NASA TECHNICAL NOTE

NASA TN D-8020 c. 1

NASA TN D-8020

4.

~~A~~ PARALLEL JACOBSON-OKSMAN OPTIMIZATION ALGORITHM

Terry A. Straeter and Athena T. Markos

Langley Research Center
Hampton, Va. 23665

3.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • SEPTEMBER 1975

5.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TN D-8020		

4. Title and Subtitle
A PARALLEL JACOBSON-OKSMAN OPTIMIZATION ALGORITHM

5. Report Date
September 1975

6. Performing Organization Code

7. Author(s)
Terry A. Straeter and Athena T. Markos

8. Performing Organization Report No.
L-10044

9. Performing Organization Name and Address
NASA Langley Research Center
Hampton, Va. 23665

10. Work Unit No.
506-25-99-02

11. Contract or Grant No.

12. Sponsoring Agency Name and Address
National Aeronautics and Space Administration
Washington, D.C. 20546

13. Type of Report and Period Covered
Technical Note

14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

A gradient-dependent optimization technique which exploits the vector-streaming or parallel-computing capabilities of some modern computers is presented. The algorithm, derived by assuming that the function to be minimized is homogeneous, is a modification of the Jacobson-Oksman serial minimization method.

In addition to describing the algorithm, conditions insuring the convergence of the iterates of the algorithm and the results of numerical experiments on a group of sample test functions are presented. The results of these experiments indicate that this algorithm will solve optimization problems in less computing time than conventional serial methods on machines having vector-streaming or parallel-computing capabilities.

17. Key Words (Suggested by Author(s))
Optimization
Numerical analysis
Parallel computers
Jacobson-Oksman algorithm

18. Distribution Statement
Unclassified – Unlimited

Subject Category 64

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	18	$3.25

*For sale by the National Technical Information Service, Springfield, Virginia 22161

A PARALLEL JACOBSON-OKSMAN OPTIMIZATION ALGORITHM

Terry A. Straeter and Athena T. Markos
Langley Research Center

SUMMARY

A gradient-dependent optimization technique which exploits the vector-streaming or parallel-computing capabilities of some modern computers is presented. The algorithm, derived by assuming that the function to be minimized is homogeneous, is a modification of the Jacobson-Oksman serial minimization method.

In addition to describing the algorithm, conditions insuring the convergence of the iterates of the algorithm and the results of numerical experiments on a group of sample test functions are presented. The results of these experiments indicate that this algorithm will solve optimization problems in less computing time than conventional serial methods on machines having vector-streaming or parallel-computing capabilities.

INTRODUCTION

In recent years gradient-dependent optimization techniques have been used to solve a number of aerospace and aeronautical engineering problems with the digital computer. The most widely used gradient-dependent techniques are the conjugate gradient (CG) and Davidon-Fletcher-Powell (DFP) methods (see refs. 1 to 3). These methods are similar in many ways. The similarity of interest here is the fact that the methods are serial in nature. That is, the n-dimensional minimization is replaced with a sequence of univariate minimizations where the new direction is determined as a result of the value of the gradient of the function and the previous univariate minimization. When these methods were developed, their serial nature did not seem to be a particular disadvantage. In fact, many felt it an advantage since the digital computers for which the algorithms were programed were also serial. However, a new generation of computers with vector-streaming or parallel-computing capabilities has appeared (for example, Illiac IV, C. mmP, and STAR). These capabilities are used here to synchronously or simultaneously evaluate the function and its gradient at a number of different points. These advanced computing capabilities cannot be exploited by the CG and DFP methods.

Thus far, some work has been done to develop optimization algorithms which exploit these parallel capabilities. Avriel, Karp, and their associates (refs. 4 and 5) have worked on the univariate minimization problem. Miranker et al., have developed a nongradient

"parallel" version of the Powell-Zangwill method (refs. 6 to 8). Straeter (ref. 9) has developed a parallel variable metric (PVM) technique which is gradient dependent and a close relative of the Davidon-Broyden (refs. 10 to 12) rank one techniques. Although the PVM, CG, and DFP differ in that the former exploits parallel-computing capabilities and the others do not, they are similar in that they are all closely tied to quadratic functions because the model function used to develop the algorithms is a quadratic. Algorithms like PVM, DFP, and CG which minimize positive definite quadratics in a finite number of steps exhibit good convergence properties on more general functions because near its minimum a sufficiently smooth function is closely approximated by a quadratic. However, this quadratic may be singular. In such cases, the convergence of these algorithms is slowed considerably.

Recently, Jacobson and Oksman (ref. 13) have proposed an algorithm (denoted by JO) which assumes that the model function is homogeneous. The class of homogeneous functions contains the quadratics as a subclass and is therefore richer than the quadratics. Functions which have a singular matrix of second partials at the minimum can be more accurately approximated by the homogeneous model. Jacobson and Oksman report that their algorithm is competitive with the DFP and CG; however, as originally formulated, the JO algorithm is a serial technique. In fact, it was the serial nature of the then-current computers that would appear to have influenced Jacobson and Oksman to form their algorithm as they did.

The purpose of this report is to describe an algorithm based on a homogeneous function model which exploits vector-streaming or parallel-computing capabilities. The algorithm is called the parallel Jacobson-Oksman (PJO) method. The term parallel was chosen to describe the synchronous or simultaneous evaluation of the function and its gradient at a number of different points. In addition to describing the algorithm and a modification, some conditions guaranteeing convergence are presented. Also presented are the results of numerical experiments involving the application of these methods to a collection of test functions.

SYMBOLS

A	positive definite $(n \times n)$ symmetric matrix
A^{-1}	inverse of A
b	n vector of constants
C	$(n + 2) \times (n + 2)$ matrix defined in equation (4)

e_i	elementary vector
f_o	fixed scalar
$f(x)$	function of n variables
$g(x)$	gradient vector of f
i,j,k	integers
n	index
R^n	vector space of n-tuples
s_k	kth search direction
t	scalar
x	variable, n vector
x^j	jth component of n-vector x
$\tilde{x}_j$	intermediate iterates (n-vectors)
v	$n + 2$ vector with components v_j $(j = 1, 2, \ldots, n + 2)$
v_j	scaler defined by $\tilde{x}_j^T g(\tilde{x}_j)$
α	$= \left[\tilde{\beta}^T, \tilde{\gamma}, \tilde{\omega}\right]^T$
β	n-vector, location of minimum
$\tilde{\beta}$	n-vector estimate of β
γ	scalar, degree of homogeneity
$\tilde{\gamma}$	scalar, estimate of γ
λ,λ_k	scalars, step size

$\sigma_1, \sigma_2, \ldots, \sigma_{n+1}$ set of n vectors which spans R^n

τ scalar $(0 < \tau < 1)$

ω $= \gamma f(\beta)$

$\tilde{\omega}$ estimate of ω

$( )^T$ transpose

$\|\cdot\|$ norm of n vector; for example $\|x\| = \sqrt{x^T x}$

HOMOGENEOUS FUNCTIONS AND THE PJO ALGORITHM

Let f be a function of the n-tuple x. The function f is said to be homogeneous of degree γ about the point β if for all x throughout a region R containing β

$$f(tx + \beta) - f(\beta) = t^{\gamma}\left[f(x + \beta) - f(\beta)\right] \tag{1}$$

for all t such that $tx \in R$.

Notice that quadratic functions are homogeneous of degree two about their minimum point. This relationship follows because if

$$f(x) = f_o + b^T x + \frac{1}{2} x^T A x$$

for A an $n \times n$ positive definite symmetric matrix, b a fixed vector in R^n, f_o a scalar, the gradient is $g(x) = b + Ax$ and the minimum is at $\beta = -A^{-1}b$. As a result, $f(tx + \beta) = f_o - \frac{1}{2} b^T A^{-1} b + \frac{1}{2} t^2 x^T A x$, and thus

$$f(tx + \beta) - f(\beta) = \frac{1}{2} t^2 x^T A x = t^2\left[f(x + \beta) - f(\beta)\right]$$

This definition is a generalization of the usual one where $\beta = 0$ and $f(\beta) = 0$. If f is differentiable with gradient g(x), homogeneous of degree γ about β, then Euler's theorem states that for all $x \in R$

$$f(\beta) = f(x) + \frac{1}{\gamma}(\beta - x)^T g(x) \tag{2}$$

This relationship is established by differentiating equation (1) with respect to t, setting $t = 1$, and replacing x by $x - \beta$.

The preceding definition and model equations are as given by Jacobson and Oksman. (See ref. 1.) The model equation (eq. (2)) for a homogeneous function forms the basis of the PJO algorithm. For each $x \in R^n$, equation (2) contains $n + 2$ constants. These constants are β, the location of the minimum, the minimum value of f and γ, and the degree of homogeneity. Moreover, if $\gamma f(\beta)$ is denoted by ω equation (2) can be rewritten as

$$\omega = \gamma f(x) + (\beta - x)^T g(x) \tag{3}$$

which is linear in the $n + 2$ unknowns, ω, γ, and the n-tuple β. This observation is, of course, the same observation that led Jacobson and Oksman to their algorithm. However, their algorithm is a serial algorithm as each evaluation of equation (3) results in an estimate of β, γ, and ω. The next location for evaluation of equation (3) is determined by this estimate of β. Their formulation of the solution of equation (3) at $n + 2$ points was based on the sequential or serial nature of the then-present-day computers. A short description of their method follows the description of the new algorithm.

Most of the computer time required to solve significant optimization problems is spent evaluating the function and its gradient. The parallel-computing capabilities of advanced computers allow the evaluation of the function and its gradient at a number of points simultaneously at a cost in time of not much more than a single evaluation. However, if a number of simultaneous gradient and function evaluations were made in parallel, the original JO algorithm could not use this inexpensive information because after each function and gradient evaluation, a decision is made concerning the location of the next evaluation. Hence, the algorithm as stated is inherently serial.

The modification of the JO algorithm presented here involves parallel evaluation of the function and its gradient at $n + 2$ distinct points (that is, values for x), and the solution at those points of the resulting linear system of $n + 2$ equations in $n + 2$ unknowns, ω, γ, and β being defined by equation (3). This estimate of β then defines a search direction. The algorithm proceeds in that direction until a point is found at which the value of the object function is decreased. The process is repeated until the norm of the gradient is sufficiently small. A formal statement of the algorithm is as follows:

Step 0: Let x_0 be the initial estimate of the location of the minimum and compute $f(x_0)$ and $g(x_0)$. Let $\sigma_1, \sigma_2, \ldots, \sigma_{n+1}$ be a set of n-vectors which spans R^n and any proper subset which is linearly independent on R^n and let $k = 0$.

Step 1: Define

$$\tilde{x}_j = x_k + \sigma_j \qquad (j = 1, 2, \ldots, n + 1)$$

Evaluate f and g in parallel at $\tilde{x}_j$ for $j = 1, 2, \ldots, n+1$. Set $\tilde{x}_{n+2} = x_k$ and evaluate equation (3) at the $n+2$ points $\tilde{x}_j$ and form the linear system

$$C\alpha = v \tag{4}$$

where

$$\alpha = \left[\tilde{\beta}^T, \tilde{\gamma}, \tilde{\omega}\right]^T \qquad (\tilde{\beta} \text{ being an estimate of } \beta)$$

$$v_j = \tilde{x}_j^T g\left(\tilde{x}_j\right) \qquad (j = 1, 2, \ldots, n+2)$$

$$C = \begin{cases} \dfrac{\partial f}{\partial x^j}\left(\tilde{x}_i\right) & (i = 1, 2, \ldots, n+2; \quad j = 1, 2, \ldots, n) \\ f\left(x_i\right) & (i = 1, 2, \ldots, n+2; \quad j = n+1) \\ -1 & (i = 1, 2, \ldots, n+2; \quad j = n+2) \end{cases}$$

Solve equation (4) for $\tilde{\beta}$, $\tilde{\gamma}$, and $\tilde{\omega}$.

Step 2: Define the search direction s_k by $s_k = \tilde{\beta} - x_k$ and evaluate $f(\tilde{\beta})$ and $g(\tilde{\beta})$; if $\left\|g(\tilde{\beta})\right\|$ is sufficiently small, stop. If not and $f(\tilde{\beta}) < f\left(x_k\right)$, then set $x_{k+1} = \tilde{\beta}$, $k = k + 1$, and return to step 1. Otherwise, choose the step size by minimizing $f\left(x_k + \lambda s_k\right)$ with respect to λ. Denote the minimizing λ by λ_k. Let $x_{k+1} = x_k + \lambda_k\left(\tilde{\beta} - x_k\right)$, set $k = k + 1$, and return to step 1.

In the original Jacobson-Oksman algorithm, step 1 is replaced by assuming that at step 1 one has a current estimate of C^{-1} and α. The oldest information in C is then replaced by the corresponding information about f at x_k. This means that one row of C has been replaced. The new C^{-1} and α are then computed by Householder's formula. (See ref. 14.) Finally, step 2 is taken and the process repeats as necessary.

CONVERGENCE RESULTS

If f is a homogeneous function of n variables with minimum at β and $\sigma_1, \sigma_2, \ldots, \sigma_{n+1}$ are such that the system of $n+2$ linear equations formed by evaluating equation (3) at x_0, $x_0 + \sigma_j$ $(j = 1, 2, \ldots, n+1)$ has a unique solution, then the algorithm described herein locates the minimum in one cycle. This is immediate by the construction of the algorithm and definition of a homogeneous function.

By slight modifications of the algorithm defined, one can obtain more powerful convergence results. First, it is required that the step direction s_k as chosen in step 2 satisfies

$$\left|g(x_k)^T s_k\right| \geqq \tau \left|\left| g(x_k) \right|\right| \cdot \left|\left| s_k \right|\right| \tag{5}$$

for some scalar τ where $0 < \tau < 1$. If this condition is not satisfied, then s_k is chosen as $-g(x_k)$. Additionally, it is required that the step size λ_k is chosen so that

$$f(x_k) - f(x_k + \lambda_k s_k) - \tau \lambda_k \left(\frac{g^T(x_k)\, s_k}{s_k^T s_k} \right)^2 \geqq 0$$

Then the following stronger convergence result can be established.

If at every iteration these modifications to the search direction and step size are made, the iterates of the PJO algorithm will converge to the location of the minimum of a C^2 strongly convex function. (A C^2 strongly convex function is a function with a continuous, uniformly positive definite matrix of second derivatives.) This result follows from a more general result given by Ortega and Rheinboldt. (See ref. 14.)

As will be seen in a later section, in practice the modified algorithm, for which the stronger convergence result held, did not perform better than the basic algorithm. In fact, the performance of the modified algorithm was sometimes worse in terms of function evaluations required to solve the problem. This behavior of minimization algorithms has been observed by others (for example, Daniel, ref. 15).

NUMERICAL CONSIDERATIONS

In order to insure that the elements of the C matrix were approximately of the same magnitude, the $n + 2$ column was scaled by $0.5f(x_k)$. This scaling meant that the linear equation solver returned $2\omega/f(x_k)$ instead of ω. Since ω was not used in any part of the algorithm, this condition was of no consequence. Recall that to locate the minimum of a homogeneous function, the vectors $\sigma_1, \sigma_2, \ldots, \sigma_{n+1}$ required by step 1 of the algorithm need only span R^n and generate a unique solution to equation (4); however, to be more likely that equation (4) has a unique solution, the σ vectors were chosen so that any subset of n vectors is also linearly independent. In fact, for the numerical examples, σ_{n+1} is chosen as the sum of the first n σ vectors. Of interest, of course, is the sensitivity of the performance to other criteria for choosing these vectors of the algorithm when minimizing general functions. Clearly, the length of the vectors σ_i should be small enough to insure that the local representation of $f(x)$ as an

homogeneous function is valid; however, this means that the entries of the rows of the C matrix of equation (4) are very similar. For example, the $n + 1$ column of C is given by $\left[f(x_k), f(x_k + \sigma_1), \ldots, f(x_k + \sigma_{n+1})\right]^T$. Since the norm of σ_i is small for every i, this means that the entries are very close numerically. Hence, for very small σ_i, the system given by equation (4) may not have full rank or may be very close to being singular. Therefore, the solution of the system of equations may be sensitive to small changes in the entries of the C matrix. Thus, it would seem that there are two factors affecting the performance of the algorithm: (1) the criteria for choosing the σ vectors and (2) the linear solver of equation (4). In order to investigate these factors, two techniques of solving the linear system and several schemes of choosing σ vectors were used. The solution techniques used were Gaussian elimination with total pivotal strategy (ref. 16) and a singular value decomposition technique (ref. 17). The latter gives the rank of the matrix, its singular values, and the least-squares solution to the linear system given by equation (4). The latter algorithm is generally considered to give the better (that is, more accurate) solution.

In general, the following observations are made after many numerical experiments involving the test functions described:

(1) The performance of the PJO algorithm was independent of the linear solver used. By independent, it is meant that with both linear solvers, convergence to the minimum was achieved in essentially the same number of function evaluations for the same problem.

The use of the singular value decomposition linear solver did indicate the fact that the C matrix is not necessarily of full rank. On two example problems, the rank of the C matrix was computed as $n - 1$ for the first iteration; however, as the iterations proceeded, C was of full rank and remained so. Table I lists the rank of C, the largest, and the smallest singular values for each iteration while minimizing one of the sample functions. Note that after the first iteration, the singular values are well within the 15-place accuracy of the computers used in the experiments. Although the performances were generally independent, in one anomolous case the algorithm using the Gaussian elimination minimized one of the test functions in 3 cycles and only 19 function evaluations, whereas this function usually required at least 11 cycles and about 67 function evaluations. This one case of phenomenal performance is due to a fortuitous first cycle which, while decreasing the function value from 215 to only 214, gave rise to a good starting point for the second cycle. The second cycle then decreased the function to 10^{-5}, and the final step to 10^{-10}. Ordinarily, the decrease in the first step is from 215 to about 150. Then the algorithm proceeds to decrease f(x) by about an order of magnitude per cycle.

(2) The experiments indicated that the manner in which the σ_i vectors were chosen was more important in terms of overall performance than the linear equation-solving technique.

The following schemes referred to as methods (a) and (b) for choosing the σ_i vectors were investigated:

(a) The first method (method (a)) for choosing the σ_i vectors was to let $\sigma_i = e_i c$ for $i = 1, 2, \ldots, n$ and $\sigma_{n+1} = c\{1, 1, \ldots, 1\}^T$ where e_i denotes the ith elementary vector and c is a scalar fixed for all iterations. Figure 1 illustrates the performance of the algorithm in terms of function evaluations on three sample problems, for $c = 10^{-2}$, 10^{-3}, 10^{-4}, 10^{-5}, and 10^{-6}. Notice that as c decreases, the number of function evaluations required for convergence decreases until a minimum is reached; thereafter, the performance deteriorates rapidly. The term robustness is used herein to describe the relative insensitivity of the algorithm to the magnitude of $||\sigma_i||$ and the techniques of

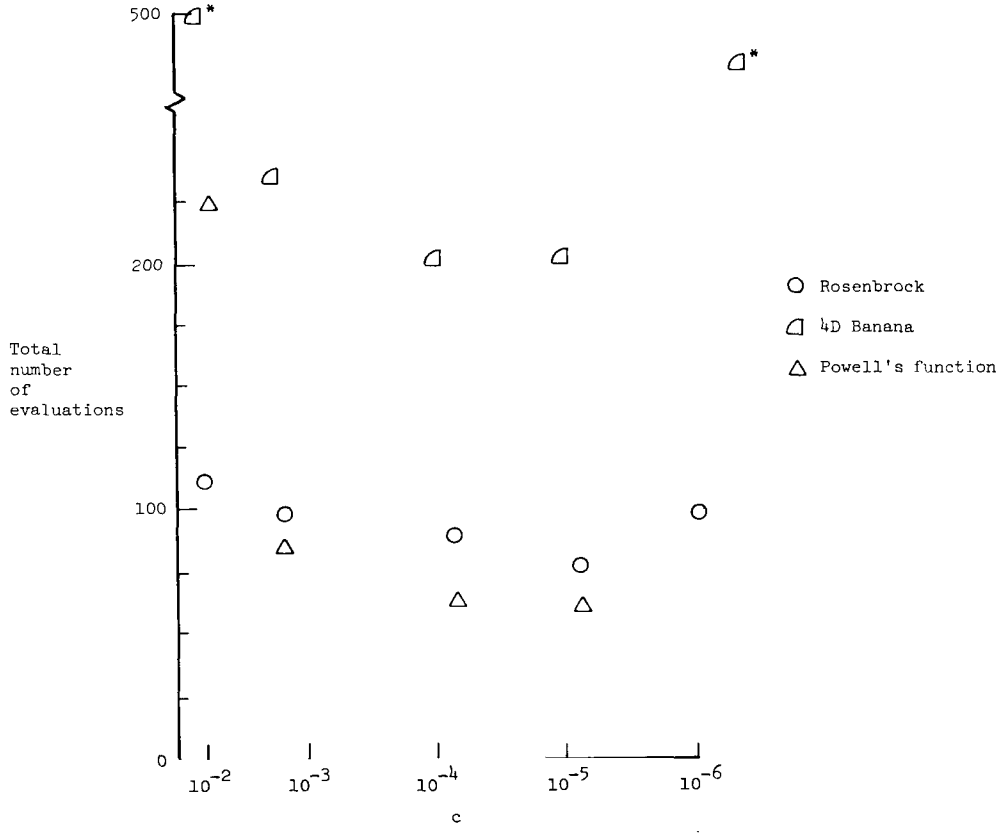

Figure 1.- Robustness of PJO (σ vectors chosen by method (a)). Symbol with asterisk denotes that function did not converge in 500 function and gradient evaluations; Powell's function for norm of $\sigma = 10^{-6}$ was not tested.

choosing σ_i. For the test problems solved herein, it was observed that PJO had about a 2 to 3 order of magnitude deadband on performance in terms of function evaluations required to converge to the solution.

(b) The second technique (method (b)) for choosing the σ_i vectors was to let $\sigma_i = -ce_i$ times sgn $\left(\text{ith component of } g(x_k)\right)$ for $i = 1, 2, \ldots, n$ and $\sigma_{n+1} = \sum_{i=1}^{n} \sigma_i$ $\left(\text{ith component of } g(x_k)\right)$ where c is a scalar. This technique seemed to improve the overall performance of the algorithm, but not to a great degree. The improvement was probably due to the fact that the approximation to f by a homogeneous form was localized and coordinated with the probable n-dimensional orthant of decrease. (An orthant is an n-dimensional quadrant.)

A final numerical consideration was that the one-dimensional minimization required in step 3 of the algorithm was carried out as follows. The initial step-size estimate is given by

$$\lambda_k = \min\left\{1, \hat{\gamma}_k \frac{f(x_k) - f_m}{g_n^T} s_k\right\}$$

where $\hat{\lambda}_k$ is the estimate of λ at the kth cycle and f_m is the estimated value of f at the minimum. If $f(x_k + \gamma_k s_k) < f(x_k)$, the next cycle is begun; otherwise, Davidon's cubic interpolation (eq. (3)) is performed to locate the one-dimensional minimum.

EXAMPLE PROBLEMS AND RESULTS

To illustrate the performance of the minimization algorithms defined herein, numerical experiments were conducted on several standard example problems. Although the experiments were conducted in single precision on a Control Data Corporation (CDC) 6000 series computer, the results of the computations were used as if they had been done in a parallel fashion. Five sample functions were used for the numerical experiments.

The first function was a simple quadratic function of three variables

$$f_1(x,y,z) = x^2 + 2y^2 + 5z^2 - 2xy$$

As indicated in the section "Convergence Results," convergence was achieved in one cycle.

The second function was Rosenbrock's parabolic valley function

$$f_2(x,y) = \left(x^2 - y\right)^2 + 0.01(x - 1)^2$$

This is a particularly difficult function to minimize because of the long parabolic valley $y = x^2$ along which the minimization must travel. The traditional starting point is (-1.2, 1.0) and the minimum is located at (1,1).

The third function known as Powell's function is

$$f_3(x_1,x_2,x_3,x_4) = (x_1 - 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

with starting values at (3,-1,0,1). This function is particularly difficult for a variable metric algorithm to minimize because at the minimum the Hessian is singular. The algorithm described herein performed extremely well on this problem.

The fourth test function is called the 4-D Banana and is defined by

$$f_4(x_1,x_2,x_3,x_4) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2 + 90(x_3^2 - x_4)^2 + (1 - x_3)^2 + 10.1\left[(x_2 - 1)^2 + (x_4 - 1)^2\right] + 19.8\,(x_2 - 1)(x_4 - 1)$$

The traditional starting estimate is at (-3,-1,-3,-1). This function is difficult to minimize because the quadratics $x_1^2 - x_2$ and $x_3^2 - x_4$ make the level surfaces banana shaped.

The fifth function is the helical valley function defined by

$$f_5(x_1,x_2,x_3) = 100\left[(x_3 - 10\theta)^2 + \left(\sqrt{x_1^2 + x_2^2} - 1\right)^2\right] + x_3^2$$

where

$$2\pi\theta = \begin{cases} \tan^{-1} x_2/x_1 & (x_1 \geq 0) \\ \pi + \tan^{-1} x_2/x_1 & (x_1 < 0) \end{cases}$$

The usual starting estimate is at (-1,0,0).

The key factor in the performance of any minimization technique is the number of function evaluations required for convergence as the computation required for evaluating the function is usually much greater than that involved in the algorithm. It is for this reason that the results of the experiments are given in terms of the number of function evaluations required to achieve convergence.

Table II gives some of the results of these numerical experiments. For these experiments the convergence criteria was that the largest component of the gradient be less in magnitude than $\epsilon = 10^{-6}$. The σ vectors were defined by method (b) with

$e = 10^{-4}$, that is, $\sigma_i = (10^{-4}e_i)(\text{sgn ith component of } g(x_k))$ and $\sigma_{n+1} = -10^{-4}e_i(\text{sgn ith component of } g(x_k) \times 10^{-4})$ where e_i is the ith elementary vector. For each test function the number of cycles required to achieve convergence is listed. Also listed is the total number of function and gradient evaluations required for convergence on the serial computer and on an ideal parallel computer. By assuming that the computations had been done on an ideal parallel computer with p processors ($p \geqq n + 1$), step 1 would have been performed by utilizing these capabilities; therefore, if overhead costs are ignored and an ideal parallel computer is assumed, these gradient and function evaluations would take essentially the same time as one such computation. It is for this reason that the entry in column (3) is found by taking the number of variables times the number of cycles and subtracting the result from the entry in column (2). Columns (4) and (5) give measures of the accuracy of the minimization.

Table III shows the same information for the modified algorithm for which the results given in "Convergence Results" hold with $\tilde{\tau} = 0.001$. A comparison of tables II and III shows that the modification which guarantees convergence does not help the performance of the algorithm to any significant degree.

Table IV shows the results for the modified PJO with the σ vectors chosen by method (a) given and with $c = 10^{-4}$. A comparison of tables III and IV shows that the performance of the modified algorithm is not radically affected by either of the two ways of choosing the σ vectors. The same situation held for the PJO itself.

Table V gives the performance of the algorithm by use of the singular value decomposition linear solver. Note here again that the performance of the algorithm is not radically affected.

Table VI illustrates a comparison of the performance of the DFP algorithm with the PJO algorithm on the four example functions, f_2, f_3, f_4, and f_5. The DFP method was chosen as the standard of comparison because of its wide use and generally favorable comparison with other techniques. Table VI lists the number of function evaluations required by the DFP method to locate the minimum starting from the same initial conditions. These results were reported by Jacobson and Oksman (ref. 13). Finally, in table VI, the performance of the PJO is given for two cases. In the first case, the method is used on a serial computer; hence the operations in step one (that is, gradient evaluations) are not done in parallel. For the second case it is assumed that the operations in step 1 are carried out in parallel. Hence the $n + 1$ gradient and function evaluations of step 1 will require only the time to carry out one evaluation. So the entry in table V for the parallel case is merely the same as that for the serial case minus n times the number of cycles. The next two entries give the performance of the PVM algorithm for these same problems.

CONCLUDING REMARKS

The numerical experiments indicate that the parallel Jacobson-Oksman (PJO) algorithm outlined herein can be used to exploit effectively parallel computing capabilities to solve unconstrained optimization problems in less computation time. In fact, the performance of the algorithm indicates that for small problems (that is, two or three variables) the PJO can be competitive with currently used serial algorithms. Also, in the parallel mode the PJO appears to be competitive with another proposed parallel optimization algorithm, the parallel variable metric algorithm.

It is interesting that the modified PJO did not perform as well on the sample problems as did the PJO, in spite of the more general convergence theory for the modified algorithm; however, this phenomenon is not uncommon. A note of concern about the PJO is the limited robustness of the algorithm; however, the 2 to 3 order of magnitude robustness band was encouraging.

Langley Research Center
National Aeronautics and Space Administration
Hampton, Va. 23665
July 8, 1975

REFERENCES

1. Fletcher, R.; and Reeves, C. M.: Function Minimization by Conjugate Gradients. Computer J., vol. 7, no. 2, July 1964, pp. 149-154.

2. Davidon, William C.: Variable Metric Method for Minimization. ANL-5990 Rev. (Contract W-31-109-eng-38), Argonne Nat. Lab., Nov. 1959.

3. Fletcher, R.; and Powell, M. J. D.: A Rapidly Convergent Descent Method for Minimization. Computer J., vol. 6, no. 2, July 1963, pp. 163-168.

4. Avriel, M.; and Wilde, D. J.: Optimal Search for Maximum With Sequences of Simultaneous Function Evaluations. Manage. Sci., vol. 12, no. 9, May 1966, pp. 722-731.

5. Karp, Richard M.; and Miranker, Willard L.: Parallel Minimax Search for a Maximum. J. Combinatorial Theory, vol. 4, 1968, pp. 19-35.

6. Powell, M. J. D.: An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives. Comput. J., vol. 7, no. 2, July 1964, pp. 155-162.

7. Zangwill, W. I.: Minimizing a Function Without Calculating Derivatives. Comput. J., vol. 10, Nov. 1967, pp. 293-296.

8. Chazan, D.; and Miranker, W. L.: A Nongradient and Parallel Algorithm for Unconstrained Minimization. SIAM J. Contr., vol. 8, no. 2, May 1970, pp. 207-217.

9. Straeter, Terry A.: A Parallel Variable Metric Optimization Algorithm. NASA TN D-7329, 1973.

10. Powell, M. J. D.: A Survey of Numerical Methods for Unconstrained Optimization. SIAM Rev., vol. 12, no. 1, Jan. 1970, pp. 79-97.

11. Broyden, C. G.: A Class of Methods for Solving Nonlinear Simultaneous Equations. Math. Comput., vol. 19., no. 92, Oct. 1965, pp. 577-593.

12. Davidon, William C.: Variance Algorithms for Minimization. Computer J., vol. 10, no. 4, Feb. 1968, pp. 406-410.

13. Jacobson, D. H.; and Oksman, W.: An Algorithm That Minimizes Homogeneous Functions of N Variables in N + 2 Iterations and Rapidly Minimizes General Functions. Tech. Rep. No. 618 (Contract N00014-67-A-0298-0006), Div. Eng. & Appl. Phys., Harvard Univ., Oct. 1970.

14. Ortega, J. M.; and Rheinboldt, W. C.: Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, Inc., 1970.

15. Daniel, James W.: Partial Analysis of an Algorithm for Minimizing Functions Based on Homogeneous Model. CNA-15, Center Numer. Anal., Univ. of Texas at Austin, Apr. 1971.

16. Fox, L.: An Introduction to Numerical Linear Algebra. Oxford Univ. Press, 1965.

17. Golub, G. H.; and Reinsch, C.: Handbook Series Linear Algebra – Singular Value Decomposition and Least Squares Solutions. Tech. Rep. CS-133 (Contract N00014-07-A-0112-0029), Stanford Univ., May 1969. (Available from DDC as AD 687 718.)

TABLE I.- TYPICAL RANK, SINGULAR VALUE RANGE OF C MATRIX

Iteration	Computed rank, full rank = 5	Largest singular value	Smallest singular value
1	4	6.9×10^{3}	0
2	5	81	2.5×10^{-8}
3	5	115	7.7×10^{-10}
4	5	80.3	4.8×10^{-9}
5	5	67.4	2.7×10^{-9}
6	5	34.3	1.5×10^{-9}
7	5	60.9	1.8×10^{-8}
8	5	20.7	6.5×10^{-9}
9	5	18.7	7.3×10^{-9}
10	5	16.4	8.3×10^{-9}
11	5	14.6	9.2×10^{-9}
12	5	13.8	10×10^{-8}
13	5	11.8	1×10^{-8}
14	5	8	9×10^{-9}
15	5	4.3	7.7×10^{-9}
16	5	1.2	5.5×10^{-9}
17	5	5.9×10^{-2}	4.6×10^{-9}
18	5	1.2×10^{-2}	2.46×10^{-11}

TABLE II.- RESULTS OF NUMERICAL EXPERIMENTS FOR THE PARALLEL JACOBSON-OKSMAN ALGORITHM (PJO)

[Experimental conditions: σ_i chosen by method (b); $c = 10^{-4}$]

Function	① Number of cycles	② Total evaluation (serial)	③ Total evaluation (parallel)	④ Function value at convergence	⑤ Norm of gradient
Quadratic	1	6	3	6.6×10^{-20}	9.7×10^{-10}
Rosenbrock	20	87	47	9×10^{-14}	1.6×10^{-6}
Powell	3	19	7	1.9×10^{-10}	1.3×10^{-6}
4-D Banana	36	227	87	6.6×10^{-12}	3×10^{-8}
Helical valley	14	74	32	5.9×10^{-18}	5×10^{-8}

TABLE III.- RESULTS OF NUMERICAL EXPERIMENTS FOR THE MODIFIED PARALLEL JACOBSON-OKSMAN ALGORITHM

[Experimental conditions: σ_i chosen by method (b); $c = 10^{-4}$; $\tau = 0.001$]

Function	Number of cycles	Total evaluation (serial)	Total evaluation (parallel)	Function value at convergence	Norm of gradient
Rosenbrock	20	87	47	9×10^{-14}	5×10^{-8}
Powell	13	80	28	2×10^{-9}	3×10^{-6}
4-D Banana	34	223	87	7×10^{-13}	4×10^{-6}
Helical valley	14	74	32	3×10^{-16}	5×10^{-7}

TABLE IV.- RESULTS OF NUMERICAL EXPERIMENTS FOR THE MODIFIED PARALLEL JACOBSON-OKSMAN ALGORITHM

[Experimental conditions: σ_i chosen by method (a); $c = 10^{-4}$; $\tau = 0.001$]

Function	Number of cycles	Total evaluation (serial)	Total evaluation (parallel)	Function value at convergence	Norm of gradient
Rosenbrock	21	91	49	4.5×10^{-15}	3×10^{-8}
Powell	11	67	23	1.98×10^{-9}	4×10^{-6}
4-D Banana	33	208	76	7×10^{-14}	2.9×10^{-6}
Helical valley	18	92	38	3.6×10^{-14}	3.5×10^{-6}

TABLE V.- PERFORMANCE OF PJO WITH SINGULAR VALUE DECOMPOSITION LINEAR SOLVER

[Experimental conditions: σ_i chosen by method (b); $c = 10^{-4}$]

Function	Number of cycles	Total evaluation (serial)	Total evaluation (parallel)
Rosenbrock	20	87	47
Powell	11	67	23
4-D Banana	34	223	76
Helical valley	14	74	32

TABLE VI.- RESULTS OF NUMERICAL EXPERIMENTS: TOTAL NUMBER OF FUNCTION AND GRADIENT EVALUATIONS REQUIRED TO ACHIEVE CONVERGENCE FOR EXAMPLE FUNCTIONS

	Quadratic function	Rosenbrock's function	Powell's function	4-D Banana	Helical valley
Parallel Jacobson-Oksman (Serial computer)	6	74	19	204	74
Parallel Jacobson-Oksman (Parallel computer)	3	40	7	76	32
Davidon-Fletcher-Powell* (Serial computer)	Not reported	165	80	161	76
PVM (Serial computer)	7	58	118	185	75
PVM (Parallel computer)	5	45	61	95	47

*As reported by Jacobson and Oksman (ref. 13).

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

—NATIONAL AERONAUTICS AND SPACE ACT OF 1958

NASA SCIENTIFIC AND TECHNICAL PUBLICATIONS

TECHNICAL REPORTS: Scientific and technical information considered important, complete, and a lasting contribution to existing knowledge.

TECHNICAL NOTES: Information less broad in scope but nevertheless of importance as a contribution to existing knowledge.

TECHNICAL MEMORANDUMS: Information receiving limited distribution because of preliminary data, security classification, or other reasons. Also includes conference proceedings with either limited or unlimited distribution.

CONTRACTOR REPORTS: Scientific and technical information generated under a NASA contract or grant and considered an important contribution to existing knowledge.

TECHNICAL TRANSLATIONS: Information published in a foreign language considered to merit NASA distribution in English.

SPECIAL PUBLICATIONS: Information derived from or of value to NASA activities. Publications include final reports of major projects, monographs, data compilations, handbooks, sourcebooks, and special bibliographies.

TECHNOLOGY UTILIZATION PUBLICATIONS: Information on technology used by NASA that may be of particular interest in commercial and other non-aerospace applications. Publications include Tech Briefs, Technology Utilization Reports and Technology Surveys.

Details on the availability of these publications may be obtained from:

SCIENTIFIC AND TECHNICAL INFORMATION OFFICE

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546