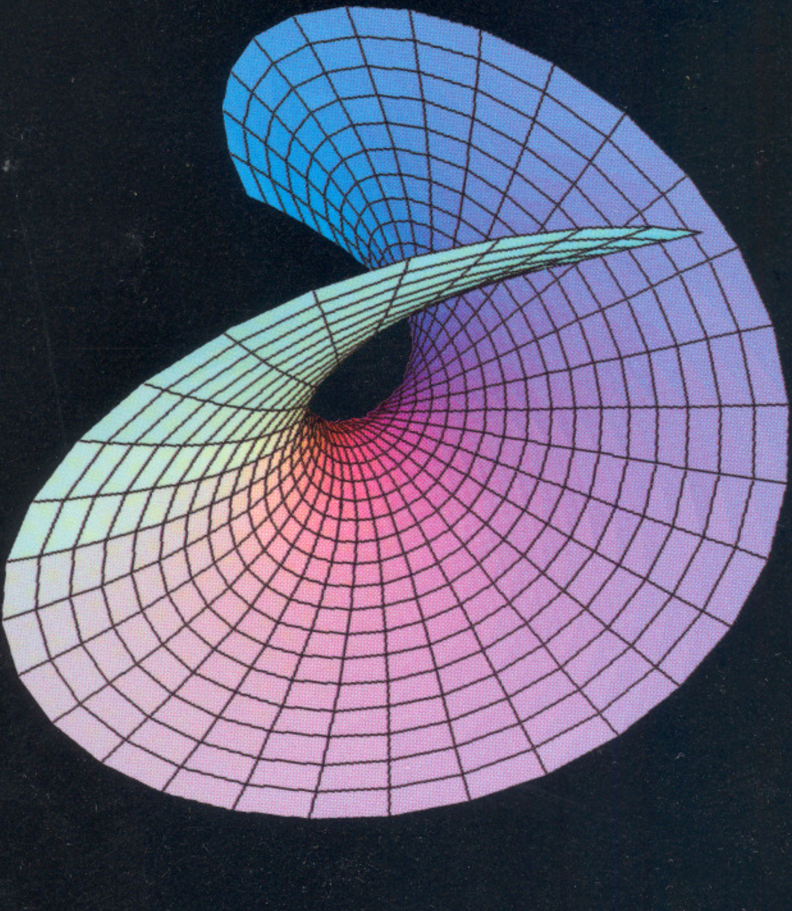
sun
microsystems

SunMathematica™

A System for Doing Mathematics by Computer

User's Guide



■ Contents

■ 3. Textual Output in *SunMathematica*

SunMathematica outputs mathematical expressions as text. When *SunMathematica* outputs a long expression, it must break the expression into several lines. To do this, *SunMathematica* must know how wide your screen or window is.

Unless your initialization file specifies otherwise, *SunMathematica* starts by assuming a width of 78 characters. You can find out how wide your screen or window actually is by executing the SunOS command `stty all`. (When you are running over a network using `rsh`, this command may sometimes give you incorrect results.)

`ResetMedium[PageWidth->`*n*`]`	assume a screen or window width of *n* characters

Setting your line width.

With this line width, the expression is broken onto two lines.

```
In[3]:= t = Expand[ (1 + x) ^ 10 ]

                            2         3         4         5
Out[3]= 1 + 10 x + 45 x  + 120 x  + 210 x  + 252 x  +

             6         7        8        9    10
     210 x  + 120 x  + 45 x  + 10 x  + x
```

This tells *SunMathematica* that the page is 90 characters wide.

```
In[4]:= ResetMedium[PageWidth->90]
```

If the width you tell *SunMathematica* to use is greater than the actual width of your screen or window, output will wrap around and be illegible.

```
In[5]:= t
                            2         3         4         5
6         7        8        9
Out[5]= 1 + 10 x + 45 x  + 120 x  + 210 x  + 252 x  + 210
x  + 120 x  + 45 x  + 10 x  +

      10
     x
```

This specifies a width of 40 characters.

```
In[6]:= ResetMedium[PageWidth->40]
```

Now the output lines are shorter.

```
In[7]:= t

                            2         3
Out[7]= 1 + 10 x + 45 x  + 120 x  +

             4         5         6
     210 x  + 252 x  + 210 x  +

             7        8        9    10
     120 x  + 45 x  + 10 x  + x
```

4. Graphics Output in *SunMathematica*

4.1 Basic Graphics

When *SunMathematica* starts up, it tries to work out what display manager or window system, if any, you are using. Then it sets up the appropriate graphics, and prints a line such as `--SunView graphics installed--` to let you know what it has done.

If you are using *SunMathematica* on the console of your Sun Workstation, under SunView, NeWS or the X Window System, then *SunMathematica* creates a new window for each piece of graphics you generate.

You can manipulate the new window using standard menus. (These are typically accessed by clicking the right mouse button in the frame of the window.) If you resize the window, the graphics within it are resized accordingly. You can also close the window, so it appears on your screen as an icon.

Note: Particularly under SunView, it is very important that you kill windows (using the `quit` menu item) when you have finished with them. There is a limit on the total number of windows that you may have at one time.

4.2 Graphics Hardcopy

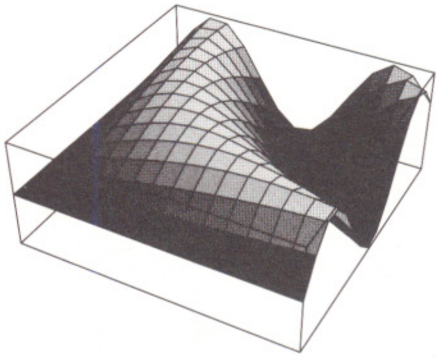
`LaserPrint[`*graphics*`]`	send graphics to a laser printer, using the `lpr` command

Getting graphics hardcopy.

If you have a PostScript™-based hardcopy device (such as a LaserWriter™), then you can get graphics hardcopy directly from *SunMathematica*.

Here is a plot produced by *Mathematica*. On a Sun Workstation, the picture appears in a separate window.

```
In[8]:= Plot3D[Sin[x y], {x, 0, 3}, {y, 0, 3}]
```



This generates a hardcopy of the plot.

```
In[9]:= LaserPrint[%]
Out[9]= LaserPrint[-SurfaceGraphics-]
```

There are many options for producing graphics hardcopy. Section 5.3 of the *SunMathematica Installation Manual* discusses how to redefine the `LaserPrint` function to suit the needs of a particular application or a particular system configuration.

□ Possible Problems

Here are a couple of problems that you may encounter in getting graphics hardcopy from *SunMathematica*.

`LaserPrint` sends output to the wrong printer.

> You need to modify the definition of `LaserPrint`. See Section 5.3 of the *SunMathematica Installation Manual*.

Graphics makes the printer run for a long time.

> There is probably nothing wrong – LaserWriter output is considerably slower than output on the Sun Workstation screen. If the LaserWriter is working, its Ready/In Use light should be flashing.

■ 4.3 Animation

SunMathematica allows you to produce "movies" by displaying a sequence of graphical images from *SunMathematica* in rapid succession. (At present, this facility is available only under the SunView window system.)

The *SunMathematica* package `Animation.m` contains definitions for making animated sequences of images. The package includes versions of the plotting functions in Section 1.8 of The *Mathematica* Book, generalized to include an additional time parameter.

`Animate[{`g_1`, `g_2`, ...}]`	produce a movie of the sequence of *SunMathematica* graphics objects g_i
`Movie[`*command*`, {`*t*`, `*tmin*`, `*tmax*`, `*t*`}]`	iterate *command* to produce a a movie
`MoviePlot[`*f*`, {`*x*`, `*xmin*`, `*xmax*`}, {`*t*`, `*tmin*`, `*tmax*`}]`	generate a sequence of two-dimensional plots as a function of time parameter *t*
`MovieParametricPlot[{`*f*`, `*g*`}, {`*x*`, `*xmin*`, `*xmax*`}, {`*t*`, `*tmin*`, `*tmax*`}]`	generate a sequence of two-dimensional parametric plots as a function of time parameter *t*
`MoviePlot3D[`*f*`, {`*x*`, `*xmin*`, `*xmax*`}, {`*y*`, `*ymin*`, `*ymax*`}, {`*t*`, `*tmin*`, `*tmax*`}]`	generate a sequence of three-dimensional plots as a function of time parameter *t*
`MovieDensityPlot[`*f*`, {`*x*`, `*xmin*`, `*xmax*`}, {`*y*`, `*ymin*`, `*ymax*`}, {`*t*`, `*tmin*`, `*tmax*`}]`	generate a sequence of density plots as a function of time parameter *t*
`MovieContourPlot[`*f*`, {`*x*`, `*xmin*`, `*xmax*`}, {`*y*`, `*ymin*`, `*ymax*`}, {`*t*`, `*tmin*`, `*tmax*`}]`	generate a sequence of contour plots as a function of time parameter *t*
`SpinShow[`*graphics*`]`	generate a movie showing a rotating three-dimensional object

Animated graphics from the package `Animation.m`.

You can include options for the plotting functions in `Animation.m` just like those for their non-animated counterparts discussed in Section 1.8 of The *Mathematica* Book.

You will often need to specify a definite value for the `PlotRange` option (see page 109 of The *Mathematica* Book). With the default setting `PlotRange->Automatic`, different plots in the sequence for a movie may be scaled differently.

When you evaluate a function like `MoviePlot`, *SunMathematica* produces a sequence of frames, then you shows you them as a movie. *SunMathematica* pops up a window to show you each frame it generates. When all the frames have been generated, *SunMathematica* creates a "movie window", which shows the resulting movie.

You can move the movie window around on the screen, but you cannot resize it. (The movie window works by displaying a sequence of bitmaps.) The movie will go on "playing" until you quit the movie window.

The movie window has various controls, which you can manipulate with the left mouse button. It has a slider, which allows you to set the speed of the movie in frames/second. If you put the slider all the way to the right, the movie will run as fast as possible. There are also buttons which allow you to choose whether to view the movie forwards, backwards, or alternately forwards and backwards ("loop").

□ Possible Problems

Here are some possible problems you may encounter in producing movies from *SunMathematica*.

SunOS prints `file system full`.

> *SunMathematica* uses `/tmp` files to store the raster files that it displays in a movie. You can try deleting unnecessary other files in `/tmp`. If this does not work, you could make a movie with fewer frames, or request more disk space for your `/tmp` directory from your system administrator.

The movie does not run at a uniform rate.

> There is probably not enough physical memory available on your Sun Workstation to store all the frames of the movie. Try killing off some other processes or making a shorter movie. This problem may be particularly severe when you use color.

The motion in the movie is jerky.

> You may simply need to include more frames. You should also check that the same plotting scale is used for all frames. To ensure this you may need to give an explicit value for the option `PlotRange`, because the default scaling algorithm may choose a different value for each frame.

5. Editing Input for *SunMathematica*

5.1 Introduction

There is no text editor built into *SunMathematica*. Different people like to use different editors, and we do not force everyone to learn and use a new editor. Instead, *SunMathematica* allows you to hook into the editor you already use.

Maintain two windows: one for running your editor, and one for *SunMathematica*.

Run your editor from inside *SunMathematica*.

Two ways to edit input for *SunMathematica*.

5.2 Editing in a Separate Window

When you are setting up complicated *SunMathematica* definitions, you will often find it convenient to build up the definitions in a file, reading in each revision of the definitions as you produce it.

A convenient way to do this on a Sun Workstation is to maintain two windows: one to edit your file, and one to run *SunMathematica*. Each time you get your definitions to a stage where you want to try them out, you can write out the file using your editor, and then read the latest version of the file into *SunMathematica*.

<<*filename*	read a file into *Mathematica*

Reading a file of definitions into *Mathematica*.

A Possible Problem

Revised definitions do not overwrite older ones.

In certain complicated cases, described in Section 2.2 of The *Mathematica* Book, *SunMathematica* may not be able to tell whether a new definition for a function (named f, for example) should overwrite an older one you gave. In such cases, *SunMathematica* keeps both definitions, putting the newer one later. If you want to make sure that *SunMathematica* removes the old definitions, you can put `Clear[`*f*`]` at the top of your file.

5.3 Editing from Within *SunMathematica*

`Edit[ ]`	start your editor with an empty buffer; read in the expression you write out from your editor
`Edit[`*expr*`]`	start your editor, with the `InputForm` of *expr* in the buffer
`Edit[`$expr_1$`,` $expr_1$`, ...]`	start your editor, with the `InputForm` for a sequence of expressions in the buffer
`EditIn[ ]`	start your editor, with the text of your latest input line in the buffer
`EditIn[`*n*`]`	start your editor, with the text of `In[`*n*`]` in the buffer
`EditIn[-`*n*`]`	start your editor, with the text of the input line *n* back in the buffer
`EditDef[`f_1`,` f_2`, ...]`	start your editor, with the definitions of symbols f_1, . . . in the buffer
`Recall[ ]`	display the `InputForm` of your last input line
`Recall[`n_1`,` n_2`, ...]`	display the input expressions `In[`n_1`]`, `In[`n_2`]`, . . .

Calling your editor from within *SunMathematica*.

When you evaluate the *SunMathematica* function `Edit[`*expr*`]`, *SunMathematica* will start up your editor, placing the input form of *expr* in the edit buffer. (*SunMathematica* uses the shell environment variable `EDITOR` to tell which editor to call. The default is `vi`.)

You can now use your editor to edit what is in the buffer. When you have finished, you should exit your editor, saving what you have done. (If you use `vi`, for example, you should use `:wq` to exit.)

SunMathematica then reads back the contents of your editor buffer to use as input. Note that *SunMathematica* assigns each separate expression in the buffer to a different *SunMathematica* input line `In[`*n*`]`. The results generated from these input lines are assigned to the corresponding `Out[`*n*`]`.

`Edit` works by using a temporary file, with a name of the form `/tmp/`*uniquename*. You may see this filename if your editor shows you what file it is editing.

Note that if you enter a line that contains a syntax error directly into *SunMathematica*, without using `Edit`, you will not be able to recover your input. If you are trying to enter a long expression, it is therefore often a good idea to use `Edit[ ]`. If you make a mistake, you can then simply use your editor to correct it. `Edit` will let you re-edit a particular line of input as many times as are needed to get a correct form.

Note that if you call `EditDef` on a `Protected` symbol (such as a built-in function, as described in Section 2.2.18 of The *Mathematica* Book), then *SunMathematica* puts the definition into your edit buffer, enclosing it in a comment. You can make a new definition take effect by removing the `(*` and `*)`, which delimit the comment, and then exiting your editor. Of course, you can never modify the value of a `Locked` symbol.

□ Possible Problems

Here are some problems you may encounter when you edit from within *SunMathematica*.

Your editor behaves strangely when called from `Edit`.

This is probably not the fault of `Edit`. Try experimenting by suspending *SunMathematica* with CONTROL-Z, and see whether your editor works as you expect in the window you are using to run *SunMathematica*.

Your editor does not work at all when called from `Edit`.

Many editors do not work from inside `cmdtool` windows. You may have to start *SunMathematica* in a `shelltool` window in order to have `Edit` work properly.

Your editor clears the screen when called from `Edit`.

If you use an editor that clears the screen on startup, such as `vi`, this will always happen.