Enriched Methods for Large-Scale Unconstrained Optimization.

José Luis Morales [*] Jorge Nocedal [†]

January 25, 2000

Abstract

This paper describes a class of optimization methods that interlace iterations of the limited memory BFGS method (L-BFGS) and a Hessian-free Newton method (HFN) in such a way that the information collected by one type of iteration improves the performance of the other. Curvature information about the objective function is stored in the form of a limited memory matrix, and plays the dual role of preconditioning the inner conjugate gradient iteration in the HFN method and of providing a warm start for L-BFGS iterations. The lengths of the L-BFGS and HFN cycles are adjusted dynamically during the course of the optimization. Numerical experiments indicate that the the new algorithms is very effective and is not sensitive to the choice of parameters.

Key words: limited memory method, Hessian-free Newton method, truncated Newton method, L-BFGS, conjugate gradient method, quasi-Newton preconditioning.

[*]Departamento de Matemáticas, Instituto Tecnológico Autónomo de México, Río Hondo 1, Col Tizapán San Angel, México D.F. CP 01000, México. jmorales@gauss.rhon.itam.mx. This author was supported by CONACyT grant 25710-A and by the Asociación Mexicana de Cultura.

[†]ECE Department, Northwestern University, Evanston Il 60208. nocedal@ece.nwu.edu.This author was supported by National Science Foundation grants CDA-9726385 and INT-9416004 and by Department of Energy grant DE-FG02-87ER25047-A004.

1 Introduction

In optimization, as in other areas of scientific computing, much effort is expended in designing algorithms that do not require the setting of parameters. This is a desirable goal, for it frees the user from having to perform trial tests to attempt to discover a good setting of parameters for each application. Hessian-free Newton (HFN) methods for unconstrained optimization are notorious for their sensitivity on the termination rule used in the inner conjugate gradient (CG) iteration. There is a delicate trade-off between the quality of the step and the cost of computing it, and finding termination rules that are efficient in a variety of settings has proved to be very difficult.

In this paper we describe a method for unconstrained optimization that retains some of the key features of Newton-type iterations, and that is both fairly insensitive to the choice of parameters and efficient over a wide range of problems. The method interlaces iterations of the limited memory BFGS method (L-BFGS) and HFN iterations in such a way that the information gathered by one type of iteration improves the performance of the other. Because of this property we call it "enriched method", and the goal of this paper is to show that it constitutes a promising new approach for large-scale optimization. One of the salient features of the enriched method is the use of limited memory matrices to provide both a preconditioner for the CG iteration in the HFN step and warm-start information for the L-BFGS iterations.

2 Motivation and Outline of the New Method

The problem addressed in this paper consists of the unconstrained minimization of a function of a large number of variables,

$$\text{minimize} \quad f(x), \qquad f : \mathbf{R}^n \to \mathbf{R}. \tag{2.1}$$

We will assume that the gradient g of f is available but that computing second derivatives is not possible. Two of the most effective algorithms for these types of problems are: (i) Hessian-free inexact Newton methods (HFN) [13, 10, 3], which are often called "truncated Newton methods"; and (ii) limited memory quasi-Newton methods, such as L-BFGS [11, 5, 7].

Inexact Newton methods that use the exact Hessian matrix and employ a preconditioned CG iteration to compute the step have proved to be very effective in the solution of large problems; see, for example, Lin and Moré [6] and Conn, Gould and Toint [4]. Hessian-free methods cannot make use of the best preconditioning techniques since these require direct access to the Hessian matrix, but they can benefit from the limited memory quasi-Newton preconditioners described in [8].

In this paper we take the use of limited memory matrices one step further. We argue that it is not economical to compute HFN steps at every iteration, and that it can be advantageous to interlace them with L-BFGS steps. The key is to save, in the form of a limited memory matrix, some of the information generated by the inner CG iteration of the HFN method, and use this matrix to improve the quality of the L-BFGS iterations. In this manner we will be able to often bypass the expensive HFN steps and reuse some of the valuable information they have collected.

Another way of motivating our approach is to note that the strengths and weaknesses of the HFN and L-BFGS methods are complementary. The HFN method normally requires much fewer iterations to approach the solution, but the effort invested in one iteration can be very high and the curvature information gathered in the process is lost after the iteration is completed. The L-BFGS method, on the other hand, performs inexpensive iterations, but the quality of the curvature information it gathers can be poor, and as a result it can be very inefficient on ill-conditioned problems. Enriched methods aim to combine the best features of both methods in an dynamic manner.

In the enriched method that will be tested below, l steps of the L-BFGS method are alternated with t steps of the HFN method, where the choice of l and t will be discussed below. We illustrate this as

$$\left[\, l * (\text{L-BFGS}) \overset{H(m)}{\longrightarrow} t * (\text{HFN(PCG)}) \overset{H(m)}{\longrightarrow} \text{ repeat } \right].$$

During the cycle of L-BFGS iterations, a limited memory matrix $H(m)$ is updated, where m denotes the number of correction pairs stored. The matrix obtained at the end of this cycle is used to precondition the first of the t HFN iterations. During each of the remaining $t-1$ HFN iterations, the limited memory matrix $H(m)$ is updated using information generated by the inner preconditioned conjugate gradient (PCG) iteration, and is used to precondition the next HFN iteration; see [8] for a detailed description of the preconditioning process. Once the t HFN steps have been executed, the most current matrix $H(m)$ is used as the initial limited memory matrix in the new cycle of L-BFGS steps. The process continues in this manner, alternating cycles of L-BFGS and HFN iterations, and transmitting curvature information from one cycle to the next.

Clearly, the L-BFGS and HFN methods are particular cases of the enriched method, since they are obtained by setting $t = 0$ and $l = 0$, respectively. In our implementation of the enriched method, the lengths of the cycles, l and t, are chosen dynamically as the optimization process takes place. A detailed description of the algorithm will be given in section 3.

An earlier attempt to develop an enriched method is described in [2]. The results presented in that paper indicate that it is beneficial to transmit curvature information from the HFN iteration to the L-BFGS iteration, and they hint at the potential of enriched methods. The algorithm used in [2] was, however, quite rigid:

(a) 20 L-BFGS iterations were followed by one HFN iteration ($l = 20$, $t = 1$);

(b) the HFN steps were computed using an unpreconditioned CG iteration;

(c) to avoid the difficulties of choosing a good stopping test for the CG iteration, the algorithm performed always 5 CG iterations to compute the HFN step; the only exception was when negative curvature was encountered, in which case the CG iteration terminated immediately.

That simple algorithm was useful for testing some of the features of enriched methods, but was not versatile enough to perform well over a large set of test problems, such as the CUTE [1] collection.

In this paper we present a sophisticated implementation of an enriched method which includes preconditioning of the CG iteration, a dynamic strategy for determining the lengths of the L-BFGS and HFN cycles, and a standard stopping test for the CG iteration. Depending on the characteristics of the optimization problem, the enriched method may resemble the L-BFGS or HFN methods, or it may be a hybrid that combines the features of each method to a degree that varies during the optimization process.

An important advantage of our implementation is that the enriched method is not very sensitive to the choice of termination test in the CG iteration. The only parameter that must be selected is the number m of correction pairs stored in the limited memory matrices $H(m)$. Our tests indicate that, as in the L-BFGS method, the performance of the enriched method varies gradually with the choice of m.

3 Description of the Algorithm

We first show that the L-BFGS and enriched iterations can formally be viewed as a HFN iteration. This framework will facilitate the description and implementation of enriched methods.

Let us consider an inexact Newton-type iteration for solving problem (2.1) given by

$$x_+ = x + \alpha p, \tag{3.2}$$

where α is a steplength and the search direction p is an approximate minimizer of the quadratic model

$$q(p) = f(x) + p^T g(x) + \frac{1}{2} p^T B p. \tag{3.3}$$

Here g denotes the gradient of the objective function f, and B is some symmetric and positive definite matrix. The approximate minimization of the quadratic q is performed by the CG method. In this paper we assume that, if negative curvature is encountered, the CG iteration terminates immediately without exploring this negative curvature direction.

A Hessian-free inexact Newton method is a particular instance of this method in which B is intended to be the Hessian $\nabla^2 f(x)$, but is not computed explicitly. All products of the form Bv are either approximated by finite differences,

$$Bv \approx \frac{g(x+\tau v) - g(x)}{\tau},$$

where τ is a small parameter, or are computed by automatic differentiation techniques. There is no consensus on the best termination test for the CG iteration, and various rules are used in practice [10, 14]. Preconditioning can be performed using (limited memory) quasi-Newton matrices, as described in [8]. This HFN method with quasi-Newton preconditioning will be central to the derivation that follows.

The L-BFGS iteration is a special case of the preconditioned HFN iteration, in which only one CG iteration is allowed in the minimization of the model (3.3). The enriched method will also be viewed as a special case of the preconditioned HFN method in which the number of CG iterations varies during the course of the optimization calculation. Following is a broad outline of the new algorithm.

ENRICHED ALGORITHM

```
Choose a starting point $x$, the memory parameter $m$, and an initial choice of the
length $l$ of the L-BFGS cycle; set method $\leftarrow$ 'L-BFGS'; first $\leftarrow$ .true.

While a convergence test is not satisfied:

    Repeat
        compute $p$: call PCG ( $p$, method, status, maxcg );
        compute $\alpha$: call LNSRCH ( $\alpha$ );
        compute $x_+ = x + \alpha p$;
        store $s = x_+ - x$ and $y = g_+ - g$;
        call ADJUST ( $l$, $t$, $\alpha$, method, status, first, maxcg );
    End repeat

End while.
```

The procedure `PCG` implements the preconditioned CG iteration. Its input parameters are `method`, which can have the values 'L-BFGS' or 'HFN', and `maxcg`, which determines the maximum number of CG iterations allowed. This procedure returns a search direction p, and the value of `status`, which will be used to modify the lengths l and t of the L-BFGS and HFN cycles. The procedure `LNSRCH` is a standard backtracking line search routine enforcing the Wolfe conditions (cf. [12]).

The procedure `ADJUST` is invoked at every iteration of the enriched algorithm, regardless of the value of `method`. When `method` = 'L-BFGS', the procedure simply increases a variable k that counts the number of L-BFGS steps performed in the current cycle, and if $k \geq l$, it also sets `method` = 'HFN'. If `method` = 'HFN', `ADJUST` takes various actions based on the quality of the current and previous Newton steps. A Newton step is considered "profitable" if the step size α associated with the direction p lies in the interval $[0.8, 1]$. We distinguish two cases of unprofitable iterations: a) the step size is relatively small ($\alpha < 0.8$); b) the CG iteration detected an indefinite Hessian. `ADJUST` keeps a record of the number of consecutive profitable Newton steps in the variable `profit`, and uses this number to update the lengths l and t of the L-BFGS and HFN cycles.

We now list the situations in which `ADJUST` modifies the lengths of these cycles.

1. *Indefinite Hessian.* If the CG iteration generates a direction of negative curvature we judge that we are in a region where L-BFGS steps are to be preferred over HFN steps. We therefore reset $t \leftarrow 1$, increase l by 1, and set `method`='L-BFGS'.

2. *Small steps in HFN iteration.* If $\alpha < 0.8$ in a HFN iteration, the iterates do not appear to have reached the region where a Newton-type iteration is rapidly convergent. In this case we set $t = \max\{2, t-1\}$, and define `method`='L-BFGS'.

3. *A full cycle of t successful Newton steps was completed.* Our experience indicates that once the algorithm has reached the region where Newton's method is rapidly convergent, it is advisable to take as many HFN as is economically possible. Therefore in this case we increase t by one.

4. *At least 2 successful Newton iterations were performed in the cycle.* We use the variable `force2` to ensure that at least two HFN iterations are computed in succession, regardless of the outcome of the first iteration. This variable is introduced because the full benefit of limited memory preconditioning is obtained only if more than one HFN iteration is performed in succession.

   If the cycle of HFN steps is at least moderately successful, in that 2 or more Newton iterations were profitable, then we set `force2` to the value .true.

During the very first HFN iteration in the algorithm, our CG stopping test may not be appropriate, and we adopt the cautious strategy of allowing a maximum of 5 CG iterations. Also, t is initially set to two, and `force2` is set to .false. Subsequent calls to `ADJUST` reset `maxcg` to its default value.

We can now provide a description of the procedure `ADJUST`.

```
PROCEDURE ADJUST ($l$, $t$, $\alpha$, method, status, first, maxcg )

 $k \leftarrow k + 1$

 if  method = 'L-BFGS'  then

     if  $k \geq l$  then              % Reset counters, switch to HFN and leave
         if  first = .true.  then maxcg $\leftarrow$ 5; $t \leftarrow 2$; force2 $\leftarrow$ .false.
         method $\leftarrow$ 'HFN'; $k \leftarrow 0$; profit $\leftarrow$ 0;
     end if

 else                                  % Examine quality indicators of the Newton step

     if  status = 'Indefinite Hessian'  then
         $t \leftarrow 1$; force2 $\leftarrow$ .false.; $l \leftarrow \min\{(3/2)l,\ 30\}$;
         method $\leftarrow$ 'L-BFGS'; $k \leftarrow 0$; return
     end if
     if  $\alpha \geq 0.8$  then profit $\leftarrow$ profit + 1
     else
         if  force2 = .true. and $k = 1$  then return
         else
             $t \leftarrow \max\{2,\ k - 1\}$
             method $\leftarrow$ 'L-BFGS'; $k \leftarrow 0$; return
         end if
     end if
     if  $k \geq t$  then              % Check if the cycle is complete
         if  profit = $k$  then $t \leftarrow t + 1$
         if  profit $\geq 2$  then force2 $\leftarrow$ .true. else force2 $\leftarrow$ .false. end if
         method $\leftarrow$ 'L-BFGS'; $k \leftarrow 0$; return
     end if

 end if

 return
```

4 Numerical Experiments

We tested the enriched method on all the large unconstrained problems in the CUTE collection [1]. We set the initial values $m = 20$ and $l = 15$. The CG inner iteration was stopped when one of the following conditions was satisfied:

(a) 30 iterations were performed (in the very first iteration only 5 CG iterations were allowed).

(b) $||\hat{r}|| \leq \eta ||\hat{g}||$, where $\eta = 1/10$, and $\hat{r}$ and $\hat{g}$ are defined as follows

$$\hat{r} = H(m)r, \qquad \hat{g} = H(m)g.$$

The vector r stands for the CG residual, $r = -g - \nabla^2 f(x)p$, and $H(m)$ is the preconditioner.

We also tested two HFN methods that differ only in their stopping rule for the inner CG iteration. HFN1 uses the rule described by Nash [10], whereas HFN2 terminates when the CG residual at the j-th iteration of the optimization algorithm satisfies

$$||r|| \leq \eta_j ||g_j||, \quad \eta = \min(0.5/j, ||g_j||),$$

which is one of the rules discussed in [3]. The inner CG iteration of methods HFN1 and HFN2 is preconditioned using the same quasi-Newton preconditioner as in the enriched method.

All tests were performed on a DEC Personal Workstation with 512 Mb of main memory, using double precision FORTRAN. The optimization calculation was terminated for the three methods (Enriched, HFN1 and HFN2) when

$$||g_j||_2 / \max(1, ||x_j||_2) \leq 10^{-5}.$$

The results are given in Tables 1 and 2.

PROBLEM	n	Enriched	HFN1	HFN2
ARWHEAD	1000	11	11	8
BDQRTIC	100	42	330	304
BROYDN7D	1000	416	723	741
BROWNAL	10	18	48	18
BRYBND	1000	47	101	54
CRAGGLVY	1000	86	136	158
CHAINWOO	1000	5056	8368	9089
COSINE	1000	16	21	22
DIXMAANA	1500	13	22	19
DIXMAANB	1500	12	20	19
DIXMAANC	1500	13	22	21
DIXMAAND	1500	16	24	23
DIXMAANE	1500	192	338	337
DIXMAANF	1500	162	250	249
DIXMAANG	1500	165	248	247
DIXMAANH	1500	166	246	245
DIXMAANI	1500	2602	4754	4753
DIXMAANK	1500	1437	2672	2687
DIXMAANL	1500	1771	2612	2613
DQDRTIC	1000	18	17	18
DQRTIC	500	36	53	50
EDENSCH	2000	34	49	48
EIGENALS	110	121	1317	792
EIGENBLS	110	885	700	626
EIGENCLS	462	1458	3521	3519
ENGVAL1	1000	18	38	35
FLETCBV3	1000	18	4	22
FLETCHCR	100	679	2133	2112
FMINSRF2	1024	190	526	566
FMINSURF	1024	297	691	640

Table 1: Number of function/gradient evaluations for three optimization methods.

Our results indicate that the enriched method is significantly more efficient than the two HFN methods. We tried several other variants of the HFN method, using different stopping rules for the CG iteration and other strategies for handling negative curvature, but their performance was not better than that of methods HFN1 and HFN2. As pointed above, we observed that the the enriched method is not very sensitive to the stopping test of the CG iteration; we experimented with several variants and obtained similar results.

In conclusion the numerical results suggest that the enriched method should be considered as a serious competitor to Hessian-free Newton methods

References

[1] I. Bongartz, A.R. Conn, N.I.M. Gould, and Ph.L. Toint, *CUTE: Constrained and unconstrained testing environment*, ACM Trans. Math. Software, 21 (1995), pp. 123–160.

[2] R.H. 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, 1996, pp. 1–12.

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

[4] A.R. Conn, N.I.M. Gould, and Ph.L. Toint, *LANCELOT: A Fortran package for large-scale nonlinear optimization (Release A)*, no. 17 in Springer Series in Computational Mathematics, Springer-Verlag, New York, 1992.

[5] J.C. Gilbert and C. Lemaréchal, *Some numerical experiments with variable storage quasi-Newton algorithms*, Math. Programming, 45 (1989), pp. 407–436.

[6] C.-J. Lin and J.J. Moré *Newton's method for large bound-constrained optimization problems*, SIAM J. Optim. 9 (1999), pp. 1100–1127.

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

[8] J.L. Morales and J. Nocedal, *Automatic Preconditioning by Quasi-Newton Updating.* Technical Report OTC 97/08. Optimization Technology Center. Northwestern University, Evanston, IL, 1997.

[9] J.L. Morales and J. Nocedal, *Algorithm PREQN: Fortran Subroutines for Preconditioning the Conjugate Gradient Method.* Technical Report OTC 99/02. Optimization Technology Center. Northwestern University, Evanston, IL, 1999.

[10] S.G. Nash, *Preconditioning of truncated-Newton methods*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 599–616.

PROBLEM	n	Enriched	HFN1	HFN2
FREUROTH	1000	47	96	69
GENROSE	500	1494	2713	2612
GENHUMPS	1000	3225	5433	5432
LIARWHD	1000	24	34	39
MOREBV	1000	83	119	125
MSQRTALS	1024	1918	3461	3715
MSQRTBLS	1024	1585	3041	3293
NCB20B	1000	824	6169	7714
NCB20	1010	361	831	835
NONCVXU2	1000	1496	2187	2260
NONCVXUN	1000	1856	4567	5350
NONDQUAR	100	260	3339	3271
PENALTY1	1000	80	103	104
PENALTY2	100	71	147	147
PENALTY3	100	87	147	166
POWELLSG	1000	32	3276	108
POWER	1000	140	562	320
QUARTC	1000	38	62	53
SCHMVETT	1000	46	65	67
SINQUAD	1000	189	308	474
SPARSINE	1000	3566	7464	7747
SPARSQUR	1000	28	41	48
SPMSRTLS	1000	120	212	201
SROSENBR	1000	18	27	20
TOINTGSS	1000	19	39	38
TQUARTIC	1000	26	9	42
TRIDIA	1000	535	672	673
VARDIM	100	37	73	52
WATSON	31	54	565	429
WOODS	1000	117	448	250
TOTAL		34371	76224	75689

Table 2: Number of function/gradient evaluations for three optimization methods.

[11] J. Nocedal, *Updating quasi-Newton matrices with limited storage*, Math. Comput., 35 (1980), pp. 773-782.

[12] J. Nocedal and S.J. Wright, *Numerical Optimization*, Springer Series in Operations Research, New York, 1999.

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

[14] T. Schlick and A. Fogelson, *TNPACK–a truncated Newton minimization package for large-scale problems II. Implementation, examples* ACM Trans. Math. Software, 18 (1992), pp. 71–111.