An improved version of the original leap-frog dynamic method for unconstrained minimization: LFOP1(b)

J. A. Snyman

Department of Applied Mathematics, University of Pretoria, Pretoria 0002, Republic of South Africa
(Received November 1982)

A modified version of the author's original dynamic algorithm for unconstrained minimization is proposed. It employs time step selection procedure which results in a more efficient utilization of the original dynamic algorithm. The performance of the new algorithm is compared with that of a well established conjugate gradient algorithm when applied to three different extended test functions. Based on a comparison of the respective CPU times required for convergence, the new algorithm appears to be competitive.

Key words: unconstrained minimization, optimization, mathematical model

Introduction

This paper presents a modified version of the author's[1] original and heuristically constructed dynamic algorithm for unconstrained minimization, LFOP1 ('Leap-Frog Optimizer version 1'). The version presented here, LFOP1(b), employs an improved time step selection routine in which the time step is automatically reduced or increased so as to ensure an efficient and economic utilization of the basic dynamic algorithm. The modifications were prompted by the successful development and implementation of time step selection procedures in a more rigorously constructed algorithm,[2] LFOP2, developed subsequent to LFOP1. In comparing the performance of LFOP2 with LFOP1 it was realised that, in spite of its more heuristic foundations, the performance of the latter was surprisingly good and that it could probably be improved by the application of a more flexible time step selection procedure.

This paper contains only a brief statement of the original dynamic approach and the bare resultant algorithm without any further background. It is therefore recommended that it be read together with the original paper.[1]

Basic dynamic approach[1]

We consider the unconstrained problem:

$$\text{minimize} \quad F(\boldsymbol{x}); \boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T \in R^n \qquad (1)$$

where $F \in C^1$. This problem may be tackled by considering the associated dynamic problem of the motion of a particle of unit mass in a n-dimensional conservative force field in which the potential energy of the particle at point $\boldsymbol{x}$ at time t is given by $F(\boldsymbol{x}(t))$, i.e. it is required to solve the equations of motion:

$$\ddot{\boldsymbol{x}}(t) = -\nabla F(\boldsymbol{x}(t)) \qquad (2)$$

subject to initial conditions:

$$\boldsymbol{x}(0) = \boldsymbol{x}_0 \qquad (3)$$

$$\dot{\boldsymbol{x}}(0) = \boldsymbol{v}(0) = \boldsymbol{v}_0 \qquad (4)$$

If the associated trajectory is computed and an interfering strategy is adopted to allow for a drastic reduction in kinetic energy whenever the kinetic energy appears to decrease along the path, it follows from energy conservation arguments that a systematic reduction in the potential energy, F, is obtained. The particle is thus forced to follow a path to a local minimum at $\boldsymbol{x}^*$.

Time step reduction

If the specified time step Δt is too large the trajectory will become inaccurate, and although the local accuracy of the computed trajectory is not important far away from $\boldsymbol{x}^*$, it is important near $\boldsymbol{x}^*$ if convergence is to be assured. A measure of the inaccuracy of the path may be taken to be the difference in the directions of the gradient vector at two successive points $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1}$ along the computed path. We may say that an angle of greater than 90° between two successive gradient directions is an indication that the step-size, as a result of too large a time step, has become dangerously large. However, one should be careful not to slow down the procedure too much by intervening at the first instance at which such an angle occurs. The following seems a sensible strategy to adopt. At each step k compute $\boldsymbol{a}_k^T \boldsymbol{a}_{k-1}$. If this quantity is nonpositive for m consecutive steps then halve the time step and restart from $(\boldsymbol{x}_k + \boldsymbol{x}_{k-1})/2$ with velocity $(\boldsymbol{v}_k + \boldsymbol{v}_{k-1})/4$. Typically we may choose $m = 3$ as in the examples which will be given here.

Time step increment

To allow for the possibility of an excessive reduction of the time step during the initial part of the trajectory allowance

0307-904X/83/03216-03/$03.00

is made for the magnification of Δt if a successful step, in the sense that $a_{k+1}^T a_k > 0$ and $\|\Delta x_k\| < \delta$, is made. Here δ denotes the maximum allowable stepsize.[1] The following simple procedure was therefore adopted. Δt is magnified by a factor:

$$p = (1 + N\delta_1) \quad \delta_1 > 0$$

whenever $a_{k+1}^T a_k > 0$ *and* $\|\Delta x_k\| < \delta$. Here N denotes the number of consecutive successful steps carried out. If $a_{k+1}^T a_k \leqslant 0$ then N is reset to the value 1 and if $\|\Delta x_k\| \geqslant \delta$ then N is not stepped up but kept at its current value. A value of $\delta_1 = 0.001$ was used in the experiments reported here. The starting value for Δt was taken as 0.5.

New version of basic algorithm

The algorithm which follows, LFOP1(b), incorporates the time step procedures described in the previous two paragraphs into the original basic dynamic algorithm.[1] The parts enclosed in blocks represent the alterations and additions to the original algorithm.

LFOP1(b)

1. Assume x_0, Δt, δ, $\boxed{m, \delta_1}$ and ϵ given; set $i = 0, j = 2$, $\boxed{s = 0}$, $\boxed{p = 1}$ and $k = -1$.
2. Compute $a_0 = -\nabla F(x_0)$, $v_0 = \frac{1}{2} a_0 \Delta t$.
3. Set $k = k + 1$ and compute $\|\Delta x_k\| = \|v_k\| \Delta t$.
4. If $\|\Delta x_k\| < \delta$, go to $\boxed{5a;}$

   otherwise set $v_k = \delta v_k / (\Delta t \|v_k\|)$ and go to $\boxed{5b;}$

5a. Set $p = p + \delta_1$, $\Delta t = p \Delta t$.

5b. If $s < m$, go to 5;

   otherwise set $\Delta t = \Delta t/2$, $x_k = (x_k + x_{k-1})/2$,

   $v_k = (v_k + v_{k-1})/4$, $s = 0$ and go to 5.

5. Set $x_{k+1} = x_k + v_k \Delta t$.
6. Compute $a_{k+1} = -\nabla F(x_{k+1})$, $v_{k+1} = v_k + a_{k+1} \Delta t$.

7a. If $a_{k+1}^T a_k > 0$, set $s = 0$ and go to 7;

   otherwise set $s = s + 1$, $p = 1$ and go to 7.

7. If $\|a_{k+1}\| \leqslant \epsilon$, stop; otherwise go to 8.
8. If $\|v_{k+1}\| > \|v_k\|$, set $i = 0$ and go to 3;

   otherwise set $x_{k+2} = (x_{k+1} + x_k)/2$, $i = i + 1$ and go to 9.

9. If $i \leqslant j$, set $v_{k+1} = (v_{k+1} + v_k)/4$, set $k = k + 1$ and go to 6;

   otherwise set $v_{k+1} = \mathbf{0}$, $j = 1$, $k = k + 1$ and go to 6.

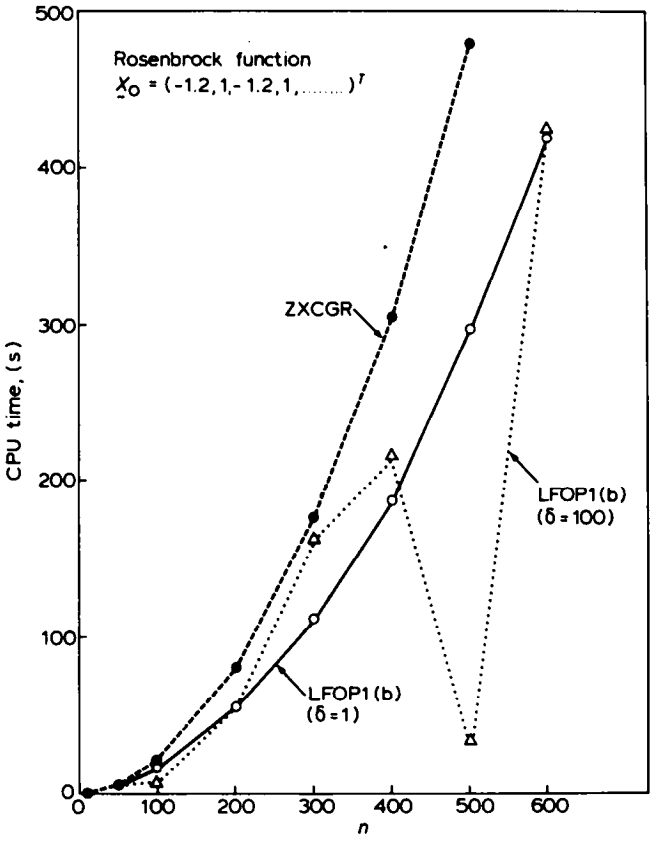

Figure 1

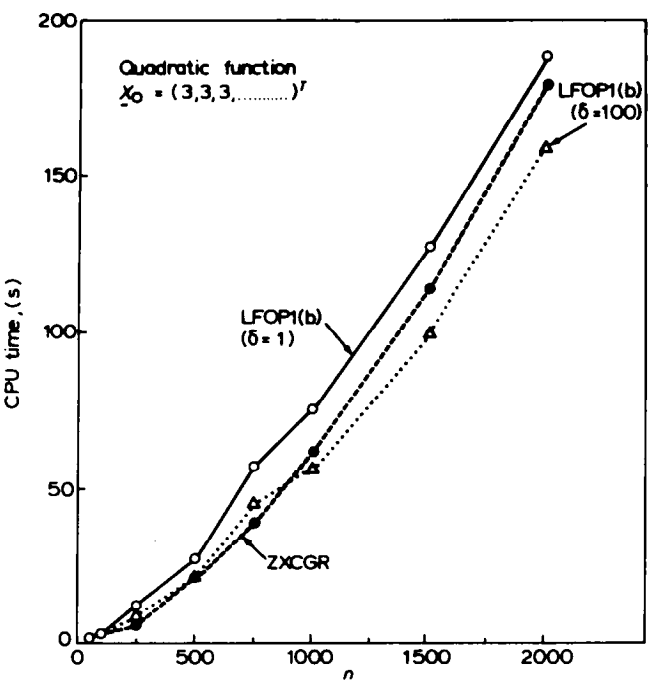

Figure 2

Experimental results

A number of simple numerical tests were carried out to obtain an assessment of the performance of the modified algorithm. A FORTRAN IV program which embodies this algorithm was written.[1] The Rosenbrock, homogeneous quadratic and Oren's extended functions were used as test functions. These functions are well known and are also given in Snyman.[1,2] They are suitable test functions since they are representative of the extreme topographic situations which may arise. The starting points are as given in *Figures 1, 2* and *3*. For purposes of comparison results are also given for the IMSL Subroutine ZXCGR.[3] The programs were compiled using IBM's FORTRAN G compiler

* A listing of this program is available from the author on request

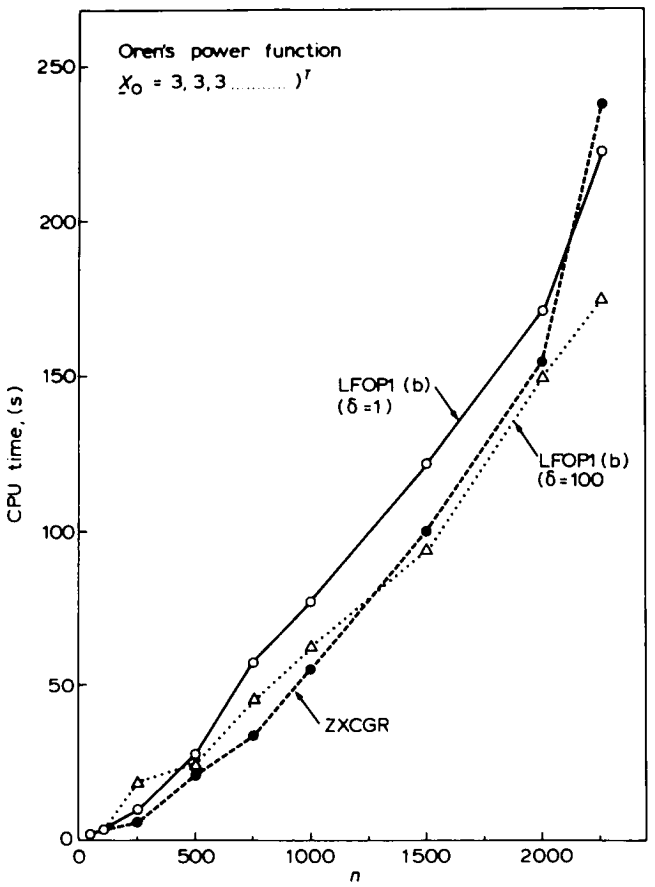

Figure 3

and all computations were performed in double precision on an IBM 4341 model 1 computer. In the execution of the programs all intermediate output was suppressed and the CPU times required for convergence were recorded. The termination criterion in all cases was $\| \nabla F \| \leqslant \epsilon = 10^{-5}$

For the different functions, the CPU times required for convergence of the different algorithms are plotted as functions of the dimension of the problem in *Figures 1*, *2* and *3*. Results are shown for LFOP1(b) applied with two different maximum stepsizes, namely $\delta = 1$ and $\delta = 100$. An initial value for Δt of 0.5 was used throughout.

The results show that the performance of LFOP1(b) is competitive with that of the well established conjugate gradient program, ZXCGR, for all three different kinds of test problems used. The previous criticism that LFOP1 appeared to be tuned to perform well only on Rosenbrock type problems is therefore avoided in the case of the modified algorithm. The effect of changing the maximum stepsize, δ, is also interesting and overall it appears that it may be advantageous to use a large value for δ.

References

1 Snyman, J. A. 'A new and dynamic method for unconstrained minimization', *Appl. Math. Modelling* 1982, 6, 449
2 Snyman, J. A. 'A new convergent minimization method for functions with positive definite Hessian matrices'. Research Report UP TW 28, Department of Applied Mathematics, Univeristy of Pretoria, 1982
3 International Mathematical and Statistical Libraries, Vol. 3, Routine ZXCGR (November, 1979), IMSL Inc., GNB Building, 7500, Bellaire Blvd, Houston, Texas 77036