

PARALLEL MULTISPLITTINGS FOR OPTIMIZATION*

R. A. RENAUT[†] AND H. D. MITTELMANN[‡]

Abstract. The philosophy of multisplitting methods is the replacement of a large-scale linear or nonlinear problem by a set of subproblems, each of which can be solved locally and independently in parallel by taking advantage of well-tested sequential algorithms. Because of this formulation most compute-intensive operations can be calculated independently and the algorithms are highly parallel. Recent developments for optimization, constrained and unconstrained, are described. These new algorithms are, in some cases, faster in sequential mode than conventional algorithms. Results of implementations on the Intel Paragon and on a cluster of workstations using PVM3 demonstrate superlinear speedup when compared with a standard test algorithm programmed in sequential mode. Further, the same algorithm when programmed in sequential mode also exhibits speedup when compared to the non-split algorithm.

Key words. parallel algorithms, multisplitting, unconstrained optimization, QR decomposition, Householder QR decomposition, hypercube computer, variable metric method, DFGS update.

AMS subject classifications. 65H10, 65K05, 65K10, 90C30

1. Introduction. We consider herein the parallel solution of large nonlinear programming problems. All the problems to be considered can be described by one of the following equations:

Problem Formulation:

$$\begin{aligned} (1) \quad & f(x) = 0 \\ (2) \quad & \min_{x \in D} f(x) \\ (3) \quad & \min_{g(x) \leq 0} f(x). \end{aligned}$$

Here, in (1) and (2), $f(x)$ may refer to a set of linear or nonlinear equations for $x \in \mathcal{R}^n$, $f: \mathcal{R}^n \rightarrow \mathcal{R}^m$, $m \leq n$, or, in (2) and (3), $f(x)$ may be a nonlinear functional for which a minimum is required over either a subset $D \subseteq \mathcal{R}^n$, or the subset defined by the constraints $g(x) \leq 0$, respectively.

There are a variety of approaches that have been considered for the parallel solution of such problems. In particular, Coleman and Li [4] discuss parallel nonlinear equation solvers in which parallelism is introduced either

- (i) in the evaluation of the Jacobian matrix, particularly for separable $f(x)$,
- (ii) in the solution of the resulting system of linear equations or,
- (iii) by producing a rank-q secant update method.

Coleman and Plassman [5] consider a parallel least squares solver based on parallelization of the underlying orthogonal transformations required for the QR factorization. In these instances the emphasis is on parallelization of existing algorithms, and not on parallelization by problem decomposition. The latter approach is, however, more appealing because it offers the possibility of continued use of well-tested

* This research was supported in part by access to the Intel Paragons at Caltech, on behalf of the Concurrent Supercomputing Consortium, and at the San Diego Computing Center. Access to these facilities was provided by NSF.

[†] Department of Mathematics, Arizona State University, AZ 85287-1804. Fax. (602) 965 0461, Tel. (602) 965 3795, e-mail renaut@math.la.asu.edu.

[‡] Department of Mathematics, Arizona State University, AZ 85287-1804. Fax. (602) 965 0461, Tel. (602) 965 6595, e-mail mittelmann@math.la.asu.edu.

sequential algorithms for the solution of reduced-size subproblems. Algorithms which attack (2) and (3) in this way are presented, for example, by Ferris and Mangasarian [6], and Bertsekas and Tsitsiklis [2]. Although not presented as such, these latter algorithms can be considered as algorithms of multisplitting type. In this paper the multisplitting technique is extended to allow for the solution of the constrained problems (2) and (3). Results demonstrating the parallel (and sequential) efficiency of parallel multisplitting (MS) for (2) and (3) are given in Section 4. Sequential results are from implementation on an HP9000 735 workstation. The parallel algorithm was developed and implemented on the Intel Paragon, at the San Diego Supercomputing Center and at the California Institute of Technology, and on a cluster of workstations utilizing PVM3.

2. Review.

2.1. Multisplitting for $Ax = b$. Iterative methods based on a single splitting, $A = M - N$, are well known [11]. Multisplittings, O'Leary and White [10], take advantage of the computational capabilities of parallel computers. A multisplitting of A is defined as follows:

DEFINITION 2.1. *Linear Multisplitting (LMS)*

Given a matrix $A \in \mathcal{R}^{n \times n}$ and a collection of matrices $M^j, N^j, E^j \in \mathcal{R}^{n \times n}$, $j = 1 : p$, satisfying

- (i) $A = M^j - N^j$ for each j , $j = 1 : p$,
- (ii) M^j is nonsingular, $j = 1 : p$,
- (iii) E^j is a nonnegative diagonal matrix, $j = 1 : p$ and $\sum_{j=1}^p E^j = I$.

Then the collection of triples (M^j, N^j, E^j) , $j = 1 : p$ is called a multisplitting of A and the LMS method is defined by the iteration:

$$(4) \quad x^{k+1} = \sum_{j=1}^p E^j (M^j)^{-1} (N^j x^k + b), \quad k = 1, \dots$$

The advantage of this method is that at each iteration there are p independent iterations of the kind

$$(5) \quad M^j y_j^k = N^j x^k + b, \quad j = 1 : p,$$

where y_j^k represents the solution to the local problem. Hence the work for each equation in (5) is assigned to one (or a set of) processor(s) and communication is required only to produce the update given in (4). In general, some (most) of the diagonal elements in E^j are zero and therefore the corresponding components of y_j^k need not be calculated. If the y_j^k are distinct, $j = 1 : p$, the method corresponds to block Jacobi and is called nonoverlapping. This will be the case if the diagonal matrices E^j have only zero and one entries. Otherwise the algorithm uses overlap and optimal overlap has to be determined, see eg. [13].

2.2. Nonlinear splittings for $f(x) = 0$. The MS approach can also be extended to find the solution of a nonlinear set of equations. For example, consider finding the solution of (1), $m = n$, by a Newton method. A search direction d is found as the solution of the linear system

$$(6) \quad \nabla f(x^k) d + f(x^k) = 0.$$

Hence a solution can be found by applying a LMS to the matrix $\nabla f(x)$, [16]. Alternatively, a splitting can be applied directly to $f(x)$:

DEFINITION 2.2. *Nonlinear Multisplitting (NLMS)*

For $j = 1 : p$ let $F^j : \mathcal{R}^n \times \mathcal{R}^n \rightarrow \mathcal{R}^m$ be such that $F^j(x, x) = f(x)$ for all $x \in \mathcal{R}^n$, and E^j as in Definition 2.1. Then the collection of pairs (F^j, E^j) , $j = 1 : p$, is called a nonlinear multisplitting of f and the NLMS method is defined by the iteration:

$$(7) \quad x^{k+1} = \sum_{j=1}^p E^j y^{k,j}, \quad k = 0, 1, \dots$$

where $y^{k,j}$ solves $F^j(x^k, y^{k,j}) = 0$. This is the natural extension of Definition (2.1) since the LMS method for solving $f(x) = Ax - b$ can be split in the nonlinear sense by taking $F^j(x, y) = M^j y - N^j x - b$. For any separable nonlinear function $f(x)$, i.e. one which can be decomposed as $f(x) = \sum_{j=1}^n F_j(x)$, an obvious NLMS is obtained by taking $F^j(x, y) = F_j(y) + \sum_{i \neq j} F_i(x)$ and $E^j = \frac{1}{n}I$. This leads to a parallel iterative scheme of “alternating direction” or “Peaceman-Rachford” type, [11]. But this approach does not necessarily produce subproblems of reduced size since the Newton method for each $F^j(x, y) = 0$ requires the “inversion” of a Jacobian matrix for $F^j(x, y)$ of order n , unless $E_{ii}^j = 0$ for some set of i , $1 \leq i \leq n$. To reduce the subproblem size, and hence gain the advantage of the MS philosophy, suppose, instead, that (1) is solved with respect to several, possibly overlapping, blocks of variables.

DEFINITION 2.3. *Block Nonlinear Multisplitting (BNMS)*

Let $p \in \mathbf{Z}$ and for $j = 1 : p$ the block S^j a nonempty subset of $\{1, 2, \dots, n\}$ such that $\cup_{j=1}^p S^j = \{1, 2, \dots, n\}$, where the S^j need not be distinct. Take E^j as in Definition 2.1, but impose $E_{ii}^j = 0$ if $i \notin S^j$, and define projections $P^j : \mathcal{R}^n \rightarrow \mathcal{R}^n$ for $j = 1 : p$ by

$$(8) \quad P_i^j(x) = \begin{cases} x_i, & i \in S^j \\ 0, & i \notin S^j. \end{cases}$$

Define the functions $G^j(x, y) : \mathcal{R}^n \times \mathcal{R}^n \rightarrow \mathcal{R}^m$, $j = 1 : p$ by

$$G_i^j(x, y) = \begin{cases} F_i((I - P^j)x + P^j y), & i \in S^j \\ F_i(x_1, \dots, x_{i-1}, y_i, x_{i+1}, \dots, x_n), & i \notin S^j, \end{cases}$$

where F_i is a component of f . Then the pairs (G^j, E^j) , $j = 1 : p$ define a block nonlinear multisplitting of f and the BNMS method, [7], is defined by the iteration (7) where $y_i^{j,k}$ solves $G_i^j(x^k, y_i^{j,k}) = 0$.

This idea can then be applied to solve the linear least squares problem $\min_{x \in \mathcal{R}^n} \|Ax - b\|^2$, where $A \in \mathcal{R}^{m \times n}$, $x \in \mathcal{R}^n$ and $b \in \mathcal{R}^m$, if $f(x)$ in (1) is taken to be $f(x) = Ax - b$. In this case the BNMS of $f(x)$ amounts to a partition of the matrix A by columns and hence is not a member of the class of row-projection methods as described, for example, in [3]. Further, this approach also does not yield the same kind of iterative procedure as described in [12]. On the contrary, depending on the means for solving the subproblems, this technique can be seen as an iterative QR algorithm, or as any column-blocked iterative scheme of your choice, see eg [11]. Details and parallel implementation results of this column-blocked iterative QR algorithm are detailed in [14].

3. Optimization.

3.1. Unconstrained Minimization. Now consider the nonlinear form of (2), $m = 1$, D a bounded neighborhood of x^* , a unique stationary point of $f(x)$, and suppose that f has positive semidefinite Hessian $G = \nabla^2 f(x)$.

DEFINITION 3.1. *Nonlinear Optimization Multisplitting (NOMS)*

Let $f_j(x, Y_j) = f(\bar{x}_j)$, $f_j : \mathcal{R}^n \times \mathcal{R}^{n_j} \rightarrow \mathcal{R}^n$, $j = 1 : p$, x partitioned in blocks $x = (X_1, X_2, \dots, X_p)$, and \bar{x}_j the variable partition of x with the j^{th} component replaced by Y_j . Then the functions f_j form a NOMS of f and the stationary point is found iteratively from

$$(9) \quad x^{k+1} = \sum_{j=1}^p \alpha_j^{k+1} Z^{j,k+1},$$

where the $Z^{j,k}$ is x_k with update Y_j in the j^{th} position and Y_j solves the subproblem

$$(10) \quad \min_{Y_j \in \mathcal{R}^{n_j}} f_j, \quad 1 \leq j \leq p.$$

Here α_j^k are positive scalars such that $\sum_{j=1}^p \alpha_j^k = 1$. In a sequential environment, the minimization step does not proceed concurrently and hence the minimization can use all updated values available, as in a Gauss-Seidel iteration scheme. Here we present only the parallel algorithm, but the extension to allow for the Gauss-Seidel iteration in sequential mode is clear.

ALGORITHM 1. *Parallel Nonlinear Minimization*

For all processors i do

Begin

1. $k =: 0$;

2. guess X_i^k

3. do while not converged

(i) get all X_j^k for $j \neq i$

(ii) find Y_i^{k+1} to minimize $f_i(x^k, Y_i)$

(iii) update X_i^{k+1} via $X_i^{k+1} = \alpha_i^{k+1} Y_i^{k+1} + (1 - \alpha_i^{k+1}) X_i^k$

(iv) test for convergence

(v) $k = k + 1$

end do

End

Further details of this algorithm and proofs of convergence for exact and inexact Newton methods can be found in [14]. We note that in the particular case that f is separable one should expect to see convergence in one outer iteration if the subproblems are iterated to convergence. If, however, in the implementation a reduced tolerance is used on the subproblems this will not occur. We note also that this algorithm is significantly simpler to implement on a parallel architecture than that described in [5].

3.2. Constrained Minimization. Now we show that Algorithm 1 can be used for (3). Certainly, it is of great current interest to develop efficient methods for large sparse nonlinear programming problems. This is true even for serial computations, but parallelizable algorithms are even more desirable. In the following, one possible approach is outlined. The main problem faced when splitting constrained problems is, of course, lack of separability of the constraints. Short of reducing the problem to

TABLE 1

Serial results. Minimization of $f(x_1, x_2, \dots, x_n) = x^T Cx$, $c_{i,i} = 4 + \alpha, (1 \leq i \leq n), c_{i,j} = -1, (1 \leq |i - j| \leq 2), n = 16384$. Initialized with $x_i = n^{-\frac{1}{2}}, (1 \leq i \leq n)$. Terminated when $f \leq 10^{-6}$. Elapsed time on HP735 in seconds t . # of subproblems p . # of iterations K (in parantheses for Gauss-Seidel version). Subproblems minimized using truncated Newton code TN.

	$\alpha = 1$		$\alpha = .1$	
p	$K(GS)$	t	$K(GS)$	t
1	1 (1)	8.9	1 (1)	29.5
2	2 (2)	10.5	3 (2)	33.2
4	3 (3)	7.8	4 (3)	22.3
8	3 (3)	4.1	4 (3)	11.7
16	3 (3)	3.3	5 (3)	11.4
32	4 (3)	3.9	5 (3)	10.1
64	4 (3)	3.8	5 (3)	10.1
128	4 (3)	3.9	6 (3)	12.2
256	4 (4)	4.4	6 (4)	11.9
512	5 (4)	6.0	9 (4)	14.4
1024	5 (4)	6.7	20 (4)	27.3
2048	5 (4)	4.5	31 (4)	19.0
4096	8 (4)	4.5	62 (4)	34.9

TABLE 2

Serial results. Minimization of $f(x_1, x_2, \dots, x_n) = x^T Cx$, $c_{i,i} = 4 + \alpha, (1 \leq i \leq n), c_{i,j} = -1, (1 \leq |i - j| \leq 2), n = 512$, subject to $Ax \leq b, a_{i,i} = 2, (1 \leq i \leq n), a_{i,j} = -\frac{1}{n}, (i \neq j), b_1 = b_n = 2, b_i = -2, \text{otherwise}$. Initialized with $x_i = -2, (1 \leq i \leq n)$. Terminated when the relative difference in f is less than 10^{-5} . $\gamma = \frac{1}{\alpha}$. Elapsed time on HP735 in seconds t . # of iterations K (in parantheses for Gauss-Seidel version). Subproblems minimized using truncated Newton code TNBC.

	$\alpha = 2$		$\alpha = .5$	
p	$K(GS)$	t	$K(GS)$	t
1	1 (1)	5.1	1 (1)	12.3
2	18 (7)	28	27 (12)	93.5
4	16 (8)	10.4	24 (11)	16.2
8	19 (6)	5.0	31 (10)	10.4
16	19 (7)	2.6	33 (11)	6.0
32	19 (7)	1.2	33 (11)	2.4
64	19 (6)	.7	33 (10)	1.3
128	20 (6)	.6	34 (13)	1.1
256	19 (7)	.5	30 (10)	1.1

an unconstrained one and using the techniques of the previous section, the preferable alternative appears to be the reduction to a problem with separable constraints. One such way was suggested in [6], another one could be based on [15]. Both use exact penalty functions, an advantage of the latter would be that it is parameter-free. We choose the first approach. As was shown in [8] the constrained problem, (3), may under suitable conditions be replaced by

$$(11) \quad \min_x \min_{u \geq 0} \{-f(x) - u^T g(x) + \frac{1}{2} \gamma \|\nabla f(x) + \nabla g(x)u\|_2^2\}.$$

Here, γ is a positive factor which only needs to satisfy $\gamma \geq \bar{\rho}^{-1}$ where $\bar{\rho} \geq 0$ is the smallest eigenvalue of $\nabla_{xx} L(\bar{x}, \bar{u})$, (\bar{x}, \bar{u}) a Kuhn-Tucker point of (2) and $L(x, u) = f(x) + u^T g(x)$. Hence, (11), can be rewritten

$$(12) \quad \min_{y_j \geq 0, j > n} q(y), \quad y = (x, u)^T,$$

TABLE 3

Parallel results. Minimization of $f(x_1, x_2, \dots, x_n) = x^T C x$, $c_{i,i} = 4 + \alpha$, $(1 \leq i \leq n)$, $c_{i,j} = -1$, $(1 \leq |i - j| \leq 2)$, subject to $Ax \leq b$, $a_{i,i} = 2$, $(1 \leq i \leq n)$, $a_{i,j} = -\frac{1}{n}$, $(i \neq j)$, $b_1 = b_n = 3$, $b_2 = b_{n-1} = -2 + \frac{1}{n}$, $b_3 = b_{n-2} = -2 + \frac{2}{n}$, $b_i = -2 + \frac{4}{n}$, otherwise. Initialized with $x_i = -1$, $(1 \leq i \leq n)$. Terminated when the relative difference in f is less than 10^{-4} . $\gamma = \frac{1}{\alpha}$. Slowest elapsed time t , in seconds, measured for one processor on Intel Paragon using p nodes. Problem size n . In each case the splitting is determined by $\frac{2n}{p}$. A blank indicates that no result was obtained and the * indicates that the required convergence was not achieved. Subproblems minimized using truncated Newton code TNBC. Dense matrix version.

	$\alpha = 1$		$\alpha = .1$	
p	$n = 256$	$n = 512$	$n = 256$	$n = 512$
1	184*	3775	278	5426
4	65	1612	847	
8	12	206	34	1105
16	4	17	6	47
32	2	5	2	7

TABLE 4

Parallel results. Minimization of $f(x_1, x_2, \dots, x_n) = x^T x$, subject to $Ax \leq b$, $a_{i,i} = 2$, $(1 \leq i \leq n)$, $a_{i,j} = -\frac{1}{n}$, $(i \neq j)$, $b_1 = b_n = 3$, $b_i = -2 + \frac{4}{n}$, otherwise. Initialized with $x_i = 1$, $(1 \leq i \leq n)$. Terminated when the relative difference in f is less than 10^{-4} . $\gamma = 1$. Slowest elapsed time t , in seconds, measured for one processor on Intel Paragon using p nodes. Speedup S relative to sequential code on one processor. Problem size n . In each case the splitting is determined by $\frac{2n}{p}$. Subproblems minimized using truncated Newton code TNBC. Limited memory, minimal operations.

	$n = 8192$		$n = 16384$	
p	t	S	t	S
1	13.77	1.0	41.86	1.0
2	9.19	1.5	25.57	1.6
4	15.65	.9	25.64	1.6
8	8.57	1.6	19.39	2.2
16	4.36	3.2	9.35	4.5
32	2.93	4.7	3.88	10.8
64	3.00	4.6	4.45	9.4

and a solution of this problem with separable constraints can be found with a suitable generalization of the NOMS algorithm described above.

4. Results. Algorithm 1 was implemented and tested both in serial and parallel versions for both unconstrained and constrained test problems. In these tests f was taken to be convex. But, due to the fact that the algorithm parallelizes well and can handle large problems we expect this to be a worthwhile contribution. The results of some exemplary tests are given in the accompanying tables.

Before discussing these results in detail note also that a major consideration in parallel is to provide an efficient means for using the subproblem solutions to update the global solution. This can also be posed as a minimization procedure. In the implementations reported here this update was obtained using a refinement of Algorithm 1, in that a one-dimensional line search strategy was employed. The steplength α_i^{k+1} in Algorithm 1 is always chosen independently of i . Two approaches were used, one is a standard Goldstein-Armijo test for sufficient decrease with a maximum steplength of 1. A second strategy is based on the fact that due to the convexity of both the unconstrained and the constrained problem a steplength of $\alpha_i^{k+1} = \frac{1}{n}$ will always yield a descent but may be a rather small step. The steplength

TABLE 5

Serial results. Minimization of $f(x_1, x_2, \dots, x_n) = x^T C x$, $c_{i,i} = 4 + \alpha$, $(1 \leq i \leq n)$, $c_{i,j} = -1$, $(1 \leq |i - j| \leq 2)$, $\alpha = .1$. Various n . Initialized with $x_i = n^{-\frac{1}{2}}$, $(1 \leq i \leq n)$. Terminated when the relative difference in f is less than 10^{-5} . Elapsed time on HP735 in seconds for two different algorithms TN and LMVM.

n	TN	LMVM
1024	.3	.3
4096	3.4	1.6
16384	29.5	9.2
65536	175.8	46.2
262144	698.5	185.4

TABLE 6

Parallel results. Minimization of $f(x_1, x_2, \dots, x_n) = x^T C x$, $c_{i,i} = 4 + \alpha$, $(1 \leq i \leq n)$, $c_{i,j} = -1$, $(1 \leq |i - j| \leq 2)$, $\alpha = .1$. Initialized with $x_i = n^{-\frac{1}{2}}$, $(1 \leq i \leq n)$. # of subproblems p . # of iterations K . Subproblems minimized using limited-memory code LMVM. Elapsed times given for a PVM3 run on $p + 1$ workstations.

n	p	K	master	slave	total
900000	3	2	27	146	326
1500000	5	2	49	335	729
2100000	7	2	62	419	911

is chosen as $\alpha_j^{k+1} = \alpha_0 \beta^j$ where j is the smallest nonnegative integer such that

$$(13) \quad f_i(X_i^k + \alpha_j(Y_i^{k+1} - X_i^k)) \leq f_i(X_i^k + \alpha_{j+1}(Y_i^{k+1} - X_i^k)).$$

Here, α_0 is chosen as $2^{-\frac{1}{2}}$ in the unconstrained case and $\frac{1}{2}$ in the constrained case while β is taken as $2^{\frac{1}{2}}$ in both cases.

Results for an unconstrained case with quadratic f involving a pentadiagonal matrix whose condition is controlled by a parameter α are given in Table 1. For deteriorating condition the work per subproblem appears to increase but the saving due to the splitting is not affected except for a large number of processors. But, even in this latter case the Gauss-Seidel version is not affected. The same matrix in the objective function was used for the problem in Table 2, which also has a set of linear constraints defined by a full matrix. The previous remark concerning the condition also applies and it should be pointed out that the timings in this and the previous table are for serial implementations. Advantage was taken of the special structure of the matrices in the matrix-vector multiplies.

Results in both Tables 3 and 4 are for constrained optimization. Two versions were considered, one to simulate performance of a dense problem, and the other a minimal memory, minimal computation version, in Table 3 and Table 4, respectively. The excessively large times indicated in the former case occur due to the memory requirement for the storage of the dense matrices A and C . Because this memory requirement exceeds that available, virtual memory is used, resulting in excessive paging and consequent reduction in performance. Hence the results omitted are uninformative, and the one processor results do not provide a realistic means for comparison. Comparison, however, of the 8, 16 and 32 node timings shows the superlinear speedup predicted by the serial results. For Table 4 the form of A is taken advantage of in the coding to significantly reduce the number of flops. This is reflected in the suboptimal speedup, which indicates that ratio of computation to communication time is small, and hence that the communication overhead is significant. There are ways to min-

TABLE 7

Parallel results. Minimization of $f(x_1, x_2, \dots, x_n) = x^T x$, subject to $Ax \leq b$, $a_{i,i} = 2, (1 \leq i \leq n)$, $a_{i,j} = -\frac{1}{n}, (i \neq j)$, $b_1 = b_n = 2, b_i = -2$, otherwise. Initialized with $x_i = -2, (1 \leq i \leq n)$. Terminated when the relative difference in f is less than 10^{-5} . # of subproblems p . # of iterations K . Subproblems minimized using truncated Newton code TNBC. $\gamma = 1$. Elapsed times given for a PVM3 run on $p + 1$ workstations.

n	p	K	<i>master</i>	<i>slave</i>	<i>total</i>
150000	3	6	16	75	310
210000	5	6	19	109	382
270000	7	6	28	139	541

imise this effect, for example a "redundant-update" strategy could be employed. But there is no guarantee that this would actually reduce the total time, in particular, for less specialised A and C it is unlikely that this would be beneficial. These results clearly demonstrate that viable parallel implementations will be problem dependent, determined by the structure of the matrices and their condition.

For the solution of the unconstrained or bound-constrained subproblems two different algorithms were applied. The first one for which results are reported here is the truncated Newton method of [9]. An alternative newer algorithm is the limited-memory quasi-Newton method as implemented in LMVM [1]. To show the suitability of these methods for large-scale problems, and thus that of our approach to very large optimization problems, we present in Table 5 a comparison. For ease of comparison with the other results we have chosen the order n as power of 2.

It is clear that our splitting approach permits the solution of very large problems that would not fit into the (real) memory of most computers. Thus, the above variants of NOMS were implemented using PVM3 (version 3.2.6) on a cluster of workstations. This was done for both the unconstrained and the constrained case. We report on just one exemplary result for each case. One master process that does the outer iteration spawns several slave processes which solve the subproblems and communicate their results back to the master process. The master process needs to store just a few vectors of length n while each slave process stores vectors of length n/p and in addition the workarrays needed by the minimization routine. Results, for the latter being LMVM, and for a heterogeneous cluster of $p + 1$ workstations, are given in Table 6. The time given for the slave process denotes the average time needed for one subproblem of size n/p on the slowest workstation in the cluster. Table 7 contains corresponding results for a constrained case and TNBC as subproblem solver. Here, the subproblem size is $2n/p$.

5. Conclusions. The major advantage of these new classes of MS methods for optimization is that minimization procedures occur on independent subproblems. Not only can these subproblems be solved in parallel, hence allowing for the solution of large-scale problems in the networked workstation environment or on a parallel architecture, but also a large variety of sequential expertise can be employed for their solution. The results presented in Section 4 verify the feasibility of the approach and confirm, not only parallel speedup, but also, that sequential speedup can be achieved by choosing an optimal split. A Gauss-Seidel implementation further improves the sequential performance by reducing the number of iterations required to convergence.

Observe, also, that the one-dimensional line search employed to update the global solution could be handled by one processor while the remaining processors deal with the individual subproblem solutions. In this way synchronization of the individual

steps is not enforced and convergence is accelerated. Furthermore, it is also possible to implement the algorithm using an overlapping strategy for the subproblems, see [13]. These latter two issues will be discussed in a future paper.

REFERENCES

- [1] B. M. Averick, R. G. Carter, J. J. Moré and G. L. Xue, *The Minpack-2 Test Problem Collection*, ANL/MCS-P153-0692, (1992), Argonne National Laboratory.
- [2] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation*, Prentice Hall, (1989).
- [3] R. Bramley and A. Sameh, *Row Projection Methods for Large Nonsymmetric Linear Systems*, SIAM J. Sci. Stat. Comput., Vol. 13, No. 1, (1992), pp. 168-193.
- [4] T. F. Coleman and G. Li, *Solving Systems of Nonlinear Equations on a Message-Passing Multiprocessor*, SIAM J. Sci. Stat. Comput. 11, (1990), pp. 1116-1135.
- [5] T. F. Coleman and P. E. Plassmann, *A Parallel Non-Linear Least-Squares Solver: Theoretical Analysis and Numerical Results*, SIAM J. Sci. Stat. Comput. 13, 3, (1992), pp. 771-793.
- [6] M. C. Ferris and O. L. Mangasarian, *Parallel Variable Distribution*, CS Dept., University of Wisconsin (1993)
- [7] A. Frommer, *Parallel Nonlinear Multisplitting Methods*, Numer. Math. **56**, (1989), pp. 269-282.
- [8] S.-P. Han and O. L. Mangasarian, *A Dual Differentiable Exact Penalty Function*, Mathematical Programming 25, (1983), 293-301.
- [9] S. G. Nash, *Newton-type Minimization via the Lanczos Method*, SIAM J. Numer. Anal. 21, (1984), pp. 770-778.
- [10] D. P. O'Leary and R. E. White, *Multi-Splitting of Matrices and Parallel Solution of Linear Systems*, SIAM J. Alg. Disc. Meth. 6, (1985), pp. 630-640.
- [11] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, New York, Academic Press, (1970).
- [12] E. P. Papadopoulou, Y. G. Sardiakis, and T. S. Papatheodorou, *Block AOR Iterative Schemes for Large-Scale Least-Square Problems*, SIAM J. Numer. Anal., Vol. 26, No. 3, (1989), pp. 637-660.
- [13] B. Pohl, *Ein Algorithmus zur Lösung von Anfangswertproblemen auf Parallelrechnern*, Informatik-Dissertationen ETH Zürich NR. 37, (1992).
- [14] R. A. Renaut and Qing He, *Parallel Multisplittings for Unconstrained Optimization*, in prep. (1994).
- [15] C. Richter, *Zur globalen Konvergenz des gedämpften Wilsonverfahrens*, Mathem. Operationsforsch. Statist., Ser. Optimization, 10, (1979), 213-218.
- [16] R. E. White, *Parallel Algorithms for Nonlinear Problems*, SIAM J. Alg. Disc. Meth. **7**, (1986), pp. 137-149.