Trust-region Interior-point Algorithms for Minimization Problems with Simple Bounds

John Dennis
Luis Vicente

CRPC-TR94478-S
November 1994

Center for Research on Parallel Computation
Rice University
6100 South Main Street
CRPC - MS 41
Houston, TX 77005

Revised: June, 1995. Also available as CAAM-TR94-42 from the Department of Computational and Applied Mathematics, Rice University.
Trust-region interior–point algorithms for minimization problems with simple bounds

J. E. Dennis † Luís N. Vicente †

Abstract

Two trust–region interior–point algorithms for the solution of minimization problems with simple bounds are presented. The algorithms scale the local model in a way proposed by Coleman and Li [1], but they are new otherwise. The first algorithm is more usual in that the trust region and the local quadratic model are consistently scaled. The second algorithm proposed here uses an unscaled trust region. A first–order convergence result for these algorithms is given and dogleg and conjugate–gradient algorithms to compute trial steps are introduced. Some numerical examples that show the advantages of the second algorithm are presented.

Keywords. trust–region methods, interior–point algorithms, Dikin–Karmarkar ellipsoid, Coleman and Li scaling, simple bounds.

AMS subject classification. 49M37, 90C20, 90C30

1 Introduction

In this note we consider the box–constrained minimization problem

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad a \leq x \leq b,
\end{align*}
\]

where \( x \in \mathbb{R}^n, a \in (\mathbb{R} \cup \{-\infty\})^n, b \in (\mathbb{R} \cup \{+\infty\})^n \) and \( f \) maps \( \mathbb{R}^n \) into \( \mathbb{R} \). We assume that \( f \) is continuously differentiable in the box \( B = \{ x \in \mathbb{R}^n : a \leq x \leq b \} \).

Coleman and Li [1] give an elegant diagonal scaling for this problem. It has the flavor of the Dikin–Karmarkar scaling, but it has a direct connection to the first–order necessary optimality conditions. A diagonal element corresponding to what appears to be a constraining bound is the same as in the Dikin–Karmarkar scaling. In their algorithms, they allow the elliptical trust region thus defined to have trust radius greater than one, so that some infeasible points are inside the trust region. As is usual, in their algorithms the trust region and the quadratic model are consistently scaled.

We adopt a version of the Coleman and Li scaling for the local quadratic model in both our algorithms. The first algorithm that we propose here uses the same scaling for the trust region,
and so it is similar to the Coleman and Li algorithms. However, the trial steps are computed completely differently. The second algorithm that we suggest maintains the trust region in the unscaled variables. In both algorithms, the trial step computation is very simple and convenient with respect to staying strictly inside $B$. We present some numerical examples to illustrate the advantages of the second algorithm.

There are three points of novelty here:

1. An improved interior–point algorithm for the solution of (1).

2. A trust–region convergence analysis for a fraction of Cauchy decrease condition in which the scaling of the direction that defines the Cauchy step does not come from the ellipsoidal norm that defines the trust region.

3. A new way of extending dogleg and conjugate–gradient algorithms for the solution of trust–region subproblems that arise in unconstrained optimization, to trust–region subproblems that appear when the simple bounds are included.

A point $x_*$ in $B$ is said to be a first–order Karush–Kuhn–Tucker (KKT) point if

$$a_i (x_*) < b_i \implies (\nabla f(x_*))_i = 0,$$

$$(x_*) = a_i \implies (\nabla f(x_*))_i \geq 0 \quad \text{and}$$

$$(x_*) = b_i \implies (\nabla f(x_*))_i \leq 0,$$

or equivalently if,

$$D(x_*)^p \nabla f(x_*) = 0,$$

where $p > 0$ is arbitrary and $D(x)$ is a diagonal matrix whose diagonal elements are given by

$$(D(x))_{ii} = \begin{cases} (b - x)_i & \text{if } (\nabla f(x))_i < 0 \text{ and } b_i < +\infty \\ 1 & \text{if } (\nabla f(x))_i < 0 \text{ and } b_i = +\infty \\ (x - a)_i & \text{if } (\nabla f(x))_i \geq 0 \text{ and } a_i > -\infty \\ 1 & \text{if } (\nabla f(x))_i \geq 0 \text{ and } a_i = -\infty, \end{cases}\)$$

for $i = 1, \ldots, n$.

We show that a sequence $\{x_k\}$ generated by either of our algorithms satisfies

$$\lim_k \|D(x_k)^p \nabla f(x_k)\| = 0,$$

for any $p \geq \frac{1}{2}$. It is important to note that these results are obtained under very mild assumptions on the trial steps, and that the sequence of approximations to the Hessian matrix of $f$ is assumed only to be bounded.

This paper is organized as follows. In Section 2 we introduce the conditions that we need to impose on the trial steps for the algorithms mentioned before. These are described in Section 3. In Section 4 we give a unified first–order convergence result for the trust–region interior–point algorithms. In Section 5 we describe dogleg and conjugate–gradient algorithms to compute the trial steps. Finally, in Section 6 we present some numerical examples and some final conclusions. In this paper $\| \cdot \|$ represents the $\ell_2$ norm, and $I_l$ the identity matrix of order $l$. 

2
2 Trust-region subproblems and trial steps

In this section we motivate the computation of the trial steps and present the conditions that these trial steps have to satisfy. The algorithms generate a sequence of iterates \( \{x_k\} \) where \( x_k \) is strictly feasible, in other words where \( a < x_k < b \). Given \( x_k \) we compute a trial step \( s_k \), and decide whether to accept it or not. If \( s_k \) is accepted then \( x_{k+1} = x_k + s_k \), otherwise \( x_{k+1} = x_k \).

2.1 Motivation

Our approach begins like sequential quadratic programming (SQP). Before we think about globalization of SQP, we look at the local quadratic programming (QP) subproblem

\[
\begin{align*}
\text{minimize} & \quad \Psi_k(s) \\
\text{subject to} & \quad \sigma_k(x_k - a) \leq s \leq \sigma_k(b - x_k),
\end{align*}
\]

gotten by building a quadratic model \( \Psi_k(s) = f(x_k) + g_k^T s + \frac{1}{2} s^T H_k s \), of \( f(x_k + s) \) about \( x_k \), where \( g_k = \nabla f(x_k) \), and \( H_k \) is an approximation to the Hessian matrix \( \nabla^2 f(x_k) \) of \( f \) evaluated at \( x_k \). Here \( \sigma_k \in [0, 1) \) and \( \sigma \in (0, 1) \) is fixed for all \( k \). Enforcing these bounds at every iteration ensures that the solutions to the subproblems remains strictly feasible for the original problem.

For this quadratic problem, we like the idea of the affine-scaling algorithm, i.e., we rewrite the quadratic problem in a basis \( \hat{s} = D_{\tau}^{-1} s \), for which the distances to the constraints of the current iterate \( x_k \) are the same in order that all directions be equally free for use in decreasing the objective function from \( x_k \). Here we follow the concept of Coleman and Li [1] and choose \( D_k \) as \( D(x_k) \). They actually use \( D_k = D(x_k)^{\frac{1}{2}} \). However, either choice has the effect of only rescaling those components that appear, from the sign of the gradient, to threaten to restrict the step. This gives the local QP subproblem:

\[
\begin{align*}
\text{minimize} & \quad \Psi_k(D_k \hat{s}) \\
\text{subject to} & \quad \sigma_k D_k^{-1} (x_k - a) \leq \hat{s} \leq \sigma_k D_k^{-1} (b - x_k),
\end{align*}
\]

gotten by building a quadratic model \( \Psi_k(D_k \hat{s}) = f(x_k) + (D_k g_k)^T \hat{s} + \frac{1}{2} \hat{s}^T D_k H_k D_k \hat{s} \). In this subproblem there is an explicit scaling given by \( D_k \) in the \( \hat{s} \) basis. For instance, the \( \hat{s} \) steepest-descent direction in the \( \ell_2 \) norm is given by \( -D_k g_k \).

Thus, we would like to minimize this quadratic function over a trust region with the requirement that \( x_k + s_k = D_k(\hat{x}_k + \hat{s}_k) \) has to be strictly feasible. Although we do this in the original basis \( s \) so that we can work always with the same variables, we bring the scaling that is used in the basis \( \hat{s} \). The reference trust–region subproblem that we consider, written in the original basis, is the following:

\[
\begin{align*}
\text{minimize} & \quad \Psi_k(s) \\
\text{subject to} & \quad \|S_k^{-1} s\| \leq \epsilon_k, \\
& \quad \sigma_k(x_k - a) \leq s \leq \sigma_k(b - x_k),
\end{align*}
\]

where \( \epsilon_k \) is the trust radius, and \( S_k \) is a \( n \times n \) nonsingular matrix. This subproblem is implicitly scaled by \( D_k^2 \) so that the direction \( -D_k g_k \) defined in the \( \hat{s} \) variables is now given by \( -D_k(D_k g_k) = -D_k^2 g_k \) in the \( s \) variables.
The two algorithms suggested here differ mainly in their choice of $S_k$. We discuss that issue now.

If we continue to follow the affine-scaling idea, then we use the ellipse defined at each iterate in the original $s$ coordinates by the $\ell_2$ norm on these new coordinates $\hat{s}$ to help to enforce the bounds. In other words, we would choose $S_k = D_k$, and the shape of the trust region (3) would be ellipsoidal in the original basis. We note that the choice made by Coleman and Li [1] is $S_k = D_k^{\frac{1}{2}}$.

This substitution of one ellipsoidal constraints for all the bound constraints was a prime motivation for interior-point methods. However from the beginning of the computational study of interior-point methods, it was found to be important to allow steps past the boundary of this ellipsoid, as long as they still satisfy the subproblem bound constraints. This translates here to saying that if the trust region is to have the ellipsoidal shape, then the trust radius should be allowed to exceed one, and so the trust region really is not used to enforce the bound constraints. Sometimes, the subproblem is further biased away from the bounds by adding a barrier term to the model (2).

The motivation for the second algorithm is that there is no reason to use the ellipsoid to define the shape of the trust region if it is not useful for enforcing the bounds. In fact, there are even more good reasons not to use it here than in the linear programming problem. One of the most important is that for nonlinear programs $x_*$ may lie strictly inside $B$. This happens in problems where the bounds are really to define the region of interest. If $x_k$ is near a bound which is not active at $x_*$, then many iterations may be required to move off that bound.

Hence we choose $S_k$ to be the identity in the second algorithm.

It is important to point out that Coleman and Li [1] present a different motivation to their algorithms. They see their algorithms as applying Newton’s method to the system of nonlinear (and nondifferentiable equations) $D(x)\nabla f(x) = 0$. As result, they add to $H_k$ a diagonal matrix defined to overcome the nondifferentiability. See their paper for more details.

The first-order convergence result given in Section 4 holds for any $D_k$ of the form $D_k^p$ with $p \geq \frac{1}{2}$. We motivated the algorithms by using $p = 1$ but for the rest of this paper we assume that $p$ is any number greater or equal than $\frac{1}{2}$.

### 2.2 What to impose on the trial steps

We need to define the Cauchy step associated with the trust–region subproblem (2)–(4). This Cauchy step $c_k$ is defined as the solution of

\[
\begin{align*}
\text{minimize} \quad & \Psi_k(s) \\
\text{subject to} \quad & \|S_k^{-1}s\| \leq \delta_k, \quad s \in \text{span}\{-D_k^{2p}g_k\}, \\
& \sigma_k(x_k - a) \leq s \leq \sigma_k(b - x_k).
\end{align*}
\]

As in many trust–region algorithms, $s_k$ is required to give a decrease on $\Psi_k(s)$ smaller than a uniform fraction of the decrease given by $c_k$ for the same function $\Psi_k(s)$. This condition is often called fraction of Cauchy decrease, and in this case is

\[
\Psi_k(0) - \Psi_k(s_k) \geq \beta (\Psi_k(0) - \Psi_k(c_k)),
\]

where $\beta$ is positive and fixed across all the iterations.
Coleman and Li [1] define the fraction of Cauchy decrease condition in a different way by using \( \sigma_k = 1 \) in (5) although they suggest \( \sigma_k \in (0,1) \) in the computation of the trial step \( s_k \). Our definition leads naturally to the condition (6) holding for any trial step generated by the algorithms that we propose in Section 5.

3 TRIP algorithms

To decide whether to accept or reject a trial step \( s_k \), however it is computed, we need to evaluate the ratio \( r_k = \text{ared}(s_k)/\text{pred}(s_k) \), where \( \text{ared}(s_k) = f(x_k) - f(x_{k+1}) \) is the actual decrease and \( \text{pred}(s_k) = \Psi_k(0) - \Psi_k(s_k) \) is the predicted decrease. We describe next the steps of the algorithms leaving the computation of the trial steps for Section 5.

Trust-region interior-point (TRIP) algorithms

1. Choose \( \delta_0 > 0 \), \( x_0 \) such that \( a < x_0 < b \), and \( p, \epsilon, \sigma, \alpha \) and \( \eta \) such that \( p \geq \frac{1}{2}, \epsilon > 0 \) and \( 0 < \sigma, \alpha, \eta < 1 \).

2. For \( k = 0, 1, 2, \ldots \) do

   2.1. If \( \| D_k^p g_k \| \leq \epsilon \), stop and return \( x_k \) as a solution for (1).

   2.2. Compute a trial step \( s_k \) satisfying (6), \( \| S_k^{-1} s_k \| \leq \delta_k \) and \( \sigma_k(x_k - a) \leq s_k \leq \sigma_k(b - x_k) \), where \( \sigma_k \in [\sigma, 1) \).

   2.3. If \( r_k < \eta \) reject \( s_k \), set \( \delta_{k+1} = \alpha \| s_k \|, \ x_{k+1} = x_k \) and choose \( H_{k+1} \) (for instance \( H_{k+1} = H_k \)).

   If \( r_k \geq \eta \) accept \( s_k \), choose \( \delta_{k+1} \geq \delta_k \), set \( x_{k+1} = x_k + s_k \) and compute \( H_{k+1} \).

Of course the rules to update the trust radius can be much more involved but the above suffices to prove convergence results and to understand the trust-region mechanism.

The choices \( S_k = D_k^p \) and \( S_k = I_n \) correspond respectively to the first and second algorithms.

4 First-order convergence result

To analyze the convergence properties of the TRIP algorithms, all we need to do is to express (6) in a more useful form. We do this in the following technical lemma, and it is not a surprise to see that the proof follows the proof given by Powell (see [5, Theorem 4] and [3, Lemma 4.8]) for the unconstrained minimization case.

**Lemma 4.1** If \( s_k \) satisfies (6) then

\[
\Psi_k(0) - \Psi_k(s_k) \geq \frac{1}{2} \beta \| \hat{g}_k \| \min \left\{ \frac{\| \hat{g}_k \|}{\| H_k \|}, \min \left\{ \frac{\| \hat{g}_k \|}{\| S_k^{-1} D_k^p \hat{g}_k \|}, \delta_k, \frac{\| \hat{g}_k \|}{\| h_k \|_\infty} \right\} \right\},
\]

where \( \hat{g}_k = D_k^p g_k, \ \hat{H}_k = D_k^p H_k D_k^p \) and \( h_k \) is a vector in \( \mathbb{R}^n \) defined by

\[
(h_k)_i = \frac{|(g_k)_i|}{\min\{(x_k - a)_i, (b - x_k)_i\}},
\]

for \( i = 1, \ldots, n \).
Proof. Define \( \psi : \mathbb{R}^+ \to \mathbb{R} \) as \( \psi(t) = \Psi_k(-tD_k^p\frac{\partial^2 h}{\partial \alpha^2}) - \Psi_k(0) \). Then \( \psi(t) = -||\hat{g}_k||t + \frac{\nu}{2}t^2 \) where \( v_k = \frac{\gamma_k}{\|\partial_k\|^2} \). Now we need to minimize \( \psi \) in \([0,T_k]\) where \( T_k \) is given by

\[
T_k = \min \left\{ \frac{\|\hat{g}_k\|}{\|S_k^{-1}D_k^p\partial_k\|} \right\} \sigma_k \|\hat{g}_k\| \min \left\{ \frac{(x_k-a)}{(\gamma_k)^{1-2\nu}} : (g_k)_i > 0 \right\},
\]

\[
\sigma_k \|\hat{g}_k\| \min \left\{ -\frac{(b-x_k)}{(\gamma_k)^{1-2\nu}} : (g_k)_i < 0 \right\} \right\}.
\]

Let \( t_k^* \) be the minimizer of \( \psi \) in \([0,T_k]\). If \( t_k^* \in (0,T_k) \) then

\[
\psi(t_k^*) = -\frac{1}{2} \|\hat{g}_k\|^2 \leq \frac{1}{2} \|\hat{g}_k\|^2 H_k. \tag{7}
\]

If \( t_k^* = T_k \) then either \( v_k > 0 \) in which case \( \frac{\|\hat{g}_k\|}{v_k} \geq T_k \) or \( v_k \leq 0 \) in which case \( v_kT_k \leq \|\hat{g}_k\| \). In either event,

\[
\psi(t_k^*) = \psi(T_k) = -T_k \|\hat{g}_k\| + \frac{v_k}{2}T_k^2 \leq -\frac{T_k}{2} \|\hat{g}_k\|. \tag{8}
\]

Now we can combine (7) and (8) and get

\[
\Psi_k(0) - \Psi_k(s_k) \geq \beta(\Psi_k(0) - \Psi_k(s_k)) = -\beta \psi(t_k^*)
\]

\[
\geq \frac{1}{2} \beta \|\hat{g}_k\| \min \left\{ \|\hat{g}_k\|, T_k \right\}
\]

\[
\geq \frac{1}{2} \beta \|\hat{g}_k\| \min \left\{ \|\hat{g}_k\|, \min \left\{ \frac{\|\hat{g}_k\|}{\|S_k^{-1}D_k^p\partial_k\|} : \sigma_k \right\} \right\}. \Box
\]

Now the following result is a consequence of Lemma 4.1. The proof follows the same steps as the proof of Theorem 3.5 of Coleman and Li [1].

**Theorem 4.1** Let \( f \) be continuously differentiable and bounded below on \( \mathcal{L}(x_0) = \{ x \in \mathcal{B} : f(x) \leq f(x_0) \} \), where \{\( x_k \)\} is a sequence generated by the TRIP algorithms. If \( H_k \) and \( S_k^{-1}D_k^p \) are uniformly bounded and \( \mathcal{L}(x_0) \) is compact then

\[
\lim_k \|D_k^p g_k\| = 0.
\]

The choices \( S_k = D_k^p \) and \( S_k = I_n \) produce bounded sequences \( \{S_k^{-1}D_k^p\} \), under the assumption that \( \mathcal{L}(x_0) \) is compact. They correspond to the algorithms that we propose.

5 Algorithms to compute trial steps

As in unconstrained minimization we have dogleg and conjugate–gradient algorithms to compute a trial step \( s_k \) that satisfies the fraction of Cauchy decrease condition (6).

The conjugate–gradient algorithm to compute a trial step \( s_k \) is very similar to the conjugate–gradient algorithm proposed by Steihaug [6] and Toint [7] for unconstrained minimization. The only difference is caused by the fact that \( x_k + s_k \) has to be strictly feasible.
Conjugate–gradient algorithm for the computation of \( s_k \)

1. Set \( s^0 = 0, r_0 = -g_k, q_0 = D_k^2 r_0 \) and \( d_0 = q_0 \). Choose small positive tolerances \( \epsilon_0 \) and \( \epsilon_1 \).

   If \( \sqrt{r_0^T q_0} \leq \epsilon_0 \) stop the algorithm and set \( s_k = d_0 \).

2. For \( i = 0, 1, 2, \ldots \) do

   2.1. Compute \( \gamma_i = \frac{r_i^T q_i}{d_i^T H_k d_i} \).

   2.2. If \( \gamma_i \leq 0 \) compute \( \tau_1 \) such that \( \| S_k^{-1} (s^i + \tau_1 d_i) \| = \delta_k \) and \( \tau_2 = \min \{ \frac{(x_k + s^i d_i, (d_i)_j < 0 \} \) and \( \tau_3 = \min \{ \frac{(b - x_k - s^i, (d_i)_j > 0) \} \). Set \( \tau = \min \{ \tau_1, \sigma_k \tau_2, \sigma_k \tau_3 \} \) and stop with \( s_k = s^i + \tau d_i \).

   2.3. Iterate \( s^{i+1} = s^i + \gamma_i d_i \). If \( \| S_k^{-1} s^{i+1} \| > \delta_k \) or \( \min \{ \frac{(x_k - a + s^i)}{\gamma_i (d_i)_j}, (d_i)_j < 0 \} > 1/\sigma_k \) or \( \min \{ \frac{(b - x_k - s^i)}{\gamma_i (d_i)_j}, (d_i)_j > 0 \} > 1/\sigma_k \), then stop with \( s_k = s^i + \tau d_i \), where \( \tau \) is given as in 2.2.

   2.4. Update the residuals \( r_{i+1} = r_i - \gamma_i H_k d_i \) and \( q_{i+1} = D_k^2 r_{i+1} \). If \( \sqrt{\frac{r_{i+1}^T q_{i+1}}{r_{i}^T q_{i}}} \leq \epsilon_1 \) then stop and set \( s_k = s^{i+1} \). Otherwise compute \( \alpha_i = \frac{r_{i+1}^T q_{i+1}}{r_{i}^T q_{i}} \) and update the direction \( d_{i+1} = q_{i+1} + \alpha_i d_i \).

As before we have the choices \( S_k = D_k^p \) and \( S_k = I_n \).

A dogleg algorithm to compute a trial step \( s_k \) similar to the dogleg algorithm proposed by Powell [4] for unconstrained minimization can also be applied. Since both dogleg and the conjugate–gradient algorithms start by minimizing the quadratic function \( \Psi_k(s) \) along the direction \( -D_k^p g_k \), it is a simple matter to see that any trial step \( s_k \) computed by using these algorithms satisfies the fraction of Cauchy decrease (6) with \( \beta = 1 \).

Now we briefly describe how Coleman and Li [1] compute the trial steps. They define \( p_k \) as the solution of the trust–region subproblem

\[
\begin{align*}
\text{minimize} & \quad \Psi_k(s) \\
\text{subject to} & \quad \| D_k^{-\frac{1}{2}} s \| \leq \delta_k,
\end{align*}
\]

and compute the Cauchy point \( c_k \) for some \( \sigma_k \in (0, 1) \). They propose two algorithms. In the first, called double trust–region method, they scale \( p_k \) into the interior of \( B \) and accept or reject \( s_k \) based on actual versus predicted decrease and fraction of Cauchy decrease. In the second, called practical trust–region algorithm, they choose \( s_k \) to be either \( c_k \) or the scaled \( p_k \) according to a fraction of Cauchy decrease condition. Then they accept or reject \( s_k \) based on actual versus predicted decrease.

6 Numerical examples and conclusions

We have implemented the TRIP algorithms using MATLAB 4.2a in a Sun (Sparc) workstation. We have used \( \delta_0 = 1, p = 1, \sigma_k = \sigma = 0.99995 \) for all \( k \), \( \epsilon_0 = 10^{-10}, \epsilon_1 = 10^{-8} \) and \( \epsilon = 10^{-5} \).
We have tested the algorithms in a set of problems given in Conn, Gould and Toint [2]. This set of problems is divided in two groups, labeled by U and C (see Table 1). In problems U, the solution lies in the interior of $B$ and therefore these problems correspond to the situation described in Section 2.1. In the cases where the initial point given in [2] is not strictly feasible, we scale it back into the interior of $B$ according to the rules used in [1]. The scheme 2.3 (see Section 3) used to update the trust radius is the following:

- If $r_k < 10^{-4}$, reject $s_k$ and set $\delta_{k+1} = 0.5\|S_k^{-1}s_k\|$.
- If $10^{-4} \leq r_k < 0.1$, reject $s_k$ and set $\delta_{k+1} = 0.5\|S_k^{-1}s_k\|$.
- If $0.1 \leq r_k < 0.75$, accept $s_k$ and set $\delta_{k+1} = \delta_k$.
- If $r_k \geq 0.75$, accept $s_k$ and set $\delta_{k+1} = 2\delta_k$.

We also stopped the algorithms when the trust radius was reduced below $10^{-16}$. These failures are indicated by * in Table 1. Our stopping criteria is different from the stopping criteria used by Coleman and Li. We stop if either $\|D_k\nabla f(x_k)\| \leq 10^{-3}$ or the trust radius is reduced below $10^{-16}$. They stop when $D_k^\frac{1}{2}H_kD_k^\frac{1}{2}$ is positive definite and $\Psi_k(s_k) < 0.5 \times 10^{-12}$. Our stopping criteria is of the type given in [2].

The results are given in Table 1. In Table 2 we list the total number of function and gradient evaluations taken by both approaches to solve problems U and C. From this table we observe that the second algorithm ($S_k = I_n$) performed better. We believe that this will be more clearly the case in larger problems.

**Acknowledgments**

We are grateful to Matthias Heinkenschloss, Virginia Polytechnic Institute and State University, for his many suggestions that have improved the presentation of this paper.

**References**


<table>
<thead>
<tr>
<th>Problem</th>
<th>n</th>
<th>$S_k = D_k$</th>
<th>$S_k = I_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENROSE U</td>
<td>8</td>
<td>24 35</td>
<td>28 36</td>
</tr>
<tr>
<td>GENROSE C</td>
<td>8</td>
<td>9 9</td>
<td>8 8</td>
</tr>
<tr>
<td>CHAINROSE U</td>
<td>25</td>
<td>15 17</td>
<td>16 20</td>
</tr>
<tr>
<td>CHAINROSE C</td>
<td>25</td>
<td>22 29</td>
<td>22 26</td>
</tr>
<tr>
<td>DEGENROSE U</td>
<td>25</td>
<td>31 39</td>
<td>28 29</td>
</tr>
<tr>
<td>DEGENROSE C</td>
<td>25</td>
<td>33 42</td>
<td>27 32</td>
</tr>
<tr>
<td>GENWOOD U</td>
<td>8</td>
<td>38 38</td>
<td>39 51</td>
</tr>
<tr>
<td>GENWOOD C</td>
<td>8</td>
<td>9 9</td>
<td>8 8</td>
</tr>
<tr>
<td>CHAINWOOD U</td>
<td>8</td>
<td>38 57</td>
<td>37 50</td>
</tr>
<tr>
<td>CHAINWOOD C</td>
<td>8</td>
<td>9 9</td>
<td>10 10</td>
</tr>
<tr>
<td>HOSC45 U</td>
<td>10</td>
<td>11 11</td>
<td>12 12</td>
</tr>
<tr>
<td>HOSC45 C</td>
<td>10</td>
<td>11 11</td>
<td>13 13</td>
</tr>
<tr>
<td>BROYDEN1A U</td>
<td>30</td>
<td>14 14</td>
<td>14 14</td>
</tr>
<tr>
<td>BROYDEN1A C</td>
<td>30</td>
<td>21 21</td>
<td>22 22</td>
</tr>
<tr>
<td>BROYDEN1B U</td>
<td>30</td>
<td>7 7</td>
<td>7 7</td>
</tr>
<tr>
<td>BROYDEN1B C</td>
<td>30</td>
<td>20 20</td>
<td>17 17</td>
</tr>
<tr>
<td>BROYDEN2A U</td>
<td>30</td>
<td>16 24</td>
<td>15 15</td>
</tr>
<tr>
<td>BROYDEN2A C</td>
<td>30</td>
<td>22 22</td>
<td>23 23</td>
</tr>
<tr>
<td>BROYDEN2B U</td>
<td>30</td>
<td>9 9</td>
<td>8 8</td>
</tr>
<tr>
<td>BROYDEN2B C</td>
<td>30</td>
<td>24 24</td>
<td>23 23</td>
</tr>
<tr>
<td>TOINTBROY U</td>
<td>30</td>
<td>8 8</td>
<td>*9 *37</td>
</tr>
<tr>
<td>TOINTBROY C</td>
<td>30</td>
<td>21 21</td>
<td>21 21</td>
</tr>
<tr>
<td>TRIG U</td>
<td>10</td>
<td>11 22</td>
<td>10 15</td>
</tr>
<tr>
<td>TRIG C</td>
<td>10</td>
<td>13 13</td>
<td>12 12</td>
</tr>
<tr>
<td>TOINTTRIG U</td>
<td>10</td>
<td>*6 *27</td>
<td>6 6</td>
</tr>
<tr>
<td>TOINTTRIG C</td>
<td>10</td>
<td>22 28</td>
<td>12 12</td>
</tr>
<tr>
<td>CRAGGLEVY U</td>
<td>8</td>
<td>25 25</td>
<td>25 25</td>
</tr>
<tr>
<td>CRAGGLEVY C</td>
<td>8</td>
<td>22 27</td>
<td>21 21</td>
</tr>
<tr>
<td>PENALTY U</td>
<td>15</td>
<td>23 23</td>
<td>24 25</td>
</tr>
<tr>
<td>PENALTY C</td>
<td>15</td>
<td>27 27</td>
<td>30 30</td>
</tr>
<tr>
<td>AUGMLAGN U</td>
<td>15</td>
<td>19 23</td>
<td>20 24</td>
</tr>
<tr>
<td>AUGMLAGN C</td>
<td>15</td>
<td>26 27</td>
<td>27 29</td>
</tr>
<tr>
<td>BROWN1 U</td>
<td>10</td>
<td>18 18</td>
<td>18 18</td>
</tr>
<tr>
<td>BROWN1 C</td>
<td>10</td>
<td>28 28</td>
<td>28 28</td>
</tr>
<tr>
<td>BROWN3 U</td>
<td>10</td>
<td>8 8</td>
<td>8 8</td>
</tr>
<tr>
<td>BROWN3 C</td>
<td>10</td>
<td>10 10</td>
<td>9 9</td>
</tr>
<tr>
<td>BVP U</td>
<td>10</td>
<td>8 9</td>
<td>9 9</td>
</tr>
<tr>
<td>BVP C</td>
<td>10</td>
<td>9 9</td>
<td>10 10</td>
</tr>
<tr>
<td>VAR U</td>
<td>20</td>
<td>9 9</td>
<td>9 9</td>
</tr>
<tr>
<td>VAR C</td>
<td>20</td>
<td>8 8</td>
<td>8 8</td>
</tr>
</tbody>
</table>

Table 1: Numerical solution of small test problems. $n$ – number of variables, geval – number of gradient evaluations, feval – number of function evaluations.
\begin{table}
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
 & \multicolumn{2}{c|}{S_k = D_k} & \multicolumn{2}{c|}{S_k = I_n} \\
 & geval & feval & geval & feval \\
\hline
Problems U & 422 & 526 & 426 & 502 \\
Problems C & 444 & 472 & 429 & 440 \\
Total & 866 & 998 & 855 & 942 \\
\hline
\end{tabular}
\caption{Comparison of the two algorithms. geval – number of gradient evaluations, feval – number of function evaluations.}
\end{table}

