Two-Step Algorithms for Nonlinear Optimization with Structured Applications

Andrew R. Conn * Luís N. Vicente † Chandu Visweswariah ‡

Abstract

In this paper we propose extensions to trust-region algorithms in which the classical step is augmented with a second step that we insist yields a decrease in the value of the objective function. The classical convergence theory for trust-region algorithms is adapted to this class of two-step algorithms.

The algorithms can be applied to any problem with variable(s) whose contribution to the objective function is a known functional form. In the nonlinear programming package LANCELOT, they have been applied to update slack variables and variables introduced to solve minimax problems, leading to enhanced optimization efficiency. Extensive numerical results are presented to show the effectiveness of these techniques.

Keywords. Trust regions, line searches, two-step algorithms, spacer steps, slack variables, LANCELOT, minimax problems, expensive function evaluations, circuit optimization.

AMS subject classifications. 49M37, 90C06, 90C30

1 Introduction

In nonlinear optimization problems with expensive function and gradient evaluations, it is desirable to extract as much improvement as possible at each iteration of an algorithm. When the objective function contains a subset of variables that occurs in a predictable functional form, a second, computationally relatively inexpensive, update can be applied to these variables following a classical optimization step. The additional step provides a further reduction in the objective function and can lead to superior optimization efficiency. The two-step algorithms have been successfully applied to the updating of slack variables and to a particular formulation of minimax problems, as is indicated by numerical results on a variety of problems. In these instances a subset of variables (slack variables and variables introduced to solve minimax problems) appears in a fixed, known algebraic form in the objective function. However, since it can be applied to any problem where a subset of the variables can be optimized relatively cheaply compared with the cost of evaluating the entire function (for example if some terms require simulation and other independent terms are

*Department of Mathematical Sciences, IBM T. J. Watson Research Center, Route 134 and Taconic, Room 33-206, Yorktown Heights NY 10598.

†Departamento de Matemática, Universidade de Coimbra, 3000 Coimbra, Portugal. This work started when this author was visiting the IBM T. J. Watson Research Center at Yorktown Heights and was supported in part by Centro de Matemática da Universidade de Coimbra, Instituto de Telecomunicações, FCT, Praxis XXI 2/2.1/MAT/346/94, and IBM Portugal.

‡Computer Architecture and Design Automation, IBM T. J. Watson Research Center, Route 134 and Taconic, Room 33-156, Yorktown Heights NY 10598.

available analytically), their applicability is really rather broad. We propose modifications to existing nonlinear optimization algorithms. An alternative approach, when feasible, is to reformulate the original problem by eliminating a subset of variables and then to apply the algorithms in the remaining variables (see, for example, Golub and Pereyra [17]).

This paper deals with two-step algorithms where the second step is required to yield a decrease in the value of the objective function. The analysis given here covers the global convergence of two-step trust-region algorithms and it is presented for the unconstrained minimization problem:

$$\text{minimize} \qquad f(y)\,, \tag{1}$$

where $y \in \mathbb{R}^p$, and $f : \mathbb{R}^p \longrightarrow \mathbb{R}$ is a twice continuously differentiable function. For both trust regions and line searches, one can consider two versions of the two-step algorithms, one called greedy and the other called conservative. The greedy version exploits as much as possible the decrease obtained by the second step, whereas the conservative approach calculates the second step only after the first step has been confirmed to satisfy the traditional criteria required for global convergence. We point out that the conservative two-step line-search algorithm is not new and can be found in the books by Bertsekas [1], Section 1.3.1, and Luenberger [19], Section 7.10, where the second step is called a spacer step. A description of the greedy and conservative two-step line-search algorithms can be found in [11].

In trust regions, if the second step is guaranteed to decrease the value of the objective function, global convergence of the type $\liminf_{k\longrightarrow+\infty} \|\nabla f(y_k)\| = 0$ is immediately attained. Further, in the cases where the first step would be rejected, the sum of the first and second steps has a better chance of being accepted (see Remark 3.1). To obtain $\lim_{k\longrightarrow+\infty} \|\nabla f(y_k)\| = 0$ either the norm of the second step has to be controlled by the trust region (see condition (13)) or the decrease on the objective function attained by the second step has to be of the order of magnitude of the norm of this step (see condition (12)).

The update of the slack variables referred to above motivated the study of the local rate of convergence of a two-step Newton's method. We show that a second Newton step in some of the variables retains the q-quadratic rate of convergence of the traditional Newton's method.

This paper is structured as follows. In Section 2 we introduce the two-step trust-region algorithms, and in Section 3 we analyze their global convergence properties. The local rate of the two-step Newton's method is studied in Section 4. The application of the two-step ideas to update slack variables and variables introduced for the solution of minimax problems is described in Section 5. Section 6 presents the numerical results obtained with LANCELOT using these updates for analytic problems and dynamic-simulation-based and analytic static-timing-based circuit optimization problems. Finally, some conclusions are drawn in Section 7.

2 Two-step trust-region algorithms

We first consider the trust-region framework presented in the paper by Moré [20] for unconstrained minimization. The (classical) trust-region algorithm builds a quadratic model of the form

$$m_k(y_k + s) \;=\; f(y_k) + \nabla f(y_k)^\top s + \frac{1}{2} s^\top H_k s$$

at the current point y_k, where H_k is an approximation to $\nabla^2 f(y_k)$ (note that $m_k(y_k) \;=\; f(y_k)$). Then a step s_k is computed by approximately solving the trust-region subproblem

$$\begin{aligned} &\text{minimize} \quad m_k(y_k + s) \\ &\text{subject to} \quad \|s\| \leq \Delta_k\,, \end{aligned} \tag{2}$$

where Δ_k is called the trust-region radius and $\|\cdot\|$ is an arbitrary norm. The new point $y_{k+1} = y_k + s_k$ is tested for acceptance. If the actual reduction $f(y_k) - f(y_k + s_k)$ is larger than a given fraction of the predicted reduction $m_k(y_k) - m_k(y_k + s_k)$, then the step s_k and the new point y_{k+1} are accepted. In this situation, the quadratic model $m_k(y_k + s)$ is considered to be a good approximation to the function $f(y)$ in the region $\|y_k - y\| \leq \Delta_k$. The trust radius may be increased. Otherwise, the quadratic model $m_k(y_k + s)$ is considered not to be a good approximation to the function $f(y)$ in the region $\|y - y_k\| \leq \Delta_k$. In this case, the new point y_{k+1} is rejected, and a new trust-region subproblem of the form (2) is solved for a smaller value of the trust radius. This simple trust-region algorithm is described below.

Algorithm 2.1 (Trust-region algorithm)

1. Given y_0, the value $f(y_0)$, the gradient $\nabla f(y_0)$ and an approximation H_0 to the Hessian of f at y_0, and the initial trust-region radius Δ_0. Set $k = 0$. Choose γ and α in $(0, 1)$.

2. Compute a step s_k based on the trust-region problem (2).

3. Compute
$$\rho_k \;=\; \frac{f(y_k) - f(y_k + s_k)}{m_k(y_k) - m_k(y_k + s_k)}\,.$$

4. In the case where
$$\rho_k > \alpha\,,$$
set
$$y_{k+1} = y_k + s_k\,,$$
compute H_{k+1}, and select Δ_{k+1} satisfying $\Delta_{k+1} \geq \Delta_k$.
Otherwise, set
$$y_{k+1} = y_k\,, \qquad H_{k+1} = H_k\,, \qquad \text{and} \qquad \Delta_{k+1} = \gamma\Delta_k\,.$$

5. Increment k by one and go to Step 2.

The mechanism used to update the trust radius that is described in Algorithm 2.1 is simple and suffices to prove convergence results. In practice, with the goal of improving optimization efficiency, one uses updating schemes that are more complex involving several subcases according to the value of ρ_k.

We propose in this paper a modification of this trust-region algorithm. We are motivated by a situation where it is desirable to update slack variables and variables introduced to solve minimax problems, at every iteration of the trust-region algorithm [7] implemented in LANCELOT [9]. See Section 5 for more details on practical applications.

The two-step trust-region algorithm is quite easy to describe. Suppose that after computing a step $\bar{s}_k$ based on the trust-region subproblem (2) we know some properties of the function $f(y)$ that enables us to compute a new step $\hat{s}_k$ for which we can guarantee that $f(y_k + \bar{s}_k + \hat{s}_k) < f(y_k + \bar{s}_k)$. In this situation we would certainly like to have $y_{k+1} = y_k + \bar{s}_k + \hat{s}_k$ and to test whether this new point should be accepted or not. This modification requires a careful redefinition of the actual and predicted reductions given for Algorithm 2.1. The new actual and predicted reductions that we propose are:

$$ared(y_k, \bar{s}_k, \hat{s}_k) \;=\; f(y_k) - f(y_k + \bar{s}_k + \hat{s}_k)\,, \tag{3}$$

$$pred(y_k, \bar{s}_k, \hat{s}_k) \;=\; m_k(y_k) - m_k(y_k + \bar{s}_k) + f(y_k + \bar{s}_k) - f(y_k + \bar{s}_k + \hat{s}_k)\,. \tag{4}$$

The new predicted reduction is the predicted reduction obtained by the first step plus the (actual) reduction obtained by the second step. The choice $pred(y_k, \bar{s}_k, \hat{s}_k) = m_k(y_k) - m_k(y_k + \bar{s}_k + \hat{s}_k)$ is not appropriate since the second step $\hat{s}_k$ is not computed using the model $m_k(y_k + s)$.

The two-step trust-region algorithm is given below.

Algorithm 2.2 (Two-step trust-region algorithm – Greedy)

1. Same as in Algorithm 2.1.
2. Compute a step $\bar{s}_k$ based on the trust-region problem (2).
3. If possible, find another step $\hat{s}_k$ such that

   $$f(y_k + \bar{s}_k + \hat{s}_k) \;<\; f(y_k + \bar{s}_k)\,.$$

   Otherwise, set $\hat{s}_k = 0$.
4. Compute

   $$\hat{\rho}_k \;=\; \frac{ared(y_k, \bar{s}_k, \hat{s}_k)}{pred(y_k, \bar{s}_k, \hat{s}_k)}\,.$$

5. In the case where

   $$\hat{\rho}_k > \alpha\,,$$

   set

   $$y_{k+1} = y_k + \bar{s}_k + \hat{s}_k\,,$$

   compute H_{k+1}, and select Δ_{k+1} satisfying $\Delta_{k+1} \;\geq\; \Delta_k$.

   Otherwise, set

   $$y_{k+1} = y_k\,, \qquad H_{k+1} = H_k\,, \qquad \text{and} \qquad \Delta_{k+1} = \gamma\Delta_k\,.$$

6. Increment k by one and go to Step 2.

The two-step trust-region Algorithm 2.2 evaluates the new point $y_k + \bar{s}_k + \hat{s}_k$ for acceptance after both steps $\bar{s}_k$ and $\hat{s}_k$ have been computed. We call this version "greedy" because it tries to take as much advantage as possible of the decrease obtained by the second step $\hat{s}_k$. Note that although the function f is evaluated twice in Algorithm 2.2, the reevaluation is often computationally inexpensive. The context in which we are particularly interested involves relatively expensive evaluations at $y_k + \bar{s}_k$ and evaluations at $y_k + \bar{s}_k + \hat{s}_k$ involving only a subset of the variables that are cheap to compute (see Section 5).

We could also consider a two-step trust-region algorithm where first an acceptable step $\bar{s}_k$ is determined, and only afterwards a second step $\hat{s}_k$ is computed. This algorithm is outlined below.

Algorithm 2.3 (Two-step trust-region algorithm – Conservative)

1. Same as in Algorithm 2.1.
2. Repeat
   (a) Compute a step $\bar{s}_k$ based on the trust-region problem (2).
   (b) Compute
   $$\rho_k \;=\; \frac{f(y_k) - f(y_k + \bar{s}_k)}{m_k(y_k) - m_k(y_k + \bar{s}_k)}\,.$$
   (c) If $\rho_k > \alpha$, then set
   $$\bar{y}_k = y_k + \bar{s}_k\,,$$
   compute Δ_{k+1} satisfying $\Delta_{k+1} \;\geq\; \Delta_k$, and set `accepted = true`.
   If $\rho_k \leq \alpha$, set $\Delta_k = \gamma\Delta_k$ and `accepted = false`.

   Until `accepted`.
3. If possible, find another step $\hat{s}_k$ such that
   $$f(\bar{y}_k + \hat{s}_k) \;<\; f(\bar{y}_k)\,.$$
   Otherwise, set $\hat{s}_k = 0$.
4. Set $y_{k+1} = \bar{y}_k + \hat{s}_k$.
5. Update H_k. Increment k by one and go to Step 2.

The same comments about the function evaluations apply to Algorithm 2.3 after the computation of a successful step $\bar{s}_k$. However, in the case of Algorithm 2.3, the function f has to be evaluated twice only in iterations corresponding to successful first steps $\bar{s}_k$.

3 Global convergence of the two-step trust-region algorithms

We analyze first the two-step trust-region Algorithm 2.2, i.e., the greedy version. The analysis for the conservative Algorithm 2.3 is similar.

In this section we make the assumption that $\{H_k\}$ is a bounded sequence. So, there exists a $\sigma > 0$ for which

$$\|H_k\| \;\leq\; \sigma \quad \text{for all } k\,. \tag{5}$$

We require the step $\bar{s}_k$ to satisfy a fraction of Cauchy decrease on the trust-region problem (2). In other words we ask $\bar{s}_k$ to satisfy

$$f(y_k) - m_k(y_k + \bar{s}_k) \;\geq\; \beta\,(m_k(y_k) - m_k(y_k + c_k))\;, \tag{6}$$

for $\beta \in (0, 1]$. The step c_k is called the Cauchy step, and it is defined as the solution of the scalar problem in the unknown η

$$\begin{aligned} \text{minimize} \quad & m_k(y_k + s) \\ \text{subject to} \quad & \|s\| \leq \Delta_k\,, \\ & s \;=\; \eta\nabla f(y_k)\,,\; \eta \in \mathbb{R}\,. \end{aligned}$$

There is a variety of algorithms that compute steps satisfying this condition (see [3], [22], [23], [25], and [26]).

Proposition 3.1 *If $\bar{s}_k$ satisfies a fraction of Cauchy decrease then:*

$$f(y_k) - m_k(y_k + \bar{s}_k) \;\geq\; \frac{\beta}{2}\|\nabla f(y_k)\| \min\left\{\Delta_k, \frac{\|\nabla f(y_k)\|}{\sigma}\right\} \tag{7}$$

where β and σ are as in (6) and (5) respectively.

Proof: See Powell [24], Theorem 4, or Moré [20], Lemma 4.8. □

If we use this proposition and the fact that $f(y_k + \bar{s}_k) > f(y_k + \bar{s}_k + \hat{s}_k)$, we obtain

$$\begin{aligned} pred(y_k, \bar{s}_k, \hat{s}_k) &= f(y_k) - m_k(y_k + \bar{s}_k) + f(y_k + \bar{s}_k) - f(y_k + \bar{s}_k + \hat{s}_k) \\ &\geq \frac{\beta}{2}\|\nabla f(y_k)\| \min\left\{\Delta_k, \frac{\|\nabla f(y_k)\|}{\sigma}\right\} + f(y_k + \bar{s}_k) - f(y_k + \bar{s}_k + \hat{s}_k) \\ &\geq \frac{\beta}{2}\|\nabla f(y_k)\| \min\left\{\Delta_k, \frac{\|\nabla f(y_k)\|}{\sigma}\right\}. \end{aligned} \tag{8}$$

This inequality is crucial to prove global convergence of the two-step algorithm. In particular, if the iteration k is successful, then

$$\begin{aligned} ared(y_k, \bar{s}_k, \hat{s}_k) &= f(y_k) - f(y_k + \bar{s}_k + \hat{s}_k) \\ &\geq \tfrac{\alpha\beta}{2}\|\nabla f(y_k)\| \min\left\{\Delta_k, \tfrac{\|\nabla f(y_k)\|}{\sigma}\right\}. \end{aligned} \tag{9}$$

We are ready to prove the first convergence result.

Theorem 3.1 *Consider a sequence $\{y_k\}$ generated by Algorithm 2.2 where $\bar{s}_k$ satisfies (6). If f is continuously differentiable and bounded below on*

$$\mathcal{L}(y_0) \;=\; \{y : f(y) \leq f(y_0)\},$$

and $\{H_k\}$ is a bounded sequence, then

$$\liminf_{k \longrightarrow +\infty} \|\nabla f(y_k)\| \;=\; 0. \tag{10}$$

So, if the sequence $\{y_k\}$ is bounded, there exists at least one limit point y_* for which $\nabla f(y_*) = 0$.

Proof: The proof is similar to the proof given in [20], Theorem 4.10.

Assume by contradiction that $\{\|\nabla f(y_k)\|\}$ is bounded away from zero, i.e., that there exists an $\epsilon > 0$ such that $\|\nabla f(y_k)\| \geq \epsilon$ for all k. As in [20], Theorem 4.10, we make direct use of (9) and of the rules that update the trust radius, to obtain:

$$\sum_{k=0}^{+\infty} \Delta_k \;<\; +\infty,$$

and so $\lim_{k \longrightarrow +\infty} \Delta_k = 0$.

The next step is to show that $\lim_{k \longrightarrow +\infty} |\hat{\rho}_k - 1| = 0$. Note that from the definitions (3) and (4), we have

$$\begin{aligned} ared(y_k, \bar{s}_k, \hat{s}_k) &- pred(y_k, \bar{s}_k, \hat{s}_k) \\ &= f(y_k) - f(y_k + \bar{s}_k) + \nabla f(y_k)^T \bar{s}_k + \tfrac{1}{2}\bar{s}_k^T H_k \bar{s}_k, \end{aligned} \tag{11}$$

which in turn, by using a Taylor series expansion and $\|\bar{s}_k\| \leq \Delta_k$, implies

$$|ared(y_k, \bar{s}_k, \hat{s}_k) - pred(y_k, \bar{s}_k, \hat{s}_k)| \ \leq \ o(\Delta_k)\,.$$

This inequality and (8) show that $|\hat{\rho}_k - 1|$ converges to zero. The rest of the proof follows a classical argument in trust regions: if $\hat{\rho}_k$ converges to one, the rules that update the trust radius show that Δ_k cannot converge to zero. So, a contradiction is attained and the proof is completed. □

The result of Theorem 3.1 does not require the step $\hat{s}_k$ to be $\mathcal{O}(\Delta_k)$ which may seem surprising. This result shows the appropriateness of the definitions given in (3) and (4) for the actual and predicted reductions. These definitions allow us to obtain the conditions (9) and (11) that are crucial to establish (10).

Remark 3.1 *It is also important to note that the definitions (3) and (4) can improve the acceptability of a step. In fact, we have*

$$\hat{\rho}_k \ = \ \frac{t_k + \rho_k}{t_k + 1} \ \equiv \ \hat{\rho}_k(t_k)\,,$$

where $t_k = \frac{f(y_k+\bar{s}_k)-f(y_k+\bar{s}_k+\hat{s}_k)}{m_k(y_k)-m_k(y_k+\bar{s}_k)}$, *and* $\rho_k \ = \ \frac{f(y_k)-f(y_k+\bar{s}_k)}{m_k(y_k)-m_k(y_k+\bar{s}_k)}$, *as before. We now note that* $\hat{\rho}_k(0) = \rho_k$ *and the function* $\hat{\rho}_k(t_k)$ *is strictly increasing if* $\rho_k < 1$. *In other words, in cases where a standard trust-region algorithm rejects a step the modified criterion is always better than the usual one. Further, it can be noted that* $\hat{\rho}_k - 1 = \frac{\rho_k - 1}{t_k + 1}$, *which indicates that all successful iterations of the the standard algorithm will also be successful in the modified two-step algorithm. In particular,* $\hat{\rho}_k > 1$ *whenever* $\rho_k > 1$.

The next step in the analysis is to prove that, with additional conditions on the second step, $\lim_{k \longrightarrow +\infty} \|\nabla f(y_k)\| = 0$.

Theorem 3.2 *Consider a sequence* $\{y_k\}$ *generated by Algorithm 2.2 where* $\bar{s}_k$ *satisfies (6). Assume that* f *is continuously differentiable and bounded below on* $\mathcal{L}(y_0)$ *and that* $\{H_k\}$ *is a bounded sequence. If* ∇f *is uniformly continuous on* $\mathcal{L}(y_0)$ *and if either*

$$f(y_k + \bar{s}_k) - f(y_k + \bar{s}_k + \hat{s}_k) \ \geq \ c_1 \|\hat{s}_k\|\,, \tag{12}$$

or

$$\|\hat{s}_k\| \leq c_2 \Delta_k\,, \tag{13}$$

where c_1 *and* c_2 *are positive constants independent of* k, *then*

$$\lim_{k \longrightarrow +\infty} \|\nabla f(y_k)\| \ = \ 0\,. \tag{14}$$

So, if the sequence $\{y_k\}$ *is bounded, every limit point* y_* *satisfies* $\nabla f(y_*) = 0$.

Proof: The proof is similar to the proof given in [20], Theorem 4.14. See also Thomas [27].

We show the result by contradiction. Assume therefore that there exists an $\epsilon_1 \in (0, 1)$ and a subsequence indexed by $\{m_i\}$ of successful iterates such that, for all m_i in this subsequence, $\|\nabla f(y_{m_i})\| \geq \epsilon_1$. Theorem 3.1 guarantees the existence of another subsequence indexed by $\{l_i\}$ such that $\|\nabla f(y_k)\| \geq \epsilon_2$, for $m_i \leq k < l_i$ and $\|\nabla f(y_{l_i})\| < \epsilon_2$ (where $\{m_i\}$ is without loss

of generality the subsequence previously mentioned). Here ϵ_2 is any real number chosen to be in $(0, \epsilon_1)$. Since $\{f(y_k) - f(y_{k+1})\}$ converges to zero, for k sufficiently large corresponding to successful iterations $m_i \leq k < l_i$

$$f(y_k) - f(y_{k+1}) \;\geq\; \kappa_1 \, \Delta_k + c_1 \|\hat{s}_k\| \tag{15}$$

holds if (12) is satisfied, and

$$f(y_k) - f(y_{k+1}) \;\geq\; \kappa_1 \, \Delta_k \tag{16}$$

holds otherwise with $\kappa_1 = \frac{\alpha\beta\epsilon_2}{2}$.

We consider the cases (12) and (13) separately. In both cases we make use of:

$$\begin{aligned} \|y_{m_i} - y_{l_i}\| &\leq \sum_{k=m_i}^{l_i-1} \|y_k - y_{k+1}\| \,, \\ f(y_{m_i}) - f(y_{l_i}) &= \sum_{k=m_i}^{l_i-1} [f(y_k) - f(y_{k+1})] \,. \end{aligned}$$

In the sums $\sum_{k=m_i}^{l_i-1}$ we consider only indices corresponding to successful iterations.

If (12) holds then we use (15) to obtain

$$\begin{aligned} \textstyle\sum_{k=m_i}^{l_i-1} [f(y_k) - f(y_{k+1})] &\geq \textstyle\sum_{k=m_i}^{l_i-1} [\kappa_1 \Delta_k + c_1 \|\hat{s}_k\|] \\ &\geq \min\{\kappa_1, c_1\} \textstyle\sum_{k=m_i}^{l_i-1} [\|\bar{s}_k\| + \|\hat{s}_k\|] \\ &\geq \min\{\kappa_1, c_1\} \textstyle\sum_{k=m_i}^{l_i-1} \|y_k - y_{k+1}\| \,. \end{aligned}$$

If (13) holds then we appeal to (16) and write

$$\begin{aligned} \textstyle\sum_{k=m_i}^{l_i-1} [f(y_k) - f(y_{k+1})] &\geq \textstyle\sum_{k=m_i}^{l_i-1} \kappa_1 \Delta_k \\ &\geq \tfrac{\kappa_1}{2} \min\{1, \tfrac{1}{c_2}\} \textstyle\sum_{k=m_i}^{l_i-1} [\|\bar{s}_k\| + \|\hat{s}_k\|] \\ &\geq \tfrac{\kappa_1}{2} \min\{1, \tfrac{1}{c_2}\} \textstyle\sum_{k=m_i}^{l_i-1} \|y_k - y_{k+1}\| \,. \end{aligned}$$

In either case we obtain

$$\|y_{m_i} - y_{l_i}\| \;\leq\; \kappa_2 \left(f(y_{m_i}) - f(y_{l_i})\right) \,,$$

and since the right hand side of this inequality goes to zero, so does the left hand side $\|y_{m_i} - y_{l_i}\|$. Since the gradient of f is uniformly continuous, we have for i sufficiently large that

$$\epsilon_1 \;\leq\; \|\nabla f(y_{m_i})\| \;\leq\; \|\nabla f(y_{m_i}) - \nabla f(y_{l_i})\| + \|\nabla f(y_{l_i})\| \;\leq\; 2\epsilon_2 \,.$$

Since ϵ_2 can be any number in $(0, \epsilon_1)$ this inequality contradicts the supposition. □

In the theorem above we required the norm of the step $\hat{s}_k$ to either be $\mathcal{O}(\Delta_k)$ or $\mathcal{O}\left(f(y_k + \bar{s}_k) - f(y_k + \bar{s}_k + \hat{s}_k)\right)$. The former condition can be enforced in Step 2 of the Algorithm 2.2, although this might not be beneficial and could lead to an inferior decrease.

We can obtain global convergence to a point that also satisfies the necessary second-order conditions for optimality. For this purpose, we require the step $\bar{s}_k$ to satisfy a fraction of optimal decrease for the trust-region problem (2). In other words we ask $\bar{s}_k$ to satisfy

$$f(y_k) - m_k(y_k + \bar{s}_k) \;\geq\; \beta\,(f(y_k) - m_k(y_k + s_k^*))\;, \tag{17}$$

where $\beta \in (0, 1]$, and s_k^* is an optimal solution of (2). (This condition can be weakened in several ways [20].) A step $\bar{s}_k$ satisfying a fraction of optimal decrease can be computed by using the algorithms proposed in [22] and [25] in the case where the trust-region norm is Euclidean. The global convergence result is the following.

Theorem 3.3 *Consider a sequence $\{y_k\}$ generated by Algorithm 2.2 where $H_k = \nabla^2 f(y_k)$ and $\bar{s}_k$ satisfies (17). If $\mathcal{L}(y_0)$ is compact and f is twice continuously differentiable on $\mathcal{L}(y_0)$, then there exists at least one limit point y_* for which $\nabla f(y_*) = 0$ and $\nabla^2 f(y_*)$ is positive semi-definite.*

Proof: The proof is basically the same as the proof of Theorem 4.7 in [22]. □

To obtain stronger global convergence results to second-order points, for instance the results in Theorems 4.11 and 4.13 in [22] (see also [21], Theorem 4.17, c and d), other conditions are required like $\|\hat{s}_k\|$ being of $\mathcal{O}(\Delta_k)$.

The next results show that the second step can preserve the nice local properties of the behavior of the trust radius that are typical in trust-region algorithms.

Theorem 3.4 *Let $\{y_k\}$ be a sequence generated by Algorithm 2.2 where $\bar{s}_k$ satisfies (6) and $H_k = \nabla^2 f(y_k)$. In addition, assume that the step $\hat{s}_k$ satisfies either condition (12) or condition (13). If f is twice continuously differentiable and bounded below on $\mathcal{L}(y_0)$ and $\{y_k\}$ has a limit point y_* such that $H_* = \nabla^2 f(y_*)$ is positive definite, then $\{y_k\}$ converges to y_*, all iterations are eventually successful, and $\{\Delta_k\}$ is bounded away from zero.*

Proof: From Theorem 3.2 we can guarantee that $\lim_{k \longrightarrow +\infty} \|\nabla f(y_k)\| = 0$. So, the proof is basically the same as the proof of Theorem 4.19 in [20]. □

An alternative to this result where we do not impose conditions (12) or (13) on the second step is given below. However we need to assume that $\{y_k\}$ converges to y_*.

Theorem 3.5 *Let $\{y_k\}$ be a sequence generated by Algorithm 2.2 where $\bar{s}_k$ satisfies (6) and $H_k = \nabla^2 f(y_k)$. If f is twice continuously differentiable on $\mathcal{L}(y_0)$ and $\{y_k\}$ converges to a point y_* such that $H_* = \nabla^2 f(y_*)$ is positive definite, then all iterations are eventually successful and $\{\Delta_k\}$ is bounded away from zero.*

Proof: The first step $\bar{s}_k$ yields a decrease in the quadratic model:

$$m_k(y_k) - m_k(y_k + \bar{s}_k) \;=\; -\nabla f(y_k)^\top \bar{s}_k - \frac{1}{2}\bar{s}_k^\top H_k \bar{s}_k \;\geq\; 0\,.$$

Thus, the assumptions made on H_k and H_* guarantee

$$\|\bar{s}_k\| \;\leq\; c_3 \|\nabla f(y_k)\|\,, \tag{18}$$

for sufficiently large k, which in turn, by using (8), implies

$$pred(y_k, \bar{s}_k, \hat{s}_k) \;\geq\; c_4 \|\bar{s}_k\|^2 \,. \tag{19}$$

(The constants c_3 and c_4 are independent of k.)

A Taylor series expansion for the expression (11) gives

$$|ared(y_k, \bar{s}_k, \hat{s}_k) - pred(y_k, \bar{s}_k, \hat{s}_k)| \;\leq\; o(\|\bar{s}_k\|^2) \,. \tag{20}$$

The fact that $\{y_k\}$ converges and the result $\liminf_{k \longrightarrow +\infty} \|\nabla f(y_k)\| = 0$ of Theorem 3.1, together imply $\lim_{k \longrightarrow +\infty} \|\nabla f(y_k)\| = 0$. Thus, from (18) we get $\lim_{k \longrightarrow +\infty} \|\bar{s}_k\| = 0$.

The proof is terminated with a typical argument in trust regions. From (19), (20) and $\lim_{k \longrightarrow +\infty} \|\bar{s}_k\| = 0$, we obtain the limit

$$\lim_{k \longrightarrow +\infty} \left| \frac{ared(y_k, \bar{s}_k, \hat{s}_k)}{pred(y_k, \bar{s}_k, \hat{s}_k)} - 1 \right| \;=\; 0 \,,$$

which shows, by appealing to the rules that update the trust radius, that all iterations are eventually successful and the trust radius is uniformly bounded away from zero. □

The global convergence analysis for Algorithm 2.3 is identical to the analysis given above for Algorithm 2.2. We point out that Algorithm 2.3 is well defined since at a nonstationary point it is always possible to find an acceptable first step. Also, for every k,

$$\begin{aligned} f(y_k) - f(y_{k+1}) &= f(y_k) - f(y_k + \bar{s}_k) + f(y_k + \bar{s}_k) - f(y_{k+1}) \\ &\geq \frac{\alpha\beta}{2} \|\nabla f(y_k)\| \, \min\left\{ \Delta_k, \frac{\|\nabla f(y_k)\|}{\sigma} \right\} + f(y_k + \bar{s}_k) - f(y_{k+1}) \\ &\geq \frac{\alpha\beta}{2} \|\nabla f(y_k)\| \, \min\left\{ \Delta_k, \frac{\|\nabla f(y_k)\|}{\sigma} \right\} . \end{aligned}$$

Thus, the results given in Theorems 3.1-3.5 hold for Algorithm 2.3. The lim inf-type result (10) is obtained under the classical assumptions for trust-region algorithms for unconstrained optimization. To obtain the lim-type result (14) one of the two conditions (12) and (13) is required.

In the case of the applications considered in Section 5, the decrease obtained by the second step $\hat{s}_k$ is always guaranteed to satisfy

$$f(y_k + \bar{s}_k) - f(y_k + \bar{s}_k + \hat{s}_k) \;\geq\; c_5 \|\hat{s}_k\|^2 \,. \tag{21}$$

Moreover, the objective function strictly decreases along the segment between the points $y_k + \bar{s}_k$ and $y_k + \bar{s}_k + \hat{s}_k$. In this case we can modify Step 3 of Algorithms 2.2 and 2.3 in such a way that we meet the requirements of Theorem 3.2. This modification is given below. It is easy to verify that $\hat{s}_k \neq 0$ satisfies $f(y_k + \bar{s}_k + \hat{s}_k) < f(y_k + \bar{s}_k)$ and either (12) or (13).

Algorithm 3.1 (Step 3 for Algorithms 2.2 and 2.3 – Quadratic decrease case)

3. Compute a step $\hat{s}_k$ such that

$$f(y_k + \bar{s}_k) - f(y_k + \bar{s}_k + \hat{s}_k) \;\geq\; c_5 \|\hat{s}_k\|^2 \,.$$

If $\|\hat{s}_k\| < \nu$, then scale $\hat{s}_k$ by $\min\{1, \frac{c_2 \Delta_k}{\nu}\}$ so that $\|\hat{s}_k\| \leq c_2 \Delta_k$ and $\hat{s}_k$ is not enlarged. (Otherwise (12) holds with $c_1 = \nu c_5$.)

The positive parameters ν and c_2 should be set *a priori* in Step 1 of Algorithms 2.2 and 2.3.

Of course, we would like to prove the result of Theorem 3.2 for the case where the condition (12) is replaced by the condition (21). However, such a result is unlikely to be true.

4 Local rate of convergence of a two-step Newton's method

In the next section we are interested in two-step algorithms where the second step is calculated as a Newton-type step in some of the variables. In this section we investigate the local rate of convergence for an algorithm where each step is composed of two Newton steps, the second being computed only for a subset of the variables. For this purpose let

$$y \;=\; \begin{pmatrix} x \\ u \end{pmatrix}.$$

Suppose the first step $\bar{s}_k$ is a full Newton step, i.e., $\bar{s}_k = -\nabla^2 f(y_k)^{-1}\nabla f(y_k)$. Let also

$$\bar{y}_k \;=\; \begin{pmatrix} \bar{x}_k \\ \bar{u}_k \end{pmatrix} \;=\; y_k + \bar{s}_k\,.$$

At the intermediate point $\bar{y}_k$, a Newton step is applied in the variables u with $x = \bar{x}_k$ fixed. This two-step Newton's method is described below.

Algorithm 4.1 (Two-step Newton's method)

1. Choose y_0.
2. For $k = 1, 2, \ldots$ do
   - 2.1 Compute $\bar{s}_k = -\nabla^2 f(y_k)^{-1}\nabla f(y_k)$ and set $\bar{y}_k = y_k + \bar{s}_k$.
   - 2.2 Compute $\hat{s}_k = \begin{pmatrix} 0 \\ -\nabla^2_{uu} f(\bar{y}_k)^{-1}\nabla_u f(\bar{y}_k) \end{pmatrix}$ and set $s_k = \bar{s}_k + \hat{s}_k$.
   - 2.3 Set $y_{k+1} = y_k + s_k$.

The proof of the local convergence rate of the two-step Newton's method requires a few modifications from the standard proof of Newton's method [12], Theorem 5.2.1. Recall that that proof of Newton's method is by induction.

Corollary 4.1 *Let f be twice continuously differentiable in an open set D where the second partial derivatives are Lipschitz continuous. If $\{y_k\}$ is a sequence generated by Algorithm 4.1 converging to a point $y_* \in D$ for which $\nabla f(y_*) = 0$ and $\nabla^2 f(y_*)$ is positive definite, then $\{y_k\}$ converges with a q-quadratic rate.*

Proof: If y_k is sufficiently close to y_*, the perturbation result [12], Theorem 3.1.4, can be used to prove the nonsingularity of the Hessian matrix $\nabla^2 f(y_k)$. Furthermore,

$$\|\bar{y}_k - y_*\| \;\le\; c_6\|y_k - y_*\|^2\,. \tag{22}$$

Now we show that $\nabla^2_{uu} f(\bar{y}_k)$ is also nonsingular. First we point out that $\nabla^2_{uu} f(y)$ is Lipschitz continuous on D and $\nabla^2_{uu} f(y_*)$ is positive definite. Thus, inequality (22) and the perturbation

lemma cited above, together imply the nonsingularity of $\nabla^2_{uu} f(\bar{y}_k)$. Hence the method is locally well-defined, and the second step yields

$$\|y_{k+1} - \bar{y}_k\| \;=\; \|\hat{s}_k\| \;=\; \|\nabla^2_{uu} f(\bar{y}_k)^{-1} \left(\nabla_u f(\bar{y}_k) - \nabla_u f(y_*)\right)\| \;\leq\; c_7 \|\bar{y}_k - y_*\|\,, \tag{23}$$

since $\nabla_u f(y)$ is Lipschitz continuous near y_*. Now we use inequalities (22) and (23), and write

$$\begin{aligned} \|y_{k+1} - y_*\| &\leq \|y_{k+1} - \bar{y}_k\| + \|\bar{y}_k - y_*\| \\ &\leq (c_7 + 1)\|\bar{y}_k - y_*\| \\ &\leq c_6(c_7 + 1)\|y_k - y_*\|^2\,. \end{aligned}$$

This last inequality establishes the q-quadratic rate of convergence. □

5 Applications

We begin by considering updating the slack variables in LANCELOT. Suppose the problem we are trying to solve has the form

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & c_i(x) \geq 0\,, \quad i = 1, \ldots, m\,, \end{aligned} \tag{24}$$

where $x \in \mathbb{R}^n$, and n and m are positive integers. The technique implemented in the LANCELOT package [9] is the augmented Lagrangian algorithm proposed by Conn, Gould, and Toint in [8]. For the application of the augmented Lagrangian algorithm this problem is reformulated as:

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & c_i(x) - u_i = 0\,, \quad i = 1, \ldots, m\,, \\ & u_i \geq 0\,, \quad i = 1, \ldots, m\,, \end{aligned}$$

by adding the slack variables u_i, $i = 1, \ldots, m$. This algorithm considers the following augmented Lagrangian merit function:

$$\Phi(x, u, \lambda, S, \mu) \;=\; f(x) + \sum_{i=1}^{m} \lambda_i (c_i(x) - u_i) + \frac{1}{2\mu} \sum_{i=1}^{m} s_{ii} (c_i(x) - u_i)^2\,,$$

where:

λ_i is an estimate for the Lagrange multiplier associated with the i-th constraint,

μ is a (positive) penalty parameter,

s_{ii} is a (positive) scaling factor that is associated with the i-th constraint, and

$S = [s_{ij}]$ with $s_{ij} = 0$ for $i \neq j$.

LANCELOT [7], [9] solves a sequence of minimization problems with simple bounds of the following form:

$$\begin{aligned} \text{minimize} \quad & \Phi(x, u, \lambda, S, \mu) \\ \text{subject to} \quad & u_i \geq 0\,, \quad i = 1, \ldots, m\,, \end{aligned} \tag{25}$$

for fixed values of μ, s_{ii}, and λ_i, $i = 1, \ldots, m$. The two-step trust-region framework and analysis described in this paper for unconstrained minimization problems can be extended in an entirely straightforward way to a number of algorithms for minimization problems with simple bounds, in particular to the algorithms [7] used by LANCELOT to solve problem (25).

If x is fixed, the function $\Phi(x, u, \lambda, S, \mu)$ is quadratic in the slack variables u. Let us denote this quadratic by $q(u;\, x)$:

$$q(u;\, x) \;=\; \Phi(x, u, \lambda, S, \mu) = d(x) + e(x)^\top u + \frac{1}{2} u^\top F u\,,$$

where $d(x)$ and $e(x)$ depend on x but F is constant. (The dependency on λ_i, s_{ii}, and μ is not important since these are constants fixed before the minimization process is started.)

The key idea is to update these slack variables at every iteration k of the trust-region algorithm [7] that is used in LANCELOT to solve problem (25). The trust-region algorithm computes, at the current point y_k, a first step $\bar{s}_k$. Now, at the new point $y_k + \bar{s}_k$ we compute the step $\hat{s}_k$ by updating the slack variables u. So, we have

$$y_k \;=\; \begin{pmatrix} x_k \\ u_k \end{pmatrix}, \quad \bar{s}_k \;=\; \begin{pmatrix} (\bar{s}_k)_x \\ (\bar{s}_k)_u \end{pmatrix}, \quad \hat{s}_k \;=\; \begin{pmatrix} 0 \\ \Delta u_k \end{pmatrix},$$

$$f(y_k + \bar{s}_k) \;=\; q(\bar{u}_k;\, \bar{x}_k)\,, \quad f(y_k + \bar{s}_k + \hat{s}_k) \;=\; q(\bar{u}_k + \Delta u_k;\, \bar{x}_k)\,,$$

where

$$\bar{x}_k \;=\; x_k + (\bar{s}_k)_x\,, \qquad \bar{u}_k \;=\; u_k + (\bar{s}_k)_u\,.$$

(Here f represents the objective function of Sections 1-4.) Note that the second step $\hat{s}_k$ is exclusively in the components associated with slack variables. This step is computed as $u_{k+1} - \bar{u}_k$, where u_{k+1} is the optimal solution of

$$\begin{aligned} &\text{minimize} \quad q(u;\, \bar{x}_k) \\ &\text{subject to} \quad u_i \geq 0\,, \quad i = 1, \ldots, m\,. \end{aligned} \tag{26}$$

Due to the simple form of this quadratic, the solution is explicit:

$$(u_{k+1})_i \;=\; \max\left\{0,\, \frac{\mu\lambda_i}{s_{ii}} + c_i(\bar{x}_k)\right\}, \quad i = 1, \ldots, m\,. \tag{27}$$

It is important to remark that these updates require no further function or gradient evaluations. They have also been considered in the codes NPSOL and SNOPT [15], [16] to update slack variables after the application of a line search to the augmented Lagrangian merit function and prior to the solution of the next quadratic programming problem. Other ways of dealing with slack variables have been studied in the literature (see Gould [18] and the references therein).

For the study of the impact of the slack variable update on the global convergence of the trust-region algorithm, the step in these variables is required only to decrease the quadratic $q(u;\, \bar{x}_k)$ from $\bar{u}_k$ to $\bar{u}_k + \Delta u_k$. In such a case, we can always guarantee that the decrease in the objective function is larger than $\|\hat{s}_k\|^2$, that is that (21) holds. This result is shown in the following proposition. We drop $\bar{x}_k$ from $q(\,\cdot\,;\, \bar{x}_k)$ to simplify the notation.

Proposition 5.1 *There exists a positive constant c_5 such that, whenever $q(\bar{u}_k + \Delta u_k) < q(\bar{u}_k)$, we have*

$$q(\bar{u}_k) - q(\bar{u}_k + \Delta u_k) \;\geq\; c_5 \|\Delta u_k\|^2\,.$$

Proof: First we write down a few properties of the quadratic $q(u)$. Simple algebraic manipulations lead to:

$$q(\bar{u}_k) - q(\bar{u}_k + \Delta u_k) \;=\; -\left(F(\bar{u}_k + \Delta u_k) + e(\bar{x}_k)\right)^\top \Delta u_k + \frac{1}{2}\Delta u_k{}^\top F \Delta u_k \,. \tag{28}$$

Also, since $q(u)$ is convex:

$$q(\bar{u}_k) - q(\bar{u}_k + \Delta u_k) \;\geq\; \left|\nabla q(\bar{u}_k + \Delta u_k)^\top \Delta u_k\right| \,. \tag{29}$$

Let c be a positive constant such that $c < \frac{\lambda_{min}(F)}{2}$, where $\lambda_{min}(F)$ is the smallest eigenvalue of F. Now we consider two cases.

1. $\left|\nabla q\left(\bar{u}_k + \Delta u_k\right)^\top \Delta u_k\right| \geq c\|\Delta u_k\|^2$. In this case we use (29), to obtain

$$q(\bar{u}_k) - q(\bar{u}_k + \Delta u_k) \;\geq\; c\|\Delta u_k\|^2 \,.$$

2. $\left|\nabla q\left(\bar{u}_k + \Delta u_k\right)^\top \Delta u_k\right| < c\|\Delta u_k\|^2$. In this case we appeal to (28) and

$$\nabla q\left(\bar{u}_k + \Delta u_k\right) = F(\bar{u}_k + \Delta u_k) + e(\bar{x}_k)\,,$$

to get

$$\begin{aligned} q(\bar{u}_k) - q(\bar{u}_k + \Delta u_k) &= -\left(F(\bar{u}_k + \Delta u_k) + e(\bar{x}_k)\right)^\top \Delta u_k + \frac{1}{2}\Delta u_k{}^\top F \Delta u_k \\ &\geq \left(\frac{\lambda_{min}(F)}{2} - c\right)\|\Delta u_k\|^2 \,. \end{aligned}$$

The proof is completed by setting $c_5 = \min\{c, \frac{\lambda_{min}(F)}{2} - c\}$. □

Another example of the application of two-step algorithms arises in one approach to the solution of minimax problems. Consider the following minimax problem:

$$\min_x \;\max_{i=1,\ldots,m}\; f_i(x)\,, \tag{30}$$

where each f_i is a real-valued function defined in $\mathbb{R}^n$. One way of solving this minimax problem is to reformulate it as a nonlinear programming problem by adding an artificial variable z. See [18] for more details. This leads to

$$\begin{aligned} \text{minimize} \quad & z \\ \text{subject to} \quad & z - f_i(x) - u_i = 0\,, \quad i = 1,\ldots,m\,, \\ & u_i \geq 0\,, \quad i = 1,\ldots,m\,, \end{aligned} \tag{31}$$

where the slack variables have also been introduced. If LANCELOT is used to solve this nonlinear programming problem, then the augmented Lagrangian algorithm requires the solution of a sequence of problems with simple bounds of the type:

$$\begin{aligned} \text{minimize} \quad & \Phi(x, z, u, \lambda, S, \mu) \\ \text{subject to} \quad & u_i \geq 0\,, \quad i = 1,\ldots,m\,, \end{aligned} \tag{32}$$

where

$$\Phi(x, z, u, \lambda, S, \mu) \;=\; z + \sum_{i=1}^{m} \lambda_i (z - f_i(x) - u_i) + \frac{1}{2\mu} \sum_{i=1}^{m} s_{ii} (z - f_i(x) - u_i)^2 \,.$$

In this situation the function $\Phi(x, z, u, \lambda, S, \mu)$ is quadratic in the variables u and z for fixed values of x. (Again, λ, S, and μ are constants and not variables for problem (32).) The application of the two-step trust-region algorithm follows in a similar way. The Hessian of the quadratic is positive semi-definite with the following form

$$F \;=\; \frac{1}{\mu} \begin{pmatrix} s_{11} & 0 & \cdot & \cdot & \cdot & 0 & -s_{11} \\ 0 & \cdot & & & & & \cdot \\ \cdot & & \cdot & & & & \cdot \\ \cdot & & & \cdot & & & \cdot \\ \cdot & & & & \cdot & & \cdot \\ 0 & & & & & s_{nn} & -s_{nn} \\ -s_{11} & \cdot & \cdot & \cdot & \cdot & -s_{nn} & \sum_{i=1}^{m} s_{ii} \end{pmatrix},$$

where the last row and the last column correspond to the variable z. The solution of the quadratic program

$$\begin{aligned} &\text{minimize} \quad q(z, u;\, \bar{x}_k) \\ &\text{subject to} \quad u_i \geq 0\,, \quad i = 1, \ldots, m\,. \end{aligned} \tag{33}$$

is given by

$$(u_{k+1})_i \;=\; \max\left\{0,\, \frac{\mu\lambda_i}{s_{ii}} - f_i(\bar{x}_k) + z_{k+1}\right\}, \quad i = 1, \ldots, m\,, \tag{34}$$

where z_{k+1} is the solution of the equation

$$-\frac{1}{\mu} \sum_{i=1}^{m} s_{ii} \max\left\{0,\, \frac{\mu\lambda_i}{s_{ii}} - f_i(\bar{x}_k) + z\right\} \;+\; \frac{1}{\mu} \left(\sum_{i=1}^{m} s_{ii}\right) z \;=\; b \tag{35}$$

with right hand side

$$b = -1 - \sum_{i=1}^{m} \left(\lambda_i - \frac{s_{ii}}{\mu} f_i(\bar{x}_k)\right). \tag{36}$$

The equation (35) is solved easily with $\mathcal{O}(m)$ floating point operations and comparisons, showing that the solution of the quadratic program (33) is a relatively inexpensive calculation.

There are several nonlinear optimization problems in which some subset of the problem variables occur linearly, for example, arrival times in static-timing-based circuit optimization problems [6]. Such problems can also benefit from two-step updating.

6 Numerical tests

6.1 Analytic problems

We modified LANCELOT (Release A) [9] to include the slack variable update (27) and the slack and minimax variable updates (34)-(36). These updates were incorporated in LANCELOT using a greedy two-step modification of the trust-region algorithm [7] for minimization problems with

simple bounds that is implemented in the subroutine `SBMIN`. (The greedy two-step trust-region algorithm for unconstrained minimization problems is Algorithm 2.2.) We tested the following versions of LANCELOT:

1. LANCELOT (Release A) with the default parameter configuration `SPEC.SPC` file, except that we increased the maximum number of iterations to 4000.
2. Version 1 with the slack and minimax variable updates (27) and (34)-(36) incorporated in `SBMIN` using a greedy two-step trust-region algorithm.
3. The same as Version 2 but with no update of the variable z for minimax problems, i.e., z fixed in (34)-(36).

We compared the numerical performance of these three versions on a set of problems[1] from the CUTE collection [2]. This set of problems is listed in Table 1, and in the case of minimax formulations in Table 2, where we mention the number of variables (including slacks and, where applicable, the minimax variable z), the number of slack variables, and the number of equality and inequality constraints (excluding simple bounds on the variables). Note that the minimax problems were reformulated as nonlinear programming problems by the introduction of an additional minimax variable z as shown above (31).

The computational results are presented in Tables 3, 4, and 5. All tests were conducted on an IBM Risc/System 6000 model 390 workstation. In Table 3 we compare the results of Versions 1 and 2 for problems that are not minimax problems. In Table 4 we present the results of Versions 1 and 2 for minimax problems. In Table 5 we include the results of Versions 1 and 3 for minimax problems. In Tables 4 and 5 we include the majority of the minimax problems but not all (see Section 6.3 for numerical results on the remaining problems). In these tables we report the value of the flag `INFORM`, the number of iterations, the total CPU time, and the determined values (a single value if they are both the same) of the objective function. The values of `INFORM` have the following meaning:

`INFORM = 0` for normal return, meaning that the norm of the projected gradient of the augmented Lagrangian function has become smaller than 10^{-5}.

`INFORM = 1` for cases where the maximum number of iterations (4000) has been reached.

`INFORM = 3` for cases where the norm of the step has become too small.

Our conclusion based on these sets of problems is that the version with the slack and minimax variable updates exhibits superior numerical behavior. In fact, this version required an average of 15% fewer iterations than the version without these updates (the problems `HS109`, `HAIFAM`, and `POLAK6` were excluded from this calculation, mainly because the comparison was extraordinarily favorable in the case of the first two and worse in the last). Comparing Tables 4 and 5, updating the minimax variable z in addition to two-step updates on just the slacks is seen to yield a significant benefit. However, there are some minimax problems where the two-step algorithm performs poorly and this situation is analyzed in detail in Section 6.3.

[1] Although CUTE contains more than 56 problems with general constraints the majority of these are equality constrained problems. We excluded all problems that took more than 4000 iterations with both Versions 1 and 2. We included the rest, with the exception of some problems that are too easy, making a total of 56 problems of which 30 are minimax problems and 26 are non-minimax problems.

Problem Name	Variables	Slacks	Constraints
CAR2	209	30	146
CORE1	83	18	59
CORE2	157	26	134
CORKSCRW	106	70	10
CSFI1	7	2	4
CSFI2	7	2	4
HADAMARD	769	512	648
HS32	4	1	1
HS67	17	14	14
HS85	26	21	21
HS109	13	8	4
NET1	67	19	57
NET2	181	37	160
ORBIT2	298	30	207
PRODPL0	69	9	29
PRODPL1	69	9	29
SSEBNLN	218	24	96
SWOPF	97	14	92
TFI1F	3	101	101
TFI2F	3	101	101
TFI3F	3	101	101
VANDERM1	10	9	19
VANDERM2	10	9	19
VANDERM3	10	9	19
VANDERM4	5	4	9
ZIGZAG	74	10	50

Table 1: Non-minimax problems from the CUTE collection that were used.

6.2 Circuit optimization problems

We have built extensive experience with circuit optimization problems, where – due to expensive function evaluations, modest numerical noise levels, and practical stopping criteria – the implementation is designed to terminate before many "asymptotic" iterations are taken. The algorithms described in this paper have been used in a dynamic-simulation-based circuit optimization tool called JiffyTune (see [4], [5], and [10]). JiffyTune optimizes transistor and wire sizes of digital integrated circuits to meet delay, power, and area goals. It is based on fast circuit simulation and time-domain sensitivity computation in SPECS (see [13] and [28]). To optimize multiple path delays through a high-performance circuit, the tuning is often formulated as a minimax problem or a minimization problem with nonlinear inequality constraints.

We remark that many of the analytic problems (especially the minimax problems) are rather small and involve inexpensive function evaluations. Moreover, it is clear that two-step updating is unlikely to be helpful asymptotically in these situations. Consequently we also report numerical results with circuit optimization problems which are indicative of problems with expensive function evaluations, where termination (because of inherent noise and practical considerations) is encouraged to be before any significant asymptotic behavior. The numerical results are presented in Table 6. As in Version 1, the second step consisted of the slack and minimax variable updates (27) and (34)-(36). However the gradient and constraint tolerances used were 10^{-3} and 10^{-5}, respectively,

Problem Name	**Variables**	**Slacks**	**Constraints**
CB2	6	3	3
CB3	6	3	3
CHACONN1	6	3	3
CHACONN2	6	3	3
CONGIGMZ	8	5	5
COSHFUN	81	20	20
DEMYMALO	6	3	3
GIGOMEZ1	6	3	3
GOFFIN	101	50	50
HAIFAL	9301	8958	8958
HAIFAM	249	150	150
HALDMADS	48	42	42
KIWCRESC	5	2	2
MADSEN	9	6	6
MAKELA1	5	2	2
MAKELA2	6	3	3
MAKELA3	41	20	20
MAKELA4	61	40	40
MIFFLIN1	5	2	2
MIFFLIN2	5	2	2
MINMAXBD	25	20	20
POLAK1	5	2	2
POLAK2	13	2	2
POLAK3	22	10	10
POLAK4	6	3	3
POLAK5	5	2	2
POLAK6	9	4	4
SPIRAL	5	2	2
SPRALX	5	2	2
WOMFLET	6	3	3

Table 2: Minimax problems from the CUTE collection that were used.

with some safeguards related to an expected level of numerical noise. We can clearly observe from Table 6 that the two-step algorithm leads to better final objective function values. In practical applications where a simple function evaluation takes more than ten minutes of CPU time the effectiveness of such a simple addition is indeed significant. (There are situations where the greedy two-step trust-region algorithm is able to take advantage of the decrease given by the slack and minimax variable updates and, by doing so, this algorithm can accept steps that otherwise would have been rejected, see Remark 3.1.)

We also applied the algorithms of this paper to analytic static-timing-based circuit optimization problems (see Table 7), where it is clear that the advantage of the two-step approach is increasingly apparent for larger problems.

Problem Name	Inform	Iterations	Total CPU	Obj. Function
CAR2	0/0	80/67	15.2/12.3	2.67
CORE1	0/0	953/983	7.41/17	91.1
CORE2	0/0	1048/1086	25.6/25.7	72.9
CORKSCRW	0/0	41/42	0.55/0.54	1.16
CSFI1	0/0	112/127	0.11/0.11	-49.1
CSFI2	0/0	78/83	0.07/0.07	55
HADAMARD	0/0	1709/548	2290/276	1.14/1
HS32	0/0	5/5	0.01/0.01	1
HS67	0/0	33/21	0.08/0.07	-1.16e+03
HS85	1/0	4000/3734	27.1/23.6	-1.85/-2.22
HS109	3/3	1578/753	7.58/3.11	5.36e+03
NET1	3/0	69/60	0.57/0.54	9.41e+05
NET2	3/0	95/69	3.53/2.92	1.19e+06
ORBIT2	0/3	615/612	3020/2750	312
PRODPL0	3/0	36/26	0.29/0.23	58.8
PRODPL1	0/0	56/32	0.55/0.51	35.7
SSEBNLN	0/0	51/51	1.46/1.47	1e+12
SWOPF	0/0	204/136	7.68/5.51	0.0679
TFI1	0/0	26/24	0.4/0.25	5.33
TFI2	0/0	25/45	0.33/0.41	0.649
TFI3	0/0	23/34	0.38/0.38	4.3
VANDERM1	0/0	13/13	0.05/0.08	0
VANDERM2	0/0	13/13	0.08/0.07	0
VANDERM3	0/0	14/16	0.07/0.08	0
VANDERM4	0/0	81/82	0.1/0.1	0
ZIGZAG	0/0	35/31	0.54/0.43	1.8

Table 3: Comparison between Versions 1 and 2 for non-minimax problems (LANCELOT without/with two-step updating).

6.3 Further experiments with minimax problems

In this section we consider those minimax problems in our test set for which the two-step algorithm not only does not improve numerically the results obtained in the one-step case, but also makes them considerably worse (see the first part of Table 8). We analyze the reasons for the failure of the two-step updating on some minimax problems and discuss a few ways to enforce better numerical behavior.

We consider the general minimax problem (30). Our aim is to show that for some types of minimax problems the second step has a tendency to make the Hessian of Φ ill-conditioned. Let us assume that $\lambda_i = 0$ and $s_{ii} = 1$, for all $i = 1, \ldots, m$ (as happens by default for the first LANCELOT major iteration). Under these circumstances, we have:

$$\Phi(x, z, u, \mu) \;=\; z + \frac{1}{2\mu} \sum_{i=1}^{m} (z - f_i(x) - u_i)^2.$$

By using the notation $g_i(x, z, u) = z - f_i(x) - u_i$, we have the following expressions for the elements

Problem Name	Inform	Iterations	Total CPU	Obj. Function
CB2	0/0	17/11	0.03/0.01	1.95
CB3	0/0	14/10	0.05/0.02	2
CHACONN1	0/0	12/8	0.02/0.04	1.95
CHACONN2	0/0	13/10	0.01/0.02	2
CONGIGMZ	0/0	32/19	0.04/0.05	28
COSHFUN	0/0	127/69	1.31/1.06	-0.773
DEMYMALO	0/0	24/17	0.03/0.03	-3
GIGOMEZ1	0/0	27/19	0.04/0.02	-3
GOFFIN	0/0	14/4	1.03/0.67	0
HAIFAM	1/0	4000/136	1140/85.1	-45
HALDMADS	0/0	48/73	0.49/0.72	0.0001
KIWCRESC	0/0	19/14	0.02/0.02	0
MADSEN	0/0	29/18	0.05/0.04	0.616
MAKELA1	0/0	17/18	0.04/0.02	-1.41
MAKELA2	0/0	21/9	0.05/0	7.2
MAKELA4	0/0	6/4	0.09/0.08	0
MIFFLIN1	0/0	11/7	0.03/0.01	-1
MIFFLIN2	0/0	37/32	0.04/0.05	-1
POLAK1	0/0	35/19	0.04/0.02	2.72
POLAK2	0/0	40/24	0.09/0.07	54.6
POLAK5	0/0	28/20	0.07/0.04	50
POLAK6	0/0	124/149	0.24/0.23	-44
SPIRAL	0/0	85/93	0.1/0.07	0
SPRALX	0/0	87/93	0.13/0.08	0

Table 4: Comparison between Versions 1 and 2 for minimax problems (LANCELOT without/with two-step updating).

of the gradient of Φ:

$$\nabla_{x_j}\Phi = -\frac{1}{\mu}\sum_{i=1}^{m}\nabla_{x_j}f_i(x)g_i(x,z,u)\,, \quad j=1,\ldots,n\,,$$
$$\nabla_z\Phi = 1+\frac{1}{\mu}\sum_{i=1}^{m}g_i(x,z,u),$$
$$\nabla_{u_i}\Phi = -\frac{1}{\mu}g_i(x,z,u)\,,\ i=1,\ \ldots,m\,.$$

Similarly the elements of the Hessian matrix of Φ are given by:

$$\nabla^2_{x_jx_k}\Phi = -\tfrac{1}{\mu}\textstyle\sum_{i=1}^{m}[\nabla^2_{x_jx_k}f_i(x)g_i(x,z,u) - \nabla_{x_j}f_i(x)\nabla_{x_k}f_i(x)]\,, \qquad \nabla^2_{zz}\Phi = \tfrac{m}{\mu}\,,$$
$$\nabla^2_{u_iu_l}\Phi = \tfrac{\delta_{il}}{\mu}\,, \qquad \nabla^2_{zu_i}\Phi = -\tfrac{1}{\mu}\,,$$
$$\nabla^2_{x_jz}\Phi = -\tfrac{1}{\mu}\textstyle\sum_{i=1}^{m}\nabla_{x_j}f_i(x)\,, \qquad \nabla^2_{u_ix_j}\Phi = \tfrac{1}{\mu}\nabla_{x_j}f_i(x)\,,$$

for $i,l = 1,\ldots,m$ and $j,k = 1,\ldots,n$. If the magnitudes of the products $\nabla^2_{x_jx_k}f_i(x)g_i(x,z,u)$ are small compared to those of the products $\nabla_{x_j}f_i(x)\nabla_{x_k}f_i(x)$, then the Hessian of Φ is given

Problem Name	Inform	Iterations	Total CPU	Obj. Function
CB2	0/0	17/17	0.03/0.03	1.95
CB3	0/0	14/16	0.05/0.03	2
CHACONN1	0/0	12/10	0.02/0.03	1.95
CHACONN2	0/0	13/13	0.01/0.04	2
CONGIGMZ	0/0	32/25	0.04/0.1	28
COSHFUN	0/0	127/92	1.31/1.08	-0.773
DEMYMALO	0/0	24/18	0.03/0.03	-3
GIGOMEZ1	0/0	27/20	0.04/0.02	-3
GOFFIN	0/0	14/8	1.03/0.66	0
HAIFAM	1/3	4000/609	1140/76.7	-45
HALDMADS	0/0	48/46	0.49/0.54	0.0001
KIWCRESC	0/0	19/18	0.02/0.03	0
MADSEN	0/0	29/23	0.05/0.05	0.616
MAKELA1	0/0	17/19	0.04/0.02	-1.41
MAKELA2	0/0	21/24	0.05/0.03	7.2
MAKELA4	0/0	6/6	0.09/0.11	0
MIFFLIN1	0/0	11/11	0.03/0.03	-1
MIFFLIN2	0/0	37/37	0.04/0.03	-1
POLAK1	0/0	35/32	0.04/0.06	2.72
POLAK2	0/0	40/15	0.09/0.04	54.6
POLAK5	0/0	28/28	0.07/0.01	50
POLAK6	0/0	124/332	0.24/0.48	-44
SPIRAL	0/0	85/85	0.1/0.07	0
SPRALX	0/0	87/87	0.13/0.09	0

Table 5: Comparison of Versions 1 and 3 for minimax problems (LANCELOT without/with two-step updating only on slacks).

approximately by

$$\frac{1}{\mu}\begin{pmatrix} \sum_i a_{i1}a_{i1} & \dots & \sum_i a_{i1}a_{in} & -\sum_i a_{i1} & a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum_i a_{in}a_{i1} & \dots & \sum_i a_{in}a_{in} & -\sum_i a_{in} & a_{1n} & \dots & a_{mn} \\ -\sum_i a_{i1} & \dots & -\sum_i a_{in} & m & -1 & \dots & -1 \\ a_{11} & \dots & a_{1n} & -1 & 1 & & \\ \vdots & \ddots & \vdots & \vdots & & \ddots & \\ a_{m1} & \dots & a_{mn} & -1 & & & 1 \end{pmatrix},$$

where a_{ij} denotes $\nabla_{x_j} f_i(x)$ and the indices i in the sums go from 1 to m. This matrix is clearly singular. In fact, the $n+1$-st row is the negative sum of the last m rows. Moreover, any of the first n rows is a linear combination of the last m rows. As result of these observations, the Hessian (and the projected Hessian) of Φ is ill-conditioned if

$$\left|\frac{1}{\mu}\sum_{i=1}^{m} \nabla_{x_j} f_i(x)\nabla_{x_k} f_i(x)\right| \gg \left|\frac{1}{\mu}\sum_{i=1}^{m} \nabla^2_{x_j x_k} f_i(x) g_i(x,z,u)\right| \tag{37}$$

happens for "many" indices j and k. This is the key point in this analysis: the second step has a tendency to produce iterates that worsen property (37) because it produces a decrease on the

Problem Name	**Variables**	**Ineq.**	**Iterations**	**Total CPU**	**Obj. Function**
Non-minimax:					
IOmuxpower	102	42	21/29	7230/9220	-15100/-16000
durham2	13	2	17/17	93.5/93.5	472
chen2	2	1	14/14	91/91.2	4290
IOmux	101	41	60/61	18000/17700	-16200/-15900
Nov01power	5	1	37/54	24.5/35.6	273/268
lau2	5	1	33/32	47.9/46.3	158
Nov01	8	4	29/33	22.1/27.3	193/181
coulman_cold	33	17	22/22	69.5/68.3	271/262
clkgen	22	5	25/5	35/10.8	1.98/1.82
coulman_hot	33	17	16/32	46.2/100	283/253
davies3	16	1	30/30	368/368	254
coulman_delay	33	17	26/24	72.6/73.5	116/111
Minimax:					
bultmann_latch	39	13	17/18	41.8/46.8	95.9/84.6
stall1	30	5	23/19	3350/3050	156/86.8
coulman_cold_minmax	34	17	61/80	184/229	69.4/66.9
coulman_hot_minmax	34	17	66/44	197/134	74.4/75.1
fleischer	110	5	53/61	267/330	-458/-505
mod5	51	10	17/51	11200/33100	98.9/19
northrop_xor	18	8	67/64	78.3/77.7	-34.1/-30.2
coulman_delay_minmax	34	17	100/100	290/306	67.4/70.5

Table 6: LANCELOT without/with two-step updating for dynamic-simulation-based circuit optimization problems. **Ineq.** stands for the number of inequality constraints.

values of $g_i(x, z, u)$ for some indices i. The Hessian of Φ might very well be ill-conditioned if no second steps are applied, but there is no doubt (and the numerical results are a evidence of this claim) that the second step for some problems worsens the situation by making the Hessian of Φ more ill-conditioned.

In the presence of nonzero Lagrange multipliers λ_i, $i = 1, \ldots, m$, the formulae for the gradient and Hessian of Φ are the same with $g_i(x, z, u)$ substituted by $g_i(x, z, u) + \mu\lambda_i$ and similar conclusions could be drawn.

The second step may produce very bad results on some minimax problems because it points towards the set $\{(x, z, u) : g_i(x, z, u) = 0, \text{ for some } i\}$ (where the Hessian of the augmented Lagrangian is ill-conditioned) and this effect influences negatively the calculation of the first step at the next iteration. Given this undesirable feature of the Hessian of Φ at points close to this set, one possible improvement to the two-step algorithm is to make sure that the calculation of the first step is accurate (in the LANCELOT context this could be achieved by choosing a smaller tolerance for the stopping criterion of the conjugate-gradient technique). Another possible improvement is to reduce the ill-conditioning of the Hessian of Φ (for instance by increasing the value of the penalty parameter μ as can be seen in examples with a few variables). Indeed, these modifications improve the bad numerical results presented before: in the second part of Table 8 we compare the results obtained by the following modifications of Versions 1 and 2:

4. Version 1 with an initial value for the penalty parameter μ of 100 (the default value is 0.1).

5. Version 2 with an initial value for the penalty parameter μ of 100 and a tolerance of 10^{-12} in

Problem Name	Variables	Ineq.	Iterations	Total CPU	Obj. Function
Symmetric 3	37	2^4-1	39/40	0.12/0.15	7.7
Symmetric 4	77	2^5-1	69/60	0.63/0.6	10.2
Symmetric 5	157	2^6-1	97/81	2.09/1.64	12.7
Symmetric 6	317	2^7-1	140/118	9.38/7.14	15.2
Symmetric 7	637	2^8-1	270/183	44.3/35.3	17.6
Symmetric 8	1277	2^9-1	385/340	247/221	19.9
Symmetric 9	2557	$2^{10}-1$	901/639	1920/1300	22.1
Nonsymmetric 3	37	2^4-1	44/27	0.18/0.16	12.4
Nonsymmetric 4	77	2^5-1	58/37	0.57/0.31	16
Nonsymmetric 5	157	2^6-1	78/45	1.84/0.91	19.7
Nonsymmetric 6	317	2^7-1	75/54	5.89/3.3	23.6
Nonsymmetric 7	637	2^8-1	96/50	30.9/9.02	27.7
Nonsymmetric 8	1277	2^9-1	92/53	63.6/31.6	31.7
Nonsymmetric 9	2557	$2^{10}-1$	130/63	300/95	35.7

Table 7: LANCELOT without/with two-step updating for analytic (minimax) static-timing-based circuit optimization problems. **Ineq.** stands for the number of inequality constraints.

the stopping criterion for conjugate gradients.

The study of strategies that can make two-step updating more effective for minimax problems in general is the subject for future research.

7 Concluding remarks

In this paper we presented and analyzed a framework under which classical algorithms for nonlinear optimization can be modified to allow second computationally efficient steps that are not generated in the conventional way but that are guaranteed to yield decrease in the objective function. We gave as examples of the two-step algorithms the update of slack variables in LANCELOT, and the update of variables introduced to solve minimax problems. However, we emphasize that the two-step algorithms can be very generally applied, for example, whenever the functions defining the problem are in a known functional form in some of the variables.

We considered trust-region algorithms for which we proposed a greedy and a conservative two-step algorithm. We analyzed the convergence properties of the trust-region two-step algorithms (see [11] for line-search two-step algorithms), deriving the conditions under which they attain global convergence. We also showed that a two-step Newton's method (for which the second step is computed only for a subset of the variables) has a q-quadratic rate of convergence.

The greedy two-step algorithms are designed to exploit as much as possible the decrease attained by the second step. The trust-region framework allowed to us to design a greedy two-step trust-region algorithm that is particularly well tailored to achieve this purpose.

Finally, we included numerical evidence that this technique is effective, particularly for problems with expensive function evaluations. The two-step algorithms have already found practical applications in optimization of high-performance custom microprocessor integrated circuits.

Problem Name	Inform	Iterations	Total CPU	Obj. Function
HAIFAL	0/1	679/4000	872346.06/366146.81	-12.8/-12.7828
MAKELA3	0/0	66/2816	0.26/5.84	0
MINMAXBD	0/0	267/952	1.34/3.59	116
POLAK3	0/0	71/125	0.4/0.8	5.93
POLAK4	3/1	14/4000	0.04/3.23	0
WOMFLET	0/0	63/150	0.07/0.13	0
HAIFAL	0/0	287/41	61603.1/8480.99	-12.8
MAKELA3	0/0	20/48	0.09/0.22	0
MINMAXBD	0/0	47/43	0.25/0.22	116
POLAK3	0/0	44/14	0.22/0.18	5.93
POLAK4	3/3	31/15	0.04/0.04	0
WOMFLET	0/0	26/32	0.03/0.04	6.05/0

Table 8: In the first part, comparison of Versions 1 and 2 for minimax problems (LANCELOT without/with two-step updating). In the second part, comparison of Versions 4 and 5 for minimax problems (LANCELOT without/with two-step updating).

8 Acknowledgments

We are grateful to N. I. M. Gould (Rutherford Appleton Laboratory) for his comments and suggestions on an earlier version of this paper that led to many improvements. We are also grateful to K. Scheinberg (IBM T. J. Watson Research Center) for helping with the numerical results and explanation in Section 6.3. We would like to thank I. M. Elfadel (IBM T. J. Watson Research Center) for providing the analytic static-timing-based optimization circuit problems. Finally, we are grateful to the referees for their useful comments and suggestions.

References

[1] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, Computer Science and Applied Mathematics, Academic Press, New York, 1982.

[2] I. Bongartz, A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *CUTE: Constrained and Unconstrained Testing Environment*, ACM Trans. Math. Software, 21 (1995), pp. 123-160.

[3] R. H. Byrd, R. B. Schnabel, and G. A. Shultz, *Approximate solution of the trust-region problem by minimization over two-dimensional subspaces*, Math. Programming, 40 (1988), pp. 247-263.

[4] A. R. Conn, P. K. Coulman, R. A. Haring, G. L. Morrill, and C. Visweswariah, *Optimization of custom MOS circuits by transistor sizing*, IEEE International Conference on Computer-Aided Design (ICCAD), (1996).

[5] A. R. Conn, P. K. Coulman, R. A. Haring, G. L. Morrill, C. Visweswariah, and C. W. Wu, *JiffyTune: circuit optimization using time-domain sensitivities*, IEEE Transactions on Computer-Aided Design of ICs and Systems, vol. 17, num. 12, (December 1998), pp. 1292–1309.

[6] A. R. Conn, I. M. Elfadel, W. W. Molzen, Jr., P. R. O'Brien, P. N. Strenski, C. Visweswariah, and C. B. Whan, *Gradient-based optimization of custom circuits using a static-timing formulation*, Proc. 1999 Design Automation Conference, (June 1999).

[7] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *Global convergence of a class of trust-region algorithms for optimization problems with simple bounds*, SIAM J. Numer. Anal., 25 (1988), pp. 433-460.

[8] ——, *A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds*, SIAM J. Numer. Anal., 28 (1991), pp. 545-572.

[9] ——, *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*, Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1992.

[10] A. R. Conn, R. A. Haring, C. Visweswariah, and C. W. Wu, *Circuit optimization via adjoint Lagrangians*, ICCAD, (1997), pp. 281-288.

[11] A. R. Conn, L. N. Vicente, and C. Visweswariah, *Two-step algorithms for nonlinear optimization with structured applications*, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY 10598, Research Report RC 21198(94689), June, 1998.

[12] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[13] P. Feldmann, T. V. Nguyen, S. W. Director, and R. A. Rohrer, *Sensitivity computation in piecewise approximate circuit simulation*, IEEE Trans. on CAD of ICs and Systems, (1991), pp. 171-183.

[14] R. Fletcher, *Practical Methods of Optimization*, John Wiley & Sons, Chichester, second ed., 1987.

[15] P. E. Gill, W. Murray, and M. A. Saunders, *SNOPT: An SQP algorithm for large-scale constrained optimization*, Department of Mathematics, University of California, San Diego, Report NA 97-2, 1997.

[16] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, *User's guide for NPSOL 5.0: A Fortran package for nonlinear programming*, System Optimization Laboratory, Stanford University, Technical Report SOL 86-1, Revised July 30, 1998.

[17] G. H. Golub and V. Pereyra, *The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate*, SIAM J. Numer. Anal., 10 (1973), pp. 413-432.

[18] N. I. M. Gould, *On solving three classes of nonlinear programming problems via simple differentiable penalty functions*, J. Optim. Theory Appl., 56 (1988), pp. 89-126.

[19] D. G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley Publishing Company, Massachusetts, 1989.

[20] J. J. Moré, *Recent developments in algorithms and software for trust-regions methods*, in Mathematical programming. The state of art, A. Bachem, M. Grotschel, and B. Korte, eds., Springer Verlag, New York, (1983), pp. 258-287.

[21] ——, *Generalizations of the trust-region problem*, Optimization Methods and Software, 2 (1993), pp. 189-209.

[22] J. J. Moré and D. C. Sorensen, *Computing a trust-region step*, SIAM J. Sci. Statist. Comput., 4 (1983), pp. 553-572.

[23] M. J. D. Powell, *A new algorithm for unconstrained optimization*, in Nonlinear Programming, J. B. Rosen, O. L. Mangasarian, and K. Ritter, eds., Academic Press, New York, (1970), pp. 31-66.

[24] ——, *Convergence properties of a class of minimization algorithms*, in Nonlinear Programming 2, O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, eds., Academic Press, New York, (1975), pp. 1-27.

[25] D. C. Sorensen, *Minimization of a large-scale quadratic function subject to a spherical constraint*, SIAM J. Optim., 7 (1997) 141-161.

[26] T. Steihaug, *The conjugate gradient method and trust regions in large scale optimization*, SIAM J. Numer. Anal., 20 (1983), pp. 626-637.

[27] S. W. Thomas, *Sequential Estimation Techniques for Quasi-Newton Algorithms*, PhD thesis, Cornell University, Ithaca, New York, 1975.

[28] C. Visweswariah and R. A. Rohrer, *Piecewise approximate circuit simulation*, IEEE Trans. on CAD of ICs and Systems, (1991), pp. 861-870.