Introduction to Maple Versions 9.5 to 15

Information in this document was compiled and updated by Angie Gardiner from:
Multivariate Mathematics with Maple by James A. Carlson and Jennifer M. Johnson, copyright 1996
Introduction to Maple V.4 in the Undergraduate Computer Lab by James A. Carlson and Jennifer M. Johnson, 1993
Math 2250 Maple Tutorial by Nick Korevaar
Math 2250, 2270, 2280 Maple Tutorial edits by Grant Gustafson, 2012

What is Maple?
Maple is a general purpose software package containing tools useful in the study of a wide variety of problems. We can use Maple like a calculator to carry out numerical computation, or we can use the programming features of Maple to produce a more complicated sequence of computations. In addition to numerical computation, we can do symbolic or algebraic computations in Maple. This software package has built-in functions for solving many kinds of equations, for working with matrices and vectors, and for differentiation and integration. It also contains functions which allow us to create visual representations of curves and surfaces from their mathematical descriptions.

Using a software package like Maple gives new ways to think about problems. For example, we can use the computer to run numerical experiments. This means that we can not only look at more realistic versions of standard problems, but we can easily look at many more examples, in the process developing a better intuitive feel for the essential features of a general problem and its solutions. Often these experiments suggest further questions or interpretations that might otherwise go unnoticed. Patterns that appear in these experiments give insight into and motivation for the standard approaches to these problems - and frequently highlight the limitations of these methods.

In many situations the graphics features of Maple are a useful tool for geometrical visualization of ideas, understanding definitions and exploring techniques. Use the graphical interface to develop your intuition while learning new topics.

Before you start learning Maple, it is helpful to familiarize yourself with the facilities in the computer lab in the T. Benny Rushing Mathematics Student Center. Please refer to "Introduction to the Computer Lab" for this information. Campus departments with computer facilities have installed maple under a campus-wide site license. It may be convenient to do **xmaple** sessions in your own department, or 24/7 from home with **maple**, a non-graphical interface running in an xterm window.

The Basics

Starting Maple.

Unix Terminal Instructions. If you are using a UNIX station in our lab, either move the cursor into the local window and type **xmaple &**, use the middle mouse button on the background to get a menu from which you can select **X Maple V12** (or another version), or click on the maple leaf icon at the bottom of the screen. Soon a new window will appear. Click and hold the mouse, then move the window. When satisfied, release the mouse button.

MacLab Instructions. If you are using a Mac in our lab, click on the icon at the upper right corner of your screen labeled **/**. This will open a new window. Click on the **Applications** button in this window, and then select **XDarwin** from the applications and run it **rootless**. After a couple of minutes, you will get a terminal window. In this window, type **ssh sunfire**. Now you can type **xmaple** at the prompt to start Maple. Once Maple opens, you can use the buttons in the upper right corner of each window to resize the windows if you like.

Saving a file. To save a Maple session that you have been working on and give it a name, choose **File -> Save As** from the menus at the top of the screen. A **Save As** window will appear. If you want to save your file under the name **hwk1** (for example), enter **hwk1** in the **Filename** box. If there is already a ***.mws** in the box, make sure that you delete the *****. Then hit the **Return** (Macs) or **Enter** (UNIX) key, or click on **OK**. It is best to keep file names simple, using only letters and numbers, without any special symbols. In particular, it is a good idea to **avoid using spaces or periods**. Maple will add the **.mws** or **.mw** extension to identify the saved file as a **M**aple **work**sheet. Be careful about reusing a file name from a previous session. For example, if last week you made a file named **Lines.mws** and this week you open a new Maple session and call it **Lines.mws** also, this will ERASE the file you created last week. Notice that the **Save As** window lists the Maple files in your current directory, so you can see which names you have already used. Once you have given your file a name, you can do **File -> Save**, or use the save icon at the top of the window.

Crashed Maple Recovery. After a crash of maple, there is a file in the same directory with extension **_MAS.BAK** which contains all recent typing since the last save. Load the file that crashed and also this **_MAS.BAK** file with matching file name. Mouse-copy from the save-file into the crashed file, to get back whatever typing is possible. Be selective, then save the file in case of another crash event.

Copy/Paste. Maple rookies can avoid typing errors by copying known valid code with the mouse (highlight, then **ctrl-C**), followed by paste onto a new line (mouse click at target location, then **ctrl-V**). Copy works from text documents and also PDF documents (copy requires adobe reader). To paste, try the menu item PASTE. Then ctrl-V. Then RIGHT MOUSE BUTTON. Then MIDDLE MOUSE BUTTON. If none of that works, then hold down the SHIFT KEY and try the mouse buttons again.

Mac Workstation. If you are using a Mac, use the mouse to highlight the command(s) you want to copy, then choose **Edit -> Copy** from the menus.
Move the cursor to where you want to paste the text, then choose **Edit -> Paste**.
Unix Workstation. Press and hold down the left mouse button at the beginning of the text you want to copy. Still holding the button down, drag the mouse
until you have highlighted the desired text. Release the mouse button. Move the cursor to the position where you want to paste. Click the left mouse button
to mark this spot. Click the middle mouse button to paste.
Repeat Paste. Copied text stays in memory and anytime you press the middle mouse button (or choose **Paste** from the **Edit** menu) it will be pasted at the
current cursor position. This can cause surprises, if you accidentally hit the middle mouse button. An immediate ctrl-z will **undo** the error. For safety,
mouse-copy some blank space to clear the copied text.

Editing your Maple file. It may take more than one session at the computer to complete your analysis. Save your work on disk, and come back to the lab and open your Maple file again to continue working. Often it will be useful to take with you a printed copy of your Maple session so you can think about what you have done so far, and how you want to continue when you come back to the lab, or so that you can get help from a lab instructor, a friend, or the instructor.

Cleaning Up. You can also delete text or commands by using the mouse to highlight the area you want to delete, then hitting the **Delete** key. To delete an entire computation or a plot, move the cursor into the command block (square left bracket on the left screen edge) that produced the computation or click on the plot. Hold down the **Control** key and press the **Delete** key. The **Edit menu** has a duplicate. Accelerate typing by learning key shortcuts from the menus.

Comments . In any command line, Maple ignores anything typed after the # symbol to the end of that line. An example:

```
> solve ( 3*x + 2*y - 5*z, z);  # Solve for z in terms of symbols
  x and y
```

$$\frac{3}{5}x + \frac{2}{5}y \qquad \textbf{(1)}$$

Block Comments. If you want to add comments between commands, move the cursor into the first command. Then click on the icon **T** at the top of the screen (left of **[>**) . A new text block appears below and the cursor moves into the block. The pull-down menus and buttons at the top of the Maple window allow you to control the size and style of the text you enter. Default text is black at 12 points.

Printing. To print your Maple session directly from Maple, choose **Print** from the **File** menu (or click on the Printer icon). In the Printer Setup window that appears, check that **Print Command** is selected, rather than Output to File, and click **Print**. If the default output is the printer, then to print to a file, choose the box marked **Print To File** on the General tab of the print dialog box. Some maple versions don't have the feature.) The default print command (**lpr**) should send a printout of your Maple worksheet to the printer. If only part of your worksheet is printing, try changing the default print command (inside maple) to **lpr -l** or **lpr -oraw**.

Alternatively, you can save your Maple session to a postscript file, then print that file from a local terminall window. To do this, in the Printer Setup window choose **Output to File** and click **Print**. This will create a postscript file from your Maple worksheet and will store it with some name, like **hwk1.ps**. To send this file to the printer, go to the local window and type **lpr hwk1.ps**, then press Enter. If you are printing from the local window, make sure that you are printing a **.ps** file. The file type **.mws** can only be printed from xmaple. Print PDF files from acroread (adobe reader). Don't use lpr on PDF files, because of printer filters that silently change the print file contents.

* **If you are having trouble printing, please ask the lab assistant for help.** *

Coming back for more. When you come back to work on a Maple file that you've already started, your task now is to reopen that file and get back to where you left off. To open a file, choose **File -> Open**. An **Open File** window will appear. In this window you will see a list of Maple files. Click on the file that you want to open, then press the **Enter** key or click **OK**.

It is not enough to just OPEN the saved file!

Although Maple displays a record of your last session, it unloads variables. One way to get Maple to reload variables from last time is to go to the beginning of your file and hit the **Return** key until you get to the last line of the file. (You might want to use the arrow keys to skip the lines that displayed plots or requests for help files). This will ensure that Maple's memory of your last session is restored. A shortcut for executing the entire worksheet is to **click the icon** **!!!** at the top of the window. Inserting a **#-sign** at the front of plot or help lines will quiet down the worksheet re-execute.

Using Maple

The best way to learn Maple is by using it. But first, a few remarks about Maple. The symbol **>** is the command prompt, which Maple uses to signal you that it awaits your command. Commands normally end with a semicolon (;) if you want to see the output or a colon (:) if you don't want to see the output. Maple is a programming language. It has strict rules of punctuation, grammar, and spelling. If something is not working right, check to see if you are following the rules. For example, the Mapleengine will get confused if you write **2x** instead of **2*x**. Check for things like misspelled names or extra or missing parentheses. If further thought doesn't clear things up, ask a human for help. You will soon become an expert troubleshooter.

While learning Maple, you will often have questions about how a particular command or function is used. Fortunately, Maple can help. To ask about a command whose name you know, just type a question mark, followed by the name of the command. Thus

```
> ?solve
```

gives information on the `solve` command. Scroll forward to the examples at the end. Most questions are answered by the examples. If not answered, then use the technical information at the beginning of the help file. For specific help on how to do something in maple, use a google search in a browser - it is faster than sorting through maple's help system.

The help system in maple supports full text search or topic search of Maple's help files. You will find these options under the **Help** menu in the upper right corner of the Maple window. You will also find other handy things, such as **Introduction**, **New User's Tour**, and **Using Help**. Rookies are frustrated by the help system, but encouraged by google searches, which can answer a specific question.

Hand Calculator Examples

Let's get started! Go ahead and type along with these examples.

```
> 2 + 2; 3*5; 6-2:
```

$$4$$
$$15 \qquad (2)$$

All three computations were done, although only two results are shown (the colon `:` at the end of a command suppresses the output). If you forget the semicolon, go ahead and put it on the next line:

```
> 3*5
> ;
```

$$15 \qquad (3)$$

Addition +, subtraction -, and division / are standard, and parentheses are used as in algebra. An asterisk * indicates multiplication and a caret ^ is used for powers:

```
> ( 1 + 2) * (6 + 7) - 12 / 7;
```

$$\frac{261}{7} \qquad (4)$$

```
> 3^(2.1);
```

$$10.04510857 \qquad (5)$$

Whenever possible, Maple tries to compute exact quantities. Our first command gives its answer as a fraction, rather than as a decimal, contrary to what you might expect. The second command gives a decimal, or "floating point" answer because we used this form in our question. To force Maple to give results in floating point (decimal) form, use `evalf`:

```
> Pi; # The constant 3.1415727... prints as a greek letter
  pi; # Symbol PI, prints as a greek letter (confusing isn't it?)
```

$$\pi$$
$$\pi \qquad (6)$$

```
> evalf(Pi); # Print PI to 10 digits default
```

$$3.141592654 \qquad (7)$$

```
> exp(1);    # the number e=2.818... prints as lowercase italic e
  evalf(%);   # The % sign stands for the most recently computed quantity.
  e;         # symbol e prints as lowercase italic e
  evalf(%);
                                        e
                                   2.718281828
                                        e
                                        e                                    (8)
```

Upper and Lower Case Madness.
Maple code distinguishes upper-case letters from lower-case. Thus `evalf(pi)` is not the same as `evalf(Pi)`.
The function `evalf` can take a second (optional) argument which determines the precision of the output.

```
> evalf(Pi, 50); # Compute Pi to 50 digits.
          3.1415926535897932384626433832795028841971693993751              (9)
```

Spacing.

For the most part, spacing is unimportant in Maple. In the code line above, spaces could be omitted or added without causing any problems.
Thoughtful use of spacing makes Maple code easier to read, easier to understand, and easier to edit.

Standard mathematical functions can be used in Maple so long as we know their names.

$Let's\ compute\ |-14| + \sin(1) - \sqrt{2} + e^{\cos(1.6\,\pi)} + \arctan(3)$

```
> abs(-14) + sin(1) - sqrt(2) + exp( cos(1.6*Pi) ) + arctan(3);
```

$$14 + \sin(1) - \sqrt{2} + \mathrm{e}^{\cos(1.6\,\pi)} + \arctan(3) \quad (10)$$

```
> evalf(%);
                                   16.03838871                             (11)
```

Maple expression syntax can often be found by intelligent guessing. Thus `tan(45)` does indeed compute the tangent, and `20!` computes a factorial.
If your first guess doesn't work, then use Maple help or switch to a browser search engine, looking for sample code.

Packages.

In addition to the functions in the Maple library, there are specialized packages (libraries) of functions that can be loaded using the command `with`.
For example, to do certain work with matrices and vectors requires the linear algebra package. You do this by typing

```
> with (LinearAlgebra); # This package is recommended for
  differential equation and linear algebra courses
```

[*&x*, *Add*, *Adjoint*, *BackwardSubstitute*, *BandMatrix*, *Basis*, *BezoutMatrix*, *BidiagonalForm*, *BilinearForm*, *CARE*, *CharacteristicMatrix*, *CharacteristicPolynomial*, *Column*, *ColumnDimension*, *ColumnOperation*, *ColumnSpace*, *CompanionMatrix*, (12) *ConditionNumber, ConstantMatrix, ConstantVector, Copy, CreatePermutation, CrossProduct, DARE, DeleteColumn, DeleteRow, Determinant, Diagonal, DiagonalMatrix, Dimension, Dimensions, DotProduct, EigenConditionNumbers, Eigenvalues, Eigenvectors, Equal, ForwardSubstitute, FrobeniusForm, GaussianElimination, GenerateEquations, GenerateMatrix, Generic, GetResultDataType, GetResultShape, GivensRotationMatrix, GramSchmidt, HankelMatrix, HermiteForm, HermitianTranspose, HessenbergForm, HilbertMatrix, HouseholderMatrix, IdentityMatrix, IntersectionBasis, IsDefinite, IsOrthogonal, IsSimilar, IsUnitary, JordanBlockMatrix, JordanForm, KroneckerProduct, LA_Main, LUDecomposition, LeastSquares, LinearSolve, LyapunovSolve, Map, Map2, MatrixAdd, MatrixExponential, MatrixFunction, MatrixInverse, MatrixMatrixMultiply, MatrixNorm, MatrixPower, MatrixScalarMultiply, MatrixVectorMultiply, MinimalPolynomial, Minor, Modular, Multiply, NoUserValue, Norm, Normalize, NullSpace, OuterProductMatrix, Permanent, Pivot, PopovForm, QRDecomposition, RandomMatrix, RandomVector, Rank, RationalCanonicalForm, ReducedRowEchelonForm, Row, RowDimension, RowOperation, RowSpace, ScalarMatrix, ScalarMultiply, ScalarVector, SchurForm, SingularValues, SmithForm, StronglyConnectedBlocks, SubMatrix, SubVector, SumBasis, SylvesterMatrix, SylvesterSolve, ToeplitzMatrix, Trace, Transpose, TridiagonalForm, UnitVector, VandermondeMatrix, VectorAdd, VectorAngle, VectorMatrixMultiply, VectorNorm, VectorScalarMultiply, ZeroMatrix, ZeroVector, Zip*]

This command produces a list of all the functions in this package and gives you access to them in your current Maple session.
If you close Maple and reopen it later, you must reload any special packages you want to use.
Once familiar with a package, load it silently using a colon. (:) instead of a semicolon (;).

```
> with (linalg): # Silence output. Old package, support being
  removed by maplesoft.
```

Other Packages. Some packages of interest are `plots`, `DEtools`, and `student.`

Algebra. Maple code uses variables and algebra. Consider, for example, the expression $(a+b)^2$ with variables a,b.

```
> (a + b)^2;
```

$$(a+b)^2 \quad \textbf{(13)}$$

```
> expand (%);
```

$$a^2+2\,a\,b+b^2 \quad \textbf{(14)}$$

give the expanded form, and

```
> factor (%);
```

$$(a+b)^2 \quad \textbf{(15)}$$

brings us back to our starting point. To make long computations easier and more intelligible, we can assign values to variables using "`:=`"

```
> p := (a + b)^2; b := 1; p;
```

$$p := (a+b)^2$$
$$b := 1$$

$$(a+1)^2 \quad (16)$$

In these examples, variables store an expression or a number. Variables can store almost anything, for example, a list of points, an equation, a set, a piece of text, or a function definition:

```
> pts := [ [1,2], [3,4] ]; # a double-list or list-of-lists
```

$$pts := [[1,2],[3,4]] \quad (17)$$

```
> eqn := 2*x - 3*y = 5;     # eqn abbreviates equation 2x+3y=5
```

$$eqn := 2x - 3y = 5 \quad (18)$$

```
> eqns := { 2*x - 3*y = 5, 5*x - 3*y = 1 }; # A set of two
  equations
```

$$eqns := \{2x - 3y = 5, 5x - 3y = 1\} \quad (19)$$

```
> tag := "The nth partial sum is";  # strings are delimited by
  double quotes
```

$$tag := \text{"The nth partial sum is"} \quad (20)$$

```
> print (pts, eqn, eqns, tag);    #checks our assignments
```

$$[[1,2],[3,4]], 2x-3y=5, \{2x-3y=5, 5x-3y=1\}, \text{"The nth partial sum is"} \quad (21)$$

```
> f := x -> x^2;           # defines a function. Use 2 symbols,
  MINUS and GREATER-THAN
```

$$f := x \rightarrow x^2 \quad (22)$$

```
> g:=unapply(x^2,x);       # defines a function, with recusive
  symbol evaluation
```

$$g := x \rightarrow x^2 \quad (23)$$

```
> f(2);  f(3); g(2); g(3);
```

```
4
9
4
9
```
(24)

Assignment. Anything we can define or compute in Maple can be assigned to a variable for future reference using "`:=`". The symbol `=` by itself is used to test equality.
A space is NOT allowed between the `:` and the `=` in an assignment statement. Beware of using equal only when you meant colon-equal. Such typos are maddening to discover
and the typo generates no maple error message.

Variables can be returned to their original symbolic (unassigned) state. The commands

```
> b := 'b'; # same as unassign('b');  Removes b:=1; assignment
  made above to symbol b.
```

$$b := b \quad (25)$$

```
> p;        # re-execute formula for p, with b:=1 replaced by
  symbol b
```

$$(a+b)^2 \quad (26)$$

Similarly, clears the variables assigned above using unassign():

```
> unassign ( 'pts', 'eqn', 'eqns', 'tag');
> print (pts, eqn, eqns, tag);
```

$$pts, eqn, eqns, tag \quad (27)$$

The restart command clears ALL variables and unloads all packages. So, if you need one later, you must reload it using `with`.

```
> restart; # A drastic way to clear variables and computer memory
  # Also got rid of library loads for linalg and LinearAlgebra
  which were done above.
```

Pay special attention to the kind of quotes used in examples. The possibilities are the single quote `'`, the backquote `` ` ``, and the double quote `"`. They all play different roles.

Here is an extended example of how to use variables and assignment statements:

```
> F := m*a;                # Newton's formula for force
```

$$F := m\,a \tag{28}$$

```
> m := 2.1;  a := 5;       # set the mass and acceleration
```

$$m := 2.1$$
$$a := 5 \tag{29}$$

```
> F;                       # compute the force
```

$$10.5 \tag{30}$$

```
> a := 21.9;               # reset the acceleration
```

$$a := 21.9 \tag{31}$$

```
> F;                       # recompute force
```

$$45.99 \tag{32}$$

```
> a := 'a';                # clear a with single quotes
```

$$a := a \tag{33}$$

```
> s:="a";                  # make a one-character string, no
  substitution of symbol a
  s:="m";                  # one-character string, again no
  substitution of symbol m (m equals 2.1)
```

$$s := \text{"a"}$$
$$s := \text{"m"} \tag{34}$$

```
> F;                       # recompute F, symbol a was restored
```

$$2.1\,a \tag{35}$$

The **subs** command lets us make temporary substitutions in an expression as opposed to assigning values. For example, try this:

```
> g := (a+1)^2 / (b-1)^3 + a / (b-1);
```

$$g := \frac{(a+1)^2}{(b-1)^3} + \frac{a}{b-1} \tag{36}$$

```
> simplify (g);
```

$$\frac{a^2 + 3\,a + 1 + a\,b^2 - 2\,a\,b}{(b-1)^3} \tag{37}$$

```
> subs( a=3, b=2, g);
```

$$19 \tag{38}$$

or this

```
> subs( a = x+y, b = x+1, g); # x,y can be symbols or := assigned
```

```
values or constants
```

$$\frac{(x+y+1)^2}{x^3} + \frac{x+y}{x} \qquad (39)$$

```
> simplify(%); a; b;          # The symbol % means the answer of
  the previous computation
```

# The variables **a** and **b** were not permanently assigned a value.

$$\frac{x^2+2\,x\,y+2\,x+y^2+2\,y+1+x^3+x^2\,y}{x^3}$$

$$a$$

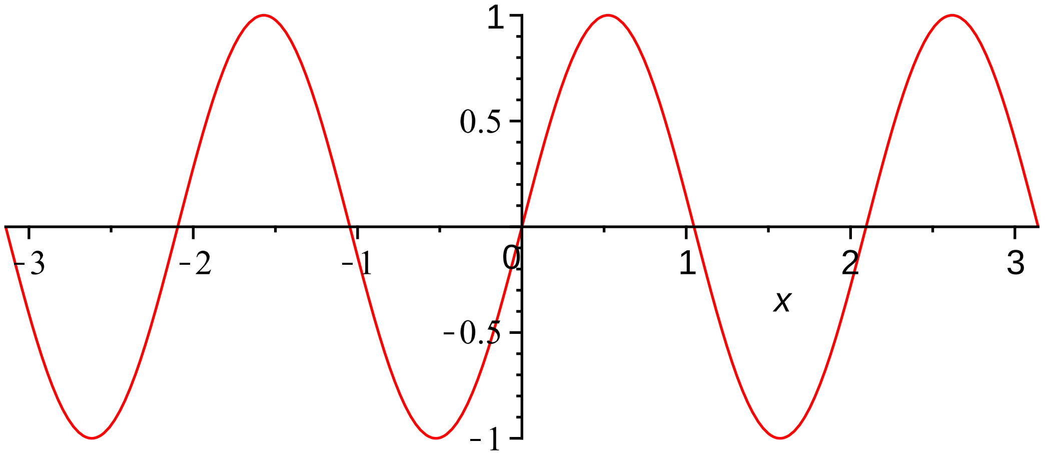
$$b \qquad (40)$$

Graphing

Standard Coordinates. Maple can construct many kinds of graphs, a feature that you can use to visualize mathematical objects and processes.

The command

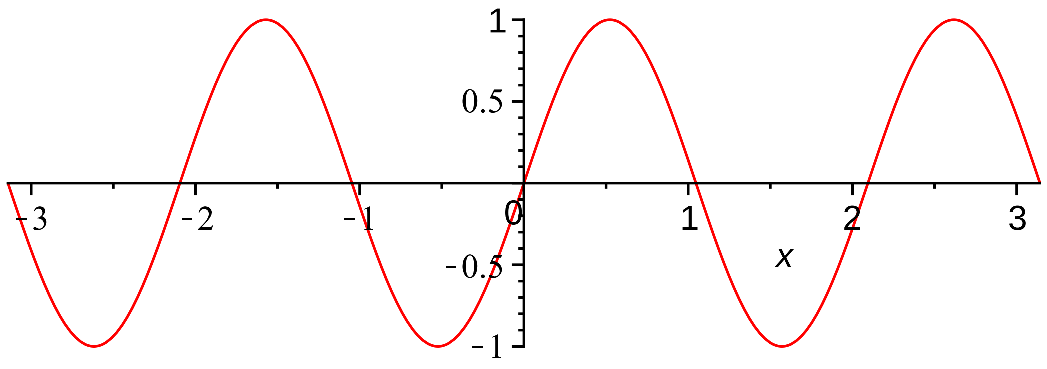
```
> plot( sin(3*x), x = -Pi..Pi );
```



produces a plot containing the curve $y = \sin(3\,x)$ for x in the interval from $-\pi$ to π. No space is allowed between the double-dots in a plot command.

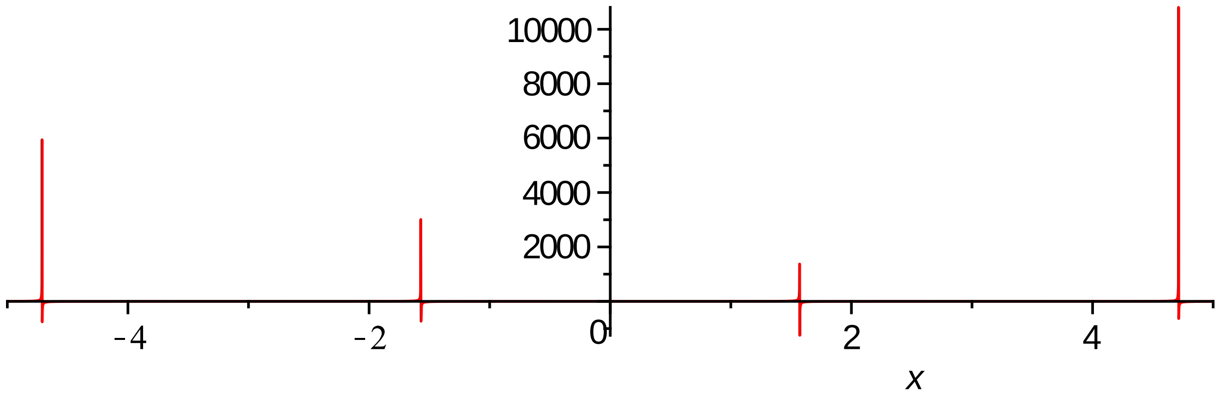
Scales on the x-axis and y-axis are chosen in the maple engine. To change this behavior:

```
> plot( sin(3*x), x = -Pi..Pi, scaling = constrained );
```



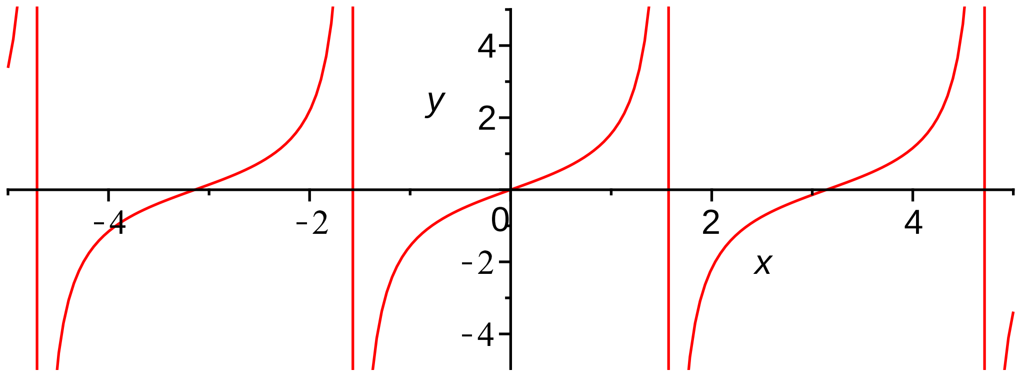
Sometimes it is useful to restrict the range over which y varies. We get a misleading graph from

```
> plot( tan(x), x = -5..5 );
```



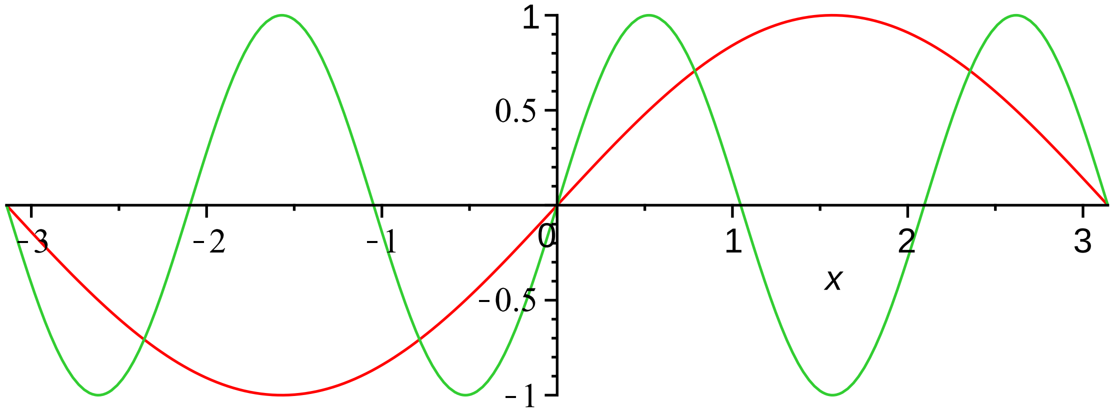
It does not accurately represent the vertical asymptotes of $y = \tan(x)$. Better results are obtained with

```
> plot( tan(x), x = -5..5, y = -5..5 );
```

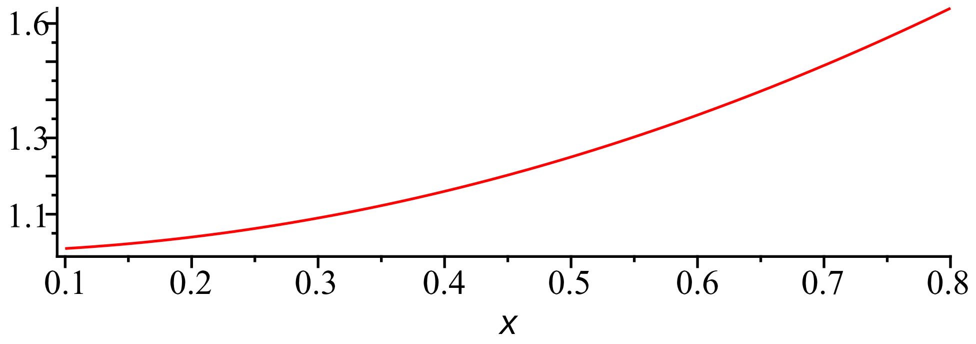


We can plot several curves at once. To plot $y = \sin(x)$ and $y = \sin(3\,x)$ together we say

```
> plot( { sin(x), sin(3*x) }, x = -Pi..Pi ); # The curve
  equations are inside set-delimiters of curly braces
```



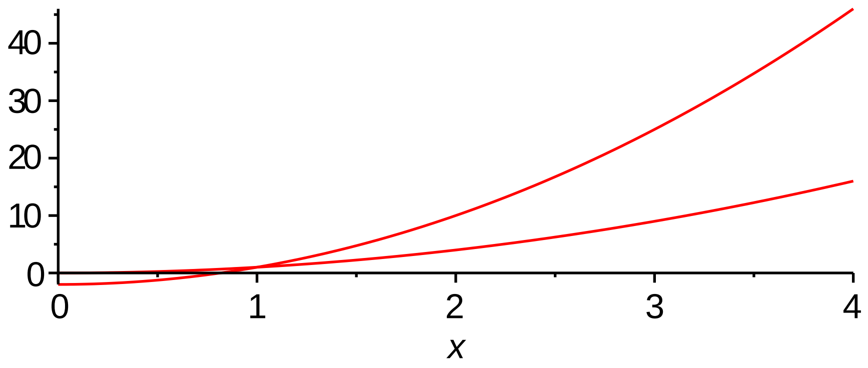
```
> plot(1+x^2,x=0.1 .. 0.8); # Decimal points can conflict with
  double-dot range parsing [compare x=.1...8]
```



Now the first argument of **plot** is a set of expressions to be graphed. Sets are enclosed in curly braces and individual items are separated by commas.

We could also use the **display** command found in the **plots** package:

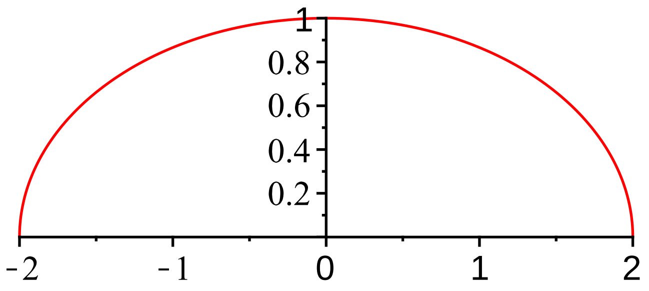
```
> with(plots):
> plot1 := plot( x^2, x = 0..4 ): # Don't display plot data, use
  colon
> plot2 := plot( 3*x^2 - 2, x = 0..4 ):
> display( [ plot1, plot2 ] );
```



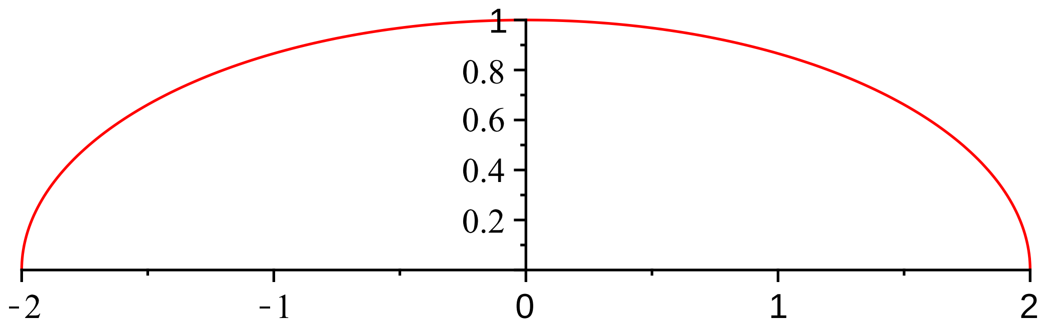
Parametric Equations

A curve in the plane can be described as the graph of a function, as in the graph of $y = \frac{1\sqrt{4-x^2}}{2}$ for x in the interval from -1 to 1.
It can be given parametrically as in $(x(t), y(t)) = (2\cos(t), \sin(t))$ for t in the interval from 0 to π.
Often we interpret such a curve as the path traced by a moving particle at time t.
Use the `plot` command to draw a curve from this parametric description:

```
> plot( [2*cos(t), sin(t), t = 0..Pi] ); # unconstrained,
  distorted graphic
```



```
> plot( [2*cos(t), sin(t), t = 0..Pi], scaling = constrained);  #
  constrained is not distorted
```



Polar Coordinates

Polar plots are a special kind of parametric plot. The polar coordinates (r, θ) of points on a curve can be given as a function of some parameter t. In many cases the parameter is just the angle θ. Consider the ellipse defined in standard coordinates by $x^2 + 4y^2 = 4$. To find an equation relating the polar coordinates r and θ of a typical point on this ellipse, we make the substitution $x = r\cos(t)$ and $y = r\sin(t)$, where $t = \theta$.

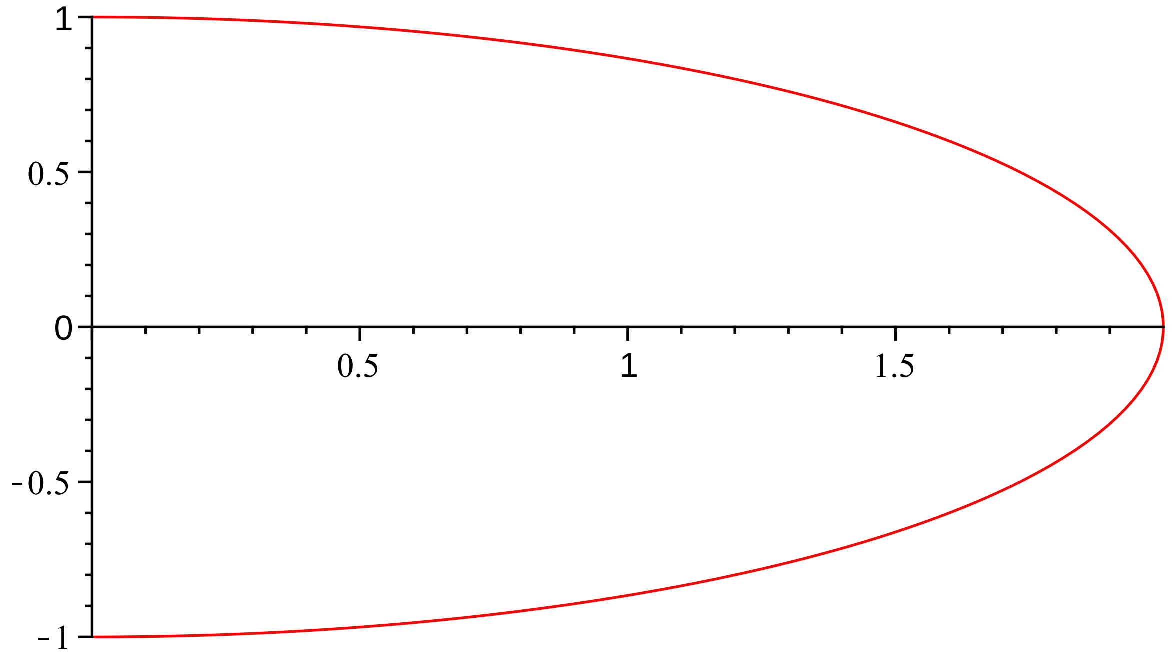
```
> subs( x = r*cos(t), y = r*sin(t), x^2 + 4*y^2 = 4);
  simplify( % );
  solve( %, r );
```

$$r^2\cos(t)^2 + 4r^2\sin(t)^2 = 4$$
$$-r^2\left(3\cos(t)^2 - 4\right) = 4$$

$$-\frac{2}{\sqrt{-3\cos(t)^2+4}}, \frac{2}{\sqrt{-3\cos(t)^2+4}} \quad \textbf{(41)}$$

We find that the ellipse is the collection of points whose polar coordinates (r, θ) satisfy $r^2 = \frac{4}{4-3(\cos(\theta))^2}$, and the following commands draw the right half of the ellipse:

```
> r := 2/sqrt( 4 - 3*cos(t)^2 ):
> plot( [ r, t, t = -Pi/2..Pi/2 ], coords = polar );
```



Here are some more of examples of polar plots that you can try:

```
> plot( [ 1, t, t = 0..2*Pi], coords=polar ):
> plot( [t, t, t = 0..2*Pi], coords=polar ):
> plot( [ sin(4*t), t, t = 0..2*Pi], coords=polar ): #
```
a more realistic picture is obtained with `scaling = constrained`.

Plotting Data
Maple can plot data consisting of pairs of x and y values. For example, if we say

```
> data := [ [0, 0.53], [1, 1.14], [2, 1.84], [3, 4.12] ];
```

$$data := [[0, 0.53], [1, 1.14], [2, 1.84], [3, 4.12]] \quad \textbf{(42)}$$

then `data` refers to a sequence of five points, (x, y) = (0, 0.53), etc. The result is a double list: something enclosed in square brackets, with comma-separatd elements in square brackets. Lists are used for collections of objects where the order matters.

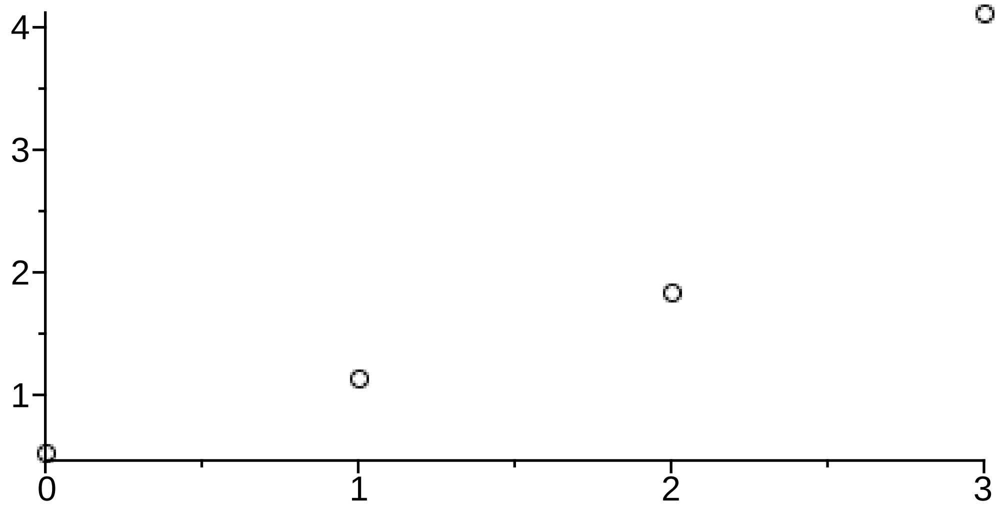
```
> data[1]; data[2]; data[3]; data[4]; # Individual items are
  accessed this way
```

$$[0, 0.53]$$
$$[1, 1.14]$$

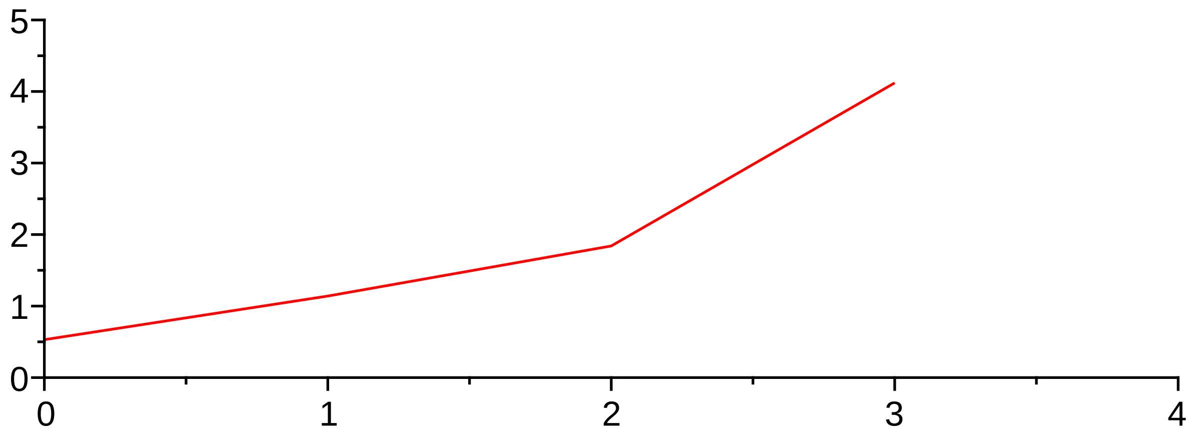
```
[2, 1.84]
[3, 4.12]                                                    (43)
> data[3][2]; # To access the second coordinate of the third data
  point
                               1.84                          (44)
```

To plot the points in a double-list we use commands like

```
> plot( data, style = point, symbol = circle, color = black);
```



```
> plot( data, style = line, view = [0..4, 0..5] );
```



```
> plot( data, style = line, title = "Experiment 1" ):
```

Here the double quote is used to specify a plot title string. See `?plot[options]` for more information, e.g., about symbols and line styles available.
You could also try `?readdata` and `?stats` to find out how to read a file of data points into a Maple session.

Solving Equations

Consider some examples:

```
> solve( x^2 + 3*x = 2.1 );
```

$$0.5856653615, -3.585665361 \quad (45)$$

```
> solve( x^3 + x = 27 ):          #This gives a complicated
  answer!
> solve( x^3 + x = 27.0 );
```

$$2.88894157222823, -1.44447078611412 + 2.69434362244525\,I, -1.44447078611412 - 2.69434362244525\,I \quad (46)$$

The second command gives an exact, but complicated, answer. Replacing 27 by 27.0 forces Maple to give decimal approximations instead, as do the commands

```
> fsolve( x^3 + x = 27 );
```

$$2.888941572 \quad (47)$$

```
> fsolve( x^3 + x=27, x, complex);
```

$$-1.44447078611412 - 2.69434362244525\,I, -1.44447078611412 + 2.69434362244525\,I, 2.88894157222823 \quad (48)$$

In general, `solve` looks for exact answers using algebraic methods, whereas `fsolve` uses numberical methods to find approximate solutions in floating-point form. Compare

```
> solve( tan(x) - x = 2 );
```

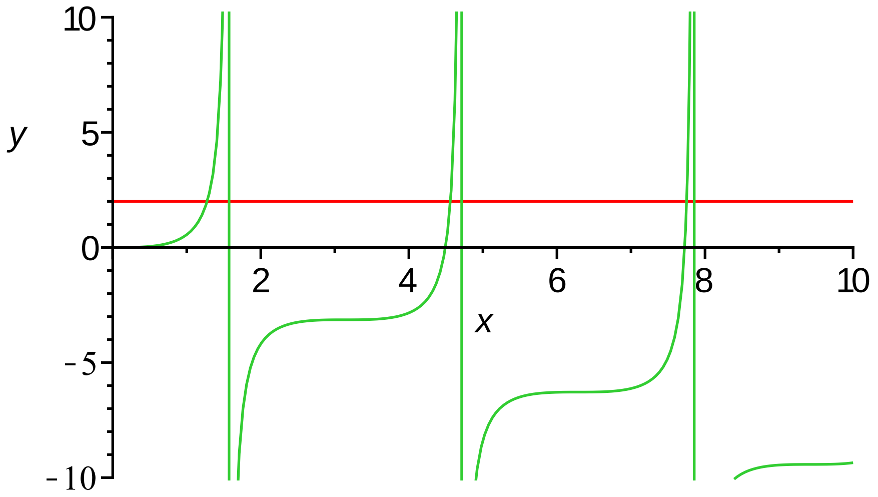
$$\mathit{RootOf}(-\tan(\_Z) + \_Z + 2) \quad (49)$$

```
> fsolve( tan(x) - x = 2 );
```

$$1.274392662 \quad (50)$$

Maple responds with an echo of the failed command, or a blank line, if it cannot find the solution you asked for.
Command `fsolve` may not find all solutions. To understand why not, it is helpful to look at a graph

```
> plot( { tan(x) - x, 2 }, x = 0..10, y = -10..10 );
```



This will give you an idea of how many solutions there are and what their approxmiate location is. Then give **fsolve** a range of x-values in which to search:

```
> fsolve( tan(x) - x = 2, x = 4..5 );
                              4.561140431                                        (51)
```

Often we need to use the solution of an equation in a later problem. To do this, assign a name to it. Here is one example.

```
> r := solve( x^2 + 3*x -2.1 = 0 );
                        r := 0.5856653615, -3.585665361                          (52)
```

The answer has the form r := $r1$, $r2$, where $r1$ is the first root and $r2$ is the second. Such an object - a bunch of items separated by commas, is called an expression sequence. One picks out items of an expression sequence this way:

```
> r[1];
                              0.5856653615                                       (53)
> r[2];
                              -3.585665361                                       (54)
```

Here are some computations with items from an expression sequence:

```
> r[1] + r[2];                 # sum of the roots
                              -3.000000000                                       (55)
> r[1]*r[2];                   # their product
                              -2.100000000                                       (56)
> subs( x = r[1], 2*x + 3 );   # find 2(first solution) + 3
                              4.171330723                                        (57)
```

We can also solve systems of equations:

```
> solve( { 2*x + 3*y = 1, 5*x + 7*y = 2 },[x,y] );
                            [[x = -1, y = 1]]                                    (58)
> x; y; # Surprise: symbols x,y unassigned by solve()
                                   x
                                   y                                             (59)
```

A system of equations is given as a *set* - a bunch of items enclosed in curly brackets and separated by commas. Sets are often used when the order of the objects is unimportant. In reply to the **solve** command above, Maple tells us how to choose x and y to solve the system, but it does not give x and y these particular values. To force it to assign these values, we use the **assign** function:

```
> s := solve( { 2*x + 3*y = 1, 5*x + 7*y = 2 }, [x,y] );
  assign( s );
  x; y;       #check that it worked
                          s := [[x = -1, y = 1]]
                                   -1
                                   1                                             (60)
```

Important note: To type multiple line commands, as above, use Shift-Return instead of Return.
And a warning: You may have trouble later if you leave numerical values assigned to the variables x, y,

and r. Maple will not forget these assigned values, even though you have gone on to a new problem where x means something different. It is a good idea to return variables to their unassigned state when you finish your problem.

```
> x := 'x'; y := 'y'; r := 'r'; # or, unassign('x','y','r'):
```

$$x := x$$
$$y := y$$
$$r := r \tag{61}$$

Recall that `restart` also clears all variables.

Finally, symbolic parameters are allowed in `solve` commands. However, in that case we have to tell Maple which ones to solve for and which ones to treat as unspecified constants:

```
> solve( a*x^2 + b*x + c, x );        # solve ax^2+bx+c=0 for x
```

$$\frac{1}{2}\,\frac{-b+\sqrt{b^2-4\,a\,c}}{a},\ -\frac{1}{2}\,\frac{b+\sqrt{b^2-4\,a\,c}}{a} \tag{62}$$

```
> solve( a*x^3 + b*x^2 + c*x +d, x):
```

```
> solve( { a*x + b*y = h, c*x + d*y = k }, [ x,y ] ):     #
  solving a system for x and y. Wrong answer for ad-bc=0. Maple
  engine
                                                          # makes
  undisclosed assumptions. Do ?solve to find out why.
```

Functions

Although Maple has a large library of standard functions, we often need to define new ones. For example, to define

$$p(x) = 18\,x^4 + 69\,x^3 - 40\,x^2 - 124\,x - 48$$

we say

```
> p := x -> 18*x^4 + 69*x^3 - 40*x^2 - 124*x - 48;
```

$$p := x \rightarrow 18\,x^4 + 69\,x^3 - 40\,x^2 - 124\,x - 48 \tag{63}$$

Think of the symbol `->` as an arrow: it tells what to do with the input x, namely, produce the output `18*x^4 + 69*x^3 - 40*x^2 -124*x - 48`.
To understand the historical origins of this notation, read about **lambda functions**, a subject of interest to computer science areas.
Once the function `p` is defined, we can do the usual computations with it, e.g.,

```
> p(-2);
```

$$-224 \tag{64}$$

```
> p( 1/2 );
```

$$-\frac{441}{4} \tag{65}$$

```
> p( a+b );
```

$$18\,(a+b)^4 + 69\,(a+b)^3 - 40\,(a+b)^2 - 124\,a - 124\,b - 48 \tag{66}$$

```
> simplify(%);
```

$$18 a^4 + 72 a^3 b + 108 a^2 b^2 + 72 a b^3 + 18 b^4 + 69 a^3 + 207 a^2 b + 207 a b^2 + 69 b^3 - 40 a^2 - 80 a b - 40 b^2 - 124 a - 124 b - 48 \quad (67)$$

It is important to keep in mind that functions and expressions are different kinds of mathematical objects. Mathematicians know this, and so does Maple. Compare the results of the following:

```
> p;     # function
```

$$p \quad (68)$$

```
> p(x); # expression
```

$$18 x^4 + 69 x^3 - 40 x^2 - 124 x - 48 \quad (69)$$

```
> p(y); # expression
```

$$18 y^4 + 69 y^3 - 40 y^2 - 124 y - 48 \quad (70)$$

```
> p(3); # expression
```

$$2541 \quad (71)$$

As further proof, try the following:

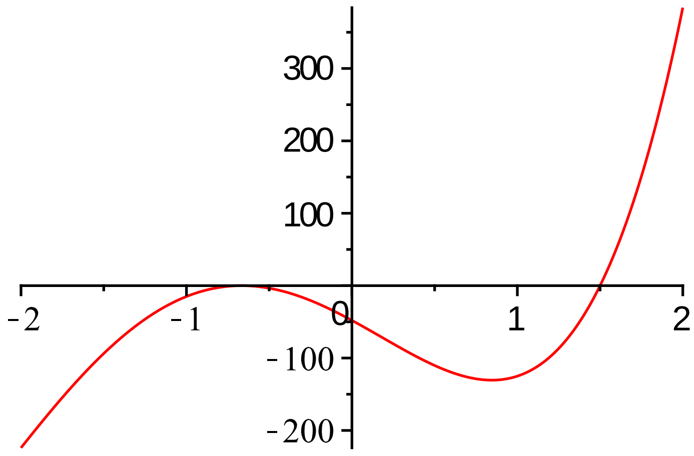
```
> factor(p);
```

$$p \quad (72)$$

```
> factor( p(x) );
```

$$(x + 4)\,(2 x - 3)\,(2 + 3 x)^2 \quad (73)$$

```
> plot( p, -2..2 );
```



```
> # plot( p(x), x = -2..2 ); # same result
```

Functions of several variables can be defined as easily as can functions of a single variable:

```
> f := (x,y) -> exp(-x) * sin(y);
```

$$f := (x, y) \rightarrow e^{-x} \sin(y) \quad (74)$$

```
> f(1,2);
```

$$e^{-1} \sin(2) \quad (75)$$

```
> g := (x,y) -> alpha*exp(-k*x)*sin(w*y);
```

$$g := (x, y) \rightarrow \alpha\, e^{-k x} \sin(w\, y) \quad (76)$$

```
> g(1,2);
```

$$\alpha\, e^{-k} \sin(2\, w) \quad (77)$$

```
> alpha := 2; k := 3; g(1,2);
```

$$\alpha := 2$$
$$k := 3$$
$$2\,e^{-3}\sin(2\,w) \qquad (78)$$

```
> w := 3.5; g(1,2);
```

$$w := 3.5$$
$$1.313973197\,e^{-3} \qquad (79)$$

```
> alpha := 'alpha'; g(1,2);
```

$$\alpha := \alpha$$
$$0.6569865987\,\alpha\,e^{-3} \qquad (80)$$

Calculus

Derivatives
To compute the derivative of the expression $x^3 - 2\,x + 9$, say

```
> diff( x^3 - 2*x + 9, x);
```

$$3\,x^2 - 2 \qquad (81)$$

To compute the second derivative, say

```
> diff( x^3 - 2*x + 9, x, x );
```

$$6\,x \qquad (82)$$

or alternatively,

```
> diff( x^3 - 2*x + 9, x$2 );
```

$$6\,x \qquad (83)$$

This works because Maple translates the expression `x$2` into the sequence `x, x`. By analogy, `x$3` would give the third derivative. Thus one can easily compute derivatives of any order. Now suppose that we are given a function g defined by

```
> g := x -> x^3 - 2*x + 9;
```

$$g := x \to x^3 - 2\,x + 9 \qquad (84)$$

It seems natural to use

```
> diff( g, x );
```

$$0 \qquad (85)$$

to get the derivative of g. However, Maple expects an expression in x and so interprets g as a constant, giving the wrong result. The command

```
> diff( g(x), x );
```

$$3\,x^2 - 2 \qquad (86)$$

which uses the expression $g(x)$, works correctly. The subtlety here is an important one: `diff` operates on expressions, not on functions - g is a function while $g(x)$ is an expression. To define the derivative of a function, use Maple's `D` operator:

```
> dg := D(g);
```

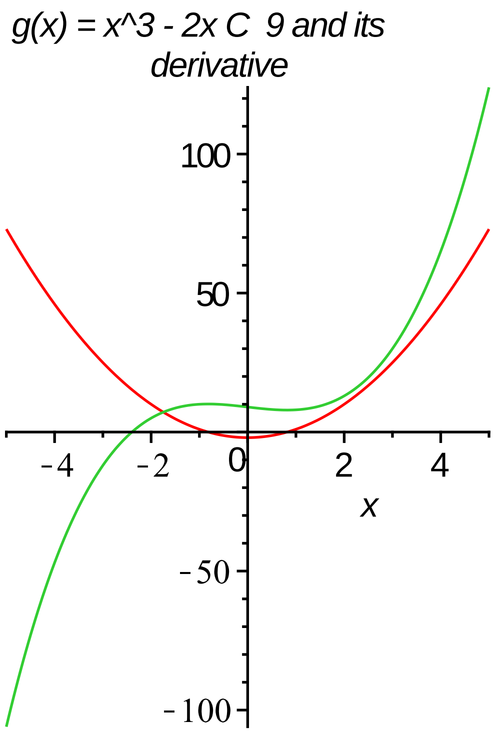
$$dg := x \to 3\,x^2 - 2 \qquad (87)$$

The result is a function. You can work with it just as you worked with g. Thus you can compute function values and make plots:

```
> dg(1);
```

$$1 \qquad (88)$$

```
> plot( { g(x), dg(x) }, x = -5..5, title = `g(x) = x^3 - 2x + 9
  and its derivative` );
```



Partial Derivatives

To compute partial derivatives:

```
> q := sin(x*y);      # expression with two variables
```

$$q := \sin(x\,y) \qquad (89)$$

```
> diff( q, x );       # partial with respect to x
```

$$\cos(x\,y)\,y \qquad (90)$$

```
> diff( q, x, y );    # compute d/dy of dq/dx
```

$$-\sin(x\,y)\,x\,y + \cos(x\,y) \qquad (91)$$

As in the one variable case, there is an operator for computing derivatives of functions (as opposed to expressions):

```
> k := (x,y) -> cos(x) + sin(y);
```

$$k := (x, y) \rightarrow \cos(x) + \sin(y) \qquad (92)$$

```
> D[1](k);         # partial with respect to x
```

$$(x, y) \rightarrow -\sin(x) \qquad (93)$$

```
> D[2](k);         # partial with repsect to y
```

$$(x, y) \rightarrow \cos(y) \qquad (94)$$

```
> D[1,1](k);      # second partial with respect to x
```
$$(x, y) \rightarrow -\cos(x) \qquad (95)$$

```
> D[1,2](k);      # partial with respect to y, then x
```
$$0 \qquad (96)$$

```
> D[1](D[2](k));  # same as above;
```
$$0 \qquad (97)$$

Integrals

To compute integrals, use `int`. The indefinite integral (antiderivative) $\int x^3 \, dx$ is given by `int( x^3, x)`.
The following examples illustrate maple engine integration by parts, substitution, and partial fractions:

```
> int( 1/x, x );
```
$$\ln(x) \qquad (98)$$

```
> int( x*sin(x), x );
```
$$\sin(x) - x\cos(x) \qquad (99)$$

```
> int( sin(3*x + 1), x );
```
$$-\frac{1}{3}\cos(3x+1) \qquad (100)$$

```
> int( x/ (x^2 - 5*x + 4), x );
```
$$-\frac{1}{3}\ln(x-1) + \frac{4}{3}\ln(x-4) \qquad (101)$$

Nonetheless, Maple can't do everything:

```
> int( sin( sqrt(1-x^3)), x );
```
$$\int \sin\left(\sqrt{1-x^3}\right) dx \qquad (102)$$

The last response echoed back the indefinite integral, a signal that the Maple engine failed to find an antiderivative for $\sin\left(\sqrt{1-x^3}\right)$. In fact, it can be proved that no such antiderivative exists.
There are cases when the engine fails but there is an antiderivative. And cases when the engine gets the wrong answer.
The lesson: don't trust maple; use maple to assist solving a problem, do not use maple as an authority.

To compute definite integrals like $\int_0^1 x^3 \, dx$ we say `int( x^3, x = 0..1 )`. Note that the only difference is that we give an interval of integration.

Numerical Integration.

Let us return to the integral $\int \sin\left(\sqrt{1-x^3}\right) dx$, which Maple did not evaluate symbolically. Maple finds a numerical value for the definite integral by this syntax:

```
> int( sin( sqrt(1 - x^3) ), x = 0..1 );
```
$$\int_0^1 \sin\left(\sqrt{1-x^3}\right) dx \qquad (103)$$

```
> evalf(%);      # Force numerical evaluation. Works only if
  there are no undefined symbols.
```
0.7315380065 **(104)**

```
> evalf( Int( sin( sqrt(1 - x^3) ), x = 0..1 ) ); # The inert
  integral Int(...) skips symbolic evaluation. Speed advantage.
```
0.7315380065 **(105)**

Another approach to numerical integration is to use the `student` package.

```
> with( student ):            # load the package
```

```
> j := sin( sqrt(1 - x^3) );  # define the integrand
```

$$j := \sin\left(\sqrt{1-x^3}\right) \qquad \textbf{(106)}$$

```
> trapezoid( j, x = 0..1 );   # apply trapezoid rule
```

$$\frac{1}{8}\sin(1) + \frac{1}{4}\sum_{i=1}^{3}\sin\left(\sqrt{1-\frac{1}{64}i^3}\right) \qquad \textbf{(107)}$$

```
> evalf(%);                   # put in decimal form
```

0.6879899163 **(108)**

By default the `trapezoid` command approximates the area under the graph of $\sin\left(\sqrt{1-x^3}\right)$ with four trapezoidal panels.
For greater accuracy use more panels, i.e., a finer subdivision of the interval of integration:

```
> panels:=10: evalf( trapezoid( j, x = 0..1, panels) );
```
0.7202948615 **(109)**

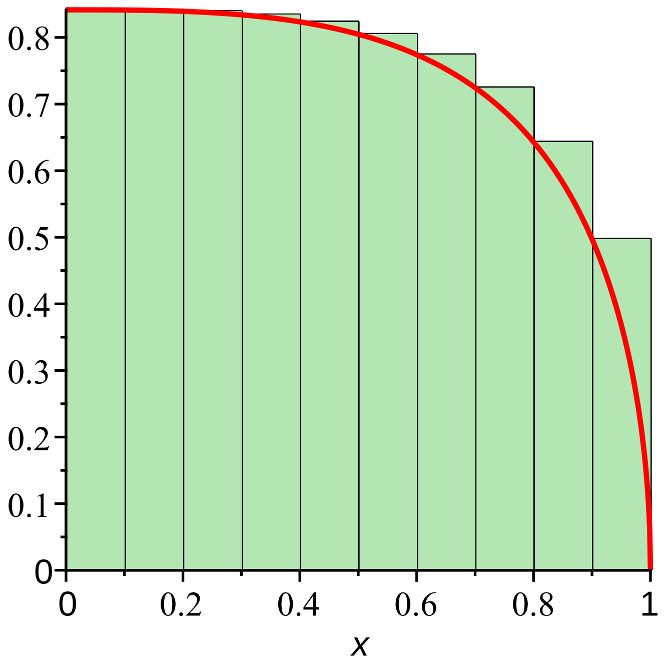
Better yet, use a more sophisticated numerical method like Simpson's rule:

```
> evalf( simpson( j, x = 0..1 ));
```
0.7130564636 **(110)**

Only an even number of subdivisions is allowed, as in `simpson( j, x = 0..1, 10 )`.

The `student` package is well worth exploring. Among other things it has tools for displaying figures which explain the meaning of integration:

```
> leftbox( j, x = 0..1, 10 );
```



The area of the figure displayed by **`leftbox`** is computed by **`leftsum`**:

```
> evalf( leftsum( j, x = 0..1, 10 ));
```

0.7623684108 **(111)**

One can also experiment with **`rightbox`** and **`middlebox`** and their companion functions **`rightsum`** and **`middlesum`**.

Multiple Integrals

Let R be the rectangular region defined by x between 0 and 1, and y between 0 and 1.

To integrate $\int\int (x^2 + y^2)\, dx\, dy$ over R in Maple, we compute the repeated integral. Here is one way to do this:

```
> int( x^2 + y^2, x = 0..1 );
```

$$\frac{1}{3} + y^2 \quad \textbf{(112)}$$

```
> int( %, y = 0..1);
```

$$\frac{2}{3} \quad \textbf{(113)}$$

The first command integrates with respect to the x variable, producing an expression in y.
The second command integrates the result with respect to y to give a number.

Other Calculus Tools

Limits:

```
> g := x -> (x^3 - 2*x + 9)/(2*x^3 + x - 3);
```

$$g := x \rightarrow \frac{x^3 - 2x + 9}{2x^3 + x - 3} \quad \textbf{(114)}$$

```
> limit( g(x), x = infinity );
```

$$\frac{1}{2} \qquad (115)$$

```
> limit( sin(x)/x, x = infinity );
```

$$0 \qquad (116)$$

```
> limit( sin(x)/x, x = 0 );
```

$$1 \qquad (117)$$

Taylor expansions and sums:

```
> taylor( exp(x), x = 0, 4 ); # expansion around x = 0
```

$$1 + x + \frac{1}{2} x^2 + \frac{1}{6} x^3 + \mathrm{O}(x^4) \qquad (118)$$

```
> sum( i^2, i = 1..100 );     # be sure i is unassigned
```

$$338350 \qquad (119)$$

```
> sum( x^n, n = 5..10 );      # needs n to be unassigned
```

$$x^5 + x^6 + x^7 + x^8 + x^9 + x^{10} \qquad (120)$$

```
> sum( 1/d^5, d=1..infinity );  evalf(%);
```

$$\zeta(5)$$
$$1.036927755 \qquad (121)$$

Differential Equations

Maple can solve differential equations for you. One way is to use **dsolve**:

```
> deq := diff( y(x), x$2 ) + y(x) = 0;
```

$$deq := \frac{d^2}{dx^2} y(x) + y(x) = 0 \qquad (122)$$

```
> dsolve( deq, y(x) );
```

$$y(x) = \_C1 \sin(x) + \_C2 \cos(x) \qquad (123)$$

Try **?dsolve** for more information. We can specify initial conditions and experiment with parameters.

```
> de := diff( y(x), x ) = y(x) - sin(x);
```

$$de := \frac{d}{dx} y(x) = y(x) - \sin(x) \qquad (124)$$

```
> sol := dsolve( {de, y(0) = 0.1}, y(x));
```

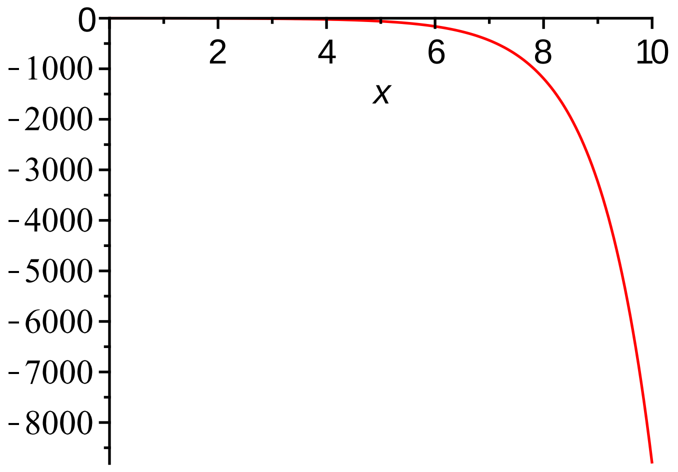
$$sol := y(x) = \frac{1}{2}\cos(x) + \frac{1}{2}\sin(x) - \frac{2}{5} e^x \qquad (125)$$

```
> sol;
```

$$y(x) = \frac{1}{2}\cos(x) + \frac{1}{2}\sin(x) - \frac{2}{5} e^x \qquad (126)$$

To graph the solution, we need only the right-hand side of **sol**, which we can get by

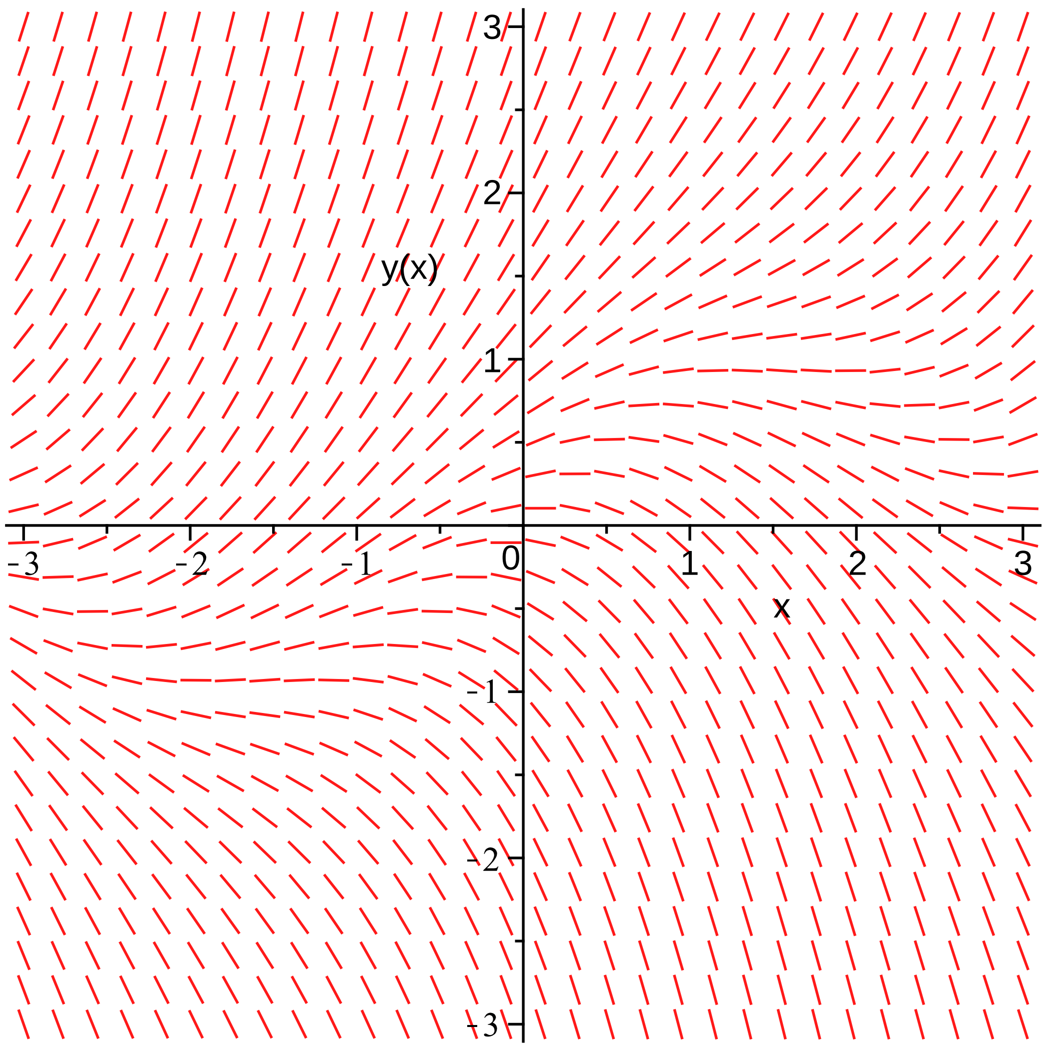
```
> plot( rhs(sol), x = 0..10);
```



Now let's try to make the slope field and draw some trajectories. We can get the slope field with the **DEplot** command from the **DEtools** package:

```
> with(DEtools):
> de := diff( y(x), x ) = y(x) - sin(x);
  DEplot( de, y(x), x = -3..3, y = -3..3, arrows = line, dirgrid
  = [30,30]);
```

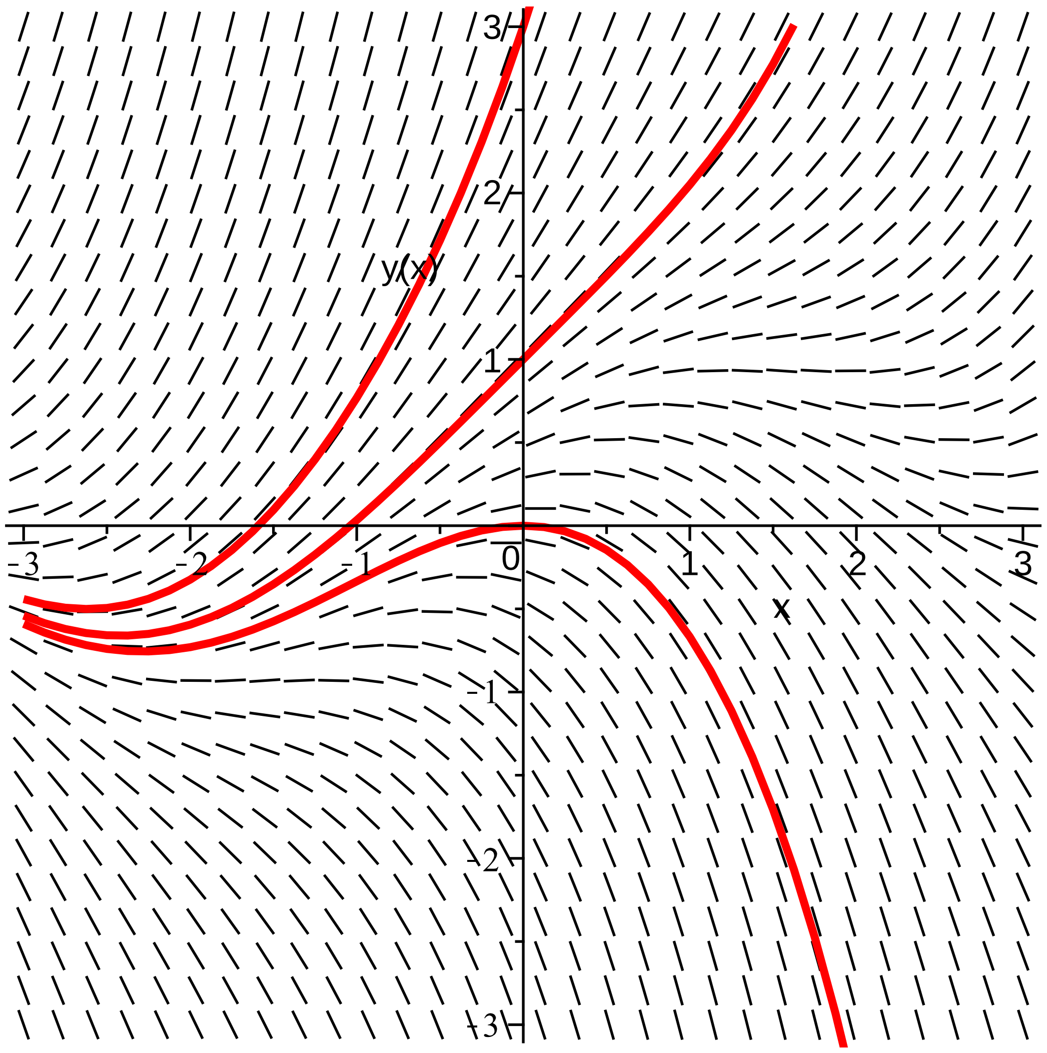
$$de := \frac{d}{dx}\, y(x) = y(x) - \sin(x)$$



Plot the slope field and a few trajectories.

```
> ics:=[ [y(0)=0], [y(0)=1], [y(0)=3] ]; # Set of initial
  conditions. A double list.
  DEplot( de, y(x), x = -3..3, y = -3..3, color='black',
  linecolor='red', arrows='line', dirgrid=[30,30], ics);
```

$$ics := [[y(0) = 0], [y(0) = 1], [y(0) = 3]]$$



Vector and Matrix Operations

To work with vectors and matrices in Maple, first load the linear algebra package:

```
> restart; # with(LinearAlgebra):
```

Define and display a vector like this:

```
> v := <1, -1>;
```

$$v := \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad (127)$$

```
> v[1];  v[2];  v;
```

$$1$$
$$-1$$
$$\begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad (128)$$

```
> print(v);
```

(129)

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad (129)$$

Note that Maple treats vectors as columns. Now define another vector **w** and do some simple computations:

```
> w := <1, 1>;
```

$$w := \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (130)$$

```
> v+w;      # vector or matrix addition
```

$$\begin{bmatrix} 2 \\ 0 \end{bmatrix} \quad (131)$$

```
> v.w;      # dot product
```

$$0 \quad (132)$$

Next we define two matrices by rows,
Syntax: A:=Matrix([row1, row2, ...]); where row1 is a comma-separated list [a,b,c ..] of row elements.

```
> A := Matrix([ [2,3], [1,2] ]);
```

$$A := \begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix} \quad (133)$$

```
> B := Matrix([ [1,1], [0,1] ]);
```

$$B := \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad (134)$$

It is easy to make new matrices (or vectors) from old ones as in

```
> augAvw:=< A| v| w >;
```

$$augAvw := \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & -1 & 1 \end{bmatrix} \quad (135)$$

Next, we do some computations with our matrices **A** and **B** and our vectors **v** and **w**:

```
> A+B;
```

$$\begin{bmatrix} 3 & 4 \\ 1 & 3 \end{bmatrix} \quad (136)$$

```
> 2*(A+B);
```

$$\begin{bmatrix} 6 & 8 \\ 2 & 6 \end{bmatrix} \quad (137)$$

```
> A.B;
```

$$\begin{bmatrix} 2 & 5 \\ 1 & 3 \end{bmatrix} \quad (138)$$

```
> A.(v+w);
```

$$\begin{bmatrix} 4 \\ 2 \end{bmatrix} \quad (139)$$

Finally, we can change the individual entries of a matrix with commands like:

```
> A[1,1] := 5;  B[2,1] := 1;
```

$$A_{1,1} := 5$$
$$B_{2,1} := 1 \quad (140)$$

```
> A,B;   # check
```

$$\begin{bmatrix} 5 & 3 \\ 1 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad (141)$$

Maple can compute inverses and transposes:

```
> A := Matrix([ [2,3], [1,2] ]);
  B := Matrix([ [1,1], [0,1] ]);
  1/A; A^(-1);
```

$$A := \begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}$$
$$B := \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$
$$\begin{bmatrix} 2 & -3 \\ -1 & 2 \end{bmatrix}$$
$$\begin{bmatrix} 2 & -3 \\ -1 & 2 \end{bmatrix} \quad (142)$$

```
> A^+; # Transpose of a matrix, swap rows and columns
```

$$\begin{bmatrix} 2 & 1 \\ 3 & 2 \end{bmatrix} \quad (143)$$

LinearAlgebra package.

This package is used to extend the basic operations illustrated above.

```
> LinearAlgebra[Determinant](A); LinearAlgebra[Determinant](B);
  with(LinearAlgebra): # load the package to shorten commands
  Determinant(A-B); Determinant(A+B);
  # Shortcut: When typing, maple tries to complete the
  command. Use the ESC key to agree, save typing.
```

```
1
1
```

$$-\frac{1}{5} \tag{144}$$

Symbolic matrices are as legitimate as numerical ones:

```
> A := Matrix([ [a,b], [c,d] ]);Determinant(A);
```

$$A := \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$a\,d - b\,c \tag{145}$$

```
> C := Matrix([ [3,2,2], [3,1,2], [1,1,1] ]);
  ReducedRowEchelonForm(C); # Put C into row-reduced form
```

$$C := \begin{bmatrix} 3 & 2 & 2 \\ 3 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{146}$$

```
> gaussjord(C);
```

$$gaussjord\left(\begin{bmatrix} 3 & 2 & 2 \\ 3 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}\right) \tag{147}$$

Elimination Steps.
The hand work on paper can be done error free by defining operations in terms of the LinearAlgebra package, which do the
three basic toolkit operations **combination, swap, multiply**. Here's the macro definition set which makes this process identical
to what is done with paper and pencil. A similar set of macros can be defined for the depreciated linalg package.

```
> # LinearAlgebra package. Reversable operations. a=matrix, t=
  target, s=source, c=constant
  combo:=(a,s,t,c)->RowOperation(a,[t,s],c); # Replace row t by c
  times row s added to row t.
   swap:=(a,s,t)->RowOperation(a,[t,s]);     # Swap rows s and t
    mult:=(a,t,c)->RowOperation(a,t,c);      # Replace row t by c
  times row t. Illegal: c=0
```

$$combo := (a, s, t, c) \rightarrow LinearAlgebra:\text{-}RowOperation(a, [t, s], c)$$
$$swap := (a, s, t) \rightarrow LinearAlgebra:\text{-}RowOperation(a, [t, s])$$
$$mult := (a, t, c) \rightarrow LinearAlgebra:\text{-}RowOperation(a, t, c) \tag{148}$$

```
> C1 := Matrix([ [3,2,2], [3,1,2], [1,1,1] ]);
```

$$C1 := \begin{bmatrix} 3 & 2 & 2 \\ 3 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \quad (149)$$

```
> C2:=combo(C1,1,2,-1);
```

$$C2 := \begin{bmatrix} 3 & 2 & 2 \\ 0 & -1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad (150)$$

```
> C3:=combo(C2,1,3,-1/3);
```

$$C3 := \begin{bmatrix} 3 & 2 & 2 \\ 0 & -1 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \quad (151)$$

```
> C4:=combo(C3,2,3,1/3);
```

$$C4 := \begin{bmatrix} 3 & 2 & 2 \\ 0 & -1 & 0 \\ 0 & 0 & \frac{1}{3} \end{bmatrix} \quad (152)$$

```
> C5:=mult(C4,2,-1);
```

$$C5 := \begin{bmatrix} 3 & 2 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{3} \end{bmatrix} \quad (153)$$

```
> C6:=mult(C5,3,3);
```

$$C6 := \begin{bmatrix} 3 & 2 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (154)$$

```
> C7:=combo(C6,2,1,-2);
```

$$C7 := \begin{bmatrix} 3 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (155)$$

```
> C8:=combo(C7,3,1,-2);
```

$$C8 := \begin{bmatrix} 3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (156)$$

```
> C9:=mult(C8,1,1/3);
```

$$C9 := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{157}$$

```
> ReducedRowEchelonForm(C1);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{158}$$

Troubleshooting
Maple's error message are irritating at first. Experience changes that.
Common Maple Coding Errors.
1. Do statements end with a semicolon or a colon?

2. Are parentheses, braces, brackets, etc., balanced? Code like `{ x,y }` is good, but `x,y}` will cause trouble.

3. Are you trying to use something as a variable to which you have already assigned a value?
To "clear" a variable x, say `unassign('x')`. You can clear many variables at once, e.g., `unassign( 'x', 'y' )`.
You will not have to reload any packages after using the unassign command. A variable that has a value assigned becomes a constant.
To display the value of an ordinary variable, type its name, followed by a semicolon, e.g., `X;`

4. If things seem hopelessly messed up, issue the command `restart;` You will have to reload any needed packages after this.

5. Are you using a function when an expression is called for, or vice versa? Syntax `g := x^2 + y^2` defines an expression while
`f := (x,y) -> x^2 + y^2` defines a function. It makes good sense to code `f(1,3);` but not so much sense to code `g(1,2);`

6. Are you using `=` when `:=` is called for? The first tests for equality (like double-equal in C++) while the second assigns a value to a variable (like single-equal in C++).

7. Are you distinguishing between the three kinds of quotes? They are: `"` the double quote, `'` the single quote, and `` ` `` the left single quote. The left signle quote
is basically used for nothing by rookies. when used, it is in expressions like

```
`"two" "three"`;   "two" "three";  `two"and"""three`;
```

`two`   `three`;   `two``three`;  `two"``"three`;

To see the difference, execute the expressions. The single back quote is not a duplicate double quote. Single quotes change a name or constant into a symbol.

For example, print(Digits) and print('Digits') have two different outputs (one is 10, the other symbol 'Digits'). Do ?quotes to settle issues.

8. If you use a function in a package, load the package first. To use **Determinant**, load LinearAlgebra. To use **display**, load **plots**. To load package **LinearAlgebra, do**

with(LinearAlgebra): Determinant(A); Simple, load a package once per session.
LinearAlgebra[Determinant](A); More typing for bulletproof code.

9. Use the shortcut **?** to inquire about the details of a Maple function, e.g., **?plot** for information about plot or just **?** for general information.

10. If all of the licenses for the current version of Maple are being used, you can use a different version by calling it up by name, i.e. by typing **xmapleV8** or **xmapleV15** in a terminal window.

References.
The WWW has many examples of Maple code. Use a search engine to find code that fits your application. Ask in the Undergraduate Computer Lab for textbooks to check out.