Large scale unconstrained optimization

M. Roma

Technical Report 10-99 March 1999

To appear on:

Encyclopedia of Optimization
C. Floudas and P. Pardalos editors
Kluwer Academic Publishers

LARGE SCALE UNCONSTRAINED OPTIMIZATION

Massimo Roma

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
via Buonarroti 12
00185 Roma, Italy
Email: roma@dis.uniroma1.it

Abstract

This paper is a survey on the methods for solving large scale unconstrained problems. The most widely used techniques and recent advances in large scale unconstrained optimization are reviewed and specific references for every method described are also given.

Key words: large scale problems, unconstrained optimization

AMS subject classification: 90C06

A large scale unconstrained optimization problem can be formulated as the problem of finding a local minimizer of a real valued function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ over the space $\mathbb{R}^n$, namely to solve the problem

$$\begin{array}{l} \min f(x) \\ x \in \mathbb{R}^n \end{array} \tag{1}$$

where the dimension n is large.

The notion of "large scale" is machine dependent and hence it could be difficult to state a priori when a problem is of large size. However, today an unconstrained problem with more than one thousand variables is usually considered a *large scale problem*.

Besides its own theoretical importance, the growing interest in the last years in solving problems of large size derives from the fact that problems with a larger and larger number of variables are arising very frequently from real world as a result of modeling systems with a very complex structure.

The main difficulty in dealing with large scale problems is the fact that effective algorithms for small scale problems do not necessarily translate into efficient algorithms when applied to solve large problems. Therefore in most cases it is improper to tackle a problem with a large number of variables by using one of the many existing algorithms for the small scale case relying on the growing powerful of the modern computers (see, e.g., [11, 13, 34] for a review on the existing methods for small scale unconstrained optimization).

A basic feature of an algorithm for large scale problems is a low storage overhead needed to make practicable its implementation. Moreover, whenever a large scale problem has some structure it should be exploited to define reliable algorithms; in fact, often the structure of a problem reflects in the sparsity of the Hessian matrix of the function f which can be efficiently exploited.

Methods for unconstrained optimization differ according to how much information on the function f is available. In the framework of large scale unconstrained optimization it is usually required that the user provides at least subroutines which evaluate the objective function and its gradient for any point x. More effective methods can be obtained if second order derivatives are known. When the derivatives are not available they can be obtained by finite difference or by using automatic differentiation. Throughout we assume that the function f is twice continuously differentiable, i.e. that the gradient $g(x) = \nabla f(x)$ and the Hessian matrix $H(x) = \nabla^2 f(x)$ of the function f exist and are continuous. Moreover, we denote by $\|v\|$ the Euclidean norm of a vector $v \in \mathbb{R}^n$.

As in the small scale case, most of the large scale unconstrained algorithms are iterative methods which generate a sequence of points according to the scheme

$$x_{k+1} = x_k + \alpha_k d_k \tag{2}$$

where $d_k \in \mathbb{R}^n$ is a search direction and $\alpha_k \in \mathbb{R}$ is a steplength obtained by means of a one-dimensional search. Obviously, also in large scale optimization it is important that an algorithm presents both the *global convergence* (i.e. convergence of the sequence $\{x_k\}$ towards a stationary point from any starting point) and a good *convergence rate*.

A basic method for solving large scale unconstrained optimization problems can be considered the *steepest descent method* obtained by setting $d_k = -g(x_k)$ in (2). This method is based on the linear approximation of the objective function f and hence only first order information are need. Due to its very limited storage required by a standard implementation, steepest descent method could be considered very attractive in the large scale setting; moreover the global convergence can also be ensured. However, its convergence rate is only linear and therefore it is too slow to be used. A particular rule for computing the stepsize α_k has been proposed [39] and this led to a significant improvement of the efficiency of the steepest descent method.

One of the most effective methods for solving unconstrained problems is the *Newton method*. It is based on the quadratic approximation of $f(x_k + w)$ given by

$$\phi_k(w) = f(x_k) + g(x_k)^T w + \frac{1}{2} w^T H(x_k) w \tag{3}$$

and it is defined by iterations of the form

$$x_{k+1} = x_k + s_k \tag{4}$$

where the search direction s_k is obtained by minimizing the quadratic model of the objective function (3) over $\mathbb{R}^n$. On the one hand, Newton method presents quadratic convergence rate and it is scale invariant, but, on the other hand, in its pure form it is not globally convergent. Globally convergent modifications of the Newton method has been defined following the *linesearch approach* and the *trust region approach* (see, e.g. [11, 12, 27]), but the main difficulty, in dealing with large scale problems, is represented by the possibility to efficiently solve, at each iteration, linear systems which arise in computing the search direction s_k. In fact, the problem dimension could be too large for any explicit use of the Hessian matrix and iterative methods must be used to solve systems of linear equations instead of factorizations of the matrices involved. Indeed, whereas in the small scale setting the Newton direction s_k is usually determined by using direct methods for solving the linear system

$$H(x_k)s = -g(x_k), \tag{5}$$

when n is large, it is impossible to store or factor the full $n \times n$ Hessian matrix unless it is a sparse matrix. Moreover the exact solution, at each iteration, of the system (5) could be too burdensome and not justified when x_k is far from a solution. In fact, since the benefits of using the Newton direction are mainly local (i.e. in the neighbourhood of a solution), it should not be necessary a great computational effort to get an accurate solution of system (5) when $g(x_k)$ is large.

On the basis of these remarks, in [8] the *inexact Newton methods* were proposed. They represent the basic approach underlying most of the Newton-type large scale unconstrained algorithms. The main idea is to approximately solve the system (5) still ensuring a good convergence rate of the method by using a particular trade-off rule between the computational burden required to solve the system (5) and the accuracy with which it is solved. The measure of this accuracy is the relative residual

$$\frac{\|r_k\|}{\|g(x_k)\|} \quad \text{where} \quad r_k = H(x_k)s_k + g(x_k) \tag{6}$$

and s_k is an approximate solution of (5). The analysis given in [8] shows that if the sequence $\{x_k\}$ generated by (4) converges to a point $x_\star$ and if

$$\lim_{k\to\infty} \frac{\|r_k\|}{\|g(x_k)\|} = 0, \tag{7}$$

then $\{x_k\}$ converges superlinearly to $x_\star$. This result is at the basis of the *truncated Newton methods* which represent one of the most effective approach for solving large scale problems. This class of methods was introduced in [9] within the linesearch based Newton-type methods. They are based on the fact that whenever the Hessian matrix $H(x_k)$ is positive definite, to solve the Newton equation (5) is equivalent to determine the minimizer of the quadratic model (3). Therefore, in these methods, a Newton-type direction, i.e. an approximate solution of (5), is computed by applying the (linear) *conjugate gradient method* (CG) [23] to approximately minimize the quadratic function (3). A scheme of a linesearch based truncated Newton algorithm is the following:

LINESEARCH BASED TRUNCATED NEWTON ALGORITHM

OUTER ITERATIONS
For $k = 0, 1, \ldots$
Compute $g(x_k)$
Test for convergence

    INNER ITERATIONS
    (Computation of the direction s_k)
    Iterate CG algorithm until a termination criterion is satisfied

Compute a stepsize α_k by a linesearch procedure
Set $x_{k+1} = x_k + \alpha_k s_k$

A scheme for a truncated Newton algorithm

Given a starting point x_0, at each iteration k, a Newton-type direction s_k is computed by truncating the CG iterates—the inner iterations—whenever a required accuracy is obtained. The definition of an effective truncation criterion represents a key aspect of any truncated Newton method and a natural choice is represented by monitoring when the relative residual (6) is sufficiently small. Moreover, by requiring that $\|r_k\| / \|g(x_k)\| \leq \eta_k$ with $\lim_{k\to\infty} \eta_k \to 0$, the condition given by (7) is satisfied and hence the superlinear convergence is guaranteed [9]. In particular η_k can be chosen to ensure that, as a critical point is approached, more accuracy is required. Other truncation criteria based on the reduction of the quadratic model can be defined [31]. Numerical experiences showed that a relatively small number of CG iterations is needed, in most cases, for obtaining a good approximation of the Newton direction and this is one the main advantage of the truncated Newton methods since a considerable computational savings can be obtained still ensuring a good convergence rate. The performance of the CG algorithm used in the inner iterations can be improved by using a preconditioning strategy based either on the information gained during the outer iterations or on some scaling of the variables. Several different preconditioning schemes have been proposed and tested [29, 40]. Truncated Newton methods can be modified to enable their use whenever the Hessian matrix is not available; in fact, the CG method only needs the product of the Hessian matrix with a displacement vector, and this product can be approximated by finite difference [35]. The resulting method is called *discrete truncated Newton method*. In [41] a Fortran package (TNPACK) implementing a linesearch based (discrete) truncated Newton algorithm which uses a preconditioned conjugate gradient is proposed. However, additional safeguard is needed within truncated Newton algorithms since the Hessian matrix could be not positive definite. In fact, the CG inner iterations may break down before satisfying the termination criterion when the Hessian matrix is indefinite. To handle this case, whenever a *direction of negative curvature* (i.e. a direction d_k such that $d_k^T H(x_k) d_k < 0$) is encountered, the inner iterations are usually terminated and a descent direction (i.e. a direction d_k such that $g(x_k)^T d_k < 0$) is computed [9]. More sophisticated strategies can be applied for iteratively solving the system (5) when it is indefinite [15, 36, 6, 43]. In particular, the equivalent characterization of the linear conjugate gradient algorithm via the *Lanczos method* can be exploited to define a truncated Newton algorithm which can be used to solve problems with indefinite Hessian matrices [28]. In fact, the Lanczos algorithm does not requires the Hessian matrix to be positive definite and hence it enables to obtain an effective Newton-type direction.

A truncated Newton method which uses a *nonmonotone linesearch* (i.e. which does not enforce the monotone decrease of the objective function values) was proposed in [20] and the effectiveness of this approach was shown especially in the solution of ill-conditioned problems. Moreover in the CG-truncated scheme proposed in [20] an efficient strategy to handle the indefinite case is also proposed.

A new class of truncated Newton algorithms for solving large scale unconstrained problems has been defined in [25]. In particular, a nonmonotone stabilization framework is proposed based on a *curvilinear linesearch*, i.e. a linesearch along the curvilinear path

$$x(\alpha) = x_k + \alpha^2 s_k + \alpha d_k \tag{8}$$

where s_k is a Newton-type direction and d_k is a particular negative curvature direction which has some resemblance to an eigenvector of the Hessian matrix corresponding to the minimum eigenvalue. The use of the combination of these two directions enables, also in the large scale case, to define a class of linesearch based algorithms which are globally convergent towards points which satisfy second order necessary optimality conditions, i.e. stationary points where the Hessian matrix is positive semidefinite. Besides satisfying this important theoretical property, this class of algorithms was also shown to be very efficient in solving large scale unconstrained problems [25], [26]. This is also due to the fact that a Lanczos based iterative scheme is used to compute both the directions without terminating the inner iterations when indefinitess is detected and, as result, more information about the curvature of the objective function are conveyed.

Truncated Newton methods have been also defined within the *trust region* based methods. These methods are characterized by iterations of the form (4) where, at each iteration k, the search direction s_k is determined by minimizing the quadratic model of the objective function (3) in a neighbourhood of the current iterate, namely by solving the problem

$$\begin{array}{l} \min \phi_k(s) \\ \|s\| \le \Delta \end{array} \tag{9}$$

where Δ is the trust region radius. Also in this framework most of the existing algorithms require the solution of systems of linear equations. Some approaches are the *dogleg methods* [38] [10] which aim to solve problem (9) over a one-dimensional arc and the method proposed in [5] which solves problem (9) over a two-dimensional subspace. However, whenever the problem dimension is large, it is impossible to rely on matrix factorizations, and iterative methods must be used. If the quadratic model (3) is positive definite and the trust region radius is sufficiently large that the trust region constraint is inactive at the unconstrained minimizer of the model, problem (9) can be solved by using the preconditioned conjugate gradient method [42], [44]. Of course, a suitable strategy is needed whenever the unconstrained minimizer of the quadratic model is no longer lying within the trust region and the desired solution belongs to the trust region boundary. A simple strategy to handle this case was proposed in [42] and [44] and it considers the piecewise linear path connecting the CG iterates, stopping at the point where this path leaves the trust region. If the quadratic model (3) is indefinite, the solution must also lie on the trust region boundary and the piecewise linear path can be again followed until either it leaves the trust region, or a negative curvature direction is found. In this latter case, two possibilities have been considered: in [42] the path is continued along this direction until the boundary is reached; in [44] the minimizer of the quadratic model within the trust region along the steepest descent direction (the *Cauchy point*) is considered. This class of algorithms represents a trust region version of truncated Newton methods and an efficient implementation is carried out within the LANCELOT package [7]. These methods have become very important in large scale optimization, due to both their strong theoretical convergence properties and good efficiency in practice, but they are known to possess some drawbacks. Indeed, they are essentially unconcerned with the trust region until they blunder into its boundary and stop. Moreover, numerical experiences showed that very frequently this untimely stop happens during the first inner iterations when a negative curvature is present and this could deteriorate the efficiency of the method. In order to overcome this drawback an alternative strategy is proposed in [16] where ways of continuing the process once the boundary of the trust region is reached are investigated. The key point of this approach is the use of the Lanczos method and the fact that preconditioned conjugate gradient and Lanczos methods generate different bases for the same *Krylov space*. Several other *large scale trust region methods* have been proposed.

Another class of methods which can be successfully applied to solve large scale unconstrained optimization problems is the wide class of the *nonlinear conjugate gradient methods* [23, 14]. They are extensions to the general (nonquadratic) case of the already mentioned linear conjugate gradient method. They represent a compromise between steepest descent method and Newton method and they are particularly suited for large scale problems since there is never a need to store a full Hessian matrix. They are defined by the iteration scheme (2) where the search direction is of the form

$$d_k = -g(x_k) + \beta_k d_{k-1} \tag{10}$$

with $d_0 = -g(x_0)$ and where β_k is a scalar such that the algorithm reduces to the linear conjugate gradient method if the objective function f is a strictly convex quadratic function and α_k in (2) is obtained by means of an exact linesearch (i.e., α_k is the one-dimensional minimizer of $f(x_k + \alpha d_k)$ with respect to α). The most widely used formulas for β_k are Fletcher-Reeves (FR) and Polak-Ribière (PR) formulas given by

$$\beta_k^{FR} = \frac{\|g(x_k)\|^2}{\|g(x_{k-1})\|^2},$$

$$\beta_k^{PR} = \frac{g(x_k)^T\left[g(x_k) - g(x_{k-1})\right]}{\|g(x_{k-1})\|^2}.$$

Many efforts have been devoted to investigate the global convergence for nonlinear conjugate gradient methods. A widespread technique to enforce the global convergence is the use of a regular *restart* along the steepest descent direction every n iterations obtained by setting $\beta_k = 0$. However, computational experiences showed that this restart can have a negative effect on the efficiency of the method; on the other hand, in the large scale setting, restarting does not play a significative role since n is large and very few restarts can be performed. Global convergence results have been obtained for the *Fletcher-Reeves method* without restart both in the case of exact linesearch [46] and when α_k is computed by means of an inexact linesearch [1]; then, the global convergence was extended to methods with $|\beta_k| \le \beta_k^{FR}$ [14]. As regards the global convergence of the *Polak-Ribière method*, for many years it was proved with exact linesearch only under strong convexity assumptions [37]. Global convergence both for exact and inexact linesearch can also be enforced by modifying the Polak-Ribière method by setting $\beta_k = \max\{\beta_k^{PR}, 0\}$ [14]; this strategy correspond to restart the iterations along the steepest descent direction whenever a negative value of β_k occurs. However, an inexact linesearch which ensures global convergence of the Polak-Ribière method for nonconvex function has been obtained only recently in [21]. As regards the numerical performance of these two methods, extensive numerical experiences showed that, in general, Polak-Ribière method is usually more efficient than the Fletcher-Reeves method. An efficient implementation of the Polak-Ribière method (with restarts) is available as routine VA14 within the Harwell Subroutine Library [22]. The reader can find a detailed survey on the nonlinear conjugate gradient methods e.g. in the paper [34].

Another effective approach to large scale unconstrained optimization is represented by the *limited memory BFGS method* (L-BFGS) proposed in [32] and then studied in [24, 30]. This method resembles the *BFGS quasi-Newton method*, but it is particularly suited for large scale (unstructured) problems because the storage of matrices is avoided. It is defined by the iterative scheme (2) with the search direction given by

$$d_k = -H_k g(x_k)$$

and where H_k is the approximation to the inverse Hessian matrix of the function f at the k-th iteration. In the BFGS method the approximation H_k is updated by means of the BFGS correction given by

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T$$

where $V_k = I - \rho_k y_k s_k^T$, $s_k = x_{k+1} - x_k$, $y_k = g(x_{k+1}) - g(x_k)$, and $\rho_k = 1/y_k^T s_k$. In the L-BFGS method, instead of storing the matrices H_k, a prefixed number (say m) of vectors pairs $\{s_k, y_k\}$ that define them implicitly are stored. Therefore, during the first m iterations the L-BFGS and the BFGS methods are identical, but when $k > m$ only information from the m previous iterations are used to obtain H_k. The number m of BFGS corrections that must be kept can be specified by the user. Moreover, in the L-BFGS the product $H_k g(x_k)$ which represents the search direction is obtained by means of a recursive formula involving $g(x_k)$ and the most recent vectors pairs

$\{s_k, y_k\}$. An implementation of L-BFGS method is available as VA15 routine within the Harwell Subroutine Library [22]. An interesting numerical study of L-BFGS method and a comparison of its numerical performance with the discrete truncated Newton method and the Polak-Ribière conjugate gradient method are reported in [30]. The results of a numerical experience with limited memory quasi-Newton and truncated Newton methods on standard library test problems and on two real life large scale unconstrained optimization applications can be found in [45]. A method which combines the discrete Newton method and the L-BFGS method is proposed in [4] to produce an efficient algorithm able to handle also ill-conditioned problems.

Limited memory quasi-Newton methods represent an adaptation of the quasi-Newton methods to large scale *unstructured* optimization. However, the quasi-Newton approach can be successfully applied to large scale problems with a particular structure. In fact, frequently, an optimization problem has some structure which may be reflected in the sparsity of the Hessian matrix. In this framework, the most effective method is the *partitioned quasi-Newton method* proposed in [18, 19]. It is based on the fact that a function f with a sparse Hessian is a *partially separable function*, i.e. it can be written in the form

$$f(x) = \sum_{i=1}^{n_e} f_i(x)$$

where the element functions f_i depends only on a few variables. Many practical problems can be formulated (or recasted) in this form showing a wide range of applicability of this approach. The basic idea of the partitioned quasi-Newton method is to decompose the Hessian matrix into a sum of Hessians of the element functions f_i. Each approximation to the Hessian of f_i is then updated by using dense updating techniques. These small matrices are assembled to define an approximation to the Hessian matrix of f used to compute the search direction. However, the element Hessian matrices may not be positive definite and hence BFGS formula cannot be used, and in this case a symmetric rank one formula is used. Global convergence results have been obtained under convexity assumption of the function f_i [17]. An implementation of the partitioned quasi-Newton method is available as VE08 routine of the Harwell Subroutine Library [22]. A comparison of the performance of partitioned quasi-Newton, L-BFGS, CG Polak-Ribière and truncated discrete Newton methods is reported in [33].

Another class of methods which has been extended to large sparse unconstrained optimization are *tensor methods* [3]. Tensor methods are based on fourth order model of the objective function and are particularly suited for problems where the Hessian matrix has a small rank deficiency.

To conclude, it is worthy to outline that in dealing with large scale unconstrained problems with a very large number of variables (more than 10^4) high performance computer architectures must be considered. See e.g. [2] for the solution of large scale optimization problems on vector and parallel architectures.

The reader can find the details of the methods mentioned in this brief survey in the specific cited references.

References

[1] M. Al-Baali, Descent property and global convergence of the Fletcher-Reeves method with inexact line search, *IMA Journal on Numerical Analysis*, 5 (1985), pp. 121–124.

[2] B. Averick and J. Moré, Evaluation of large-scale optimization problems on vector and parallel architectures, *SIAM Journal on Optimization*, 4 (1994), pp. 708–721.

[3] A. Bouaricha, Tensor methods for large, sparse unconstrained optimization, *SIAM Journal on Optimization*, 7 (1997), pp. 732–756.

[4] R. Byrd, J. Nocedal, and C. Zhu, Towards a discrete Newton method with memory for large-scale optimization, in *Nonlinear Optimization and Applications*, G. Di Pillo and F. Giannessi, eds., Plenum Publishing, 1995, pp. 1–12.

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

[6] R. Chandra, *Conjugate gradient methods for partial differential equations*, PhD thesis, Yale University, New Haven, 1978.

[7] A. R. Conn, N. I. M. Gould, and P. L. Toint, *LANCELOT: A Fortran package for large-scale nonlinear optimization (Release A)*, Springer Verlag, Heidelberg, Berlin, 1992.

[8] R. Dembo, S. Eisenstat, and T. Steihaug, Inexact Newton methods, *SIAM Journal on Numerical Analysis*, 19 (1982), pp. 400–408.

[9] R. Dembo and T. Steihaug, Truncated-Newton algorithms for large-scale unconstrained optimization, *Mathematical Programming*, 26 (1983), pp. 190–212.

[10] J. Dennis and H. Mei, Two new unconstrained optimization algorithms which use function and gradient values, *Journal of Optimization Theory and Applications*, 28 (1979), pp. 453–482.

[11] J. Dennis and R. Schnabel, A view of unconstrained optimization, in *Handbook in Operations Research and Management Science*, Vol. 1, G. Nemhauser, A. Rinnooy Kan, and M. Tood, eds., North-Holland, 1989, pp. 1–72.

[12] R. Fletcher, *Practical Methods of Optimization*, John Wiley and Sons, New York, 1987.

[13] R. Fletcher, An overview of unconstrained optimization, in *Algorithms for continuous optimization. The state of the art*, E. Spedicato, ed., Kluwer Academic Publishers, 1994, pp. 109–143.

[14] J. Gilbert and J. Nocedal, Global convergence properties of conjugate gradient methods for optimization, *SIAM Journal on Optimization*, 2 (1992), pp. 21–42.

[15] P. E. Gill, W. Murray, D. B. Ponceleon, and M. A. Saunders, Preconditioners for indefinite systems arising in optimization, *SIAM Journal on Matrix Analysis and Applications*, 13 (1992), pp. 292–311.

[16] N. I. M. Gould, S. Lucidi, M. Roma, and P. L. Toint, Solving the trust-region subproblem using the Lanczos method, Tech. Report RAL-TR-97-028, Department for Computation and Information, Rutheford Appleton Laboratory, Oxfordshire, 1997. To appear on *SIAM Journal on Optimization.*

[17] A. Griewank, The global convergence of partitioned BFGS on problems with convex decomposition and Lipschitzian gradients, *Mathematical Programming*, 50 (1991), pp. 141–175.

[18] A. Griewank and P. L. Toint, Local convergence analysis of partitioned quasi-newton updates, *Numerische Mathematik*, 39 (1982), pp. 429–448.

[19] A. Griewank and P. L. Toint, Partitioned variable metric updates for large structured optimization problems, *Numerische Mathematik*, 39 (1982), pp. 119–137.

[20] L. Grippo, F. Lampariello, and S. Lucidi, A truncated Newton method with nonmonotone linesearch for unconstrained optimization, *Journal of Optimization Theory and Applications*, 60 (1989), pp. 401–419.

[21] L. Grippo and S. Lucidi, A globally convergent version of the Polak-Ribière conjugate gradient method, *Mathematical Programming*, 78 (1997), pp. 375–391.

[22] Harwell Subroutine Library, *A catalogue of subroutines*, AEA Technology, Harwell, England, 1998.

[23] M. Hestenes, *Conjugate Direction Methods in Optimization*, Springer Verlag, New York, 1980.

[24] D. Liu and J. Nocedal, On the limited memory BFGS method for large scale optimization, *Mathematical Programming*, 45 (1989), pp. 503–528.

[25] S. Lucidi, F. Rochetich, and M. Roma, Curvilinear stabilization techniques for truncated Newton methods in large scale unconstrained optimization, *SIAM Journal on Optimization*, 8 (1998), pp. 916–939.

[26] S. Lucidi and M. Roma, Numerical experiences with new truncated Newton methods in large scale unconstrained optimization, *Computational Optimization and Applications*, 7 (1997), pp. 71–87.

[27] J. Moré and D. Sorensen, Newton's method, in *Studies in Numerical Analysis*, G. Golub, ed., The Math. Association of America, Washington, DC, 1984.

[28] S. Nash, Newton-type minimization via the Lanczos method, *SIAM Journal on Numerical Analysis*, 21 (1984), pp. 770–788.

[29] S. Nash, Preconditioning of truncated-Newton methods, *SIAM Journal on Scientific and Statistical Computing*, 6 (1985), pp. 599–616.

[30] S. Nash and J. Nocedal, A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization, *SIAM Journal on Optimization*, 1 (1991), pp. 358–372.

[31] S. Nash and A. Sofer, Assessing a search direction within a truncated-Newton method, *Operations Research Letter*, 9 (1990), pp. 219–221.

[32] J. Nocedal, Updating quasi-Newton matrices with limited storage, *Mathematics of Computation*, 35 (1980), pp. 773–782.

[33] J. Nocedal, The performance of several algorithms for large-scale unconstrained optimization, in *Large-scale numerical optimization*, T. Coleman and Y. Li, eds., Philadelphia, 1990, SIAM, pp. 138–151.

[34] J. Nocedal, Theory and algorithms for unconstrained optimization, *Acta Numerica*, (1992), pp. 199–242.

[35] D. O'Leary, A discrete Newton algorithm for minimizing a function of many variables, *Mathematical Programming*, 23 (1982), pp. 20–33.

[36] C. Paige and M. Saunders, Solution of sparse indefinite systems of linear equations, *SIAM Journal on Numerical Analysis*, 12 (1975), pp. 617–629.

[37] E. Polak and G. Ribière, Note sur la convergence de methodes de directions conjugées, *Revue Francaise d'Informatique et de Recherche Operationelle*, 16 (1969), pp. 35–43.

[38] M. Powell, A new algorithm for unconstrained optimization, in *Nonlinear programming*, O. Mangasarian and K. Ritter, eds., New York, 1970, Academic Press, pp. 31–65.

[39] M. Raydan, The Barzilai and Borwein gradient method for large scale unconstrained minimization problems, *SIAM Journal on Optimization*, 7 (1997), pp. 26–33.

[40] T. Schlick, Modified Cholesky factorization for sparse preconditioners, *SIAM Journal on Scientific Computing*, 14 (1993), pp. 424–445.

[41] T. Schlick and A. Fogelson, TNPACK - A truncated Newton package for large-scale problems: I. Algorithm and usage, *ACM Transaction on Mathematical Software*, 18 (1992), pp. 46–70.

[42] T. Steihaug, The conjugate gradient method and trust regions in large-scale optimization, *SIAM Journal on Numerical Analysis*, 20 (1983), pp. 626–637.

[43] J. Stoer, Solution of large linear systems of equations by conjugate gradient type methods, in *Mathematical Programming. The State of the Art*, A. Bachem, M.Grötschel, and B. Korte, eds., Berlin Heidelberg, 1983, Springer-Verlag, pp. 540–565.

[44] Ph. L. Toint, Towards an efficient sparsity exploiting Newton method for minimization, in *Sparse Matrices and Their Uses*, I. S. Duff, ed., London, 1981, Academic Press, pp. 57–88.

[45] X. Zou, I. Navon, K. P. M. Berger, T. Schlick, and F. Dimet, Numerical experience with limited-memory quasi-newton and truncated newton methods, *SIAM Journal on Optimization*, 3 (1993), pp. 582–608.

[46] G. Zoutendijk, Nonlinear programming computational methods, in *Integer and Nonlinear Programming*, J. Abadie, ed., Amesterdam, 1970, North-Holland, pp. 37–86.