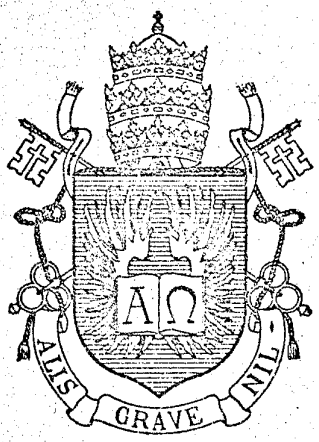

PUC

Series: Monographs in Computer Science
and Computer Applications

Nº 1/70

A TWO - LEVEL NEWTON METHOD FOR FUNCTION OPTIMIZATION

by
Antonio Luz Furtado

Computer Science Department - Rio Datacenter

A TWO - LEVEL NEWTON METHOD FOR FUNCTION OPTIMIZATION

ANTONIO LUZ FURTADO
COMPUTER SCIENCE DEPARTMENT
PUC - RIO DE JANEIRO

Abstract:

The Newton method is used iteratively at two levels to determine an unrestricted local minimum of a general function $f(x_1 , x_2 , \dots , x_n)$.

At the first level a vector $\Delta = (\delta_1, \delta_2, \dots , \delta_n)$ is computed and at the second level a scalar α is obtained to minimize

$$f (x_1 + \alpha \ \delta_1 , x_2 + \alpha \ \delta_2 , \dots , x_n + \alpha \ \delta_n).$$

Both levels involve a preliminary first and second order partial differentiation and this is performed symbolically by the computer.

Run-time compilation translates the resulting derivatives into machine code. This ensures the high speed of the repeated numerical evaluations.

The sub-program package to implement the algorithm was written in FORTRAN IV plus the SLIP (1) system, but it is feasible in any language or system supporting formula manipulation capabilities (2).

1 - The Method:

We are given an initial guess $X^{(0)}$

The first level is an iterative process, consisting of finding, at each iteration k, a direction $\Delta^{(k)}$ leading to a point

$$X^{(k + 1)} = X^{(k)} + \Delta^{(k)} \text{ nearer to the minimum.}$$

$\Delta^{(k)}$ is the solution of the system of linear equations:

$$H^{(k)} \cdot \Delta^{(k)} = - G^{(k)} \quad \text{(generalized Newton formula)},$$

where the elements g_i of vector G and the elements h_{ij} of matrix H are given by:

$$g_i = \frac{\partial f (x_1 , x_2 , \ldots , x_n)}{\partial x_i} ; \; i = 1, n$$

$$h_{ij} = \frac{\partial g_i}{\partial x_j} ; \; i = 1, n; \; j = i, n$$

However, if the inner product $(G^{(k)} , - \Delta^{(k)})$ is negative, then the matrix $H^{(k)}$ is not positive definite and the direction $\Delta^{(k)}$ would not lead to a minimum; instead we make $\Delta^{(k)} = - G^{(k)}$, which leads to a first order method (the gradient method).

Having chosen an adequate $\Delta^{(k)}$, the second level (an iterative process within each iteration of the first), consists of a minimization along that direction $\Delta^{(k)}$.

This is to say that we look for a scalar α that minimizes

$$f^* = f (x_1^{(k)} + \alpha^{(k,m)} \cdot \delta_1^{(k)} ,$$

$$x_2^{(k)} + \alpha^{(k,m)} \cdot \delta_2^{(k)} , \ldots ,$$

$$x_n^{(k)} + \alpha^{(k,m)} \cdot \delta_n^{(k)}$$

where the $\delta_i^{(k)}$'s are the components of Δ.

At iteration m = 0 we take $\alpha^{(k,m)} = 1$ if matrix H has proved to be positive definite, otherwise we take $\alpha^{(k,m)} = 0$.

Subsequent iterations m are performed by solving the linear equation:

$$v^{(m)} \cdot \beta^{(m)} = - u^{(m)} \quad \text{(Newton formula for one independent variable)},$$

where:

$$u = \frac{\partial f^*}{\partial \alpha}$$

$$v = \frac{\partial u}{\partial \alpha}$$

and a better value $\alpha^{(k,m+1)} = \alpha^{(k,m)} + \beta^{(m)}$ is obtained.

If $v^{(m)}$ is not positive we simply take $\beta^{(m)} = - \beta^{(m)}$

If the value of f* is greater at the new point $X^{(k)} + \alpha^{(k,m)} \cdot \Delta^{(k)}$, then the convergence is not stable, but this is not too bad because we can assert that the minimum along the direction $\Delta^{(k)}$ is bounded. We then make $\alpha^{(k,m+1)} = \alpha^{(k,m)} - \beta^{(m)}/2$, which again leads to a first order method (the bissection method).

At the first level the process degenerates if $H^{(k)}$ is singular. In this case, if $G^{(k)}$ does not vanish, the gradient method may be used; if $G^{(k)}$ vanishes a new point $X^{(k+1)}$ could be obtained by adding $\alpha' \Delta^{(k-1)}$ (where α' is an arbitrarily small positive number) to $X^{(k)}$ until such conditions no longer arise.

A similar procedure could be adopted if, at the second level, $|M| \neq 0$. In this case, we might repeatedly add an arbitrarily small β to $\alpha^{(k,m)}$ in order to eliminate that condition.

The termination criteria are:

- for the second level, $| \beta^{(m)} | < \xi$

  for some small ξ.

- for the first level, $|| \alpha^{(k,m)} \Delta^{(m)} || < \xi$,

using any suitable definition of norm of a vector.

A comparison with other methods would show that two-level minimization is a common practice; the second level uses generally Fibonacci or golden rule search, or quadratic or Davidon's cubic interpolation.

Also, the Fletcher and Powell method [3] adopts the same strategy of resorting to the gradient method whenever the matrix is not positive definite.

2 - The Implementation:

In order to implement the method the following programming capabilities are required.

a. conversion of an expression, given as data, into some canonical form, e.g. prefix Polish form [4];

b. symbolic differentiation, together with of followed by some simplification of the resulting formulas, to avoid their frequently explosive growth [5];

c. some way of relating symbols in a formula with the corresponding variables in the program; this could be done by associating with a symbol, say x_3, the machine address of a variable of the same name; this makes the possibly varying value x_3 available during numeric evaluation [6];

d. run-time compilation into machine language, which is obtained by OR-ing operation codes to machine addresses or function entry points, and by storing such code into an array (4);

e. evaluation by a calling type transfer to those arrays:

f. a conventional routine (e.g. Gauss-Jordan with pivot optimization and a test for singularity) to solve the n^{th} order systems of linear equations.

Availability of a list-processing system embedded in a computation-oriented language provides practical mechanisms for symbolic manipulation without incurring the generally poor numerical performance of purely list-processing languages.

On the other hand, run-time compilation would be even more convenient for a class of problems not discussed here, in which symbolic and numerical phases alternate several times. For such problems the well-known practice of punching FORTRAN statement cards as the output of a symbolic phase program and then introducing them into the numeric phase program would be operationally inconvenient.

3 - Some results:

- functions:

a) $2\,(x_1 - x_2)^2 + (x_1 + x_2)^2$

b) $100\,(x_2 - x_1^2)^2 + (1 - x_1)^2$ - Rosenbrock function

c) $(x_1 - 10x_2)^2 + 5\,(x_3 - x_4)^2 + (x_2 - 2x_3)^2 + 10\,(x_1 - x_4)^2$

d) $(x_1 - 2)^2 + x_2^2 + 1.5\,x_3^2 + (x_2 - x_4)^2 + (x_5 - x_3)^2$

Function

e) $(1 - x_1)^2 + (x_1 - 10x_2)^2 + (x_1 - 100x_3)^2 + (x_3 - 10x_4)^2 + (x_3 - 100x_5)^2 +$

a

$+ (x_5 - 10x_6)^2 + (x_5 - 100x_7)^2 + (x_7 - 10x_8)^2$

b

f) $(1 - x_1)^2 + (2 - x_1 x_2)^2 + (3 - x_2 x_3)^2 + (4 - x_3 x_4)^2 + (5 - x_4 x_5)^2 +$

c

$+ (6 - x_5 x_6)^2 + (7 - x_6 x_7)^2 + (8 - x_7 x_8)^2 + (9 - x_8 x_9)^2 + (10 - x_9 x_{10})^2$

d

g) $\sum_{i=1}^{3} (y_1 - c_1 e^{c_2 x_1})^2$ - a least squares curve fitting problem

e

f

h) $(x_1 - 0.7 \sin(x_1) - 0.2 \cos(x_2))^2 + (x_2 - 0.7 \cos(x_1) + 0.2 \sin(x_2))^2$ -

g

h an application of least squares to the solution of a system of two non-linear equations.

function	initial guess	minimum
a	(1.5 , 0.5)	0
b	(-1.2 , 1)	0
c	(3, -1, 0,1)	0
d	(1, 1, 1, 1, 1)	0
e	(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)	0
f	(1,1,1,1,1,1,1,1,1,1)	0
g	(1, 1)	10^{-2}
h	(0, 0)	0

results in an IBM - 7044, 32k, 36 bit words, single precision, underflow trap suppressing routine

function	iterations	minimum	execution time (sec)
a	2	0 (exactly)	7
b	7	0 (exactly)	10
c	3	0 (exactly)	19
d	3	0 (exactly)	15
e	3	10^{-16}	51
f	16	10^{-14}	239
g	6	10^{-2}	12
h	6	10^{-17}	29

The quadratic convergence, afforded by Newton method, together with the fact that the true values of derivatives were used and not finite differences approximations, resulted in a remarkable accuracy.

Thus, for functions a, c, d,e the first level was enough, and in fact no improve - ment was obtained through the use of the second level.

Function f required both levels. Function g took 35 iterations using the first level; using both levels it took only 6 iterations.

Function b behaved in an apparently abnormal way, taking 7 iterations with one level and 13 when the two-level technique was applied. In the latter case the stability constraint we imposed prevented us from reaching a certain point where the function value increased; but the location of that point was very favorable for a quick convergence (see reference [7]p. 301). A similar thing happened with function h (33 iterations with two levels); function h is included here just as an illustration, since the Newton-Raphson method is more convenient for systems of non-linear

equations (except, however, when the system is over-determined).

4 - Conclusions:

a) The method described here is definitely not the best one for all cases; its main advantages in comparison with the purely numerical methods lie in fast convergence and accuracy; however execution time is somewhat longer, and core requirements are significantly larger.

b) These results should encourage further attempts to computerize a wide class of methods that are partly symbolic and partly numerical.

c) Run-time compilation seems to be a desirable capability to be added to formula manipulation systems, in consideration of problems involving heavy numerical evaluations of the same expressions.

5 - References:

1 - Weizenbaum, J. - "Symmetric List Processor" - Comm. ACM, p. 534 September, 1963.

2 - "Symbol Manipulation Languages and Techniques" - Proceedings of the IFIP Working Conference on Symbol Manipulation Languages, Pisa, 1967 North - Holland, 1968.

3 - Fletcher, R. and Powell, M. J. D. - "A Rapidly Convergent Descent Method for Minimization" - The Computer Journal, vol. 6 p. 163 - 1968.

4 - Graham, R. M. - "Bounded Context Translation" - "Programming Systems and Languages" - edited by Rosen, S. - McGraw - Hill - 1967.

5 - Barron, D. W. and Strachey, C. - "Programming" - Advances in Programming and Non Numerical Computation" - edited by Fox, L. Pergamon, 1966.

6 - Furtado, A. L. and Freire, F. - "Formula Manipulation in the Fletcher and Powell Optimization Method" - presented at the Joint Conference for Mathematical and Computer Aids to Design, 1969.

7 - Wilde, D. J. and Beightler, C. S. - "Foundations of Optimization" Prentice - Hall, 1967.