Problem	MS tasks	Jacobi?	Newton itns	KMS itns	line search itns
Rosenbrock	8	N	12	12	12
Rosenbrock	2	Y	13	25	13
Powell	8	N	9	9	9
Powell	2	Y	9	32	9
Trig	8	N	3	9	3
Trig	2	Y	3	68	3
Elliptic	2	N	2	19	2

TABLE 1

Performance of the KMS Inexact Newton Algorithm. The algorithm was stopped when $f(x) < 10^{-5}$. *The forcing sequence was* $\epsilon_k = 10^{-3}$.

[5] P. CONCUS, G. H. GOLUB, AND D. P. O'LEARY, *Numerical solution of nonlinear elliptic partial differential equations by a generalized conjugate gradient method*, Computing, 19 (1978), pp. 321–339.

[6] R. S. DEMBO, S. C. EISENSTAT, AND T. STEIHAUG, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[7] R. S. DEMBO AND T. STEIHAUG, *Truncated-Newton algorithms for large-scale unconstrained optimization*, Mathematical Programming, 26 (1983), pp. 190–212.

[8] J. DENNIS JR. AND R. B. SCHNABEL, *Numerical Methods for Unconstrained Optimization and Nonlinear Equation*, Prentice-Hall, New Jersey, 1893.

[9] C.-M. HUANG AND D. P. O'LEARY, *A Krylov multisplitting algorithm for solving linear systems of equations*, Linear Algebra and Its Applications, to appear.

[10] ———, *Preconditioning parallel multisplittings for solving linear systems of equations*, in 6th ACM International Conference on Supercomputing, Washington, DC, July 1992.

[11] J. J. MORÈ, B. S. GARBOW, AND K. E. HILLSTROM, *Testing unconstrained optimization software*, ACM Transactions Mathematical Software, (1981), pp. 17–41.

[12] S. G. NASH AND A. SOFER, *A parallel line search for a Newton-type method*, in Computer Science and Statistics: Proceedings of the 21st Symposium on the Interface, K. Berk and L. Malone, eds., Alexandria, VA, 1990, American Statistical Assoc., pp. 134–137.

[13] D. P. O'LEARY, *A discrete Newton algorithm for minimizing a function of many variables*, Mathematical Programming, 23 (1982), pp. 20–33.

[14] D. P. O'LEARY AND R. E. WHITE, *Multi-splittings of matrices and parallel solution of linear systems*, SIAM J. Disc. Math., 6 (1985), pp. 630–640.

[15] R. E. WHITE, *Multisplitting with different weighting schemes*, SIAM J.Matrix Anal. Appl., 10 (1989), pp. 481–493.

We place a uniform mesh of width $\Delta = \frac{1}{32}$ on the unit square and denote the approximation to $v(x, y)$ at the mesh point $x = i\Delta$, $y = j\Delta$ by $u_{i,j}$. Then at an interior point we obtain, using the standard five-point discretization,

$$\frac{1}{h^2}(-u_{i,j-1} - u_{i-1,j} + 4u_{i,j} - u_{i+1,j} - u_{i,j+1}) + (1 - e^{-5x_i})e^{u_{i,j}} = 1.$$

with $\text{stp}_{\max} = 6$ and $m = 4$ An "inner" SSOR splitting is used in this Example.
$\diamond$

For all of the examples, the convergence criterion for the function F is 1.0^{-5} and the finite difference parameter was $h = 1.0^{-6}$.

We summarize the numerical results in Table 1. Some test cases made use of an "inner" Jacobi splitting while in others the multisplitting equations were solved directly. In all of our tests, the Inexact Newton algorithm with KMS converged rapidly.

The work involved in determining the search direction for the inexact Newton algorithm with KMS is quite parallel. Set up requires n/p evaluations of the function g in each multisplitting task $Task_l$ in order to evaluate M. For each KMS iteration we need one gradient evaluation in each multisplitting task and one in the accumulation task $Task_0$ to form H times a new column. In many cases, only part of the gradient vector is required by each task, so the work can be further reduced. The line search requires one evaluation of f per task per iteration.

7. Conclusions. We have implemented an inexact Newton algorithm using a Krylov subspace method for solving large scale systems of nonlinear equations and unconstrained optimization problems on machines with MIMD architecture. The number of synchronization points between the multisplitting tasks and the accumulation task is greatly reduced. This implementation has many advantages for parallel computations:

1. There is more flexibility in this algorithm for adding or dropping directions from the Krylov subspace than in traditional implementations.
2. The KMS algorithm can be an efficient way to solve the Newton equation in parallel machines.
3. Given enough processors, the number of function evaluations is small in each multisplitting task. With the exploitation of sparsity of the Hessian matrix, this number can be further decreased.
4. The algorithms are locally convergent. A local quadratic rate of convergence is achievable, and global convergence can be achieved under additional assumptions using a line search.

For future research, there is work remaining to be done in using the structure of the problems in order to develop effective multisplittings.

REFERENCES

[1] C. Bischof, A. Carle, G. Corliss, A. Griewank, and P. Hovland, *Generating derivative codes from* FORTRAN *programs*, Scientific Computing, (to appear).

[2] P. N. Brown, *A local convergence theory for combined inexact-Newton / finite difference projection methods*, SIAM J. on Numer. Anal., 24 (1987), pp. 407–434.

[3] P. N. Brown and Y. Saad, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM J. on Sci. and Stat. Computing, 11 (1990), pp. 450–481.

[4] T. Chan, *Rank-revealing QR factorization*, Lin. Alg. and Its Applics., 88/89 (1987), pp. 67–82.

where γ is chosen so that the largest step-length is equal to $\text{stp}_{\max}$. If there is no step-length satisfying the Goldstein-Armijo criteria (with $\alpha = 1.0^{-4}$ and β between 0.7 and 0.9), then the line search can be repeated with $\text{stp}_{\max}$ divided by 2. For further detail see [12].

The multisplitting we use is the block Jacobi splitting: let M be the block diagonal part of the Hessian, where each block has dimension n/p. Referring to equations (7) and (8), we define $M_l = M$, $l = 1, ..p$ and D_l to be zero, with an identity matrix in the lth diagonal block. Linear systems involving the matrix M_l are either solved directly or solved iteratively using 5 iterations of the (point) Jacobi splitting (i.e., preconditioning by the diagonal elements). $Task_0$ minimized the norm of the residual over the span of all vectors generated by the multisplitting tasks.

Example 1. Extended Rosenbrock function [11]

$$\begin{aligned} g_{2j-1}(x) &= 10(x_{2j} - x_{2j-1}^2), \\ g_{2j}(x) &= 1 - x_{2j-1}, \end{aligned}$$

with the number of variables $n = 64$. The inital guess was 1 for the even components of x_0 and -1.2 for the odd components. We used the parameters $\text{stp}_{\max} = 4$ and $m = 8$.

Example 2. Extended Powell singular function [11]

$$\begin{aligned} g_{4j-3}(x) &= x_{4j-3} + 10x_{4j-2} \\ g_{4j-2}(x) &= 5^{1/2}(x_{4j-1} - x_{4j}) \\ g_{4j-1}(x) &= (x_{4j-2} - 2x_{4j-1})^2 \\ g_{4j}(x) &= 10^{1/2}(x_{4j-3} - x_{4j})^2 \end{aligned}$$

with $n = 64$. The initial guess was $x_0 = (3, -1, 0, 1, ..., 3, -1, 0, 1)$. We used the parameters $\text{stp}_{\max} = 8$ and $m = 8$.

Example 3. Trigonometric function [11]

$$g_j(x) = n - \sum_{l=1}^{n} \cos x_l + j(1 - \cos x_j) - \sin x_j$$

with $n = 64$, $\text{stp}_{\max} = 8$ and $m = 16$.

The initial guess was $x_0 = 10(1/n, \ldots, 1/n)^T$ The two-stage method with an "inner" Jacobi splitting is expensive for the Trigonometric function, since its second derivative is a dense matrix.

Example 4. Consider a nonlinear elliptic equation [5]

$$-v_{xx} - v_{yy} + (1 - e^{-5x})e^v = 1 \tag{14}$$

Equation (14) is to be solved on the unit square subject to the boundary conditions

$$\begin{aligned} &\text{on} \quad x = 0: \quad v = 0 \\ &\text{on} \quad x = 1: \quad v = 1 \\ &\text{on} \quad y = 0 \quad \text{and} \quad y = 1: \quad v = x. \end{aligned}$$

4.2. Using Difference Quotients to Approximate Matrix-Vector Products. If the matrix H, or a symmetric approximation to it, is not available, then the conjugate gradient method is not guaranteed to produce a downhill direction. Using (10) to approximate the product of $H(x)$ with a vector v effectively produces a nonsymmetric approximation to the Hessian matrix, and the conjugate gradient algorithm may fail to converge. In practice, the scheme works well, but it is not possible to prove convergence [2]. If global convergence is desired, then we should apply the algorithms in §3 to the problem of minimizing $\|g(x)\|$.

5. Multisplitting Examples for Solving the Newton Equation.

Example 1. Let the Hessian matrix be split as $H(x) = M(x) - N(x)$, where

$$M(x) = \begin{pmatrix} M_1(x) & & & \\ & \cdot & & \\ & & \cdot & \\ & & & \cdot \\ & & & & M_p(x) \end{pmatrix},$$

is a block diagonal matrix. Each $M_l(x)$, $1 \le l \le p$, is a relatively small size matrix, and we assume that each is evaluated explicitly. Then each multisplitting task $Task_l$ solves or approximates the equation

$$M_l(x) z_l^{j+1} = N_l(x) z^j + b_l.$$

If $N_l(x)$ is not explicitly available, we can approximate the product $N_l(x)z^j$ by

$$N_l(x) z^j \approx \frac{g^{(l)}(x + hv) - g^{(l)}(x)}{h}$$

where v equals z^j, except that its l-th block is set to zero and $g^{(l)}$ denotes the lth block of g. Thus, if necessary, we can perform a block splitting without evaluating the off-diagonal blocks of the derivative matrix.

Thus we can solve the multisplitting equations without evaluating the Hessian matrix.

Example 2. We can replace the equation $M_l(x) z_l^{j+1} = N_l(x) z^j + b_l$ using a splitting $M_l(x) = F_l(x) - G_l(x)$. Define $c_l = N_l(x) z^j + b_l$, and set $y_l^0 = 0$. Then we use the *inner iteration* $F_l(x) y_l^{m+1} = G_l(x) y_l^m + c_l$ for $0 \le m \le s-1$ to form a vector $z_l^{j+1} = y_l^s$. As above, it is not necessary to explicitly calculate $G_l(x)$.

6. Numerical Examples. In this section we report the results of some numerical experiments using the Inexact Newton Method with KMS.

The test problems of this section are solved on MATLAB, and a parallel line search described in [12] was simulated. Let assume that we have m processors, each capable of evaluating the objective function $f(x)$. The function is evaluated at m distinct points with maximum step-length $\mathrm{stp}_{\max}$. We use the first m points of the sequence

$$\{1, 1/2, \gamma, 1/4, \gamma^2, 1/8, \gamma^3, \ldots\}, \quad \text{if } \mathrm{stp}_{\max} > 1$$

$$\{\gamma, \gamma/2, \gamma/4, \ldots\}, \quad \text{if } \mathrm{stp}_{\max} \le 1$$

The first term is the computed residual, and the second can be bounded in terms of the finite difference parameter h_k and a Lipschitz constant for H (Lemma 4.1.12 of [8]:

$$|g(x_k)^T r_k| \leq |g(x_k)^T r_k^{\text{computed}}| + \frac{\gamma}{2} h_k \|g(x_k)\| \|d_k\|$$

Thus, if h_k and ϵ_k are small enough, the direction will be downhill.

We add a Goldstein-Armijo line search in order to ensure that progress is made at each iteration: we choose parameters $\alpha \in (0,1)$ and $\beta \in (\alpha, 1)$, and we demand that the line search parameter t_k is chosen so that $s_k = t_k d_k$ satisfies

$$F(x_{k+1}) < F(x_k) + \alpha g(x_k)^T H(x_k) s_k \tag{12}$$

$$g(x_{k+1})^T H(x_{k+1}) s_k \geq \beta g(x_k)^T H(x_k) s_k . \tag{13}$$

THEOREM 3.3. **Global Convergence Conditions** *In addition to the assumptions of Theorem 3.1 or Theorem 3.2, assume that the line search produces a step that satisfies (12) and (13), $\alpha < 1/2$, and the sequences $\{h_k\}$ and $\{\epsilon_k\}$ are sequences are chosen to ensure that each direction is downhill. Then the algorithm will be globally convergent, and the convergence rate is ultimately superlinear.*

Proof. This result follows from standard arguments using Theorems 6.3.3 and 6.3.4 in Dennis and Schnabel [8]. □

4. KMS Truncated-Newton for Unconstrained Optimization. In this section we consider Problem 2, the unconstrained optimization problem.

4.1. Using the Hessian or a Symmetric Approximation to It. If it is possible to form matrix-vector products involving the Hessian matrix $A \equiv H(x_k)$ (computed, for example, from ADIFOR [1]) or a symmetric positive definite approximation $A \approx H(x_k)$, then the KMS-CG algorithm can be used to find an approximation to the Newton direction. The multisplitting matrix G of (6) should be symmetric positive definite.

Although not required to prove local convergence of the algorithm, a satisfying consequence of using the KMS-CG algorithm is that each direction d_k computed by the algorithm is guaranteed to be downhill for the Newton iteration, independent of the choice of finite difference parameter h_k or relative residual tolerance ϵ_k. To see this, we use the fact that d_k minimizes the error function (9) over all vectors in the space spanned by the columns of Q_j. Thus,

$$d_k = -Q_j (Q_j^T A Q_j)^{-1} Q_j^T g(x_k).$$

It then follows that $g(x_k)^T d_k < 0$ if A is positive definite.

The other KMS variants can also be used, but the downhill property is not common to all of them.

The local convergence of the inexact Newton algorithm using KMS to find the search direction follows from Theorem 3.1 of §3, when step lengths $t_k = 1$ are used.

For global convergence, we use a Goldstein-Armijo line search: we choose parameters $\alpha \in (0,1)$ and $\beta \in (\alpha, 1)$ and demand that the line search parameter t_k is chosen so that $s_k = t_k d_k$ satisfies

$$f(x_{k+1}) < f(x_k) + \alpha g(x_k)^T s_k$$

$$g(x_{k+1})^T s_k \geq \beta g(x_k)^T s_k .$$

Then Theorem 3.3 guarantees global, and ultimately quadratic, convergence.

Choose $\epsilon_k = \frac{1}{2}\eta_k \|g(x_k)\|_2$, *and suppose that*

$$\frac{\delta\gamma h_k}{\sigma_{\min}} < \min\left(1, \frac{\eta_{\max}\|g(x_k)\|_2}{2}\right).$$

Then the iterates satisfy (11).

Proof. If finite differences are used to form the matrix-vector, then in order to apply Brown's result, we need to verify that $\|r_k\|_2 \leq \eta_k \|g(x_k)\|_2^2$. We note that $h_k \leq \delta/\sqrt{n}$, and we denote the finite difference approximation to the product $H(x_k)v$ by $[H(x_k)v]_h$. Then

$$\|r_k\|_2 \leq \|r_k^{comp}\|_2 + \|[H(x_k)Q]_h - H(x_k)Q\|_2 \|z\|_2,$$

where $d_k = Qz$ is the solution computed by the accumulation task and r_k^{comp} is the corresponding residual computed using finite differences. By Lemma 4.1.12 of [8],

$$\|g(x_k + h_k v_j) - g(x_k) - h_k H(x_k)v_j\|_2 \leq \frac{\gamma}{2}h_k^2,$$

so

$$\|[H(x_k)Q]_h - H(x_k)Q\|_2 \leq \frac{\gamma}{2}h_k^2\sqrt{n} \leq \frac{\delta\gamma h_k}{2}.$$

For R-KMS-GMRES, the solution is given by $z = R^{-1}\hat{g}$, where R is the right-triangular factor in the QR factorization of $[H(x_k)Q]_h$ and $\hat{g}$ has the same norm as $g(x_k)$. Therefore,

$$\|z\|_2 \leq \|R^{-1}\|_2 \|g(x_k)\|_2.$$

The norm of R^{-1} is its smallest singular value, equal to the inverse of the smallest singular value $\tau_{\min}$ of $[H(x_k)Q]_h$, where

$$\tau_{\min} > \sigma_{\min} - \frac{\delta\gamma h_k}{2} = \sigma_{\min}(1 - \frac{\delta\gamma h_k}{2\sigma_{\min}}) \geq \frac{\sigma_{\min}}{2}.$$

Therefore,

$$\|r_k\|_2 \leq \frac{1}{2}\eta_{\max}\|g(x_k)\|_2^2 + \frac{\delta\gamma h_k}{2}\frac{2}{\sigma_{\min}}\|g(x_k)\|_2 \leq \eta_{\max}\|g(x_k)\|_2^2$$

as desired. □

A locally linear rate of convergence can be established under a weaker condition for the KMS tolerances ϵ_k: $\epsilon_k \leq \eta_k$ [2].

We can obtain a global convergence result if we add some further restrictions, as discussed by Brown and Saad [3]. For these results, we return to the minimization formulation, $F(x) = \frac{1}{2}\|g(x)\|^2$. We note that any search direction d_k produced by the KMS algorithm is a downhill direction for $F(x)$ as long as the residual $r_k = H(x_k)d_k + g(x_k)$ is small enough, since

$$-g(x_k)^T H(x_k)d_k = g(x_k)^T g(x_k) - g(x_k)^T r_k,$$

and $H(x_k)^T g(x_k)$ is the gradient of $F(x_k)$. To ensure that the direction is downhill, it is sufficient to force the relative residual (3), evaluated with the exact matrix $H(x_k)$, to be less than 1. If we are using an approximate derivative matrix, then

$$g(x_k)^T r_k = g(x_k)^T(Ad_k + g(x_k)) + g(x_k)^T(H(x) - A)d_k.$$

2. **R-KMS-Arnoldi:** Make the residual $g + Ad$ orthogonal to all vectors in the Krylov subspace.
3. **R-KMS-GMRES:** Minimize the norm of the residual $\|g + Ad\|$ over all vectors d in the Krylov subspace spanned by the columns of Q_j.

If $\frac{||g+Ad||_2}{||g||_2} \leq \epsilon_k$ then send halt signal to $Task_1, \ldots, Task_p$, and Exit.

endfor

3. KMS Truncated-Newton for Nonlinear Equations. The GMRES or Arnoldi variants (left or right preconditioning) of the KMS algorithm can be used to compute an approximate Newton direction for a system of nonlinear equations.

Matrix-vector products involving the derivative matrix can be evaluated, for example, using the ADIFOR package of Bischof et al [1]. This package generates a representation of the derivative matrix from the FORTRAN code for evaluating $g(x)$.

If a representation of the derivative matrix is not available, a common method for approximating the product of H times a vector v is to form the difference quotient

$$[H(x)v]_h \equiv \frac{g(x + hv) - g(x)}{h}. \tag{10}$$

In the limit as $h \to 0$ this quantity equals Hv. Unfortunately, though, the basis generated using finite differences can differ quite substantially from that for the Krylov subspace, and this makes the convergence analysis more difficult.

The local convergence of the inexact Newton algorithm using KMS to find the search direction follows from results of Brown [2], who studied the use of the GMRES and Arnoldi iterations to compute an inexact Newton direction. This result is applicable only because we use an orthogonal basis in $Task_0$. His result, applied to our algorithm, can be restated as follows:

THEOREM 3.1. *Brown Thm 2.2 [2]* **Local Convergence Conditions** *Assume that there exists a point x^* such that $g(x^*) = 0$, $H(x^*)$ is nonsingular, and $H(x)$ is Lipschitz continuous in a neighborhood of x^* with constant γ. Let the largest and smallest singular values of $H(x^*)$ be denoted as $\sigma_{\max}$ and $\sigma_{\min}$. Let the KMS tolerance be $\epsilon_k = \eta_k \|g(x_k)\|_2$, where $0 \leq \eta_k \leq \eta_{\max} < 1$.*

Then there exists a $\beta > 0$ such that, if $\|x_0 - x^*\| \leq \beta$, then the sequence $\{x_k\}$ formed by using steplengths $t_k \equiv 1$ is well defined, convergent to x^*, and satisfies

$$\|x_{k+1} - x^*\| \leq \frac{1}{\sigma_{\min}} \left(\gamma + 2\eta_{\max} \left(\sigma_{\max} + \frac{\gamma\beta}{2} \right)^2 \right) \|x_k - x^*\|^2, \quad k = 0, 1, 2, \ldots. \tag{11}$$

Some further assumptions are needed to apply this result to finite difference approximations. We use R-KMS-GMRES as an example of such a result.

THEOREM 3.2. *In addition to the assumptions of Thm 3.1, suppose R-KMS-GMRES is used in $Task_0$ and that the finite difference step lengths h_k used at iteration k form a nonincreasing sequence, small enough that the vectors $\{(g(x_k + h_k v_j) - g(x_k))/h_k\}$ computed in $Task_0$ are linearly independent, $j = 1, 2, \ldots, m$. Assume that there exists a positive constant δ such that*

$$\|[h_1, h_2, \ldots, h_{\max(n,k)}]\|_2 \leq \delta.$$

Although these algorithms are quite efficient implementations of the GMRES or Arnoldi iterations, they are inefficient for the conjugate gradient iteration if many iterations are required, since all of the old vectors need to be stored.

The Right-KMS algorithm corresponds to a change of variables: instead of solving $Ad = -g$, we solve $A\hat{G}\hat{d} = -g$, where $d \equiv \hat{G}\hat{d}$ and

$$\hat{G} \equiv \sum_{l=1}^{p} M_l^{-1} D_l .$$

The resulting Krylov subspace is $\mathcal{K}(A\hat{G}, g, k) \equiv \text{span}\{g, A\hat{G}g, ...(A\hat{G})^{k-1}g\}$. We see that

$$\begin{aligned} A\hat{G} &= \sum_{l=1}^{p} A M_l^{-1} D_l \\ &= \sum_{l=1}^{p} (M_l - N_l) M_l^{-1} D_l \\ &= I - \sum_{l=1}^{p} N_l M_l^{-1} D_l \\ &\equiv I - \hat{B}. \end{aligned}$$

Thus the Krylov subspace $\mathcal{K}(A\hat{G}, g, k)$ is equivalent to $\mathcal{K}(\hat{B}, g, k)$.

The resulting algorithm is as follows.

The Right-KMS Algorithm

Algorithm for multisplitting $Task_l$, $l = 1, ..., p$:

Initialize $\hat{g}^0 = g$.
Form $M_l^{-1} D_l \hat{g}^0$
For $j = 0, 1, \ldots$, until receiving a halt signal from $Task_0$,
Form $N_l(M_l^{-1} D_l \hat{g}^j)$ and participate with the other multisplitting tasks in forming $\hat{g}^{j+1}$ by summing these values.
Form $M_l^{-1} D_l \hat{g}^{j+1}$.
Send $\hat{G}(\hat{g}^{j+1} - \hat{g}^j)$ to $Task_0$ for accumulation.
endfor

Algorithm for accumulation $Task_0$:

Given termination criteria ϵ_k.
For $j = 0, 1, \ldots$
Receive vector v_j from the multisplitting tasks.
Update orthogonal basis Q_j to include v_j. The last column q_j is our new basis vector.
Choose d to satisfy one of the following conditions:

1. **R-KMS-CG:** Minimize the error function

$$\|d - d^*\|_A^2 \equiv (d - d^*)^T A (d - d^*)$$

over all vectors d in the Krylov subspace spanned by the columns of Q_j.

Receive the latest multisplitting iterate, $\hat{z}^j$ and call it z^0.
Determine z^1 by solving $M_l z^1 = N_l z^0 - g$.
Form $z_l^{j+1} = D_l z^1$, and participate with the other multisplitting tasks in forming $\hat{z}^{j+1}$ by summing the z_l^{j+1}, $l = 1, ..., p$.
Send $\hat{z}^{j+1} - \hat{z}^j$ to $Task_0$ for accumulation.
endfor

Algorithm for accumulation $Task_0$:
Given termination criteria ϵ_k.
For $j = 0, 1, \ldots$
Receive vector v_j from the multisplitting tasks.
Update orthogonal basis Q_j to include v_j. The last column q_j is the new basis vector.
Choose d to satisfy one of the following conditions:

1. **L-KMS-CG:** Minimize the error function

$$\|d - d^*\|_A^2 \equiv (d - d^*)^T A (d - d^*) \tag{9}$$

over all vectors d in the Krylov subspace spanned by the columns of Q_j.
2. **L-KMS-Arnoldi:** Make the residual $G(g + Ad)$ orthogonal to all vectors in the Krylov subspace.
3. **L-KMS-GMRES:** Minimize the norm of the residual $\|G(g + Ad)\|$ over all vectors d in the Krylov subspace spanned by the columns of Q_j.

If $\frac{||g+Ad||_2}{||g||_2} \leq \epsilon_k$ then send halt signal to $Task_1, \ldots, Task_p$, and exit.
endfor

In theory, the vectors v_j are guaranteed to be linearly independent as long as the residual vector is nonzero. It is possible that finite difference approximations to H could produce linear dependence. In this case, we can either generate a new basis vector for Q_j orthogonal to the previous ones or restart the iteration from the current d vector.

There are several variants on this basic algorithm that can improve the parallel utilization. For example, the multisplittings can run several iterations at a time before generating the next Krylov vector, or two stage (inner-outer) methods can be used to solve linear systems involving the matrix M_l. See [10] and §5 for more details.

When the conjugate gradient variant is used (L-KMS-CG), the orthogonalization of the basis vectors ensures that the matrix $A_j = Q_j^T A Q_j$ involved in determining the vector d is at least as well-conditioned as A is. We use a rank-revealing QR factorization of $A_j P = QR$ where the rectangular matrix Q is orthogonal, R is an upper triangular matrix, and the columns of A_j causing near linear dependencies are pushed to the right by the permutation matrix P_j. We pick a maximal leading principal submatrix R_a of R that corresponds to a well conditioned subset of the orthogonal basis Q_a, and solve a reduced system. (See [4] and [9]). Thus the minimization of the error function is performed over a subspace of the Krylov subspace if ill-conditioning is evident. The rank-revealing factorization is updated at each iteration using standard techniques. Analogous techniques are used for L-KMS-A and L-KMS-GMRES.

Multisplittings induce the iterative method $z^{j+1} = Bz^j + Gb$, computed by forming

$$z^{j+1} = \sum_{l=1}^{p} D_l \hat{z}_{(l)}^{j+1}, \tag{7}$$

where

$$M_l \hat{z}_{(l)}^{j+1} = N_l z_{(l)}^j + b. \tag{8}$$

Such iterations can be painfully slow unless accelerated by Krylov subspace iteration.

Huang and O'Leary ([9, 10]) have developed a Krylov multisplitting algorithm (KMS algorithm) that can achieve much higher parallelism than other implementations of Krylov subspace iterations. In this algorithm there are $p + 1$ tasks, p for the multisplitting and one for accumulation of the approximate solution for the linear system. Each splitting is assigned to a subset of processors. The multisplittings report individually to the accumulation task and do not need to wait for a response. This reduces waiting time at the cost of some additional complication in the accumulation, and it is shown that the subspace over which the error function is minimized equals the Krylov subspace used in the standard algorithm. Although no synchronization signals are sent from the accumulation task to the multisplittings, the accumulation task can be implemented in a way that makes the algorithm deterministic rather than chaotic. In practice, due to round-off error, the accumulation task must periodically reinitialize the multisplittings, but this can be done infrequently. (For more detail, see [9].)

We are concerned in this work with algorithm variants that preserve superlinear convergence of the inexact Newton iteration. In §2 we present some variants of the Krylov Multisplitting (KMS) algorithm. In §3 we discuss the use of these algorithms in solving Problem 1 (nonlinear equations). Convergence of the algorithms is verified. Problem 2 (unconstrained optimization) is studied in §4. Section 5 gives some example multisplittings for Newton equations. The results of numerical examples on sequential machines are presented in §6.

2. Some Variants of the KMS algorithm. We study two classes of algorithms: the left-preconditioned KMS algorithm and the right-preconditioned KMS algorithm. Right multisplittings were first proposed by White [15], and we now propose their use as KMS preconditioners (postconditioners).

Suppose we wish to solve the nonsingular linear system $Ad = -g$. The Left-KMS algorithm corresponds to a rescaling of the linear system to

$$GAd = -Gg,$$

where G is defined by (6). The Left-KMS algorithm generates the Krylov subspace

$$\mathcal{K}(GA, Gg, k) \equiv \text{span}\{Gg, GAGg, ..., (GA)^{k-1}Gg\}$$

In the following algorithm, $Task_0$ accumulates the approximate solution to the linear system, and the multisplitting tasks are denoted by $Task_l$, $l = 1, \ldots, p$.

The Left-KMS Algorithm

Algorithm for multisplitting $Task_l$, $l = 1, ..., p$:

Initialize $\hat{z}^0 = 0$.

For $j = 0, 1, \ldots$, until receiving a halt signal from $Task_0$,

Inexact Newton methods form an approximate solution to this equation, so that the residual

$$r_k \equiv H(x_k)d_k + g(x_k)$$

satisfies

$$\frac{||r_k||}{||g(x_k)||} \leq \epsilon_k . \tag{3}$$

The quantity on the left of (3) is sometimes called the *relative residual*; the nonnegative *forcing sequence* $\{\epsilon_k\}$ is used to control the level of accuracy when solving the linear system, allowing a rather inaccurate Newton direction when far from the solution x^* and expending greater effort in the computation of the direction when Newton's method is working from a point close to x^* [7, 6, 13].

Combining the inexact Newton methods with a line search, and exploiting parallelism in the algorithm, we have following schema for solving the problems:

A Parallel Inexact Newton Method

For $k = 0, 1, \ldots$ until convergence
Use a parallel algorithm to find some step d_k so that
$r_k = H(x_k)d_k + g(x_k)$ satisfies $\frac{||r_k||}{||g(x_k)||} \leq \epsilon_k$.
Perform a parallel line search, and produce a step
$s_k = t_k d_k$, a multiple of d_k.
Set $x_{k+1} = x_k + s_k$.

In order to obtain an efficient parallel algorithm, it is crucial that the solution to the linear system and the line search both have good parallel utilization. In many situations the line search can be avoided by setting $t_k \equiv 1$ for all k. Parallel line search algorithms are needed, however, to establish global convergence of the algorithms. A method of Nash and Sofer [12] takes advantage of simultaneous function evaluations, one per processor. If the line search conditions are satisfied by the parameter $t_k = 1$, the full step $x_{k+1} = x_k + d_k$ is taken; otherwise, some parameter $t_k < 1$ is chosen.

This paper focuses on efficient algorithms for obtaining an approximate solution to the linear system $r_k = 0$, preserving superlinear convergence of the Newton algorithm.

One way to achieve efficiency in the solution of the linear system is to exploit structure in the problem by the use of multisplitting preconditioners. If we wish to solve the linear system $Az = b$, and the matrix can be partitioned in several ways,

$$A = M_l - N_l, \quad l = 1, ..., p, \tag{4}$$

then O'Leary and White [14] define a *multisplitting* of A to be

$$B = \sum_{l=1}^{p} D_l M_l^{-1} N_l \tag{5}$$

where the matrices D_l are nonnegative diagonal matrices that sum to the identity matrix. We denote by G the effective preconditioning matrix

$$G = \sum_{l=1}^{p} D_l M_l^{-1}. \tag{6}$$

A PARALLEL INEXACT NEWTON METHOD USING A KRYLOV MULTISPLITTING ALGORITHM

CHIOU-MING HUANG* AND DIANNE P. O'LEARY†

Abstract. We present a parallel variant of the inexact Newton algorithm that uses the Krylov multisplitting algorithm (KMS) to compute the approximate Newton direction. The algorithm can be used for solving unconstrained optimization problems or systems of nonlinear equations. The KMS algorithm is a more efficient parallel implementation of Krylov subspace methods (GMRES, Arnoldi, etc.) with multisplitting preconditioners. The work of the KMS algorithm is divided into the multisplitting tasks and a direction forming task. There is a great deal of parallelism within each task and the number of synchronization points between the tasks is greatly reduced. We study the local and global convergence properties of the algorithm and present results of numerical examples on a sequential computer.

Key words. conjugate gradient algorithm, preconditioning, GMRES, Krylov subspaces, multisplittings, inexact Newton method, solving nonlinear equations, unconstrained optimization.

AMS(MOS) subject classifications. 65H10, 65K10, 65F10.

July 30, 1993
Running title: KMS Multisplitting

1. Introduction. As we attempt to solve larger optimization problems and systems of nonlinear equations, it becomes more important to develop efficient ways to exploit sparsity and inherent parallelism. This paper concerns these issues. We consider two problems:

Problem 1: Nonlinear Equations:

$$g(x^*) = 0 \tag{1}$$

where $g : D \subseteq R^n \longrightarrow R^n$, D is an open convex set, and g is continuously differentiable on D. We denote $\nabla g(x)$ by $H(x)$.

Problem 2: Unconstrained Optimization:

$$\min_{x \in D} f(x) \tag{2}$$

where $f : D \subseteq R^n \longrightarrow R$, D is an open convex set, f is twice continuously differentiable and bounded below, and the second derivative $H(x) \equiv \nabla^2 f(x) \in R^{n \times n}$ is symmetric positive definite. We denote the gradient of f by $g(x)$, and note that any solution x^* to the problem satisfies $g(x^*) = 0$.

The only essential difference between the problems is that $H(x)$ is symmetric positive definite for Problem 2, but not for Problem 1. To unify the notation, we will denote the function to be minimized, either $f(x)$ or $\frac{1}{2}\|g(x)\|^2$, by $F(x)$.

Many algorithms have been proposed for these problems. These include Newton methods, the conjugate gradient algorithm, and the inexact Newton methods. The Newton search direction is defined to be the solution d to the linear system

$$H(x_k)d + g(x_k) = 0.$$

* Department of Computer Science, University of Maryland, College Park, MD 20742.
† Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. This work was supported under grant NSF CCR 9115568.