, *Mathematica* can't find every antiderivative, simply because not every integrable function has an easily written antiderivative.

A standard example of such a case is $\int e^{-x^2}\,dx$, for which there is no closed-form antiderivative:

```
► Clear[f]
  f[x_] := Exp[-x^2]
  Integrate[f[x],x]
```

$$\triangleright\ \tfrac{1}{2}\sqrt{\pi}\,\mathrm{erf}(x)$$

This answer is returned in terms of the special function *erf*, whose value at x is defined

to be the definite integral

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt$$

Indeed, except for the particular choice of constant, this solution is nothing other than the original integral, written as a function of its upper limit of integration. The (First) Fundamental Theorem of Calculus guarantees that its derivative is exactly the correct multiple of f. That is,

$$\frac{d}{dx}\left(\frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt\right) = \frac{2}{\sqrt{\pi}} e^{-x^2}$$

The function *erf* is routinely recognized as the integral of the Gaussian distribution in probability theory having mean 0 and standard deviation 2, and is often called the *error function*. Its values are well known, and it appears with regularity in both probability and statistics.

A second example is the case of $\displaystyle\int \frac{\sin x}{x}\, dx$, which is again an integral for which no closed-form antiderivative exists. The First Fundamental Theorem of Calculus certainly guarantees that

$$\frac{d}{dx} \int_0^x \frac{\sin t}{t} dt = \frac{\sin x}{x}$$

and *Mathematica* reports as much:

▶ `Integrate[Sin[x]/x,x]`

▷ $\operatorname{Si}(x)$

A quick search through *Mathematica*'s help system reveals that *Si* represents the **SinIntegral** command (if written in **TraditionalForm**), where the value of **SinIntegral[x]** is exactly the same as "**Integrate[Sin[t]/t, {t,0,x}]**." In other words,

$$\operatorname{Si}(x) = \int_0^x \frac{\sin t}{t} dt$$

This particular integral arises often in applications, and it is for that reason that *Mathematica* knows about it.

Finally, other integrals will be reported in terms of special functions such as *erf* and *Si*. For example, the following calculation can be carried out using integration by parts, but produces the sine integral:

$$\int \frac{\cos x}{x^2}\, dx = \int \cos x \left(\frac{dx}{x^2}\right) = -\frac{\cos x}{x} - \int \frac{\sin x}{x}\, dx$$

▶ `Integrate[Cos[x]/x^2,x]`

▷ $-\operatorname{Si}(x) - \dfrac{\cos(x)}{x}$

8.3 Integrate & Definite Integrals

8.3.1 Using the Fundamental Theorem

Of central importance in the study of integration is the (Second) Fundamental Theorem of Calculus:

Theorem. *Let f be continuous on $[a, b]$, and suppose that there exists a continuous function F on $[a, b]$ such that $F'(x) = f(x)$, for $x \in (a, b)$. Then*

$$\int_a^b f(x)\,dx = F(x)|_{x=a}^{x=b} = F(b) - F(a)$$

A typical use of the Fundamental Theorem would be to compute the area under the curve $y = x^2$, over the interval $[1, 2]$ to be

$$\int_1^2 x^2\,dx = \left.\frac{x^3}{3}\right|_{x=1}^{x=2} = \frac{2^3}{3} - \frac{1^3}{3} = \frac{7}{3}$$

Mathematica's **Integrate** command carries out the calculation of definite integrals via the Fundamental Theorem whenever its second parameter is specified in the form of an iterator to denote an interval over which a definite integration is to be performed. In syntax, $\int_a^b f(x)\,dx$ is written as **Integrate[f[x],{x,a,b}]** in *Mathematica*. The integral above would be computed simply as:

► `Integrate[x^2,{x,1,2}]`

▷ $\frac{7}{3}$

Students sometimes forget to check whether a given integrand is actually integrable over a given interval and obtain erroneous results such as

$$\int_{-1}^1 \frac{1}{x^2}\,dx = \left.\frac{-1}{x}\right|_{x=-1}^{x=1} = -2 \quad \text{!!!}$$

In purely symbolic terms, the result is questionable, since the integrand appears to always positive; in fact, the answer is meaningless, since the integrand is neither continuous nor even bounded near 0. Fortunately, *Mathematica* recognizes such a situation:

► `Integrate[1/x^2,{x,-1,1}]`

▷ Integrate::idiv: Integral does not converge. Indeterminate

8.3.2 Tables of Definite Integrals

Integral tables usually have a collection of values of many definite integrals. In some cases, the values are listed only for convenience, since certain integrals appear frequently. A typical entry of this form might be:

$$\int_0^{\pi/2} \sin^n(x)\,dx = \int_0^{\pi/2} \cos^n(x)\,dx = \frac{2 \times 4 \times \ldots \times (n-1)}{3 \times 5 \times \ldots \times n}, n \text{ an odd integer, } n \geq 3$$

Certainly, such integrals could be evaluated directly using reduction formulae, but it's reasonable to believe that *Mathematica* recognizes definite integrals such as the one above and returns a result without actually performing the integration.

Other definite integrals appear in tables because their value is known, despite the fact that their antiderivative may not be known. An example of this form is:

$$\int_0^\infty e^{-x^2}\,dx = \frac{\sqrt{\pi}}{2}$$

Mathematica produces the correct result, since the special function *erf* on which the integral is based is well known, as we discussed earlier.

```
► Integrate[Exp[-x^2],{x,0,Infinity}]
```

$\triangleright \dfrac{\sqrt{\pi}}{2}$

Similarly, some definite integrals involving SinIntegral are known, since well-known techniques learned in a Complex Variables course can be applied to transform them into integrals in the complex plane. For example

```
► Integrate[Sin[a*x]^2/x^2,{x,0,Infinity}]
```

$\triangleright \dfrac{\pi\,|a|}{2}$

8.4 Numerical Integration

8.4.1 N and Integrate

The results of definite integrals obtained from **Integrate[f[x],{x,a,b}]** will, of course, be symbolic. If the antidifferentiation can be performed or otherwise recognized according to its form, then a numerical approximation for the value of the integral can be found simply by applying **N** to the result:

```
► Integrate[x^2,{x,1,2}] // N
```

▷ 2.33333

Here, the integral $\displaystyle\int_1^2 x^2\,dx$ is first evaluated symbolically as $\left.\dfrac{x^3}{3}\right|_1^2 = \dfrac{8}{3} - \dfrac{1}{3} = \dfrac{7}{3}$, and a numerical approximation of the result is given.

8.4.2 NIntegrate

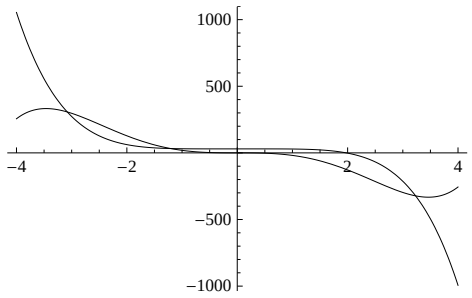
Applying **N** to the result of **Integrate** can be rather time-consuming, since a symbolic antidifferentiation is attempted first. Preferably, the **NIntegrate** command should be used whenever numerical integration results will be acceptable.

In syntax, **NIntegrate[f[x],{x,a,b}]** causes a numerical approximation to be generated for the integral $\displaystyle\int_a^b f(x)\,dx$. Numerical techniques do not perform any symbolic integration and are generally returned quickly (in the same way that **NSolve** outperforms **Solve**).

Methods by which **NIntegrate** produces its results will be studied in a course on Numerical Analysis. Reasonably accurate methods for numeric integration that you have probably studied already include the Trapezoidal, Midpoint and Simpson's Rules. *Mathematica* uses much more sophisticated methods allowing a very high degree of accuracy (which can be user-controlled using appropriate options with **NIntegrate**).

Example. Consider approximating the area bounded by the curves $p(x) = x^5 - 20x^3$ and $q(x) = 30 - x^5$. A graph of these curves near the origin gives:

```
► Clear[p,q]
  p[x_] := x^5 - 20 x^3
  q[x_] := 30 - x^5
  Plot[{p[x],q[x]},{x,-4,4}]
```



These curves intersect in (most likely only) three places, and numerical approximations of the x-coordinates of the intersections can be found using **NSolve**:

```
▶ NSolve[p[x]==q[x],x]
```

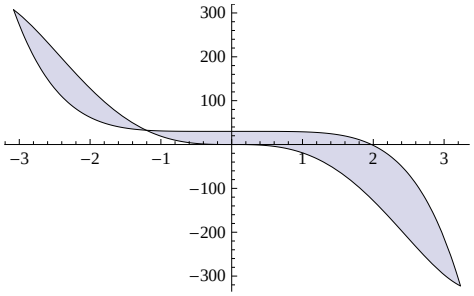
▷ $\{\{x \to -3.08004\}, \{x \to -1.20632\}, \{x \to 0.527291 - 0.985474i\}, \{x \to 0.527291 + 0.985474i\}, \{x \to 3.23178\}\}$

The real values $a = 3.08004$, $b = 1.20632$ and $c = 3.23178$ then determine, with reasonable precision, the x-coordinates of the intersections. We'll carry these over into *Mathematica*with

```
▶ a = x /. solutions[[1]];
  b = x /. solutions[[2]];
  c = x /. solutions[[5]];
```

Additionally, to highlight the area between the curves over an interval containing these points, use the **Plot** command together with the **Filled** option (*although we'll leave it to the reader to research why the syntax of the FilledPlot option is what it is*).

```
▶ Plot[{p[x],q[x]}, {x,a,c}, Filling→{1→{2}}]
```



To compute the area bounded by the curves, notice that the graph of p lies above that of q on the (approximate) interval $[a, b]$, while the graph of p lies below that of q on the (approximate) interval $[b, c]$. Hence the area between the curves will be given (approximately) by:

$$\int_a^b p(x) - q(x)\, dx + \int_b^c p(x) - q(x)\, dx$$

Mathematica can then produce an approximation of the area bounded by the curves with:

```
▶ Integrate[p[x]-q[x],{x,a,b}] + Integrate[q[x]-p[x],{x,b,c}]
```

▷ 388.854

More appropriately, though, the area can be written simply as $\int_a^c |p(x) - q(x)|\,dx$, without regard for the points at which the curves intersect within the interval $[a, c]$. While a symbolic antiderivative is not easily computed involving the absolute value of the integrand, a numeric result is readily available:

```
► NIntegrate[Abs[p[x]-q[x]], {x,a,c}]
```

▷ 388.854

8.5 Exercises

1. Evaluate each of the following using *Mathematica*. Check your answers using D as best you can to obtain the original integrand.

(a) $\displaystyle\int \frac{\sqrt{x}}{\sqrt{1+\sqrt[4]{x}}}\,dx$ (This should be integrable directly)

(b) $\displaystyle\int \sec^5 x\,dx$ (Check the result against the standard reduction formula)

(c) $\displaystyle\int \cos^8 x\,dx$ (Check the result against the standard reduction formula)

2. Consider working with $\displaystyle\int \frac{\cos x\,dx}{\cos^2 x + \sin x}$ by hand. A simple substitution along with rewriting $\cos^2 x = 1 - \sin^2 x$ should yield an antiderivative. Attempt to work with the same integral in *Mathematica*. What success can you achieve?

3. Evaluate each of the following in *Mathematica*. Verify that the expression returned is an antiderivative by evaluating D and manipulating the expression to be the original integrand as best as you can.

(a) $\displaystyle\int \frac{dx}{x\sqrt{ax+b}}$

(b) $\displaystyle\int \frac{\sqrt{2ax-x^2}}{x}\,dx$

(c) $\displaystyle\int \frac{dx}{x^2\sqrt{ax+b}}$

(d) $\displaystyle\int \frac{\sqrt{2ax-x^2}}{x^2}\,dx$

4. Demonstrate the necessary *Mathematica* syntax to find the solution to the differential equation $y' = x^3 - 3\sin x$ that satisfies the initial condition $y(1) = 2$. (*A proper sequence would involve using* ***Integrate****, defining a general antiderivative by adding an arbitrary constant of integration to the result, then using* ***Solve*** *to determine the value of the constant.*)

5. Verify that *Mathematica* cannot produce a closed-form antiderivative for $\displaystyle\int \frac{\cos(bx)\,dx}{x^2+a^2}$, but can produce a value for $\displaystyle\int_0^\infty \frac{\cos(bx)\,dx}{x^2+a^2}$. What is the value of $\displaystyle\int_0^x \frac{\cos(bt)\,dt}{t^2+a^2}$ in *Mathematica*?

6. Verify that *Mathematica* produces a closed-form antiderivative for $\displaystyle\int \frac{\sin^2(ax)}{x^2}\,dx$ in terms of **SinIntegral**, and that it produces an exact value for $\displaystyle\int_0^\infty \frac{\sin^2(ax)}{x^2}\,dx$. Show

by hand how $\displaystyle\int \frac{\sin^2(ax)}{x^2}\,dx$ may be rewritten in terms of **SinIntegral**.

7. Verify that *Mathematica* produces a closed-form antiderivative for $\displaystyle\int \cos x^2\,dx$ in terms of **FresnelC**, and that it produces an exact value for $\displaystyle\int_0^\infty \cos x^2\,dx$. Show by hand how $\displaystyle\int_0^\infty \cos x^2\,dx$ may be rewritten in terms of **FresnelC**.

8. Verify that *Mathematica* produces a closed-form antiderivative for $\displaystyle\int \frac{\sin x}{\sqrt{x}}\,dx$ in terms of **FresnelS**, and that it produces an exact value for $\displaystyle\int_0^\infty \frac{\sin x}{\sqrt{x}}\,dx$. Show by hand how $\displaystyle\int \frac{\sin x}{\sqrt{x}}\,dx$ may be rewritten in terms of **FresnelS**.

9. Estimate the area between the curves $f(x) = x^4 - 3x^2 - 2x$, $g(x) = x^3 + 2x^2 - 1$, and the lines $x = -1$ and $x = 3$. Use **Plot** with the **FilledPlot** option to first identify the area, and then **NIntegrate** with the appropriate results extracted in *Mathematica* from **NSolve** to produce the approximation.

9 Differential Equations

This Chapter introduces the notion of an ordinary, first-order differential equation and shows how to find a numerical solution using Euler's Method.

9.1 Ordinary, First-Order Differential Equations

9.1.1 Terminology & Examples

Definition. *An ordinary, first-order differential equation is an equation that involves only an independent variable x, a function y of the variable x, and its derivative $\dfrac{dy}{dx}$, and that can be written in the form*

$$\frac{dy}{dx} = F(x, y).$$

A differentiable function $y = f(x)$ is said to be a solution to this equation over a domain D if $f'(x) = F(x, f(x))$, for all $x \in D$.

We'll demonstrate the terminology of this definition with three examples. Only in the first example will we show how you can actually find the proposed solution. Numerical techniques for finding solutions are discussed later in the Chapter.

Example 1. Given the differential equation $\dfrac{dy}{dx} = -32x$, the function $y = f(x) = -16x^2$ is a solution of the equation, for all real numbers x. Many other functions are also solutions to the equation, although all must be of the form $y = f(x) = -16x^2 + C$, where C is a constant.

Mathematica can verify that this function satisfies the differential equation (note: we use a lower-case letter for the constant of integration here, because the uppercase C already has a meaning in *Mathematica*):

```
► Clear[x,c,f]
  f[x_] := -16x^2 + c
  f'[x] == -32x
```

▷ True

In fact, the solution to this equation is easy to find, since you know from integration that $\int -32x\, dx = 16x^2 + C$. However, since the right-hand side of the equation $\dfrac{dy}{dx} = F(x, y)$ will not always be a function of x alone, simple integration won't always be enough to find a solution.

Example 2. One solution of the differential equation $\dfrac{dy}{dx} = -2xy$ is the function $y = f(x) = e^{-x^2}$, for all real x. Additionally, any function of the form $y = f(x) = Ce^{-x^2}$, where C is a constant, will also be a solution to this equation. Again, it is easy to verify that this function satisfies the differential equation:

```
► Clear[x,c,f]
  f[x_] := c Exp[-x^2]
  f'[x] == -2*x*f[x]
```

▷ True

Note that in this case, you cannot simply integrate to discover a solution, because the integrand of $\int -2xy\, dx$ is ambiguous. If y were a constant, we could do the integration, but here, y depends on x.

Example 3. Consider the equation $\dfrac{dy}{dx} = -\dfrac{x}{y}$ (defined only for values of $y \neq 0$). Any function y that satisfies $x^2 + y^2 = C$, with $C > 0$, will be a solution.

This can be verified using implicit differentiation. Indeed, differentiating the equation $x^2 + y^2 = C$ with respect to x, and treating y as a function of x, we get $2x + 2y\dfrac{dy}{dx} = 0$, and upon solving for the term $\dfrac{dy}{dx}$, we see that $\dfrac{dy}{dx} = -\dfrac{x}{y}$.

In this case, a solution is defined implicitly via the equation $x^2+y^2 = C$, although explicit, differentiable solutions will be of the form $y = f(x) = \pm\sqrt{C - x^2}$, defined for x satisfying $-\sqrt{C} \leq x \leq \sqrt{C}$ with $C \geq 0$. Frequently, we will not be able to find explicit solutions to a differential equation, so we will have to settle for implicitly-defined solutions.

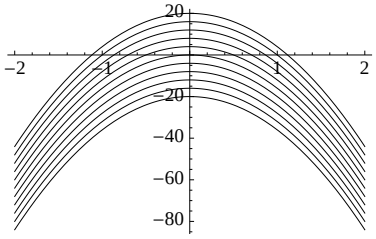
9.1.2 Particular Solutions to Equations

As we've seen, solutions to differential equations usually will not be unique. For equations in the form $\dfrac{dy}{dx} = F(x, y)$, solutions will, in fact, be members of a one-parameter family of functions; that is, all solutions will be determined uniquely up the specification of a single constant.

In the examples above, the families of solutions were $\{-16x^2 + C : C \text{ is constant}\}$, $\{Ce^{-x^2} : C \text{ is constant}\}$, and $\{x^2 + y^2 = C : C \text{ is constant}\}$, respectively.

To specify a solution uniquely, more information must be given than an equation alone. To motivate what type of additional condition would be sufficient, remember that each of these families of solutions represents a collection of curves in the plane. For example, $\{-16x^2 + C : C \text{ is constant}\}$ represents a collection of parabolas, all opening down, with vertex on the Y-axis. Indeed, consider making a Table of such functions and Plotting them:

```
Clear[c,x]
functions = Table[ -16x^2 + c, c,-20,20,4 ];
Plot[Evaluate[functions],x,-2,2]
```



To identify any of these several curves uniquely, it's enough to specify one point through which the curve passes. For example, if we want the solution to the equation $\dfrac{dy}{dx} = -32x$ to also be a function whose graph passes through $(1, 1)$, we must choose the value of C so that $-16(1)^2 + C = 1$, or choose $C = 17$.

This notion of finding one particular solution to the differential equation by attaching an identifying characteristic leads to the following:

Definition. *If f is a solution to a differential equation $\dfrac{dy}{dx} = F(x, y)$ on a domain D, then f is said to satisfy an initial condition at a point $x_0 \in D$ of the form $y(x_0) = y_0$ if $f(x_0) = y_0$.*

Example 4. Given the differential equation $\dfrac{dy}{dx} = -2xy$ together with the initial condition $y_0 = y(1) = 2$, any function of the form $y = Ce^{-x^2}$ will be a solution of the equation (as we saw previously), where C is a constant. However, only the particular solution $y = (2e)e^{-x^2} = 2e^{1-x^2}$ additionally satisfies the initial condition $y_0 = y(1) = 2$.

9.2 Algebraic Solutions to Differential Equations

A differential equations course is where you'll learn the algebra of how to find symbolic solutions to a given differential equation. Calculus students have likely seen at least the simplest of symbolic techniques called "separation of variables," which can be used with the equation of Example 2.

Example 5. We reconsider the differential equation $\dfrac{dy}{dx} = -2xy$. Treating the terms dy and dx formally, we rewrite the equation with all references to y on the left side, and all references to x on the right side as

$$\frac{dy}{y} = -2x dx.$$

Formally integrating both sides, we have

$$\int \frac{dy}{y} = \int -2x\ dx, \text{ or } \ln(y) = -x^2 + C$$

where C denotes an arbitrary constant of integration. Exponentiating, we get

$$y = e^{\ln(y)} = e^{-x^2+C} = e^C e^{-x^2} = C' e^{-x^2},$$

where C' denotes an arbitrary constant of integration. In other words, the formalism says that every solution to the equation has the form $y = f(x) = Ce^{-x^2}$, where C is a constant.

We will not discuss any other symbolic techniques here, but you'll want to know that *Mathematica* has a built-in operator that implements most of the standard, symbolic techniques used to find solutions to differential equations.

Indeed, the **DSolve** operator can usually provide a closed-form, symbolic solution for many ordinary, first-order differential equations. Arguments for **DSolve** include the independent variable (`x` below), a function that depends on `x` (`y` below, that must be written in the form `y[x]`), and the equation to be solved (where `y'[x]` denotes the derivative of the function `y` below).

DSolve gives the following results for the equations of Examples 1, 2, and 3 as:

- ▶ `DSolve[y'[x]==-32x, y[x], x]`
- ▷ $\{\{y(x) \to c_1 - 16x^2\}\}$
- ▶ `DSolve[y'[x]==-2x*y[x], y[x], x]`
- ▷ $\left\{\left\{y(x) \to c_1 e^{-x^2}\right\}\right\}$
- ▶ `DSolve[y'[x]==-x/y[x], y[x], x]`
- ▷ $\left\{\left\{y(x) \to -\sqrt{2c_1 - x^2}\right\}, \left\{y(x) \to \sqrt{2c_1 - x^2}\right\}\right\}$

In all cases, the symbol c_1 denotes some constant of integration inserted in the solution process. (The fact that $2c_1$ appears in the third of the solutions above should not be a problem, since such an expression is as much an arbitrary constant as c_1 itself.)

If an initial condition is specified for a given differential equation, it may be added as another equation, as the first argument of **DSolve** is allowed to be a *list* of equations. If we use *Mathematica* to solve the problem of Example 4, solving the equation $\dfrac{dy}{dx} = -2xy$ subject to the initial condition $y_0 = y(1) = 2$, then we write:

```
▶ DSolve[y'[x]==-2x*y[x],y[1]==2, y[x], x]
```

$\triangleright \left\{\left\{y(x) \to 2e^{1-x^2}\right\}\right\}$

Not every ordinary, first-order differential equation can be solved using **DSolve**. Given the differential equation $\dfrac{dy}{dx} = \sin(x) + e^y$, **DSolve** has no explicit solution (the exact output will not be shown here, since the computation took several minutes and resulted in an indefinite integral).

```
▶ DSolve[y'[x]==Sin[x]+Exp[y[x]],y[x],x]
```

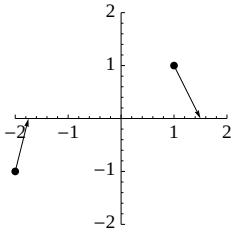
In fact, just as we learned earlier with the **Integrate** operator, not every first-order differential equation has a solution that can be written in closed form. Our hope in general is that we will be able to find and be satisfied with numerical solutions for the equations.

9.3 The Geometry of Differential Equations

The problem of finding solutions to differential equations in the form $\dfrac{dy}{dx} = F(x, y)$ can be viewed quite easily in geometric, rather than algebraic terms. Suppose that $y = f(x)$ is a solution to the equation. The derivative $\dfrac{dy}{dx}$ gives the slope of the line tangent to the graph of f at the point $(x, f(x))$. Requiring that f be a solution of the equation is to require that whenever (x, y) is on the graph of f, the tangent slope be given by the value of $F(x, y)$.

To see what this means, reconsider Example 2, where the differential equation is given to be $\dfrac{dy}{dx} = F(x, y) = -2xy$. Let $\mathscr{F}$ denote a small, finite collection of equally-spaced points (x, y) in the plane for values $-2 \le x, y \le 2$. At each point (x, y), sketch a short arrow segment to the right with slope $F(x, y) = -2xy$.

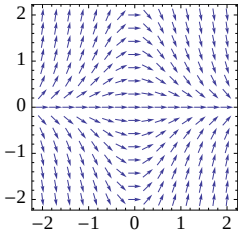
For example, at $(-2, 1)$, sketch a short arrow segment with slope $2(2)(1) = 4$; at $(1, 1)$, sketch a short arrow segment with slope $2(1)(1) = 2$. The arrow is to point "to the right" in all cases, to indicate that from that point, we continue with increasing x-coordinate in the direction of the arrow. The arrows should be the same length at all points. The diagram below illustrates the concept.



Such a collection of points and arrows is technically a *vector field* and is usually studied in the later semesters of Calculus. However, in this case, only the slope of the arrow is of interest. The *Mathematica* operator **VectorField** presents such vector fields graphically,

although it must be loaded from an external package. Hence, the collection of points $\mathscr{F}$ and the short arrow segments attached at each point is shown with:

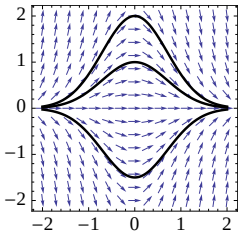
```
▶ field = VectorPlot[{1, -2 x*y}, {x, -2, 2}, {y, -2, 2},
        VectorScale → {0.05, 1, None}]
```



The arguments for **VectorField** may require a little research in the Help pages, but for the moment, it is enough to identify the arguments {x,−2,2} and {y,−2,2} as defining the x- and y-coordinate range. The arguments for the **VectorScale** option guarantee that all vectors are drawn at the same length and fit comfortably into the graphic.

If $y = f(x)$ is to be a solution to the equation $\dfrac{dy}{dx} = F(x, y) = -2xy$, then whenever its graph contains one of the illustrated points from which the arrow segments are drawn, the arrow segment will be tangent to the graph of f. All solutions to this particular equation are known to be given in the form $y = Ce^{-x^2}$, for some constant C. Plotting just a few such solutions, say for C having values from among $\{1, -1.5, 2\}$, we can see how these solutions match up graphically with the collection of tangent arrow segments.

```
▶ solutions = Plot[{Exp[-x^2],-1.5Exp[-x^2],2Exp[-x^2]},{x,-2,2},
        Axes→False,PlotStyle→Thickness[0.012]]
  Show[field,solutions]
```



To conclude, any ordinary, first-order differential equation provides a specification of tangent slopes. The graphs of solutions to such equations necessarily follow the directions prescribed by these tangent arrows.

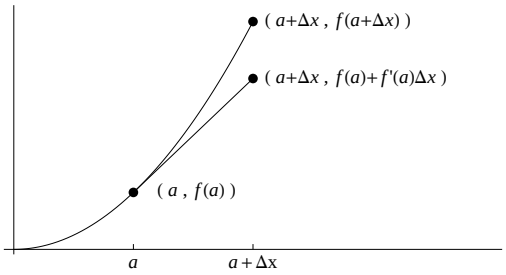
9.4 Numerical Solutions to Differential Equations

Numerical solutions are our best hope for solving arbitrary differential equations. The first subsection below provides a *Mathematica* implementation of the well-known Euler's Method and will be of interest for pedagogical reasons. For the reader who wishes to quickly see what numerical solution operators *Mathematica* has available, the second subsection below provides the basic details.

9.4.1 Euler's Method – The Hands-On Experience

Suppose that f is a solution to the differential equation $\dfrac{dy}{dx} = F(x, y)$ on the interval $[a, a + \Delta x]$, where $\Delta x > 0$. Suppose f satisfies the initial condition $f(a) = y_0$. We may not know the value $f(a+\Delta x)$ exactly, but we can certainly approximate the value $y_1 = f(a+\Delta x)$ by using a tangent line approximation for f.

Indeed, if Δx is small and f is reasonably well-behaved, then the value $y_1 = f(a + \Delta x)$ should not be significantly different from the value projected by its tangent line at $x = a$. That is, as we see below, the value $y_1 = f(a + \Delta x)$ should be approximated by the value $\tilde{y}_1 = f(a) + f'(a)\Delta x$.



This leads naturally to generating an approximate, numerical solution to a differential equation over an interval $[a, b]$ with initial condition $f(a) = y_0$. We partition the interval $[a, b]$ into subintervals of short length, and project the solution across each of the subintervals using tangent line approximations as we described above. That process is now described in detail, and is due to Leonhard Euler.

Euler's Method. Suppose that at each point (x, y) in the plane, $\dfrac{dy}{dx}$ has a specified value $F(x, y)$. We wish to determine a function f for which $f'(x) = F(x, f(x))$ for all x in some interval $[a, b]$, given a prescribed initial value $f(a) = y_0$. That is, f is to be a solution to the equation $\dfrac{dy}{dx} = F(x, y)$ that satisfies the initial condition $f(a) = y_0$.

To describe an approximate, numerical representation of the solution f,

- Subdivide the interval $[a, b]$ into n closed subintervals of equal length, with disjoint interiors. That is, write
$$[a, b] = [a, x_1] \cup [x_1, x_2] \cup \ldots \cup [x_{n-1}, b]$$
If $\Delta x = \dfrac{b-a}{n}$ represents the common width of these subintervals, then the endpoints of these subintervals are given as $x_i = a + i\Delta x$, for $1 \le i \le n$. It is convenient in this representation to write $a = x_0$ and $b = x_n$.
- Use a tangent line approximation for f at the point (a, y_0) to estimate the value $f(x_1) = f(a + \Delta x)$. Since the derivative at (a, y_0) is exactly $F(a, y_0)$, observe that $f(x_1)$ will then be approximated by $\tilde{y}_1 = y_0 + F(a, y_0)\Delta x$. For notational convenience, we write $\tilde{y}_0 = y_0$.
- Having obtained estimated function values $\{\tilde{y}_0, \tilde{y}_1, \ldots, \tilde{y}_i\}$ at the points $\{x_0, x_1, \ldots, x_i\}$, using tangent line approximations at the points $\{(x_0, \tilde{y}_0), (x_1, \tilde{y}_1), \ldots, (x_i, \tilde{y}_i)\}$, an estimated value for $f(x_{i+1})$ is obtained using the tangent line approximation with $\tilde{y}_{i+1} = \tilde{y}_i + F(x_i, \tilde{y}_i)\Delta x$.
- Repeat this process n times, to obtain points $\{(x_0, \tilde{y}_0), (x_1, \tilde{y}_1), \ldots, (x_n, \tilde{y}_n)\}$ to numerically represent the solution function f.

9.4.1.1 Implementing Euler's Method in *Mathematica*

First, we will reserve certain notation in the current, global *Mathematica* environment. This will greatly simplify the *Mathematica* you see below.

We're interested in finding a numerical approximation for a function f that satisfies a differential equation $\dfrac{dy}{dx} = F(x, y)$ on an interval $[a, b]$, using n subinterval approximations with $\Delta x = \frac{b-a}{n}$, subject to an initial condition $f(a) = y_0$.

Thus, we'll assume from now on that

- F denotes the function $\dfrac{dy}{dx} = F(x, y)$,
- a and b denote endpoints of the interval $[a, b]$,
- we're using n subintervals of common width $\Delta x = \dfrac{b-a}{n}$, and that
- y_0 represents the value for the initial condition $f(a) = y_0$.

Euler's approximation method is now very naturally implemented in *Mathematica* using **NestList** to produce the desired sequence of points $\{(x_i, \tilde{y}_i) : 0 \leqslant i \leqslant n\}$, by repeatedly applying a transition function $t = t_{F,\Delta x}$ that transforms a point $(x, \tilde{y})$ into the point $(x + \Delta x, \tilde{y} + F(x, \tilde{y})\Delta x)$. The definition of the transition function can be given as follows:

```
▶ Clear[t]
  t[{x_,y_}] := N[{x+dx,y+F[x,y]*dx}]
```

The **N** operator is used in the definition above to ensure a numerical, rather than symbolic result (which would be unwieldy and slow to execute). The examples that follow will assume that we've made this definition.

Example 6. Consider numerically estimating a solution for the equation $\dfrac{dy}{dx} = -2xy$ on the interval $[a, b] = [0, 2]$, subject to the initial condition $y(0) = y_0 = 5e$, using $n = 10$ subintervals.

First, a symbol to represent the function F of the differential equation is given. Additionally, we define symbols for the interval $[a, b]$, the number of subintervals and the initial value y_0.

```
▶ Clear[F,a,b,n,y0]
  F[x_,y_] := -2x*y
  a = 0.0;
  b = 2.0;
  n = 10;
  y0 = N[5E];
```

Since the solution satisfies the initial condition $f(a) = f(0) = y_0 = 5e$, the point $(0, 5e)$ lies on the solution curve. The transition function t will be repeatedly applied, starting at this point, a total of $n = 10$ times, and the distance Δx traversed by each iteration will be $\Delta x = \dfrac{2-0}{10} = \dfrac{2}{10}$. After defining Δx, we get a list of approximate values for the solution:

```
▶ dx = (b-a)/n;
  approximateValues = NestList[t,{a,y0},n]
```

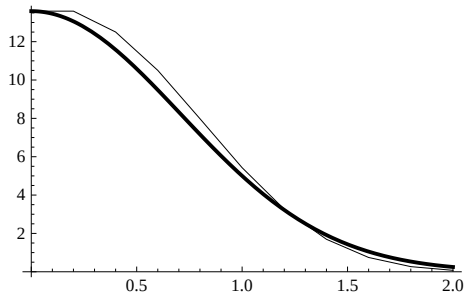
```
▷ {{0., 13.5914}, {0.2, 13.5914}, {0.4, 12.5041}, {0.6, 10.5034}, {0.8, 7.98262},
   {1., 5.42818}, {1.2, 3.25691}, {1.4, 1.69359}, {1.6, 0.74518},
   {1.8, 0.268265}, {2., 0.0751142}}
```

The result of this computation is exactly the list $\{(x_i, \tilde{y}_i) : 0 \leqslant i \leqslant n\}$. Using **ListPlot**,

we can see the result (shown below)..

The actual solution of this equation is $y = f(x) = (5e)e^{-x^2} = 5e^{1-x^2}$. Using this, the approximate values are now compared graphically (connecting the points with thinner line segments) with the actual solution (the thicker curve) to the equation over the interval $[0, 2]$:

```
Clear[f]
f[x_] := 5.0 Exp[1-x^2]
approximation = ListPlot[approximateValues,
    Joined→True,
    PlotStyle→{Black,AbsoluteThickness[0.5]}];
actualsolution = Plot[f[x],{x,0,2},
    PlotStyle→{Black, Thick}]
Show[actualsolution,approximation]
```



Note that the actual solution (thicker curve) is concave down in $[0, 1]$, where tangent lines will be above the curve. Hence, Euler's tangent line approximation will necessarily overestimate the true solution. Over the interval $[1, 2]$, the actual solution is concave up (where tangent lines will be below the curve). Graphically, Euler's tangent line approximation then underestimates the true solution throughout most of $[1, 2]$.

Example 7. Consider the problem of solving the equation $\dfrac{dy}{dx} = y \cos x$, on the interval $[1, 7]$, subject to the initial condition $y(1) = 1$, using $n = 20$ subdivisions.

First, the differential equation will be represented in the global environment using the symbols:

```
Clear[F,a,b,n,y0]
F[x_,y_] := y*Cos[x]
a = 1.0;
b = 7.0;
n = 20;
y0 = 1.0;
```

The desired solution satisfies the initial condition $y(1) = y_0 = 1$, so the point $(1, 1)$ must lie on the solution curve. The transition function $t = t_{F,\Delta x}$ will be repeatedly applied, starting at this point, a total of $n = 20$ times, and the distance Δx traversed by each iteration will be $\Delta x = (7-1)/20 = 6/20$. The approximate values for the solution are easily generated:

```
dx = (b-a)/n;
approximateValues = NestList[t,{a,y0},n]
```

▷ {{1., 1.}, {1.3, 1.16209}, {1.6, 1.25535}, {1.9, 1.24435}, {2.2, 1.12367}, {2.5, 0.925282}, {2.8, 0.702897}, {3.1, 0.504211}, {3.4, 0.353079}, {3.7, 0.250672}, {4., 0.186894}, {4.3, 0.150245}, {4.6, 0.13218}, {4.9, 0.127732}, {5.2, 0.134879}, {5.5, 0.153837}, {5.8, 0.186543}, {6.1, 0.2361}, {6.4, 0.305744}, {6.7, 0.396843}, {7., 0.505703}}

The general solution to this equation is a function of the form $f(x) = Ce^{\sin(x)}$, where C is a constant. By choosing $C = \dfrac{1}{e^{\sin 1}} = e^{-\sin 1}$, a solution $f(x) = e^{-\sin(1)}e^{\sin(x)} = \frac{e^{\sin(x)}}{e^{\sin(1)}}$ is found that satisfies $f(1) = 1$. We're not concerned in this chapter with how you go about finding this to be the solution, but we can have *Mathematica* verify what we said:

```
▶ Clear[f]
  f[x_] := Exp[Sin[x]]/Exp[Sin[1]]
  f'[x] == f[x] Cos[x] (* does f satisfy the equation? *)
```

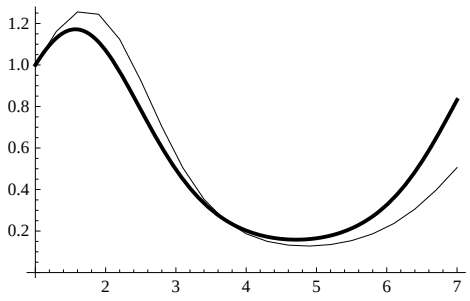
▷ True

```
▶ f[1] == 1 (* is f[1] equal to 1? *)
```

▷ True

Graphically combining both the estimated solution and the exact solution generates:

```
▶ approximation = ListPlot[approximateValues,
      Joined→True,
      PlotStyle→{Black,AbsoluteThickness[0.5]}];
  actualsolution = Plot[f[x],{x,1,7},
      PlotStyle→{Black, Thick}]
  Show[actualsolution,approximation, PlotRange→All]
```



Note that when the actual solution is concave down (approximately in the interval $[1, 2.5]$), the estimated solution tends to overestimate the actual solution. The estimated solution underestimates the actual solution when the actual solution is concave up (approximately in the interval $[2.5, 7]$). This behavior is necessarily the case in general with estimated solutions, since they are based on tangent line approximations.

9.4.1.2 Error Analysis with Euler's Method

To compare Euler's results numerically with an actual solution, we can generate a table of numerical values of the solution across a given interval, and compare them directly with those of the estimated solution. In the case of Example 7, where approximate solution values were obtained throughout $[1, 7]$ at equally-spaced x-values differing by $\Delta x = \frac{6}{20}$, values for the actual solution $f(x) = \frac{e^{\sin(x)}}{e^{\sin(1)}}$ can be given with:

```
▶ actualValues = Table[N[{x,f[x]}],{x,1,7,6/20}]
```

▷ {{1., 1.}, {1.3, 1.12985}, {1.6, 1.17129}, {1.9, 1.11052}, {2.2, 0.967563}, {2.5, 0.784272}, {2.8, 0.602611}, {3.1, 0.449378}, {3.4, 0.333867}, {3.7, 0.253775}, {4., 0.202245}, {4.3, 0.172452}, {4.6, 0.159588}, {4.9, 0.161391}, {5.2, 0.178186}, {5.5, 0.212883}, {5.8, 0.270882}, {6.1, 0.359287}, {6.4, 0.484362}, {6.7, 0.646216}, {7., 0.831533}}

Using listability and the **Map** operator to pick out only the y-coordinates, the differences between y-coordinates of the estimated and actual solutions can be given with:

```
► errors = N[Map[Last,actualValues-approximateValues]]
```

▷ {0., -0.0322381, -0.0840617, -0.133831, -0.156102, -0.14101, -0.100286, -0.0548331, -0.0192116, 0.00310302, 0.0153519, 0.0222069, 0.0274081, 0.033659, 0.0433069, 0.0590459, 0.0843383, 0.123187, 0.178618, 0.249374, 0.32583}

The largest of these differences between estimated solution and actual solution, in absolute value, can be found with:

```
► Max[Abs[errors]]
```

▷ 0.32583

The calculation above may be repeated using $n = 40$, 80 and 160 subintervals, with the absolute value of the largest error between estimated and actual values reported, to produce the following table:

n	largest error
20	0.32583
40	0.0.179613
80	0.0949202
160	0.0488847

These results provide evidence that as the number of subintervals doubles, the largest error between observed and estimated values is approximately halved. This suggests (and it indeed can be shown) that Euler's Method is generally an order-one method; that is, the error introduced by the method is approximately proportional to $\frac{1}{n}$. Hence, increasing n will generally improve accuracy of the estimation procedure – *but one would certainly be interested in finding better numerical techniques to speed up convergence!*

9.4.2 *Mathematica* Support for Numerical Solutions

NDSolve is primary operator for finding numerical solutions to differential equations in *Mathematica*.

Example. To find a solution to the differential equation $\dfrac{dy}{dx} = F(x, y) = -2xy$, subject to the initial condition $y(1) = 2$ over the interval $[-2, 2]$, (the problem of of Example 2), we use

```
► solution = NDSolve[{y'[x]==-2 x*y[x], y[1]==2}, y[x], {x,-2,2}]
```

▷ $\{\{y(x) \to \text{InterpolatingFunction}\left[(-2.\quad 2.), <>\right](x)\}\}$

The output is, indeed, unusual for our experience, and includes a reference to a special *Mathematica* operator named **InterpolatingFunction**. As a user, you don't really to need to know much more than that a result defined in terms of an **InterpolatingFunction** is a complete, numerical method for giving approximate solutions to the differential equation

on the interval $[-2, 2]$.

To work with the result, you must strip out one enclosing layer of curly braces and use the "dummy variable trick" for the value of $y(x)$ that we've seen before to turn this into a function definition (*note: the use of =, rather than :=, is important below!*).

```
► Clear[f]
  f[x_] = y[x] /. solution[[1]]
```

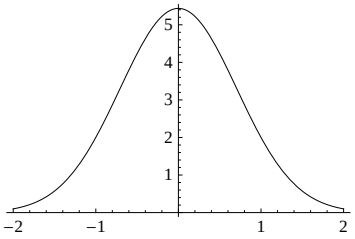
You can see that the function so defined has the correct value at $x = 1$:

```
► f[1]
```

▷ 2

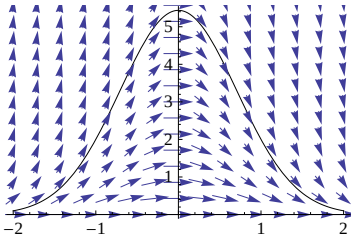
And further, you can see a graph of this solution over the interval $[-2, 2]$:

```
► g1 = Plot[f[x], {x,-2,2}]
```



Finally, you can see that the numerical approximating function f very well matches the slope (vector) field associated with the differential equation.

```
► g2 = VectorPlot[{1,-2x*y}, {x,-2,2}, {y,0,6},
        VectorScale→{0.05, 1, None}];
  Show[g1,g2]
```



9.5 Exercises

1. Use Euler's Method to find approximate, numerical solutions for each of the following equations, using $n = 20$ subintervals. In each case, i) ListPlot the result against a Plot of the known solution, and comment as to whether the result is good or not visually; ii) use *Mathematica* to demonstrate that the indicated actual solution satisfies both the equation and the initial condition; and iii) use DSolve to see whether *Mathematica* can solve the equation and whether it finds the same result as the one given.

(a) $\dfrac{dy}{dx} = \dfrac{\sin 3x - 2xy}{x^2}$ on $\left[\dfrac{\pi}{2}, \pi\right]$, with initial condition $y\left(\dfrac{\pi}{2}\right) = y_0 = \dfrac{4}{3\pi^2}$.

(b) $\dfrac{dy}{dx} = 3y + xe^{3x}$ on $[0, 1]$, with initial condition $y(0) = 4$.

(c) $\dfrac{dy}{dx} = y^2 \cos x$ on $[0, 1]$, with initial condition $y(0) = \dfrac{1}{2}$.

(d) $\dfrac{dy}{dx} = \dfrac{2y - x^3 e^x}{x}$ on $[1, 2]$, with initial condition $y(1) = 0$.

(e) $\dfrac{dy}{dx} = e^{2x} - 3y$ on $[0, 2]$, with initial condition $y(0) = 1$.

(f) $\dfrac{dy}{dx} = \dfrac{4x^3 y}{1 + x^4}$ on $[0, 2]$, with initial condition $y(0) = 1$.

(g) $\dfrac{dy}{dx} = \sin(3x) - 2y$ on $[0, 2]$, with initial condition $y(0) = \dfrac{10}{13}$.

Actual Solutions: a.) $f(x) = \frac{1-\cos 3x}{3x^2}$ b.) $f(x) = \frac{(x^2+8)e^{3x}}{2}$, c.) $f(x) = \frac{1}{2-\sin x}$, d.) $f(x) = -x^2(e^x - e)$, e.) $f(x) = \frac{4+e^{5x}}{5e^{3x}}$, f.) $f(x) = 1 + x^4$, g.) $f(x) = e^{-2x} - \frac{3\cos 3x - 2\sin 3x}{13}$.

2. Consider the differential equation $\frac{dy}{dx} = 30 - 5y$ on the interval $[0, 6]$, subject to the initial condition $y(0) = 1$.

a) Use DSolve to find the solution of this equation and Plot the result.

b) Use Euler's Method for n = 10 subintervals and ListPlot the result against the Plot of the solution. What do you observe (e.g., is the estimated solution good, does it underestimate the actual solution, etc.) ?

c) What course of action should you follow at this point?

3. Consider the differential equation $\frac{dy}{dx} = \frac{3x^2 - 2xy}{x^2 - 2y}$ on the interval $[0, 2]$, subject to the initial condition $y(0) = 2$.

a) Verify that DSolve cannot find an explicit solution to this equation.

b) Use implicit differentiation to show that any function y defined implicitly by the equation $x^3 - x^2 y + y^2 = C$ is a solution to the equation.

c) The curve $x^3 - x^2 y + y^2 = 4$ contains the point $(0, 2)$. Sketch this curve over the interval $[0, 2]$ using ImplicitPlot.

d) Use Euler's Method to construct a numerical solution with n = 40 on $[0, 2]$. Compare the result with the curve produced in part c). Is the result good from a visual standpoint ?

e) Repeat part d), but with initial condition $y(0) = 1$ and compare the result against the curve defined by the equation $x^3 - x^2 y + y^2 = 1$. What abnormalities arise? How do you explain them ?

4. Verify that DSolve cannot solve either of the following equations, and produce a numerical estimate using Euler's Method with n = 50 subintervals on $[-1, 1]$, subject to the initial condition $y(-1) = 0$: a) $\frac{dy}{dx} = (\sin x)(\cos y)$ b) $\frac{dy}{dx} = \sin(xy)$

5. Find numerical error estimates, for values of n = 10, 20, 40, 80 and 160, for the equation of Example 5 in the text (as was done in the error analysis for Example 4). Verify that doubling n decreases the maximum error of the approximation by about a factor of 2.

6. A sky diver jumps out of a plane, and his parachute opens after 2 seconds of free fall, at which time his velocity has already reached 64 feet per second. His velocity y (measured in feet per second) at any time $x \geqslant 2$ (measured in seconds) obeys the differential equation $\frac{dy}{dx} = -32.0 + 0.17y^2$. An initial condition on velocity is, of course, $y(2) = -64$.

a) Use Euler's Method to approximate the velocity function y over the time interval $2 \leqslant x \leqslant 5$, using n = 50 subintervals. What approximate value do you find for $y(5)$?

b) Approximately at what velocity does the parachutist eventually land ? What result or observation from your solution to part a.) helps you predict this ?