10/4-16-96 (1)

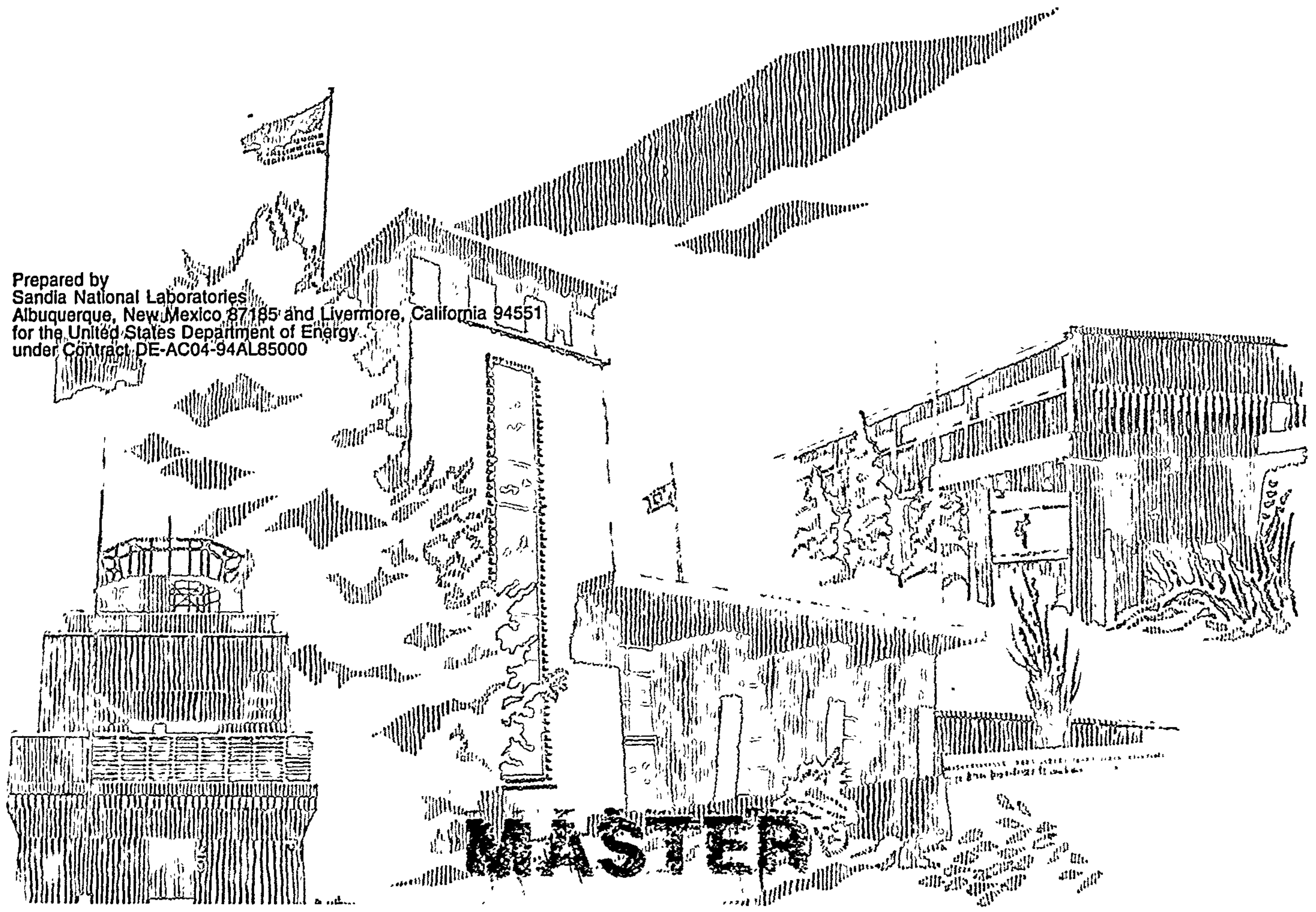
SANDIA REPORT

SAND96-8221 • UC-405
Unlimited Release
Printed March 1996

Preconditioning a Product of Matrices Arising in Trust Region Subproblems

MASTER

Mary E. Hribar and T. D. Plantenga



Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94551
for the United States Department of Energy
under Contract DE-AC04-94AL85000

MASTER

SF2900Q(8-81)

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED BS

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.
NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of the contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof or any of their contractors or subconractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof or any of their contractors or subcontractors.

This report has been reproduced from the best available copy.

Available to DOE and DOE contractors from:

Office of Scientific and Technical Information
P. O. Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from:

National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd.
Springfield, VA 22161

SAND96-8221
Unlimited Release
Printed March 1996

UC-405

Preconditioning a Product of Matrices Arising in Trust Region Subproblems

Mary E. Hribar and T.D. Plantenga
Scientific Computing Department
Sandia National Laboratories
Livermore, CA 94551-0969

ABSTRACT

In solving large scale optimization problems, we find it advantageous to use iterative methods to solve the sparse linear systems that arise. In the ETR software for solving equality constrained optimization problems, we use a conjugate gradient method to approximately solve the trust region subproblems. To speed up the convergence of the conjugate gradient routine, we need to precondition matrices of the form Z^TWZ, which are not explicitly stored. Four preconditioners were implemented and the results for each are given.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

1 Introduction

This paper focuses on solving certain large scale linear systems efficiently. Solving a system $Ax = b$ is a recurring problem in numerical methods. The specific linear system we will consider is one which arises in solving optimization problems.

We are interested in solving large optimization problems of the form:

$$\begin{aligned} \text{minimize} \quad & f(x) && (1.1)\\ \text{subject to} \quad & h(x) = 0 && (1.2)\\ & g(x) \leq 0, && (1.3) \end{aligned}$$

where $x \in \mathrm{R}^n$, $f : \mathrm{R}^n \rightarrow \mathrm{R}$, $h : \mathrm{R}^n \rightarrow \mathrm{R}^q$, and $g : \mathrm{R}^n \rightarrow \mathrm{R}^t$, and $f, h, g \in \mathrm{C}^1$. In these problems, n is large and each component function of $f(x)$, $h(x)$, and $g(x)$ may be a function of relatively few unknowns. Thus, the matrices of first and second derivatives are often sparse.

In many methods for solving (1.3), a linear system, which has the form $Z^TWZd = -g$, will have to be solved at each iteration. The vector g is the gradient of the Lagrangian function (defined in Section 1.1), the matrix W is an $n \times n$ matrix of exact or approximated second derivatives of the Lagrangian, and Z is a basis for the tangent subspace of the constraints. These Z and W matrices are large and the product Z^TWZ is often dense. Thus, an iterative method, such as conjugate gradients, which doesn't require the explicit storage of Z^TWZ, is usually used to solve the system.

Our approach to solving (1.3), involves solving a quadratic problem at each iteration. We are also using a trust region to limit our step size to guarantee global convergence. We solve the quadratic problem by decomposing it into two subproblems: a vertical suproblem and a horizontal subproblem. It is the horizontal subproblem that contains the linear system of interest. We use a conjugate gradient method to solve the linear system $Z^TWZd = g$, subject to a scaled trust region. The solution of this subproblem dominates the execution time of the code. We are interested in improving this routine; thus we are looking for a preconditioner for Z^TWZ.

We will briefly describe this horizontal subproblem and then return to the discussion of preconditioning conjugate gradients.

1.1 The Horizontal Subproblem Definition

We will consider the solution of the optimization problem with only equality constraints. We will solve problems with inequalities by transforming them to equality constrained problems. Solving (1.3) is equivalent to solving a series of equality constrained problems, with the addition of slack variables to the inequality constraints and a log barrier term to the objective function.

This equality constrained problem is

$$\begin{aligned} \text{minimize} \quad & \phi(x) && (1.4)\\ \text{subject to} \quad & c(x) = 0, && (1.5) \end{aligned}$$

where $\phi : \mathrm{R}^n \rightarrow \mathrm{R}$ and $c : \mathrm{R}^n \rightarrow \mathrm{R}^m$. The Lagrangian for this problem is

$$\mathcal{L}(x, \lambda) = \phi(x) - \lambda^T c(x).$$

Our approach to solving the optimization problem is to solve a sequence of quadratic problems, with a trust region for global convergence. Each problem involves minimizing a quadratic approximation of the Lagrangian subject to a linearization of the constraints. There is also a trust region constraint which limits the step to be within a region where we trust our quadratic model to be accurate. The quadratic problem is:

$$\begin{aligned} \text{minimize} \quad & \nabla\phi(x_k)^T d + \frac{1}{2} d^T W(x_k, \lambda_k) d && (1.6) \\ \text{subject to} \quad & A(x_k)^T d + c(x_k) = 0 && (1.7) \\ & \|d\| \leq \Delta_k. && (1.8) \end{aligned}$$

where $A(x) = [\nabla c_1(x), \ldots, \nabla c_m(x)]$ contains the gradients of the constraints, and $W(x, \lambda) = \nabla^2_{xx}\mathcal{L}(x, \lambda)$ contains the second derivatives of the Lagrangian, or some approximation to them.

Unfortunately, the problem in (1.6) - (1.8) may not have any feasible solutions. So, our approach, which is Byrd and Omojokun's method [2, 7], decomposes the problem into two subproblems, the vertical subproblem and the horizontal subproblem, each of which is guaranteed to have a solution.

The vertical subproblem finds the point closest to satisfying the linearization of the constraints, while also staying within a fraction of the trust region radius. The vertical subproblem is:

$$\begin{aligned} \text{minimize} \quad & \|A_k^T v + c_k\| && (1.9) \\ \text{subject to} \quad & \|v\| \leq \zeta\Delta_k, && (1.10) \end{aligned}$$

where $A_k = A(x_k)$ and $c_k = c(x_k)$. The relaxation parameter ζ may have any value in the interval $(0, 1)$.

The horizontal subproblem finds the full step d which minimizes the objective function (1.6), while not moving any closer to the feasible manifold than the solution to the vertical subproblem, v_k, does. The step d must also lie within the trust region radius. The formulation of the horizontal subproblem is:

$$\begin{aligned} \text{minimize} \quad & \nabla\phi(x_k)^T d + \frac{1}{2} d^T W(x_k) d && (1.11) \\ \text{subject to} \quad & A_k^T d = A_k^T v_k && (1.12) \\ & \|d\| \leq \Delta_k. && (1.13) \end{aligned}$$

Once the step d is found by solving the two subproblems, the remaining algorithm is a standard trust region method. If the step provides good decrease, we accept the step and increase the trust region. Otherwise, we reject the step and decrease the trust region.

The ETR software, (Equality constrained optimization using Trust Regions), is an implementation of Byrd and Omojokun's method by Lalee, Nocedal, and Plantenga [5, 8]. The code was written to solve large scale problems, so the subproblems are only approximately solved while maintaining reasonable storage requirements. Computing the horizontal step dominates the execution time for many problems. We are, thus, currently interested in improving the efficiency of the routines which solve the horizontal problem.

We can eliminate the equality constraints (1.12) in the horizontal problem by making a substitution for d. If we let Z_k be a basis for the nullspace of A_k, we can define the step to be the sum of two orthogonal components: $d = v_k + Z_k u$. The solution to the vertical problem, v_k is in the range space of A_k. We form a $n \times (n-m)$ matrix Z_k which is a basis for the nullspace of A_k. Now, the subproblem (1.13) can be rewritten as:

$$\begin{aligned} &\text{minimize} && \bar{g}_k^T Z_k u + \frac{1}{2} u^T Z_k^T W_k Z_k u && (1.14) \\ &\text{subject to} && \|Z_k u\| \leq \bar{\Delta}_k, && (1.15) \end{aligned}$$

where $W_k = W(x_k)$, $\bar{g}_k = (\nabla\phi(x_k) + W_k v_k)$, and $\bar{\Delta}_k = \sqrt{\Delta_k^2 - \|v_k\|^2}$. This problem is now in the form of a standard trust region problem: minimizing a quadratic objective subject to the trust region constraint.

There are a couple of options for computing Z_k, so that $A_k^T Z_k = 0$. One is to use the QR factorization of A_k. However, this factorization may become dense, even if A_k is sparse. So, we use the direct elimination of A_k instead. We partition A_k^T into

$$A_k^T = [B_k \;\; N_k] \tag{1.16}$$

and then define Z_k to be

$$Z_k = \begin{bmatrix} -B_k^{-1} N_k \\ I \end{bmatrix}. \tag{1.17}$$

To preserve sparsity, we only store the LU factors of B_k. So, Z_k is not explicitly stored, but the products $Z_k u$ and $Z_k^T x$ can be computed.

We will use Steihaug's method [9], which is a modification to the conjugate gradient method to solve trust region problems, to solve the horizontal problem (1.14) - (1.15).

2 Solution of Horizontal Problem

Since the reduced Hessian, $Z_k^T W_k Z_k$, is not explicitly stored, we need to use an iterative method to approximately solve the horizontal problem. However, we need to allow for the reduced Hessian to be indefinite and for the trust region constraint. Steihaug's method uses the conjugate gradient iteration, but has three termination tests, instead of one [9]. If the minimizer of (1.14) lies within the trust region, the method will terminate at this answer. However, the method will also terminate when an iterate violates the trust region bound or when a direction of negative curvature is detected. In either of these cases the step is truncated to the length of the trust region.

Steihaug's method gives only an approximate solution to the trust region problem. It uses the steepest descent direction as the first direction. The first iterate is the point where the quadratic objective function is minimized along the steepest descent direction, subject to the trust region constraint, which is the Cauchy point. All of the iterates after this point will decrease the objective function. Since the solution found by Steihaug's method is as much as a fraction of the decrease from the Cauchy point, we are guaranteed global convergence of the entire algorithm.

In addition, this method will take Newton steps, if the trust region is big enough, guaranteeing local quadratic convergence.

The trust region constraint in (1.15) is scaled. Figure 1 contains the details of Steihaug's method to solve this scaled trust region problem. We will refer to this algorithm as the correct Steihaug.

```
Given $\epsilon_0$, $\epsilon > 0$
$(Zu)_0 = 0$; $r_0 = -Z_k^T \bar{g}_k$; $t_0 = (Z_k^T Z_k)^{-1} r_0$; $(Zp)_0 = Z_k t_0$; $j = 0$;
If $\sqrt{r_0^T t_0} < \epsilon_0$
    Then return the step $(Zu) = 0$
Loop
    If $(Zp)_j^T W_k (Zp)_j \leq 0$
        Then find $\tau$ such that $(Zu) = (Zu)_j + \tau (Zp)_j$ minimizes (1.14),
        $\|(Zu)\|_2 = \bar{\Delta}_k$. Return $(Zu)$.
    $\alpha_j = r_j^T t_j / (Zp)_j^T W_k (Zp)_j$
    $(Zu)_{j+1} = (Zu)_j + \alpha_j (Zp)_j$
    If $\|(Zu)_{j+1}\|_2 \geq \bar{\Delta}_k$
        Then find $\tau > 0$ such that $(Zu) = (Zu)_j + \tau (Zp)_j$,
        $\|(Zu)\|_2 = \bar{\Delta}_k$. Return $(Zu)$.
    $r_{j+1} = r_j - \alpha_j Z_k^T W_k (Zp)_j$
    $t_{j+1} = (Z_k^T Z_k)^{-1} r_{j+1}$
    If $\sqrt{r_{j+1}^T t_{j+1}} < \epsilon \sqrt{r_0^T t_0}$ Then return $(Zu)_{j+1}$
    $\beta_{j+1} = r_{j+1}^T t_{j+1} / r_j^T t_j$
    $(Zp)_{j+1} = Z_k t_{j+1} + \beta_{j+1} (Zp)_j$
    $j = j + 1$
Continue
```

Figure 1: The correct Steihaug routine to solve horizontal problem.

At each iteration, the solution of the linear system $Z_k^T Z_k t_{j+1} = r_{j+1}$ is computed approximately, using conjugate gradients. Solving this linear system can become very costly. If we eliminate solving this system at each iteration, we are then solving the trust region problem with an unscaled trust region constraint:

$$\text{minimize} \quad \bar{g}_k^T Z_k u + \frac{1}{2} u^T Z_k^T W_k Z_k u \tag{2.18}$$

$$\text{subject to} \quad \|u\| \leq \bar{\Delta}_k, \tag{2.19}$$

When the solution of (2.19) lies within the trust region and satisfies $\|Zu\| \leq \bar{\Delta}_k$, that solution also solves the scaled horizontal problem (1.14)- (1.15). So, if we suspect that we will be taking a Newton step, we can eliminate the scaling matrix $(Z_k^T Z_k)^{-1}$ from the algorithm. (We call this unscaled Steihaug the deconditioned Steihaug.) We don't want to eliminate the scaling altogether since we need to use the accurately scaled Cauchy step as the solution when the trust region gets small.

This scaling matrix $(Z_k^T Z_k)^{-1}$ acts also like a preconditioner. So, using the correct Steihaug will often converge in less iterations than the deconditioned Steihaug. However, the cost is great. So, the first improvement we made to the horizontal problem was to use the deconditioned Steihaug all of the time. Only when the initial direction lies outside of the trust region do we use the correct Steihaug routine, so that the correctly scaled Cauchy step will be taken when the trust region is small enough. This proves to be robust and to reduce the execution time to compute the horizontal step.

The following figures show the results of this change on ten test problems, which are taken from the CUTE (Constrained and Unconstrained Testing Environment) [1]. A table giving a brief description of the problems is given in Appendix A. Figure 2 shows a graph of the percent change in the time to solve the horizontal subproblem, the number of conjugate gradient iterations, and the total number of function evaluations when using the deconditioned Steihaug all of the time instead of the correct Steihaug. For almost all of the problems, it took half the time to solve the horizontal step even though for several of the problems, the number of conjugate gradient (CG) steps increased. Without the scaling, often more conjugate gradient iterations are needed to solve the trust region problem. since these iterations are cheap, the total time to find a solution still decreased. Also, the approximate solutions found by the deconditioned routine are not as good as those found by the scaled routine for convergence of the entire algorithm. So, more outer iterations of ETR are needed and the number of function evaluations increases, which is also demonstrated in Figure 2. The only problem in which the time to compute the horizontal step did not decrease was the OM problem. In this problem, using either the deconditioned routine or the scaled routine resulted in only Cauchy steps. So, the deconditioned routine added the overhead of trying the deconditioned step first, doubling the number of conjugate gradient iterations.

Figure 3 shows the percent change in the total time for the entire algorithm to terminate when the deconditioned routine is always used instead of the scaled routine. For most of the problems, the time decreased. As expected, the OM problem took slightly more time. The R1 problem, for which the time to solve the horizontal problem decreased, overall took longer to solve. The added conjugate gradient iterations and function evaluations, for this problem, proved to be costly. Overall, this change to using the cheaper deconditioned routine all of the time, served to decrease the running time of the algorithm by about a third for most of the problems.

Now, we would like to decrease the number of deconditioned Steihaug steps. So, we will attempt to precondition $Z_k^T W_k Z_k$.

3 Preconditioners for $Z_k^T W_k Z_k$

Using the conjugate gradient method to solve the system $Ax = b$ works well when A is well conditioned or has few distinct eigenvalues. So, we would like to use conjugate gradient routine on a preconditioned system $\tilde{A}\tilde{x} = \tilde{b}$, where $\tilde{A} = C^{-\frac{1}{2}} A C^{-\frac{1}{2}}$, $\tilde{x} = C^{\frac{1}{2}} x$ and $\tilde{b} = C^{-\frac{1}{2}} b$. We want

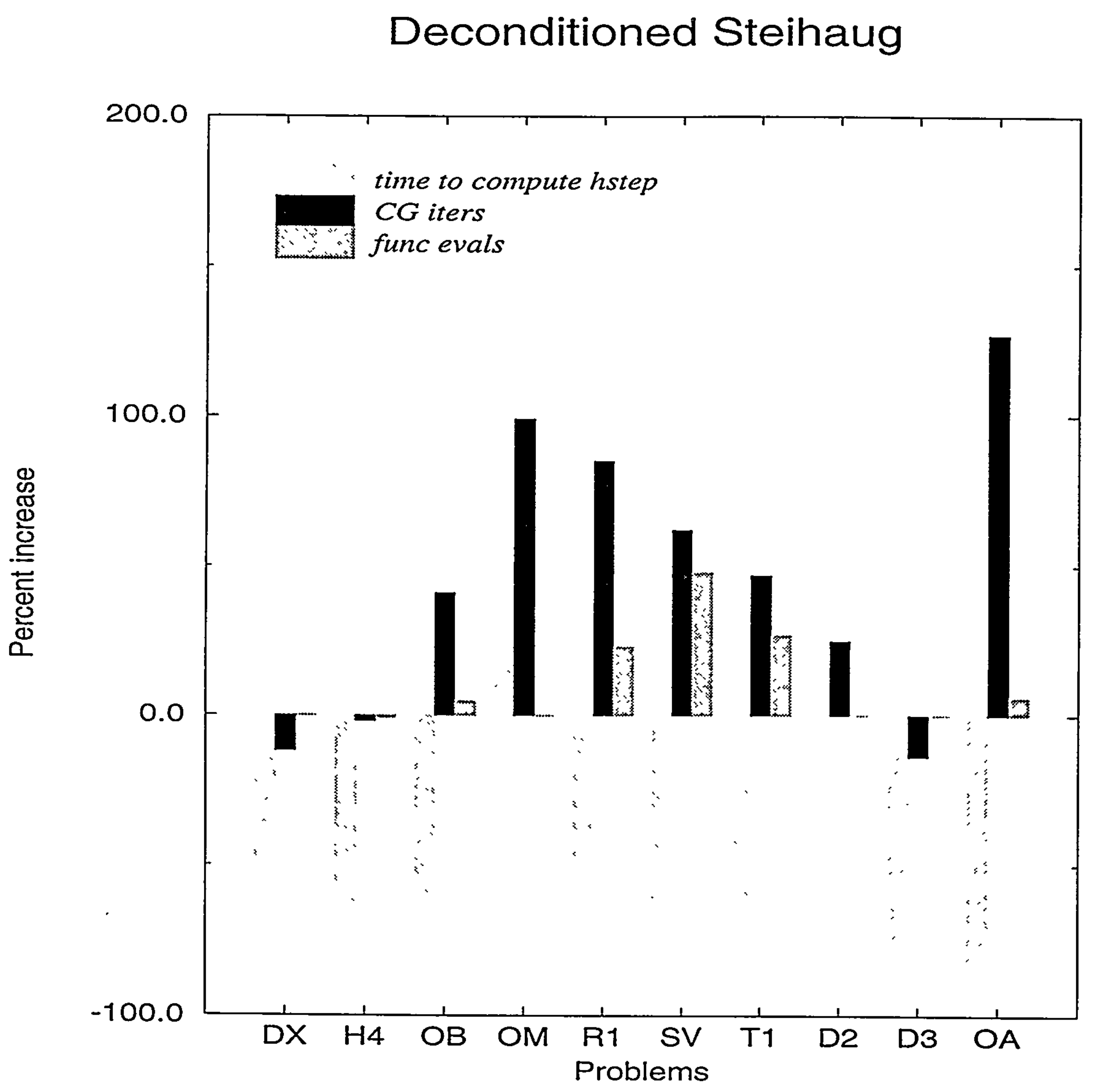


Figure 2: Plot of the percent change in time to compute horizontal step, number of CG iterations, and total number of function evaluations when using the deconditioned Steihaug at each inner iteration instead of correct Steihaug

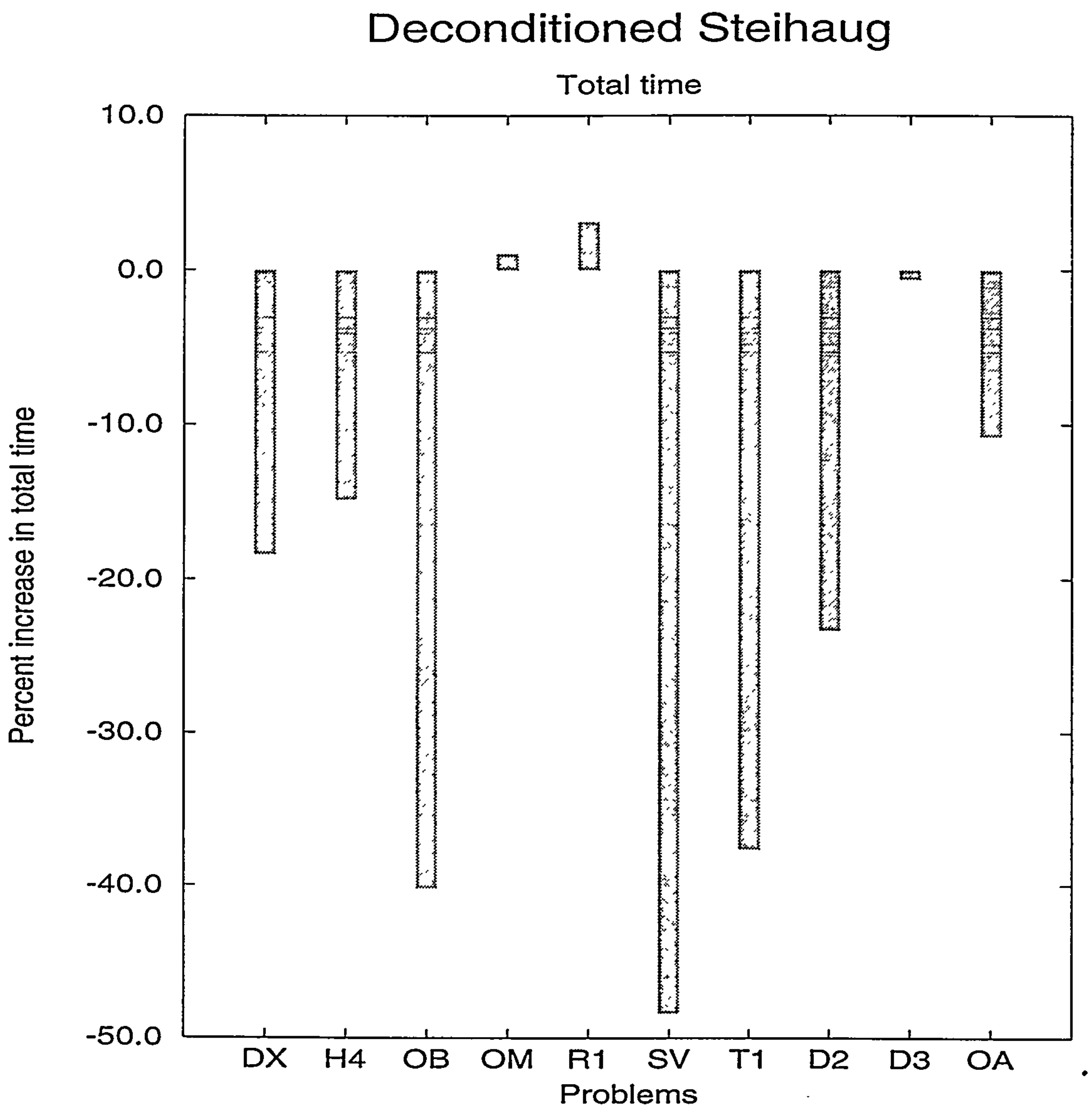


Figure 3: Plot of the percent change in total time when using the deconditioned Steihaug at each inner iteration instead of correct Steihaug

to use a preconditioner $C \approx A$ so that $\tilde{A}$ is close to the identity matrix; thus ensuring rapid convergence. The preconditioner C must be positive definite and easily invertible.

The matrix which we are preconditioning is $Z_k^T W_k Z_k$, which is not explicitly stored. The matrix W_k may be stored, but, as mentioned earlier, Z_k is not. Finding a preconditioner $C \approx Z_k^T W_k Z_k$ is difficult. For instance, neither the diagonal of the matrix nor approximate factorizations are readibly available, which are two common preconditioners. Others have suggested preconditioners for this problem. In both [6, 3], the authors suggest preconditioners based on the following formula:

$$(Z_k^T W_k Z_k)^{-1} = Y_k^T W_k^{-1} Y_k - Y_k^T W_k^{-1} A_k^T (A_k W_k^{-1} A_k^T)^{-1} A_k W_k^{-1} Y_k, \tag{3.20}$$

where Y_k is a left inverse for Z_k. In [6], Nash and Sofer suggest approximating (3.20) with a power series. In [3], Conn, Gould and Toint suggest using (3.20) with a symmetric approximation, $\bar{W}_k \approx W_k$, which is easily invertible. In both of these approaches, the preconditioner can be very expensive to use. The first approach requires eigenvalue information from relevant matrices, and the latter requires the computation of $(A_k \bar{W}_k A_k^T)^{-1}$.

We implemented four different preconditioners. Three of them had similar results, so we will discuss these first. The first is the exact diagonal of the matrix $Z_k^T W_k Z_k$. To form this preconditioner, the algorithm must do $n - m$ multiplies by Z_k, which will prove to be a very costly operation to execute for each horizontal problem we solve. The second preconditioner is an estimate of the diagonal. This preconditioner is $C = \text{diag}(\tilde{Z}_k^T W_k \tilde{Z}_k)$, where

$$\tilde{Z}_k = \begin{bmatrix} -(\text{diag}(B_k))^{-1} N_k \\ I \end{bmatrix}$$

For both of these preconditioners, the diagonal elements are forced to be positive by using the absolute value of the actual diagonal entries. The third preconditioner is the tridiagonal of $Z_k^T W_k Z_k$. This preconditioner is made to be positive definite by modifying the diagonal elements so that they are a small fraction greater than the sum of the off-diagonal elements. The tridiagonal preconditioner also requires $n - m$ Z_k multiplies.

The following graphs show the change in the performance of code when the three preconditioners were used in the deconditioned Steihaug routine. The bars represent the percent increase in iterations, function evaluations, or time when each of the preconditioners was used compared with the deconditioned Steihaug with no preconditioner. (The negative bars represent decrease.) Figure 4 shows the percentage of change in the number of conjugate gradient iterations. For three of the problems, all of the preconditioners caused a decrease in the number of CG iterations. For three more problems, using the exact diagonal and the tridiagonal preconditioners resulted in fewer conjugate gradient steps. The estimated diagonal was not as effective in reducing the conjugate gradient steps. For problem D2, the algorithm did not converge when using the estimated diagonal preconditioner. Figure 5, shows the percentage change in the number of function evaluations when using the three preconditioners. A few problems have relatively small changes in the number of evaluations. Most have no change. Only one problem, OA, demonstrates a significant increase in function evaluations.

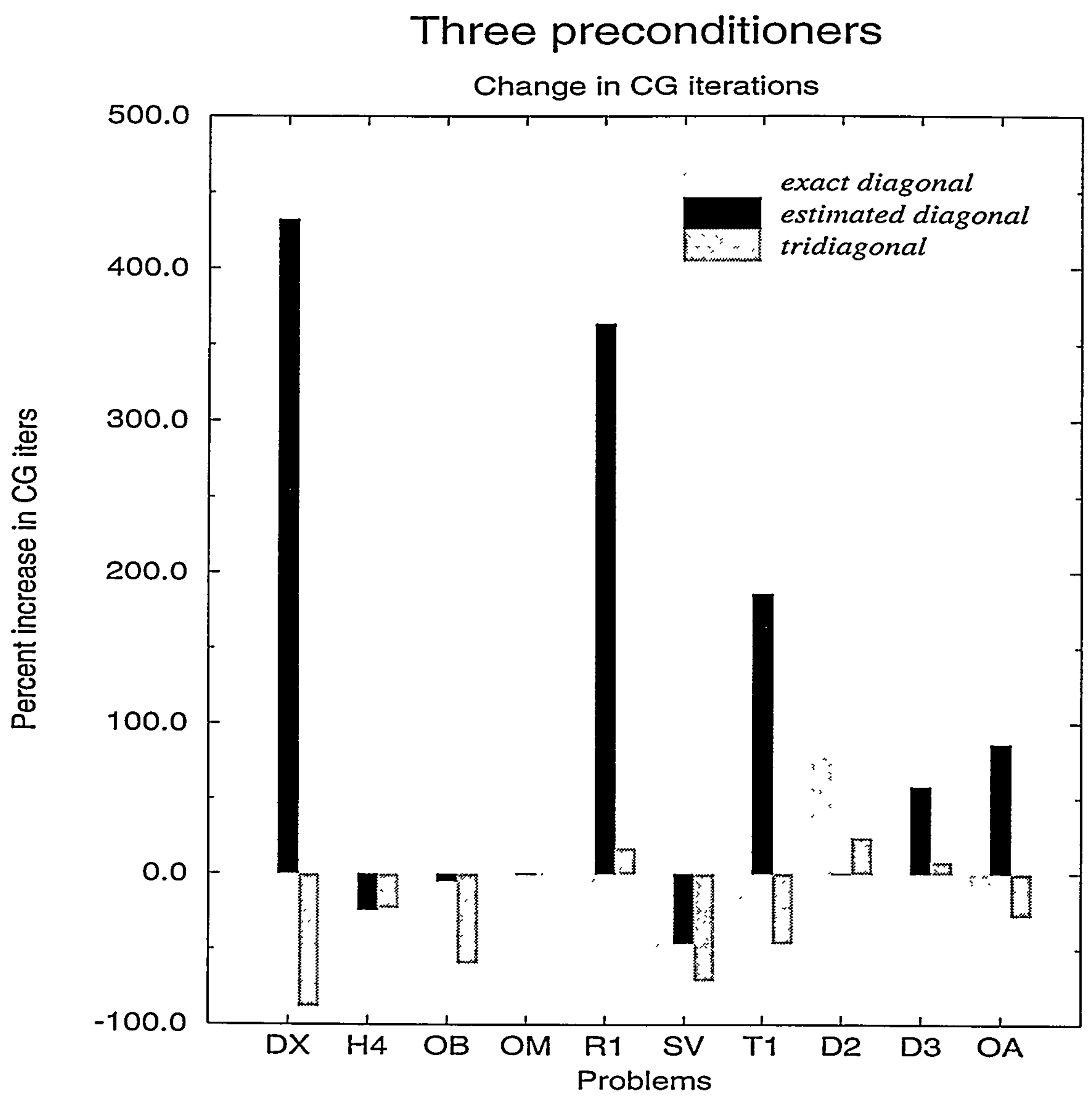


Figure 4: Plot of the percent change in number of CG iterations when using the three preconditioners

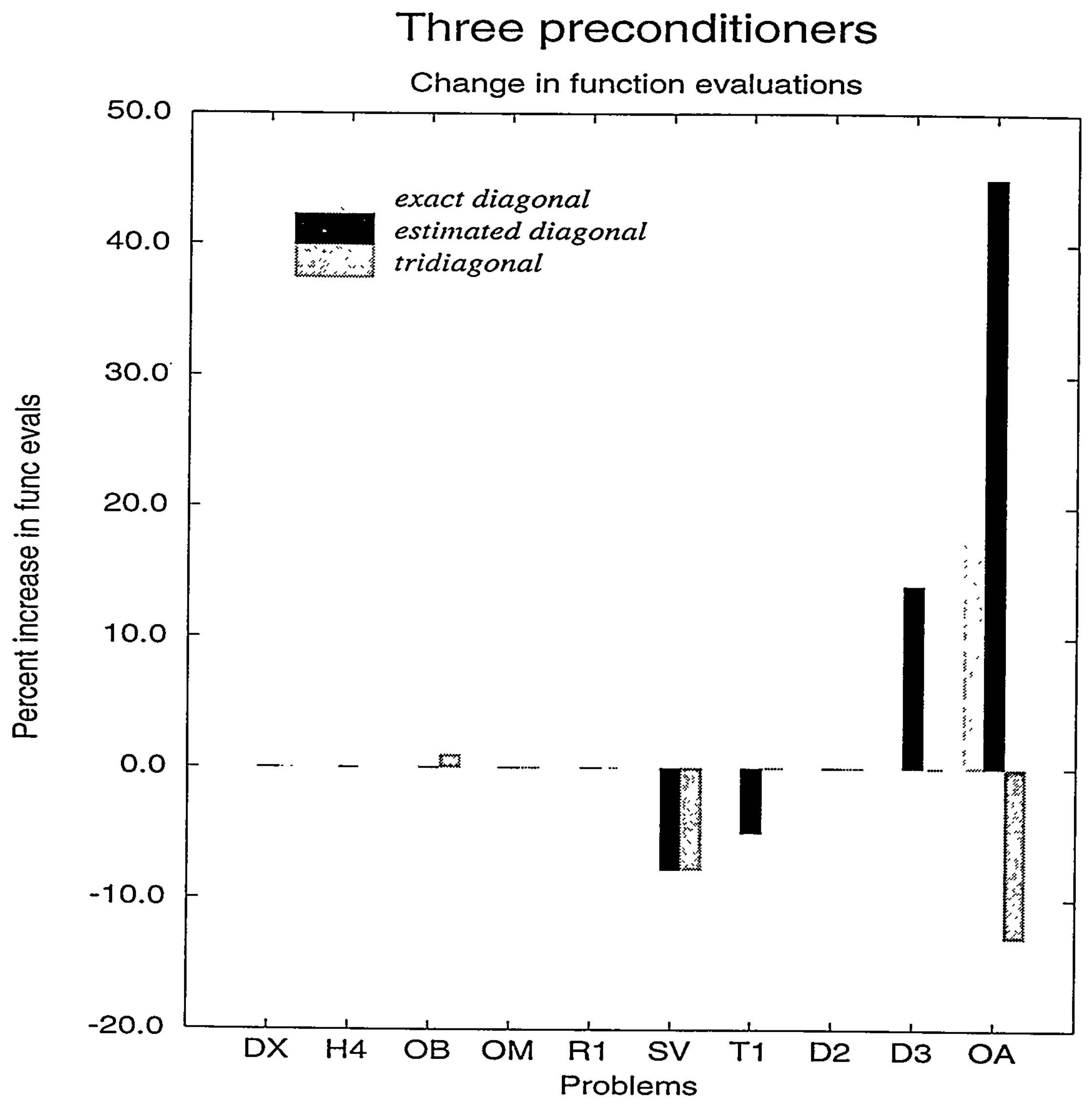


Figure 5: Plot of the percent change in function evaluations when using the three preconditioners

Even though the number of conjugate gradient steps decreased and the number of function evaluations didn't increase significantly, these preconditioners still proved to be ineffective. As shown in figure 6, the time to compute the horizontal step increased dramatically because the expense of computing the preconditioners was too costly. The two problems, D3 and OA which are missing from this graph, are even more expensive. Also, when using the preconditioners, the more expensive scaled routine is used more often. Often, in the first iteration of Steihaug's method the initial preconditioned direction will violate the trust region constraint when the unpreconditioned direction would not. Finally, figure 7 shows the increase in the total time of the algorithm.

These three preconditioners proved to be too costly to use for each horizontal problem. Thus we tried using them only when a Newton step was expected. As a result, only a few horizontal problems were preconditioned and any improvements in performance were negligible.

4 Lanczós preconditioner

Another idea for a preconditioner is to estimate the reduced Hessian matrix in iteration k by an estimate of the reduced Hessian in iteration $k-1$. If the steps in the algorithm are small, we don't expect $Z_k^T W_k Z_k$ to change drastically from iteration to iteration. By storing the α, β, and residual vectors generated in the congugate gradient routine in the previous iteration, we can form a preconditioner $C \approx Z_{k-1}^T W_{k-1} Z_{k-1}$, which requires only a tridiagonal solver to solve $Ct = r$.

Using conjugate gradients to solve the system $Ax = b$ has a very close connection to the Lanczos method, which estimates the extremal eigenvalues of A by generating a sequence of tridiagonal matrices T_j [4]. The method involves iteratively generating orthonormal directions q_j such that $T_j \approx Q_j^T A Q_j$, where $Q_j = [q_1 \dots q_j]$ and

$$T_j = \begin{bmatrix} \gamma_1 & \delta_1 & & \cdots & 0 \\ \delta_1 & \gamma_2 & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \delta_{j-1} \\ 0 & \cdots & & \delta_{j-1} & \gamma_j \end{bmatrix}.$$

The conjugate gradient method can also generate Q_j and T_j. In the conjugate gradient routine, from the equations $p_j = r_{j-1} + \beta_j p_{j-1}$, we can set $R_j = P_j B_j$, where $R_j = [r_0, \dots, r_{j-1}]$, $P_j = [p_1, \dots p_j]$, and

$$B_j = \begin{bmatrix} 1 & -\beta_2 & & \cdots & 0 \\ & 1 & -\beta_3 & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & -\beta_j \\ 0 & \cdots & & & 1 \end{bmatrix}.$$

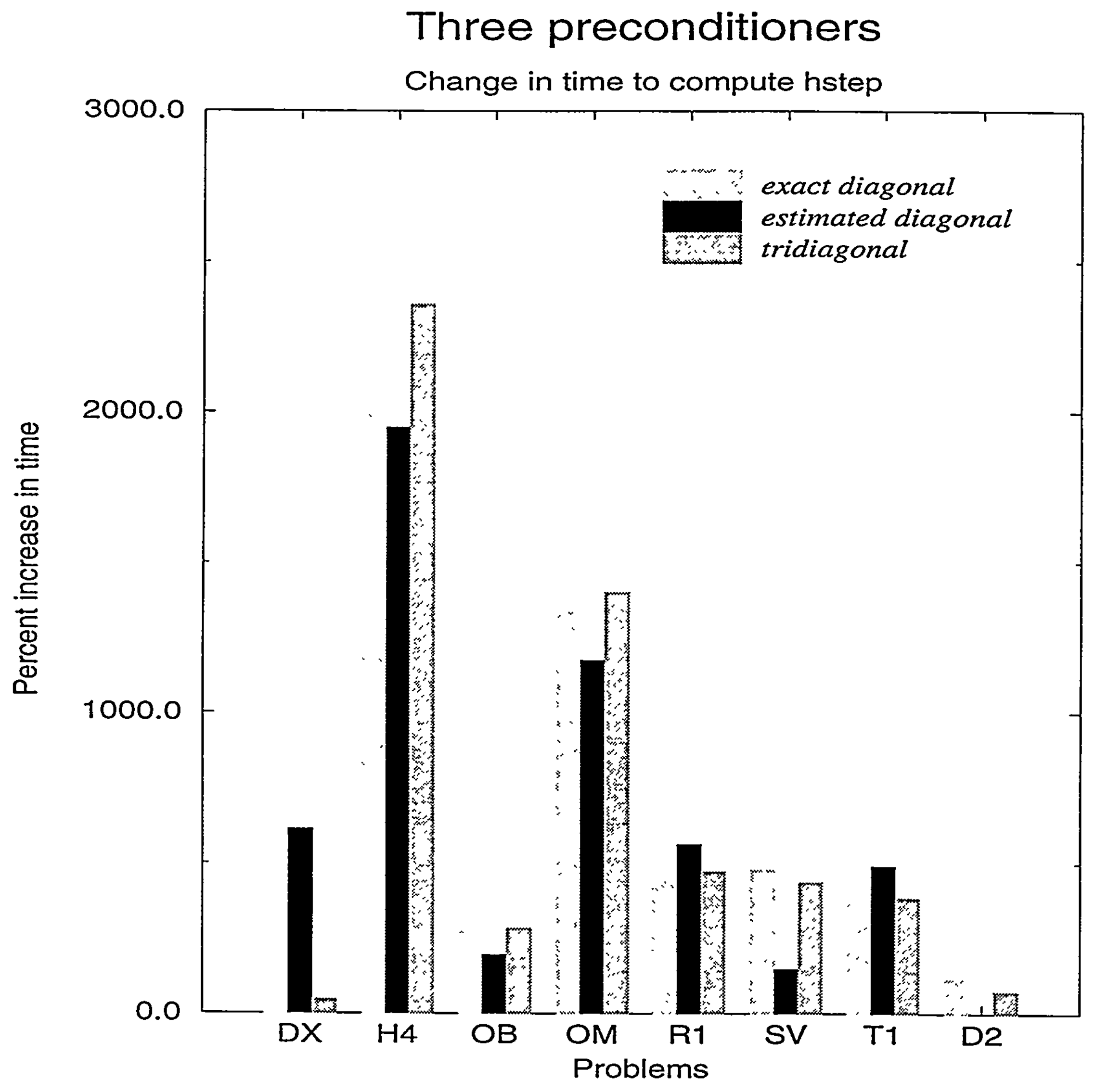


Figure 6: Plot of the percent change in time to compute horizontal step when using the three preconditioners

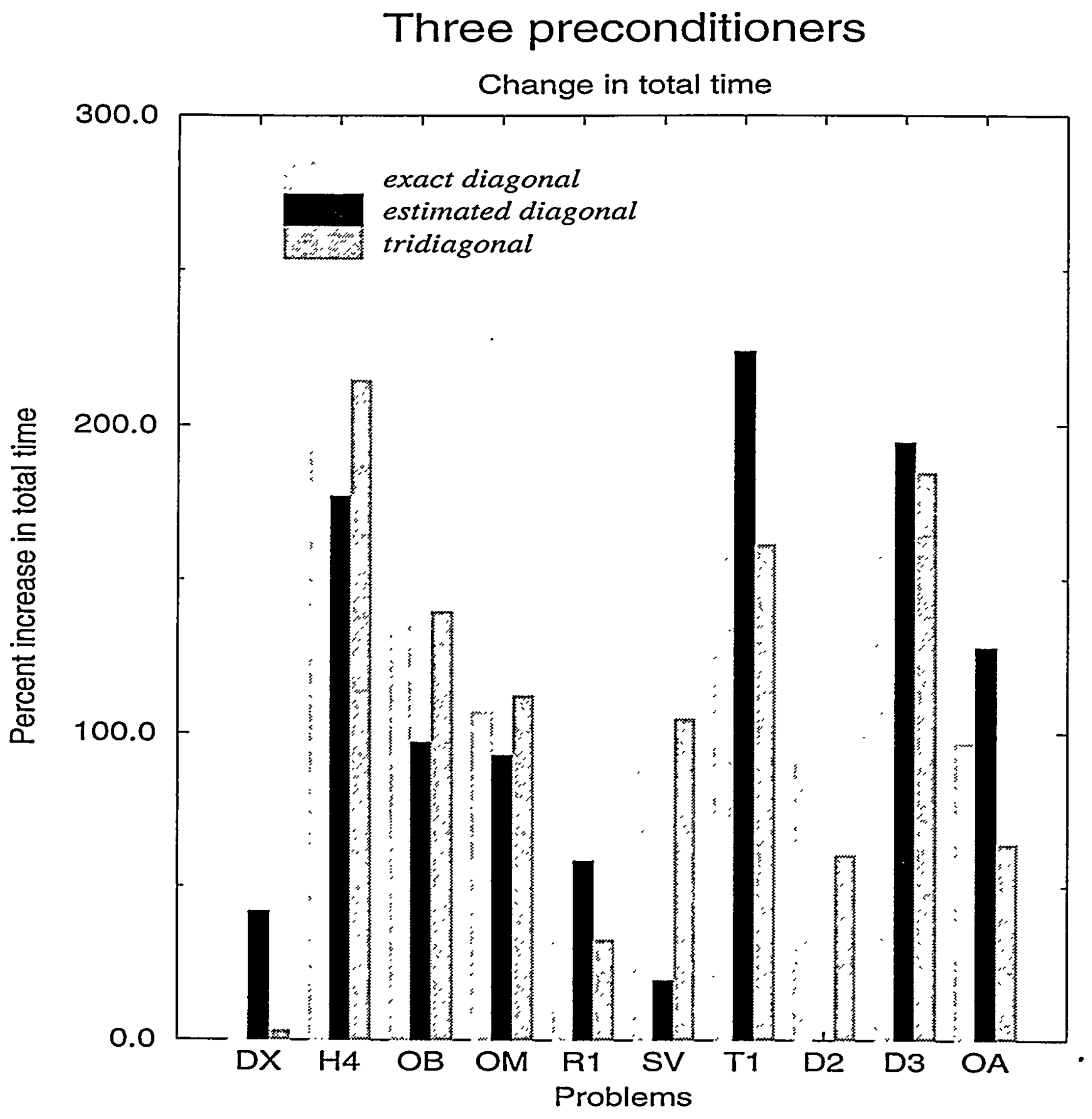


Figure 7: Plot of the percent change in total time when using the three preconditioners

Since $p_1, \ldots, p_j$ are A conjugate, the product $R_j^T A R_j$ is tridiagonal. From the theory of conjugate gradients, we know that the set of r_j vectors are orthogonal. Let

$$\tilde{q}_i = \pm r_{i-1}/||r_{i-1}||_2,$$

and

$$\tilde{Q}_j = [\tilde{q}_1, \ldots, \tilde{q}_j].$$

The columns of $\tilde{Q}_j$ are orthonormal. Then

$$\tilde{T}_j = \tilde{Q}_j^T A \tilde{Q}_j = \begin{bmatrix} \frac{1}{\alpha_1} & -\frac{\sqrt{\beta_2}}{\alpha_1} & & \cdots & 0 \\ -\frac{\sqrt{\beta_2}}{\alpha_1} & -\frac{\beta_2}{\alpha_1} + \frac{1}{\alpha 2} & -\frac{\sqrt{\beta_3}}{\alpha_2} & & \vdots \\ & -\frac{\sqrt{\beta_3}}{\alpha_2} & -\frac{\beta_3}{\alpha_2} + \frac{1}{\alpha 3} & \ddots & \vdots \\ & & \ddots & & \\ \vdots & & \ddots & \ddots & -\frac{\sqrt{\beta_j}}{\alpha_{j-1}} \\ 0 & \cdots & \cdots & -\frac{\sqrt{\beta_j}}{\alpha_{j-1}} & -\frac{\beta_j}{\alpha_{j-1}} + \frac{1}{\alpha_j} \end{bmatrix}.$$

Using these equations, we can use $C = \tilde{Q}_j \tilde{T}_j \tilde{Q}_j^T$ as the preconditioner for $Z_k^T W_k Z_k$, where $\tilde{Q}_j$ and $\tilde{T}_j$ are computed in iteration $k-1$. To solve the linear system $Ct = r$, we use a tridiagonal solver to find $\tilde{T}_j w = \tilde{Q}_j^T r$ and then assign $t = \tilde{Q}_j w$.

This preconditioner is cheap to store and to invert. However, there are difficulties involved in using it. In preconditioned CG iterations, the residual vectors are no longer orthogonal in the unscaled l_2 norm. So, only during unpreconditioned CG steps are the vectors and scalars stored to form the preconditioner. This preconditioner is then reused to precondition a fixed number of horizontal problems, before the preconditioning is turned off and another preconditioner is stored. So, the preconditioner may not accurately estimate $Z_k^T W_k Z_k$. Also, the preconditioner is positive semidefinite, which may cause the Steihaug method to halt after only a few iterations. If the residual vector r lies in the nullspace of $\tilde{Q}_j \tilde{T}_j \tilde{Q}_j^T$, then t will be zero, and the Steihaug method will end.

Figures 8 and 9 are graphs of the results of using this preconditioner in the deconditioned Steihaug routine, building the preconditioner from 20 iterations and reusing it for five horizontal problems. They show the percentage of increase in time, iterations, and function èvaluations when using the Lanczos preconditioner instead of the deconditioned Steihaug with no preconditioner. Figure 8 shows the change in time to compute the horizontal step, number of conjugate gradient steps and the number of function evaluations. For the few problems where the number of conjugate gradient iterations decreased, the number of function evaluations increased. For OB, even though the number of CG steps decreased, since there were more function calls, the running time increased. For four of the problems, there were significant increases in the number of function evaluations. As mentioned earlier, since the preconditioner is not positive definite, the CG routine will terminate early, with a bad step. Figure 9 shows the increase in total time from the slow convergence. Even problems which had a decrease in time to solve the horizontal problem took longer to converge and the total time increased. And, for H4, using this preconditioner prevented convergence completely.

Although inexpensive to use, the Lanczos preconditioner is not a viable option for a preconditioner.

5 Conclusion

In solving optimization problems, using the trust region method, a linear system arises of the form $Z^T W Z d = -g$. When this matrix is not stored, it is difficult to precondition. Computing the diagonal and tridiagonal preconditioners for each horizontal subproblem is expensive. The Lanczos preconditioner is cheap, but not stable to use. None of the preconditioners decreased the number of CG iterations significantly.

Thus, preconditioners for this problem must be very cheap to implement, and they must approximate $Z_k^T W_k Z_k$ well. Finding such a preconditioner may be too difficult; therefore, we are looking at other ways to improve the horizontal subproblem. One idea is to use the correct Steihaug method, but with a preconditioned routine for solving linear systems involving the matrix $Z_k^T Z_k$. Another idea is to rewrite the correct Steihaug method so that we can express the scaling $(Z_k^T Z_k)^{-1}$ as a projection. Thus, we have $Zt_j = Z_k(Z_k^T Z_k)^{-1} Z_k^T r_j$, and we can then substitute the projection $(I - A_k(A_k^T A_k)^{-1} A_k^T)$. This projection is cheaper to use since A_k is explicitly stored, and should reduce the time to compute the horizontal step.

6 Acknowledgment

The authors thank Juan Meza for his input to this project and for his comments on the report.

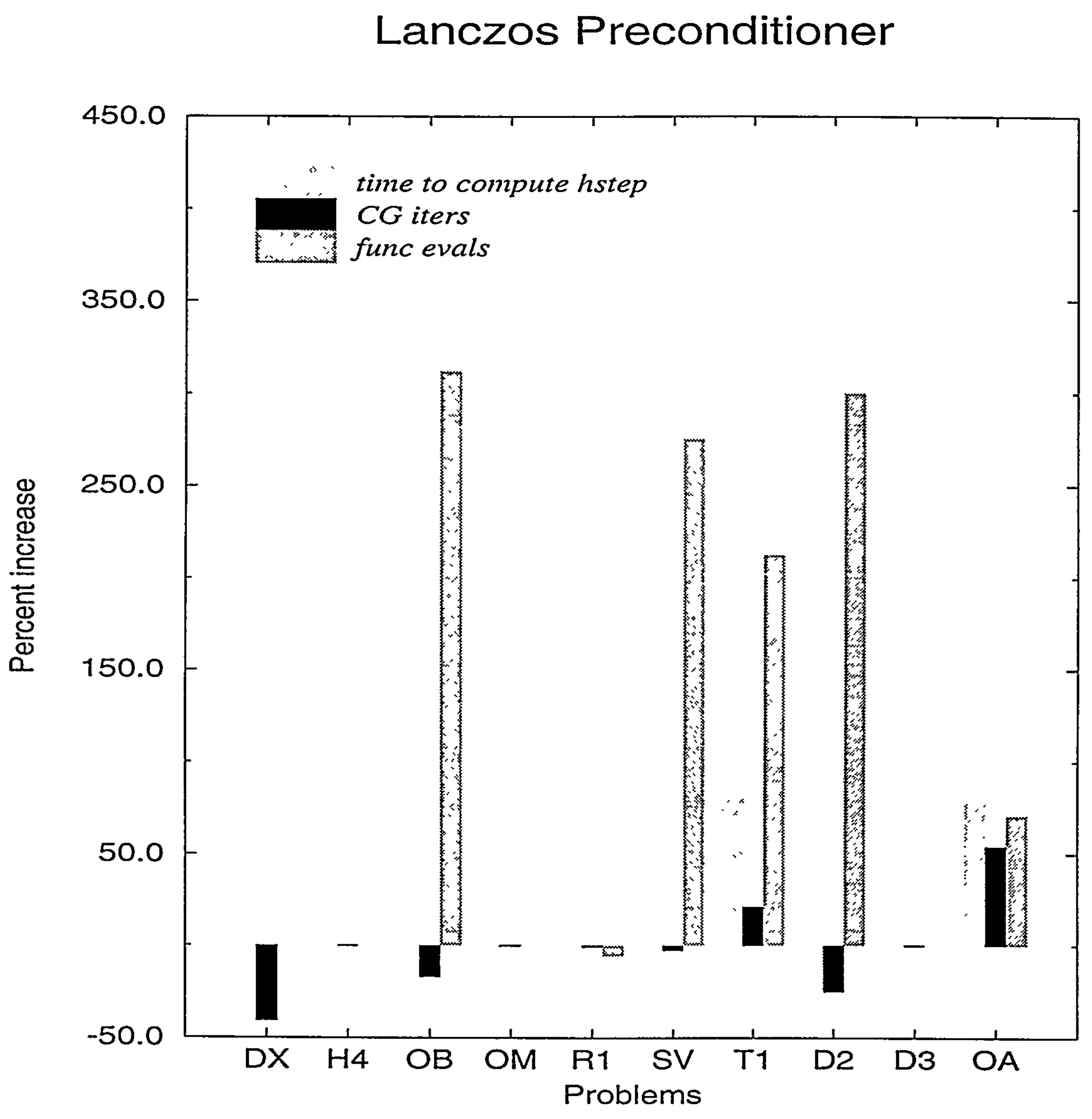


Figure 8: Plot of the percent change in time to solve horizontal problem, number of CG iterations and function evaluations when using the Lanczos preconditioner.

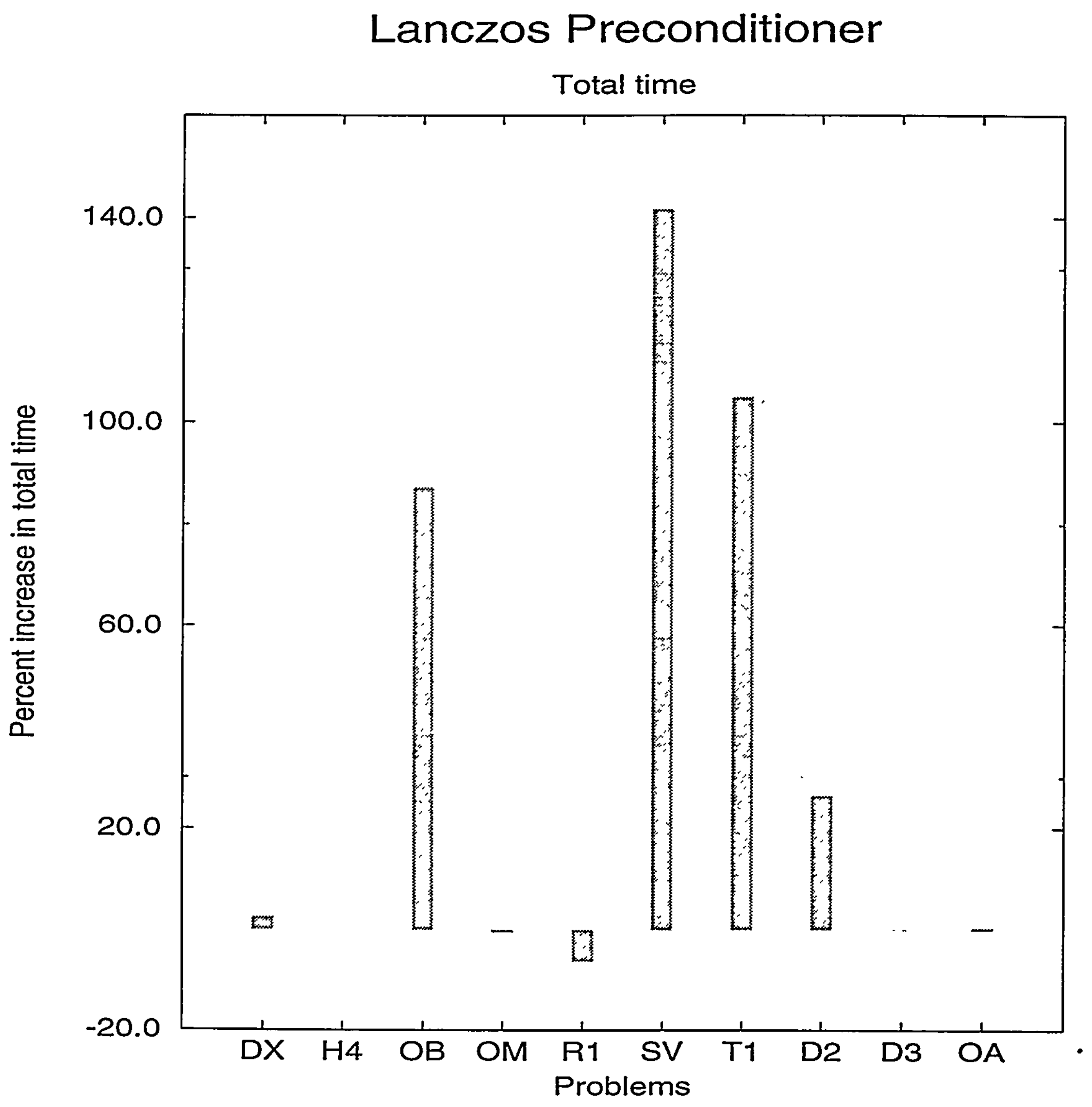


Figure 9: Plot of the percent change in total time when using the Lanczos preconditioner.

Appendix A

	Name	n	m	Constraints	Variables
DX	DIXCHLNV	100	50	nonlinear equality	bounds
H4	HAGER4	1001	500	linear equality	free, bounds, fixed
OB	OBSTCLAE	529	0		bounds, fixed
OM	OPTMASS	610	505	linear equality, nonlinear inequality	free, fixed
R1	READING1	202	100	nonlinear equality	bounds, fixed
SV	SVANBERG	500	500	nonlinear inequality	bounds
T1	TORSION1	484	0		bounds, fixed
D2	DTOC2	2998	1996	nonlinear equality	free, fixed
D3	DTOC3	14999	9998	linear equality	free, fixed
OA	ORTHREGA	2053	1024	nonlinear equality	free

Figure 10: Set of test problems from CUTE collection

References

[1] I. Bongartz, A.R. Conn, N. Gould, and Ph. L. Toint. CUTE: Constrained and unconstrained testing environment. Technical Report 93/10, Facultès Universitaires de Namur, 1993.

[2] R. Byrd. Robust trust region methods for constrained optimization. In *Third SIAM Conference on Optimization*, Houston, TX, May 1987.

[3] A.R. Conn, N. Gould, and Ph.L. Toint. Conjugate gradient methods for nonconvex quadratic programs. Communication with A.R. Conn, April, 1995.

[4] G. Golub and C. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, 1989.

[5] M. Lalee, J. Nocedal, and T. Plantenga. On the implementation of an algorithm for large-scale equality constrained optimization. Submitted to SIAM J. Optimization, December, 1993.

[6] S. Nash and A. Sofer. Preconditioning of reduced matrices. Technical Report 93-01, Department of Operations Research and Engineering George Mason University, 1993.

[7] E. Omojokun. *Trust Region Algorithms for Optimization with Nonlinear Equality and Inequality Constraints.* PhD thesis, University of Colorado, 1989.

[8] T. Plantenga. *Large-scale nonlinear constrained optimization using trust regions.* PhD thesis, Northwestern University, 1994.

[9] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM J. Numer. Anal*, 20:409–426, 1983.

Unlimited Release

Initial Distribution:

MS9001	Tom Hunter, Attn:	
	2200	J. B. Wright
	5200	E. E. Ives
	8200	L. A. West
	8300	W. J. McLean
	8400	R. C. Wayne
	8700	T. M. Dyer
	8800	L. A. Hiles
	8900	D. L. Crawford

MS9004	M. E. John
MS9214	L. M. Napolitano, Jr.
MS9214	T. D. Plantenga (10)
MS9021	Technical Communications, 8815, for OSTI (10)
MS9021	Technical Communications,8815/Technical Library, MS0899, 4414
MS0899	Technical Library, 4414 (4)
MS9018	Central Technical Files, 8950-2 (3)