Algorithms for Choosing the Domain and Range when Plotting a Function

Leland Wilkinson
SPSS, Inc.
Northwestern University
leland@spss.com

ABSTRACT: A computer program that plots functions from algebraic input should be able to determine automatically a useful plotted domain and range for as many types of functions as possible. Doing so requires some notion of an informative plot, as well as a method for achieving it. This paper presents a set of algorithms for this purpose.

KEY WORDS: domain, range, plot, computer graphics

Leland Wilkinson is Senior VP, SPSS, Inc. and Adjunct Professor, Department of Statistics, Northwestern University. Email: leland@spss.com.

Revised from paper with the same title published in A. Buja and P.A. Tukey (Eds.), *Computing and Graphics in Statistics*, Springer-Verlag, 1991, pp. 231-237.

1.0 Introduction

Suppose we have an algebraic function $f : X \to Y$ where $X \subseteq \mathbf{R}$ and $Y \subseteq \mathbf{R}$. We wish to plot this function such that the graphic representation is *informative*. A definition for informative should allow us to create plots that reveal whether *f* is continuous, differentiable, monotonic, bounded, periodic, symmetric, etc. Our strategy for finding a range and domain that produce an informative plot will be to sample values from *X* and examine their behavior under *f*.

2.0 Informative plots

The definition of an informative plot depends on the definition of a plot itself. Definition 2.1 covers a plot and Definition 2.2 covers an informative plot.

Definition 2.1. Let G be the graph of an algebraic function $f : X \rightarrow Y$. A plot P is a bounded subset of this graph defined by restricting its domain to the closed interval $A \subseteq X$ and its range to the closed interval $B \subseteq Y$. We call $F = A \times B$ the plot *frame*. Thus, $P = F \cap G$.

Definition 2.2. A plot P is informative with respect to f if for all $c \in X - A$ and $d \in f^{-1}(Y - B)$ we may approximate $f(c)$ and $f(d)$ arbitrarily closely using graphic information contained in the frame. An informative domain is the domain of an informative plot. An informative range is the range of an informative plot.

Definition 2.1 defines a plot as a subset of the graph of a function. Definition 2.2 bases the notion of an informative plot on our ability to extrapolate from the plot. An analogy might help. Suppose we are looking at a display of P in a (small) frame on a (large) computer screen and have a drafting utility that includes polylines, splines, and other tools for drawing smooth curves and straight lines. With these tools, we should be able to extrapolate an informative function plot on this computer screen to predict aproximately $f(x)$ for any $x \in X$ outside the frame and inside the screen bounds without knowing the definition of f. Thus, if a qualitative feature, such as a discontinuity, occurs outside the bounds of an informative plot, we should have some indication of its existence and location from information inside the frame.

Figure 1 shows some examples of informative and uninformative plots. Notice that a plot can be uninformative for different reasons. The plotted domain and/or range may be too small for a viewer to recognize asymptotes or periodicity. On the other hand, it may be too large to allow sufficient resolution inside the frame. Of course, not all algebraic functions are amenable to informative plotting, but we seek some practical methods that will produce recognizable results for at least a few.

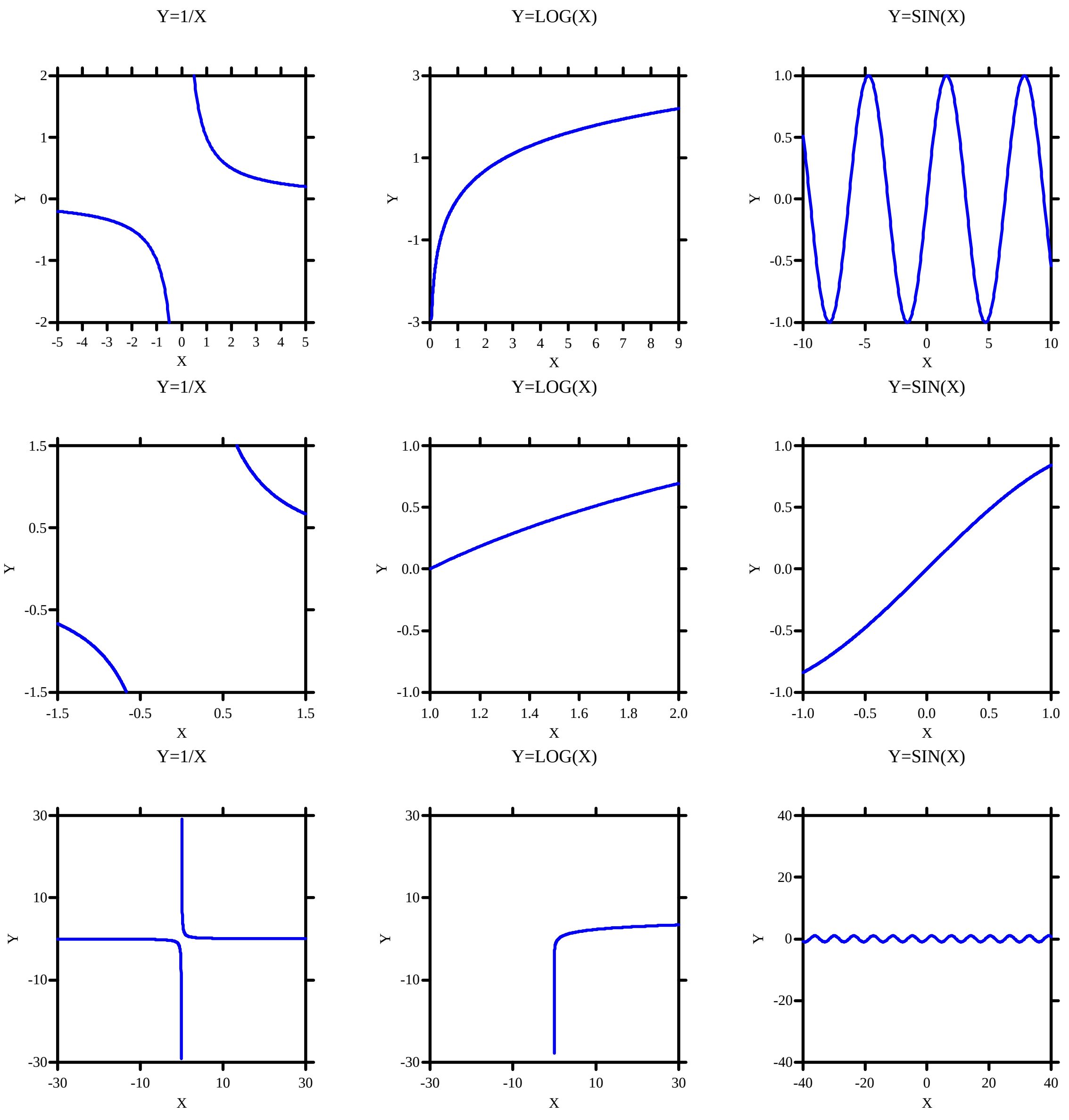
3.0 Algorithms for Approximating an Informative Domain

A simple greedy strategy for creating an informative plot begins with finding an informative domain. After this, various methods can be applied to finding an informative range associated with that domain. Finding an informative domain for a given function depends on at least three features: *periodicity*, *asymptotics*, and *monotonicity.* To allow a robust search for these features, the function generating routine, algorithms 3.1 and 3.2, and the plotting software should be written to handle incalculable values (domain errors and numerical instabilities). Note that these algorithms will yield an informative plot only if the interesting activity of the function occurs in a domain near zero.

3.1 Periodicity

The following algorithm is designed to determine the period of a periodic function. The initial parameter settings determine the space searched. *Factor* is the product of a few small prime numbers. *Small* and *large* determine the smallest and largest period the algorithm can detect. *Cycles* determines how many cycles to test before assuming *f* is periodic. *Delta* is calculated from *midrange*(*f*(*x*), *n*)***fuzz* for a small sample (*n*) of values of *x*. The value of *fuzz* tunes *delta*. The *midrange*() function computes (x_{75} - x_{25}) for an ordered set of values, where x_{75} is the value in the set nearest the 75th percentile and x_{25} is the value nearest the 25th percentile of the sorted values. The values in this set can be sampled randomly or by a binary search beginning with a small number.

FIGURE 1. Informative (top row) and Uninformative Function Plots



```
factor := 210
small := π / factor
large := factor * π
cycles := 4
n := 100
fuzz := .1
delta := midrange(f(x), n) * fuzz
period := small
while (period < large) do begin
    for x := small to large step period do begin
        fa := f(x)
        periodic = true
        for i := 1 to cycles step 1 do begin
            fb :=  f(x + i*period)
            fc :=  f(x + i*period + period/2)
            if  abs(fb - fa)  > delta or abs(fc - fa) < delta then periodic = false
        end
        if periodic then stop
    end
    period := period + delta
end
FAIL: no period identified
```

This is a direct search procedure that looks for similar function values at periodic points in the domain. This search should be performed first for positive and then for negative domain values. The algorithm requires a *delta* value for comparing whether two values of *f*(*x*) for different values of *x* are equivalent. The inner loop checks that comparable values in the range occur at periodic points on the domain and not in between. This net traps simple trigonometric functions and some compositions of them.

This algorithm can be refined to search more densely among smaller values of *period*. Other algorithms, particularly some in the frequency domain, may be more effective for testing periodicity. Some may require more storage or order of magnitude in number of computations.

3.2 Asymptotics and Monotonicity

The following algorithm will determine an informative domain for functions that are not periodic. It should be run only if algorithm 3.1 fails. As with algorithm 3.1, the initial parameter settings determine the sensitivity of the procedure.

```
n := 50
large := 1000
x := 1 / large
signSlope := sign(f1(x));
for i := 1 to n step 1 do begin
    if  signSlope = sign(f1(x)) and (abs(f2(x)) < x^2 / large or abs(f2(x)) > large / x^2)) then stop
    signSlope := sign(f1(x))
    x := 2 * x
end
FAIL: no monotonicity identified
```

This is a binary search procedure. We are looking for a plotted domain within which numerical second derivatives are not extreme. Monotonicity is tested by checking the first numerical derivative (*f1*) on each iteration. Asymptotic behavior is tested by looking for extremely small or large numerical second derivatives (*f2*). The test limits for second derivatives are conditioned on the value of x^2 in order to add a penalty for extremely small domains. The final value of $x = 2$ is chosen if both algorithms 3.1 and 3.2 fail. Like the periodicity test in 3.1, this algorithm should be performed for positive and then for negative domain values.

3.3 Trimming the Domain

If *f1*(*x*) = 0 on termination of algorithm 3.2, then it is possible that the function is a constant at the edge of the plotted domain. In this case, it is useful to trim the plotted domain from the ends up to the point where variation in *f* is visible. To do this, it is helpful to store in an array all the values of *x* and *f2*(*x*) called so far. Next, sort these values on *x*. Finally, trim *x* until the corresponding value of *f1*(*x*) differs by more than delta from *f1*(*x*) at the end of the series.

3.4 Odds and Ends

The plotted domain is defined by the bounds found in the positive and negative passes through algorithms 3.1 and 3.2. If *f* is undefined for all positive or negative domain values, the plotted domain should be truncated on the right or left. If the positive and negative bounds for the plotted domain are similar in absolute value, the domain should be made symmetric about 0.

4.0 Algorithms for Finding an Informative Range

Finding an informative range is similar to the procedures for finding an informative domain, except the search can be confined to the plotted domain. Caveats concerning undefined values in calculating the plotted domain apply as well to the calculation of the range.

4.1 Compute Range for Plotted Domain

This algorithm computes values of the range and first derivatives inside the bounds of the plotted domain. The array dimensions ($n = 100$) determine the sensitivity. After the arrays are filled, the values in the array *af* and *af1* are sorted on *af*. The sorted values will be used later to trim the range.

```
n := 100
**dimension** af(n), af1(n)
xmin := <*lower bound of plotted domain found in Section 3*>
xmax := <*upper bound of plotted domain found in Section 3*>
delta := (xmax - xmin) / n
x := xmin
npt := 0
for i := 1 to n step 1 do begin
    while <f(x) and f1(x) are calculable> do begin
        npt := npt + 1
        af(npt) := f(x)
        af1(npt) := f1(x)
        x := x + delta
    end
end
sort(af, af1)
```

4.2 Trim Range

For some functions, such as *tan*(*x*), the range may be infinite within a plotted domain. We must trim the plotted range so that variation within the plot will be visible. A way to do this is to limit the physical slopes of plotted segments. Cleveland [1] used a similar approach in setting the median absolute physical slope of time series plots to 45 degrees. Computing the physical slope requires knowing the plotted range and domain, however. The following algorithm successively trims and recomputes physical slopes, assuming a square physical plot, until an acceptable range is found. If none can be determined this way, or if *npt* < 2 after algorithm 4.1, then the plotted domain is used for the range.

```
big := 10
huge := 10^8
n1 := 1
n2 := npt
for i := 1 to npt step 1 do begin
    ymin := af(n1)
    ymax := af(n2)
    if  ymax ≤ ymin or n1 ≥ n2 then stop
    delta := (ymax - ymin) / (xmax - xmin)
    if  delta > huge or abs(af1(n1)) / delta > big then n1 := n1 + 1
    if  delta > huge or abs(af1(n2)) / delta > big then n2 := n2 - 1
end
```

4.3 Odds and Ends

If the range is similar to the domain in length, then the length of the range should be equated to the length of the domain. Next, if the minimum values of the range and domain are close, they should be equated to the lesser value. If the positive and negative bounds for the plotted range are similar in absolute value, the range should be made symmetric about 0. Finally, it is convenient to use nice round numbers and 3 to 10 tick marks in plotting scale values [2, 3]. These constraints may require modifying slightly the plotted domain and range.

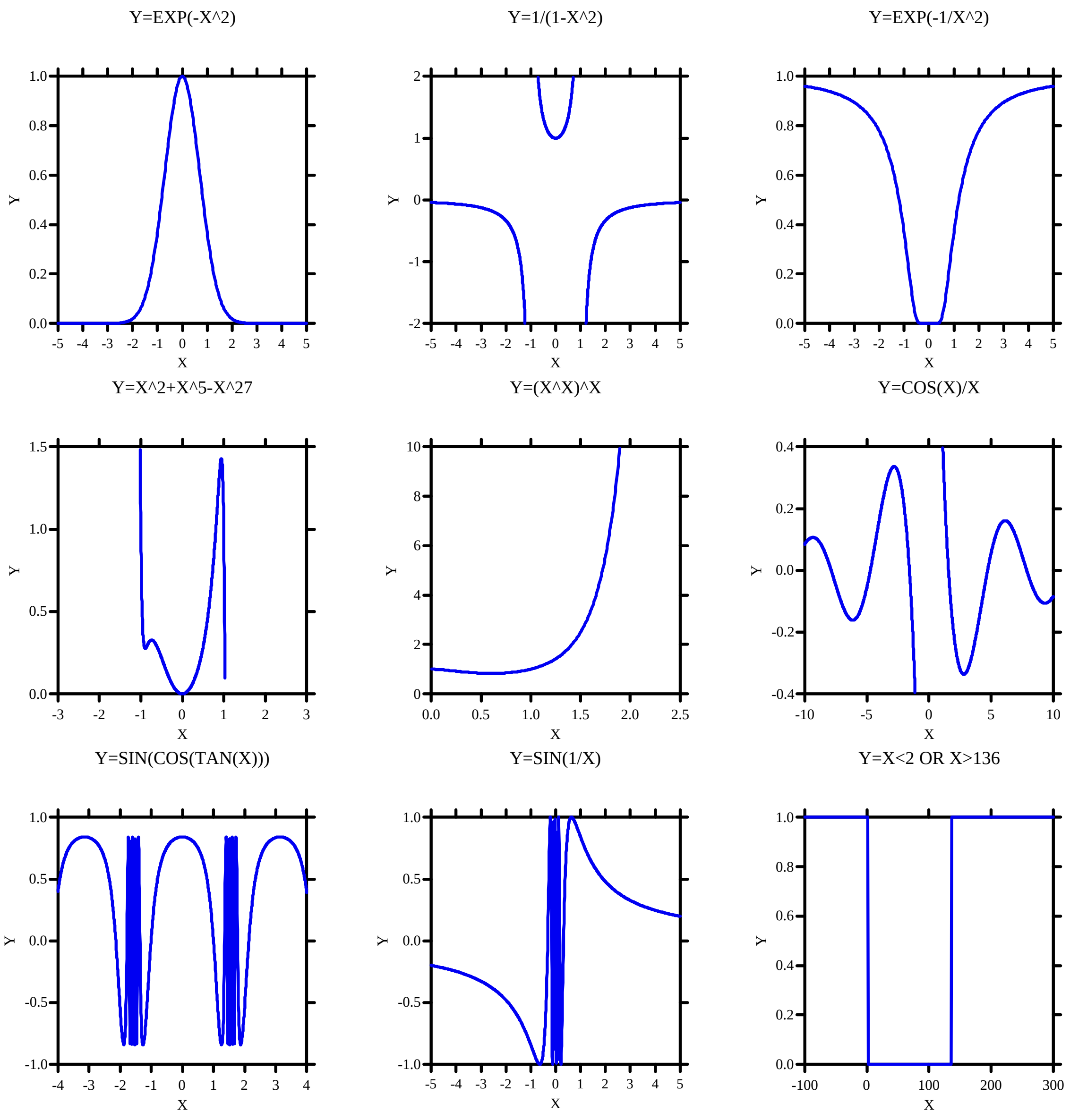
5.0 Three-dimensional Plots

These algorithms work quite well when applied separately to each axis in a three-dimensional plot.

6.0 Some Examples

Figure 2 shows some examples of functions plotted using these algorithms. The titles are the arguments in SYSTAT [4] that produced the plots with the FPLOT command. The SYSTAT algorithm is set to favor exact [0, 1] intervals for domains and ranges in that neighborhood. Collisions with frame borders can be avoided by using crossed axes.

FIGURE 2. Sample Automated Function Plots using SYSTAT



References

[1] W.S. Cleveland, M.E. McGill, and R. McGill. *The shape parameter of a two-variable graph*. J. Am. Statist. Assoc. 83 (1988), pp. 289-300.

[2] C.R. Lewart, Algorithms SCALE1, SCALE2, and SCALE3 *for determination of scales on computer generated plots*, Comm. A.C.M. 16 (1973), pp. 639-640.

[3] W.D. Stirling, *Scale selection and formatting*, in P. Griffiths and I.D. Hill (Eds.), Applied Statistics Algorithms, Ellis Horwood Limited, West Sussex, 1985.

[4] L. Wilkinson, SYSTAT: *The System for Statistics*, SYSTAT, Inc., Evanston, IL, 1988.