Classroom Tips and Techniques: The Explore Command in Maple 18

Dave Linder
Software Architect, Mathematical Software
Maplesoft

Introduction

When I worked in older versions of Maple and wanted to plot something which depended upon a real-valued parameter I would often get a feel for how the plot depended upon that parameter by producing an animation. Sometimes I would produce a few different plots by substituting particular values for the parameter. The technique was to create an animation from the plots, where each frame represented a different value for the changing parameter. The command I would primarily go to in order to accomplish this was plots[animate].

This often worked out well. But there were some situations where it wasn't quite what I wanted. Sometimes I would have several parameters to make sense of, sometimes some parameters would be discrete valued and setting up the animation was more involved, sometimes I wanted so many frames that together they would use more memory than I was willing to accept, and sometimes I simply did not want to wait while all the frames were computed before I could start watching.

Now I turn first to the Explore command, which gives me an interactive experience with my parameter-dependent plots, but which also handles the above situations. It also lets me explore mathematical expressions as well as plots.

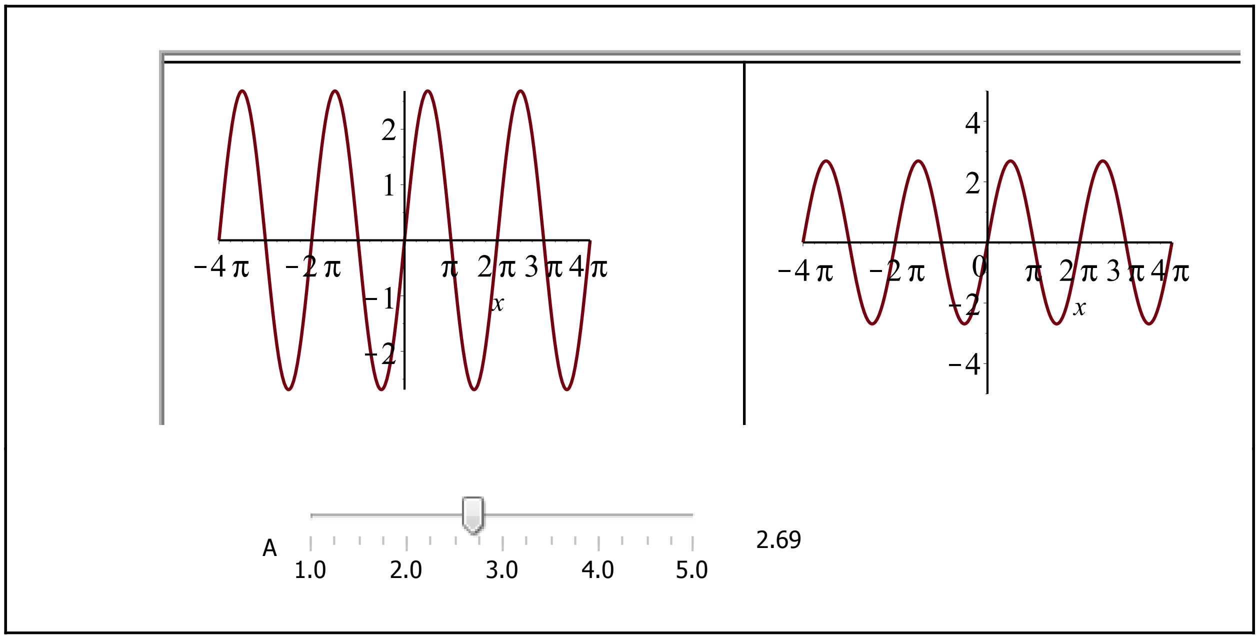
The Explore functionality has been significantly enhanced in Maple 18, with improvements to both the command form and the interactive Exploration Assistant. There is a much expanded Example worksheet which illustrates its use. In this Tips and Techniques article, I will focus on some key usage points of using the Explore command with plots, including explorations based on simple Maple plots as well as user-defined plotting procedures.

The plot view

Let's start with a basic example and usage tip. When viewing a sequence of plots generated by varying a parameter, the range or vertical scale of the plots can change significantly. If the the **view** option is not supplied to enforce a common vertical scale then plots' features may appear somewhat static while the plots' vertical axis range changes visually. This might well not be what's wanted. Often we wish to let the plotted feature be dynamic while viewed over a static range.So the first tip is to specify the **view** of the plots.

The exploration below contains an Array of two plots which depend upon the parameter A, and the difference between them is that only the plots on the right have a forced, common vertical view. In contrast, the curve in the plots on the left appear to stay fixed while the tickmarks on the y-axis move. Often it is the varying behavior in the plots on the right that is wanted.

$$Explore\Big(\, plots[display]\big(\, Array\big(\,[\, plot(A \sin(x), x=-4\,\pi..\,4\,\pi),\; plot(A \sin(x), x=-4\,\pi..\,4\,\pi, view=-5\,..5)\;]\big)\;\big),\; parameters = [A = 1.0\;..\;5.0\,],\, size = [300, 300\,]\Big)$$



Exploration of a function call

As we've seen above, explorations of plots require something that evaluates to a plot (or Array of plots) as the return value (ie. what would normally be the output). That *something* could be a call to a command such as plot, plot3d, or one of the many other stock commands in Maple that return 2D or 3D plots.

But the something could also involve a call to a user-defined procedure. If you have defined procedure `F` such that the function call `F(a,b,c)` returns a plot for particular types of arguments a, b, and c then that function call can just as easily be explored. Here, a, b, and c are the parameters.This is more general usage than simply calling a plotting command. The procedure can also do preliminary computations which depend upon the parameters.

In our second example, the technique will be to create a procedure with three parameters as its arguments. The first and second parameters represent the pair of coordinates that define a point in the x-y plane, while the third parameter takes on only positive integers and represents a number of iterations of a computation. There are too many pieces of varying information for a sequence of plots generated by **plots[animate]** to illustrate easily.

The Explore command allows us to easily construct an interface where the parameters may be animated all together, or individually adjusted interactively. Or we can choose a mix of such interactive modes, and that choice can be changed even after the fact. We are not forced to choose a single mode of interaction in advance.

The computation we are going to perform is known as tetration, and the example is inspired directly by the Complex Power Towers entry of the excellent Walking Randomly blog of Mike Croucher.The key idea is that of repeated exponentiation by a given complex value $z = x + \mathrm{i}\, y$. For a given point z in the complex plane the n-th iterate is the value of z exponentiated with itself n times. For any given input point z (two input parameters x and y actually) we will compute and make a point plot of the first n iterates. Some starting points z will generate a sequence of interates that converge, and some will not, and some will generate a sequence that eventually cycles.The boundaries of the regions where such behavior changes forms an interesting fractal, and you may read more about that at www.tetration.org.

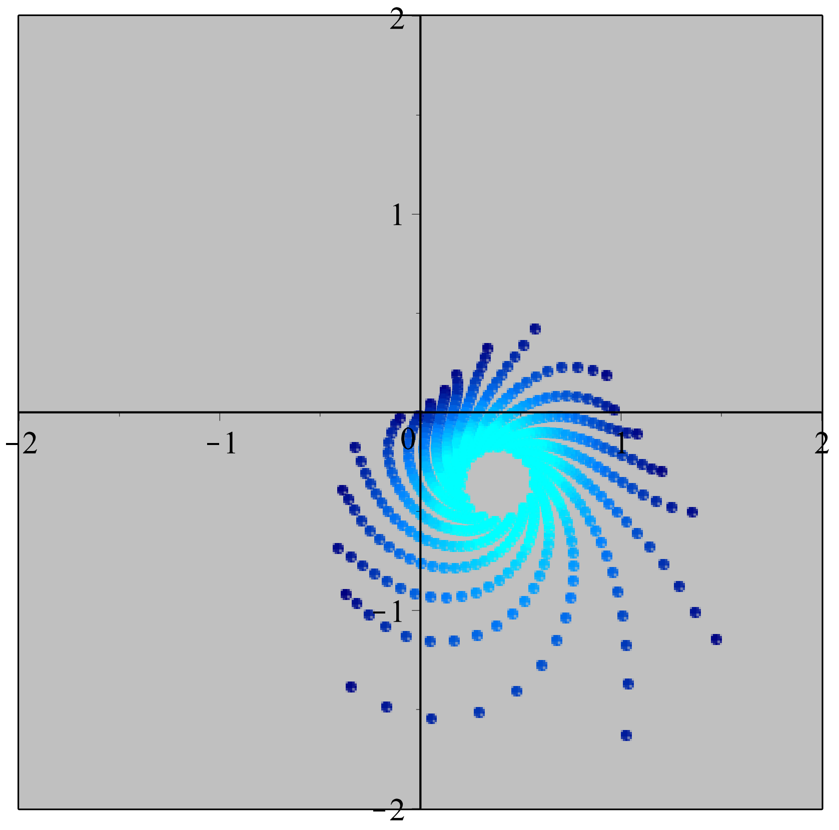
We won't go into further details of the computation because that's mostly incidental here. The idea central to this article is that we have a procedure F which has parameters and which returns a plot. And we wish to interactively explore the results of varying those parameters.

Tip: By including the definitions of procedures and variable in the Statup Code region of the worksheet (or in a Document Block set to autoexecute) the exploration will be ready to use upon re-opening the Document.

```
Definition of procedure F.
```

Let's call F at a particular set of three values, and see the kind of plot it can produce. The plotted points below are the first 500 iterates of the repeated exponentiation of the complex point -0.37 -0.92 I where the points start out shaded dark blue and gradually become shaded lighter blue.

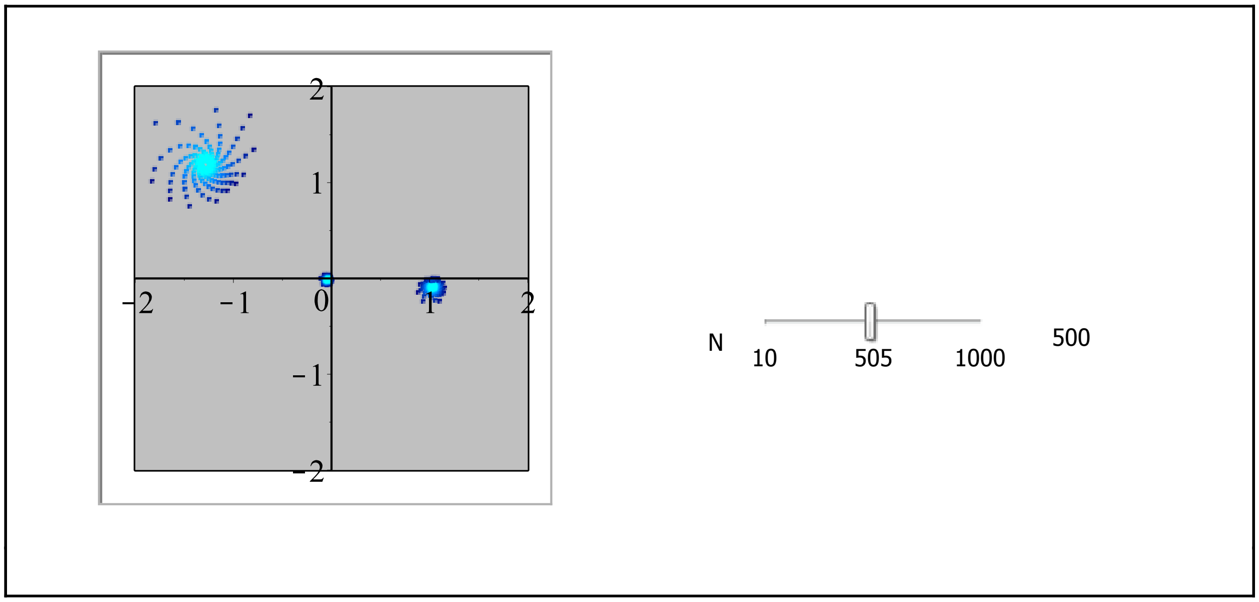
$F(\,-0.37,\,-0.92,\,500\,)$



Now we will create our first exploration of the plots produced by procedure F.

Tip: By specifying parameters x and y together as a marker we can interact with them by either single-clicking or click-and-dragging the cross symbol that appears in the 2D plot.

$$\mathit{Explore}(\ F(\ x, y, N\ ),\ \mathit{parameters} = [x = -2.0\,..2.0,\ y = -2.0\,..2.0,\ N = 10\,..1000],\ \mathit{initialvalues} = [x = -0.973,\ y = 0.988,\ N = 500\ ],\ \mathit{markers} = [\,[x, y]\,],\ \mathit{placement} = \mathit{right})$$

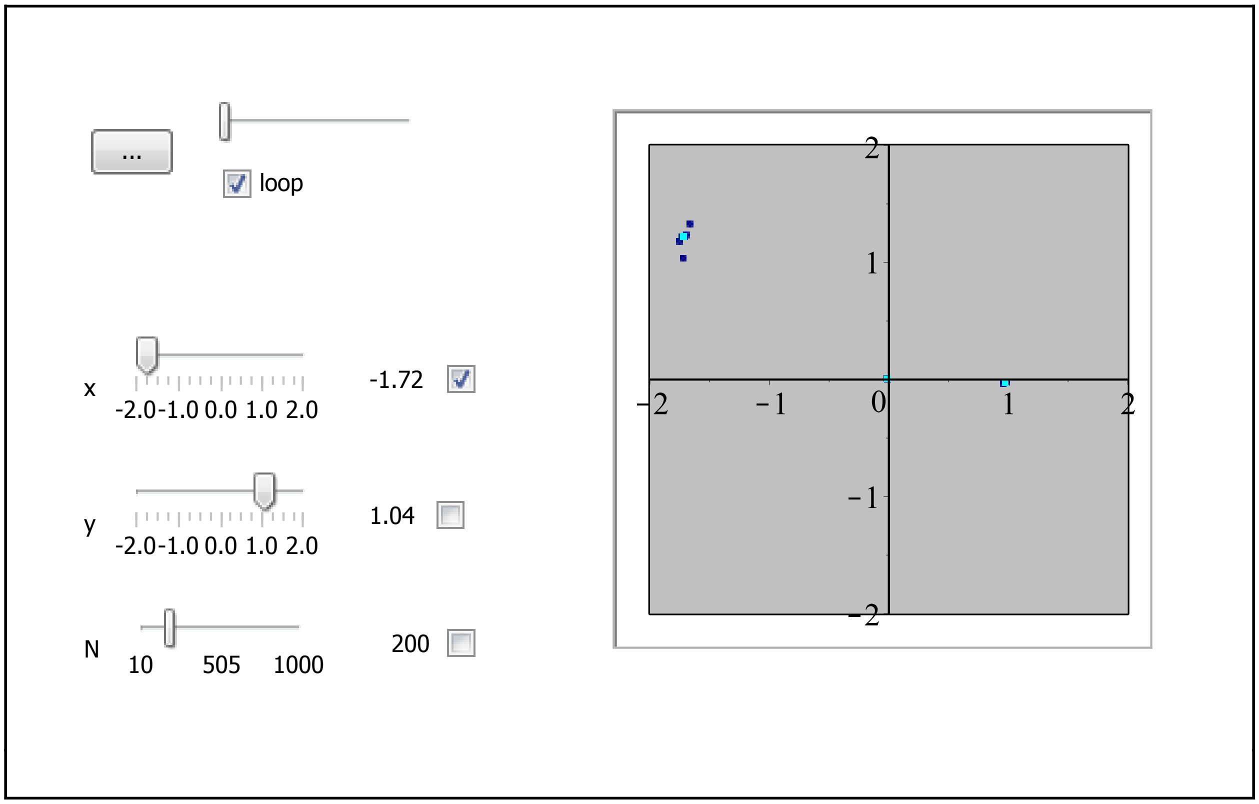


Now we'll animate the above exploration, and have the x parameter value be animated by default.

Tip: By using the **animate** suboption within the **parameters** option to the **Explore** command we can specify which parameters are initially set ready to play, and which have the choice of subsequently being toggled to play.

In the embedded exploration we can press the Play button, or manually interact with the Sliders, or Click or Drag the x-y marker symbol.

```
Explore(  F( x, y, N ),
           parameters = [ [x =-2.0 ..2.0, animate = true],
                          [y =-2.0 ..2.0, animate = false],
                          [N = 10 ..1000, animate = false] ],
          initialvalues = [ y = 0.988,  N = 200  ],
          markers = [ [x, y] ], showmarkercontrols, placement = left, loop = true  )
```



[✓] loop

x -1.72 [✓]

y 1.04 [ ]

N 200 [ ]

What makes that last example particularly effective is that the Sliders which control the values for parameters y and N can be adjusted even while the animation is running. Or the checkboxes, which denote which parameters will "play", can be toggled on and off, while playing or looping.

At this point we may even have forgotten to notice that the whole exploration plays right way -- we don't have to wait until the full set of frames is computed. Instead, we see the first frame as soon as it is computed, and similarly for all subsequent frames.

I'm having a lot of fun building my explorations, and I hope that you do too.