16 Creating user interfaces with Maple Components

16.1 Chapter Overview

It is generally recognized that graphical user interfaces (GUIs) make it easier to input certain kinds of values (typically, numbers or selections from a small number of alternatives), and to present information in a more readily grasped format.

We discuss how to set up simple GUIs in Maple using the fixed selection offered by Maple Components, sometimes referred to as widgets.

16.2 Introduction

We have promoted the use of scripts and procedures as a way of facilitating reuse of code. The textual form of instructions has gained preeminence as the way of specifying the instructions in scripts and procedures because of its flexibility and the proven utility of the text editing paradigm of working with the written word.

Programming a script is different from using it (or using repeatedly). Invoking a Maple procedure or a script requires specification of many parameter values. The task of entering parameter values through typing can be error prone because there is little that can be done to change the way keyboards work. Users can always type the wrong key by mistake.

Anyone who has used the Web is aware that there are many other ways of getting information in or out. We show some of the alternatives built into Maple. These built-in alternatives are referred to as Maple Components. The colloquial terminology for them, however, is widget.

<table>
<thead>
<tr>
<th>widget 1. A small mechanical device or control; a gadget. 2. An unnamed or hypothetical manufactured article. 3. a. An element of a GUI, such as a text box or button, that displays information or settings that can be entered or altered by the user. b. A program that performs some simple function, such as providing a weather report or stock quote, and can be accessed from a computer desktop, webpage, mobile phone or subscription television service.</th>
</tr>
</thead>
<tbody>
<tr>
<td>widget 2. A graphical representation of an object, such as a button, that is used to interact with a user interface.</td>
</tr>
<tr>
<td>widget 3. A small mechanical device or control; a gadget.</td>
</tr>
</tbody>
</table>


An assortment of input widgets

<table>
<thead>
<tr>
<th>Slider</th>
<th>RadioButton</th>
<th>Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 20 40 60 80 100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rotary dial</th>
<th>CheckBox</th>
<th>Combo Box</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An assortment of output widgets

In this chapter we explain how to set up in Maple simple GUIs connected to programming you have written.

16.3 Placing a Maple Component in a worksheet

To place a component on a worksheet, open up the Components panel of the Palette on the left hand side of the Maple window, shown below. Drag the desired component (for example, the slider), to the desired location on the worksheet.

The Component Palette in Maple

You can separate multiple components on a line by typing spaces between them, or place them on separate lines by typing *enter* (*return* on some keyboards). A more precise alignment may be achieved by inserting a table (through Insert-
>Table and selecting the number of rows and columns you want) and putting components in the desired positions. Document tables can be a mixture of text and Maple calculations as well as components.

### 16.4 Configuration of Maple Components

Maple Components typically need to be configured before they are usable. Widget configuration determines how a widget looks (for example, the label of a button), and how they work (for example, the range of values that a slider supports, or whether the component will let the user edit it once the script/program using the widget begins execution).

To configure a component, right-click on the component placed in the worksheet. A window will pop-up which allows specification of each settable component property. After configuration is complete, click on the OK button to exit configuration.

Every component has a name, which is one of the settable properties. Choosing a suggestive name allows one to more easily program applications where there are multiple components of the same type. For example, if a particular slider is to be used to specify the starting ambient temperature of an HVAC simulation, then it might be named outdoor\_Temp or something similar.

One of the settable properties is the programming that executed when the widget is used. We discuss this kind of programming in the next section.

### 16.5 Programming Maple Components

Maple components have the following theory of operation:

1. The user clicks on, types into, or drags within an input widget. This causes the widget to change its state, triggering the programming set up earlier, when it was configured.

2. The programming contained inside that widget describes what to do when the state changes. While any ordinary Maple programming (e.g. assignments, if, for, etc.) is allowed, there is also a special kind of programming that can be done with the help of the Do operation of the DocumentTools package. Typically the programming includes one or more lines of the form Do( %\text{\text{\text{\text{Name}}}} = \text{\text{\text{\text{expression}}}}); The expression can also include the name of widgets. Anything in the expression that does not begin with a % is taken as ordinary Maple.

Typically names on the right hand side of the equation are input widgets, and the values that they are set to are used in the expression to compute a result, which then is displayed by the output widget called \text{\text{\text{\text{Name}}}} listed on the left hand side of the equation. An example of this is shown in the next section.

### 16.6 An example of programming Maple Components

We're going to write an application that will plot $\sin(k\times x)$, $x = 0..10$, where $k$ is a value determined by a slider, and the color of the plot is determined from a combo box. The plot will be triggered whenever we click on the plot window. When it is working, it will look like this:
A simple application to plot a curve specified by GUI widgets

| Plotting \( \sin(k \cdot x) \) for \( x \) between 0 and 10, for a specified color |
|---|---|
| **Directions** | Operate the slider to select a value of \( k \). Select a color from the combo box. Then hit the button and a plot of \( \sin(k \cdot x) \) will be drawn. |

Value of \( k \) selected: 8

Color of plot: "red"

[Draw Plot]

How to do it

1. Create a table and place a slider, a text area, a combo box, and an embedded plot area in it. Add descriptive text to the left of the text area and the combo box. This can be done in the same way as if you were to enter text into the table without any widgets around: position the cursor where you want to enter text, enter text mode by Insert->Text or control-T (command-T on Macintosh), and then use the keyboard.

2. By right-clicking, open up the configuration box of the slider and set it to allow integer values 1 to 10. Name the widget `SliderK`. Set Major Tick marks to 5, and minor tick marks to 1. Check "show axis labels", and "update continuously", as shown:
Slider configuration

Close the configuration box by clicking on the "OK" button.

3. Right click on the text area widget and open its configuration box. Configure its name, number of visible rows, and make it not editable. We set the latter property because we want to force the user to use the slider to set the value of k.

Text Area Configuration

Close the configuration box of the text area by clicking "OK".

4. Configure the combo box to allow the options "Choose color", "red", "green", and "blue". Name the widget "colorBox".
5. Configure the plot area widget so that its name is *sinPlotter*.

6. We will now program the slider and the button so that when operated they affect the text area and plot area, respectively. Right-click again on the slider, and select "Configuration Properties". When the configuration box appears, click on the "Edit...." button for "Action When Contents Change". This brings up a code edit box. Add the line `Do(%kText = %SliderK)` to the window, so that it looks like this:

**Programming of Slider**

```maple
use DocumentTools in
    # Enter Maple commands to be executed when the specified
    # action is carried out on the component.
    # Use:
    #    Do(%component_name);
    # and
    #    Do(%component_name - value);
    # to set and get properties of the component.
    # You can also use arbitrary expressions
    # involving components, e.g.:
        #    Do(%target = %input1 + %input2);
        # Note the %-prefix to each component name.
        # See ?CustomizingComponents for more information.

    Do(%Text = %SliderK)
end use;
```

Most of the code that appears in the box are comments that explain what to do -- add a `Do(...)` expression in the space provided between the `use DocumentTools in ...` and the `end use;` commands. `Do(...)` does a `with` with temporary rather than permanent effect. Recall from Chapter 8 that a `with` makes all the operations of a package available through their short name. `use DocumentTools ... end use` makes it possible to refer to the operation `DocumentTools[Do]` as just `Do` within the region of code delimited by the `use ... end use`. Click on "OK" for the code edit box, and then again on "OK" in the configuration box to complete the programming of the slider.

5. In a similar fashion, program the *Draw Plot* button so that its code edit box looks like this:
6. You should now be able to operate the widgets -- select a value of k, select a color, and then hit the *Draw Plot* button to draw the plot.

### 16.7 Troubleshooting Maple Components

**Syntax errors in Component code** The code edit box of Maple components has a "check syntax" button that can be operated before you close the programming window. Furthermore, if the programming window has "check syntax before closing", the window cannot be closed until all syntax errors are fixed in the window. We recommend that you do fix all syntax errors before closing the code edit box, since that will simplify subsequent attempts to debug the code. Of course, guaranteeing that the programming is free of typos does not mean that it will work properly, if there's still a difference between what you say and what you mean to say.

**Code edit box with syntax error**
A component code region with a syntax error in it. The programmer has mistakenly used `:=` rather than `=` in the attempt to cause the plot to be displayed in the GUI widget sinPlotter. This error message appears if you click on the OK button or the Check Now button of the code region box.

**Symptom: proper output does not appear in an output widget.** The programming for widgets is a combination of ordinary Maple programming and the %/equation programming that references the value or output of a widget. Double check what you are writing until you become expert on this variant of ordinary Maple programming.

**Example of GUI widget programming with subtle mistake**

<table>
<thead>
<tr>
<th>Example of GUI widget programming with subtle mistake</th>
</tr>
</thead>
<tbody>
<tr>
<td>use DocumentTools in</td>
</tr>
<tr>
<td># Enter Maple commands to be executed when the specified # action is carried out on the component.</td>
</tr>
<tr>
<td># Use:</td>
</tr>
<tr>
<td># <code>Do( %component_name );</code></td>
</tr>
<tr>
<td># and</td>
</tr>
<tr>
<td># <code>Do( %component_name = value );</code></td>
</tr>
<tr>
<td># to set and get properties of the component.</td>
</tr>
<tr>
<td># You can also use arbitrary expressions</td>
</tr>
<tr>
<td># involving components, e.g.:</td>
</tr>
<tr>
<td># <code>Do( %target = %input1 + 2*%input2 );</code></td>
</tr>
<tr>
<td># Note the <code>%</code>-prefix to each component name.</td>
</tr>
<tr>
<td># See ?CustomizingComponents for more information.</td>
</tr>
<tr>
<td><code>Do( sinPlotter = plot(sin(%SliderK * x), x = 0..10, color=%colorBox));</code></td>
</tr>
<tr>
<td>end use;</td>
</tr>
<tr>
<td><img src="image" alt="Check syntax before saving" /> <img src="image" alt="Check Now" /> <img src="image" alt="Cancel" /> <img src="image" alt="OK" /></td>
</tr>
</tbody>
</table>

The mistake is that the programmer used `sinPlotter` rather than `%sinPlotter`. Doing this produces no error message, but the plot area named `sinPlotter` does not display any plots as a result of this line of code. Editing the window so that the line of code says `Do (%sinPlotter = ...)` fixes the problem.

**Ambiguous settings**

Input widgets such as sliders or dials have the drawback that if there are many possible values it can be unclear to the user which value was selected. This is not a bug in programming, but it may cause malfunction of the application. For this reason, it can be good to set up a label or text area that displays the numerical value of the selection to supplement the reading of the dial or gauge, or to consider alternative tick mark and labeling.
Three ways of reading the setting of a dial

Three ways of reading the setting of a dial

| Dial set to | 
| --- | --- |
| ![Dial1](image1.png) | This dial is set to an integer value, but is it 27, 28 or 29? |
| ![Dial2](image2.png) | It's unambiguous what this dial is set to. It may not be that easy to set the dial to a particular value, but at least we know what the setting is. |
| ![Dial3](image3.png) | We have configured this dial in a way so that it's clear which of the possible settings was selected. Each setting has its own tick mark, and it's not too hard to figure out the value even if not every tick mark has a label. We couldn't do this very easily for a dial that needed to support 100 different settings. |

16.8 Programming of Maple Components contrasted with other kinds of programming

Specialized mini-languages

Widget programming is simple but uses a special kind of syntax that works only with the DocumentTools package and with widgets. In this respect it is similar to the programming done within printf for formatted output — %f, %e, 'n', etc. are found only in strings used within printf (and its cousins, sprintf, and fprintf) and are a way of doing the mini-programming necessary for a fine control of the output in Maple of numbers, strings, and other kinds of expressions.

The use of specialized mini-languages is a general phenomenon in computer systems. Such mini-languages tend to arise in situations where having a succinct notation that is limited to certain applications nevertheless leads to highly productive use of programmer time. Other examples of mini-languages are the language used to format web pages, HTML (hypertext markup language — see http://www.w3.org/MarkUp/Guide/), the language of regular expressions used to specify patterns in text search (http://www.regular-expressions.info/tutorial.html).

Such languages underscore the notion that proficient computer usage requires the willingness to learn several languages. Furthermore, learning languages is an on-going phenomenon: new languages will be invented and become popular as new types of problems are tackled on computers.

Programming by configuring icons

"Drag and configure to program" is a paradigm used in a number of other languages, such as the NXT language used to program Lego robots, or LabView (http://www.ni.com/labview/), a language used to control and process data from devices in a laboratory. These languages differ from Maple Component programming in that a) the objects found on the screen with NXT or LabView programming stand and control other things such as wheels, voltage meters, light switches, etc whereas the programming of Maple components controls the component seen in the worksheet. b) The programming diagrams in NXT or LabView includes not only the icons, but lines of control and flow between them. One can see the order in which things are executed in a NXT or LabView programming by following the connecting lines. With Maple, the connections between components are implicit rather than visual — mentioning the name of a
component in another component's code edit box establishes the cause-and-effect between components, but there are no visual connections between components.

**NXT programming**

The yellow and green icons are individual instructions. Each icon can be clicked on to configure its properties and therefore affect the programming. This diagram describes a complete program for control of a robot. Execution begins at the left and follows the paths and arrows indicated in the diagram. In this case, the backward orange arrow at the time indicates that a looping occurs.
### LabView programming

LabView programming (from http://www.ni.com/labview/whatis/)

<table>
<thead>
<tr>
<th>BASICS</th>
<th>Faster Programming</th>
</tr>
</thead>
</table>

**Graphical Programming**  
Program with drag-and-drop, graphical function blocks instead of writing lines of text

**Dataflow Representation**  
Easily develop, maintain, and understand code with an intuitive flowchart representation

LabView is used to control devices. Connections between icons indicate the flow of data between lab devices that "emit" data and those that process, record or use data.

### 16.9 Final words on user interfaces

Interactive systems such as Matlab, Mathematica, and Python have similar techniques for quickly setting up GUIs and connecting them to computational code. There are some languages such as Tcl/Tk, Java, or C# where custom-built GUI widgets can be implemented. This allows designers to go beyond the limitations of whatever built-in components may provide. It may be possible to import a custom-made widget written in Java into one of the interactive systems and thus get both customizability and lots of pre-defined mathematical functionality.

In some applications, the program designer envisions that all the interaction will be through the GUI widgets. Interaction with the normal Maple worksheet may even be disadvantageous if the typical user does not know Maple syntax or cannot be trusted to avoid entering critical values that would break the application. For these kinds of situations, Maple provides Maplets, which is a way of running Maple so that only the GUI is shown and it is the only way the user can interact with the program. It is also possible to have a Maplet encoded in a web page so that the user's computer need only connect to a web server that has Maple and need not run Maple itself.
16.10 Summary of the chapter

To use a Maple Component as for GUI input or output:

a) Drag a copy of a component from the Components palette, to the desired location of the worksheet. b) Right-click (control-click) on the component in the worksheet to open the configurations box. c) Specify the configuration for the component. Name the widget something that suggests its purpose, rather than a generic name such as "Slider1". This will make it easier to remember names when you are using many widgets in an application.

d) Open and edit the code edit box to specify the actions that should occur when the user changes the state of the component. Typically, one affects other components by putting in an equation that mentions the widgets by name. The component programming distinguishes between ordinary Maple variables and functions, and widgets by preceding widget names with a %.

---

**Slider configuration**

<table>
<thead>
<tr>
<th>Name: Slider</th>
</tr>
</thead>
<tbody>
<tr>
<td>ToolTip:</td>
</tr>
<tr>
<td>Value at Lowest Position: 0</td>
</tr>
<tr>
<td>Value at Highest Position: 10</td>
</tr>
<tr>
<td>Current Position: 8</td>
</tr>
<tr>
<td>Spacing of Major Tick Marks: 5</td>
</tr>
<tr>
<td>Spacing of Minor Tick Marks: 1</td>
</tr>
</tbody>
</table>

**Action When Value Changes:**

- Enable Input
- Visible
- Show Track
- Orient Vertically
- Show Axes Labels
- Show Axis Tick Marks
- Snap to Axis Tick Marks
- Update Continuously while Dragging

---

**Programming of Slider**

```
use DocumentTools in
# Enter Maple commands to be executed when the specified
# action is carried out on the component.
# Use:
# Do( %component_name );
# and
# Do( %component_name = value );
# to set and get properties of the component.
# You can also use arbitrary expressions
# involving components, e.g.:
# Do( %target = %input1 + 2*%input2 );
# Note the %-prefix to each component name.
# See ?CustomizingComponents for more information.

Do( %Text = %SliderK )

end use;
```

---

- Check syntax before saving
- Check Now
- Ok
- Cancel
17 Calculus and optimization

17.1 Chapter Overview

One way that Maple and similar systems differ from calculators is that they can draw upon the hardware and storage resources to provide knowledge beyond arithmetic, being able to handle most of collegiate-level mathematics. We introduce the built-in functions for doing differentiation, limits, and finding minima and maxima. Like solve, the minima and maxima-finders come in two styles -- one that tries to calculate formulae for answers (exact calculation) and one that uses approximation techniques that work only when the answer is numeric.

17.2 Working with more mathematics

We have become accustomed to Maple's ability to do numerical calculation. Contrasting this with the kind of calculation available from today's calculators, we observe that it is different qualitatively from the calculation provided by a calculator because a) Programming (e.g. looping) allows us to do much larger quantities of calculation than we could through operations only through keystroke. b) Programming allows us to more easily reuse the calculation on varieties of similar problems. c) The document interface allows us to record both what we did and the results we got for future reference and/or use in presentations and documents d) We have taken advantage of some relatively powerful operations, such as solve which makes solving certain common problems much more convenient (even with the significant work to enter the expression). e) We have taken advantage of the more powerful visualization features available on a computer to enhance understanding of the phenomena being simulated. In addition to programmability and a more extensive user interface, computers in 2010 have another edge over calculators -- there is more room to store much more systems programming. Systems such as Maple, Mathematica and Matlab use this room to provide literally thousands of extra mathematical features. Some of it is immediately useful to first year university/college students in the form of symbolic computation -- results from calculus. Some of it involves operations and functions that you will hear about after the first year -- linear algebra, multivariate calculus, ordinary and partial differential equations, Laplace transforms, etc. Learning how to operate a system such as Maple gives you the understanding to pick up many of these extra features on your own, after you learn the additional mathematics. We give you a taste of this by exploring some of operations that are comprehensible with first year mathematics.

17.3 Differentiation, simplification

As it is taught in traditional first-year calculus, differentiation is an operation on functions. Maple knows how to differentiate all of the common functions found in calculus. It is particularly useful for performing differentiation when it would take a lot of algebraic manipulation to do the operations by hand.

diff as a function takes two or more arguments. The first argument must be (or evaluate to) an expression. The second argument must be (or be an expression that evaluates to) the variable of differentiation. The there are third, fourth, etc. arguments provided, they are used as variables of higher derivatives.

The result of diff is an expression or a number if the derivative is a numerical constant.

Evaluation of a derivative expression can occur through eval as with other expressions.

Symbolic differentiation using the expression palette

<table>
<thead>
<tr>
<th>Expression</th>
<th>Differentiate w.r.t. x</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^2 - 2x + 5$</td>
<td>$\frac{d}{dx}$</td>
<td>$2x - 2$</td>
</tr>
<tr>
<td>$\sin(\omega t + 5)^2$</td>
<td>$\frac{d}{d\omega}$</td>
<td>$2\sin(\omega t + 5)\cos(\omega t + 5)$</td>
</tr>
<tr>
<td>$\frac{d}{d\omega}$</td>
<td>$\frac{d}{dt}$</td>
<td>$2\cos(\omega t + 5)\omega - 2\sin(\omega t + 5)^2\omega^2$</td>
</tr>
<tr>
<td>simplify symbolic</td>
<td></td>
<td>$2\omega^2(2\cos(\omega t + 5)^2 - 1)$</td>
</tr>
</tbody>
</table>

Enter the expression, and then right-click (control-click on Macintosh) to bring up the calculation options. Select differentiate->x.

We can calculate the second derivative with respect to $t$ of this expression by performing differentiation twice. We simplify the expression a bit by performing the operation simplify->symbolic.
\[
\frac{d}{d\omega} \sin(\omega t + 5)^2 = 2 \sin(\omega t + 5) \cos(\omega t + 5) t
\]

If we want the document to display the mathematical notation for the derivative, we can select \( \frac{d}{d\omega} f \) from the expression palette and then fill in the slots \( f \) and \( x \). We can then get Maple to calculate the derivative by typing control–=.

### Symbolic differentiation and evaluation of derivatives, textually

<table>
<thead>
<tr>
<th>Example</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>( expr := x^2 - 2x + 5 )</td>
<td>Find the first derivative of the expression with respect to ( x ).</td>
</tr>
<tr>
<td>( \text{diff}(\text{expr}, x) )</td>
<td>( x^2 - 2x + 5 ) (17.1)</td>
</tr>
<tr>
<td>( posExpr := \sin(\omega t + 5)^2 )</td>
<td>Find the first derivative of the expression with respect to ( t ).</td>
</tr>
<tr>
<td>( \text{diff}(\text{posExpr}, t) )</td>
<td>( \sin(\omega t + 5)^2 ) (17.3)</td>
</tr>
<tr>
<td>( \text{diff}(\text{posExpr}, t, t) )</td>
<td>( 2 \sin(\omega t + 5) \cos(\omega t + 5) \omega ) (17.4)</td>
</tr>
<tr>
<td>( \text{diff}(\text{expr}, t) )</td>
<td>( 0 ) (17.6)</td>
</tr>
<tr>
<td>( \text{simplify}(2 \cos(\omega t + 5)^2 \omega^2 - 2 \sin(\omega t + 5)^2 \omega^2) )</td>
<td>diff's second argument must be the variable of differentiation. Note that if the variable doesn't occur in the expression, the derivative (according to the mathematical definition) is zero. If you are surprised by getting a zero derivative, check that you are using the correct variable to differentiate with respect to.</td>
</tr>
<tr>
<td>( \text{eval}(2x - 2, x = 3) )</td>
<td>( 4 ) (17.8)</td>
</tr>
<tr>
<td>( \text{eval}(2 \omega^2 (2 \cos(\omega t + 5)^2 - 1), t = 47.0) )</td>
<td>This is a way to compute ( \frac{d}{dx} x^2 - 2x + 5 \bigg</td>
</tr>
<tr>
<td>( \text{eval}(2 \omega^2 (2 \cos(47.0 \omega + 5)^2 - 1), t = 47.0) )</td>
<td>This is a way to compute ( \frac{d^2}{dt^2} \sin(\omega t + 5)^2 \bigg</td>
</tr>
</tbody>
</table>

### Plotting a function and its derivative together

<table>
<thead>
<tr>
<th>Example</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f := \langle a, b \rangle \rightarrow \sin(a) + \cos(b) )</td>
<td>We're already used to functions that take one or two arguments. Note that unlike mathematical convention, in programming vari-</td>
</tr>
</tbody>
</table>
\[(a, b) \rightarrow \sin(a) + \cos(b)\] (17.10) \n
able names often do have names that are more than one letter long. This is to increase intelligibility to readers looking at the programming. Although it seems minor, ease of comprehension can play an significant role in the cost of developing and using software, so is an important engineering concern.

\[g := (\alpha) \rightarrow f\left(\frac{\alpha}{2}, \frac{\alpha}{3}\right)\]

\[\alpha \rightarrow f\left(\frac{1}{2} \alpha, \frac{1}{3} \alpha\right)\] (17.11) In this example, we plot a "strange" function built out of trigonometric parts, and plot it and its derivative. Since \(g\) is a function \(g(t)\) will evaluate to the expression \(\cdot\) Thus the plotting variable should be \(t\) rather than \(\alpha\) or some other variable.

\[\text{plot}(g(t), t = 0 \ldots 0.30)\]

\[fderiv := \text{diff}(g(\alpha), \alpha)\]

\[\frac{1}{2} \cos\left(\frac{1}{2} \alpha\right) - \frac{1}{3} \sin\left(\frac{1}{3} \alpha\right)\] (17.12) The value of \(fderiv\) is an expression involving alpha.

\[\text{plot}(\{g(\alpha), fderiv\}, \alpha = 0 \ldots 0.30)\]

We plot two expressions involving the variable \(\alpha\) on the same plot. The use of a set \(\{\ldots\}\) as a first argument to \textit{plot} to do multiple plots was first explained in section 6.3.
We need to use \( \alpha \) as the plotting variable here since the value of \( \text{fileriv} \) is an expression involving \( \alpha \).

### 17.4 Limits

You can use the clickable interface to compute limits by selecting the appropriate item from the Expression palette and then filling in the template as needed. Maple uses calculus techniques (e.g. l'Hôpital's Rule) to compute limits symbolically.

**Clickable interface version of limits**

<table>
<thead>
<tr>
<th>Example</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \lim_{x \to 3} \frac{1}{(x-3)^2} ]</td>
<td>( \infty ) (17.13) The result of ( \text{limit} ) can be an expression (possibly involving positive or negative infinity) as well as the symbol ( \text{undefined} ).</td>
</tr>
<tr>
<td>[ \lim_{x \to a} \frac{\sin(x)}{x} ]</td>
<td>0 (17.14) Even though the limit exists as ( t ) approaches from the right or left, they do not agree, so there is no &quot;two-sided&quot; limit.</td>
</tr>
<tr>
<td>[ \lim_{t \to 0} \frac{1}{t} ]</td>
<td>( \text{undefined} ) (17.15)</td>
</tr>
<tr>
<td>[ \lim_{t \to 0^+} \frac{1}{t} ]</td>
<td>( -\infty ) (17.16) To take a one-sided limit, add a &quot;+&quot; or &quot;,&quot; superscript to the limit point.</td>
</tr>
<tr>
<td>[ \lim_{x \to a} \frac{\sin(x)}{x} ]</td>
<td>( \frac{\sin(a)}{a} ) (17.17) Note that the value returned as the limit for ( x=a ), while true for most values of ( a ), is not really valid for ( a=0 ).</td>
</tr>
</tbody>
</table>

The textual version of taking limits involves the \( \text{limit} \) function. \( \text{limit} \) takes at least two arguments. The first argument is the expression that you wish to take the limit of. The second argument is an equation indicating the limit variable and the limiting value. If you supply a third argument, it indicates whether a "right sided", "left sided" limit is desired instead of a two-sided limit.

**Textual version of limits**

<table>
<thead>
<tr>
<th>Example</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{limit}(1/(x-3)^2,x=3) )</td>
<td>( \infty ) (17.18) The first argument is the expression that you wish to take the limit of. The second argument is an equation indicating the limit variable and the limiting value.</td>
</tr>
<tr>
<td>( \text{limit}(\sin(x)/x,x=\text{infinity}) )</td>
<td>0 (17.19) You can use &quot;infinity&quot; or &quot;-infinity&quot; as the way of specifying ( \infty ) or ( -\infty ) textually without use of the palette.</td>
</tr>
</tbody>
</table>
\[
\begin{array}{|l|l|}
\hline
\text{limit(1/t,t=0,left)} & -\infty \\
\text{(17.20)} & \text{The third optional third argument to limit can specify a one sided limit. This is the textual way of specifying } \lim_{t \to 0^-} \frac{1}{t}. \text{ There is no simple way of specifying this.} \\
\hline
\text{limit(1/t,t=0,right)} & \infty \\
\text{(17.21)} & \text{This is the textual way of specifying } \lim_{t \to 0^+} \frac{1}{t}. \\
\hline
\end{array}
\]

One could make a case that the second argument for the textual form of \textit{limit} should be something looking like \textit{var -> limit value} since that is the more conventional terminology. But because of the limitations of Maple's processing capabilities for its programming language, it would be more expensive to support arrows for both this meaning in limits and the use of -> in function definitions. So users must get used to using equations rather than the standard math symbol for "approaches".

\subsection*{17.5 Finding minima and maxima by using Maple as a calculus calculator}

A standard topic in elementary calculus is how to find the minimum or maximum of a continuous function in an interval (sometimes referred to as the \textit{local extrema of the function}) through the use of derivatives and a little algebra. The function that you are doing this to is sometimes referred to as the \textit{objective function}. Maple can be used to do the same calculations as you would do by hand -- to find the extrema of the objective function, take its derivative with respect to \(x\) (using \textit{diff}), and then find where the derivative is equal to zero using \textit{solve} or \textit{fsolve}. The advantages of doing it with a system such as Maple are the same as with other calculations: a) it is easier to do correctly if the formula is involved or if you don't remember all the math, b) you can easily organize the information about the problem and its solution using Maple documents so that it can be easier to recall and presented, c) the speed of re-execution makes it easy to handle a bundle of similar extrema problems.

\textbf{Finding the minimum and maximum using textually-specifed commands}

From Anton, Calculus 8th ed. p. 307 problem 13 (modified)

Find the absolute maximum and minimum values of

\[
f(x) = 2 - 2 \sin(x) \quad \text{for } x \in \left[ -\frac{\pi}{4}, \pi \right]
\]

\textbf{Solution}

\[
f(x) = 2 - 2 \sin(x)
\]

\[
x \rightarrow 2 - 2 \sin(x) \quad \text{(17.22)}
\]

\[
\text{We define a continuous function}
\]

\[
\text{And compute its derivative symbolically.}
\]

\[
\text{We generate a plot to get an idea of where to look for minima and maxima. Evidently the minimum is in the interior of the interval middle and the maximum is at the left.}
\]
Finding where the derivative is zero should produce the minimum.

\[
x_{\text{min}} := \text{solve}(\text{deriv} = 0, x)
\]

\[
\frac{1}{2} \pi
\]  
(17.24)

\[f(x_{\text{min}})\]

\[0\]  
(17.25)

The minimum is confirmed to be zero -- the graph seems to indicate that but the picture doesn't let us know whether it's really zero or just really small.

\[f\left(-\frac{\pi}{4}\right)\]

\[2 + \sqrt{2}\]  
(17.26)

Standard calculus procedure says to check out the value of the end points for the maximum. One is clearly larger than the other, and is bigger than the place within where the derivative is zero.

\[f(\pi)\]

\[2\]  
(17.27)

Thus we conclude that \(f\) attains a maximum at \(x = -\frac{\pi}{4}\). The value there is \(2 + \sqrt{2}\). The minimum is zero, at \(x = \frac{\pi}{2}\).

Finding the minimum and maximum using textually specified commands, part 2

**Finding the** minimum and maximum using textually specified commands, part 2

Anton, Calculus, 8th ed. Problem 307 (modified)

Find the absolute minimum of \((x^2 + x + 5)^{\frac{2}{3}}\) for \(x\) in \([-2..3]\)

\[
(x^2 + x + 5)^{\frac{2}{3}}
\]

\[
\rightarrow
\]

We enter the expression, and plot it on -2.3 using the 2-D plot builder.

We do a chain of calculations to find what is obviously the minimum point.

Compare to example 12.1.2 where because we assigned results to variables, we could just type in the name of the variable again. Here we have to copy the expression.

We had to copy the expression again, and type in "-1/2" to calculate the minimum value. Once we have a number, we can also see approximately what it is.
We turn the sequence of actions into a textual script. The parameters are the expression to be minimized, the variable in the expression and the region. The plot is just to help us have a visual check that the numeric answer result is probably correct. We could get Maple to tell us what the variable was, automatically, but that may be more work than we want to do.

```
expr := (x^2 + x + 5)^(2/3)

var := x

interval := -2..3

plot(expr, var = interval)
```

\[ \frac{2}{3} \frac{2x + 1}{(x^2 + x + 5)^{1/3}} \]  \hspace{1cm} (17.31)

\[ \frac{1}{2} \]  \hspace{1cm} (17.32)

\[ \frac{1}{4} \cdot 4^{1/3} \]  \hspace{1cm} (17.33)

\[ 2.0825719660 \]  \hspace{1cm} (17.34)

We turn this into a textual script that does the same thing. Unless we print the plot, it doesn’t appear because by default Maple only displays the result of the procedure. print causes something to be printed in addition to the final result.

```
findMin := proc(expr, var, interval)
    local deriv, soln, valAtPoint;
    print(plot(expr, var=interval));
    deriv := diff(expr, var);
    soln := solve(deriv, var);
    valAtPoint := eval(expr, var=soln);
    return evalf(valAtPoint);
end;

proc(expr, var, interval)
    local deriv, soln, valAtPoint;
    print(plot(expr, var=interval));
    deriv := diff(expr, var);
    soln := solve(deriv, var);
    valAtPoint := eval(expr, var=soln);
    return evalf(valAtPoint)
end proc
```
We solve the minimization problem.

\[
\text{findMin}\left(\left(x^2 + x + 5\right)^{\frac{2}{3}}, x, 2.3\right)
\]

\[
2.825719660
\]  

(17.35)

Now that we have a procedure, we can solve a similar problem: minimize \(\sqrt{x^2 + x + 3}\) for \(y\) in the interval \([-1, 4]\). We see that the plot supports the numerical calculation that the value of the expression at the minimum is approximately 1.65.

\[
\text{findMin}(\sqrt{y^2 + y + 3}, y, -1.4)
\]

\[
1.658312395
\]  

(17.36)
17.6 One-step extrema finding using maximize and minimize

Maple also provides both exact and approximate numerical functions that do the entire sequence of steps needed to find extrema. In this section, we discuss the Maple functions that provide exact solutions to such problems. The Maple function for finding maxima is, appropriately enough, called maximize. If expr is an expression involving a variable \( x \), then maximize(expr;x=range) will produce maximum value of the expression for \( x \) in that range. If there is no maximum value, or if Maple can't find it, NULL is returned. Thus if maximize returns an answer, it should be "believable", but the absence of an answer doesn't meant that there is no answer. infinity or -infinity may be an answer if the function is unbounded within the range.

We have already seen max and min, which are also built-in Maple functions. However, they only work on lists, sets of values. They don't work at finding minima or maxima of expressions.

It is often a good idea when working on extrema problems to do a plot of the expression in question, so that you can some notion of where the maxima will be and what their values will be like.

**minimize and maximize**

From Anton, Calculus 8th ed. p. 307 problem 13 (modified)

Find the absolute maximum and minimum values of

\[
f := \left(x \rightarrow 2 - 2 \sin(x) \right) \text{ for } x \in \left[\frac{\pi}{4}, \pi\right]
\]

**Solution**

\[
f := \left(x \rightarrow 2 - 2 \sin(x) \right)
\]

\[
x \rightarrow 2 - 2 \sin(x)
\] (17.37)

We define a function.

We generate a plot to get an idea of where to look for minima and maxima. Evidently the minimum is in the middle and the maximum is at the left.

**plot** \[
\frac{\pi}{4}, \pi
\]

The minimum is confirmed to be zero -- the graph seems to indicate that but the picture doesn't let us know whether it's really zero or just really small. Note again that this is the minimum value, it is not the location of the minimum value.
17.6 One-step extrema finding using maximize and minimize • 251

\[ \text{largestValue} := \text{maximize}\left(f(x), x = -\frac{\pi}{4}, \pi \right) \]

\[ 2 + \sqrt{2} \]  

(17.39)

Finding the maximum is a copy/edit job once we've worked out how to do the minimum.

Sometimes, you want not only what the maximum or minimum value is, but also where it is. If you give maximize or minimize an extra third parameter, location, then it will return a sequence for a result. This sequence has a logical but somewhat intricate structure. You can use a chain of selections to extract the location \textit{var= location point} from the answer given by Maple.

**Solving a problem with minimize and maximize**

From Anton, Calculus 8th ed. p. 307 problem 13 (modified)

Find the absolute maximum and minimum values of

\[ f := (x) \to 2 - 2 \cdot \sin(x) \quad \text{for} \quad x \in -\frac{\pi}{4}, \pi \]

Solution

\[ f := (x) \to 2 - 2 \cdot \sin(x) \]

\[ x \to 2 - 2 \sin(x) \]

(17.40)

We define a function.

We generate a plot to get an idea of where to look for minima and maxima. Evidently the minimum is in the middle and the maximum is at the left.

\[ \text{plot}\left(f, -\frac{\pi}{4}, \pi \right) \]

\[ \text{smallestValue} := \text{minimize}\left(f(x), x = -\frac{\pi}{4}, \pi \right) \]

\[ 0 \]

(17.41)

This is what we did in Example 12.2.1 -- we just get the smallest value of \( f \).

\[ \text{minResult} := \text{minimize}\left(f(x), x = -\frac{\pi}{4}, \pi, \text{location} \right) \]

\[ 0, \{x = \frac{\pi}{2}, 0\} \]

(17.42)

This is what minimize does with the extra parameter location.

\[ \text{minLocation} := \text{rhs}(\text{minResult}[2][1][1][1]) \]

This longwinded sequence of selections is needed to get just the value of the minimum. We need to extract the second part of the
\[ \frac{1}{2} \pi \] (17.43)

sequence minResult to get the set \( \left\{ \left( x = \frac{1}{2} \pi, 0 \right) \right\} \), then the first part of the first part of that set to get \( x = \frac{1}{2} \pi \). Evaluating \( x \) at \( x = \frac{1}{2} \pi \) gives the result. We could try to do this using the right-click menu but it would probably be more trouble than convenient. This is a situation where the textual form of the command to produce the result is about the same level of difficulty to enter as the clickable interface.

\[ \text{maxResult} := \text{maximize} \left( f(x), x = -\frac{\pi}{4}, \frac{\pi}{4}, \text{location} \right) \]
\[ \left\{ x = -\frac{1}{4} \pi, 2 + \sqrt{2} \right\} \] (17.44)

Finding the maximum is a copy/edit job once we've worked out how to do the minimum.

\[ \text{maxValue} := \text{maxResult}[1] \]
\[ 2 + \sqrt{2} \] (17.45)

\[ \text{maxLocation} := \text{rhs}(\text{maxResult}[2][1][1][1]) \]
\[ -\frac{1}{4} \pi \] (17.46)

If you want to evaluate another expression at a minimum point, then \( \text{eval}(\text{expr}, \text{result}[2][1][1]) \) will do that. This is because result[2] contains a set which describes the location/value pair. result[2][1] returns the first element of that, which is a list containing a particular location/value pair. result[2][1][1] is a set of the form \{var = extreme value\}. \( \text{eval}(\text{expr}, \{\text{var} = \text{value}\}) \) will evaluate \( \text{expr} \) with \( \text{value} \) replacing all occurrences of \( \text{var} \) in \( \text{expr} \).

Optimization in a production problem with minimize and maximize


A liquid form of penicillin manufactured by a pharmaceutical firm is sold in bulk at a price of $200 per unit. If the total production cost (in dollars) for \( x \) units is

\[ C(x) = 500000 + 80 \cdot x + 0.003 \cdot x^2 \]

and if the production capacity of the firm is at most 30,000 units in a specified time, how many units of penicillin must be manufactured and sold in that time to maximize the profit? What will the revenue be at the maximal profit?

Solution

\[ C := (x) \rightarrow 500000 + 80 \cdot x + 0.003 \cdot x^2 \]
\[ x \rightarrow 500000 + 80 \cdot x + 0.003 \cdot x^2 \] (17.47)

\[ R := (x) \rightarrow 200 \cdot x \]
\[ x \rightarrow 200 \cdot x \] (17.48)

\[ P := (x) \rightarrow R(x) - C(x) \]
\[ x \rightarrow R(x) - C(x) \] (17.49)

\[ \text{plot}(P(x), x = 0..30000) \]

We define a Cost, Revenue, and Profit function, using the information given in the problem about cost, and the standard definitions from economics for revenue and profit. We could use the proc..end proc way of defining a function to equal effect but the arrow notation was designed to make it convenient to enter these short definitions that don't need \textbf{if} or \textbf{for}.

We plot the profit function to scope out its behavior. Evidently the function does hit a maximum, roughly around \( x=20000 \). If we got \( x=35 \) as the answer from our maximum calculation in Maple, we would wonder if we did things correctly.
### 17.7 One-step approximations to extrema using Optimization[Maximize] and Optimization[Minimize]

Maple has solve and fsolve that provide exact and approximate solutions. Similarly there is Optimization[Maximize] and Optimization[Maximize] play the role of fsolve to maximize and minimize. These functions only work when there is a numerical answer, and then only try to produce a "best effort" approximation which despite "best effort" of the programmers may not necessarily be that good an approximation. Its quality should always be tested against expectations, and to see how well it works. The clickable interface's "Optimization" operation also invokes the approximate optimizers.

Optimization[Maximize] and Optimization[Minimize] always return a list. The first item of the list is the extreme value, and the second item in the list is a list of equations describing the location of the extreme value. As with the exact optimizing functions maximize and minimize, the second item can be used as a parameter to eval to evaluate an expression at the location of an extreme value.

<table>
<thead>
<tr>
<th>Finding an approximation to the minimum with Optimization[Minimize]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anton, Calculus 8th ed. problem 47, p. 321 (modified)</td>
</tr>
</tbody>
</table>
Two particles, A and B, are in motion in the xy-plane. Their coordinates at each instant of time $t$ ($t \geq 0$) are given by $x_A = t$, $y_A = 2t$, $x_B = 1 - t$, $y_B = t$. Find the minimum distance between $A$ and $B$.

$$x_A := t$$

$$y_A := 2 \cdot t$$

$$x_B := 1 - t$$

$$y_B := t$$

$$p1 := [x_A, y_A]$$

$$p2 := [x_B, y_B]$$


$${dist}(p1, p2) = \sqrt{5t^2 - 4t + 1}$$

$p1$ and $p2$ are lists of the x and y positions for each point. $dist$ is the objective function. This is the Euclidean distance between the two points. We can compute the distance between $p1$ and $p2$ using it.

It's always good to have an independent way of evaluating the answer you get from a "solver". In this case, we can plot the
function and from that see approximately where the answer ought to be. Our first plot does not show the minimum that clearly, so we do another plot that emphasizes the region of interest.

\[
\text{plot(dist(p1, p2), t = 0..2)}
\]

Using with, we can then say Minimize rather than Optimization[Minimize].

Note that the exact location is \( t = 0.4 \), but that rounding error made during the operation of Minimize leads to only an approximation of the exact answer. This is typical.

According to the structure returned by this function, the location information is in the second element of the result.

We can find the location of particles A and B at \( t=0.4 \) by evaluating the expression \( p1 \) and \( p2 \) at that value of \( t \). Note that minLocation is a list consisting of an equation, which \( \text{eval} \) can use as its second argument.
\[ [0.6000000000, 0.400000000000000022] \] (17.67)

\[ \text{minimize}(\text{dist}(p1, p2), t = 0.2) \]

\[
\frac{1}{5} \sqrt{5} \quad \text{(17.68)}
\]

at 10 digits

\[ 0.4472135954 \] (17.69)

\[ T := t^2 + 5t^2 - 35t + 3 \]

\[
t^2 + 5t^2 - 35t + 3 \quad \text{(17.70)}
\]

\[ \text{plot}(T, t = 0.2) \]

This expression has a complicated expression for the exact value. We could try to approximate it using the right-click->approximate menu, or we could give it to the approximate minimizer instead.

\[ \text{minimize}(T, t = 0.2) \]

\[
3 + \text{RootOf}(7 \cdot Z^6 + 10 \cdot Z - 35, \text{index} = 1)^7 \\
+ 5 \cdot \text{RootOf}(7 \cdot Z^6 + 10 \cdot Z - 35, \text{index} = 1)^2 \\
- 35 \cdot \text{RootOf}(7 \cdot Z^6 + 10 \cdot Z - 35, \text{index} = 1) \]

at 5 digits

\[ -28.236 \] (17.72)

\[ \text{Optimization}[\text{Minimize}](T, t = 0.2) \]

\[ [-28.2356100235686540, \{t = 1.21771410298014682\}] \] (17.73)
### 17.8 Multivariate operations in Maple

`diff`, `minimize`, `solve`, and the rest can handle problems and situations where there are multiple variables involved. Approximation methods (e.g. `fsolve` or Optimization[Maximize] often take more prominence with multivariate problems because of the difficulty and/or expense of finding of succinct formulas that express the solution. We won't explore Maple's abilities to handle multivariate problems here, but they are there ready to be explored and used when you're ready for it.

#### Multivariate operations in Maple

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{\partial}{\partial x} \frac{\partial}{\partial x} \sin \left( \frac{x}{y} \right)^2 )</td>
<td>Compute the second partial derivative with respect to x.</td>
</tr>
<tr>
<td>( \frac{2 \cos \left( \frac{x}{y} \right)}{y^3} \left( \frac{x}{y^2} \right) + \frac{2 \sin \left( \frac{x}{y} \right)}{y^3} - \frac{2 \sin \left( \frac{x}{y} \right)}{y^2} \cos \left( \frac{x}{y} \right) )</td>
<td>(17.74)</td>
</tr>
<tr>
<td><code>diff(\sin(\frac{x}{y}), x, y)</code></td>
<td>The textual way of entering partial derivatives.</td>
</tr>
<tr>
<td><code>eqns := [x + 2 \cdot y = \frac{5}{4}, x^2 + y^2 = 1]</code></td>
<td>(17.76)</td>
</tr>
<tr>
<td><code>solve(eqns, \{x, y\})</code></td>
<td>We are interested in finding the points of intersection with the unit circle with center at (0,0), and the line defined by x+y=5. Doing it exactly gives an answer, but it's not easy to understand -- x and y can be either the positive or negative solution to a quadratic expression. Introducing a floating point number into the equation (1.25 instead 10/8) tells <code>solve</code> that it's all right to approximate the results.</td>
</tr>
<tr>
<td>( {x = -0.4916198487, y = 0.8708099244}, {x = 0.9916198487, y = 0.1291900756}, {x = 0.125, x^2 + y^2 = 1} )</td>
<td>(17.77)</td>
</tr>
<tr>
<td><code>eqnsf := [x + 2 \cdot y = 1.25, x^2 + y^2 = 1]</code></td>
<td>(17.78)</td>
</tr>
<tr>
<td><code>solve(eqnsf, \{x, y\})</code></td>
<td>(17.79)</td>
</tr>
<tr>
<td><code>expr := sqrt((x^2 + \sin(y + \frac{1}{10}))^2) + 5</code></td>
<td>We want to find the minimum value of an expression in two variables. <code>minimize</code> takes a long time to work and doesn't seem to come up with much. However, the approximate minimizer finds a value of 5.</td>
</tr>
<tr>
<td>( \sqrt{x^2 + \sin(y + \frac{1}{10})^2} + 5 )</td>
<td>(17.80)</td>
</tr>
<tr>
<td><code>minimize(expr, x=-1..1, y=-1..1)</code></td>
<td></td>
</tr>
<tr>
<td>Warning, computation interrupted</td>
<td></td>
</tr>
<tr>
<td><code>Optimization[Minimize](expr, x=-1..1, y=-1..1)</code></td>
<td></td>
</tr>
</tbody>
</table>
17.9 Chapter summary

<table>
<thead>
<tr>
<th>Symbolic differentiation using the expression palette, right-click, control-=.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( x^2 - 2x + 5 ) <strong>differentiate w.r.t.</strong> ( x )</td>
<td>( 2x - 2 )</td>
</tr>
<tr>
<td>Enter the expression, and then right-click (control-click on Macintosh) to bring up the calculation options. Select differentiate-( \rightarrow )x</td>
<td></td>
</tr>
<tr>
<td>( \sin(\omega \cdot t + 5)^2 ) <strong>differentiate w.r.t.</strong> ( t )</td>
<td>( 2\omega \cdot \sin(\omega \cdot t + 5) \cos(\omega \cdot t + 5) )</td>
</tr>
<tr>
<td><strong>differentiate w.r.t.</strong> ( \omega )</td>
<td>( 2\cos(\omega \cdot t + 5)^2 \omega^2 - 2\sin(\omega \cdot t + 5)^2 \omega^2 )</td>
</tr>
<tr>
<td><strong>simplify symbolic</strong></td>
<td>( 2\omega^2 (2\cos(\omega \cdot t + 5)^2 - 1) )</td>
</tr>
<tr>
<td>We can calculate the second derivative with respect to ( t ) of this expression by performing differentiation twice. We simplify the expression a bit by performing the operation simplify-( \rightarrow )symbolic.</td>
<td></td>
</tr>
<tr>
<td>( \frac{d}{d\omega} \sin(\omega \cdot t + 5)^2 = 2\sin(\omega \cdot t + 5) \cos(\omega \cdot t + 5) \cdot t )</td>
<td></td>
</tr>
<tr>
<td>If we want the document to display the mathematical notation for the derivative, we can select ( \frac{d}{dx} f ) from the expression palette and then fill in the slots ( f ) and ( x ). We can then get Maple to calculate the derivative by typing control-=.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbolic differentiation and evaluation of derivatives, textually</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example</strong></td>
<td><strong>Commentary</strong></td>
</tr>
<tr>
<td>( expr := x^2 - 2x + 5 )</td>
<td>Find the first derivative of the expression with respect to ( x ).</td>
</tr>
<tr>
<td>( x^2 - 2x + 5 )</td>
<td>Find the first derivative of the expression with respect to ( t ).</td>
</tr>
<tr>
<td>(17.82)</td>
<td>Find the second derivative of the expression with respect to ( t ). The result is the same as if we had taken the derivative of 1.2.4.</td>
</tr>
<tr>
<td>( diff(expr, x) )</td>
<td>( 2x - 2 )</td>
</tr>
<tr>
<td>(17.83)</td>
<td>( diff )'s second argument must be the variable of differentiation. Note that if the variable doesn't occur in the expression, the derivative (according to the mathematical definition) is zero. If you are surprised by getting a zero derivative, check that you are using the correct variable to differentiate with respect to.</td>
</tr>
<tr>
<td>( posExpr := \sin(\omega \cdot t + 5)^2 )</td>
<td></td>
</tr>
</tbody>
</table>
\[
\sin(\omega t + 5)^2
\]

\[
\text{diff(} \text{postExpr, t)}
\]
\[
2 \sin(\omega t + 5) \cos(\omega t + 5) \omega
\]  

(17.84) \hspace{2cm} \text{simplify can reduce the size of the expression, although factor or expand sometimes work better. Sometimes additional trigonometric identities need to be applied, through simplify(...., trig).}

\[
\text{diff(} \text{postExpr, t, t)}
\]
\[
2 \cos(\omega t + 5)^2 \omega^2 - 2 \sin(\omega t + 5)^2 \omega^2
\]  

(17.85) \hspace{2cm} \text{This is a way to compute } \frac{d}{dx} x^2 - 2 \cdot x + 5 \bigg|_{x=3}.

\[
\text{diff(} \text{expr, t)}
\]
\[
0
\]  

(17.86) \hspace{2cm} \text{This is a way to compute } \frac{d^2}{dt^2} \sin(\omega t + 5)^2 \bigg|_{t=47}. \text{ The result of evaluation does not have to be a number even if a numeric value is being supplied for one of the variables in the expression being evaluated.}

\[
\text{simplify(1.9.5)}
\]
\[
2 \omega^2 \left(2 \cos(\omega t + 5)^2 - 1\right)
\]  

(17.87) \hspace{2cm} \text{eval(1.9.2), } x = 3

\[
\text{eval(1.9.7), } t = 47.0
\]
\[
4
\]  

(17.88) \hspace{2cm} \text{eval(1.9.7), } t = 47.0

\[
2 \omega^2 \left(2 \cos(47.0 \omega + 5)^2 - 1\right)
\]  

(17.89)

**Textual version of limits**

<table>
<thead>
<tr>
<th>Example</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{limit(}1/(x-3)^2, x=3) \infty</td>
<td>The first argument is the expression that you wish to take the limit of. The second argument is an equation indicating the limit variable and the limiting value.</td>
</tr>
<tr>
<td>\text{limit(} \sin(x)/x, x=infinity) 0</td>
<td>You can use &quot;infinity&quot; or &quot;-infinity&quot; as the way of specifying ( \infty ) or (-\infty) textually without use of the palette.</td>
</tr>
<tr>
<td>\text{limit(}1/t, t=0, left) - \infty</td>
<td>The third optional third argument to limit can specify a one sided limit. This is the textual way of specifying ( \lim_{t \to 0^-} \frac{1}{t} ). There is no simple way of specifying this</td>
</tr>
<tr>
<td>\text{limit(}1/t, t=0, right) \infty</td>
<td>This is the textual way of specifying ( \lim_{t \to 0^+} \frac{1}{t} ).</td>
</tr>
</tbody>
</table>

**Minima and maxima**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact minimum or maximum value</td>
<td>( f := (x) \to 2 - 2 \sin(x) ) for ( x ) in ( \frac{-\pi}{4} \ldots \pi )</td>
</tr>
<tr>
<td>Solution</td>
<td>( f := (x) \to 2 - 2 \sin(x) )</td>
</tr>
<tr>
<td></td>
<td>( x \to 2 - 2 \sin(x) ) (17.95)</td>
</tr>
</tbody>
</table>
\[
\text{plot} \left( f, -\frac{\pi}{4}, \pi \right) \]
It's usually a good idea to plot the objective function to get some sense of where the extrema are. Remember that the function to be investigated should be continuous if you expect good results from calculus techniques.

\[
\text{smallestValue} := \text{minimize} \left( f(x), x = -\frac{\pi}{4}, \pi \right) \quad 0 \quad (17.96)
\]

\[
\text{largestValue} := \text{maximize} \left( f(x), x = -\frac{\pi}{4}, \pi \right) \quad 2 + \sqrt{2} \quad (17.97)
\]

Exact minimum or maximum value location

\[
\text{minResult} := \text{minimize} \left( f(x), x = -\frac{\pi}{4}, \pi, \text{location} \right) \\
0, \left\{ \left[ x = \frac{1}{2} \pi \right], 0 \right\} \quad (17.98)
\]

\[
\text{maxResult} := \text{maximize} \left( f(x), x = -\frac{\pi}{4}, \pi, \text{location} \right) \\
2 + \sqrt{2}, \left\{ \left[ x = -\frac{1}{4} \pi \right], 2 + \sqrt{2} \right\} \quad (17.99)
\]

Evaluation of an function at an extremum point

\[
g := (t) \rightarrow \sin(2 \cdot t) \quad \text{Evaluate } g \text{ at the location of the smallest value of } f.
\]

\[
t \rightarrow \sin(2t) \quad (17.100)
\]

\[
\text{minLocation} := \text{rhs} \left( \text{minResult}[2][1][1][1] \right)
\]
\[
g(\text{minLocation})
\]
\[
\begin{align*}
\frac{1}{2} \pi \\
0
\end{align*}
\] (17.101)

Evaluation of an expression at an extremum point
\[
expr := x^2 + \sin(x)
\]
\[
\begin{align*}
x^2 + \sin(x) & \\
\text{minEvalpt} := \text{minResult}[2][1][1] & = \left\{ x = \frac{1}{2} \pi \right\} \\
\text{eval}(expr, \text{minEvalpt}) & = \frac{1}{4} \pi^2 + 1
\end{align*}
\] (17.103)

Approximate minimum or maximum value and location
\[
\text{approxMinResult} := \text{Optimization}[\text{Minimize}] \left( f(x), x = -\frac{\pi}{4}, \pi \right)
\]
\[
[0, [x = 1.57079632679489478]]
\] (17.106)

\[
\text{approxMaxResult} := \text{Optimization}[\text{Maximize}] \left( f(x), x = -\frac{\pi}{4}, \pi \right)
\]
\[
[3.1415926535897932385, [x = -0.785398151694103208]]
\] (17.107)

\[
g(\text{rhs}(\text{approxMaxResult}[2][1]))
\]
\[
-1.000000000
\] (17.108)

\[
\text{eval}(expr, \text{approxMaxResult}[2])
\]
\[
-0.0902565162
\] (17.109)
Index

Symbols
! (exclamation point factorial), 12
# (program comment), 131
% missing in component programming, 236
%d, 126, 145
%f, 126, 145
(, 11
), 11
* (minus or dash), 12
-infinity, 244, 250
:, 105
: (colon), 123, 130
:=, 43
; (semi-colon), 130, 212
< (less than), 179
<= (less than or equals), 179
<default>, 156, 179, 184, 238
= (equals), 179
= (equation), 20
> (greater than), 179
>= (greater than or equals), 179
^ (caret), 12

A
abs(a), 73
accumulation, 182, 185
action
  naming of, 212
Action When Contents Change, 234
actual parameter, 59, 211
acute accent marks, 64
adding lists, 175
addition (+), 11
algebraic unknown, 33
Algol 68 (programming language), 156
Alpha, 73
alpha, 73
altitude, 115
animate, 120
  function, 113
animation of a moving object, 113
animation speed, 114
animation tool bar, 114
animations
  exporting to GIF format, 122
apostrophe, 95
apostrophes, 64
arc cosine, 25
arc sine, 25
arc tangent, 25
argument, 59
arguments
  of function, 77, 211
arithmetic, 10
    approximate, 26
calculator-style, 34
    exact, 13
    operations, 11
arrow, 81
assignment, 21, 43
  of a plot structure, 106
assistant
  Unit Converter, 98
asterisk, 11
autosaving, 19
average
  numbers in a list, 70
Axiom, 5

B
backquotes, 64
backspace key, 15
backup, 145, 184
backups
  retrieving, 19
ball, 115
binomial, 73
Blackberry, 70
blue, 109
box
  code edit, 234
brackets, 45
button, 229
  code edit box for, 234
by, 158

C
C, 195
C (the programming language), 5, 7, 129, 195
C# (the programming language), 195, 239
C++ (the programming language), 5, 195, 212
Canadian roots, 109
caret (power), 12
checkbox, 229
chemical concentration, 203
CHI, 73
chi, 73
circle, 109, 111, 117
  function, 109
circle-X, 9

263
clickable interface, 59
code edit box, 234
code edit region, 125
coding, 133
coffee, 222
colon, 105, 123, 130
Colorado, 46
colors
  plotting, 69
combo box, 229, 232
comma, 45
command—, 42
command-C (copy operation), 16, 48
command-e (execute region), 130
command-L, 42
command-R, 41, 42
command-T, 41, 42, 232
command-V (paste operation), 16, 48
command-X (cut operation), 16
comment
  program (#), 131
comments
  header, 131
  program, 125
completion
  command, 78
component name, 231
components
  Maple, 229
  need for unambiguous settings, 236
  programming, 231
composition
  of functions, 92, 93
computing
  algebraic, 1
  graphical, 1
  high performance, 3
  logical, 1
  numerical, 1
configuration
  combo box, 233
  plot area, 234
  text area, 233
configuration box, 232
constant
  symbolic, 33
containers
  data, 149
continuation condition, 178
control—, 34, 42, 242
control-C (copy operation), 16, 48
control-click, 20
control-e (execute region), 130
control-L, 42
control-R, 41, 42
control-T, 41, 42, 232
control-V (paste operation), 16, 48
control-X (cut operation), 16
copy a list into a sequence, 71, 75
copy a list into a set, 71, 75
copy a list into a string, 71, 75
copy (editing operation), 16
copy and paste, 48
cost
  of programming, 212
crash
  Maple, 19
curly braces, 45
currency, 94
CurveFitting, 105
cut (editing operation), 16
D
daisy chaining
  functions, 93, 212
dashed rectangle, 9
data structures, 149
days, 95
debugging, 212
decimal point, 33
delimiters in Maple: ( ) [ ] { }, 14
derivative, 242
dial, 229
diff, 241, 243, 245, 257
differentiate
  right-click menu item, 247
differentiate menu item, 241
differentiation, 241
  of expression palette, 242
digits, 26
display, 107, 119
  function, 107, 109
divide and conquer, 49
division (/), 11
Do, 234
do, 155
documentation
  need for, 2
DocumentTools, 234
dot
  centered, 13
double-quotes, 64
down the road
  just, 70
evaluate, 42
  at a point, 48
evaluate and display in-line (control-='), 34
evaluate at point, 46
evaluating
  preventing, 95
  state, 19
evaluating a formula, 111
Evaluating..., 184
evaluation, 72
  of expressions, 10
evaluation of a derivative expression, 241
execute region, 130
execute worksheet, 73
execution
  halting, 216
execution trace, 162, 216
exp(x), 219
exponential function (exp), 73
exponentiation, 16
expression, 149
expressions
  algebraic, 20
  from palette, 24
extrema, 245

F
factor, 20, 85
factorial (!), 12
false, 178
fence post error, 27
File->New->Document, 49
final result variable, 201, 202
final value, 156
find something similar, 49
fixed value variable, 201, 202
floating point number, 33
for, 156, 183
formal parameter, 211
formatted output (%a
  %f
    , %d, etc.), 127
Formatting
  output
    as currency, 94
Fortran, 5
fraction, 33
freeze frame
  animation, 114
from, 158
fruit flies, 80
fsolve, 85, 149
ft, 95
function
  idea of, 77
function definition, 83, 211
function definition alternative, 84
functions
  define, 80
designing from word problems, 91
invoking, 211
thousands of built-in, 78
fuzzy notions, 49

G
  gallon, 95
gallons
    Imperial, 70
GAMMA, 73
gamma, 73
  gathering variable, 201, 202
gigaflop, 3
global variable, 213
Google, 206
  graphical user interface, 229
  graphical user interface (GUI), 5
  Graphics Exchange Format (GIF), 122
greater than, 179
greater than or equals, 179
Greek letters, 73
green, 109
GUI, 229

H
  halting execution, 216
  header comments, 131
help
  on-line, 65, 74, 79
  quick, 9
    using on-line, 80
horsepower, 95
hour, 95
HTML, 237
  hypotenuse, 100

I
  i
    imaginary, 73
  imaginary number, 34, 37
icon programming, 237
icons, 237
Ideal Gas Law, 89
if, 191, 193
immutable, 149
inch, 95
index, 44, 109, 150, 178
index variable, 156
indices, 151, 153
infinite loop, 168
infinity, 33, 47, 73, 203, 244, 250
initial value, 156
Insert-> Table, 231
Insert->2-D Math, 41
Insert->Label, 42
Insert->Text, 41
integer, 33
interactive, 5
interpreted system, 73
interrupting Maple, 184
inverse trig functions, 25
invoking
  a function, 211

J
  Java, 5
  Java (the programming language), 195, 239
justification of formatted output, 126

K
  Kessel Run, 96
key, 150
kilometer, 96
kilopascals, 90
kilowatt, 95
km, 95

L
  label, 11, 42
  labels for axes
    plotting, 69
LabView, 237
languages
  mini-, 237
LeastSquares, 105
left arrow key, 15
left hand side (right-click menu), 27, 30
less than, 179
less than or equals, 179
lhs, 86, 149
limit
  left sided, 245
  one sided, 245
  right sided, 245
limited-precision number, 33
line
  function, 108, 109
linear algebra, 14, 175
Lisp, 206
list, 45, 62, 149
    add together items from, 70
count number of items in, 70
empty, 58, 149
one item from, 70
operations on, 70
part of, 70
ln, 34
local (function definition keyword), 212
local variable, 213, 221
local variables, 214
logarithm
    base 10, 73
    base b, 73
    base e, 25
logic error, 18, 27
loop, 155
loop body, 155
looping, 155
Los Alamos National Laboratory, 3

M
Macsyma, 5
map, 205, 211
Maple, 5, 195
    as a word processor, 41
document mode, 9
Maple 13, 5
Maplets, 239
mapreduce, 206
match parentheses, 49
math entry mode, 41
Mathematica, 5, 6, 195, 206
mathematical model, 1
mathematical models, 110
Matlab, 5, 38, 129, 195, 206
    Symbolic Toolkit, 6
matrices, 175
maxdigits, 17
maximize, 251, 252
mean, 178
meter, 95, 230
micrometer, 95
mile, 95
millimeter, 95
min, 204
mini-languages, 237
minimize, 251, 255, 257
minimum, 203, 204
minus, 11
minute, 95
mistake
    typographical

how to fix, 15
most recent value variable, 201, 203
mu, 73
multiple systems, 7
multiplication (*), 11
multivariate, 257

N
name
    of a component, 231
National Forest Service, 47
elegation (-), 12
negative infinity, 244
nops, 149, 150
not equals, 179
NULL, 125, 126, 250
NULL (the empty sequence), 58
numbers
    exact, 33, 34
    floating point, 33, 34
    limited precision, 34
    limited-precision, 33
    whole, 33
NXT, 237
NXT (the programming language), 195

O
Oak Ridge National Laboratory, 3
objective function, 245
Octave, 5
omega, 73
on-line documentation, 69
op, 149
opening worksheet, 73
Optimization, 255
Optimization[Maximize], 255
Optimization[Minimize], 255, 257
orange, 109
oscilloscope, 49

P
packages, 105
palette, 24
    Common Symbols, 33, 34, 64, 73
component, 230
Expression, 33, 61, 73, 77, 80, 82, 85, 176, 242
Greek, 73
    square root, 25
parameter, 48
    actual, 211
    formal, 211
parameterized plots, 111
parentheses, 11, 59
parse, 96
paste (editing operation), 16
patterns of looping, 181
Perl (the programming language), 195
petaflop, 3
Pi, 73
symbolic constant, 33, 36
pick names, 48
pitfalls
common
    in Maple, 77
playing
    animation, 114
plot, 86, 149
    builder, 65
colors, 64
labels, 64
options, 69
structure, 106
plot area, 230
plot3d, 258
PlotBuilder, 23
plots
    package, 107, 113
plotting, 21
plotting a function, 111
plottool, 108
plottools, 109
plus, 11
pop-up, 84
power (\(^\)), 12
precision
    fixed, 35
pressure, 100, 106
preventing
    evaluation, 95
previous value variable, 201, 203
print, 125
printf, 126, 237
proc (function definition keyword), 212
procedures
    as functions, 212
program code, 133
programming
    Maple components, 231
programming by icons, 237
Python, 5, 6, 38, 206
Python (the programming language), 195
Q
quotation, 95, 96

R
radian, 33
radio button, 229
range, 149
Ready, 184
ready
    Maple, 19
rectangle
    dashed (entry), 9
red, 109
red stop hand, 184
refactor
code, 93
regular expressions, 237
relational operators, 179
repeat n times, 181
replay
    animation, 114
retrieving
    a Maple session, 18
return, 217
return (function definition keyword), 212
return (key), 41
reuse
    of code, 212
    of software, 2
rhs, 86, 149
right click
    on a Maple component, 231
right hand side, 46
right hand side (of equation), 26
right hand side (right-click menu), 27, 30
right-click, 20, 241
    optimization menu item, 253
    right-click->solve, 37
    right-click->solve->numerically, 37
roles of variables in a loop, 201
    rotary dial, 229
    rounding, 35
S
Sage, 5
Saving
    a Maple document, 18
scaling=constrained, 111
scientific notation, 34
script
    turning into a function, 219
scripting, 47
search field
    for help, 65
select
ey entry, 44
item from a set, 26
select entry (right-click menu), 27, 30
semi-colon, 130, 212
seq, 174, 204
sequence, 15, 45, 100, 149
empty, 58, 149
set, 45, 150
empty, 58, 150
sheep, 46
shortcut, 207
side effect, 125
simplify, 241, 242
simplify(...
  trig), 242
simulation, 1
single-quotes, 64
slash, 11
slider, 229, 232
small steps, 49
Solo
  Han, 96
solve, 44, 59, 74, 85, 149
  a system of equations, 71
  by right-clicking or control-clicking, 20
  numerical, 26
solving an equation, 111
sprintf, 127
square root
  from palette, 25
standard deviation, 177
Star Wars, 96
state
  of variables in worksheet, 73
state display
  of Maple, 19
statement
  program, 130
stepper variable, 156
stepping, 182
stepping variable, 201, 202
stop
  execution of a function, 216
stop playing
  animation, 114
string, 126, 145
sublist, 70
subscription, 42
subtraction (-), 11
sum, 176, 178, 204
supercomputer, 3
symbol size, 117
syntax errors, 235

T
  table, 151, 232
    document, 231
tan, 34
Tau, 73
tau, 73
Tcl/Tk (the programming language), 239
temperature, 99, 106, 222
template
  textual, 78
teraflop, 3
terminating
  Maple computation, 184
test, 49
testing
  code, 212
text area, 232
text entry mode, 41
textual entry, 59
then, 191, 193
to, 155
tol, 178
tool bar
  animation, 114
Tools->Spellcheck, 42
trace
  execution, 139
teval, 216
triangle, 100
troubleshooting, 49
  function definitions, 83, 218, 221
  functions, 217
  if statements, 197
  loops, 167
Maple components, 235
tables, 153
textual input, 74
unit conversion, 96
true, 178
U
  unassign, 44, 49
  undefined, 244
underscore, 42
Unit Converter assistant, 98
units, 115
Units Calculator, 70
use, 234
Use of printf, 126, 145

V
  Vancouver stock exchange, 38
variable
global, 213
local, 213
variable of differentiation, 242
vectors, 175
version
  number, 144
vertical range, 60
Visual Basic (the programming language), 195
visualization, 1
volume gauge, 230

W
Warning
  ... is implicitly declared local, 221
  premature end of input, 218
    use <Shift> + <Enter> to avoid this message, , 218
  premature end of input use <Shift> + <Enter> to avoid
    this message, 156
whattype, 128
while, 179, 182, 183
widget, 229, 231
widget name, 231
with, 105, 234
worksheet
  appearance, 73
  state, 73

Y
yard, 95

Z
zeroes
  leading or trailing, 126