Chapter 10. Minimization or Maximization of Functions

10.0 Introduction

In a nutshell: You are given a single function f that depends on one or more independent variables. You want to find the value of those variables where f takes on a maximum or a minimum value. You can then calculate what value of f is achieved at the maximum or minimum. The tasks of maximization and minimization are trivially related to each other, since one person's function f could just as well be another's $-f$. The computational desiderata are the usual ones: Do it quickly, cheaply, and in small memory. Often the computational effort is dominated by the cost of evaluating f (and also perhaps its partial derivatives with respect to all variables, if the chosen algorithm requires them). In such cases the desiderata are sometimes replaced by the simple surrogate: Evaluate f as few times as possible.

An extremum (maximum or minimum point) can be either *global* (truly the highest or lowest function value) or *local* (the highest or lowest in a finite neighborhood and not on the boundary of that neighborhood). (See Figure 10.0.1.) Finding a global extremum is, in general, a very difficult problem. Two standard heuristics are widely used: (i) find local extrema starting from widely varying starting values of the independent variables (perhaps chosen quasi-randomly, as in §7.7), and then pick the most extreme of these (if they are not all the same); or (ii) perturb a local extremum by taking a finite amplitude step away from it, and then see if your routine returns you to a better point, or "always" to the same one. Relatively recently, so-called "simulated annealing methods" (§10.9) have demonstrated important successes on a variety of global extremization problems.

Our chapter title could just as well be *optimization*, which is the usual name for this very large field of numerical research. The importance ascribed to the various tasks in this field depends strongly on the particular interests of whom you talk to. Economists, and some engineers, are particularly concerned with *constrained optimization*, where there are *a priori* limitations on the allowed values of independent variables. For example, the production of wheat in the U.S. must be a nonnegative number. One particularly well-developed area of constrained optimization is *linear programming*, where both the function to be optimized and the constraints happen to be linear functions of the independent variables. Section 10.8, which is otherwise somewhat disconnected from the rest of the material that we have chosen to include in this chapter, implements the so-called "simplex algorithm" for linear programming problems.

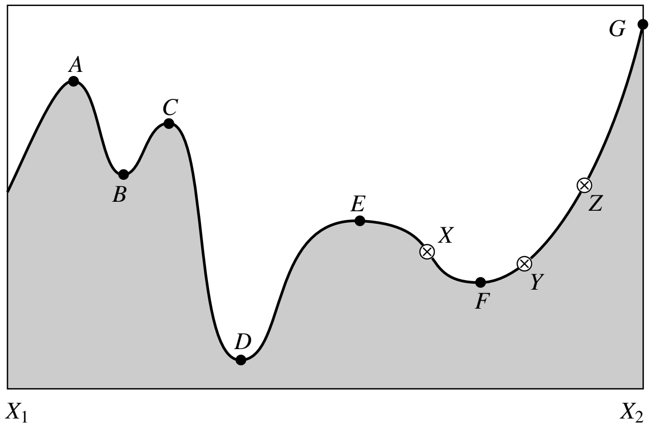

Figure 10.0.1. Extrema of a function in an interval. Points A, C, and E are local, but not global maxima. Points B and F are local, but not global minima. The global maximum occurs at G, which is on the boundary of the interval so that the derivative of the function need not vanish there. The global minimum is at D. At point E, derivatives higher than the first vanish, a situation which can cause difficulty for some algorithms. The points X, Y, and Z are said to "bracket" the minimum F, since Y is less than both X and Z.

One other section, §10.9, also lies outside of our main thrust, but for a different reason: so-called "annealing methods" are relatively new, so we do not yet know where they will ultimately fit into the scheme of things. However, these methods have solved some problems previously thought to be practically insoluble; they address directly the problem of finding global extrema in the presence of large numbers of undesired local extrema.

The other sections in this chapter constitute a selection of the best established algorithms in unconstrained minimization. (For definiteness, we will henceforth regard the optimization problem as that of minimization.) These sections are connected, with later ones depending on earlier ones. If you are just looking for the one "perfect" algorithm to solve your particular application, you may feel that we are telling you more than you want to know. Unfortunately, there is *no* perfect optimization algorithm. This is a case where we strongly urge you to try more than one method in comparative fashion. Your initial choice of method can be based on the following considerations:

- You must choose between methods that need only evaluations of the function to be minimized and methods that also require evaluations of the derivative of that function. In the multidimensional case, this derivative is the gradient, a vector quantity. Algorithms using the derivative are somewhat more powerful than those using only the function, but not always enough so as to compensate for the additional calculations of derivatives. We can easily construct examples favoring one approach or favoring the other. However, if you *can* compute derivatives, be prepared to try using them.
- For one-dimensional minimization (minimize a function of one variable) *without* calculation of the derivative, bracket the minimum as described in §10.1, and then use *Brent's method* as described in §10.2. If your function has a discontinuous second (or lower) derivative, then the parabolic

interpolations of Brent's method are of no advantage, and you might wish to use the simplest form of *golden section search*, as described in §10.1.

- For one-dimensional minimization *with* calculation of the derivative, §10.3 supplies a variant of Brent's method which makes limited use of the first derivative information. We shy away from the alternative of using derivative information to construct high-order interpolating polynomials. In our experience the improvement in convergence very near a smooth, analytic minimum does not make up for the tendency of polynomials sometimes to give wildly wrong interpolations at early stages, especially for functions that may have sharp, "exponential" features.

We now turn to the multidimensional case, both with and without computation of first derivatives.

- You must choose between methods that require storage of order N^2 and those that require only of order N, where N is the number of dimensions. For moderate values of N and reasonable memory sizes this is not a serious constraint. There will be, however, the occasional application where storage may be critical.
- We give in §10.4 a sometimes overlooked *downhill simplex method* due to Nelder and Mead. (This use of the word "simplex" is not to be confused with the simplex method of linear programming.) This method just crawls downhill in a straightforward fashion that makes almost no special assumptions about your function. This can be extremely slow, but it can also, in some cases, be extremely robust. Not to be overlooked is the fact that the code is concise and completely self-contained: a general N-dimensional minimization program in under 100 program lines! This method is most useful when the minimization calculation is only an incidental part of your overall problem. The storage requirement is of order N^2, and derivative calculations are not required.
- Section 10.5 deals with *direction-set methods*, of which *Powell's method* is the prototype. These are the methods of choice when you cannot easily calculate derivatives, and are not necessarily to be sneered at even if you can. Although derivatives are not needed, the method does require a one-dimensional minimization sub-algorithm such as Brent's method (see above). Storage is of order N^2.

There are two major families of algorithms for multidimensional minimization *with* calculation of first derivatives. Both families require a one-dimensional minimization sub-algorithm, which can itself either use, or not use, the derivative information, as you see fit (depending on the relative effort of computing the function and of its gradient vector). We do not think that either family dominates the other in all applications; you should think of them as available alternatives:

- The first family goes under the name *conjugate gradient methods*, as typified by the *Fletcher-Reeves algorithm* and the closely related and probably superior *Polak-Ribiere algorithm*. Conjugate gradient methods require only of order a few times N storage, require derivative calculations and one-dimensional sub-minimization. Turn to §10.6 for detailed discussion and implementation.

- The second family goes under the names *quasi-Newton* or *variable metric* methods, as typified by the *Davidon-Fletcher-Powell (DFP)* algorithm (sometimes referred to just as *Fletcher-Powell*) or the closely related *Broyden-Fletcher-Goldfarb-Shanno (BFGS)* algorithm. These methods require of order N^2 storage, require derivative calculations and one-dimensional sub-minimization. Details are in §10.7.

You are now ready to proceed with scaling the peaks (and/or plumbing the depths) of practical optimization.

CITED REFERENCES AND FURTHER READING:

Dennis, J.E., and Schnabel, R.B. 1983, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* (Englewood Cliffs, NJ: Prentice-Hall).

Polak, E. 1971, *Computational Methods in Optimization* (New York: Academic Press).

Gill, P.E., Murray, W., and Wright, M.H. 1981, *Practical Optimization* (New York: Academic Press).

Acton, F.S. 1970, *Numerical Methods That Work*; 1990, corrected edition (Washington: Mathematical Association of America), Chapter 17.

Jacobs, D.A.H. (ed.) 1977, *The State of the Art in Numerical Analysis* (London: Academic Press), Chapter III.1.

Brent, R.P. 1973, *Algorithms for Minimization without Derivatives* (Englewood Cliffs, NJ: Prentice-Hall).

Dahlquist, G., and Bjorck, A. 1974, *Numerical Methods* (Englewood Cliffs, NJ: Prentice-Hall), Chapter 10.

10.1 Golden Section Search in One Dimension

Recall how the bisection method finds roots of functions in one dimension (§9.1): The root is supposed to have been bracketed in an interval (a, b). One then evaluates the function at an intermediate point x and obtains a new, smaller bracketing interval, either (a, x) or (x, b). The process continues until the bracketing interval is acceptably small. It is optimal to choose x to be the midpoint of (a, b) so that the decrease in the interval length is maximized when the function is as uncooperative as it can be, i.e., when the luck of the draw forces you to take the bigger bisected segment.

There is a precise, though slightly subtle, translation of these considerations to the minimization problem: What does it mean to *bracket* a minimum? A root of a function is known to be bracketed by a pair of points, a and b, when the function has opposite sign at those two points. A minimum, by contrast, is known to be bracketed only when there is a *triplet* of points, $a < b < c$ (or $c < b < a$), such that $f(b)$ is less than both $f(a)$ and $f(c)$. In this case we know that the function (if it is nonsingular) has a minimum in the interval (a, c).

The analog of bisection is to choose a new point x, either between a and b or between b and c. Suppose, to be specific, that we make the latter choice. Then we evaluate $f(x)$. If $f(b) < f(x)$, then the new bracketing triplet of points is (a, b, x);