A Toolkit for Optimizing Functions in Economics*

William L. Goffe
Dept. of Economics and International Business
University of Southern Mississippi
Hattiesburg, MS 39406
Bill.Goffe@usm.edu

June, 1997

Abstract

Optimization algorithms must be among the most common numerical methods used by economists. Yet, there is surprisingly little guidance on choosing the appropriate one. This problem is most notable with regard to conventional versus global optimizers. Typically, a global optimizer is used when a conventional one fails after substantial "fiddling" with a conventional optimizer. This paper introduces three different, easy-to-use, tools (cross-sections, radius plots, and a measure of the non-quadratic behavior of a function) that are designed to indicate when a global optimizer is needed. With their use, researchers should spend less time fiddling and more time generating results.

Introduction

Recent years have seen economists starting to use global optimizers in their econometric and modeling work; examples include the genetic algorithm (Dorsey and Mayer, 1995) and simulated annealing (Goffe et al., 1994). These algorithms, besides being designed for global optimization, are also much more robust than conventional algorithms, which were designed for unimodal, and frequently, roughly quadratic functions. However, global optimization algorithms often have longer run-times, and perhaps more importantly, learning to use them effectively takes some time and effort. Unfortunately, there is little guidance on when these algorithms are needed, or when they are overkill and conventional algorithms are appropriate. This paper attempts to help fill this gap: it describes three easy to use tests that evaluate functions and thus recommend the type of optimization algorithm to use.

In practice, researchers tend to go to substantial lengths with one or two optimization algorithms when estimating or simulating a model. One might try other starting values, or

*I would like to thank, without implicating, Michael Veall for providing sample code and data for the Hoffman and Schmidt (1981) example, Robert Fourer for help with the literature, and Mark Dickie for a useful insight.

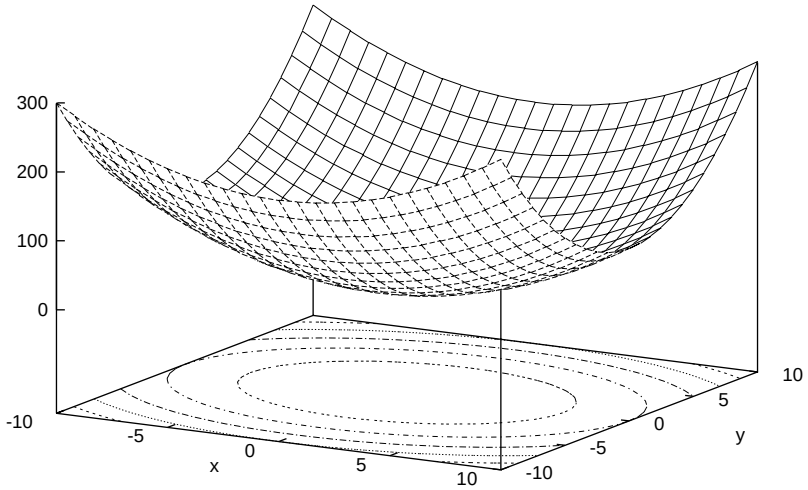
try different parameters that control the optimization algorithm. Only when these efforts fail after substantial effort will another algorithm be tried. In effect, optimization algorithms are used to examine the function to be optimized. Clearly, it would be better to use methods that directly evaluate the function to determine what sort of algorithm is appropriate. Further, the methods introduced here are quite easy to use and implement, so little time will be lost in their use, and often, substantial aggravation will be avoided. As it is difficult to quantify the exact behavior of conventional and global algorithms, the tests described here are qualitative in nature, while providing very useful clues to the nature of the function.

This paper is organized as follows. The next section describes four different functions to illustrate these tests. Three of these are 2-dimensional; this permits a fuller explanation of how the tests work. The following sections describe the three different tests: cross-sections, radius plots, and a direct measure for quadraticity. A conclusion follows.

Functions

To demonstrate the methods used here, four different functions (all in minimization form) are analyzed with these methods. Three of the test functions have only two variables to better illustrate the properties of the methods introduced here. To start, a very simple quadratic function, $(x^2 + 2y^2)$ is used. It is shown in Figure 1; note the contours on the bottom of the plot—contours are shown with other figures as well to aid understanding.

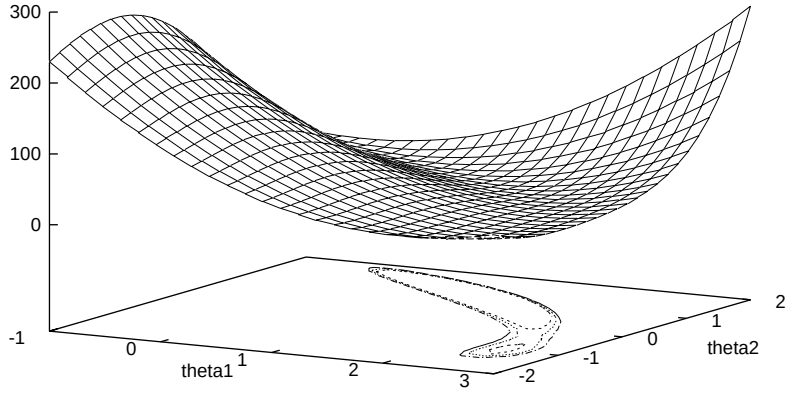
Figure 1: Quadratic Function



A somewhat more difficult function, taken from the economics literature Judge et al. (1985, pp. 956-8), is shown in Figure 2. As the contours suggest, it contains two minimums

on a long, curved valley. There is a local minimum of 20.98 at (2.35, -0.319), and a global minimum of 16.08 at (0.865, 1.24).

Figure 2: Judge Function



A third function, from the optimization literature, is shown in Figure 3 and described by

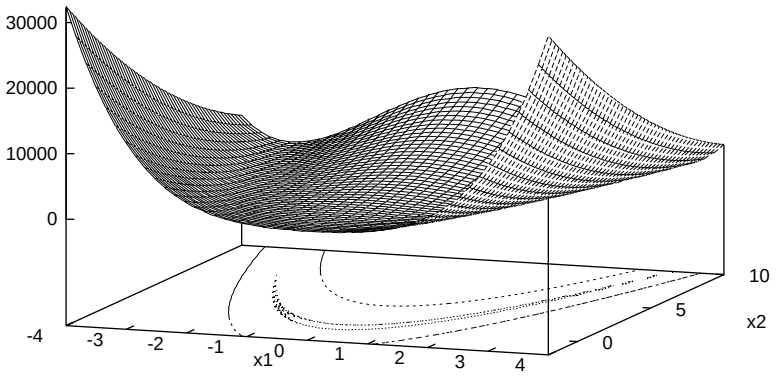
$$f(x_1, x_2) = 100 \cdot (x_2 - x_1^2)^2 + (1 - x_1)^2.$$

This is the Rosenbrock (Rosenbrock (1960), and Moré et al. (1981)) function of two dimensions—it contains a long, very curved valley with the minimum is at (1,1) with a function value of 0. This shape challenges many optimization algorithms. The fourth function is from Hoffman and Schmidt (1981), with synthetic data from Veall (1990). It is a rational expectations model estimated by maximum likelihood[1] Unlike the other functions, this one has 8 variables. To be specific, they are γ, β_1, β_2, λ_1, λ_2, σ_0^2, σ_1^2, and σ_2^2, and the model is described by

$$\begin{aligned}
Y_t &= \gamma E_{t-1} Y_t + \beta_1 X_{1t} + \beta_2 X_{2t} + \epsilon_{0t},\ |\gamma| < 1, && (1)\\
X_{1t} &= \lambda_1 X_{1t-1} + \epsilon_{1t}, && (2)\\
X_{2t} &= \lambda_2 X_{2t-1} + \epsilon_{2t}, && (3)\\
\epsilon_{it} &\sim i.i.d.\ N(0, \sigma_i^2), && (4)\\
Y_t &= \beta_1 X_{1t} + \beta_2 X_{2t} + (\gamma/(1-\gamma))(\beta_1 \lambda_1 X_{1t-1} + \beta_2 \lambda_2 X_{2t-1}) + \epsilon_{0t}. && (5)
\end{aligned}$$

Equation (5) is derived from the other equations, and (2), (3) and (5) are used to estimate the model. As this function has eight variables, it is not plotted.

Figure 3: Rosenbrock Function



Methods

Cross-sections

While the 3-d graphs used above were illustrative, generating useful ones can be surprisingly difficult.[2] Plus, 3-d graphs are only suitable for functions of two variables, or plotting a subset of two variables; of course, many functions of interest have many more than two variables and the number of needed plots could be quite large. In addition, generating 3-d graphs requires software that might not be readily available. What is needed is some way of rapidly and easily looking at the surface of an n-dimensional function; the "cross-sections" introduced here are one such method.

A cross-section simply evaluates an n-dimensional function along a line. Since a cross-section is two dimensional, it can easily be plotted. With a number of cross-sections, some sense of the general shape of the function can be determined, and the appropriate optimization algorithm selected. A cross-section is defined as follows. First, "box" the region of interest in $\mathtt{R}^{\mathtt{n}}$ by $\mathtt{l(i)}$ and $\mathtt{u(i)}, \mathtt{i} = 1, \ldots, \mathtt{n}$, where $\mathtt{l(i)} < \mathtt{u(i)}, \forall \mathtt{i}$, where $\mathtt{l}$ is the lower bound and $\mathtt{u}$ is the upper bound. From these bounds, two points are selected, connected by a line, and the function is evaluated along this line. The resulting cross-section can easily be plotted.

Since *a priori*, one does know what points might be of interest, randomly selected end point are desirable; in fact, as cross-sections do not cover much of the function, a number of cross-sections, each with different end points, are needed. Thus, randomly select two points, $\mathtt{x_0}$ and $\mathtt{x_1}$, where $\mathtt{l(i)} < \mathtt{x_0(i)} < \mathtt{u(i)}, \forall \mathtt{i}$ and, $\mathtt{l(i)} < \mathtt{x_1(i)} < \mathtt{u(i)}, \forall \mathtt{i}$. Next, randomly select

j_0 and $j_1, \in j = 1, \ldots, n$, and set $x_0(j_0) \leftarrow l(j_0)$ and $x_1(j_1) \leftarrow u(j_1)$. The resulting two points, x_0 and x_1 are on the "box" bounding the area of interest. From x_0 to x_1, evaluate f(x) at a given number of points. The resulting set of f(x) evaluated along the line inside the box, is the cross-section. The vertical axis is the values of the function, and the horizontal axis is a combination of the values of the variables of the function.

Cross-sections were generated for all test functions but the quadratic (which given its simple shape, would generate unremarkable cross-sections). For each, after selecting upper and lower bounds, 20 different cross-sections were generated. Figure 4 shows one of the cross-sections of the Judge function (for this and the Rosenbrock function, bounds of $\pm$ 4 were used). It shows that the function is non-quadratic, and may even have more than one minimum. Of the 20 cross-sections, three showed this "double dip" pattern – thus, with a relatively few cross-sections, one sees a key element of this function – its possible multimodal character, and its certain non-quadratic character. Figure 5 shows the Rosenbrock function; four of them showed a similar double dip pattern. In this case, as is known *a priori*, this function has only one minimum; the double dip comes from intersecting the very curved valley twice. Thus, these two functions demonstrate that a double dip in the cross-section may mean two things: multiple minima, or a curved valley. Yet, they agree on the key point: conventional algorithms could well experience difficulty with these functions.[3]

Figure 4: Judge Cross-Section

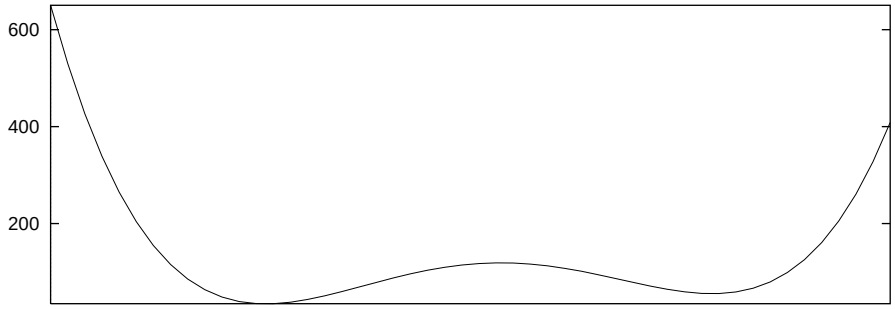


Figure 6 shows one of the 20 cross-section for the Schmidt function; it looks as nice as any function could for minimization. Figure 7 is another cross-section; needless to say, it demonstrates the difficulty this function presents.

With the Judge function, Goffe et al. (1994) found conventional algorithms are almost evenly split between finding the local and global optimum (however, they had no trouble finding *an* optimum). Goffe (1996) reports a different experience with the the Schmidt function. Many different randomly selected starting values were used, and if they were very far from the optimum,[4] the algorithm, DFP from Press et al. (1992), the optimum was rarely, if ever, found. Thus, with both functions, if cross-section were used first, which indicates the need for a global optimizer, puzzlement and even aggravation could have been avoided.

Figure 5: Rosenbrock Cross-Section

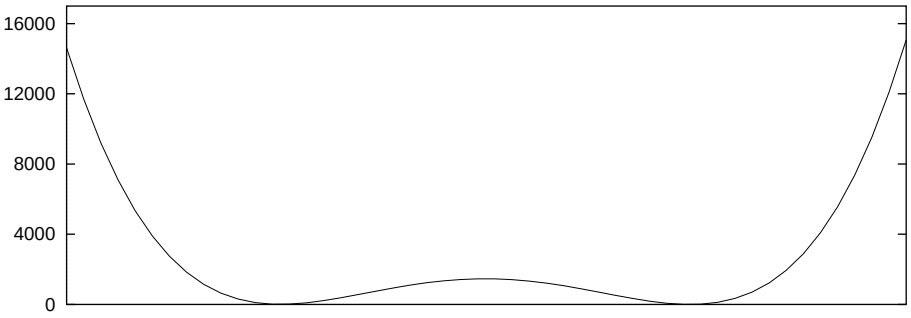


Figure 6: Schmidt Cross-Section #1

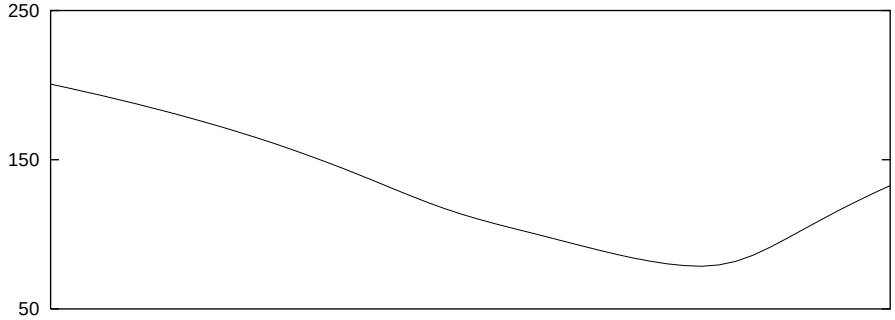
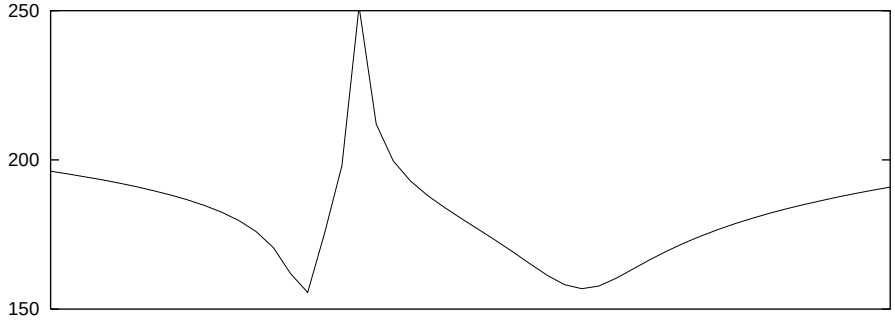


Figure 7: Schmidt Cross-Section #2



Radius Plots

This section describes another method of viewing functions. As with cross-sections, the goal here is to take an n-dimension function and somehow view it in two dimensions, while retaining useful information for determining the appropriate optimization algorithm. Radius plots are generated as follows. First, the function is sampled at many different points, and

the minimum function value from this set of points is recorded. From the minimum point, distances to all other sampled points are recorded, and the function value at each point is recorded. Finally, a plot of the distance versus function value is generated. As the horizontal axis of this plot is the distance from the sampled minimum, and the vertical axis is the function value at that distance, the term radius plot was coined.

Consider the radius plot of a flat surface—its radius plot will be flat. Starting at the sampled minimum, every point at every distance from there will have the same function value. Next, consider the radius plot of the function $(x^2 + y^2)$. As you move away from the sampled minimum, the function has the same larger value in all directions. Thus, the radius plot will be the same as a cross-section drawn through the sampled minimum—a quadratic.

In general, when moving away from the minimum, the function value rises continually, then the function likely does not have multiple minimum or long valleys. If, on other the other hand, function values far from the minimum are little different than those near the sampled minimum, then long valleys or multiple optima are likely, so a robust or global optimizer is appropriate.

To sample the function's parameter space, a "quasi-random" sequence is used. As described in general terms in (Press et al., 1992, pp. 229-306), such sequences were originally used in numerical quadrature to "better" sample an integrand than Monte Carlo methods. Rather than selecting elements randomly, each point is chosen so that the distance between points is minimized. They use the term "maximally avoiding" for the points selected. A number of different methods for computing quasi-random sequences are available; the Bratley and Fox (1988) implementation of Faure (1982) was used here.[5]

Figure 8 shows the radius plot from the quadratic function (in logarithmic form) with 1,000 sampled points from (-4,4). The minimum point found has a function value of 0.0029 at (0.03125 -0.03125), which of course is quite close to the actual minimum. In general terms, as one moves away from the sampled minimum, the function value rises. Note the variations in the function value in the broad middle of the plot. Recall that this function is $(x^2 + 2 \cdot y^2)$. The "2" coefficient on the second term ensures that at the same distance from the sampled minimum, the function has different values. The lack of variation at the ends of the plot is due to very few points very near the optimum, and few in the "corners" of the sample space. In addition, with a log plot, the variation in function values falls. The generally steady rise in this plot indicates that a conventional optimization algorithm is probably appropriate (which of course is the case for this simple function).

Figure 9 shows the radius plot for the Judge function. Again, 1,000 points were evaluated from (-4,4). Unlike the quadratic plot, there is quite substantial variation as one moves from the sampled minimum (a function value of 16.15 at (0.742 1.29); the actual minimum has a value of 16.08 at (0.865, 1.24)). At some points from the optimum, values are found almost as low as the sample minimum. Since some values at the same radius are much higher, this indicates a long valley in one direction, with possible multiple minimum. As described in a previous section, this is an accurate characterization of the Judge function. It turns out that the global minimum is a distance of 2.75 from the local minimum, so at about this point the lower part of the graph starts to rise—at much greater distances, all lower points are significantly larger.

Figure 10 shows the radius plot for the 2-dimensional Rosenbrock function. Again, 1,000

Figure 8: Quadratic Radius Plot

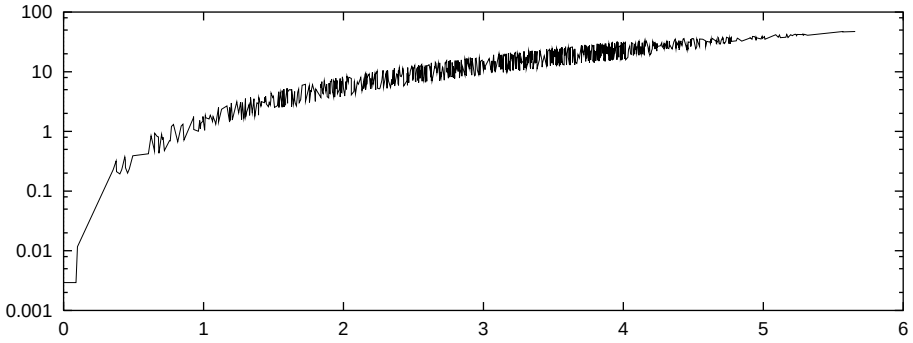


Figure 9: Judge Radius Plot

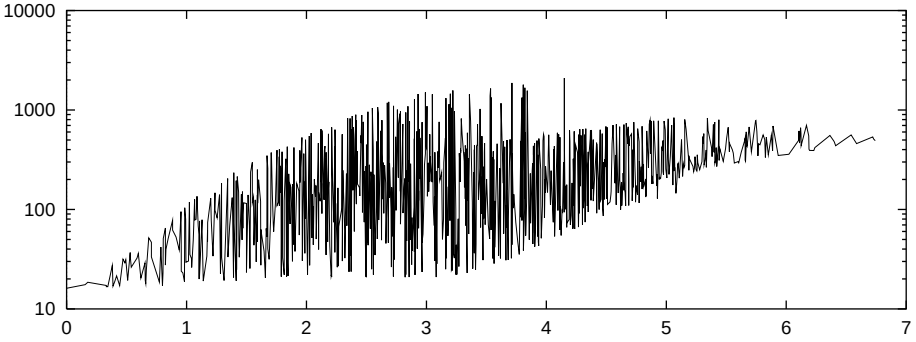
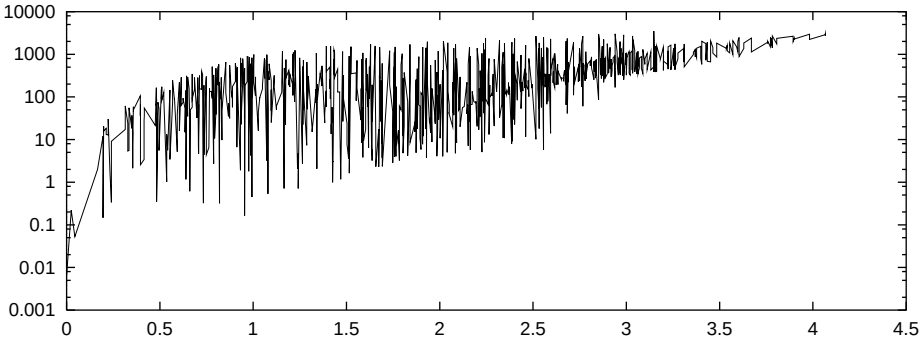


Figure 10: Rosenbrock Radius Plot



points were sampled from (-4,4), and the sampled minimum function value was 0.0254 at (1.015 1.015); the actual minimum is 0 at (1.0, 1.0). The radius plot shows substantial variation close to the minimum; Table 1 shows the actual values (the first point is the sampled minimum). In particular, note the $6^{\texttt{th}}$ value and its much larger value.

Table 1	
Rosenbrock Radius Plot Data	
radius	f(x)
0.0000	0.0062
0.0247	0.2222
0.0247	0.2176
0.0441	0.0529
0.1657	1.9931
0.1952	11.955
0.1952	0.1458

As can be seen perhaps more clearly in the table than in the plot, some points quite close the minimum are not much larger, while other are. Thus, like the Judge function, there might be a long valley or even multiple minima, and a robust or global optimizer might be appropriate.

Figure 11: Schmidt Radius Plot

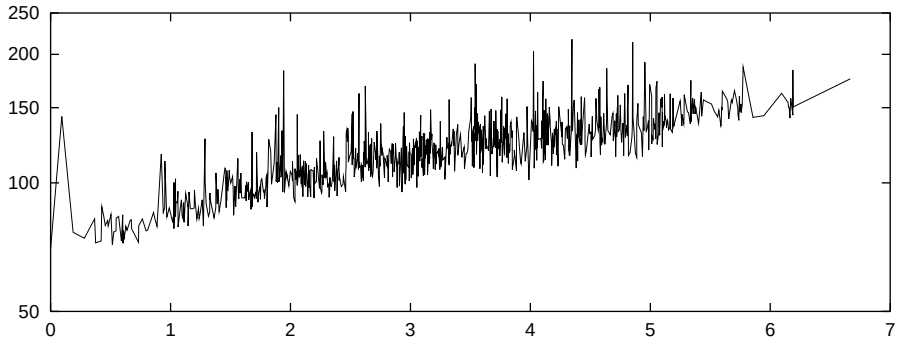


Figure 11 shows the radius plot for the Schmidt function. The minimum found this way, 70.15, is close to the actual minimum of 66.86. In this case, the radius plot is not terribly instructive; as the radius increases, the function's value rises fairly steadily. However, unlike the other radius plots, there are isolated values, which may indicate a surface with isolated peaks. This correlates well with the cross-sections, where relatively isolated peaks were sometimes observed. These peaks indicate that the function is non-quadratic, and thus a robust or global optimizer is appropriate.

For the Judge, Rosenbrock, and Schmidt functions, radius plots indicate that a global optimizer is appropriate since function values far from the sampled minimum were nearly the same size as the sampled minimum. This suggestion is in accord with the findings of cross-sections, so these methods are complementary.

Testing for Quadraticity

This final test is based on the idea that many conventional optimization algorithms assume the function to be optimized is roughly quadratic (a notable exception is the simplex method). While many algorithms allow for non-quadratic forms far from the optimum (NEOS: Network-Enabled Optimization System, 1997), many use the line-searches (basically move downhill) or a trust-region (which keeps the algorithm in a local neighborhood), common experience shows that they can fail. Thus, the test described here is a measure of the "non-quadraticity" of a function. Clearly, if a function is highly non-quadratic, then a robust or even a global optimizer is appropriate. If, on the other hand, the function is approximately quadratic, then a conventional algorithm is in order.

Clearly, there are many possible ways of measuring how quadratic a function might be. One option, of course, would be to check the adequacy of a second order Taylor series approximation. However, it can be difficult to generate an accurate numerical Hessian[6], and supplying analytical second derivatives is generally at least tedious and sometimes very difficult. Thus, a different method is desirable. The one chosen here relies on the idea that conjugate gradient algorithms theoretically converge in n steps *if* the function is quadratic; if the function is not quadratic, then they take more steps. This starts a conjugate gradient algorithm on a number of different starting points, and for each starting point, terminates the algorithm after n iterations. The resulting set of final points are compared; if the function is approximately quadratic, then the final points will be close together at the optimum; if the the function is far from quadratic, the final points will be far apart. The cg implementation of the Fletcher-Reeves conjugate gradient from Hager was used.

To measure how far apart the points are, the following measure was used: the distance from each point is compared to the point with the lowest function value, and the mean of these values is reported. Since this measure might be dependent upon the range of starting values (i.e. if the starting values are far apart, and the function is highly non-quadratic, the algorithm may hardly converge), the same measure is used for for the starting values, and then the ratio of the two is used. Thus, the measure of non-quadraticity is

$$\alpha = \frac{\sum_{j=1,j\neq j_{min}}^{m} \sum_{i=1}^{n} (x_{i,j}^{1} - x_{i,j_{min}}^{1})^2}{\sum_{j=1,j\neq j_{min}}^{m} \sum_{i=1}^{n} (x_{i,j}^{0} - x_{i,j_{min}}^{0})^2},$$

where n the number of variables in the function, m is the number of points, the superscript 0 refers to the starting points, the superscript 1 refers to the final points from cg, and $x_{i,j}$ is the j^{th} element of the i^{th} point. If the function is quadratic, then the α will be close to zero, and as the function becomes increasingly non-quadratic, its value will rise. This measure also has another, rougher interpretation—when optimizing a difficult function, some algorithms will run for extended times. This is likely due to the non-quadratic nature of the function, and α is a measure of this.

To illustrate this measure, consider the function

$$f(x_1, x_2) = x_1^2 + x_2^2 + \gamma x_1^2 x_2^2$$

For $\gamma = 0$, this function is quadratic, and as γ increases from 0, this function becomes increasingly dominated by a non-quadratic term. This measure puts a value to this problem.

Table 2 shows values of α and γ. Ten points from were chosen from the Bratley and Fox (1988) implementation of Faure (1982) on (-2,2). As expected, as γ increases, so does α; this suggests that this measure works as expected.

Table 2	
α Measure Example	
γ	α
0.00	1.45^{-8}
0.01	0.0267
0.02	0.0483
0.04	0.0696
0.08	0.0971
0.16	0.1284
0.32	0.1885
0.64	0.3096
1.28	0.3271
2.56	0.4015

Table 3 shows the results of this measure on the different test functions used here (in this case, 100 points from (-4,4) were selected).

Table 3	
Quadraticity Measure	
function	α
quadratic	1.48^{-8}
Judge	0.609
Rosenbrock	0.491
Schmidt	0.101

Since this is a new measure, it was tested with a different range of input values: (-10,10); the results are shown in Table 4. As can be seen, the results are similar to the smaller range, and suggests that this measure is somewhat robust. Finally, both show that the three non-quadratic test functions are indeed non-quadratic, although it is a bit surprising that the Schmidt function has such a low value given the difficult of optimizing it. Perhaps the difficulty comes the structure of the function near the optimum, and not far away—this measure is rather global in nature.

Table 4	
Quadraticity Measure	
function	α
quadratic	3.84^{-8}
Judge	0.620
Rosenbrock	0.613
Schmidt	0.152

Summary of the Methods

Finally, it is useful to examine the results function by function. For the quadratic function, all methods behaved exactly as expected: the radius plots showed a gradually rising function from the sample minimum, and α was approximately zero. For both the Judge and Rosenbrock functions, the cross-sections and radius plots indicated a multimodal or long, curved valley. For both of these, α was significantly greater than zero, indicating a non-quadratic component to the surface, which is of course true. Finally, the difficulty with the Schmidt function is well illustrated by the remarkable looking cross-section in Figure 7. The isolated peaks show here were picked up by the radius plot as well. Finally, α showed the expected value.

Conclusion

This paper introduced three different easy-to-use methods for analyzing a function prior to optimizing it. As economists use increasingly sophisticated models, they have begun to turn to more sophisticated algorithms to solve them. However, there is little guidance on the appropriate method to use. This paper is an attempt to provide some of that guidance. Three different, complementary methods were introduced here: cross-sections, radius plots, and a direct measure of quadraticity. The first two generate easy to interpret two dimensional plots, while the third measures how non-quadratic a function might be (this is a useful measure as many optimization algorithms assume the function is roughly quadratic). Taken together, these methods provide some aid to the researcher whose work involves difficult to optimize functions.

Notes

[1]Michael Veall kindly supplied the code and the data used in that study.

[2]Working with GNUPLOT, it took some time to get the Rosenbrock function looking "right."

[3]The actual problems turn out to be different: conventional algorithms cannot determine the global minimum for the Judge function, and many conventional algorithms have trouble with the Rosenbrock function.

[4]This optimum was found with simulated annealing.

[5]Such sampling of the function will likely provide good starting points for any optimization algorithm.

[6]Of course, many optimization algorithms generate a Hessian in the process of optimization, but this assumes that the algorithm terminates successfully, which is not a reasonable assumption here.

References

Bratley, P. and B. L. Fox (1988). Algorithm 659: Implementing Sobol's Quasirandom Sequence Generator. *ACM Transactions on Mathematical Software 14*(1), 88–100. <http://www.netlib.org/toms/659>.

Dorsey, R. E. and W. J. Mayer (1995). Genetic Algorithms for Estimation Problems with Multiple Optima, Nondifferentiability, and Other Irregular Features. *Journal of Business and Economic Statistics 13*(1), 53–66.

Faure, H. (1982). Discrépance de Suites Associées à un Système de Numération (en Dimension s). *Acta Arithmetica XLI*, 337–351.

Goffe, W. L. (1996). SIMANN: A Global Optimization Algorithm using Simulated Annealing. *Studies in Nonlinear Dynamics and Econometrics 1*(3). <http://mitpress.mit.edu/e-journals/SNDE/>.

Goffe, W. L., G. Ferrier, and J. Rogers (1994, Jan./Feb.). Global Optimization of Statistical Functions with Simulated Annealing. *Journal of Econometrics 60*(1/2), 65–99.

Hager, B. NAPAK. <http://www.netlib.org/napack/>. Dept. of Mathematics, University of Florida, Gainesville, FL 32611.

Hoffman, D. L. and P. Schmidt (1981). Testing the Restrictions Implied by the Rational Expectations Hypothesis. *Journal of Econometrics 15*, 265–87.

Judge, G. G., W. E. Griffiths, R. C. Hill, H. Lutkepohl, and T. Lee (1985). *The Theory and Practice of Econometrics* (2nd ed.). New York: John Wiley and Sons.

Moré, J. J., B. S. Garbow, and K. E. Hillstrom (1981). Testing Unconstrained Optimization Software. *ACM Transactions on Mathematical Sowftware 7*(1), 17–41.

NEOS: Network-Enabled Optimization System (1997). NEOS Guide. <http://www.mcs.anl.gov/home/otc/Guide/>.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery (1992). *Numerical Recipes in FORTRAN, The Art of Scientific Computing* (2nd ed.). New York: Cambridge University Press. <http://nr.harvard.edu/numerical-recipes>.

Rosenbrock, H. (1960). An Automatic Method of Finding the Greatest or Least Value of a Function. *Computer Journal 3*, 175–184.

Veall, M. (1990). Testing for a Global Maximum in an Econometric Context. *Econometrica 58*, 1459–65.