Honest Plotting, Global Extrema, and Interval Arithmetic

Richard Fateman
University of California at Berkeley

Abstract

A computer program to honestly plot curves $y = f(x)$ must locate maxima and minima in the domain of the graph. To do so it may have to solve a classic problem in computation – global optimization. Reducing an easy problem to a hard one is usually not an advantage, but in fact there is a route to solving both problems if the function can be evaluated using interval arithmetic. Since some computer algebra systems supply a version of interval arithmetic, it seems we have the ingredients for a solution.

In this paper we address a particular problem how to compute and display "honest" graphs of 2-D mathematical curves. By "honest" we mean that no significant features (such as the location of poles, the values at maxima or minima, or the behavior of a curve at asymptotes) are misrepresented. By "mathematical" we mean curves like those generally needed in scientific disciplines where functions are represented by composition of common mathematical operations: rational operations (+, −, *, /), exponential and log, trigonometric functions as well as continuous and differentiable functions from applied mathematics.

1 Introduction

Several popular computer systems provide adaptive 2-dimensional plotting. That is they can plot a curve $y = f(x)$ over a range $x = [a, b]$ with "automatically chosen" abscissa points that are not necessarily evenly spaced between a and b. Near locations at which $f(x)$ apparently changes rapidly, more points will be chosen so as to have the plot reflect more accurately the changes in the curve in that region. (This assumes that f is available in a form that can be evaluated for any $x \in [a, b]$.) Unfortunately, this does not guarantee an injudicious choice of points. Some programs choose more points than needed to understand part of a curve while choosing insufficiently many points elsewhere to identify singularities, maxima, minima, or other special behavior.

Fortunately problematic types of curves for 2-D plotting are easily constructed with no more complicated constructions than rational functions combined with some periodic but unbounded functions like the tangent. Many of our examples will come from this class. (See the note in the section on "impossible" adversary functions below for problems we can't handle.).

2 Difficulties in Plotting

The primary source of difficulty in plotting is the possibility that, because of insufficient sampling, a plot will not exhibit some qualitatively significant phenomenon. For example, given $y = f(x)$, a narrow peak or valley in the region of some $x = x_0$ might be missed or its value $f(x_0)$ truncated. Another common difficulty is a situation where $f(x)$ becomes infinite or "very large" compared to most of the points. It is plausible (or in the case of an actual infinite value, mandatory) to allow a curve to go "off the screen". In general there is a tension between excluding too much of the curve, and on the other hand including too many outliers, in which case interesting small-scale phenomena might be obscured.

3 Directions for solutions

An approach to plotting which in theory solves the problem of truncated extrema, as well as other errors caused by insufficient sampling, is to use interval arithmetic (for a general reference see, for example, Alefeld and Herzberger [2]).

On the other hand, the usual technique for plotting a function $y = f(x)$ mapping real numbers to real numbers is to choose n equally-spaced points between the finite real numbers a and b such that $x_0 = a$ and $x_{n-1} = b$. A plot is produced by connecting the points $(x_i, f(x_i))$ to $(x_{i+1}, f(x_{i+1}))$ for $i = 0$ to $n - 1$, with straight lines. An adaptive plotting program would use some technique for adding points between the original ones to more accurately portray rapidly changing functions.

[0]

The interval technique requires considering line *segments* rather than points. It plots a sequence of rectangles constructed as follows: Given a segment from x_i to x_{i+1} we construct a rectangle from (x_i, y_i) to (x_{i+1}, y_{i+1}) where the interval y-value $[y_i, y_{i+1}]$, is guaranteed to include all possible $f(\xi)$ for $\xi \in [x_i, x_{i+1}]$. The guarantee that the rectangle does in fact enclose the function requires a correct implementation of interval arithmetic[1] In the case of a function such as $y = \sin(1/x)$ near 0, a function which rapidly oscillates between -1 and 1, this construction will provide a bounding envelope of height 2 as $x \to 0$. This construction can result in a "bumpy" display of a collection of boxes rather than a line (see the figure 1 for a preview of this). One simple remedy is to use smaller subdivisions. Another remedy is to re-draw the curve with line segments, after using interval techniques to locate sensitive areas.

Other than the bumps, there are a few difficulties with interval arithmetic: it may be overly pessimistic as well as relatively slow (The basic arithmetic operations performed on intervals requires at least about 4 times the arithmetic and twice the space of regular computation, but in some circumstances, especially where it is being used in a symbolic context, it is likely that its use will be far more expensive.) It also requires that the functions being plotted be evaluable via intervals. In some cases this may mean the function has to be defined by different formulas (see section 4); Yet this does in fact provide a guarantee not to mislead, and the extra cost, especially on a high-speed workstation, may be, in fact, quite acceptable. As a side-effect of plotting, maxima and minima can be identified. In general, promoters of interval arithmetic have proffered this approach to solving global optimization problems. It is definitely a solution at a cost.

For some 2-D curves to be honestly displayed, and for truly global extrema to be identified, it may be appropriate to try to allow the endpoints a and/or b to be infinite. An accurate portrayal then requires the plotting program to bring points at or near infinity into finite regions by functional mappings. The Maple [5] program has a partial approach to this for plotting an infinite x-range; we show the same kind of approach works for an infinite y-range.

Other proposals are briefly indicated in section 6. For a further discussion of "honesty" in plotting, see Ager and Maas [1] and Bedner and Fateman [3].

[1] It would be a mistake to assume Mathematica or other systems are necessarily correct in this regard.

4 Plotting With Interval Arithmetic

The simplest technique to write a program for graphing interval rectangles seemed to be to modify a plotting program in an existing computer algebra system. *Mathematica* seemed to include a number of tools we needed including the ability to evaluate functions at interval "points" and to plot rectangles.

We discovered some problems, however. While *Mathematica*'s graphics programs will often produce good default plots, almost any attempt to make a modification that leads outside of *Mathematica's* built-in toolkit of options requires substantial work. Ultimately we had to make numerous compromises on appearance and speed.

Stumbling blocks included:

- Errors in design and implementation of interval arithmetic in *Mathematica*. Dealing with complex numbers, complex infinity and indeterminate quantities must be considered, since part of the point of this exercise is to deal with singularities and functions with undefined regions. Experimentation with *Mathematica* 2.0's `RealInterval` data type except in the most naive fashion, met with unfortunate failures to conserve correctness [3].

- *Mathematica*'s default plotting algorithm chooses to rescale, omitting outlier objects. Some of these objects must be forced at least partly into the frame[2].

- Unbounded intervals are sometimes generated. These are not "proper" graphics objects and therefore they must be truncated (clipped) before being plotted. Error messages must be intercepted.

- Additional problems and some solutions are listed in a more complete report on this effort [3]. Some of them are particular to *Mathematica*'s conventions, but others may be expected in *any* graphics package using PostScript.

Here we report on two functions that approximate the default built-in ones by having lower-case names (`plot, parametricPlot`) and have similar usage. Since we did not include choosing points adaptively in these experimental version, the user will probably want to consider setting the `PlotPoints` option far

[2] Often, *Mathematica's* automatic choices for "off-screen" ranges must be over-ridden in particular ways. Chosing between `PlotRange->All` and `PlotRange-> Automatic` is difficult. Plotting *all* the points is impossible in cases where a curve goes to infinity. Yet the "automatic" default choice leads to a certain mystery—determining which points *Mathematica* leaves out requires either access to proprietary algorithms or a duplication of a fair amount of computation.

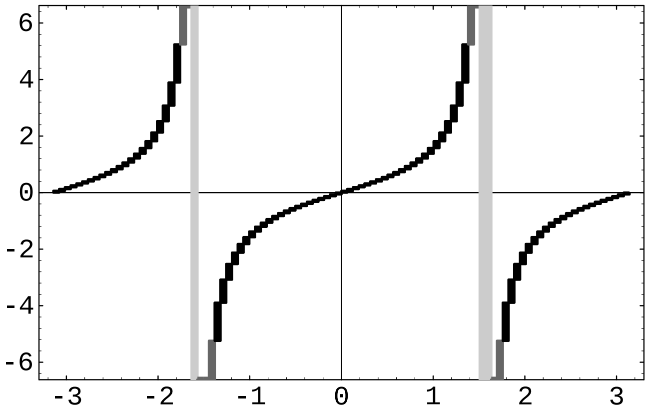

Figure 1: `plot[Tan[x], {x,-Pi,Pi}, PlotPoints->100]`

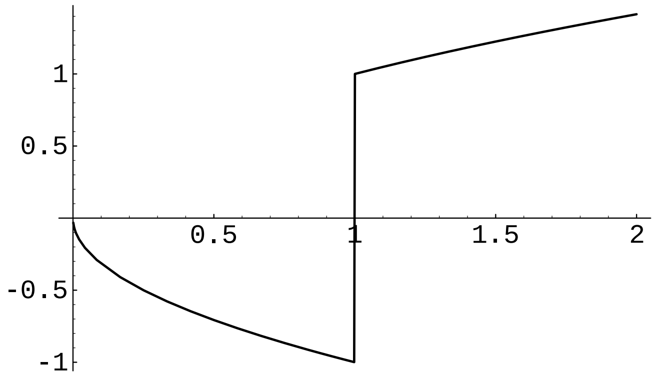

Figure 2: `Plot[g[x],{x,0.001,2}]`

more often. It is also advisable to check that functions being plotted can in fact be evaluated as intervals. (In *Mathematica*'s terminology the interval $[1, 2]$ is written `RealInterval[{1, 2}]`).

4.1 Using `plot`

Observe the results of the following commands, illustrating the different sorts of graphs that can be produced. This command produces Figure 1.

```
In[1]:= plot[Tan[x],{x,-Pi,Pi},PlotPoints->100]
(* fig. 1*)
```

Notice the light gray rectangle at $-\pi/2$, and $\pi/2$ in Figure 1. This light gray rectangle indicates that the rectangle as plotted extends vertically from $-\infty$ to ∞ in these regions.

Other color variations and markings are used to encode various data; marking *on the edge* of the frame, difficult to notice on the printed image, are more visible on a screen, and are used as follows:

A black rectangle indicates that the rectangle is not modified and so the points of the plot lie inside it. A dark gray box indicates that the rectangle has been truncated on the top or the bottom. If this dark gray box has a light gray line across the top or the bottom, then this rectangle goes to infinity in that direction If a section of the graph cannot be seen within the `PlotRange` at all, then a gray line is drawn at the edge of graph in the direction of the graph. A Dark gray indicates the rectangle goes off to a finite value, light gray indicates the rectangle goes to infinity.

In some cases the function does not actually become infinite, but we cannot compute a finite bounding box. We encountered this problem in the function discussed below.

A function you must reformulate for intervals

In the course of testing functions for this paper, we came across the function

$$y(x) := \frac{x\sqrt{x-2+\frac{1}{x}}}{x-1}.$$

Because it is not easily computed at $x = 0$ (although in the limit it is 0) we plot it from 0.001 to 2 in Figure 2. Notice the discontinuity at $x = 1$.

Using our interval plotting package, we find that the plot is computed very although correctly[3] although not very attractively. The reason for the unpleasantness in Figure 3 is that the formula as stated provides a gratuitous opportunity to divide by intervals that include very small values when x is near 1. In particular, when the point 1 is within some interval X, the subexpression $1/(X-1)$ requires division by an interval containing zero: the result on the graph is an "infinite" swath of grey, even if the singularity is removable (as it is here, since it is multiplied by a "stronger" zero.). This apparent singularity and the "blow-up" in the graph near $x = 1$ is clearly visible in Figure 3.

If we re-program the function in a better fashion for interval arithmetic and in particular remove the singularity by defining

$$y(x) = \begin{cases} -\sqrt{x}, & \text{if } x < 1; \\ \sqrt{x}, & \text{otherwise} \end{cases}$$

then the behavior, as shown in Figure 4, is much better. Note that if x is an interval, evaluating $x < 1$ requires in this case, that *all* of the interval be less that 1 [4]. In addition to problems at singularities, the need for choosing samples — perhaps arbitrarily

[3] After we patched Mathematica to provide interval computation of square-root.

[4] We had to patch Mathematica again to enable it to compare intervals.

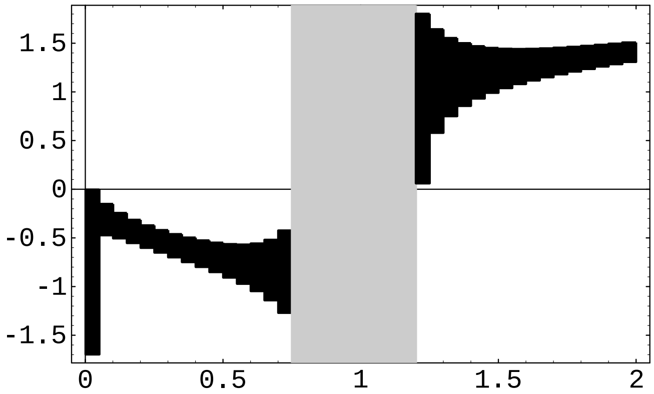

Figure 3: `plot[g[x],{x,0.001,2}]`

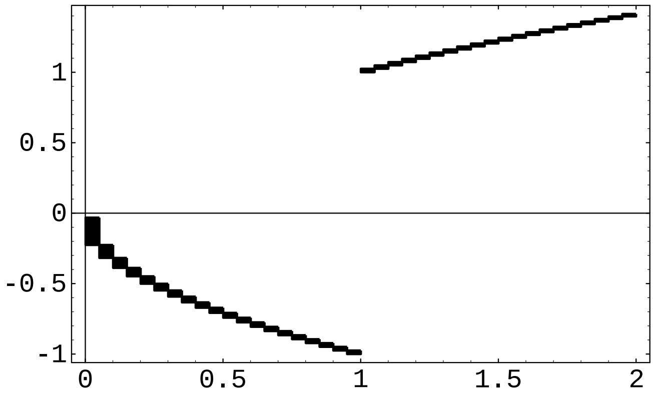

Figure 4: `plot [If[x>1,1,-1] * Sqrt[x], {x, 0.001, 2}]`

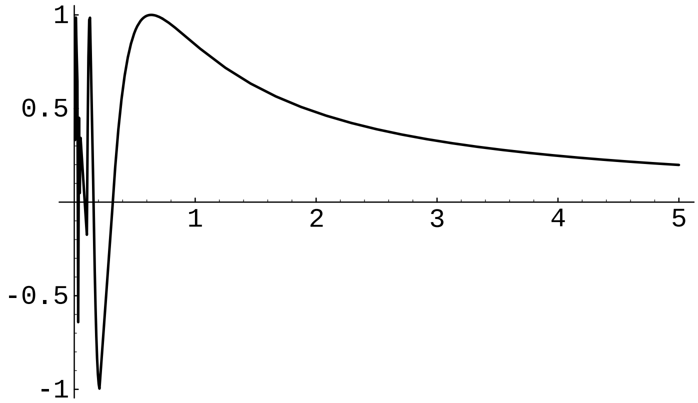

Figure 5: `Plot[Sin[1/x],{x,0,5}]`

closely — causes problems. Consider plotting $\sin(1/x)$ near $x = 0$.

Compare the graph in Figures 5 and 6, produced respectively by *Mathematica's* function and then our interval-arithmetic plot.

```
In[1]:= Plot[Sin[1/x],{x,0,5}]        (* fig. 5 *)
In[2]:= plot[Sin[1/x],{x,0,5},PlotRange->All]
                                      (* fig. 6 *)
```

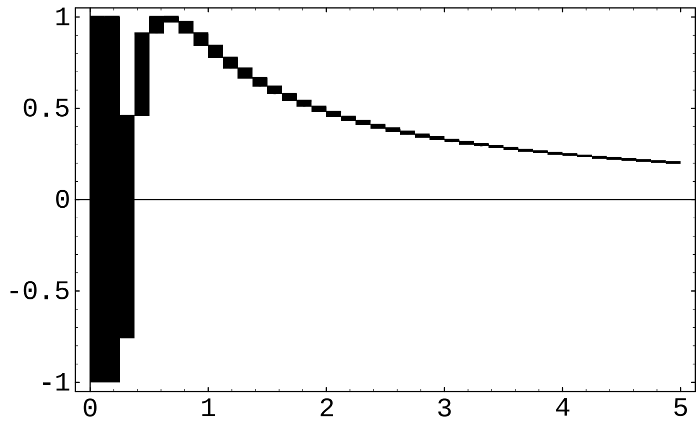

Figure 6: `plot[Sin[1/x],{x,0,5}]`

Notice in Figure 5 that the graph becomes erratic as $x \to 0$. This is a consequence of the sampling method. *Mathematica* is calculating points and then connecting them, and can't "keep up" with the ever-more-rapid oscillation of $\sin(1/x)$ as $x \to 0$. In figure 6 the interval-plotting program portrays the region near $x = 0$ as a solid block with range $[-1, 1]$. Choosing a higher value for `PlotPoints` will smooth out some of the curve, but the region near $x = 0$ will still be a block. This picture seems far more appropriate than the default plot.

A third example that may be of interest is a function suggested by W. Kahan to fool plotting programs. It features a narrow notch at $x = 4/3$ that drops to $-\infty$.

```
In[5]:= f[x_]:= 1+x^2 +0.0125*Log[Abs[1-3*(x-1)]]
In[7]:= Plot[f[x],{x,-2,10}](* fig 7*)
In[6]:= plot[f[x],{x,-2,10}] (*fig 8*)
```

Figure 7, *Mathematica's* plot, completely fails to display the notch at $x = 4/3$ where the function goes to $-\infty$. In our plot, shown in Figure 8, near $x = 4/3$ there is a dark grey box with a light grey line at the bottom. This indicates that the plot approaches $-\infty$.

Again, this picture seems far more appropriate than the default plot.

5 Using Tan and ArcTan to Plot Functions over an Infinite Range

If you are asked to plot $y = f(x)$ across a wide range of x or y it is sometimes advantageous to use a log or log-log plot. Unfortunately this does not help very much if you wish to plot a function with infinite extent, say

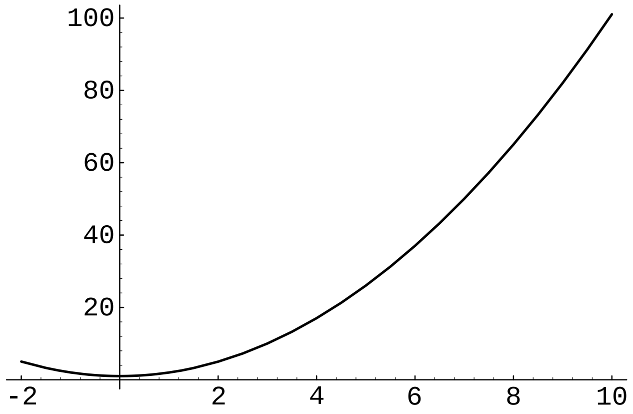

Figure 7: `Plot[f[x],{x,-2,10}]`

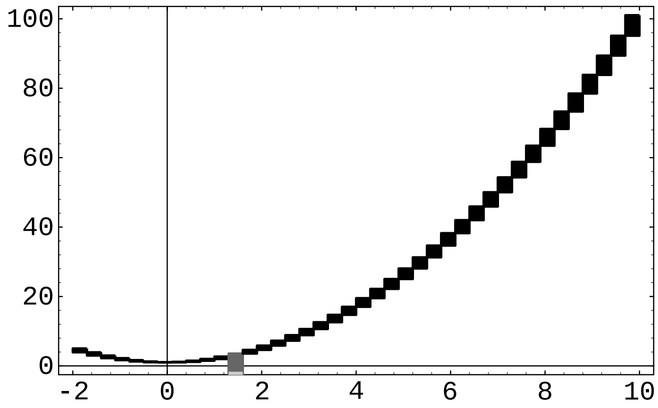

Figure 8: `plot[f[x],{x,-2,10}]`

$y = f(x)$ for x ranging from $-\infty$ to ∞. There is a solution, though, even if we have no *a priori* idea of where there is an interesting region for $f(x)$ With a clever enough program, we can try to narrow down the area of interest interactively. To start, however, consider a parameterization $x = t(\xi)$ that maps 0 to 0 but moves $x = \pm\infty$ to (say) $x = \pm 1$. Then plot $y = f(t(\xi))$. To be concrete, consider trying to plot $y = \sin(x)$ from $-\infty$ to ∞. Clearly it is impossible to divide the x-axis by any uniform spacing. We consider using a parameterization such as $t(\xi) := k\tan(\pi\xi/2)$. As ξ varies from -1 to 1, $x = t(\xi)$ varies from $-\infty$ to ∞. Choosing some value for k gives a kind of stretch factor. For $k = 4$, namely a plot of $y = \sin(4\tan(\pi x/2))$ we can get an idea of the curve even though it becomes progressively more distorted near the edges. There seems to be an acceptable ratio between y and x for about half the width of the plot. (see Figure 9). Naturally it compresses the x axis more and more severely as ± 1 represents $\pm\infty$. Maple [5] provides this kind of mapping for its so-called "infinity plots" if either of the supplied horizontal range end-points are infinite.

For the y axis, another approach works. If the y extent of a function is very large, we plot $\eta = (2/\pi)\arctan(y)$ instead of y. Then as $y \to \infty$, $\eta \to 1$.

In practice, the greatest difficulty in this transformation is to successfully lay out plot tick marks so the labels do not overlap. Assistance in the translation of a function so that the area of interest is stretched across the zero-point of the x scale, could also be provided.

Another, related, graphing problem concerns functions with infinite range. Take `f[x]:=1/x`. This approaches infinity as x approaches zero. A graph which represents `ArcTan[f[x]]`, could be used, with appropriately modified tick marks.

A good question regarding functions that approach infinity is how to determine this before graphing, and how "close" to infinity does a function have to be to toggle an `ArcTan[]` plot instead of a normal linear plot? A plausible (and in principle, rapid) technique would be to use a single interval evaluation of a function over the plot-range to determine if the function is infinite somewhere on that interval.

5.1 Using "ArcTan" Plots

Graphing functions with `ArcTan[]`, and `Tan[]` transformations to reduce the scale is again primarily an exercise in learning the details of getting *Mathematica* to graph ticks marks.

Examples

The following are some examples using the two functions, `arcTanPlot[]`, and `tanPlot[]`

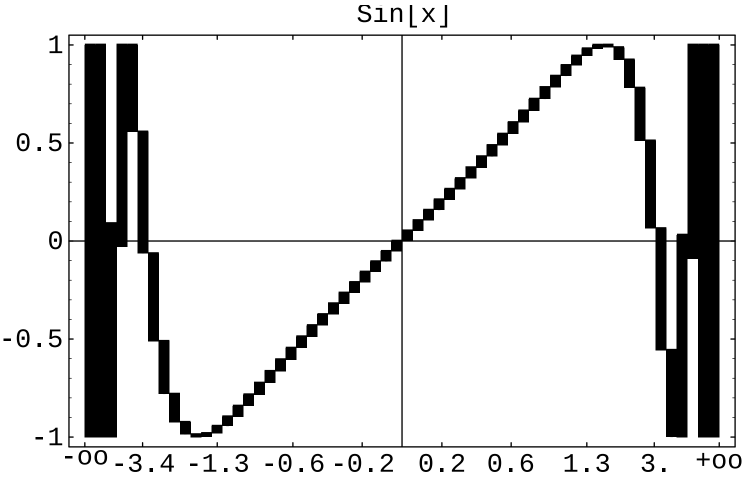

Figure 9: `arcTanPlot[Sin[x], {x,-Infinity, Infinity}, plot, PlotPoints->60]`

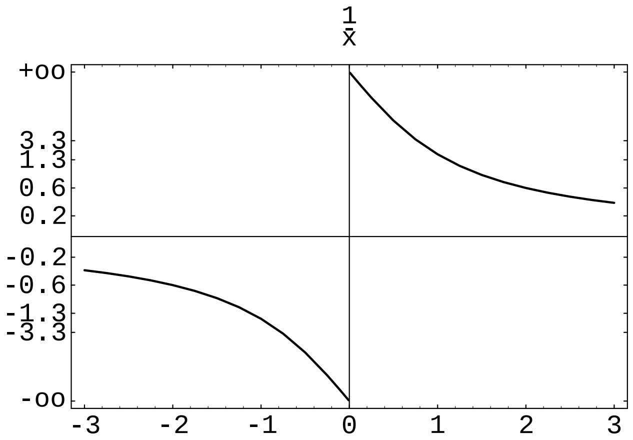

Figure 10: `tanPlot[1/x,{x,-3,3}]`

```
In[1]:= arcTanPlot[Sin[x],{x,-Infinity, Infinity},
          plot, PlotPoints->60]   (*fig.  9 *)
In[2]:= tanPlot[1/x, {x,-3,3}]      (*fig. 10 *)
```

Polynomials of high degree are tricky to understand. A good plot would show the asymptotic tendencies (to $\pm\infty$ but which direction at which end?) as well as the location of relative extrema and zeros. Since a naive plot of a polynomial may very well conceal some or even all of these attributes, it is worth looking at an example or two. Consider the polynomial of degree 13 $518400\,x - 773136\,x^3 + 296296\,x^5 - 44473\,x^7 + 3003\,x^9 - 91\,x^{11} + x^{13}$ which is more simply understood as

$$\prod_{k=-6}^{6} (x - k).$$

It has zeros at the integers from -6 and 6, and relative maxima or minima between them. If we plot it naively, say between -7 and 7, we lose most of the details, and see two lines going to infinity at the edges, with a nearly straight line along the x-axis between them. The locations of the zeros are obscured. If we use `tanPlot` we have a more useful result, although it appears distorted from the usual "polynomial" shape we might expect from low degree plots.

6 Conclusions and Future Work

6.1 How should this facility be integrated into a system?

If we wish to combine honesty (that is, security that we are not deceiving the user) and a modest approach to speed, it seems advisable to plot a graph first at a low resolution, but using intervals. Using this information the user or the system can detect qualitatively the areas that are potentially troublesome. Replotting using reasonable effort to produce an attractive and accurateline-plot provides another picture. If the two pictures are qualitatively the same, we can be more confident that the line version is, in fact, correct. It is also plausible to use the tactic of a preliminary interval plot to isolating smaller sections of the plotting field for detailed consideration

6.2 Implementation

6.2.1 Intervals

After some brief experiments it became clear that the provision of interval arithmetic in the underlying system was an afterthought. Its inadequacy became a considerable handicap in implementing plotting system. We have also tried out the Maple system's `evalr` facility, and it too seems to have a similar problem in that many operations that could, at least in principle, be carried out on intervals, are not. Again, it appears that some of the difficulty arises from trying to add on this data type late in the evolution of the system. A more satisfactory implementation "from scratch" of an interval-compatible computer algebra system is underway at Berkeley. A good design should allow for growth in precision of the endpoints, proper rounding rules, and use of intervals with infinite endpoints.

6.2.2 Plotting

Although the *Mathematica* plotting package design is quite flexible, and benefits substantially, in our opinion, by the interactive nature of the underlying system, it is difficult to control precisely. Perhaps a suitable analogy is to think of it as an automobile that running well on the highway; if you try to use it for "off-road" driving, it is rather difficult. It required substantial time to understand the subtle ways in which the user's intentions may be thwarted by the built-in algorithms. Some of the problems [3] could only be overcome by directly emitting PostScript instructions from *Mathematica* to the plotting engine.

In spite of these difficulties, we found the representation of graphics objects to be quite expressive, and the rendering generally quite attractive. Probably no other system available in the summer of 1991 would have been quite as convenient in this regard. We understand that other systems have progressed, and may be competitive for graphics now.

6.3 Intervals for global optimization and constants

If using interval arithmetic for plotting becomes more widespread, we may see an increased appreciation for the use of interval arithmetic. The implications for global optimization based on interval arithmetic in computer algebra systems is also clear – one can apply fairly powerful techniques at low intellectual overhead in such a system to get guaranteed answers about global optima. Given a correct implementation of intervals, the down-side risk is that one may have to wait longer for any answer whatsoever.

Another plausible use of interval arithmetic in a symbolic context is for determining the sign of constants [7]. While this is not a decision procedure, it is an excellent heuristic that may succeed in bounding a non-zero constant away from zero.

6.4 Future work

Other techniques for discovering "hidden" properties of $y = f(x)$ at astonishingly low cost via divided differences [9] will be discussed in a future paper. A preliminary discussion of some of these issues (omitted from this paper for reasons of length) are contained in a draft report [3]. As a brief example, the anomaly in the function plotted in figures 7 and 8 can be easily detected by divided difference techniques. Although these technques do not share the "guarantees" of interval arithmetic they do take advantage of the additional precision usually provided by most computations. Consider that most displays are good to about 1 part in 2^{10} and that thereore a double-precision floating-point fraction has another 43 bits of information that is *not visible.* By computing differences we can extract useful information about higher-order derivatives and singularities.

Another direction for extension is to surfaces or higher dimensions. We have experimented with "honesty" in higher dimensions (specifically projections of 3-D surfaces). The interval techniques are less appealing since the display of rectangular polyhedra in three-space is messy. We have, however, examined the use of *adaptive surface plotting* as an approach to honest 3-D graphs [4]. The current (version 2.0) *Mathematica* system offers surface plotting on rectangular meshes only; it is clear that a more flexible grid structure is advantageous for the display of surfaces where there are large smooth sections and small complex ones. Our approach, using repeated subdivisions on a "restricted quad-tree" base, seems to provide a manageable subdivision algorithm that can be incorporated into a computer algebra system.

7 Acknowledgments

We are grateful to G. Seroussi and F. Kitson of Hewlett-Packard Research Laboratory for encouragement, and for the financial support of Hewlett-Packard Inc. Ilja Bedner assisted with the programming and testing of functions used here. W. Kahan has also provided useful comments. Partial support for computation was supplied by National Science Foundation Infrastructure Grant number CDA-8722788.

Programs used for these displays may be obtained from the author (`fateman@cs.berkeley.edu`).

References

[1] Tryg A. Ager and Robert Elton Maas. *Generating "honest" mathematical graphs.* IMSSS Technical Report 317. Stanford Univ. Draft.

[2] G. Alefeld and J. Herzberger. *Introduction to Interval Computations.* Academic Press, 1983.

[3] I. Bedner and R. Fateman. "Adventures in the Honest Plotting of 2D Mathematical Functions." (Tech. Rept. CS Division, EECS, UC Berkeley, 1991).

[4] I. Bedner and R. Fateman. "Adventures in Plotting of 3D Mathematical Functions." (Tech. Rept. CS Division, EECS, UC Berkeley, 1991).

[5] B. W. Char, K. O. Geddes, G. H. Gonnet, M. B. Monagan, S. M. Watt. *MAPLE Reference Manual.* Symbolic Computation Group, Dept. of Cmpt. Sci., Univ. of Waterloo, Waterloo Maple Publ., 1990; Springer-Verlag, 1991.

[6] Theodore W. Gray and Jerry Glynn. *Exploring Mathematics with Mathematica.* Addison-Wesley Publishing Company, Inc., Redwood City, California, 1991.

[7] W. Kahan. "Fear of Constants" (this symposium: ISSAC-92, Berkeley, CA).

[8] Roman Maeder. *Programming in Mathematica.* Addison-Wesley Publishing Company, Redwood City, California, Second Edition, 1991.

[9] J. C. P. Miller. Checking by Differences. *Mathematical Tables and Other Aids to Computation* Vol. 4, 1950. 3–11

[10] L. M. Milne-Thompson. *The Calculus of Finite Differences.* Macmillan and Co., London, 1933.

[11] Nancy Blachman. *Mathematica: A Practical Approach.* Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1991.

[12] Stephen Wolfram. *Mathematica—A System for Doing Mathematics by Computer.* Addison-Wesley Publishing Company, Inc., Redwood City, California, Second Edition, 1991.