

# TRUNCATED NEWTON METHODS FOR OPTIMIZATION WITH INACCURATE FUNCTIONS AND GRADIENTS \*

C.T. KELLEY<sup>†</sup> AND E.W. SACHS<sup>‡</sup>

**Abstract.** We consider unconstrained minimization problems that have functions and gradients given by “black box” codes with error control. We discuss several modifications of the Steihaug truncated Newton method that can improve performance for such problems. We illustrate the ideas with two examples.

**Key words.** Trust region methods, inexact Newton methods, optimal control

**AMS subject classifications.** 49K20, 65F10, 49M15, 65J15, 65K10

**1. Introduction.** Consider an unconstrained minimization problem

$$\min f$$

where the objective function  $f$  and its gradient  $\nabla f$  are computed inaccurately with absolute errors  $\tau_{af}, \tau_{ag}$  and relative errors  $\tau_{rf}, \tau_{rg}$  that can be controlled. We ask how the errors should be set so that a truncated or inexact Newton iteration [10], [5], [11], [12] such as Newton-CG or the CG-trust region method from [15] will perform like the error-free algorithm while  $\|\nabla f\| \gg \tau_{ag}$  and continue to produce an improving sequence of iterations until  $\|\nabla f\| = O(\tau_{ag})$ . The case considered here is different from that considered in [3] and [4], which assumed fully accurate function values and gradient errors that were  $O(\|\nabla f\|)$ , a condition that is impractical when  $\|\nabla f\|$  is small, and used information on the iteration to change the accuracy to which  $f$  and  $\nabla f$  were computed.

**1.1. Motivating Problem.** An example of such a situation which motivates this work is the simple optimal control problem:

$$(1.1) \quad \min_u f$$

where

$$(1.2) \quad f(u) = \int_0^T L(y(t), u(t), t) dt,$$

where  $u \in L^\infty[0, T]$  is the control and the state variable  $y$  is the solution of the initial value problem (with  $\dot{y} = dy/dt$ )

$$(1.3) \quad \dot{y}(t) = \phi(y(t), u(t), t), y(0) = y_0.$$

If  $L$  and  $\phi$  are continuously differentiable, then the gradient with respect to the  $L^2$  inner product,  $\nabla f(u)$ , can be represented as a continuous function of  $t$ :

$$(1.4) \quad \nabla f(u)(t) = p(t)\phi_u(y(t), u(t), t) + L_u(y(t), u(t), t).$$

In (1.4)  $p$ , the adjoint variable, satisfies the final-value problem on  $[0, T]$

$$(1.5) \quad -\dot{p}(t) = p(t)\phi_y(y(t), u(t), t) + L_y(y(t), u(t), t), p(T) = 0.$$

---

\*Version of July 16, 1999.

<sup>†</sup> North Carolina State University, Department of Mathematics and Center for Research in Scientific Computation, Box 8205, Raleigh, N. C. 27695-8205 (Tim.Kelley@ncsu.edu). The research of this author was supported by National Science Foundation grants #DMS-9700569 and #DMS-9714811.

<sup>‡</sup>Universität Trier, FB IV – Mathematik and Graduiertenkolleg Mathematische Optimierung, 54296 Trier, Germany (sachs@uni-trier.de).

If one solves (1.3) and (1.5) with the explicit Euler method, then the discretized gradient is also the gradient of the discrete problem. This means that approximating Hessian-vector products to high accuracy can easily be done by differencing for the discrete problem, since analytic gradients are available by computation of the discrete adjoint state. This is not the case if higher order methods are used [8]. If one uses variable-step and variable-order codes that control the local truncation error [13], [2], [1], [14] the error in  $\nabla f$  will depend on the errors that come from the numerical integration of (1.3) and (1.5). Moreover, after (1.3) has been solved, the values of  $y$  obtained will have to be used in an interpolation during the integration of (1.3). That interpolation error will also affect the accuracy of  $\nabla f$ .

The accuracy in  $\nabla f$ , in turn, will affect the performance of a Newton-CG algorithm that uses finite difference Hessian-vector products. We will denote by  $\tau_{af}$  and  $\tau_{ag}$  the absolute errors in the computation of the function and gradients and by  $\tau_{rf}$  and  $\tau_{rg}$  the relative errors. Our scenario is that when a function value  $f(u)$  is requested the computed value  $f^c(u)$  satisfies

$$(1.6) \quad |f^c(u) - f(u)| \leq \tau_{rf}|f(u)| + \tau_{af}$$

and the computed gradient, which we denote by  $g$ , satisfies

$$(1.7) \quad \|g(u) - \nabla f(u)\| \leq \tau_{rg}|\nabla f(u)| + \tau_{ag}.$$

For the example problem, the errors in  $f$  are of the same order as the errors in  $y$ . Hence  $\tau_{af} = O(\tau_{ay})$  and  $\tau_{rf} = O(\tau_{ry})$ . The gradient errors are different. If  $\tau_{ap}$  and  $\tau_{rp}$  are the absolute and relative errors in the computation of  $p$  then, neglecting products of errors, the computed gradient  $g$  is

$$\begin{aligned} & (p(1 + O(\tau_{rp})) + O(\tau_{ap}))\phi_u(y(1 + O(\tau_{ry})) + O(\tau_{ay}), u, t) \\ & + L_u(y(1 + O(\tau_{ry})) + O(\tau_{ay}), u, t). \end{aligned}$$

Assuming that  $y$ ,  $u$ , and  $p$  are bounded and  $f$  and  $L$  are sufficiently smooth we have

$$\begin{aligned} & (p(1 + O(\tau_{rp})) + O(\tau_{ap}))\phi_u(y(1 + O(\tau_{ry})) + O(\tau_{ay}), u, t) \\ & = (1 + O(\tau_{rp}))p\phi_u(y, u, t) + O(\tau_{ap} + \tau_{ry} + \tau_{ay}) \end{aligned}$$

and

$$L_u(y(1 + O(\tau_{ry})) + O(\tau_{ay}), u, t) = L_u(y, u, t) + O(\tau_{ry} + \tau_{ay}).$$

So the relative error in the  $L_u$  term can be taken to be zero. Hence,

$$g(u) = \nabla f(u)(1 + O(\tau_{rg})) + O(\tau_{ag}),$$

where

$$(1.8) \quad \tau_{rg} = O(\tau_{rp}) \text{ and } \tau_{ag} = O(\tau_{ap} + \tau_{ay} + \tau_{ry}).$$

In addition to the errors that are controlled by the integrator, interpolation errors in  $u$  and  $y$  and integration errors in the computation of  $f$  must be considered. These errors are independent of the choice of integrator. We illustrate these errors with a simple example. Let the discrete unknown be a vector  $U \in R^N$  with components  $U_i$ , which represents the values of  $u$  on a uniform temporal mesh  $\{t_i\}_{i=1}^N$  with mesh width  $h = 1/(N - 1)$ . When the numerical integrator needs values of  $u$  at points other than one of the  $t_i$ s, and interpolation needs to be done. After the state equation has been solved, the values of the solution  $x$  at  $\{t_i\}$  are stored in a vector  $X \in R^N$  and then a  $f(u)$  is approximated by a numerical integration. If  $U$  is smooth and a cubic spline interpolation is used, one may approximate  $f$  with Simpson's rule for an integration error of

$O(h^4)$ , which is also the interpolation error. If piecewise linear interpolation is used and  $f$  is approximated by the trapezoid rule, the integration and interpolation errors are  $O(h^2)$ .  $U$  and  $X$  must be interpolated for the integration of the adjoint equation. So if  $\tau_{int}$  is the integration error and  $\tau_{intrp}$  the interpolation error, we have

$$\tau_{rg} = O(\tau_{rp} + \tau_{intrp} + \tau_{int}) \text{ and } \tau_{ag} = O(\tau_{ap} + \tau_{ay} + \tau_{ry} + \tau_{intrp} + \tau_{int})$$

In the experiments reported in this paper, the values of  $\tau$  are consistent with the interpolation and integration errors. So

$$\tau_{rp} = \tau_{ap} = \tau_{ay} = \tau_{ry} = O(h^r)$$

where  $r = 2$  if piecewise linear interpolation and trapezoid rule integration is used and  $r = 4$  if cubic spline interpolation and Simpson's rule integration is used.

**2. Hessian-Vector Products.** With the interpolatory relation between the vector  $U$  and the function  $u$  in mind, we will no longer distinguish between them.

We approximate the product  $\nabla^2 f(u)w$  by differences with a difference increment of  $\delta$ . We will scale the difference increment by  $\|w\|$  when  $w \neq 0$

$$\hat{\delta} = \delta/\|w\|$$

and obtain a forward difference approximation:

$$(2.1) \quad D^2(f, u : w, \delta) = \begin{cases} 0 & \text{if } w = 0 \\ \frac{g(u + \hat{\delta}w) - g(u)}{\hat{\delta}} & \text{if } w \neq 0 \end{cases}$$

The floating point arithmetic error in the computation of  $u + \hat{\delta}w$  is  $O(\epsilon_M \|u + \hat{\delta}w\|)$ , where  $\epsilon_M$  is machine roundoff. Hence the error in the computation of the numerator of the difference quotient in (2.1) is

$$\begin{aligned} E_Q &= \nabla f(u + \hat{\delta}w) - \nabla f(u) - g(u + \hat{\delta}w) + g(u) \\ &= O(\|\nabla f(u)\| \tau_{rg} + \tau_{ag} + \epsilon_M \|u + \hat{\delta}w\|) \end{aligned}$$

Therefore

$$\begin{aligned} \nabla^2 f(u)w - D^2(f, u : w, \delta) &= O(\|w\|^2 \hat{\delta} + \|E_Q\|/\hat{\delta}) \\ &= \|w\| O(\delta + \|E_Q\|/\delta), \end{aligned}$$

and, since  $\epsilon_M \ll 1$ ,

$$\|E_Q\|/\delta = O\left(\frac{\|\nabla f(u)\| \tau_{rg} + \tau_{ag} + \epsilon_M \|u\|}{\delta}\right).$$

Assuming that the  $\epsilon_M \|u\|$  can be neglected, which is reasonable for  $u$  of moderate size, the difference is first order accurate if  $\tau_{ag} = O(\delta^2)$  and  $\tau_{rg} \nabla f(u) = O(\delta^2)$ . This indicates that near optimality, where  $\nabla f(u)$  is small, one can allow the relative error in the gradient to increase somewhat. A similar observation was made in [3] and [4], where large relative errors in the gradient had fairly benign effects.

The situation is similar with central differences where

$$D_C^2(f, u : w, \delta)w = \begin{cases} 0 & \text{if } w = 0 \\ \frac{(g(u + \|u\|\hat{\delta}w) - g(u - \|u\|\hat{\delta}w))}{2\hat{\delta}} & \text{if } u \neq 0 \text{ and } w \neq 0. \end{cases}$$

Here

$$\nabla^2 f(u) - D^2(f, u : w, \delta) = O(\delta^2 \|w\|) + \|w\| O\left(\frac{\|\nabla f(u)\| \tau_{rg} + \tau_{ag}}{\delta}\right).$$

To maintain second order accuracy we must have  $\tau_{ag} = O(\delta^3)$  and  $\tau_{rg} \nabla f(u) = O(\delta^3)$ .

To summarize and make the constant in the O-term explicit, there is  $C_H$  such that for all  $w \in \mathbb{R}^n$ ,

$$(2.2) \quad \|\nabla^2 f(u)w - D^2(f, u : w, \delta)\| \leq C_H \|w\| \left( \delta^q + \frac{\|\nabla f(u)\| \tau_{rg} + \tau_{ag}}{\delta} \right).$$

where  $q = 1$  for forward differences and  $q = 2$  for centered differences.

**3. Errors induced by differencing.** In this section we illustrate four ways in which difference errors can affect the algorithm from [15] and propose ways to address them.

From now on we will assume that

$$\tau_{af} = \tau_{ag} = \tau_{rf} = \tau_{rg} = \tau,$$

and that  $\|\nabla f(u)\| \leq M_g$  throughout the iteration. In this case there is  $C_g$  such that

$$(3.1) \quad \|g(u) - \nabla f(u)\| \leq C_g \tau$$

and (2.2) can be written more simply as

$$(3.2) \quad \|\nabla^2 f(u)w - D^2(f, u : w, \delta)\| \leq (C_H + M_g) \|w\| (\delta^q + \tau/\delta).$$

The right side of (3.4) is minimized when

$$(3.3) \quad \delta = \tau^{1/(q+1)}$$

and we will enforce that for the remainder of the paper. Hence (3.2) becomes

$$(3.4) \quad \nabla^2 f(u) - D^2(f, u : w, \delta) = C_D \|w\| \delta^q,$$

where  $C_D = 2(C_H + M_g)$ .

**3.1. The Finite Difference-CG Iteration.** For  $w \in \mathbb{R}^N$  we let  $d(w) = d(\delta, w)$  be either the forward or central difference approximation of the Hessian-vector product. Let  $\{p_j\}$  be the search directions formed by the usual implementation [9] of CG with either a forward or central difference Hessian-vector products. Let  $w_j = d(p_j)$  and let  $\sigma_j = p_j^T w_j$ . The first  $K$  CG iterates are the same as those that would be obtained with the matrix

$$A_K = \sum_{j=0}^{K-1} \sigma_j^{-1} w_j w_j^T.$$

The finite difference CG iteration (in exact arithmetic) is equivalent to that for the matrix  $A_K$ . However the matrix  $A_K$  need not be a good approximation to  $\nabla^2 f(u)$ .

If  $\sigma_j \leq 0$  then the CG-TR algorithm moves in the direction  $p_j$  to the trust region boundary and returns a step. Therefore, if the approximate solution to the trust region problem is obtained in  $K$  iterations, it is also the one that would be obtained with  $A_K$  as the model Hessian.

**3.2. Termination of the Linear Iteration.** In most Newton-iterative methods the inner iteration is terminated when

$$(3.5) \quad \|\nabla^2 f(u)s + \nabla f(u)\| \leq \eta \|\nabla f(u)\|,$$

where the parameter  $\eta$  is called the forcing term. However, when using finite difference Hessian-vector products and low-resolution functions and gradients, we expect that the step is on the trust region boundary or

$$(3.6) \quad \|d(s) + g\| \leq \eta \|g\|.$$

Assuming that the step is in the interior of the trust region, the CG iteration returns (at least in exact arithmetic [7]) when

$$(3.7) \quad \|A_K s + g\| \leq \eta \|g\|.$$

Neither (3.6) or (3.7) imply (3.5). Moreover, since  $d(s)$  is not linear in  $s$ , (3.6) is not equivalent to (3.7).

Following the analysis in [9] and [10] one can prove

**THEOREM 3.1.** *Let  $0 < \sigma_l \leq \sigma_u$  be the smallest and largest eigenvalues of  $A_K$ . Then*

$$(3.8) \quad \|(A_K - \nabla^2 f(u))z\| = KC_D \sqrt{\sigma_u/\sigma_l} \delta^q \|z\|,$$

for all  $z$  in the  $K$ th Krylov space for  $A_K$ .

*Proof.* We let  $\{p_j\}$  be the CG search directions, which form an  $A_K$ -orthogonal basis for the Krylov space. By construction,  $A_K p_j = d(p_j) = w_j$ . Hence,

$$\|(A_K - \nabla^2 f(u))p_j\| \leq C_D \|p_j\| \delta^q$$

by (3.4). If  $z$  is in the  $K$ th Krylov space, then we can expand  $z$  using  $A_K$ -orthogonality of the basis as

$$z = \sum_{j=0}^{K-1} \sigma_j^{-1} (z^T A_K p_j) p_j = \sum_{j=0}^{K-1} \sigma_j^{-1} (z^T w_j) p_j.$$

Hence,

$$A_K z - \nabla^2 f(u)z = \sum_{j=0}^{K-1} \sigma_j^{-1} (z^T w_j) (A_K p_j - \nabla^2 f(u)p_j),$$

and therefore,

$$\|(A_K - \nabla^2 f(u))z\| \leq C_D \delta^q \sum_{j=0}^{K-1} \|p_j\| \left| \frac{z^T w_j}{\sigma_j} \right|.$$

By  $A_K$ -orthogonality,

$$\left| \frac{z^T w_j}{\sigma_j} \right| = \left| \frac{z^T A_K p_j}{p_j^T A_K p_j} \right| \leq \sqrt{(z^T A_K z) / (p_j^T A_K p_j)} \leq \sqrt{\sigma_u/\sigma_l} \|z\| \|p_j\|,$$

which completes the proof.  $\square$

If  $\sigma_{K-1} \leq 0$  the trust region algorithm will move to the trust region boundary and no further CG iterations will be taken.

**LEMMA 3.2.** *Let  $0 < \sigma_l \leq \sigma_u$  be the smallest and largest eigenvalues of  $A_K$ . Assume (3.8) holds. Then (3.7) implies*

$$(3.9) \quad \|\nabla^2 f(u)s + \nabla f(u)\| \leq \bar{\eta}_1 \|\nabla f(u)\| + \xi_1,$$

and (3.5) implies

$$(3.10) \quad \|A_K s + g\| \leq \bar{\eta}_2 \|g\| + \xi_2,$$

where, for  $j = 1, 2$ ,

$$\bar{\eta}_j = \eta + O(\delta^q) \text{ and } \xi_j = O(\tau).$$

*Proof.* We prove that (3.7) implies (3.9). The other half of the proof is similar. By (3.8) and (3.1)

$$\begin{aligned} \|\nabla^2 f(u)s + \nabla f(u)\| &= \|A_K s - g\| + O(\delta^q \|s\| + \tau) \\ &\leq \eta \|g\| + O(\delta^q \|s\| + \tau) \\ &\leq \eta \|\nabla f\| + O(\delta^q \|s\| + \tau). \end{aligned}$$

(3.7) implies that

$$\|s\| \leq \sigma_l^{-1}(1 + \eta) \|g\|$$

and hence

$$\|\nabla^2 f(u)s + \nabla f(u)\| \leq (\eta + O(\delta^q)) \|\nabla f(u)\| + O(\tau)$$

which is (3.9)  $\square$

Termination of the CG iteration when (3.8) holds implies that (3.5) holds for a slightly larger  $\eta$  (ie the step is a useful step for the  $f$ , if

$$(3.11) \quad \tau + \delta^q \|g\| \leq \rho \|\nabla f\|$$

and  $\rho \ll 1 - \eta$ . Since

$$\tau + \delta^q \|g\| = \left( \frac{\tau}{\|g\|} + \delta^q \right) \|f\|$$

(3.11) will hold if, for example

$$\|g\| > C\tau$$

for a sufficiently large  $C$  and

$$(3.12) \quad \eta = \max(\delta^q, \tau/\|g(u)\|).$$

In summary, to guarantee that the inexact Newton iteration will behave correctly, the termination criterion for the nonlinear iteration, say,

$$\|g\| \leq C\tau,$$

for some  $C > 0$ , must be connected to the forcing term  $\eta$  in the inexact Newton iteration and to the error in the numerical differencing. One way to do this is to use (3.12).

Even larger choices for  $\eta$  can be more efficient in the earlier phases of the iteration [6].

**3.3. Accuracy of the Quadratic Model.** Having generated a step  $s$  the next stage in CG-TR is to test the step for acceptability and adjust the trust region radius. These decisions are based on comparing the predicted reduction  $pred$  (the reduction in the quadratic model) with the actual reduction  $ared$ . In the present case, both computations can be in error.

To compute  $pred$ , the reduction in the quadratic model, one approximates

$$pred_{ideal} = s^T \nabla f(u) + .5s^T \nabla^2 f(u)s$$

with

$$(3.13) \quad pred = s^T g + .5s^T d(s).$$

If the ideal quadratic model is used,  $pred < 0$  is guaranteed in TR methods, such as CG-TR, that enforce Cauchy decrease. As mentioned above, we do not use the ideal quadratic model, but one based on  $A_K$ . However,

$$v^T A_K v \neq v^T d(v).$$

unless  $v = p_j$  for some  $j$ . Hence the value of  $pred$  we compute is neither the ideal quadratic model or the one based on  $A_K$ . The error in  $pred$  can be estimated:

$$|pred - pred_{ideal}| = O(\|s\|\tau + \|s\|^2\delta^q).$$

Using the choice  $\delta = \tau^{1/(q+1)}$ , we obtain

$$(3.14) \quad |pred - pred_{ideal}| = O(\|s\|\tau + \|s\|^2\tau^{q/q+1}).$$

Near optimality, there is  $c > 0$  such that

$$pred_{ideal} \geq c(\|s\|\|e\| + \|s\|^2)$$

and this will dominate the error in (3.14) while  $\|e\| \geq \tau$ , *i. e.* until convergence.

Far from the solution, however,  $\nabla^2 f(u)$  can have small or negative eigenvalues, as can  $A_K$ . In this case, when  $\|s\|$  can be large. A detection of negative curvature from the CG iteration may not be confirmed when  $pred$  is computed using (3.13). Simply reducing the TR radius when  $pred > 0$  will solve this problem, as the error is entirely in the quadratic term.

**3.4. Measurement of Decrease.** If  $\tau_{af} = \tau_{rf} = \tau_f$  and  $\tau_{ag} = \tau_{rg} = \tau_g$ , then  $ared$ , as observed with errors taken into account, is

$$ared = (f(u_c + s) - f(u_c))(1 + O(\tau_f)) + O(\tau_f).$$

The relative errors will not affect the high-order bits of  $ared$  and are therefore harmless. However, the absolute error can make  $ared$  useless. If, now,  $u_c$  is near  $u^*$ , then

$$\begin{aligned} f(u_c + s) - f(u_c) &= s^T \nabla f(u_c) + .5s^T \nabla^2 f(u_c)s + O(\|s\|^3) \\ &= O(\|s\|\|e\|). \end{aligned}$$

If  $\|s\| = O(\|g\|)$  and  $\|e\| = O(\|g\|)$  then as soon as  $\|g\| = O(\sqrt{\tau_f})$ ,  $ared$  will have no accuracy at all and can mislead the trust region algorithm.

In the case where  $\tau_f = \tau_g = \tau$ , acceptance of an inexact Newton step near  $u^*$  implies that

$$\|g(u_+)\| = O(\eta\|g(u_c)\| + \tau).$$

So if  $\eta$  is sufficiently small and if  $ared$  has only one or two digits of accuracy, the step will be accepted and a good reduction obtained. If  $\eta$  is large, on the other hand, inaccuracy in  $ared$  may result in stagnation.

Two possible solutions are to make sure that  $\tau_{af} = O(\tau_g^2)$  or to abandon the test for decrease once  $\|g\|$  becomes sufficiently small or when  $ared \approx \tau$ . This means that the optimization algorithm becomes a Newton-CG iteration and only seeks to find a root of  $g$ .

**4. Changes to the Trust Region Algorithm.** The previous discussion suggests several modifications of the algorithm in [15]:

- Terminate iteration when TR radius is below  $\tau$ . This is consistent with a more standard practice of terminating when too many reductions in the TR radius have been taken. We implement this in all the experiments.
- Modification **pred**: Reduce TR radius if  $pred \geq 0$ .
- Modification  $\eta$ : Enforce  $\|g(u)\|/\tau$  and  $\delta^q$  as lower bounds for  $\eta$ .

- **Modification *ared*:** Switch to an equations algorithm (currently Newton-CG) when  $\|g\| < \sqrt{\tau}$  or  $ared = O(\tau)$ .

The trust region-CG code from [10] was modified to incorporate these changes. The trust region parameters were left unchanged. Both examples have tolerances that can be controlled, exactly for the simple example in § 4.1.1 and approximately in § 4.1.2. In all the examples we set

$$\tau_{af} = \tau_{rf} = \tau_{ag} = \tau_{rg} = \tau,$$

use centered differences with a difference increment of  $(10\tau)^{1/3}$ , and terminate the iteration when  $\|\nabla f(u)\|$  was small or when the trust region radius has been decreased more than 20 times, the latter an indication that the limit of resolution of the function has been reached.

#### 4.1. Examples.

**4.1.1. Perturbed Quadratic.** The purpose of this example is to show that  $pred > 0$  is possible and can lead to failure of the optimization. The modification **pred** solves this problem and the other modifications lead to a more efficient algorithm.

The error-free problem is a quadratic

$$f(u) = .5(u - 2e)^T H(u - 2e) + 1$$

where  $e = (1, \dots, 1)^T$  and the diagonal Hessian is given by

$$H_{ii} = 1 - \frac{(K-1)(i-1)}{K(N-1)}.$$

The Hessian has condition number  $K$ .

We designed perturbations that vary rapidly with  $u$  in the following way. The perturbation for the function was

$$\tau(a_f(u) + r_f(u)f(u))$$

where

$$a_f(u) = \cos(200\pi z), r_f(u) = \sin(200\pi z),$$

and

$$z = \sum_{i=1}^N \cos(100u_i).$$

The perturbation for the gradient

$$\tau(a_g(u) + r_g(u)\|\text{grad } f(u)\|_\infty)$$

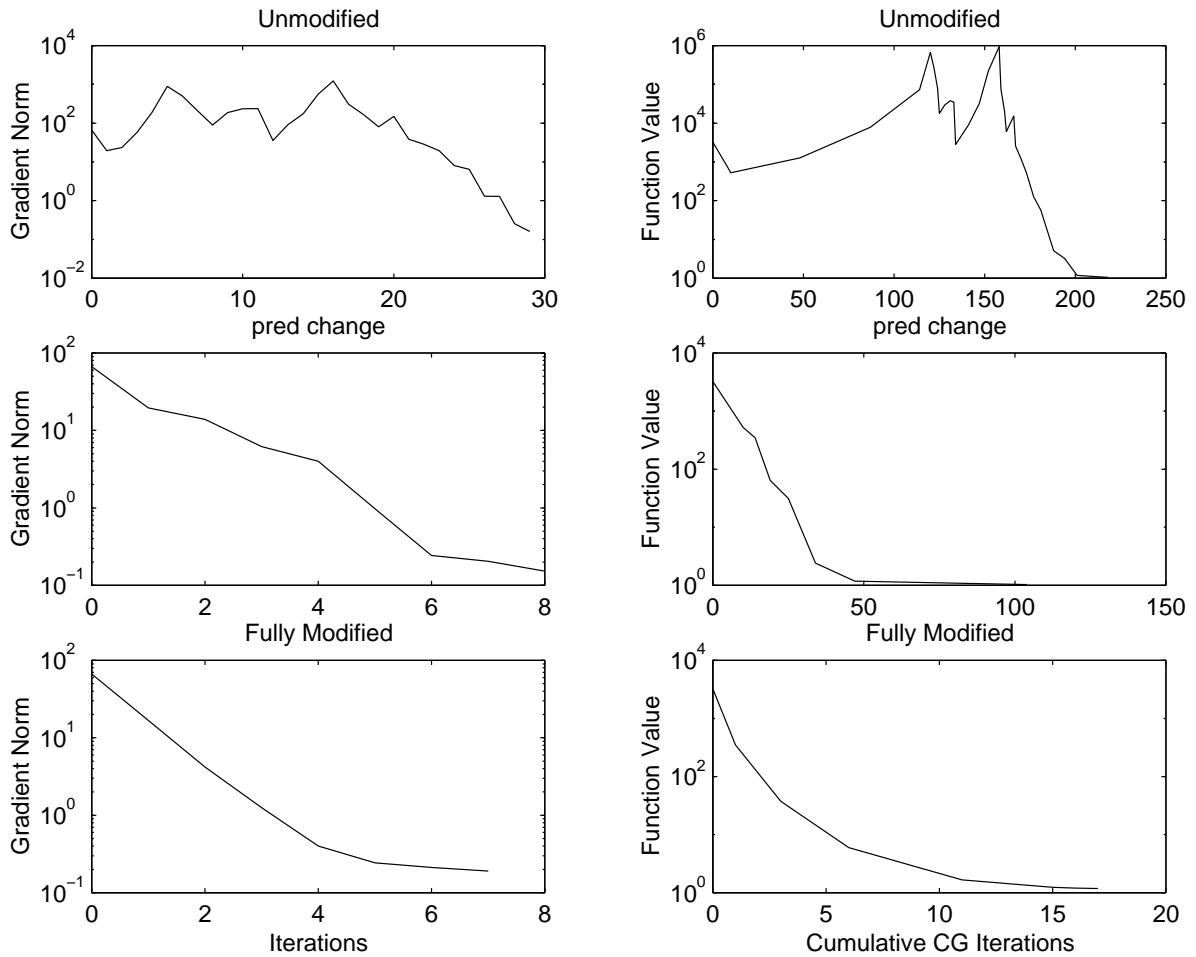
was constructed similarly. Here

$$a_g(u)_i = \cos(200\pi \cos(u_i)) \text{ and } r_g(u)_i = \sin(200\pi \cos(u_i)).$$

In the computations reported in this section the forcing term  $\eta$  in (3.5) was set to .1 when modification  $\eta$  was inactive.

In the computation reported in Figure 4.1  $N = 200$ ,  $\tau = .01$ , and  $K = 200$ . We terminated the iteration when  $\|g\| < .2$ . From the plots one can see the increase in the function value if the sign of  $pred$  is not tested, the significant reduction in the number of CG iterations if modification  $\eta$  is enforced. The **ared** modification did not become active in this example.

FIG. 4.1. *Quadratic Example*



**4.1.2. Optimal Control Problem.** In this example we set

$$L(y, u, t) = (y - 3)^2 + .01u^2$$

in (1.2) and

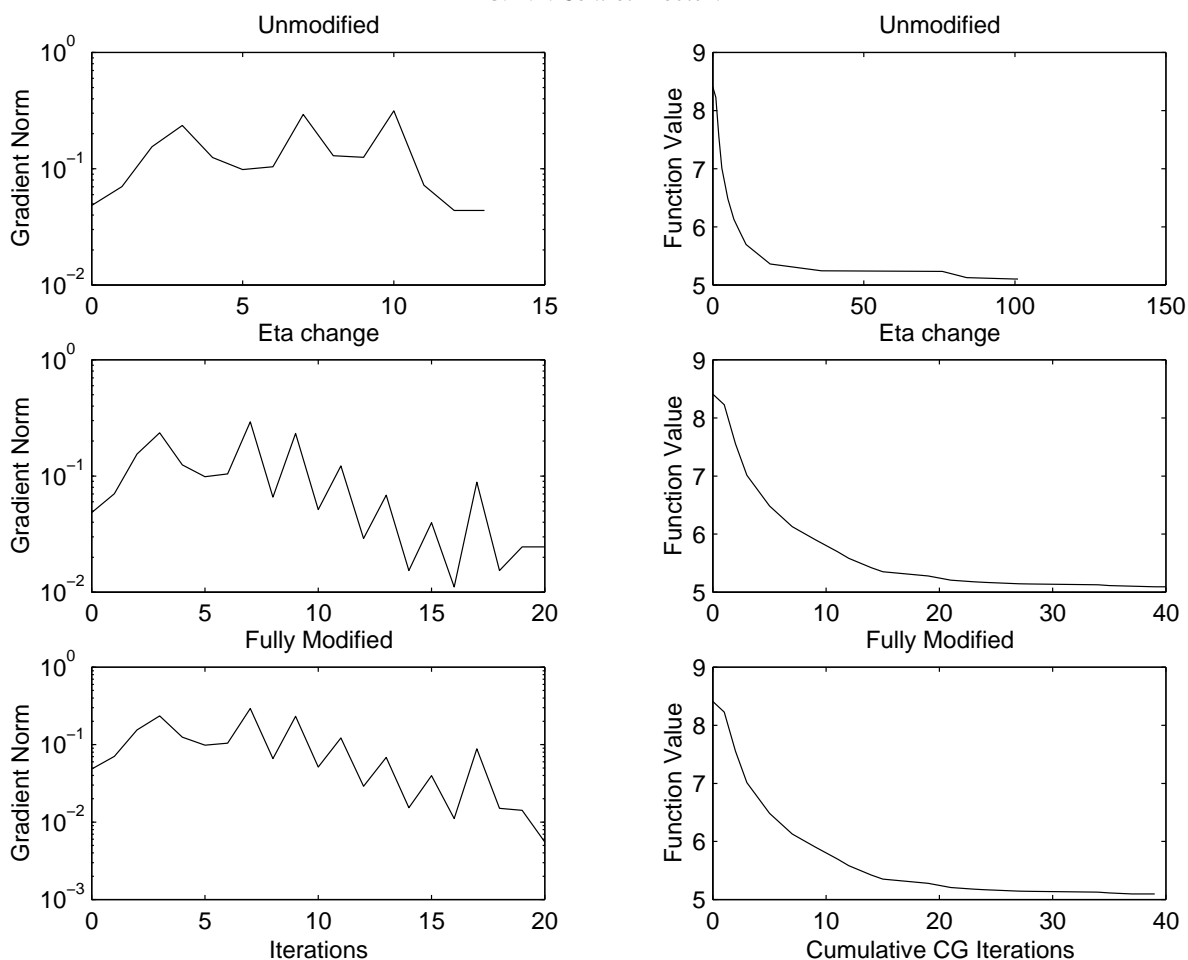
$$\phi(y, u, t) = yu + t^2, \text{ and } y_0 = 0$$

in (1.3).

The discretized control  $u$  was a piecewise linear spline with 10 nodes and the unknowns were the values at the nodes, which were equally spaced on  $[0, T] = [0, 1]$ . In view of the expected second order accuracy, we set the relative and absolute error tolerances in the ODE integrator to  $h^2$ . We solved (1.3) and (1.5) with the `ode15s` stiff integrator in MATLAB. The solution of (1.3) was reported at the nodes by the integrator and  $x$  was extended to all of  $[0, T]$  with piecewise linear interpolation.

In the computations reported in this section the forcing term  $\eta$  in (3.5) was set to .01 when modification  $\eta$  was inactive,  $\tau = .01$ , and the iteration was terminated when  $\|g\| < .01$ .

FIG. 4.2. Control Problem



## REFERENCES

- [1] P. N. BROWN, G. D. BYRNE, AND A. C. HINDMARSH, *VODE: A variable coefficient ode solver*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1038–1051.
- [2] P. N. BROWN, A. C. HINDMARSH, AND L. R. PETZOLD, *Using Krylov methods in the solution of large-scale differential-algebraic systems*, SIAM J. Sci. Comput., 15 (1994), pp. 1467–1488.
- [3] R. G. CARTER, *On the global convergence of trust region algorithms using inexact gradient information*, SIAM J. Numer. Anal., 28 (1991), pp. 251–265.
- [4] ———, *Numerical experience with a class of algorithms for nonlinear optimization using inexact function and gradient information*, SIAM J. Sci. Comput., 14 (1993), pp. 368–388.
- [5] R. DEMBO AND T. STEihaug, *Truncated Newton algorithms for large-scale optimization*, Math. Prog., 26 (1983), pp. 190–212.
- [6] S. C. EISENSTAT AND H. F. WALKER, *Globally convergent inexact Newton methods*, SIAM J. Optim., 4 (1994), pp. 393–422.
- [7] A. GREENBAUM, *Iterative Methods for Solving Linear Systems*, no. 17 in Frontiers in Applied Mathematics, SIAM, Philadelphia, 1997.
- [8] W. W. HAGER, *Rates of convergence for discrete approxiamtions to unconstrained optimal control problems*, SIAM J. Numer. Anal., 13 (1976), pp. 449–472.
- [9] C. T. KELLEY, *Iterative Methods for Linear and Nonlinear Equations*, no. 16 in Frontiers in Applied Mathematics, SIAM, Philadelphia, 1995.
- [10] ———, *Iterative Methods for Optimization*, no. 18 in Frontiers in Applied Mathematics, SIAM, Philadelphia, 1999.
- [11] S. G. NASH, *Newton-type minimization via the Lanczos method*, SIAM J. Numer. Anal., 21 (1984), pp. 770–789.
- [12] ———, *Preconditioning of truncated Newton methods*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 599–616.
- [13] K. RADHAKRISHNAN AND A. C. HINDMARSH, *Description and use of LSODE, the Livermore solver for ordinary differential equations*, Tech. Rep. URCL-ID-113855, Lawrence Livermore National Laboratory, December 1993.
- [14] L. F. SHAMPINE AND M. W. REICHELt, *The MATLAB ODE suite*, SIAM J. Sci. Comput., 18 (1997), pp. 1–22.
- [15] T. STEIHAUG, *The conjugate gradient method and trust regions in large scale optimization*, SIAM J. Numer. Anal., 20 (1983), pp. 626–637.