Comparison of NLP Search Methods in Nonlinear Optimization Using MAPLE

Dr. William P. Fox and Dr. William H. Richardson
Department of Mathematics
Francis Marion University
Florence, SC 29501
wfox@fmarion.edu, wrichardson@fmarion.edu

INTRODUCTION

Teaching a mathematical modeling course or a multivariable variable calculus course should introduce numerical search algorithms in order to solve optimization problems. It is important to stress the numerical search procedure's concepts without requiring the degree of sophistication required for a graduate course in nonlinear programming. Our courses attempt to tie the nonlinear optimization concepts back to their fundamental concepts taught in a multivariable calculus course. Many students have problems fathoming the problem or understanding the concepts. This is due to spending all of their valuable time figuring out how to perform the required iterative computations by pencil and paper. It is in this area, iterative computations, that the computer can be most useful and powerful.

The practicality of using the computer suits our modern methods of teaching mathematics. With advanced spreadsheets like EXCEL and Computer Algebra Systems (CAS) such as MAPLE, we have been moving towards a more *lean* and *lively* mathematics program. Our program for mathematics and computer science majors consists of single variable calculus, multivariable calculus, linear algebra and elective courses. The students become more conversant with MAPLE and spreadsheet packages, such as EXCEL, as they take more courses. In our electives and more advanced courses, we can readily take advantage of our student abilities with the computer and concentrate on the mathematical procedures. Additionally, all our majors take a computer science course, which gives many of them a programming experience in a basic language such as "C++", FORTRAN, or Visual-BASIC.

Gradient Search Methods

Suppose we want to solve the following unconstrained nonlinear programming problem (NLP):

$$Max\ z = f(x_1, x_2, x_3, \dots, x_n) \qquad (1)$$

In calculus, if equation (1) is a concave function, then the optimal solution (if there is one) will occur at a stationary point x^* having the following property:

$$\frac{\partial f(x^*)}{\partial x_1} = \frac{\partial f(x^*)}{\partial x_2} = \dots = \frac{\partial f(x^*)}{\partial x_n} = 0 \qquad (2)$$

In many problems, equation (2) is not an easy task to solve to find the stationary point. Thus, the Method of Steepest Ascent (maximization problems) and the Method of Steepest Descent (minimization problems) offers an alternative to finding an approximate stationary point. We will continue to discuss the gradient method for the Steepest Ascent.

Given a function, like the one in figure 1, assume that we want to find the maximum point of the function. If we started at the bottom of the hill then we might proceed by finding the gradient. The gradient is the vector of the partial derivatives that points "up the hill". We define the gradient vector as follows:

$$\nabla f(x) = [\frac{\partial f(x^*)}{\partial x_1}, \frac{\partial f(x^*)}{\partial x_2}, \ldots, \frac{\partial f(x^*)}{\partial x_n}]$$

If we were lucky, the gradient would point all the way to the top of the function but the contours of functions do not always cooperate and rarely do. Thus, the gradient "points up hill" but for how far? We need to find the distance along the gradient for which to travel that maximizes the height of the function in that direction. From that new point, we re-compute a new gradient vector to find a new direction that "points up hill." We continue this method until we get to the top of the hill.

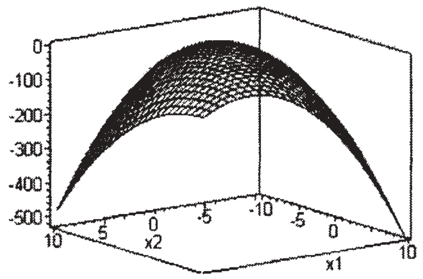


Figure 1. The Function, Z= f(x1,x2)= $2x_1x_2 + 2x_2 - x_1^2 - 2x_2^2$

To find a maximum solution to given a multivariable unconstrained function, ***f(x)***

INPUT:	starting point $\mathbf{x_0}$; tolerance, t
OUTPUT:	Approximate $\boldsymbol{x^*}$, and $\boldsymbol{f(x^*)}$
Step 1.	Initialize the tolerance, $t > 0$.
Step 2.	Set $\mathbf{x}=\mathbf{x_0}$ and define the gradient at that point. $\nabla f(x_0)$
Step 3.	Calculate the maximum of the new function $f(x_i + t_i\ \nabla f(x_i))$, where $t_i \geq 0$, by finding the value of t_i.
Step 4.	Find the new x_i point by substituting t_i into $x_{i+1} = x_i + t_i\ \nabla f(x_i)$
Step 5.	If the length (magnitude) of $\mathbf{x}$, defined by $\|\mathbf{x}\| = (x_1^2 + x_2^2 + \ldots + x_n^2)^{\frac{1}{2}}$, is less than the tolerance specified, then

continue.

```
             Otherwise, go back to Step 3.
Step 6.      Use $x^*$ as the approximate stationary point and compute, $f(x^*)$, the
estimated
             maximum of the function.
STOP
```

Figure 2. Steepest Ascent Algorithm

Newton's Method

An alternative search method is the Newton-Raphson numerical method illustrated in two variables. This numerical method appears to do a more efficient and faster job in converging to the near optimal solution. It is an iterative root finding technique using the partial derivatives of the function as the new system of equations. The algorithm uses Cramer's Rule to find the solution of the system of equations.

Newton's Method for multivariable optimization searches is based on Newton's single variable algorithm for finding the roots and Newton-Raphson Method for finding roots of the first derivative, given a x_0, iterate $x_{n+1} = x_n - f'(x_n)/f''(x_n)$ until $|x_{n+1} - x_n|$ is less than some small tolerance. In several variables, we may use a vector $\mathbf{x_0}$, or two variables, (x_0,y_0). The algorithm is expanded to include partial derivatives with respect to each variable's dimension. In two variables (x,y), this would yield a system of equations where F is the derivative of $f(x,y)$ with respect to x and G is the derivative of $f(x,y)$ with respect to y. Thus, we need to find both $F=0$ and $G=0$ simultaneously.

This yields a matrix equation $\sum_{j=1}^{N} \alpha_{ij}\delta x_j = \beta_i$, where

$$\alpha_{ij} = \frac{\partial f_i}{\partial x_i}, \beta_i = -f_i.$$

The matrix equation can be solved by LU decomposition or in the case of a 2 x 2 by Cramer's Rule. The corrections are then added to the solution vector

$$x_i^{new} = x_i^{old} + \delta x_i, i = 1,\ldots N$$

and iterated until it converges within a tolerance.

```
INPUT: x(0), y(0), N, Tolerance
OUTPUT: x(n), y(n)
Step 1. For n= 1 to N do
        Step 2. Calculate the new estimate for x(n) and y(n) as follows:
```

```
        $\frac{\partial F}{\partial x}(x(n-1), y(n-1) \to q$
        $\frac{\partial F}{\partial y}(x(n-1), y(n-1) \to r$
        $\frac{\partial G}{\partial x}(x(n-1), y(n-1) \to s$
        $\frac{\partial G}{\partial x}(x(n-1), y(n-1) \to t$
        $-F(x(n-1), y(n-1)) \to u$
        $-G(x(n-1), y(n-1)) \to v$
        $qt - rs \to D$
        $x(n-1) + (ut - vr)/D \to x(n)$
        $y(n-1) + (qv - su)/D \to y(n)$
**Step 3**. If $((x(n)-x(n-1))^2 + (y(n)-y(n-1))^2)^{1/2}$ < tolerance,
        Then **Stop**
        Else , Go back to Step 2.
**STOP**
```
Figure 3. Pseudo-Code for Newton's Method

Quasi-Newton Conjugate Directions

Another method is the Method of Davidon, Fletcher, and Powell (1963). The steps are displayed in Figure 4.

Initialization Step: Let $e>0$ be the termination scalar. Choose an initial point $\boldsymbol{x}$ and an initial symmetric negative definite matrix $\boldsymbol{D}$. Let $y_1=x_1$, let $k=j=1$ and go to main step.
Main Step.
Step 1. If $\|\nabla f(y_j)\| < e$ stop; otherwise let $\mathbf{d_j} = -\mathbf{D}\,\nabla \mathbf{f(y_j)}$. Maximize $f(y_j+\lambda_j d)$ for the optimal λ_j, $\lambda_j \geq 0$. Let $y_j = y_j+\lambda_j d$. If $j < n$, go to step 2. If $j=n$, let $y_1=x_{k+1}=y_{n+1}$, replace k by k+1, let j=1, and repeat step 1.
Step 2. Construct $\boldsymbol{D_{j+1}}$

$$D_{j+1} = D_j + \frac{p_j p_j^T}{p_j^T q_j} - \frac{D_j q_j q_j^T D_j}{q_j^T D_j q_j}$$

$$p_{j=\lambda_j d_j = y_{j+1} - y_j}$$

$$q_j = \nabla f(y_{j+1}) - \nabla f(y_j)$$

Replace j by j+1 and repeat step 1.

Figure 4. Method of Davidon-Fletcher-Powell

EXAMPLE SEARCH USING MAPLE

Maximize $f(x_1,x_2) = 2x_1x_2 + 2x_2 - x_1^2 - 2x_2^2$

Gradient Method

```
> f:=2*x1*x2+2*x2-x1^2-2*x2^2;
```

$$f := 2\,x1\,x2 + 2\,x2 - x1^2 - 2\,x2^2$$

```
> UP(100,.01,0,0,f);

Approximate Solution:   (    .9922,    .9961)
                     Maximum Functional Value:                      1.0000
                  Number gradient evaluations:                          17
                  Number function evaluations:                          16
```

Newton's Method

```
> f2:=(x1,x2)->2*x1+2-4*x2;
```

$$f2 := (x1, x2) \to 2\,x1 + 2 - 4\,x2$$

```
> Steepest(f,f1,f2,100,.05,0,0);#example 1

Hessian: [    -2.000     2.000   ]
         [     2.000    -4.000   ]
eigenvalues:  -5.236      -.764
pos def: false
new x=   1.000    new y=   1.000

final new x=   1.000    final new y=   1.000
final fvalue is    1.000

It converges to the point (1,1) after 2 iterations.
```

Method of Davidon-Fletcher-Powell

$$oldx1 := 1.000000000$$

$$oldx2 := 1.000000000$$

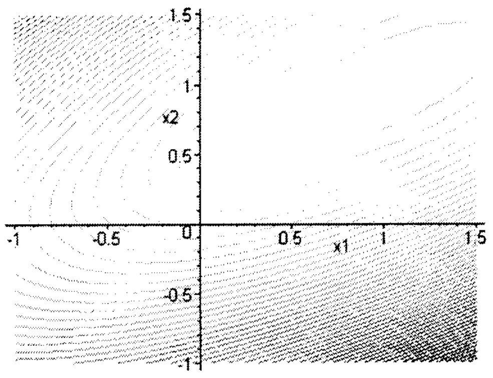
$$oldy1 := 1.000000000$$

$$oldy2 := 1.000000000$$

$$1.000000000, 1.000000000$$

```
Converges to (1,1) in 1 iteration.
```

The Hessian matrix, $\begin{bmatrix} -2 & 2 \\ 2 & -4 \end{bmatrix}$, is negative definite so the point x* is a maximum.


x1
x2
1.5
1
0.5
0
-0.5
-1
-1
-0.5
0.5
1
1.5