An Iterative Approach to Computing a Direction of Negative Curvature*

Erik Boman† and Walter Murray‡

Copper Mountain Conference, March 1998

Abstract

Any method for minimization that can be shown to converge to a point that satisfies the *second-order* necessary conditions must explicit or implicitly compute a direction of negative curvature of an indefinite matrix. In this paper we investigate computing such a direction using an iterative approach, or by combining direct and iterative methods. While our interest here is motivated by the use of such directions in linesearch methods for optimization the work reported has relevance for other types of optimization methods.

Keywords: optimization, minimization, second-order methods, Newton's method, negative curvature, large-scale problems, Lanczos.

1 Introduction

Given a function $F(x) \in C^2, x \in \mathbb{R}^n$, whose Hessian matrix $\nabla^2 F(x)$ is denoted by H (or $H(x)$ when the argument is relevant), a direction of negative curvature, say d, has the property that

$$d^T H d < 0.$$

Any method for minimization such as a linesearch method [4], trust-region method [4] or a gradient flow approach [5] for which convergence to second-order optimality conditions can be shown must either explicitly or implicitly compute a direction of negative curvature. It is easy to see why. Suppose the current estimate is a point satisfying the first-order but not the second-order conditions. To proceed such algorithms must determine a point in the neighborhood of the saddle point that has a lower value than the current iterate. Such points must lie along a direction of negative curvature. It is also our contention that the relevance of computing

*The material contained in this report is based upon research supported by the National Science Foundation Grant DDI-9500668, the Office of Naval Research Grant N00014-96-1-0274, and the Research Council of Norway Grant 110860/410.

†Scientific Computing and Computational Mathematics Program, Stanford University, Stanford CA 94305-9025 (boman@sccm.stanford.edu).

‡Systems Optimization Laboratory, Department of EES & OR, Stanford University, Stanford, CA 94305-4023 (walter@sol-walter.stanford.edu).

directions of negative curvature grows with the dimension of the problem. This latter hypothesis is based on the probability that a direction of negative curvature occuring by chance decreases significantly with increasing dimension. For example, the probability of a random direction being a direction of negative curvature for a matrix with 99 eigenvalues of 1 and one of -1 is less then 10^{-15}.

In this paper we discuss methods to compute directions of negative curvature when n is large. We wish the effort to compute the direction to be similar to that of computing the direction of descent.

An interesting property of directions of negative curvature that is not possessed by directions of descent is we can define precisely what is meant by a good direction simply by comparing it with the best possible. The best possible is the one that minimizes $d^T H d / d^T d$ and that is given by the eigenvector corresponding to the smallest eigenvalue of H. Clearly the problem of computing a direction of negative curvature is similar to that of computing the eigenvector corresponding to the smallest eigenvalue. It differs in two ways, first we do not need it to be accurate and secondly we may have additional information as a result of computing a direction of descent. Moreover, we shall be concerned with computing a sequence of directions and the matrices that arise typically have very few negative eigenvalues.

A key property that is required to prove convergence to second-order KKT points is that of a *sufficient* direction of negative curvature. Formally, a *sequence* of directions, $\{d_k\}$, is said to be sufficient if any nonzero d_k is a non-ascent direction of negative curvature i.e., $g_k^T d_k \leq 0$, $d_k^T H_k d_k < 0$, where x_k is the kth iterate. Furthermore, the norm of d_k has to be bounded and the curvature has to be sufficient in the sense that

$$\lim_{k \in S} d_k^T H_k d_k = 0 \quad \Rightarrow \quad \liminf_{k \in S} \lambda_{\min}(H_k) \geq 0 \quad \text{and} \quad \lim_{k \in S} d_k = 0, \tag{1}$$

where S is any subsequence. In practice this has come to require that if d_k satisfies $d_k^T d_k = 1$ then

$$d_k^T H_k d_k \leq \beta \lambda_{\min}, \tag{2}$$

where $\beta > 0$ and $\lambda_{\min}$ is the smallest eigenvalue of H_k. In other words relative to the best possible direction of negative curvature d_k is not arbitrarily poor. Computing such a direction is not easy. For example, it has been shown (see [6]) that the method for computing a direction of negative curvature described by Gill and Murray [3] does not have this property. The Gill–Murray algorithm first forms the *modified* Cholesky factorization of H, which satisfies

$$R^T R = H + E,$$

where R is an upper-triangular matrix and $E \geq 0$ is a diagonal matrix. The effort to form this factorization is essentially identical to that of the normal Cholesky algorithm. Moreover, pivoting for numerical stability is *not* required. During the course of the factorization an index s is identified such that d, where

$$Rd = e_s \tag{3}$$

is a direction of negative curvature. It can be shown that (2) is not always satisfied by this direction. Methods do exist for determining a sufficient direction of negative curvature (see [2] and [6]). However, these methods require numerical pivoting. Such pivoting almost always leads to denser factors than R when H is ordered to generate little fill. With pivoting we are also unable to take advantage of the sparsity pattern of the factors being constant at every iteration.

It should be noted that the strictures placed on the direction of negative curvature by the need to prove convergence in a linesearch method while important are not overriding. The likelihood of the modified Cholesky algorithm failing to obtain a direction of sufficient negative curvature is small (since β may be small). Our interest is in obtaining *good* directions of negative curvature, if this can be done routinely. It is our belief that this will enhance the efficiency of minimization algorithm that use such directions.

2 An Iterative Approach

As has been noted the problem we face is similar to that of determining the eigenvector corresponding to the smallest eigenvalue. A natural step is to examine the methodology to determine such a direction. It has long been folk lore in numerical analysis circles that the Lanzcos and similar algorithms quickly determine the eigenvectors corresponding to extreme eigenvalues. Quite how quickly is often ommitted. Calculating only a few extreme eigenvalues and eigenvectors is usually quick compared to determining all the eigenvectors, but it can typically take about $2\sqrt{n}$ iterations [7, p.259]. While that is low compared to n it is more work than we wish to expend.

There are many approaches to determining eigenvectors and we have investigated many of them plus others that are more akin to minimizations algorithms applied to $d^T Hd/d^T d$ or $d^T Hd$, subject to $d^T d = 1$. Here we report on the use of Lanczos, Chebyshev and the two–step Lanczos algorithms with and without the utilization of knowledge of a direction of negative curvature obtained from the modified Cholesky factorization.

The $(s+1)$–step Lanczos method can be shown to be closely related to the s–step conjugate gradient algorithm applied to $d^T Hd/d^T d$ [1, Ch.7]. As a special case, we have that the 2–step Lanczos method is essentially the steepest descent method applied to the Rayleigh quotient. In a talk presented at the 1993 PWACM workshop in Caracas, Murray reported that even one or two iterations of the steepest descent algorithm could dramatically improve a poor direction of negative curvature when a good one exists.

The idea behind our Chebyshev method is as follows. Given an initial vector d_0, we form the iterates $d_j = P_j(H)d_0$ where P_j is a polynomial of degree j. We want a polynomial that is small on $[0, \lambda_{\max}(A)]$, but grows rapidly to the left of the origin. The natural choice for P_j is the Chebyshev polynomial of degree j on $[0, \hat{\rho}]$, where $\hat{\rho}$ is an upper bound on $\lambda_{\max}(A)$. We used $\hat{\rho} = \|H\|_1$ since the 1–norm is inexpensive to compute, but one might consider spending more effort getting a tighter bound, which should give faster convergence.

3 Numerical results

We tested the methods on a variety of randomly generated indefinite matrices. Within an optimization method it is essential to obtain a relatively good direction of negative curvature whenever a good direction exists. When the smallest eigenvalue is close to zero it is often the case that a direction of negative curvature need not be computed. Nonetheless we have included this case since it represents a potentially hard case for the algorithms. Since many iterative methods work extremely well on matrices whose eigenvalues are clustered, such distributions of the eigenvalues were avoided. We expect the hard case for iterative methods to be when the negative eigenvalues are small in magnitude and there are also some small but positive eigenvalues.

We tested a set of $n \times n$ random symmetric indefinite matrices with specified spectra. Each test matrix was of the form $H = Q\Lambda Q^T$ with Q a random orthogonal matrix and Λ a diagonal matrix with t negative elements. We used the following distributions for the eigenvalues:

$$\lambda_i = \begin{cases} -\alpha i/t, & i = 1, \ldots, t \\ (i-t)/(n-t), & i = t+1, \ldots, n \end{cases} \quad \text{(the semi-uniform distribution)}$$

where $0 < \alpha < 1$ is a constant,

$$\lambda_i = \begin{cases} \beta^{i-1}, & i = 1, \ldots, n-t \\ -\beta^{i-1}, & i = n-t+1, \ldots, n \end{cases} \quad \text{(the semi-geometric distribution)}$$

where $0 < \beta < 1$ is a constant. Similar distributions were used in [2].

For each distribution we generated 10 different random matrices corresponding to $t = 1, \ldots, 10$. A new Q was generated for each H. We used the matrix dimension $n = 100$ for most tests. This is not very large, but our matrices are dense since a random orthogonal matrix is dense. Setting up the problems actually took longer time than solving them. We ran a few examples with $n = 400$. For larger n one should generate sparse orthogonal matrices and use a sparse modified Cholesky code.

We compared the three iterative methods described in Section 2: Lanczos, 2–step Lanczos and Chebyshev iteration. All the algorithms were implemented in MATLAB. The Lanczos method was a simple version with no reorthogonalization.

In the figures we show the ratio of the Rayleigh quotient $d_k^T H d_k / d_k^T d_k$ to $\lambda_{\min}(H)$ as a function of the number of iterations j. This ratio should go to 1 as $j \to \infty$. For each method, we plotted the minimum and maximum ratio over the ten trials. If the ratio was negative then the curvature was positive and these data points are not shown in the graphs. The dashed lines are Lanczos, the whole lines are two–step Lanczos, and the dotted lines are Chebyshev.

In Figure 1 we show the semi-uniform distribution with $\alpha = 1$ and $\alpha = 10^{-3}$ starting with a random initial vector. We observe that in the harder case, only Lanczos produces a good direction of negative curvature in a small number of iterations. Figure 2 shows the same examples starting with the direction from modified Cholesky. In this case, all three methods do well.

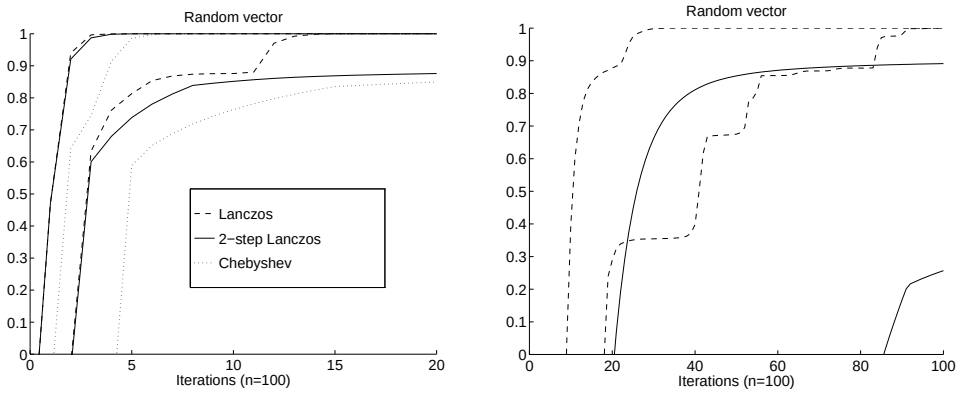

Figure 1: Semi-uniform distribution, $\alpha = 1$ (left) and $\alpha = 10^{-3}$ (right).

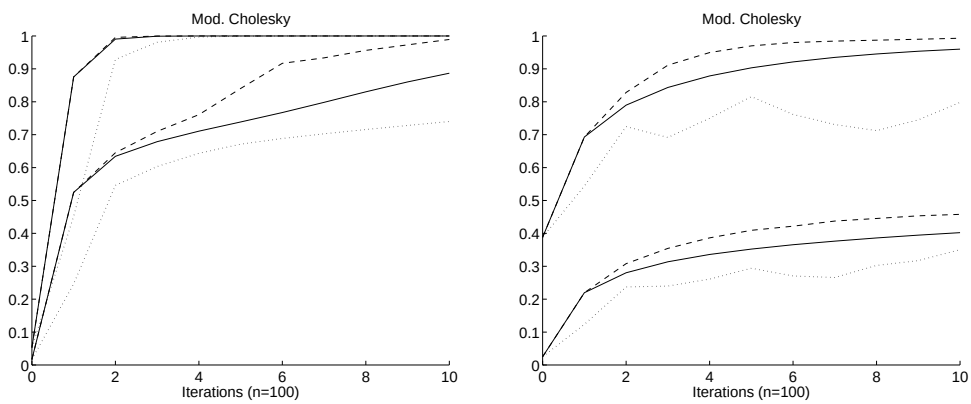

Figure 2: Semi-uniform distribution, $\alpha = 1$ (left) and $\alpha = 10^{-3}$ (right).

In Figure 3 we used the semi-geometric distribution with $\beta = 0.95$ and $\beta = 0.9$. This last example is an extremely hard case for any algorithm since the negative eigenvalues are very close to zero. For even smaller values of β the matrix becomes numerically positive semidefinite in double precision floating-point arithmetic, and eventually modified Cholesky fails to give a direction of negative curvature and our iterative methods also fail.

In Figure 4 we increased the problem size to $n = 400$. Modified Cholesky followed by any of the iterative methods still works well. The number of iterations needed to get good curvature does not change much with increasing size n. We observe that when $\alpha = 1$ we achieve at least 0.4 of optimal curvature after only two iterations of any of the methods tested. But for $\alpha = 10^{-3}$ the ratio is slightly above 0.1 after two iterations and we gain little per iteration after that.

From these experiments we conclude that when starting with a poor initial vector (e.g. a random vector), our problem is very similar to the symmetric eigenvalue problem and we prefer the Lanczos method. But if we already have a direction of negative curvature, any of the iterative methods will give much better negative curvature in a few iterations. We prefer the 2–step Lanczos method to Lanczos since

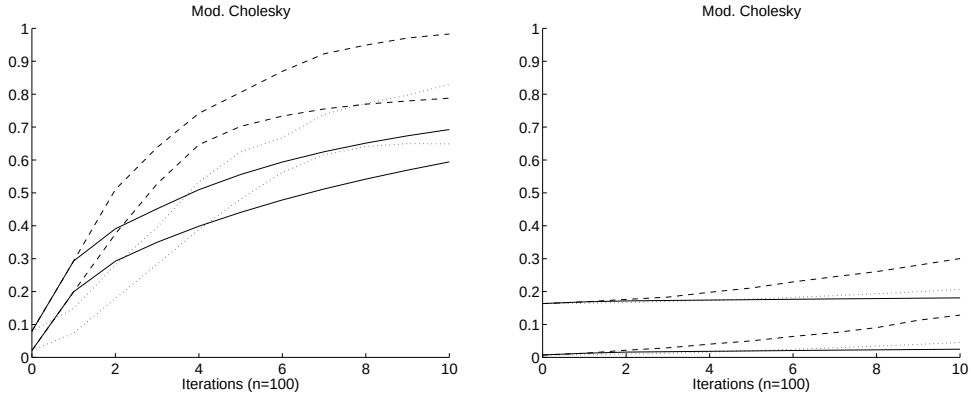

Figure 3: Semi-geometric distribution, $\beta = 0.95$ (left) and $\beta = 0.9$ (right).

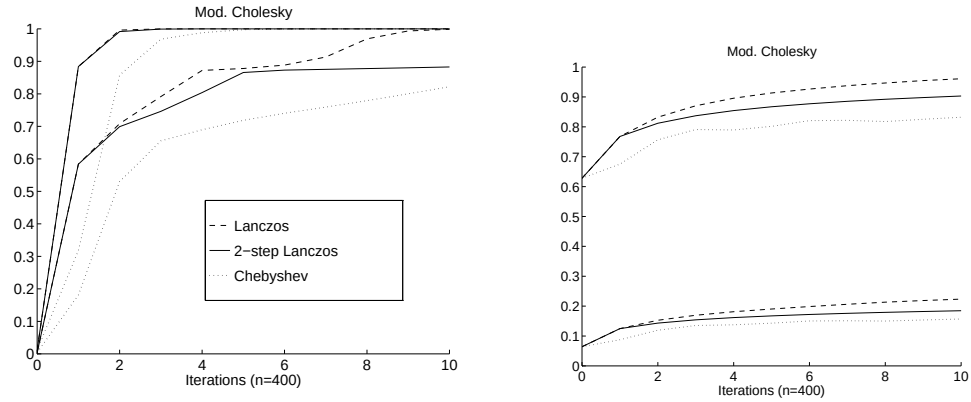

Figure 4: Semi-uniform distribution with $n = 400$, $\alpha = 1$ (left) and $\alpha = 10^{-3}$ (right).

it requires less memory. (An alternative to storing all the Lanczos vectors is to store only the two most recent vectors, but then one must either repeat the whole Lanczos process or do inverse iteration in order to recover the eigenvector.) The Chebyshev iteration is similar in cost to 2–step Lanczos but the Rayleigh quotient is in general not monotonically decreasing, a desirable property, which holds for both the other methods. Consequently, the 2–step Lanczos algorithm looks most attractive as a refinement step in a hybrid direct–iterative algorithm of the methods considered here, while the full Lanczos method may be needed when starting from a random vector.

4 Concluding remarks

We have shown that a dramatic improvement in the quality of a direction of negative curvature obtained from a direct method can be achieved using an iterative procedure. The direct factorization we use is usually available at no extra cost since

it is also needed to find a descent direction. No attempt was made to precondition the iterative procedures using the available Cholesky factorization of $H + E$. For problems with few negative eigenvalues it is typically the case that E is of very low rank, which suggests the Cholesky factorization will be an excellent preconditioner within a method such as the Jacobi-Davidson method [8]. For problems of the size studied here the use of a preconditioner is hardly merited except for pathological examples, but for larger problems that may not be the case. For very large problems, any explicit factorization will require too much storage so a purely iterative method must be used. In this case, an iterative method has normally already been invoked to compute a descent direction and it may be possible to use information from this iterative process when computing a direction of negative curvature.

Acknowledgements

We thank Gene Golub and Andy Wathen for suggesting that we use the Chebyshev iteration.

References

[1] Jane K Cullum and Ralph A Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol.1.* Birkhauser, 1985.

[2] A Forsgren, P E Gill, and W Murray. Computing modified Newton directions using a partial Cholesky factorization. *SIAM J. of Scientific Computing*, 16(1):139–150, 1995.

[3] Philip E Gill and Walter Murray. Newton-type methods for unconstrained and linearly constrained optimization. *Math. Prog.*, 7:311–350, 1974.

[4] Philip E Gill, Walter Murray, and Margaret H Wright. *Practical Optimization.* Academic Press, 1981.

[5] Uwe Helmke. *Optimization and dynamical systems.* Springer-Verlag, 1994.

[6] Jorge J Moré and Danny C Sorenson. On the use of directions of negative curvature in a modified Newton method. *Math. Prog.*, 16:1–20, 1979.

[7] Beresford N Parlett. *The Symmetric Eigenvalue Problem.* Prentice-Hall, 1980.

[8] G L G Sleijpen and H A van der Vorst. A Jacobi-Davidson iteration method for linear eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 17:401–425, 1996.