

Enriched Methods for Large-Scale Unconstrained Optimization.

José Luis Morales * Jorge Nocedal †

September 21, 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 an initial matrix 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 new algorithms are both effective and 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 good parameter settings for each application. Inexact Newton 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 uses only first derivative information. It interlaces iterations of the limited memory BFGS method (L-BFGS) and inexact Newton iterations in such a way that the information gathered by one type of iteration improves the performance of the other, without an increase in computation. 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 an initial matrix 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,

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

We will assume that the gradient g of f is available but that computing the Hessian matrix is not possible. Two of the most effective algorithms for these types of problems are: (i) Hessian-free inexact Newton methods (HFN) [14, 10, 3], and (ii) limited memory quasi-Newton methods, such as L-BFGS [12, 5, 7]. By a “Hessian-free Newton method” we mean a Newton method in which the Hessian matrix is not available, and where products of the Hessian times a vector are either be computed by automatic differentiation or approximated by finite differences.

Inexact Newton methods that have access to the exact Hessian matrix and that 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 may not be 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 slow on ill-conditioned problems. Enriched methods aim to combine the best features of both methods in a 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

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

where $H(m)$ is a limited memory matrix that approximates the inverse of the Hessian matrix $\nabla^2 f(x)$, and m denotes the number of correction pairs stored.

During the cycle of L-BFGS iterations, $H(m)$ is updated using the most recent pairs. 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; they are obtained by setting $t = 0$ and $l = 0$, respectively. In Table 1 we display the arithmetic cost and memory requirements for one step of the L-BFGS iteration and the HFN iteration using $H(m)$ as preconditioner. In our implementation, the lengths of the cycles, l and t , are chosen dynamically, as described in the next section.

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;

Method	flops	memory
L-BFGS	$4mn + O(n)$	$2mn + 2n$
HFN	$\text{NoCG} \times (4mn + \text{f-g} + O(n))$	$4mn + 6n$

Table 1: Arithmetic costs (flops) of a step computation, and memory demands of the L-BFGS and HFN iterations. NoCG denotes the number of CG iterations, and f-g stands for an evaluation of the function and gradient.

- (c) to avoid the difficulties of choosing a good stopping test for the CG iteration, the algorithm always performed 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 varying degree during the course of 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 usually improves 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 a 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; this will be described in more detail in the next sections.

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,

$$Bv \approx \frac{g(x + \tau v) - g(x)}{\tau}, \quad (3.4)$$

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, 15, 16]. 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` (B , p , g , `method`, `status`, `maxcg`);

 compute α : call `LNSRCH` (α);

 compute $x_+ = x + \alpha p$;

 store $s = x_+ - x$ and $y = g_+ - g$;

 call `ADJUST` (l , t , α , `method`, `status`, `first`, `maxcg`);

End repeat

End while.

The vectors s and y are used to update the limited memory matrix $H(m)$.

The procedure `PCG` implements the preconditioned CG iteration. The parameter `method` can have the values 'L-BFGS' or 'HFN', and `maxcg` determines the maximum number of CG iterations allowed. This procedure returns a search direction p , and the value of `status`, which is used by procedure `ADJUST` 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. [13]).

A more detailed description of procedures `PCG` and `ADJUST` is given below.

3.1 Preconditioned CG Method

In addition to the step-computation of the enriched method, procedure PCG provides the information needed to update the quasi-Newton matrix $H(m)$ used as preconditioner. We assume here that the memory parameter m is fixed throughout the course of the optimization and that the matrix $H(m)$ is updated by means of PREQN [9], a Fortran package for the automatic computation of quasi-Newton preconditioners.

```

PROCEDURE PCG ( B, p, g, method, status, maxcg )

  set  $r^{(0)} \leftarrow -g$ ,  $p^{(0)} = -g$ 
  for  $j = 1, 2, \dots$  maxcg
    1. compute  $z^{(j-1)} = H(m)r^{(j-1)}$  by calling PREQN
    2. if method = 'L-BFGS' then set  $p \leftarrow z^{(0)}$ ; return
    3. if convergence test is satisfied then set  $p \leftarrow p^{(j-1)}$ ; return
    4.  $\rho_{j-1} = r^{(j-1)T} z^{(j-1)}$ 
    5. if  $j = 1$  then
       $v^{(1)} = -z^{(0)}$ 
    else
       $\beta_{j-1} = \rho_{j-1} / \rho_{j-2}$ 
       $v^{(j)} = -z^{(j-1)} + \beta_{j-1}v^{(j-1)}$ 
    end if
    6. compute or approximate the product  $Bv^{(j)}$ 
    7. if  $v^{(j)T} Bv^{(j)} \leq \text{TOL}$  then status = 'Indefinite Hessian'; set  $p \leftarrow p^{(j-1)}$ 
      return
    8.  $\alpha_j = \rho_{j-1} / v^{(j)T} Bv^{(j)}$ 
    9.  $p^{(j)} = p^{(j-1)} + \alpha_j v^{(j)}$ 
    10.  $r^{(j)} = r^{(j-1)} + \alpha_j Bv^{(j)}$ 
  end for
  set  $p \leftarrow p^{(\text{maxcg})}$ 
  return

```

Note that when `method = 'L-BFGS'` the procedure immediately returns with $p = -H(m)g$, as desired.

The preconditioning step $H(m)r^{(j-1)}$ is performed by a call to PREQN. This procedure also updates a new preconditioner (that will be used at the next HFN iteration) based on the most recently generated pair

$$\{v^{(j)}, Bv^{(j)}\}.$$

In other words, PREQN maintains two limited memory matrices: one that is used to precondition the current CG iteration, and a second matrix that is built in the course of the PCG iteration and that will define the preconditioner for the next outer iteration. A detailed description of the preconditioning process is given in [9].

3.2 Dynamic Adjustment of the Cycles

The enriched algorithm makes use of the procedure `ADJUST` to modify the lengths (l, t) of the L-BFGS and HFN cycles.

`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'`, the procedure `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]. Hessian-vector products were approximated by finite differences as in (3.4). All tests were performed on a DEC Personal Workstation with 512 Mb of main memory, using double precision FORTRAN. We set the initial values $m = 20$ and $l = 20$. 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) $\|r\|_{H(m)} \leq \eta \|g\|_{H(m)}$, where $\eta = 1/10$. The vector r stands for the CG residual, $r = -g - Bp$, and $H(m)$ is the preconditioner.

We imposed a limit of 30 CG iterations because this is sufficient to collect useful information in the preconditioner, and because we want to control of the cost of the HFN iteration.

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

$$\|r\|_2 \leq \eta_j \|g_j\|_2, \quad \eta = \min(0.5/j, \|g_j\|_2),$$

which is one of the rules discussed in [3]. (We chose these two stopping tests because they are well established in the literature, but we note that using the same test as for the enriched method did not improve the performance of the HFN method.) The inner CG iteration of methods HFN1 and HFN2 is preconditioned using the same quasi-Newton preconditioner as in the enriched method.

The optimization calculation was terminated for the three methods (Enriched, HFN1 and HFN2) when

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

or when 5000 CG iterations performed. The results are given in Tables 2 and 3, which report the number of function/gradient evaluations. We recall that each CG iteration requires one evaluation of the gradient vector, and we assume that function and gradient evaluations are always performed together. The symbol * indicates that the limit of 5000 CG iterations was reached.

These 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 enriched method is not very sensitive to the stopping test of the CG iteration; we experimented with several variants and obtained similar results.

Next, in Table 4 we compared the enriched method with the L-BFGS algorithm using $m = 20$ correction pairs. We observe that in many of these problems, L-BFGS is more economical, but it is interesting to note that the enriched method performed well in the most difficult problems.

In conclusion, the numerical results suggest that the enriched method should be considered as a serious competitor to Hessian-free Newton methods. In these tests, the enriched method did not outperform L-BFGS, but since it is known (see e.g. [11]) that Newton-type methods are more effective than L-BFGS on ill-conditioned problems, the approach described here constitutes a useful addition to the arsenal of large scale optimization techniques.

PROBLEM	n	Enriched	HFN1	HFN2
ARWHEAD	1000	11	9	7
BDQRTIC	100	42	78	70
BROYDN7D	1000	416	784	969
BROWNAL	10	18	31	17
BRYBND	1000	47	215	123
CRAGGLVY	1000	86	104	119
CHAINWOO	1000	5056	5597	*
COSINE	1000	16	22	21
DIXMAANA	1500	13	28	26
DIXMAANB	1500	12	25	18
DIXMAANC	1500	13	28	20
DIXMAAND	1500	16	37	24
DIXMAANE	1500	192	211	196
DIXMAANF	1500	162	188	241
DIXMAANG	1500	165	288	411
DIXMAANH	1500	166	235	348
DIXMAANI	1500	2602	2622	2541
DIXMAANK	1500	1437	3730	1620
DIXMAANL	1500	1771	1421	1705
DQDRTIC	1000	18	21	15
DQRTIC	500	36	71	72
EDENSCH	2000	34	57	43
EIGENALS	110	121	191	151
EIGENBLS	110	885	1086	1759
EIGENCLS	462	1458	1810	5226
ENGVAL1	1000	18	28	29
FLETGBV3	1000	18	24	22
FLETCHCR	100	679	1027	3514
FMINSRF2	1024	190	611	779
FMINSURF	1024	297	870	1034

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

PROBLEM	n	Enriched	HFN1	HFN2
FREUROTH	1000	47	35	35
GENROSE	500	1494	2579	*
GENHUMPS	1000	3225	5083	5700
LIARWHD	1000	24	37	35
MOREBV	1000	83	95	103
MSQRTALS	1024	1918	1960	2685
MSQRTBLS	1024	1585	1570	1766
NCB20B	1000	824	1085	988
NCB20	1010	361	409	564
NONCVXU2	1000	1496	1726	*
NONCVXUN	1000	1856	3265	*
NONDQUAR	100	260	460	952
PENALTY1	1000	80	132	101
PENALTY2	100	71	106	92
PENALTY3	100	87	94	86
POWELLSG	1000	32	58	58
POWER	1000	140	275	303
QUARTC	1000	38	80	74
SCHMVETT	1000	46	44	56
SINQUAD	1000	189	280	522
SPARSINE	1000	3566	3256	3927
SPARSQUR	1000	28	60	59
SPMSRTL	1000	120	123	224
SROSENBR	1000	18	29	22
TOINTGSS	1000	19	29	34
TQUARTIC	1000	26	11	32
TRIDIA	1000	535	578	465
VARDIM	100	37	78	52
WATSON	31	54	61	145
WOODS	1000	117	183	174

Table 3: Number of function/gradient evaluations for three optimization methods.

PROBLEM	L-BFGS	Enriched	PROBLEM	L-BFGS	Enriched
BDQRTIC	40	42	MOREBV	52	83
BROYDN7D	380	416	MSQRTALS	1805	1918
BRYBND	40	47	MSQRTBLS	1565	1585
CRAGGLVY	79	86	NCB20B	1285	824
CHAINWOO	4409	5056	NCB20	397	361
DIXMAANE	191	192	NONCVXU2	1106	1496
DIXMAANF	143	162	NONCVXUN	4563	1856
DIXMAANG	139	165	NONDQUAR	361	260
DIXMAANH	139	166	PENALTY1	74	80
DIXMAANI	2858	2602	PENALTY2	62	71
DIXMAANK	998	1437	PENALTY3	75	87
DIXMAANL	1097	1771	POWELLSG	38	32
DQRTIC	35	36	POWER	132	140
EDENSCH	32	34	QUARTC	37	38
EIGENALS	172	121	SCHMVETT	42	46
EIGENBLS	900	885	SINQUAD	190	189
EIGENCLS	2317	1458	SPARSINE	3826	3566
FLETCHCR	592	679	SPARSQUR	27	28
FMINSRF2	139	190	SPMSRTL	107	120
FMINSURF	217	297	TQUARTIC	25	26
FREUROTH	55	47	TRIDIA	513	535
GENROSE	1314	1494	VARDIM	36	37
GENHUMPS	3452	3225	WATSON	52	54
LIARWHD	23	24	WOODS	108	117

Table 4: Number of function/gradient evaluations for two optimization 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 limited memory quasi-Newton updates*. SIAM, J. Optim. 10 (2000), pp. 1079–1096.
- [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.
- [11] S.G. NASH AND J. NOCEDAL, *A Numerical study of the limited memory BFGS method and the truncated-Newton method for large-scale optimization*, SIAM J. Optim. 1 (1991), pp. 358–372.
- [12] J. NOCEDAL, *Updating quasi-Newton matrices with limited storage*, Math. Comput., 35 (1980), pp. 773–782.
- [13] J. NOCEDAL AND S.J. WRIGHT, *Numerical Optimization*, Springer Series in Operations Research, New York, 1999.
- [14] D.P. O’LEARY, *A discrete Newton algorithm for minimizing a function of many variables*, Math. Programming, 23 (1982), pp. 20–33.

- [15] D. XIE AND T. SCHLICK, *Remark on the updated truncated Newton minimization package, Algorithm 702*, ACM Trans. Math. Software, 25 (1999), pp. 108-122.
- [16] D. XIE AND T. SCHLICK, *Efficient implementation of the truncated-Newton algorithm for large-scale chemistry applications*. SIAM J. Optim., 10 (1999) pp. 132-154.