ELSEVIER

Journal of Computational and Applied Mathematics 124 (2000) 45–59

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

A survey of truncated-Newton methods

Stephen G. Nash[1]

Systems Engineering and Operations Research Department, George Mason University, Fairfax, VA 22030, USA

Received 18 June 1999; received in revised form 8 December 1999

Abstract

Truncated-Newton methods are a family of methods for solving large optimization problems. Over the past two decades, a solid convergence theory has been derived for the methods. In addition, many algorithmic enhancements have been developed and studied, resulting in a number of publicly available software packages. The result has been a collection of powerful, flexible, and adaptable tools for large-scale nonlinear optimization.

1. Introduction

Truncated-Newton methods are a family of methods suitable for solving large nonlinear optimization problems. At each iteration, the current estimate of the solution is updated (i.e., a step is computed) by approximately solving the Newton equations using an iterative algorithm. This results in a doubly iterative method: an outer iteration for the nonlinear optimization problem, and an inner iteration for the Newton equations. The inner iteration is typically stopped or "truncated" before the solution to the Newton equations is obtained.

More generally, an "inexact" Newton method computes a step by approximately solving the Newton equations. This need not be done using an iterative method. These definitions, however, are not universal. In some papers, "inexact" Newton methods refer to methods for solving systems of nonlinear equations, and "truncated" Newton methods refer to methods for solving optimization problems. I focus here on truncated-Newton methods and optimization problems.

A truncated-Newton method will be effective if

- a small number of inner iterations is sufficient to produce a "good" step,
- each inner iteration can be performed efficiently,

[1] Partially supported by National Science Foundation grant DMI-9800544.

0377-0427/00/$ - see front matter
PII: S0377-0427(00)00426-X

- the overall method is implemented with appropriate safeguards (a "globalization" strategy) to guarantee convergence to a stationary point or local optimum, in cases where the optimization problem satisfies appropriate assumptions.

These issues motivate much of my discussion.

Choices are available for the components of a truncated-Newton method:

- the globalization procedure (some form of line search or trust region strategy),
- the inner iterative algorithm,
- the preconditioner for the inner algorithm,
- the truncation rule for the inner algorithm,
- the technique for computing or estimating second-derivative information.

These choices provide a great deal of flexibility, and allow the method to be adapted to the optimization problem and the computing environment. "Black-box" software is available, but a sophisticated practitioner can enhance the basic method in many ways when faced with a difficult optimization problem.

In much of this paper I focus on the unconstrained problem

$$\min f(x) \tag{1}$$

since the ideas can be explained more simply in this setting, and many of the ideas carry over directly to the constrained case. The first-order optimality condition for this problem is

$$\nabla f(x) = 0,$$

which is a system of nonlinear equations. For this reason, results for nonlinear equations provide insight in the optimization setting.

Given some guess x_k of a solution x_*, Newton's method computes a step p_k as the solution to the linear system

$$\nabla^2 f(x_k)\, p = -\nabla f(x_k) \tag{2}$$

and then sets $x_{k+1} \leftarrow x_k + p_k$. In this simple form, Newton's method is not guaranteed to converge.

In a truncated-Newton method, an iterative method is applied to (2), and an approximate solution accepted. In [3], the rate of convergence of the outer iteration is proven to be related to the accuracy with which (2) is solved. The paper [3] focuses on nonlinear equations, but the results apply (with minor modification) to optimization problems. These results clarify the local convergence behavior of a truncated-Newton method (i.e., the behavior of the method when x_k is close to the solution x_*).

If the problem (1) satisfies appropriate assumptions, then global convergence (to a local solution) can be guaranteed in either a line search or a trust region framework by making adjustments to the inner algorithm (see Section 3). (In this paper, "global convergence" for an unconstrained problem means that the limit of the gradient norms is zero.) Building upon this foundation, many practical enhancements can be made to the overall method.

A basic question in a truncated-Newton method is the choice of an inner iterative algorithm for solving (2). Some variant of the linear conjugate-gradient method is almost always used. The conjugate-gradient method is an optimal iterative method for solving a positive-definite linear system $Ap = b$, in the sense that the ith iterate p_i minimizes the associated quadratic function $Q(p) = \frac{1}{2} p^{\mathrm{T}} A p - p^{\mathrm{T}} b$ over the Krylov subspace spanned by $\{b, Ab, \ldots, A^{i-1}b\}$.

The Hessian matrix $\nabla^2 f(x_k)$ need not be positive definite, so the assumptions underlying the conjugate-gradient method may not be satisfied. However, the Hessian matrix is always symmetric. At a local minimizer of (1), the Hessian is guaranteed to be positive semi-definite; in nondegenerate cases it will be positive definite. Thus, as the solution is approached (and the Newton model for (1) is more accurate and appropriate) we can anticipate that the requirements for the conjugate-gradient method will be satisfied. If the Hessian matrix is not positive definite, then the techniques discussed in Section 5 should be used.

A truncated-Newton method will only be competitive if further enhancements are used. For example, a preconditioner for the linear system will be needed, and the stopping rule for the inner algorithm will have to be chosen so that it is effective both close to and far from the solution. With these enhancements, truncated-Newton methods are a powerful tool for large-scale optimization.

Because of all the choices that can be made in designing truncated-Newton methods, they form a flexible class of algorithms. For this reason, the method can be adapted to the problem being solved. Thus, if "black box" software is not able to solve a problem successfully, it is possible to modify the inner algorithm, the preconditioner, the stopping rule for the inner iteration, or a number of other details to enhance performance.

A constrained optimization problem

$$\begin{array}{ll} \min & f(x) \\ \text{subject to} & g(x) = 0, \\ & h(x) \geqslant 0, \end{array}$$

can be solved using a penalty-barrier method, in which one solves a sequence of unconstrained problems of the form

$$\min_x f(x) + \rho_j \sum g_i(x)^2 - \frac{1}{\rho_j} \sum \log(h_i(x))$$

for an increasing sequence of values of $\rho_j \to \infty$ [19]. Each of the unconstrained problems can be solved using a truncated-Newton method, and so all of the above comments apply in this case. (There are also some new issues; see Section 10.) This is not the only possible approach to constrained problems, but it does indicate the relevance of unconstrained optimization techniques in this setting.

Some applications where truncated-Newton methods have been effective include:

- weather modeling,
- potential-energy minimization,
- molecular geometry,
- multicommodity flow,
- medical imaging,
- molecular conformation.

Truncated-Newton methods have been extended to the infinite-dimensional case, at least in the setting of nonlinear equations. See, for example, [23].

Many of the above ideas are discussed in greater detail in the remainder of the paper. Here is an outline of the topics covered:

- controlling the convergence rate (Section 2),
- guaranteeing convergence (Section 3),

- computing second-derivative information (Section 4),
- handling nonconvex problems (Section 5),
- preconditioning (Section 6),
- parallel algorithms (Section 7),
- practical behavior (Section 8),
- software (Section 9),
- constrained problems (Section 10).

A version of this paper containing an expanded reference list can be obtained from `http://iris.gmu.edu/~snash/` under "New Papers", or by contacting the author.

1.1. Basics

The default norm $\|\cdot\|$ used in this paper is the 2-norm: for a vector $x = (x_1, \ldots, x_n)$, $\|x\| = \sqrt{x_1^2 + \cdots + x_n^2}$. All vectors are column vectors.

The conjugate-gradient method for solving a linear system $Ap = b$ is initialized with $p_0 = 0$, $r_0 = b$ (r_i is the ith residual $b - Ap_i$), $v_{-1} = 0$, and $\beta_0 = 0$. Then

For $i = 0, 1, \ldots$
  If stopping rule satisfied, stop
  If $i > 0$ set $\beta_i = r_i^{\mathrm{T}} r_i / r_{i-1}^{\mathrm{T}} r_{i-1}$
  Set $v_i = r_i + \beta_i v_{i-1}$
  Set $\alpha_i = r_i^{\mathrm{T}} r_i / v_i^{\mathrm{T}} A v_i$
  Set $p_{i+1} = p_i + \alpha_i v_i$
  Set $r_{i+1} = r_i - \alpha_i A v_i$

Stopping rules are discussed in Section 2. The algorithm requires the computation of the matrix–vector product Av_i, but other information about A need not be supplied.

A line-search method for solving (1) has the following basic form: Specify some initial guess of the solution x_0. Then

For $k = 0, 1, \ldots$
  If stopping rule satisfied, stop
  Compute a search direction p_k
  Determine an improved estimate of the solution $x_{k+1} = x_k + \alpha_k p_k$
    [line search]

"Improvement" is often measured in terms of the function value $f(x_{k+1})$. For example, the new estimate x_{k+1} might be required to satisfy a "sufficient decrease" condition of the form

$$f(x_{k+1}) \leqslant f(x_k) + \mu \alpha p_k^{\mathrm{T}} \nabla f(x_k)$$

for some $0 < \mu < 1$ [19]. That is, there must be a decrease in the function value that is a fraction of the decrease predicted by the first-order Taylor series approximation to $f(x_k + \alpha p_k)$.

A trust-region method for solving (1) has the following basic form: Specify some initial guess of the solution x_0, and specify Δ_0, the bound on the size of the "trust region", i.e., the bound on the

length of the allowable step at the current iteration. Then

For $k = 0, 1, \ldots$

If stopping rule satisfied, stop

Choose p_k so as to minimize some approximation $\psi_k(p) \approx f(x_k + p)$,

subject to the constraint $||p|| \leqslant \Delta_k$

Compute x_{k+1} and Δ_{k+1} using p_k.

Algorithms for constrained problems can also be imbedded inside line search or trust region approaches, but the details are more complicated (both practically and theoretically). For more information on these topics, see [19].

2. Controlling the convergence rate

The basic local convergence theorem appeared in [3], in the context of nonlinear equations. Here is an adaptation of that theorem to unconstrained optimization. The definition of a q [strong] rate of convergence can be found in [3].

Theorem 1. *Assume that ∇f is continuously differentiable in a neighborhood of a local solution x_* of (1). In addition, assume that $\nabla^2 f(x_*)$ is nonsingular and that $\nabla^2 f$ is Lipschitz continuous at x_*. Assume that iteration k of the truncated-Newton method computes a step p_k that satisfies*

$$||\nabla f(x_k) + \nabla^2 f(x_k) p_k|| \leqslant \eta_k ||\nabla f(x_k)||$$

for a specified value of η_k; the new estimate of the solution is computed using $x_{k+1} \leftarrow x_k + p_k$. If x_0 is sufficiently close to x_* and $0 \leqslant \eta_k \leqslant \eta_{\max} < 1$ then $\{x_k\}$ converges to x_* q-linearly in the norm $||\cdot||_*$ defined by $||v||_* \equiv ||\nabla^2 f(x_*)v||$, with asymptotic rate constant no greater than $\eta_{\max}$. If $\lim_{k\to\infty} \eta_k = 0$, then the convergence is q-superlinear. If $\eta_k = \mathrm{O}(||\nabla f(x_k)||^r)$ for $0 < r \leqslant 1$, then the convergence is of order at least $(1 + r)$.

The sequence $\{\eta_k\}$ is referred to as a "forcing" sequence. The theorem shows that there is a direct relationship between the forcing sequence and the rate of convergence of the truncated-Newton method for (1). In [3] the authors suggest using

$$\eta_k = \min\{\tfrac{1}{2}, c||\nabla f(x_k)||^r\}$$

as a practical forcing sequence, where c is a positive constant, and $0 < r \leqslant 1$. This sequence leads to a method with a fast asymptotic convergence rate. However, it is not scale invariant, i.e., the behavior of the truncated-Newton method will change if the objective function $f(x)$ is multiplied by a positive constant.

If the conjugate-gradient method is used for the inner iteration, then the ith inner iteration finds a minimizer of the quadratic model

$$f(x_k + p) \approx f(x_k) + p^{\mathrm{T}} \nabla f(x_k) + \tfrac{1}{2} p^{\mathrm{T}} \nabla^2 f(x_k) p \equiv Q_k(p) \tag{3}$$

over the Krylov subspace spanned by $\{\nabla f(x_k), \ldots, [\nabla^2 f(x_k)]^{i-1} \nabla f(x_k)\}$. The model (3) has a global minimum when the residual of the Newton equations is zero. If p is not the minimizer of the

quadratic model, however, the magnitudes of the residual and the quadratic model can be dramatically different [15], and the residual can be a deceptive measure of the quality of the search direction. For this reason, it may be preferable to base a stopping rule on the value of the quadratic model.

Let p_i be the search direction computed at the ith inner iteration, and let $Q_i = Q(p_i)$. The stopping rule suggested in [15] is to accept a search direction if

$$i(Q_i - Q_{i-1})/Q_i \leqslant \eta_k.$$

Quoting from [15]: "This criterion ... compares the reduction in the quadratic model at the current iteration ($Q_i - Q_{i-1}$) with the average reduction per iteration (Q_i/i). If the current reduction is small relative to the average reduction (with 'small' measured by η_k), then the inner iteration is terminated."

Newton's method is based on the Taylor series approximation (3). If this approximation is inaccurate then it may not be sensible to solve the Newton equations accurately. ("Over-solving" the Newton equations will not produce a better search direction.) In this circumstance, the inner algorithm should be truncated after a small number of iterations.

The conjugate-gradient method minimizes the quadratic model (3); in particular, it computes the value of the quadratic model. The quadratic model predicts the amount of decrease that will be obtained in the objective value. The outer iteration will typically compute $f(x_k + p)$, and hence determines the actual decrease in the objective value. By comparing these two quantities, the algorithm can determine if the quadratic model is accurate. If not, an alternative value of the forcing term η_{k+1} can be used at the next outer iteration. A simple rule of this type is used in [16]; more sophisticated approaches are analyzed in [5].

The paper [5] identifies two successful forcing sequences (in the context of solving nonlinear equations, but adapted here for optimization). Let p_k be the search direction at the kth outer iteration, and let α_k be the step length. The first stopping sequence uses $\eta_0 \in [0,1)$, and then for $k = 0, 1, \ldots$

$$\eta_{k+1} = \frac{|\,||\nabla f(x_{k+1})|| - ||\nabla f(x_k) + \alpha_k \nabla^2 f(x_k) p_k||\,|}{||\nabla f(x_k)||}.$$

The second uses $\eta_0 \in [0,1)$, $\gamma \in [0,1]$, $\phi \in (1,2)$, and then for $k = 0, 1, \ldots$

$$\eta_{k+1} = \gamma \left(\frac{||\nabla f(x_{k+1})||}{||\nabla f(x_k)||} \right)^{\phi}.$$

Both are designed to provide good asymptotic performance while at the same time preventing over-solving.

All of these results can be applied directly to the sequence of unconstrained problems that arise when a penalty-barrier method is used to solve a constrained problem. The convergence of the overall penalty-barrier method is discussed in [19]. Convergence results for a gradient-projection method for linearly-constrained problems can be found in [8].

3. Guaranteeing convergence

Convergence can be guaranteed by imbedding a truncated-Newton method in either a line-search or a trust-region framework. This is straightforward, although minor adjustments to the inner algorithm must be made. Basic convergence theorems for line-search and trust-region frameworks can be found (for example) in [19].

Global convergence results for the more general setting of inexact-Newton methods are developed in [4]. I assume here that a truncated-Newton method is used, with the conjugate-gradient method as the inner algorithm.

A variety of convergence results are available for line-search methods. In one such (from [19]), the line search method can be guaranteed to converge (in the sense that the limit of the gradient norms is zero) if the following assumptions are satisfied:

- the level set $S = \{x: f(x) \leqslant f(x_0)\}$ is bounded,
- ∇f is Lipschitz continuous for all $x \in S$,
- the search directions p_k satisfy a sufficient-descent condition

$$-\frac{p_k^{\mathrm{T}} \nabla f(x_k)}{||p_k|| \cdot ||\nabla f(x_k)||} \geqslant \varepsilon > 0$$

  for some fixed ε,
- the search directions are gradient related: $||p_k|| \geqslant m||\nabla f(x_k)||$ for some fixed $m > 0$,
- the search directions are bounded: $||p_k|| \leqslant M$ for some fixed M,
- an "appropriate" line search is used.

The first two conditions are assumptions on the optimization problem, and the final condition is independent of the inner algorithm.

Before discussing the other three conditions, it is useful to discuss the Lanczos method. The Lanczos method can be applied to any symmetric matrix A. It determines a sequence of orthogonal matrices V_i and tridiagonal matrices T_i such that

$$V_i^{\mathrm{T}} A V_i = T_i.$$

The Lanczos method is equivalent to the conjugate-gradient method. If $A = \nabla^2 f(x_k)$ and p_i is the result of the ith iteration of the conjugate-gradient method applied to (2), then

$$p_i = -V_i T_i(x_k)^{(-1)} V_i^{\mathrm{T}} \nabla f(x_k).$$

See [21] for further details. The other three conditions for convergence will be satisfied if the eigenvalues of the matrices T_i are uniformly bounded for all x_k:

$$0 < c_1 \leqslant \lambda_{\min}[T_i(x_k)] \leqslant \lambda_{\max}[T_i(x_k)] \leqslant c_2.$$

The upper bound can be guaranteed if the level set S is bounded, and if the Hessian is continuous on S. The lower bound can be guaranteed by making adjustments to the conjugate-gradient method. These ideas are discussed further in Section 5.

These convergence results are based on a "traditional" line search, i.e., the new estimate of the solution is obtained as $x_{k+1} \leftarrow x_k + \alpha p_k$, where α is chosen to ensure that the objective function decreases at every iteration. Convergence can also be proved for algorithms that use a curvilinear line search [9]; the new estimate of the solution is of the form

$$x_{k+1} = x_k + \alpha^2 p_k + \alpha d_k$$

where d_k is a direction of negative curvature (see Section 5). In addition, convergence can be proved for algorithms that use a non-monotone line search [7], where decrease in $f(x_k)$ is not required at every iteration.

When using a trust-region method, fewer adjustments need be made to the conjugate-gradient method. One significant issue, though, is to ensure that the output vector p_k satisfies the trust-region constraint:

$$||p_k|| \leqslant \Delta_k$$

where Δ_k is a parameter in the trust-region method. A technique for this was suggested by [25], and is outlined here. It is straightforward to prove that the iterates from the conjugate-gradient method increase monotonically in norm as long as the tridiagonal matrix T_i is positive definite:

$$||p_0|| < ||p_1|| < ||p_2|| < \cdots.$$

Thus, it is easy to determine at which iteration the trust-region constraint is violated, and to choose p_k as the point between p_i and p_{i+1} which exactly satisfies the constraint. Of course, if the termination rule is satisfied before the trust-region constraint is encountered, then the inner algorithm terminates before this occurs. If T_i becomes indefinite, then the quadratic model is unbounded below, and the next step in the inner iteration will cause the trust-region constraint to be violated. In [6] the authors examine more closely what happens in a truncated-Newton method when the trust-region boundary is encountered, and propose alternatives to simply truncating the inner iteration in this case.

Just as in Section 2, all of these results can be applied when a penalty-barrier method is used to solve a constrained problem. Global convergence results for a trust-region method for linearly constrained problems can be found in [8].

4. Computing second-derivative information

The conjugate-gradient method requires the computation or estimation of matrix-vector products involving the Hessian of the objective function

$$w = \nabla^2 f(x_k)v \tag{4}$$

for any vector v. This can be accomplished in a variety of ways.

If the Hessian $\nabla^2 f$ is explicitly available then (4) can be computed directly. This can be especially efficient if the Hessian is sparse. The user must be able to derive and program the formulas for the Hessian to use this technique. The remaining techniques require less effort on the part of the user.

An estimate of (4) can be obtained using finite differencing:

$$w \approx \frac{\nabla f(x_k + hv) - \nabla f(x_k)}{h}$$

for some "small" h. Each matrix–vector product requires one gradient evaluation, since $\nabla f(x_k)$ is already available as the right-hand side of (2). The choice of h is discussed in [19], as are alternative finite-difference formulas. This approach is widely used in practical truncated-Newton methods.

If it is possible to use complex arithmetic, then a more accurate finite-difference approximation to w can be obtained using

$$v_h = x_k + \sqrt{-1}hv, \quad g_h = \nabla f(v_h), \quad w \approx \mathrm{Im}(g_h)/h.$$

With this technique it is possible to choose a very small value of h (e.g., $h = 10^{-16}$) and obtain an estimate of w that is accurate to $O(h)$.

A third alternative is to use automatic differentiation [19] to compute (4). This is an exact calculation (up to the limits of computer arithmetic). The computational cost is comparable to a gradient evaluation, and thus comparable to the finite-difference technique.

5. Nonconvex problems

As was mentioned in Section 3, the conjugate-gradient method is equivalent to the Lanczos method in the sense that

$$p_i = -V_i T_i(x_k)^{(-1)} V_i^{\mathrm{T}} \nabla f(x_k),$$

where $A = \nabla^2 f(x_k)$ and

$$V_i^{\mathrm{T}} A V_i = T_i.$$

Here $\{V_i\}$ is a sequence of orthogonal matrices and $\{T_i\}$ is a sequence of tridiagonal matrices.

If A is positive definite, then the formulas for the conjugate-gradient method correspond to computing the factorization

$$T_i = L_i D_i L_i^{\mathrm{T}}$$

where D_i is diagonal (with positive diagonal entries), and L_i is lower triangular (with ones along the diagonal). This factorization exists if and only if T_i is positive definite.

If T_i is not positive definite then this factorization cannot be computed. The algorithm will break down if a diagonal entry of D_i is zero, and will be numerically unstable if a diagonal entry of D_i is negative. To guarantee convergence (see Section 3) the diagonal entries of D_i must be positive and bounded away from zero.

The same situation occurs for certain implementations of Newton's method. In that setting a variety of proposals have been made that correspond to "modifying" the Hessian (or, equivalently, the factorization) to obtain a new, positive definite matrix that then replaces the Hessian in (2).

Any of these techniques could, in principle, be applied to the factorization of the tridiagonal matrix T_i. This is not usually done, however, because the components of the matrix T_i are generated iteratively, and the matrices T_i and V_i are not stored.

An alternative approach that uses information from two successive iterations of the conjugate-gradient method is developed in [12]. This "modified" conjugate-gradient method is iterative (like the regular conjugate-gradient method), and has many of the same theoretical and practical properties as modified-Newton methods.

It is possible to use a simpler technique, and develop a "modified" method using only information from the current iteration of the conjugate-gradient method. This approach is mentioned in [12]. The drawback to this approach is that the modification to the Hessian can be very large in norm, much larger than if information from two successive iterations is used.

If the tridiagonal matrix T_i is not positive semi-definite, then the matrix D_i must have a negative diagonal entry. This corresponds to a direction of negative curvature, i.e., a vector d satisfying

$$d^{\mathrm{T}}[\nabla^2 f(x_k)]d < 0.$$

Such a direction can be used as part of a search direction, since either d or $-d$ is a direction of nonascent. This idea is discussed in [9].

The trust-region techniques discussed in Section 3 provide an alternative way of handling non-convex problems. If a diagonal entry of D_i is negative, then the quadratic model can be decreased to $-\infty$ by following this direction of negative curvature. Thus, a sufficiently long step along such a direction (or any direction) is guaranteed to violate the trust-region constraint.

The application of these ideas to constrained problems is discussed in Section 10.

6. Preconditioning

The convergence of the conjugate-gradient method is strongly influenced by the condition number of the Hessian (i.e., its extreme eigenvalues), and by the number of distinct eigenvalues of the Hessian. Reducing either of these accelerates the convergence of the method.

Ideally, a preconditioner will be chosen based on the problem being solved. This can require considerable analysis and programming to accomplish, however, and is not suitable for routine cases.

If the Hessian matrix is available, a good "generic" choice of a preconditioner is an incomplete Cholesky factorization. The preconditioner is formed by factoring the Hessian, and ignoring some or all of the fill-in that occurs during Gaussian elimination. It may be necessary to modify the factorization (as discussed in Section 5) so that the preconditioner is positive definite. This idea is discussed in [24].

It is also possible to develop preconditioners based on partial separability in the objective function [2]. (A function $f(x)$ is partially separable if it can be written as the sum of functions $f_i(x)$, each of which has a large invariant subspace.)

If neither of these is possible, "automatic" preconditioners can be developed that do not require Hessian information. These preconditioners are based on quasi-Newton approximation to the Hessian. A quasi-Newton approximation is computed based on vector pairs (s_i, y_i). Traditionally, $s_i = x - \hat{x}$ for some pair of variable values, and $y_i = \nabla f(x) - \nabla f(\hat{x})$, the corresponding difference of gradient values. In the context of a truncated-Newton method, these might be $x = x_k$ and $\hat{x} = x_{k+1}$, i.e., two successive iterates.

It is also possible to use an arbitrary vector s_i with $y_i = \nabla^2 f(x_k)s_i$. At each iteration of the conjugate-gradient method, a matrix–vector product of this form is computed or estimated, and each of these matrix–vector products can be used to help construct a Hessian approximation.

In [13], both these ideas are combined to form a preconditioner. The matrix–vector products from the inner iteration are used to construct a diagonal approximation to the Hessian, using a BFGS update formula in which only the diagonal matrix entries are computed. This is in turn used to initialize a two-step limited-memory BFGS update formula which is the actual preconditioner. The limited-memory update is constructed using pairs in which s_i is the difference between a pair of x-vectors. Precise information is given in [14]. This preconditioner is implemented in the `TN/TNBC` software discussed in Section 9.

A more elaborate preconditioner is described in [11], based on an m-step limited memory BFGS update, with the (s_i, y_i) pairs chosen as a subset of the matrix–vector products in the inner iteration. Experiments are conducted with various choices of m. The authors propose an algorithm that "dynamically stores the correction pairs so that they are as evenly distributed as possible" among the set of pairs for a complete inner iteration.

The application of these ideas to constrained problems is discussed in Section 10.

7. Parallel algorithms

A parallel algorithm could be obtained by executing each of the steps of the truncated-Newton method in parallel. This would require converting the line search and the conjugate-gradient method so that they execute in parallel. By itself, this is not likely to be an effective strategy, since the steps in these algorithms consist of

- scalar operations,
- vector operations,
- function and gradient evaluations.

The scalar operations cannot be made parallel, and the vector operations do not offer much potential for speed-up on a parallel machine (since communication and synchronization delays could easily wipe out any computational savings obtained).

The function and gradient evaluations offer more hope, but this requires that the person solving the optimization problem be willing and able to compute these values effectively in parallel. For very large and difficult problems, however, this may be essential.

An alternative is to replace the line search and the inner algorithm with alternatives that are better able to exploit parallelism. Ideally, it should be possible to take advantage of both parallel linear algebra computations as well as parallel function and gradient evaluations (that is, simultaneous evaluations of the function and/or gradient on separate processors).

An approach of this type for unconstrained problems is discussed in [16]. In this work, the block conjugate-gradient method is used as the inner algorithm; this is a generalization of the conjugate-gradient method in which a block of vectors (rather than a single vector) is updated at every inner iteration. A simple parallel line search is used to compute $x_{k+1} \leftarrow x_k + \alpha p_k$. If the block size in the block conjugate-gradient method is equal to m, then each inner iteration requires the computation of m independent matrix–vector products (which can be approximated by m independent gradient evaluations). There is also considerable opportunity for parallel linear algebra computations. Each iteration of the line search requires m independent function evaluations.

A hybrid approach (combining parallelism in the algorithm with parallelism in the individual function evaluations) is also possible within the block conjugate-gradient method. The block size m need not be equal to the number of processors. This can be an advantage if the individual function and gradient evaluations can be performed in parallel. For example, suppose that a computer with 32 processors were available, and that each function or gradient evaluation could be spread over 4 processors. Then, if the block-size were chosen as $m = 8$, an inner iteration would require 8 gradient evaluations, each of which would require 4 processors. Thus a total of $4 \times 8 = 32$ processors would be used.

This algorithm is implemented in the software package `BTN`; see Section 9.

8. Practical behavior

Truncated-Newton methods use an inner iteration to compute a search direction, and thus expend considerable computational effort at each outer iteration. In contrast, nonlinear conjugate-gradient methods and limited-memory quasi-Newton methods use relatively few computations to obtain each

search direction. (Precise operation counts can be found in [14].) A basic question is whether the effort per iteration for a truncated-Newton method can be worthwhile.

The tests in [14] compare the truncated-Newton method TN against the limited-memory quasi-Newton method L-BFGS. The tests imply that L-BFGS becomes more effective as the optimization problem (1) becomes more nonlinear. In a sense, the truncated-Newton method is more effective when the quadratic model (3) is more effective.

Attempts have been made to combine the best properties of both these methods. This consisted in

- monitoring the effectiveness of the quadratic model to avoid "over-solving" in cases where the quadratic model is poor,
- using limited-memory quasi-Newton formulas as preconditioners,
- combining both techniques in a single algorithm.

Testing of specific features of truncated-Newton software can be found in [12,13] and, for the parallel case, in [16]. The results of these tests have influenced the development of the software packages mentioned in Section 9. The paper [22] describes software for nonlinear equations, but many of the comments are also applicable to optimization. Tests of truncated-Newton methods for bound-constrained problems can be found in [8].

9. Software

Truncated-Newton software is available for unconstrained and bound constrained problems. The first three packages are available from the Netlib collection (www.netlib.org).

The package TN/TNBC solves both classes of problems. It requires that the user provide a subroutine to evaluate the function value and gradient of the objective function. The algorithm is described in [14]. This software is designed to be easy to use, and does not require or expect customization by the user.

The package TNPACK solves unconstrained minimization problems. In addition to function and gradient information, the user must provide formulas for the Hessian matrix and a user-supplied preconditioner. This software expects the user to supply information about the Hessian so that a preconditioner can be constructed. This can require considerable effort, but with the promise of improved performance.

The package BTN solves unconstrained problems on parallel computers (both shared and distributed memory). It requires that the user provide a (scalar) subroutine to evaluate the function value and gradient of the objective function. In addition BTN can take advantage of a parallel subroutine for the function and gradient if one is provided. The algorithm is described in [16]. This software comes with both easy-to-use and customizable top-level subroutines.

The TRON software [8] solves bound-constrained problems. It requires that the user supply function, gradient, and Hessian information. It uses an incomplete Cholesky factorization as a preconditioner. The software can be obtained from

```
www.mcs.anl.gov/~more/tron/
```

The Lancelot software [1] is a more general package, but a variety of truncated Newton algorithms can be used within it by appropriately selecting software parameters. Considerable customization is

possible. Information about this software can be obtained from

`www.cse.clrc.ac.uk/Activity/LANCELOT`

A variety of software packages for solving nonlinear systems of equations are mentioned in [22].

Software for the quasi-Newton preconditioner in [11] is available from

`www.ece.nwu.edu/~nocedal/preqn.html`

10. Constrained problems

Many algorithms for constrained optimization problems are built upon algorithms, techniques, or principles from unconstrained optimization. It should not be surprising that truncated-Newton methods can be used in this setting.

One approach is to use a penalty-barrier method to solve the constrained problem [19]. The constrained problem is replaced by a sequence of unconstrained problems, where violations in the constraints are included as penalty terms in the objective function (see Section 1). A truncated-Newton method can then be applied to the sequence of unconstrained problems. Under appropriate assumptions, it is possible to derive complexity results for an algorithm of this type [20].

It has been known for decades that the unconstrained problems become increasingly ill-conditioned as the barrier parameter is increased (i.e., as the solution is approached). This ill-conditioning causes the behavior of the inner algorithm to deteriorate. This ill-conditioning is not inherent, however.

In [17], an approximation to the inverse of the Hessian matrix is derived that can be used within the conjugate-gradient method. With this approximation, the conjugate-gradient method is applied to a linear system whose conditioning reflects that of the underlying optimization problem, and not that of the penalty-barrier problem. The approximation formula requires, though, that an active set be identified (a prediction of the set of constraints that are binding at the solution to the optimization problem).

Similar techniques can be applied within augmented Lagrangian and modified barrier methods [1].

It is also possible to adapt truncated-Newton techniques to constrained methods based on sequential quadratic programming. At each iteration of such a method, the nonlinear constrained problem is approximated by a quadratic program. An inner iterative method can then be applied to the quadratic program. There are a number of choices in how this is done; for example, an interior-point method could be used to solve the quadratic program.

If the quadratic program is solved using a null-space approach, then the conjugate-gradient method would be applied to a linear system with a matrix of the form

$$Z^{\mathrm{T}}HZ$$

where H is an approximation to the Hessian, and Z is a null-space matrix for the Jacobian of the constraints. Optimality conditions for the constrained optimization problem imply that this matrix will be positive semi-definite at the solution of the optimization problem. Indefiniteness can be dealt with as in the unconstrained case.

The matrix $Z^{\mathrm{T}}HZ$ may be a dense matrix even if H and the constraint Jacobian are sparse. Matrix–vector products involving this matrix should be computed in stages:

$$w_1 = Zv, \quad w_2 = Hw_1, \quad w = Z^{\mathrm{T}}w_2.$$

The convergence of the conjugate-gradient method is enhanced if a preconditioner for $Z^{\mathrm{T}}HZ$ is available. This is more challenging than in the unconstrained case. Since H is an approximation to the Hessian matrix, preconditioners for H will depend on properties of the optimization problem (i.e., this requires input from the user of the software). The null-space matrix Z depends on algorithmic details (i.e., this requires input from the developer of the optimization software). The paper [18] develops preconditioners for $Z^{\mathrm{T}}HZ$ that combine preconditioning information from these two sources.

The quadratic program might be solved by looking at the combined linear system for the variables and multipliers of the quadratic program, which has a matrix of the form

$$\begin{pmatrix} H & A^{\mathrm{T}} \\ A & 0 \end{pmatrix}. \tag{5}$$

H is an approximation to the Hessian of the Lagrangian, and A is the Jacobian matrix of the constraints. The matrix (5) is symmetric but indefinite. An iterative method can be applied to this system, derived from the Lanczos method (see Section 3). One possibility is to use `SYMMLQ` [21], which is designed for symmetric indefinite systems of equations.

It may be difficult to guarantee that the search direction that results is a descent direction. Straightforward tests require factorizations of $Z^{\mathrm{T}}HZ$, where Z is a null-space matrix for A. In an iterative method, where matrices are not stored and matrix factorizations are not available, this is not usually possible.

It is also possible to derive preconditioners based on the structure of the system (5) [10].

11. Conclusions, recommendations

Truncated-Newton methods are a flexible set of methods for solving large optimization problems. They are built upon a sound theoretical foundation. They can be adjusted to achieve a desired asymptotic convergence rate, and they can be designed to limit the waste of over-solving at points far from the solution. They can also be customized to the problem being solved.

The easiest way to use a truncated-Newton method is via the software packages discussed in Section 9. If these are not adequate, then perhaps use of parallel software will help. Customization of the method may also be necessary. The greatest improvements in performance can be obtained by improving the preconditioner. In addition, the forcing sequence can be modified, as can the technique used to compute the matrix–vector product. The references in this paper provide much guidance in these areas.

Acknowledgements

I would like to thank an anonymous referee for many helpful and perceptive suggestions.

References

[1] A.R. Conn, N.M. Gould, P.L. Toint, LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A), Springer, Berlin, 1992.

[2] M.J. Daydé, J.-Y. L'Excellent, N.I.M. Gould, Element-by-element preconditioners for large partially separable optimization problems, SIAM J. Sci. Comput. 18 (1997) 1767–1787.
[3] R.S. Dembo, S.C. Eisenstat, T. Steihaug, Inexact Newton methods, SIAM J. Numer. Anal. 19 (1982) 400–408.
[4] S.C. Eisenstat, H.F. Walker, Globally convergent inexact Newton methods, SIAM J. Optim. 4 (1994) 393–422.
[5] S.C. Eisenstat, H.F. Walker, Choosing the forcing terms in an inexact Newton method, SIAM J. Sci. Comput. 17 (1996) 16–32.
[6] N.I.M. Gould, S. Lucidi, M. Roma, P.L. Toint, Solving the trust-region subproblem using the Lanczos method, SIAM J. Optim. 9 (1999) 504–525.
[7] L. Grippo, F. Lampariello, S. Lucidi, A truncated Newton method with nonmonotone line search for unconstrained optimization, J. Optim. Theory Appl. 609 (1989) 401–419.
[8] C.-J. Lin, J.J. Moré, Newton's method for large bound-constrained optimization problems, SIAM J. Optim. 9 (1999) 1100–1127.
[9] S. Lucidi, F. Rochetich, M. Roma, Curvilinear stabilization techniques for truncated Newton methods in large scale unconstrained optimization, SIAM J. Optim. 8 (1998) 916–939.
[10] L. Lukšan, J. Vlček, Indefinitely preconditioned inexact Newton method for large sparse equality constrained non-linear programming problems, Numer. Linear Algebra Appl. 5 (1998) 219–247.
[11] J.L. Morales, J. Nocedal, Automatic preconditioning by limited memory quasi-Newton updating, Rep. OTC 97/08, Opt. Tech. Ctr., Northwestern Univ., Evanston, IL, 1997.
[12] S.G. Nash, Newton-type minimization via the Lanczos method, SIAM J. Numer. Anal. 21 (1984) 770–788.
[13] S.G. Nash, Preconditioning of truncated-Newton methods, SIAM J. Sci. Statist. Comput. 6 (1985) 599–616.
[14] S.G. Nash, J. Nocedal, A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization, SIAM J. Optim. 1 (1991) 358–372.
[15] S.G. Nash, A. Sofer, Assessing a search direction within a truncated-Newton method, Oper. Res. Lett. 9 (1990) 219–221.
[16] S.G. Nash, A. Sofer, A general-purpose parallel algorithm for unconstrained optimization, SIAM J. Optim. 1 (1991) 530–547.
[17] S.G. Nash, A. Sofer, A barrier method for large-scale constrained optimization, ORSA J. Comput. 5 (1993) 40–53.
[18] S.G. Nash, A. Sofer, Preconditioning reduced matrices, SIAM J. Mat. Anal. Appl. 17 (1996) 47–68.
[19] S.G. Nash, A. Sofer, Linear and Nonlinear Programming, McGraw-Hill, New York, 1996.
[20] S.G. Nash, A. Sofer, On the complexity of a practical interior-point method, SIAM J. Optim. 8 (1998) 833–849.
[21] C.C. Paige, M.A. Saunders, Solution of sparse indefinite systems of linear equations, SIAM J. Numer. Anal. 12 (1975) 617–629.
[22] M. Pernice, H.F. Walker, NITSOL: A Newton iterative solver for nonlinear systems, SIAM J. Sci. Comput. 19 (1998) 302–318.
[23] E. Sachs, Rates of convergence for adaptive Newton methods, J. Optim. Theory Appl. 48 (1986) 175–190.
[24] T. Schlick, Modified Cholesky factorizations for sparse preconditioners, SIAM J. Sci. Comput. 14 (1993) 424–445.
[25] T. Steihaug, The conjugate gradient method and trust regions in large scale optimization, SIAM J. Numer. Anal. 20 (1983) 626–637.