A LIMITED MEMORY STEEPEST DESCENT METHOD

ROGER FLETCHER

Dept. of Mathematics, University of Dundee,
Dundee DD1 4HN, Scotland, UK.
e-mail fletcher@uk.ac.dundee.mcs

Edinburgh Research Group in Optimization
Technical Report ERGO 09-014, December 2nd, 2009

Abstract. The possibilities inherent in steepest descent methods have been considerably amplified by the introduction of the Barzilai-Borwein choice of step-size, and other related ideas. These methods have proved to be competitive with conjugate gradient methods for the minimization of large dimension unconstrained minimization problems. This paper suggests a method which is able to take advantage of the availability of a few additional 'long' vectors of storage to achieve a significant improvement in performance, both for quadratic and non-quadratic objective functions. It makes use of certain Ritz values related to the Lanczos process. Some underlying theory is provided, and numerical evidence is set out showing that the new method provides a competitive and more simple alternative to the state of the art l-BFGS limited memory method.

1. Introduction

This paper considers the problem of finding an unconstrained local minimizer $\mathbf{x}^*$ of a given continuously differentiable function $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^n$, where the gradient vector $\mathbf{g}(\mathbf{x})$ of first partial derivatives is available. The study has been motivated by some on-going work concerning a Sequential Linear Programming (SLP) algorithm for large scale Nonlinear Programming (NLP), in which a suitable algorithm is required for carrying out unconstrained optimization in the null space. The algorithm would need to be effective for both small and large numbers of variables, and would need to be matrix-free (in the sense of not storing any potentially large reduced Hessian matrices). Currently the obvious Conjugate Gradient (CG) methods have been used, but these have not proved to be very suitable. In particular, in an SLP algorithm, once the correct active set is identified, the null space basis usually changes smoothly as the solution is approached. However the CG algorithm must be restarted for each SLP iteration, and it is not very convenient to make use of information from the previous SLP iteration.

Also CG algorithms can be very slow, and it would be an advantage if a limited memory approach could be used to make use of a limited number, $\overline{m}$ say, of additional 'long' vectors. The l-BFGS method (Nocedal [16]) is such a method, and has proved to be very successful in the context of unconstrained optimization, but is less convenient for NLP when changes in the dimension or span of the null space basis take place on each SLP iteration.

I have therefore returned to some thoughts that I had some 20 years ago (Fletcher [8]), occasioned by innovative ideas inherent in the Barzilai-Borwein (BB) methods [1]. These are steepest descent methods with a novel choice of the step-length on each iteration, and have proved considerably superior to the largely ineffective classical steepest descent method (Cauchy [3]). Other related choices of step-length have also been proposed since that time (for example [6],[11],[19],[20],[22]). What little is known about the theoretical properties of such methods is largely confined to the case in which $f(\mathbf{x})$ is a quadratic function whose Hessian, A say, is a positive definite matrix. The relevance of the eigenvalues of A to the analysis is pointed out in [8] and [9], and it is suggested that a limited memory approach might be fashioned by using a limited number of eigenvalue estimates, based on certain Krylov sequence properties. However the idea was not taken any further at the time, and it is a version of that idea that is explored in this paper.

The methods that we consider are *steepest descent methods*, that is the generic expression for a new iterate $\mathbf{x}^+$ is

$$\mathbf{x}^+ = \mathbf{x}^c - \alpha_c \mathbf{g}^c \tag{1}$$

in which $\mathbf{g}^c$ refers to $\mathbf{g}(\mathbf{x}^c)$, where $\mathbf{x}^c$ is the current iterate, $\alpha_c > 0$ is a step-length whose choice is determined by the method under consideration, and $-\mathbf{g}^c$ is the *steepest descent direction* at $\mathbf{x}^c$. We recall a result (see for example [7]) that steepest descent methods are invariant under an orthogonal transformation of variables.

The theoretical basis of the methods under consideration derives from the minimization of a positive definite quadratic function, and this is set out in Section 2, where the new concept of a *sweep method* is described. This method might be considered as a generalization of the BB method, insofar as the unmodified BB method corresponds to the case $\overline{m} = 1$. Numerical evidence indicates that a worthwhile improvement over the BB method can be gained by increasing $\overline{m}$. As with the BB method, the basic sweep method is non-monotonic in regard to both the sequences $\{f^c\}$ and $\{||\mathbf{g}^c||\}$. Convergence can be proved using a generalization of the approach given by Raydan [17], and this is set out in the Appendix.

In Section 3, again for a quadratic function, a modification of the basic sweep method is suggested which provides a certain monotonicity property in regard to $\{f^c\}$. Some numerical evidence indicates that these modifications do not slow down the rate of convergence of the algorithm. Further difficulties arise when the algorithm is generalised to minimize a non-quadratic function, and suggestions for dealing with them are made. In Section 5 an implementation of the new algorithm is tested against other common gradient methods, and in particular against the l-BFGS method, with satisfactory results. It is well known that improvements in the performance of CG methods can be obtained for certain types of problem by the use of *preconditioning*. It is likely that similar improvements can be obtained with the sweep methods of this paper. The necessary changes are set out in Section 6, and it is shown that the extra storage and housekeeping demands are modest. Conclusions are drawn in Section 7, along with suggestions for future work.

Unless otherwise specified in the paper, $||\cdot||$ refers to the L_2 vector norm.

2. The Quadratic Case

In this section we consider the case in which $f(\mathbf{x})$ is a positive definite quadratic function. We may also take $\mathbf{x}^*$ to be the zero vector and so express

$$f = \tfrac{1}{2}\mathbf{x}^T A\mathbf{x} \qquad \text{and} \qquad \mathbf{g} = A\mathbf{x} \tag{2}$$

where A is a positive definite matrix.

To analyse the convergence of any gradient method in this case, we can assume without loss of generality that an orthogonal transformation has been made that transforms A to a diagonal matrix $\Lambda = \text{diag}(\lambda_i)$ of its eigenvalues. Moreover, if there are any eigenvalues of multiplicity $m > 1$, then we can choose the corresponding eigenvectors so that the initial (transformed) gradient vector, $\mathbf{g}^1$ say, has $g_i^1 = 0$ for at least $m-1$ of its corresponding components. It follows from (1) and (2) that gradients recur according to

$$\mathbf{g}^+ = \mathbf{g}^c - \alpha_c A\mathbf{g}^c \tag{3}$$

or component-wise, when $A = \Lambda$, as

$$g_i^+ = (1 - \alpha_c\lambda_i)g_i^c \qquad i = 1, 2, \ldots, n \tag{4}$$

and hence, if g_i^c is zero for some i, then this property is preserved by the iteration formula. Also if $\alpha_c = \lambda_i^{-1}$ at any time, then $g_i^+ = 0$ for all subsequent iterations, and such indices have no further effect on the convergence of the algorithm. Thus we can ignore such indices and assume without any loss of generality that Λ has distinct eigenvalues

$$0 < \lambda_1 < \lambda_2 < \ldots < \lambda_n, \tag{5}$$

and that

$$g_i^c \neq 0 \qquad i = 1, 2, \ldots, n \tag{6}$$

on all iterations. In order to fashion a limited memory method, we shall assume on any iteration that in addition to $\mathbf{x}^c$ and $\mathbf{g}^c$, the most recent m back values,

$$G = [\mathbf{g}^{c-m}, \ldots, \mathbf{g}^{c-2}, \mathbf{g}^{c-1}] \tag{7}$$

say, are also available in wrap-around storage, where $m \leq \overline{m}$ is limited to an upper bound $\overline{m}$ on the number of such vectors that can be stored. When n is large, it is assumed that $\overline{m} \ll n$. The resulting method can thus be implemented (also in the non-quadratic case below) with $\overline{m} + 2$ long vectors.

An important property, possessed by all steepest descent methods (1), is that

$$\mathbf{x}^c - \mathbf{x}^{c-m} \in \text{span}\{\mathbf{g}^{c-m}, A\mathbf{g}^{c-m}, A^2\mathbf{g}^{c-m}, \ldots, A^{m-1}\mathbf{g}^{c-m}\}. \tag{8}$$

That is to say, the displacement of the current iterate $\mathbf{x}^c$ from any back value $\mathbf{x}^{c-m}$, lies in the span of the so-called *Krylov sequence* initiated from $\mathbf{g}^{c-m}$. It follows that

$$\mathbf{g}^c - \mathbf{g}^{c-m} \in \text{span}\{A\mathbf{g}^{c-m}, A^2\mathbf{g}^{c-m}, \ldots, A^m\mathbf{g}^{c-m}\}. \tag{9}$$

A remarkable property of this Krylov sequence is that it provides m distinct estimates (so-called *Ritz values*) of the eigenvalues of A, which are contained in the spectrum of A in a certain optimal sense. The theory of this is very extensive, related to the CG and Lanczos methods, and is reviewed for example in Golub

and Van Loan [12]. The Ritz values can be found by running the Lanczos iterative process for m iterations, starting from $\mathbf{g}^{c-m}$. Under the conditions (5) and (6), the Lanczos process would terminate after exactly n iterations, and so for $m \leq n$, these Ritz values exist, and if $m = n$ they are identical to the eigenvalues of A.

In practice, computation of the Ritz values from the Lanczos process requires the matrix A to be available, which will not be the case when we generalise to non-quadratic functions. An alternative way of computing the Ritz values is the following. Given m back gradients as in (7), we can rearrange the equations arising from (3) in matrix form as

$$AG = [G\ \mathbf{g}^c]J \tag{10}$$

where J is the $(m+1) \times m$ matrix

$$J = \begin{bmatrix} \alpha_{c-m}^{-1} & & & \\ -\alpha_{c-m}^{-1} & \ddots & & \\ & \ddots & \alpha_{c-2}^{-1} & \\ & & -\alpha_{c-2}^{-1} & \alpha_{c-1}^{-1} \\ & & & -\alpha_{c-1}^{-1} \end{bmatrix}. \tag{11}$$

It follows that

$$G^T AG = G^T[G\ \mathbf{g}^c]J. \tag{12}$$

If the columns of G are linearly independent, and $G = QR$ are its Gram-Schmidt QR factors, where R is nonsingular and upper triangular, and $Q^TQ = I$, then we may define the matrix

$$T = Q^T AQ = [R\ Q^T\mathbf{g}^c]JR^{-1}. \tag{13}$$

T is readily seen to be upper Hessenberg, and hence tridiagonal and positive definite, since A is symmetric and positive definite. The eigenvalues of T are readily computed, and are the Ritz values referred to above. We denote them by θ_i, $i = 1, 2, \ldots, m$. Because $T = Q^TAQ$, it follows that the Ritz values are contained in the spectrum of A.

Computation of Ritz values in this way requires the computation of QR factors, for example by the modified Gram-Schmidt method. This requires $\sim m^2n$ flops. Also it is required to find storage for the columns of Q which are 'long' vectors. Another alternative is to compute R and $\mathbf{r}$ from the partially extended Choleski factorization

$$G^T[G\ \mathbf{g}^c] = R^T[R\ \mathbf{r}], \tag{14}$$

and substitute $G = QR$ on the left hand side of (12), leading to

$$T = [R\ \mathbf{r}]JR^{-1}. \tag{15}$$

This requires only $\sim \frac{1}{2}m^2n$ flops and no extra long vectors, and is the method that has been used in preparing this paper. However there are issues relating to ill-conditioning and round-off error that have to be considered.

Following from an earlier observation above, if the exact eigenvalues of A were available, and if distinct step lengths $\alpha_c = \lambda_i^{-1}$ were chosen on n successive iterations, then all components of the gradient would have become zero, and the

exact solution would have been obtained. Thus the idea suggests itself of using step lengths which are the inverse of certain of the Ritz values. The BB method is in fact a special case of this method when $m = 1$. When $m > 1$, the question arises as to which Ritz values to choose. A previous idea that I explored of trying to choose a 'best' Ritz value on each iteration did not perform much better than the BB method.

The type of method that is explored in this paper is based on the following basic idea. The sequence of steepest descent iterations is divided up into groups of m iterations, referred to as *sweeps*. At the start of each sweep we denote the current iterate and gradient by $\mathbf{x}^k$ and $\mathbf{g}^k$, and we assume that there are available m Ritz values, $\theta_{k-1,l}$, $l = 1, 2, \ldots . m$ from a previous sweep. Within each sweep we carry out m steepest descent steps

$$\mathbf{x}^{k,l+1} = \mathbf{x}^{k,l} - \alpha_{k,l}\mathbf{g}^{k,l} \qquad l = 1, 2, \ldots, m \tag{16}$$

starting from $\mathbf{x}^{k,1} = \mathbf{x}^k$, and using step lengths $\alpha_{k,l} = (\theta_{k-1,l})^{-1}$. Then $\mathbf{x}^{k+1} = \mathbf{x}^{k,m+1}$. The gradients $\mathbf{g}^{k,l}$, $l = 1, 2, \ldots . m$ obtained on the sweep are then used to calculate Ritz values for the next sweep. There are still some issues to be addressed, namely how to choose step lengths for the first sweep, and how to order the Ritz values within a sweep. We return to these issues below.

In the notation of (1), if θ_c is the current Ritz value being used, then

$$g_i^+ = \Big(1 - \frac{\lambda_i}{\theta_c}\Big) g_i^c. \tag{17}$$

Because $\theta_c \in (\lambda_1, \lambda_n)$, we see that $|g_1|$ is monotonically decreasing, but at a potentially slow rate when θ_c is chosen close to λ_n. If however θ_c is chosen close to λ_1, then $|g_1|$ is considerably reduced, but

$$|g_n^+| = \Big|1 - \frac{\lambda_n}{\theta_c}\Big| |g_n^c| \tag{18}$$

can increase by a factor close to the condition number of A. Thus, as for the BB method, the sequence of gradient norms $\{||\mathbf{g}^k||\}$ is non-monotonic, as also is the sequence of function values $\{f^k\}$. Thus the convergence of the scheme is an issue to be addressed. In fact, Raydan [17] has proved convergence of the BB method for strictly convex quadratic functions, and it is shown in the Appendix that a similar line of argument can be used to establish convergence of the sweep method.

Firstly it is important to establish whether or not the sweep method improves on the BB method ($m = 1$) as the number of back vectors m is increased. In Table 1 the effect is shown of minimizing a quadratic function of 20 variables, with $\lambda_1 = 1$ and the other eigenvalues in geometric progression with ratio $\sqrt{2}$. The gradient components are all initialised to 1, and the calculations are terminated when a value of $||\mathbf{g}^c|| \leq 10^{-6}||\mathbf{g}^1||$ has been achieved. Initialisation and ordering of the Ritz values is carried out as described in the next section. It can be seen that a smooth and worthwhile improvement in the number of sweeps ($\#sw$) and more importantly the number of gradient calls ($\#g$) is obtained as m is increased. However, the method shows some indication of running out of steam at around $m = 7$, which is perhaps a little disappointing, given that for $m = 20$ we might expect to solve the problem with $\#g \approx 40$. A possible reason for this is ill-conditioning in the

TABLE 1. Benefits of increasing m in a sweep method

	Ritz sweep method		CG sweep method	
m	$\#sw$	$\#g$	$\#s$	$\#ls$
1	235	236	4689	4689
2	111	220	1174	2348
3	73	213	125	375
4	48	185	78	312
5	31	143	88	440
6	24	129	66	396
7	23	139	55	385
8	18	119	44	352

computation of R as discussed below. It may in fact be preferable in the quadratic case to compute the Ritz values from the Lanczos process.

The idea of using multiple Ritz values in this way is thought to be new. However a method of Yuan [22] has the property that it terminates finitely for a 2-dimensional quadratic function. Although the step length formula is derived in a quite different way to Equations (10) to (15) above, and looks quite different, I think it must be closely related to the case $m = 2$ here.

An alternative sweep-like method might be to let each sweep be m iterations of the conjugate gradient method, and this is also shown in Table 1. The case $m = 1$ is just the Cauchy steepest descent method, and we know the BB method is considerably superior, as the table shows. Moreover in the case $m = 2$ the CG approach is also considerably inferior and in fact also exhibits some Cauchy-like behaviour. For $m \geq 3$, the number of sweeps improves steadily but not the number of line searches ($\#ls$). In looking for a reason for this, we note that the BB method improves on the Cauchy method by defining its step length from the back value $\mathbf{g}^{k-1}$, rather than from the current value $\mathbf{g}^k$. Perhaps that may be the reason why the sweep method based on back values of the gradient is seen to be superior to the CG sweep method, which essentially uses current values of the gradient.

Another illustration of how the new method improves as m increases is shown in Figure 1. In this case $n = 10$, $\lambda_1 = 1$, and the eigenvalues are in geometric progression with ratio 2. The initial components of the gradient are 1's and m initial Ritz values are specified, which are equally spaced in the interior of the spectrum. The traces show a significant improvement as m increases (note that exact termination would be expected when $m = 10$). Note also the extremely small values of the gradient that are achieved as m increases. Given that the calculations are carried out to a relative precision of 10^{-16} or so, we see that the later iterations are very effective at reducing rounding errors introduced earlier in the calculation. The traces also suggest an interesting conjecture that superlinear convergence in exact arithmetic might be obtained in some cases, possibly if $2m \gtrsim n$. We know from [1] that this is so for the BB method ($m = 1$) when $n = 2$, and Dai and Fletcher [4] give reasons to believe that this is also true for $m = 1$, $n = 3$.

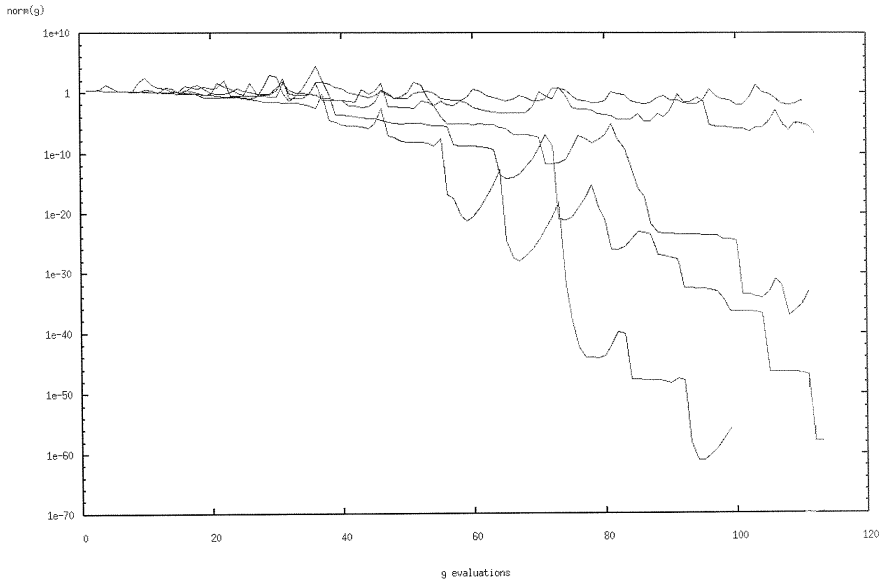


FIGURE 1. Convergence of $||\mathbf{g}||$ for $m = 1,\ 3,\ 5,\ 7,\ 9$

3. A Monotonic Sweep Method for Quadratics

Although the non-monotonic (Ritz) sweep method is effective, the main aim of this paper is to generalize to non-quadratic functions, in which case some attention has to be given to forcing convergence. Moreover, for quadratic programming with box constraints, Dai and Fletcher [5] have given counter examples to show that the BB method can cycle when used in a projection method. So our attention is focussed firstly on deriving a *monotonic* sweep method for minimizing a positive definite quadratic function.

Our key proposal is to select the Ritz values in decreasing order of size, during a sweep. If $m = n$ we know that the method terminates, in which case selecting the Ritz values in this order ensures that both f and $||\mathbf{g}||$ are monotonically decreased (see [8]). In general, this ordering provides step-lengths $\alpha_{k,l}$ that increase in size as the sweep progresses. This gives the best chance that early steps in the sweep reduce $f^{k,l}$ monotonically. However, if a step fails to improve on the value f^k at the start of sweep, then an interpolation is made to take the Cauchy step in that direction, and the sweep is terminated. The overall effect of a sweep is to follow a piecewise linear steepest descent trajectory, somewhat reminiscent of following the trajectory $\dot{\mathbf{x}} = -\mathbf{g}(\mathbf{x})$ with a differential equation solver.

We also terminate a sweep if $f^{k,l}$ improves on f^k, but $||\mathbf{g}^{k,l}|| \geq ||\mathbf{g}^{k,l-1}||$. The reasoning is that higher index components are now growing and the next step in the sweep is likely to fail. Thus we hope to save a possibly unproductive step by this means. An illustration of this is given in Table 2 of Section 4.

The algorithm might be summarized as follows

A Ritz Sweep Algorithm

```
Initialize x^c and the stack of Ritz values
For k = 1, 2, ...
     Denote f^k = f^c
     While the stack is not empty
          Take a Ritz value θ off the stack
          Set α_c = θ^{-1}
          Set x^+ = x^c − α_c g^c
          If f^+ ≥ f^k then
               Reset α_c = g^{cT} g^c / (g^{cT} A g^c)
               Set x^c = x^c − α_c g^c and clear the stack
          Else
               If ||g^+|| ≥ ||g^c|| then clear the stack End
               Set x^c = x^+
          End
     End
     Compute up to m̄ new Ritz values
     Place on the stack in increasing order
End
```

Terminating a sweep early means that fewer than $\overline{m}$ back values of the gradient are available from that sweep. However we can usually add back gradients from a previous sweep. Thus where possible we always use $\overline{m}$ back gradients when computing Ritz values. When $k \leq \overline{m}$ there are not enough back gradients. We allow the user to initialize from 1 up to $\overline{m}$ Ritz values for the first iteration. If only one value is provided, the first and second sweeps have only one (BB) step. For the third sweep there are two back gradients, and if these provide acceptable steps, the fourth sweep has four back gradients, and so on.

The main question at issue is whether imposing monotonicity in this way reduces the effectiveness of the sweep method in the quadratic case. Omitting step lengths derived from small Ritz values might be expected to cause slow convergence of small index components of the gradient to zero. In practice we have not noticed any significant loss of effectiveness. This is supported by the following example with $n = 1000$, $\lambda_1 = 1$, $\lambda_n = 1000$, and eigenvalues in geometric progression. One Ritz value (1001/2) is provided initially. Figure 2 shows traces which correspond to $\overline{m} = 2$, 4, 3, 5, 6 at the right hand edge of the graph, which mostly improve as $\overline{m}$ is increased. Note that $\sqrt{f}$ is plotted against the number of gradient evaluations (not sweeps). (Since $f^* = 0$, we may regard $2\sqrt{f}$ as the weighted norm $||\mathbf{g}||_{A^{-1}}$ of the gradient.) Figure 3 shows the corresponding behaviour of $||\mathbf{g}||$ in the case $\overline{m} = 6$, comparing the monotonic algorithm above to the non-monotonic sweep method of Section 2. We may observe that the monotonic method is in no way inferior to the non-monotonic method. Also, although $||\mathbf{g}||$ is non-monotonic in both cases, the amplitude of the non-monotonicity in $||\mathbf{g}||$ is significantly less when f decreases monotonically.

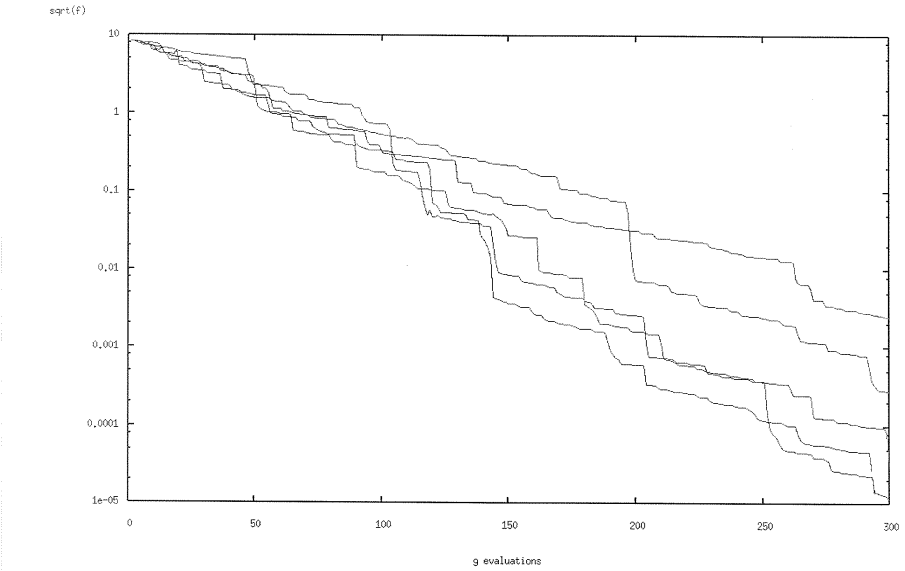
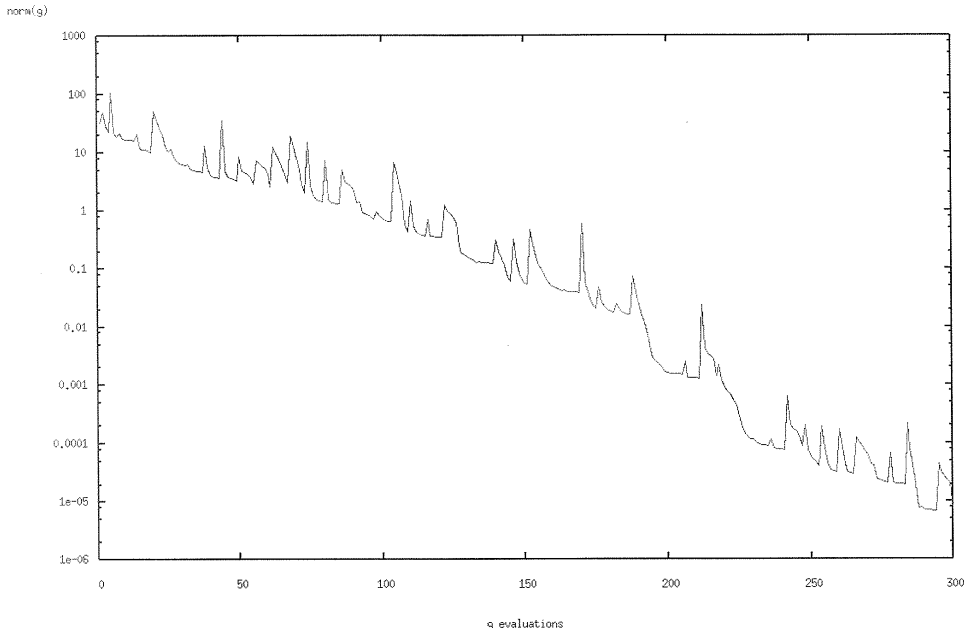


FIGURE 2. Sweep-wise monotonic convergence of f for $\overline{m} = 2, \ldots, 6$

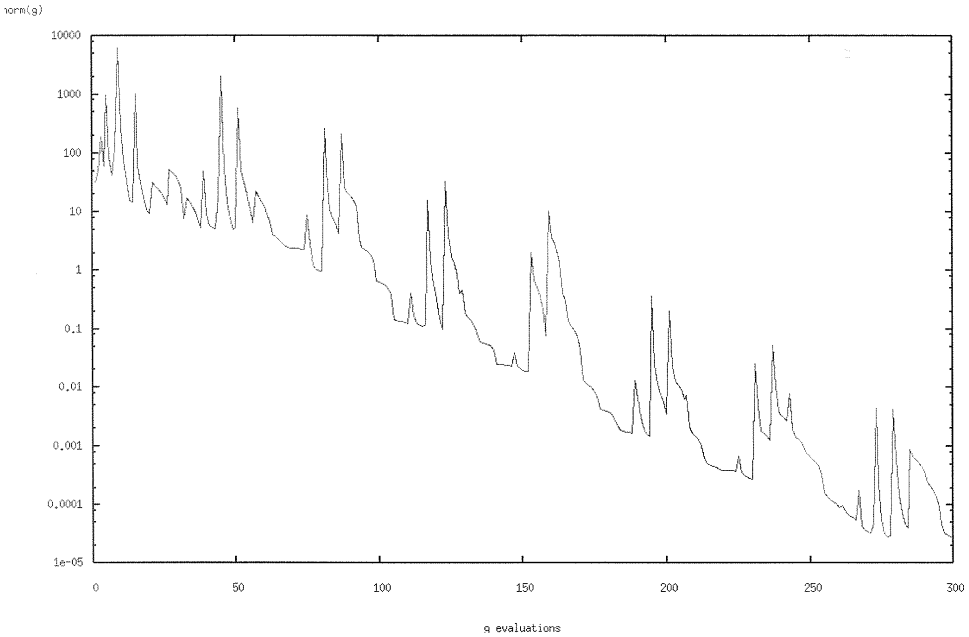
Another issue that must be addressed is possible ill-conditioning or even singularity in the matrix R. In exact arithmetic, if the problem satisfies (5) and (6), then the Lanczos process started from any $\mathbf{g}^k$ always takes n steps to terminate, and R is nonsingular for any $m \leq n$. In may be however that A has multiple eigenvalues, and/or some eigencomponents of the gradient are zero. In this case the Lanczos process terminates in fewer steps, and may not be able to supply $\overline{m}$ Ritz values. For inexact arithmetic, the back gradients can approach linear dependence as m is increased. Then the matrix R, even when computed by modified Gram-Schmidt, can become increasingly ill-conditioned, or even numerically singular. Moreover if R is computed as the Choleski factor of G^TG, then this matrix may become indefinite numerically, and computation of R would fail. In practice it is necessary to monitor the conditioning of R and be prepared to use fewer back values in the computation of R and hence T. For this reason there may be a practical limit to the size of $\overline{m}$, beyond which numerical issues restrict the effectiveness of the scheme. Values of $\overline{m} = 5$, 6 or 7 as in Table 1, seem to be typical.

4. A Monotonic Sweep Method for Non-Quadratic Functions

When the objective function $f(\mathbf{x})$ is non-quadratic, the mechanism for proving convergence set out in the Appendix is no longer valid, and some form of monotonicity is likely to be needed to drive the gradients to zero. Various researchers have recognised this fact. Raydan [18] modifies the BB method in the manner of Grippo et. al. [14], requiring sufficient improvement in f^k over the maximum of the back values f^{k-j}, $j = 1, 2, \ldots, M$. Raydan chooses $M = 10$. If sufficient improvement is not obtained, an Armijo line search is carried out. Dai and Fletcher [6] suggest a method that is different to [14], and possibly more suited to



monotonic f



non-monotonic f

FIGURE 3. Convergence characteristics of $||\mathbf{g}||$ for $\overline{m} = 6$

BB methods, for maintaining monotonicity on a subsequence. Dai and Yuan [6] investigate the long term behaviour of various monotone methods, looking for a certain clustering property in the gradients that is possessed by the BB method. Serafini, Zanghirati and Zanni [20] are able to obtain monotonicity by switching between the two BB methods and the Cauchy step, according to certain criteria. Our approach has been to require monotonic decrease of the sequence $\{f^k\}$ in the

outer iteration sequence, whilst allowing limited increases in $f^{k,l}$ within a sweep, as in the algorithm above.

TABLE 2. Local behaviour of a sweep method for a non-quadratic function

diag(Λ)	78.078	4056.9	16624	70194	495510		
	Moduli of eigencomponents of $\mathbf{g}$						
l	$\|g_1\|$	$\|g_2\|$	$\|g_3\|$	$\|g_4\|$	$\|g_5\|$	f	$\|\|\mathbf{g}\|\|$
1	$2.91e{-4}$	$4.73e{-2}$	$6.9e-2$	$1.9e-1$	$1.3e0$	$2.4e-6$	$1.3e0$
2	$2.85e{-4}$	$4.69e{-2}$	$6.7e-2$	$1.7e-1$	$2.9e-6$	$6.0e-7$	$1.9e-1$
3	$2.84e{-4}$	$4.42e{-2}$	$5.1e-2$	$8.9e-8$	$1.3e-5$	$3.2e-7$	$6.8e-2$
4	$2.83e{-4}$	$3.34e{-2}$	$1.6e-7$	$5.9e-8$	$3.9e-4$	$1.4e-7$	$3.3e-2$
5	$2.77e{-4}$	$5.86e{-7}$	$3.2e-7$	$1.2e-7$	$4.7e-2$	$2.8e-9$	$4.7e-2$
6	$5.52e{-3}$	$1.50e{-2}$	$2.0e-3$	$2.3e-2$	$3.0e2$	$9.1e-2$	$3.0e2$
replace last step with a line search step:							
6	$2.77e{-4}$	$1.06e{-6}$	$3.6e-8$	$9.6e-8$	$2.5e-1$	$6.5e-8$	$2.5e-1$

The benefit of retaining monotonicity is illustrated by the following non-quadratic example. It is an instance of the Trigonometric Test Problem (Fletcher and Powell [10]) with $n = 5$. A solution $\mathbf{x}^*$ is designated as a vector of random numbers, uniformly distributed in $\pm\pi$, and the starting point $\mathbf{x}^1$ is obtained by perturbing components of $\mathbf{x}^*$ by uniformly distributed random numbers in $\pm 10^{-5}$. The eigensolution of the Hessian is calculated at $\mathbf{x}^*$. We see in Table 2 that the condition number $\kappa = 6346$ is quite large, but not unduly so. This amount of information would be expected to provide rapid local convergence from this $\mathbf{x}^1$ with a Newton method, and we would like to see something similar for a sweep method. We use the (orthonormal) eigenvectors to transform the Hessian at $\mathbf{x}^*$ to a diagonal matrix, and the n eigenvalues diag(Λ), suitably ordered, are supplied as the initial Ritz values θ_i for an iteration of the sweep method. Under these conditions, for a quadratic problem, exact and monotonic termination would occur in one sweep.

The actual behaviour of the eigencomponents of the gradient (transformed gradients) on the sweep is shown in Table 2. The first step uses the largest Ritz value λ_5, and drives $|g_5|$ almost to zero, as would be expected from a quadratic model. However there is a small error of $2.9e{-6}$ due to non-quadratic effects. Similarly on step 2, λ_4 is used and $|g_4|$ is almost zeroed, and so on. We also see a lesser reduction in gradients to the left of the one being almost zeroed. However gradients to the right start to increase, and the size of the increase is approximately determined by Equation (17). In particular the increases in $|g_5|$ start to become significant. On the last step, $|g_5|$ has increased to $3.0e2$, resulting in $||\mathbf{g}^2||$ being almost a factor of 200 greater than $||\mathbf{g}^1||$. Moreover, even $|g_1|$ has increased, showing that non-quadratic effects have become dominant. Thus the last non-monotonic step is a disaster as regards providing local convergence. On the other hand, replacing the last step of the sweep by a line search step maintains the good overall progress of the sweep. In fact, because $||\mathbf{g}||$ increases from $l = 4$ to $l = 5$, this is taken as an indication that the next step is likely to be unsatisfactory. Thus in the algorithm above, the sweep would terminate with the $l = 5$ iterate.

There are several issues which must be addressed when generalising to the non-quadratic case, in particular

- the matrix T is necessarily upper Hessenberg, but not usually tridiagonal,
- the matrix A is no longer available to compute the Cauchy step in the algorithm, and
- there are various effects related to the existence of non-positive curvature.

None of these issues admits a single obvious solution, and various possibilities might be followed up.

In regard to the matrix T, it is required to compute some values in a way that reduces to the Ritz values in the quadratic case. Since m is likely to be small, finding all the eigenvalues of T is still an option, but then there is the possibility of complex eigenvalues to be considered. Byrd, Nocedal and Schnabel [2] address similar issues relating to limited memory methods. In practice I have observed that, due to ill-conditioning in R, the product with R^{-1} in (15) most seriously effects the upper triangle of T. Therefore my approach has been to construct a symmetric tridiagonal matrix, $\widetilde{T}$ say, essentially by replacing the strict upper triangle of T by the transpose of the strict lower triangle. Then eigenvalues of $\widetilde{T}$ are used to compute Ritz-like values for the next sweep.

Another issue is that the matrix A is no longer available to compute a Cauchy step. The obvious alternative is to enter some sort of line search when $f^+ \geq f^k$ in the algorithm. One might choose either a combination of Armijo search and interpolation, such as Raydan [19] uses, or a slightly more elaborate search aiming to satisfy Wolfe-Powell conditions (for example Fletcher [7]). I have chosen the latter as it ensures that the resulting $\widetilde{T}$ has at least one positive eigenvalue.

TABLE 3. Trigonometric Test Problem

	$n = 50$			$n = 100$		
	$\#sw/ls$	$\#f$	$\#g$	$\#sw/ls$	$\#f$	$\#g$
lmsd 2	258	658	473	433	1115	786
lmsd 3	199	714	583	356	1281	1042
lmsd 4	86	304	265	152	531	473
lmsd 5	111	429	388	141	558	494
lmsd 6	102	512	451	152	679	611
BFGS	122	145	130	219	236	224
CG PR	558	1515	1074	795	2307	1622
CG FR	319	941	678	628	1888	1337
BB-Raydan	1285	1780	1286	2182	3142	2183
l-BFGS 3	731	816	759	1109	1265	1166
l-BFGS 5	719	776	732	1005	1113	1033

Effects related to non-positive curvature also show up in that the matrix $\widetilde{T}$ may have some negative eigenvalues. Various possibilities suggest themselves. One is to discard the oldest back gradient and recompute a smaller matrix $\widetilde{T}$. Another is simply to put all the Ritz-like values on the stack as before. Then one can either terminate the sweep when a negative Ritz value is found, or carry out a line search

TABLE 4. Convex 2 Test Problem

	$n = 10^3$			$n = 10^5$		
	$\#sw/ls$	$\#f$	$\#g$	$\#sw/ls$	$\#f$	$\#g$
lmsd 2	107	271	213	126	326	250
lmsd 3	64	217	185	74	259	214
lmsd 4	39	165	146	50	218	190
lmsd 5	26	126	114	39	200	177
lmsd 6	29	164	148	34	204	182
BFGS	124	129	126			
CG PR	118	202	194	254	463	402
CG FR	108	221	215	287	387	375
BB-Raydan	172	212	173	260	330	261
l-BFGS 3	132	138	134	210	218	213
l-BFGS 5	117	122	119	232	238	234

before terminating the sweep. Some limited experiments have suggested that there can be a significant difference, and I have preferred the last option.

Finally there is the issue of how best to compute R. Although larger values of m are possible when using modified Gram Schmidt QR factors, the effect of non-monotonicity, or non-quadratic effects, or negative eigenvalues of $\widetilde{T}$, is that fewer than m of these Ritz values may actually be usable. There are also complications in storing Q, if the provision of extra long vectors is to be avoided. Since computing R as the Choleski factor of G^TG is approximately twice as fast, this is the approach that I have taken.

5. Numerical Experience

A Fortran 77 subroutine has been written which implements the ideas of previous sections. It is referred to as `lmsd` (limited memory steepest descent). It is compared with (my) implementations of other standard first derivative methods, on some standard non-quadratic test problems. A more detailed comparison of the limited memory methods lmsd and l-BFGS is also made, including some more difficult CUTEr test problems. The tests are run on a COMPAQ Evo N800v laptop (clock speed 1.3 GHz) under Linux, with optimized code from the Intel F90 compiler.

Other codes tested are the BFGS method, the Polak-Ribiere and Fletcher-Reeves CG methods (see for example [7]), the l-BFGS method (Nocedal [16]) and Raydan's non-monotonic BB method [18]. The CG methods use a fairly accurate line search with a two sided test on the slope ($\sigma = 0.1$ as in [7]). The lmsd, BFGS and l-BFGS methods all use the same Wolfe-Powell line search, with $\sigma = 0.9$ in a one sided test on the slope. Test problems include different instances of the Trigonometric Test Problem (Fletcher and Powell [10]), the Convex 2 problem (Raydan [18]), the Chained Rosenbrock Problem (Toint [21]) starting from $\mathbf{x}^1 = \mathbf{0}$, the Laplace 2 problem (Fletcher [9]) and various CUTEr test problems [13]. All codes use the same termination condition, namely $||\mathbf{g}^c|| \leq \tau||\mathbf{g}^1||$. For Tables 3, 4 and 5, $\tau = 10^{-6}$, and for Table 6, $\tau = 10^{-5}$.

The first set of tests is shown in Tables 3 through 6. A range of values of $\overline{m}$ is tested for the lmsd method. There is also a similar parameter $\overline{m}$ for the l-BFGS method, and standard values $\overline{m} = 3$ and 5 are tested. For l-BFGS, $2\overline{m} + 4$ long vectors are used in my implementation, as against $\overline{m} + 2$ for lmsd. Note that Raydan's BB method and the CG-FR method require 3 long vectors and CG-PR requires 4. Of course, the BFGS method requires $\frac{1}{2}n^2 + O(n)$ locations, so is not practical for large n. However, where applicable, it shows up as the best method or nearly so in the comparisons, as might be expected, and provides a standard of comparison for the other low storage methods.

The first observation that emerges is that overall a worthwhile improvement in the performance of the lmsd method is seen as $\overline{m}$ is increased from 2 up to about 5 or 6, beyond which little if any improvement is obtained. This is in line with what was observed in Section 3, and the reasons why there is a limit on what can be achieved are probably similar.

TABLE 5. Chained Rosenbrock Test Problem

	$n = 50$			$n = 100$		
	$\#sw/ls$	$\#f$	$\#g$	$\#sw/ls$	$\#f$	$\#g$
lmsd 2	1269	3319	2526	1689	4360	3367
lmsd 3	675	2381	1981	1028	3664	3029
lmsd 4	853	3120	2841	938	3712	3301
lmsd 5	558	2272	2067	943	4278	3772
lmsd 6	608	2725	2455	2490	10752	10265
BFGS	249	328	280	483	634	540
CG PR	551	1237	1000	874	1842	1515
CG FR	> 9999			> 9999		
BB-Raydan	> 9999			> 9999		
l-BFGS 3	305	319	308	586	605	589
l-BFGS 5	276	294	281	521	541	525

Turning to the low storage methods, the lmsd method provides the best method by far for the trigonometric problems. These are quite nonlinear, with a quite large condition number and have no special structure such as sparsity or multiple eigenvalues etc. to take advantage of. This outcome may be an indication that lmsd is likely to perform well on hard but not particularly large problems.

In the Convex 2 problem, the objective function is separable, with n distinct eigenvalues in the Hessian at $\mathbf{x}^*$. In terms of gradient counts, lmsd and l-BFGS perform similarly, with a worthwhile improvement over the CG and BB methods. Note however the timings below in Table 7, which are considerably in favour of lmsd. This reflects the more simple housekeeping of the lmsd method when the cost of evaluating f and $\mathbf{g}$ is negligible.

The Chained Rosenbrock problem provides a different picture with lmsd showing up badly relative to l-BFGS, and to CG-PR to a lesser extent. It is difficult to provide any very convincing reason for this. My impression is that, due to the way the function is constructed, the gradient path to the solution (as defined by $\dot{\mathbf{x}} = -\mathbf{g}(\mathbf{x})$) has to follow a succession of steep curved valleys, and it is something

TABLE 6. Non-quadratic Laplacian Test Problem ($n = 10^6$)

	Laplace2 (a)			Laplace2 (b)		
	$\#sw/ls$	$\#f$	$\#g$	$\#sw/ls$	$\#f$	$\#g$
lmsd 2	586	1553	1165	503	1340	1001
lmsd 3	313	1167	911	248	873	728
lmsd 4	165	732	633	193	859	746
lmsd 5	128	702	613	102	515	465
lmsd 6	111	686	626	100	606	557
CG PR	395	753	640*	423	787	675*
CG FR	332	597	521*	435	660	610*
BB-Raydan	1032	1395	1033	1187	1620	1188
l-BFGS 3	495	506	499*	521	528	524
l-BFGSl 5	395	407	400	471	480	474
* failed to achieve a relative improvement of 10^{-5} in $\|\|\mathbf{g}\|\|$						

similar that the lmsd method is doing. Methods which can take large steps may be able to 'jump over' some of the difficulties and hence reach the solution more quickly.

The Laplace2 problem is from a three dimensional p.d.e., with a mildly nonlinear term on the diagonal of the Hessian. An accuracy criterion of $\tau = 10^{-6}$ proved difficult to achieve due to full accuracy in f^* already having been obtained with lower accuracy in $\mathbf{g}$. Thus the termination criterion has been relaxed for this example. The l-BFGS 5 method is a little better on Laplace2 (a), but otherwise there is little to choose, apart from the BB-Raydan method being less successful.

TABLE 7. Comparison of Limited Memory Methods

		lmsd 5			l-BFGS 3		l-BFGS 5	
	n	$\#f$	$\#g$	time	$\#g$	time	$\#g$	time
convex2	10^6	190	168	25.6	217	81.3	218	104.5
Laplace2a	10^6	702	613	100.1	499*	199.8	400	200.9
Laplace2b	10^6	515	465	75.2	524	208.4	474	239.4
SPMSRTLS	10^4	330	299	3.6	322	4.1	288	3.7
NONCVXU2	10^4	9528	8381	53.9	2161	15.1	2198	17.0
NONCVXUN	10^4	13541	11979	77.8	3735	26.3	3732	28.6
MSQRTALS	1024	7491	6590	61.1	7283	63.3	4310	36.3
MSQRTBLS	1024	4751	4183	39.0	5267	46.4	3140	26.8
QR3DLS	610	81777	70217	125.5	330316	575.7	176732	230.0

I think these results provide some evidence that the limited extra storage available to the lmsd and l-BFGS methods does in the main lead to improved performance. The next comparison in Table 7 aims to measure the relative performance of these methods. The test set takes in some additional CUTEr test problems which are quite challenging. Mostly these are least square problems derived from a square system of nonlinear equations. Solving these as least square problems

might be expected to adversely affect the conditioning of the problem. (If A denotes the Jacobian of the equations, then the Hessian includes a term AA^T). Thus, in order to get reasonably accurate solutions in $\mathbf{x}$, the tolerance on the gradient was decreased to $\tau = 10^{-8}$ for the CUTEr problems. The timings given are in seconds. In the table, for the l-BFGS methods, no function counts are given, as these are marginally greater than the number of gradient counts, except for the QR3DLS problem.

I have selected $\overline{m} = 5$ as a reasonable compromise as to what can best be achieved with the lmsd approach. This requires 7 long vectors of storage to implement. The choices of $\overline{m} = 3$ and 5 for l-BFGS are usually recommended, and require 10 and 14 long vectors respectively (at least, in my implementation). Thus the lmsd results are achieved with less storage requirement. These figures are also reflected to some extent in the timings.

The results provide no conclusive outcome either way. The l-BFGS methods mostly do better on the NONCVX and MSQRT problems, but not by more than a factor of about 3. SPMSRTLS is about equal, and the other problems favour lmsd 5, again by a factor of up to 3 or 4. None of the methods fail to solve any of the problems (or those in Tables 3 through 6) in reasonable time.

6. Preconditioning

It is well established that the performance of conjugate gradient methods on certain types of problem can be improved by the use of preconditioning. Preconditioning has also been successfully used by Molina and Raydan [15] to accelerate the BB method. It seems reasonable to expect therefore that preconditioning might be used to advantage in the context of the sweep methods described in this paper. This section briefly sets out what would be involved.

The basis of preconditioning is to make a linear transformation of variables

$$\mathbf{y} = L^T\mathbf{x} \tag{19}$$

in which L is nonsingular, with the aim of improving the spectral distribution of some underlying Hessian matrix A. Also L should be easy to calculate, and solves with L should be inexpensive. Gradients and Hessians then transform according to $\mathbf{g} = \mathbf{g_x} = L\mathbf{g_y}$ and $A = A_\mathbf{x} = LA_\mathbf{y}L^T$ (see for example [7]). The ideal transformation would be that for which $A_\mathbf{y}$ is the unit matrix, showing that in general we choose L such that LL^T is an approximation to A. In our case we would then implicitly carry out the steepest descent method $\mathbf{y}^+ = \mathbf{y}^c - \alpha_c\mathbf{g}^c_\mathbf{y}$ in the transformed variables, which maps into

$$\mathbf{x}^+ = \mathbf{x}^c - \alpha_c L^{-T}L^{-1}\mathbf{g}^c \tag{20}$$

in the $\mathbf{x}$ variables. The step lengths α_c are to be the inverses of Ritz values computed from the Choleski factor of $G^T_\mathbf{y}[G_\mathbf{y}\ \mathbf{g}^c_\mathbf{y}]$, as indicated in (14) and (15). Thus the simplest way to implement the preconditioned sweep method is to store the transformed gradients $\mathbf{g_y} = L^{-1}\mathbf{g_x}$. Assuming that $L\mathbf{g_y} = \mathbf{g_x}$ can be solved in situ (such as when L is triangular) then no additional storage is needed, other than what is needed to store L. An additional solve with L^T is needed to update $\mathbf{x}^c$ as in (20), and this may require an extra long vector to implement.

Alternatively, a symmetric positive definite matrix B which approximates A^{-1} may be available. Then we have $B = L^{-T}L^{-1}$ and the update formula (20) becomes

$$\mathbf{x}^+ = \mathbf{x}^c - \alpha_c B\mathbf{g}^c, \tag{21}$$

and the Choleski factor of $G^T B[G\ \mathbf{g}^c]$ is used to compute Ritz values.

It will be interesting to evaluate the performance of the preconditioned sweep method on problems for which good preconditioners are available, such as when solving certain types of differential equation.

7. Summary and Discussion

The main aim of this project has been to investigate what benefit can be gained in smooth unconstrained minimization from storing a limited number of back values of the gradient vector in a steepest descent method. The approach has been to make use of Ritz values implicit in the Krylov sequence. On the basis of the variety of numerical evidence provided, I feel it is reasonable to conclude that a substantial benefit is available, and that the performance of the resulting method(s) is comparable for large scale systems to what can be obtained from the l-BFGS method. Moreover this is achieved with less extra storage and housekeeping cost.

In applications to non-quadratic problems, it is seen to be important to preserve some sort of monotonicity property in order to be assured of global convergence. The indications are that this does not interfere with the underlying effectiveness of the unmodified method for a quadratic function.

I see sweep methods as being useful in a number of situations. One is in large scale systems of elliptic p.d.e's, both linear and nonlinear, possibly taking advantage of preconditioning. Another is in projection methods for large scale box constrained optimization, both for quadratic and non-quadratic objective functions. Likewise in active set methods for general linearly constrained optimization, it is attractive to have the possibility of both termination or rapid convergence for small null spaces, and yet good performance when the null space is large. Finally, for nonlinear programming, when the null space basis changes slowly from one iteration to the next, the Ritz vectors from one iteration are likely to remain beneficial, and provide a simple and convenient way of carrying forward curvature information from one iteration to the next.

It is a little disappointing that there seems to be a limit to the number of back vectors that can be utilised effectively. It is conjectured in Section 3 that this may be due to numerical loss of rank in the bundle of back vectors, in which case there may be nothing that can usefully be done. Another explanation might be that adequate coverage of the spectrum of A can be achieved by only a few Ritz values. Fortunately the suggested choice of $\overline{m} = 5$ is a not unreasonable value for the number of extra long vectors that might be available in a large scale application.

8. Appendix: A Convergence Theorem

In this appendix a theorem is proved that the basic non-monotonic sweep method of Section 2 converges when applied to minimize a strictly convex quadratic function. It is assumed that the transformations of Section 2 have been carried out, and that (5) and (6) are valid. The theorem follows a similar type of

argument to that of Raydan [17]. First we may usefully prove the following lemma.

Lemma
Let $\{a_k\}$ be a sequence of positive numbers, let $\varepsilon > 0$ be arbitrarily small, and let $c < 1$ and $C \geq 1$ be positive constants. If

$$a_k < \varepsilon \quad \Rightarrow \quad a_{k+2} \leq C a_{k+1} \tag{22}$$

$$a_k \geq \varepsilon \quad \Rightarrow \quad a_{k+2} \leq c a_{k+1} \tag{23}$$

then $\{a_k\}$ converges to zero.

Proof If $a_k \geq \varepsilon$ for all k sufficiently large, then (23) leads to a contradiction. Any group of terms for which $a_k \geq \varepsilon$, $a_{k+1} < \varepsilon$ and $a_{k+2} \geq \varepsilon$ is also excluded, since (23) is again contradicted. Hence terms for which $a_k < \varepsilon$ must occur in groups of two or more. Let $a_{k-1} < \varepsilon$, $a_k < \varepsilon$ and $a_{k+1} \geq \varepsilon$. It follows that $a_{k+1} \leq C\varepsilon$. Moreover, if $a_{k+2} \geq \varepsilon$ then it follows that $a_{k+2} \leq C^2\varepsilon$. However, for any subsequent terms the bound is contracted, until the next term with $a_{k+j} < \varepsilon$ is reached. Thus $a_k \leq C^2\varepsilon$ for all k sufficiently large. Because ε is arbitrarily small, the sequence $\{a_k\}$ converges to zero. QED

We now come to the main theorem. In this we use various bounds on how the components of the gradient propagate. These are all simply derived from the equations

$$g_i^+ = \Big(1 - \frac{\lambda_i}{\theta_c}\Big) g_i^c \qquad i = 1, 2, \ldots, n \tag{24}$$

for a single step, where θ_c is the Ritz value being used, and superscript + denotes the successor.

Theorem
The sequence $\{\mathbf{g}^k\}$ generated by the basic $m-$step sweep method ($m \geq 1$) either terminates at, or converges to, the zero vector.

Proof We need only consider the case that the sequence does not terminate. First we show that $\{g_1^k\}$ converges to zero. It is a property of the Krylov sequence that Ritz vectors lie in the interval (λ_1, λ_n). It follows for a sweep of m steps that

$$|g_1^{k+1}| \leq \Big(1 - \frac{\lambda_1}{\lambda_n}\Big)^m |g_1^k|, \tag{25}$$

and hence $\{g_1^k\}$ converges to zero.

Now let $p > 1$ be the largest integer such that the sequences $\{g_1^k\}$, $\{g_2^k\}$, ..., $\{g_{p-1}^k\}$ all converge to zero. If $p = n + 1$, the theorem is proved, so we consider $p \leq n$ and seek to establish a contradiction. Because $\{g_p^k\}$ does not converge, there exists $\bar{\varepsilon} > 0$ such that for all $\varepsilon \in (0, \bar{\varepsilon}]$, there exists an infinite subsequence S such that $|g_p^k| \geq \varepsilon$, $k \in S$ and $|g_p^k| < \varepsilon$, $k \notin S$. We consider any such value of ε.

Consider the computation of Ritz values for $k \in S$. Because $||\mathbf{g}^k|| \geq \varepsilon$, any accumulation point $\mathbf{q}^\infty$ of the sequence $\mathbf{q}^k = \mathbf{g}^k/||\mathbf{g}^k||$, $k \in S$, has $q_1^\infty = q_2^\infty = \ldots = q_{p-1}^\infty = 0$, so Ritz values computed from $\mathbf{q}^\infty$ would lie in $[\lambda_p, \lambda_n]$. It therefore

follows by continuity of eigenvalues that we can find an iteration number $k_\varepsilon \in S$ for which

$$\theta_{k,l} \in (\tfrac{2}{3}\lambda_p,\, \lambda_n) \qquad l = 1,\, 2,\, \ldots,\, m \tag{26}$$

for all $k \in S$, $k \geq k_\varepsilon$. These Ritz values are used on iteration $k+1$, so it follows for such k that

$$|g_p^{k+2}| \leq c_p^m |g_p^{k+1}|, \tag{27}$$

where

$$c_p = \max\Big(\frac{1}{2},\, 1 - \frac{\lambda_p}{\lambda_n}\Big) < 1. \tag{28}$$

Now we consider the entire sequence $\{g_p^k\}$. Iterations with $k \in S$ have $|g_p^k| \geq \varepsilon$, and provide the bound (27). Iterations $k \notin S$ have $|g_p^k| < \varepsilon$, for which we only have the bound

$$|g_p^{k+2}| \leq C^m |g_p^{k+1}|, \tag{29}$$

where

$$C = \Big|\frac{\lambda_n}{\lambda_1} - 1\Big|. \tag{30}$$

If $C < 1$, it follows immediately that the sequence $\{g_p^k\}$ converges to zero. If $C \geq 1$ we may invoke the above lemma, again showing that the sequence converges. But this contradicts the definition of p. Thus the theorem is proved. QED

Remark
Using bounds derived from (27), it also follows that the intermediate gradients on each sweep converge to zero.

References

[1] J. Barzilai and J. M. Borwein, *Two-point step size gradient methods*, IMA J. Numer. Anal., 8, (1988), pp. 141-148.

[2] R. H. Byrd, J. Nocedal and R. B. Schnabel, *Representations of Quasi-Newton Matrices and their use in Limited Memory Methods*, Mathematical Programming, 63, (1994), pp. 129-156.

[3] A. Cauchy, *Méthode générale pour la résolution des systèms d'équations simultanées,* Comp. Rend. Sci. Paris, 25, (1847), pp. 536-538.

[4] Y. H. Dai and R. Fletcher, *On the Asymptotic Behaviour of some New Gradient Methods*, Mathematical Programming, 103, (2005), pp. 541-559.

[5] Y. H. Dai and R. Fletcher R, *Projected Barzilai-Borwein Methods for Large-Scale Box-Constrained Quadratic Programming*, Numerische Mathematik, 100, (2005), pp. 21-47.

[6] Y.H. Dai and Y. Yuan, *Analysis of monotone gradient methods*, J. Industrial and Management Optimization, 1, (2005), pp. 181-192.

[7] R. Fletcher, *Practical Methods of Optimization.* 2nd Edn., John Wiley, Chichester, 1987.

[8] R. Fletcher, *Low storage methods for unconstrained optimization,* in Computational Solution of Nonlinear Systems of Equations, Eds. E. L. Allgower and K. Georg, Lectures in Applied Mathematics (AMS) 26, (1990), pp. 165-179.

[9] R. Fletcher, *On the Barzilai-Borwein method*, in Optimization and Control with Applications, Eds. L. Qi, K. Teo and X. Yang, Kluwer Academic Publishers, Series in Applied Optimization 96, (2005), pp. 235-256.

[10] R. Fletcher and M. J. D. Powell, *A rapidly convergent descent method for minimization*, Computer J., 6, (1963), pp. 163-168.
[11] A. Friedlander, J. M. Martínez, B. Molina, and M. Raydan, *Gradient method with retards and generalizations*, SIAM J. Numer. Anal., 36, (1999), pp. 275-289.
[12] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd Edition, The Johns Hopkins Press, Baltimore, (1996).
[13] N. I. M. Gould, D. Orban and Ph. L. Toint, *CUTEr (and SifDec), a Constrained and Unconstrained Testing Environment, revisited*, ACM Transactions on Mathematical Software, 29, (2003), pp. 373–394.
[14] L. Grippo, F. Lampariello, and S. Lucidi, *A nonmonotone line search technique for Newton's method*, SIAM J. Numer. Anal., 23, (1986), pp. 707-716.
[15] B. Molina and M. Raydan, *Preconditioned Barzilai-Borwein method for the numerical solution of partial differential equations*, Numerical Algorithms, 13, (1996), pp. 45-60.
[16] J. Nocedal, *Updating quasi-Newton matrices with limited storage*, Mathematics of Computation, 35, (1980), pp. 773-782.
[17] M. Raydan, *On the Barzilai and Borwein choice of steplength for the gradient method*, IMA J. Numer. Anal., 13, (1993), pp. 321-326.
[18] M. Raydan, *The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem*, SIAM J. Optimization, 7, (1997), pp. 26-33.
[19] M. Raydan and B. F. Svaiter, *Relaxed Steepest Descent and Cauchy-Barzilai-Borwein Method*, Computational Optimization and Applications, 21, (2002), pp. 155-167.
[20] T. Serafini, G. Zanghirati and L. Zanni, *Gradient Projection Methods for Quadratic Programs and Applications in Support Vector Machines*, Optimization Methods and Software, 20, (2005), pp. 353-378.
[21] Ph. L. Toint, *Some numerical results using a sparse matrix updating formula in unconstrained optimization*, Mathematics of Computation, 32, (1978), pp. 839-852.
[22] Y. Yuan, *A new stepsize for the steepest descent method*, J. Computational Mathematics, 24, (2006), pp. 149-156.