

7. Defining Functions and Constants

If you author $f(x)$, DfW will put f x on the screen because it thinks both x and f are variables. If you wish to define for example, $f(x) = x^2 + 3x + 1$, you could author $f(x) := x^2 + 3x + 1$. Note that we use $:=$ for assignments and $=$ for equations. Alternatively, you can choose DDeclare/Function and fill in the pop up form with $f(x)$ for the function name and $x^2 + 3x + 1$ for its value. Constants are treated just like functions except there are no arguments. To set $a = 2\pi$, you enter $a:=2\pi$. Then whenever you simplify an expression containing a , each occurrence of a is replaced by 2π .

To use variables with more than one letter or symbol, use the technique of declaring a variable. For example, if we enter $x1:=$, then $x1$ will be treated as the single variable $x1$. Alternatively, we change DfW to word input mode. We do this by choosing DDeclare/Algebra State/Input and then clicking the Word button. In this mode, variables can have several letters but then we must be careful with spaces. To get bx^2 we should enter $b x^2$ and not bx^2 , otherwise bx will be treated as a variable. Because of this it is best to use the previous method for multi-lettered variables.

8. Recursive Functions

In DfW, we use functional programming. One example is the use of the IF(test, true, false, unknown) construct. For instance, if n is a non-negative integer, entering $F(n):=IF(n = 0, 1, nF(n-1))$ will define $n!$ for n larger or equal to 0. The following will define the n th Fibonacci number. Enter, $G(n):=If(0 < n < 3, 1, G(n-1)+G(n-2))$ and $G(n)$ will give the n th Fibonacci number for integer $n > 0$. This

way of computing Fibonacci number is very costly in terms of storage. It is better to use the ITERATES function. For example, the second component in the vector $\text{FIB}(n)$ defined by $\text{FIB}(n) := \text{ITERATE}([\text{v SUB } 2, \text{v SUB } 1 + \text{v SUB } 2], \text{v}, [0, 1], n)$ also gives the n th Fibonacci number. This is a more efficient definition.

9. Defining Function in a Piecewise Manner

The $\text{IF}(\text{test}, \text{true}, \text{false}, \text{unknown})$ construct can be used to define function in a piecewise manner. For example: to define the function


$$f(x) = \begin{cases} 2x + 3, & x < 1 \\ x^2, & 1 \leq x \leq 2 \\ 5, & 2 < x \end{cases}$$

we author $F(x) := \text{IF}(x < 1, 2x + 3, \text{IF}(x \leq 2, x^2, 5))$. Notice how the use of the nested IF statement includes the condition in the definition of the function. You can have as many nested IF statements as you want depending on the function. When you differentiate, DfW will try to differentiate the functions it knows how but it will return the derivative in the IF format. It does not test for differentiability at the end points of the intervals in the definition. For instance, in the above example, it does not test for the differentiability of f at $x=1$ and $x=2$. DfW will give the impression in the form of the IF statement that it knows the derivative at $x=1$ and at $x=2$. You cannot use DfW to differentiate any piecewise defined functions at end points of the interval in the definition. Instead you should use the definition of derivative in terms of limit. DfW cannot do everything for you.

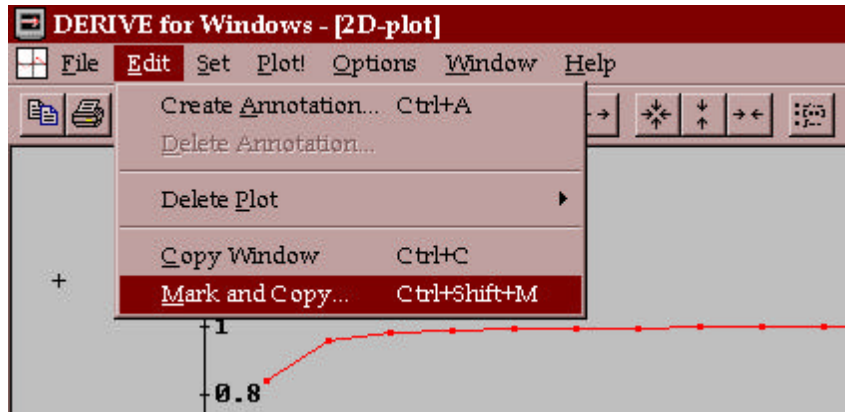
10. Vectors

Vectors are quite useful in DfW. Square brackets are used in DfW for vectors. Useful for plotting functions in the same plot window and for plotting points. We use the VECTOR functions to specify a vector by generating its components. The VECTOR function has 5 argument. The first is the defining expression of the component in terms of an index variable specified in the second argument. The third argument is the first value of the index variable; the fourth is the last value of the index variable. The last argument is optional; it is to be the step size of the index variable. The default is 1. This is very useful for generating a table of function values. For example, if you want to know the behaviour of $\frac{\sin(x)}{x}$ near $x = 0$, you plot the function VECTOR([n, nsin(1/n)], n, 1, 20) and you can see the points are getting closer and closer to 1.


11. Printing and Saving to a Diskette


You can save the expressions in an algebra window to either a floppy disk or the network hard drive. Unfortunately the plot windows are not saved. Click File/Save As and save to your desired destination. After the first save you can update the file by simply clicking on the  button. The plots can be put on the clipboard and saved as graphics files or pasted onto any document right in Windows. You might want to paste it into your report. You can select a rectangular region by clicking Edit/Mark and Copy from the menu bar in the plot window and then dragging



your mouse over the desired rectangular region. This region is then copied to the clipboard and you can paste it into any document opened in Windows.




You can recall the expressions saved previously by simply using File/Open or File/Load/Math option. The second method is used to add or append expressions to an existing algebra window. The files are saved with MTH extension. Before saving you work to a file or before printing your file you should erase unneeded entries and

clean up the file using the three buttons, . You can select several expressions by dragging the mouse pointer over them with the left button held down.

If you now click the  button, the highlighted expressions will be removed.

Clicking the  button will undo the last delete. You can move a block of highlighted expressions to a new location by just holding down the right mouse button and dragging the block to a new location. You can then renumber the expressions by clicking the  button.

You can print all the expressions in the algebra window by clicking the  button. To print a graph in the plot window do the same. The help button is very useful. When in doubt, always use the online help.