SWIFT--A NEW TYPE OF FORTH APPLICATION

John P. Mullen
Department of Industrial Engineering
Iowa State University
Ames, Iowa 50011

INTRODUCTION

Nonlinear Optimization

Nonlinear optimization (NLO) is the search for the best solution within a set of feasible solutions. There are no restrictions on the type of constraints that determine the feasible region or on the ordering function that is used to compare feasible solutions. Search techniques, analogous to groping for the highest spot in a field by night, are often used to find solutions to this type of problem. Specific algorithms differ in the types of problems they can solve, in how efficient they are, and in what information they require.

Because of the variety of such problems, no single algorithm can be guaranteed to work at all times. In addition, results often depend on the starting point of the search and on the working parameters of the chosen algorithm. Thus, the analyst must often apply each of several techniques many times in a trial and error process to find an optimal point.

In addition, the form of the constraint functions and the ordering function may be arbitrarily complex. Thus the analyst must be able to specify them in a general way, usually by using external functions or subroutines. Finally, since many calculations may be required for each point and since many points have to be evaluated, high execution speed is a necessity [1].

A Choice of Computer Language

Because of the power, flexibility and speed required, most such analysis has been done in FORTRAN. This is a good choice if one is using a large computer, but microcomputers are often used to solve moderately large problems. Current implementations of FORTRAN do not allow one to make full use of the microcomputer's interactive nature, and much of the potential advantage in this approach is lost. Forth is a good alternative to FORTRAN. It is fast, modular, and interactive. In addition, the advent of the mathematic coprocessor has made needed functions available and floating-point operations efficient.

THE FORTH NLO SYSTEM

The thrust of this project is to develop an interactive environment in which an analyst can use Forth to solve nonlinear optimization problems effectively. To this end, three major components have been developed: ORDERED-POINTS, SIMPLEX-TOOLS, and GRAPH-TOOLS. In addition, one pattern-search algorithm called SWIFT is currently being tested. An initial version of this system was developed on a

Commodore-64 [2] using SuperFORTH-64 [3]. Once the utility of this approach was demonstrated, the system was adapted to an IBM-AT [4] using PolyFORTH [5].

ORDERED-POINTS

Optimization algorithms have several common characteristics: possible solutions can be represented as points in N-space; new points are generated from linear combinations of existing points; and solutions are evaluated in terms of functions of their coordinate values. The Forth component ORDERED-POINTS conceptualizes these characteristics.

The word PVARIABLE defines an instance of the data structure {lf:y(X):x1 x2 ... xn} where the xi represent the coordinates of the point X, y(X) is the value of the ordering function at that point, and the flag indicates whether or not that value is valid. Other words facilitate the manipulation of these data elements. For example,

```
... X1 X2  2DUP  P< IF PSWAP  ELSE 2DROP  THEN ...
```

compares the two points X1 and X2 on the basis of the ordering function and interchanges them if X1 is less than X2.

SIMPLEX-TOOLS

A simplex is a N+1 straight-sided figure in N-space. For example, a simplex in 2-space is a triangle. Pattern-search algorithms work by generating a series of simplexes and examining the value of the ordering function at their vertices. They first attempt to include the optimum point within a simplex and then to reduce the size of the simplex until it is arbitrarily small. This boxes in the result. Each algorithm differs in the way it performs the search and in the way the ordering function is constructed.

Because several algorithms deal with simplexes, SIMPLEX-TOOLS was developed in which POINT, an array of PVARIABLEs, contains the coordinates and function values for the vertices of a simplex. This component also contain words that perform other subfunctions common in pattern-search algorithms. For example, PH?! determines which vertex of the simplex has the highest function value. Another word, .SIMPLEX displays the entire simplex in tabular form. SIMPLEX-TOOLS facilitates the manipulation of NLO problems so well that it is possible to solve simple problems without resorting to automatic algorithms. It is also useful for examining points in the region of an algorithm's solution.

GRAPH-TOOLS

Two-dimensional problems may be displayed as graphs. For example, pattern searches appear as a series of triangles such as in Fig. 1. This component assists in debugging algorithms and is also useful as an aid to help explain to students how a given algorithm works.

SWIFT

SWIFT was chosen as the first algorithm to implement because it often yields good results, it does not require one to compute partial derivatives, and it is typical of pattern-search algorithms. The SWIFT

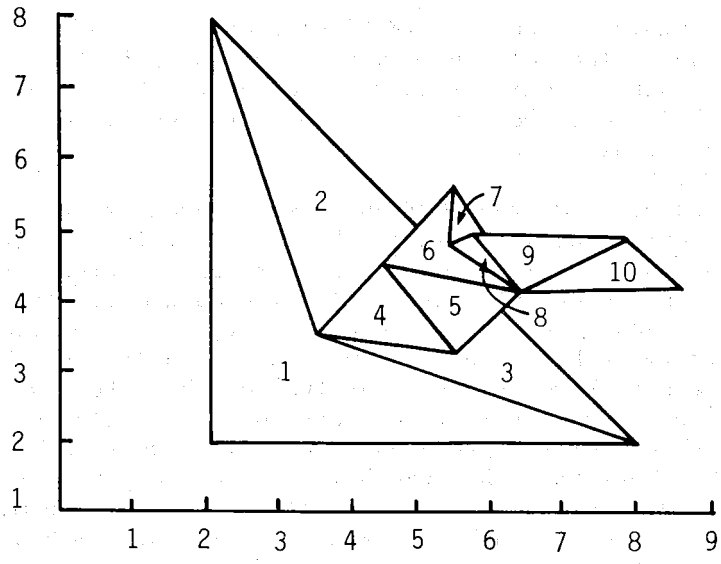

Figure 1. Graph of ten consecutive simplexes.

algorithm combines the optimization function and the constraint functions into a single ordering function of the form

$$y(\underline{X}) = f(\underline{X}) + w * \left(\sum_{z=1}^{N} gi'(\underline{X}) \right)$$

where $f(\underline{X})$ is the original function to be minimized, and each $gi'(\underline{X})$ represents how much the point $\underline{X}$ violates the ith constraint. Initially, w, the weighting factor, is a small number, which causes the constraint functions to be almost ignored; but when the algorithm narrows in on the best point, w is made progressively larger until it is so large that the solution is virtually guaranteed to be feasible.

Implementation

Thanks to the ORDERED-POINTS and SIMPLEX-TOOLS, the implementation of SWIFT in Forth was rather straightforward. The component SWIFT is a set of words that allows an analyst to set up a problem, to control the SWIFT algorithm, and to examine its results. This component's development involved the mechanics of how the analyst invoked the proper Forth screens in addition to the algorithm itself.

The use of software modules [6] allows one to load the entire system with a phrase such as "5 CONSTANT #DIM SWIFT." However, there is a chicken-and-egg problem in that the components are needed to facilitate the definition of the ordering function, but the ordering function must be known in order to define the algorithm. This was resolved by establishing the Euclidean norm as a default function with vectored execution [7]. Another function can be designated by storing its CFA in the variable 'YCALC.

Performance

Identical problems were solved using both FORTRAN and Forth versions of SWIFT on the IBM-AT. Although Forth execution speeds currently are slightly slower than those for FORTRAN, they are compar-

able. For example, in one case FORTRAN took 19 seconds while Forth took 24 seconds. The major disadvantage of using Forth seems to be its unusual language, which requires one to restate algebraic expressions into their Forth equivalents. However, one could argue that this is a matter of taste.

On the other hand, the FORTRAN version requires at least two full minutes to compile and link, while the Forth version requires only four seconds. In addition, it is far more difficult working with the FORTRAN program than the Forth one. Typically, I needed multiple printouts of modules in FORTRAN in order to coordinate my efforts whereas I was comfortable working with screen displays only in PolyFORTH. This is a great advantage of Forth since an analyst may have to try several versions of the problem statement in order to find a solution. In addition, the FORTRAN version produces a specific result whereas the Forth version is interactive. This capability allows one to test constraint and ordering functions as they are developed, to examine many aspects of the solution, to examine the characteristics of neighboring points, or to use the result as a starting point for some other algorithm.

CURRENT WORK

Although there is little doubt that the interactivity of Forth gives it a clear advantage over FORTRAN in this area, there is still much work to do. Currently emphasis is on improving the utility and speed of the words in all three system components. A major problem that affects transportability is dealing with the variety of floating point implementations. Once the underlying system is improved, the next phase will be to implement other algorithms.

REFERENCES

1. Tillman, F., Hwang, C. and Kuo, W., Optimization of Systems Reliability, Marcel Dekker, Inc., New York, 1980, Chap. 2.
2. Commodore Business Machines, Norristown, PA.
3. Parsec Research, Drawer 1766-P, Fremont, CA.
4. International Business Machines Corp., Box 1328-C, Boca Raton, FL.
5. Forth, Inc., 2309 Pacific Coast Highway, Hermosa Beach, CA.
6. Furman, Alan, "Module Management System," presented at the 1985 Rochester Forth Conference, June 1985 (unpublished).
7. Brodie, Leo, Starting FORTH, Prentice-Hall, Inc., Englewood Cliffs, 1981, pp. 217-219.