EFFICIENT IMPLEMENTATION OF THE TRUNCATED-NEWTON ALGORITHM FOR LARGE-SCALE CHEMISTRY APPLICATIONS *

DEXUAN XIE AND TAMAR SCHLICK †

Abstract. To efficiently implement the truncated-Newton (TN) optimization method for large-scale, highly nonlinear functions in chemistry, an unconventional modified Cholesky (UMC) factorization is proposed to avoid large modifications to a problem-derived preconditioner, used in the inner loop in approximating the TN search vector at each step. The main motivation is to reduce the computational time of the overall method: large changes in standard modified Cholesky factorizations are found to increase the number of total iterations, as well as computational time, significantly. Since the UMC may generate an indefinite, rather than a positive definite, effective preconditioner, we prove that directions of descent still result. Hence, convergence to a local minimum can be shown, as in classic TN methods, for our UMC-based algorithm. Our incorporation of the UMC also requires changes in the TN inner loop regarding the negative-curvature test (which we replace by a descent direction test) and the choice of exit directions. Numerical experiments demonstrate that the unconventional use of an indefinite preconditioner works much better than the minimizer without preconditioning or other minimizers available in the molecular mechanics and dynamics package CHARMM. Good performance of the resulting TN method for large potential energy problems is also shown with respect to the limited-memory BFGS method, tested both with and without preconditioning.

Key words. truncated-Newton method, indefinite preconditioner, molecular potential minimization, descent direction, modified Cholesky factorization, unconventional modified Cholesky factorization

AMS subject classifications. 65K10 92E10

1. Introduction.

Optimization of highly nonlinear objective functions is an important task in biomolecular simulations. In these chemical applications, the energy of a large molecular system—such as a protein or a nucleic acid, often surrounded by water molecules—must be minimized to find a favorable configuration of the atoms in space. Finding this geometry is a prerequisite to further studies with molecular dynamics simulations or global optimization procedures, for example. An important feature of the potential energy function is its ill conditioning; function evaluations are also expensive, and the Hessian is typically dense. Moreover, a minimum-energy configuration corresponds to a fairly accurate local optimum. Since thousands of atoms are involved as independent variables and, often, the starting coordinates may be far away from a local minimum, this optimization task is formidable and is attracting an increasing number of numerical analysts in this quest, especially for global optimization (see [17, 18], for example).

The practical requirements that chemists and biophysicists face are somewhat different from those of the typical numerical analyst who develops a new algorithm. The computational chemists seek reliable algorithms that produce answers quickly, with as little tinkering of parameters and options as possible. Thus, theoretical performance is not as important as practical behavior, and CPU time is of the utmost importance. A prominent example is the current preference in the biomolecular community for

* Received by the editors September 9, 1997; accepted for publication (in revised form) July 3, 1998; published on SIAM J. Optim., Vol. 9, 1999. This work was supported in part by the National Science Foundation through award number ASC-9318159. T. Schlick is an investigator of the Howard Hughes Medical Institute.

† Departments of Chemistry, Mathematics, and Computer Science, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012, (dexuan@cims.nyu.edu, schlick@nyu.edu). Phone: (212) 998-3596, fax: (212) 995-4152.

Ewald summation techniques (for periodic systems) over fast-multipole approaches for evaluating the long-range forces in molecular simulations; the latter have smaller complexity in theory ($O(n)$, where n is the system size, rather than the $O(n \log n)$ associated with Ewald), but the Ewald procedure is easy to program and very fast in practice for a range of molecular sizes.

The present paper focuses on implementation details of a truncated-Newton (TN) method that are important in practice for performance efficiency in large-scale potential energy minimization problems. The algorithmic variations we discuss are motivated by optimization theory but depart from standard notions (e.g., of a positive-definite preconditioner) for the sake of efficiency. Algorithmic stability and convergence properties are still retained in theory as in the traditional approach, but performance in practice is enhanced by the proposed modifications.

Our interest in such chemistry applications first led to the development of a TN method adapted to potential-energy functions [25]. Our TN package, TNPACK [23, 24], was then adapted [5] for the widely used molecular mechanics and dynamics program CHARMM [1].

In TN methods, the classic Newton equation at step k,

$$H(X^k)P = -g(X^k), \tag{1}$$

where g and H are the gradient and Hessian, respectively, of the objective function E at X^k, is solved iteratively and approximately for the search vector P [4]. The linear conjugate gradient (CG) method is a suitable choice for this solution process for large-scale problems, and preconditioning is necessary to accelerate convergence. A main ingredient of TNPACK is the use of an application-tailored preconditioner M_k. This matrix is a sparse approximation to $H_k \equiv H(X^k)$, formulated at each outer minimization step k. The preconditioner in chemical applications is constructed naturally from the local chemical interactions: bond length, bond angle, and torsional potentials [25]. These terms often contain the elements of largest magnitude and lead to a sparse matrix structure which remains constant (in topology) throughout the minimization process [5]. Since M_k may not be positive-definite, our initial implementation applied the modified Cholesky (MC) factorization of Gill and Murray [7] to solve the linear system $M_k z = r$ at each step of PCG (preconditioned CG). Thus, an effective positive-definite preconditioner, $\widetilde{M_k}$, results.

Why is a TN scheme a competitive approach? First, analytic second-derivative information is available in most molecular modeling packages and should be used to improve minimization performance. That is, curvature information can guide the search better toward low-energy regions. Second, the basic idea of not solving the Newton equations exactly for the search vector when far away from a minimum region saves unnecessary work and accelerates the path toward a solution. Third, the iterative TN scheme can be tailored to the application in many ways: handling of the truncated inner loop, application of a preconditioner, incorporating desired accuracy, and so on. These implementation details are crucial to realized performance in practice.

In our previous studies, we have discussed alternative minimization approaches to TN [5, 23, 24, 25]. We showed that modified Newton methods are computationally too expensive to be feasible for large systems [23, 24, 25] since the large Hessian of potential energy function is dense and highly indefinite. Nonlinear CG methods can take excessively long times to reach a solution [5]; this is not only because of the known properties of these methods but also due to the expense of evaluating the objective function at each step, a cost that dominates the CPU time [15]. A competitive approach to TN, however, is the limited-memory BFGS algorithm (LM-BFGS) [11], which also uses curvature information to guide the search. A study by Nash and Nocedal [15] comparing the performance of a discrete TN method[1] to LM-BFGS found both schemes to be effective for large-scale nonlinear problems. They suggested that the former performs better for nearly quadratic functions and also may perform poorly on problems associated with ill-conditioned Hessians. However, as Nash and Nocedal point out, since TN almost always requires fewer iterations than LM-BFGS, TN would be more competitive if the work performed in the inner loop were reduced as much as possible. This is the subject of this article.

Our experiences to date in chemical applications for medium-size problems suggest that the CPU time of the TN approach can be smaller than LM-BFGS since the total number of function evaluations is reduced. Examples shown in the present work, for larger problems as well, reinforce this. Surely, both methods can be efficient tools for large-scale optimization, and superiority of one scheme over another cannot be claimed.

In this paper, we focus on an important aspect of the TN method that affects its performance profoundly: the formulation and handling of the preconditioner in the inner PCG loop that is used to approximate the search vector at each step of the method. The use of a standard modified Cholesky factorization applied to physically constructed preconditioner leads to excessively large modifications, which in turn means many function evaluations and thus a large total CPU time for the minimization method. Pivoting strategies [6, 8] can reduce the size of the modifications but not necessarily the problem condition number, and thus are not a clear solution. The problem we address here is thus a general one, associated with other modified Cholesky factorization methods [3, 6, 7, 8, 26]: how to handle large modifications to matrices that are far from positive definite. However, we address this problem only for the TN minimization context, where the solution of such a linear system is not as important as progress in the overall minimization method.

In chemistry problems, a large negative eigenvalue often corresponds to a transition or saddle point. We argue that in our special context (large-scale computational chemical problems and TN), a standard MC factorization is inappropriate. Rather, it is sufficient to require only that the preconditioner be nonsingular and often positive-definite near a minimum point. This leads to development of our simple unconventional modified Cholesky (UMC) factorization.

We present details of the resulting TN algorithm along with many practical examples that illustrate how the use of an indefinite preconditioner outperforms other variants (e.g., no preconditioning, positive-definite preconditioning) in the TN framework. We detail analysis that shows that the directions produced are still descent directions, and thus the global convergence of the method (to a local minimum) can be proven in the same way as for the "classic" TN scheme [4]. We also offer comparisons with LM-BFGS that suggest the better performance of TN for large potential energy problems.

The remainder of the paper is organized as follows. In the next section, we summarize the structure of a general descent method and describe the new PCG inner loop we develop for the TN method. In section 3 we present the UMC designed for our applications. In section 4, we present numerical experiments that demonstrate the overall performance of the modified TNPACK minimizer, along with a comparison to

[1] The discrete TN method computes Hessian and vector products by finite differences of gradients.

LM-BFGS and other minimizers available in CHARMM (a nonlinear CG and a Newton method). Conclusions are summarized in section 5. For completeness, analyses for the PCG inner loop of TN are presented in Appendix A, and the full algorithm of the TN method is described in Appendix B. The modified package is also described in [28].

2. Descent methods and the truncated Newton approach. We assume that a real-valued function $E(X)$ is twice continuously differentiable in an open set $\mathcal{D}$ of the n-dimensional vector space R^n. Descent methods for finding a local minimum of E from a given starting point generate a sequence $\{X^k\}$ in the form

$$X^{k+1} = X^k + \lambda_k P^k, \tag{2}$$

where the search direction P^k satisfies

$$g(X^k)^T P^k < 0. \tag{3}$$

(The superscript T denotes a vector or matrix transpose.) Equation (3) defines the descent direction P^k which yields function reduction. The steplength λ_k in (2) is chosen to guarantee sufficient decrease, e.g., such that [12]

$$E(X^k + \lambda_k P^k) \le E(X^k) + \alpha \lambda_k g(X^k)^T P^k \tag{4}$$

and

$$|g(X^k + \lambda_k P^k)^T P^k| \le \beta |g(X^k)^T P^k|, \tag{5}$$

where α and β are given constants satisfying $0 < \alpha < \beta < 1$.

Condition (5) is referred to as the strong Wolfe condition. A steplength λ_k satisfying (5) must satisfy the usual Wolfe condition:

$$g(X^k + \lambda_k P^k)^T P^k \ge \beta g(X^k)^T P^k.$$

According to the line search algorithm of Moré and Thuente [12] (used in TNPACK), such a steplength λ_k is guaranteed to be found in a finite number of iterations. Hence, according to the basic theory of descent methods [10], a descent method defined in the form (2) guarantees that

$$\lim_{k \to \infty} g(X^k) = 0.$$

The challenge in developing an efficient descent method is balancing the cost of constructing of a descent direction P^k with performance realized in practice.

To reduce the work cost of the classic modified Newton method and develop a globally convergent descent algorithm, Dembo and Steihaug proposed a clever variation known as the truncated-Newton method [4]. Since then, several variants have been developed and applied in various contexts; see, e.g., [13, 14, 15, 23, 25, 29]. The linear PCG framework is the most convenient generator of descent directions in the inner TN loop due to its efficiency and economic storage requirements for solving large positive-definite linear systems. Since the PCG method may fail at some step when the matrix H_k is indefinite, a termination strategy is required to guarantee that the resulting search directions are still descent directions. In addition, the PCG inner loop of the TN method can be made more effective by employing a *truncation test*.

We present our PCG inner loop of the TN scheme in Algorithm 1. The changes with respect to a "standard" PCG inner loop include allowing an indefinite preconditioner, the UMC factorization (discussed in the next section), and a new descent direction test.

ALGORITHM 1 (PCG INNER LOOP k OF TN FOR SOLVING $H_k p = -g_k$ WITH A GIVEN PRECONDITIONER M_k).

Let p_j represent the jth PCG iterate, d_j the direction vector, and r_j the residual vector satisfying $r_j = -g_k - H_k p_j$. Let $\mathrm{IT}_{\mathrm{PCG}}$ denote the maximum number of allowable PCG iterations at each inner loop.

Set $p_1 = 0$, $r_1 = -g_k$, and $d_1 = z_1$, where z_1 solves a system related to $M_k z_1 = -g_k$ by UMC.

For $j = 1, 2, 3, \ldots,$

1. [SINGULARITY TEST]
   If either $|r_j^T z_j| \le \delta$ or $|d_j^T H_k d_j| \le \delta$ (e.g., $\delta = 10^{-10}$),
   exit PCG loop with search direction $P^k = p_j$ (for $j = 1$, set $P^k = -g_k$).
2. Compute $\alpha_j = r_j^T z_j / d_j^T H_k d_j$ and $p_{j+1} = p_j + \alpha_j d_j$.
3. [DESCENT DIRECTION TEST] (replaces *negative curvature test*)
   If $g_k^T p_{j+1} \ge g_k^T p_j + \delta$,
   exit PCG loop with $P^k = p_j$ (for $j = 1$, set $P^k = -g_k$).
4. Compute $r_{j+1} = r_j - \alpha_j H_k d_j$.
5. [TRUNCATION TEST]
   If $\|r_{j+1}\| \le \min\{c_r/k\ ,\ \|g_k\|\}\|g_k\|$, or $j + 1 > \mathrm{IT}_{\mathrm{PCG}}$,
   exit PCG loop with $P^k = p_{j+1}$.
   (By default, $c_r = 0.5$ and $\mathrm{IT}_{\mathrm{PCG}} = 40$).
6. Compute $\beta_j = r_{j+1}^T z_{j+1} \,/\, r_j^T z_j$, and $d_{j+1} = z_{j+1} + \beta_j d_j$,
   where z_{j+1} solves a system related to $M_k z_{j+1} = r_{j+1}$ by UMC.

Since the effective preconditioner $\widetilde{M_k}$ generated by our UMC (see the next section) and the Hessian matrix H_k may be indefinite, it may happen that $r_j^T z_j$ or $d_j^T H_k d_j$ is exactly zero for some j.(So far, we have not encountered this in practice). Hence, to ensure that the PCG recursive formulas are well defined, *the singularity test* has been added in step 1 above.

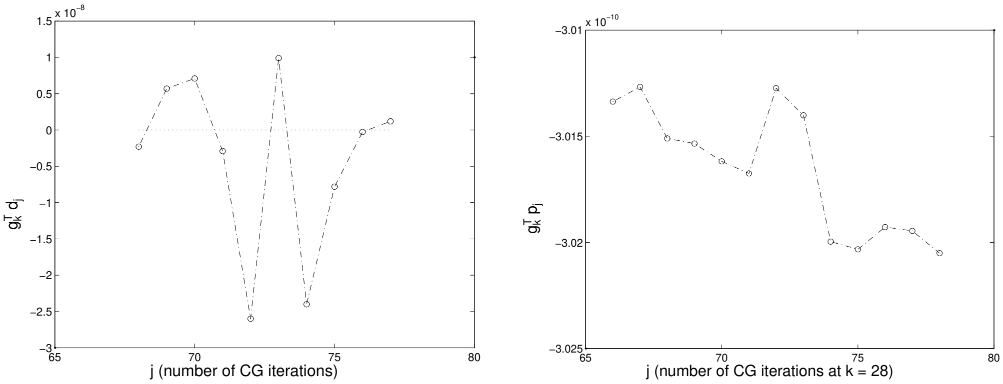
Our descent direction test (step 3) is equivalent in theory to the following *negative curvature test* [4]: if $d_j^T H_k d_j < \delta d_j^T d_j$, halt the PCG loop with exit search direction $P^k = -g_k$ if $j = 1$ or p_j if $j > 1$. We prove this equivalence in Theorem 3 of Appendix A.

In practice, however, due to computer rounding errors, the negative curvature test may not guarantee that the inner product $g_k^T p_j$ decreases monotonically as theory predicts (see Theorem 2 in Appendix A). See also Box 1 for numerical examples. The descent direction test in step 3 halts the PCG iterative process as soon as the situation $g_k^T p_{j+1} > g_k^T p_j$ (or $|g_k^T p_{j+1}| < |g_k^T p_j|$) is encountered. We have observed better performance in practice for large-scale problems with this modification.

In the standard implementation of the negative curvature test in TN [4], P^k can be set to p_j or d_j for $j > 1$, both directions of descent. We have now removed the option in step 3 of using the auxiliary directions d_j as exit search vectors. We show in Theorem 1 of Appendix A that d_j may be an ascent direction when the effective preconditioner is indefinite. Even in the standard implementation (i.e., positive-definite effective

Box 1: *Examples for the negative curvature test in finite precision arithmetic.*

We consider the negative curvature test implemented in TN as originally described [4] for minimizing the alanine dipeptide potential function (22 atoms, 66 Cartesian variables). For simplicity, we do not use a preconditioner (i.e., M_k is the identity matrix); results thus reflect the case of using a positive-definite preconditioner.



The left figure shows that some d_j are ascent directions even if $d_j^T H_k d_j > 0$. This is possible because the basic relation $\{g_k^T r_j\} = 0$ (see (13) in Appendix A) may deteriorate due to computer rounding errors. Namely, the inner product $g_k^T r_j$ may become positive, and thus d_j may be an ascent direction because $g_k^T d_j = g_k^T r_j + \beta_{j-1} g_k^T d_{j-1}$.

The right figure shows that the inner product $g_k^T p_j$ does not necessarily decrease monotonically due to computer rounding errors. This can be understood since if d_j is an ascent direction for some j but not a direction of negative curvature (i.e., $d_j^T H_k d_j > 0$), then we have $\alpha_j > 0$ and $g_k^T d_j > 0$, so that $g_k^T p_{j+1} = g_k^T p_j + \alpha_j g_k^T d_j \geq g_k^T p_j$ or $g_k^T p_{j+1} > 0$. Hence, the negative curvature test may not guarantee a descent direction or even a "good" descent direction in some sense, as theory predicts for TN in finite precision arithmetic (see Theorem 2 in Appendix A).

preconditioner), we argue that p_j is a better choice than d_j according to Theorem 4 of Appendix A.

Although not used here, we leave the option of using the negative curvature test (with the d_j choice removed) in the general TNPACK package; see the full algorithm in Appendix B.

3. The UMC method. We discuss our motivation for developing the UMC in section 3.1 and describe the factorization in section 3.2.

3.1. Motivation. Recall that our major goal is to reduce the computational effort in the inner loop of the TN method. Therefore, we choose a preconditioner that is sparse (sparsity less than 5% for medium-size molecules) and rapid to compute. The factorization of the linear system involving M is handled efficiently within the framework of the Yale Sparse Matrix Package (YSMP) [19, 20]. YSMP routines use special pointer arrays to record data positions and manipulate only the nonzero elements. Efficiency is further enhanced by reordering M at the onset of the minimization method

TABLE 1
The CPU time distribution (%) among TNPACK components

Protein	E, g, & H evals.	M evals.	Solve $Mz = r$	Hd evals.	Other tasks
BPTI	21	0.14	6.8	69	3.06
Lysozyme	22	0.08	2.5	74	1.42

to minimize fill-in. This works because the structure of the preconditioner, or the connectivity structure of our molecular system, remains constant. This reordering is optional.

The main advantage of this sparsity-based factorization is efficiency. As we show in Table 1, the CPU percentage involved in solving $Mz = r$ within YSMP is less than 7% and 3% of the total TN cost of minimization for the proteins bovine pancreatic trypsin inhibitor, BPTI (568 atoms, 1704 Cartesian variables) and lysozyme (2030 atoms, 6090 variables), respectively. Since the Hessian H is dense and we evaluated the Hessian and vector products Hd in PCG directly (i.e., not by finite differences of gradients [23]), this part consumes the majority of the CPU time: about 70%. The finite-differencing approximation of Hd may be more competitive for large systems. In addition, function and derivative evaluations consume about 20% of the total CPU time.

A disadvantage of our approach is the absence of pivoting strategies based on numerical values of the matrix elements. Pivoting would increase the computational time but possibly lead to a lower condition number for the modification $\widetilde{M}$ of M, and a smaller error bound $\|E\|_\infty$ in the MC process. Here $E = \widetilde{M} - M$ is a nonnegative diagonal matrix. Our experiences suggest that pivoting strategies in the context of a standard MC are far less effective than our UMC in the TN context. Namely, our numerical experiments demonstrate that the Gill-Murray-Wright MC (GMW MC) with pivoting [8] can reduce the error bound $\|E\|_\infty$ but far less significantly the condition number of $\widetilde{M}$ (see Box 2). This is a consequence of our highly indefinite preconditioner near regions far away from a local minimum.

Our experiments studied three other MC algorithms: the partial Cholesky (PC) factorization described by Forsgren, Gill & Murray [6], the Schnabel and Eskow (SE) MC [26], and the Cheng and Higham (CH) MC [3] (see Box 2). As the authors state, all methods can produce unacceptably large perturbations in special cases. In our application, these large perturbations typically lead to poor performance when the objective matrix M is highly indefinite; a very large condition number or a very large error bound $\|E\|_\infty$ (much larger than the magnitude of the negative minimum eigenvalue $\lambda_{min}(M)$ of M) can result.

To see this analytically, recall that the GMW MC process modifies a symmetric $n \times n$ matrix M into a positive-definite matrix $\widetilde{M}$ and factors it as $\widetilde{M} = M + E = LDL^T$, where L, D, and E are, respectively, unit lower-triangular, diagonal, and diagonal $n \times n$ matrices. The elements $e_j = d_j - \overline{d_j}$ of E are defined by

$$\overline{d_j} = m_{jj} - \sum_{k=1}^{j-1} l_{jk} c_{jk} \quad \text{and} \quad d_j = \max\left\{|\overline{d_j}|, \delta, \frac{\theta^2}{\beta^2}\right\}, \tag{6}$$

where $c_{ij} = l_{ij} d_j$, $\theta = \max_{j+1 \le i \le n} |c_{ij}|$, and positive numbers δ and β are introduced to ensure the numerical stability (e.g., $\delta = 10^{-9}$ and $\beta = \xi/\sqrt{n^2-1}$, where ξ is the

Box 2: Examples of Large modifications in MC methods

We experimented with four MC methods in MATLAB for the symmetric 42×42 matrix M constructed from the second partial derivatives of the butane potential energy terms coming from bond length, bond angle, and torsional potentials [16]. M is indefinite with three negative eigenvalues: $-5.718, -74.703$, and -218.475, and the condition number is 1.553×10^3.

These four MC methods are due to Gill, Murray, and Wright (GMW) [8]; Schnabel and Eskow (SE) [26]; Forsgren, Gill, and Murray [6], a partial Cholesky (PC) factorization; and Cheng and Higham (CH) [3]. The MATLAB M-files for GMW, SE, and CH were provided by Wright, Eskow, and Cheng, respectively. We used their default tolerance δ. In addition, we wrote corresponding files for GMW without pivoting and for PC according to [7] and [6]. Note that the error matrix E in PC and CH may not be diagonal.

Modified Cholesky factorization	$\widetilde{M}$ condition number	$\widetilde{M}$ minimum eigenvalue	Error $\|E\|_\infty$
GMW	8.42×10^6	1.70×10^{-3}	1.38×10^4
GMW, no pivoting	1.78×10^7	2.80×10^{-2}	5.13×10^5
PC	6.79×10^6	1.30×10^{-3}	4.59×10^2
SE	3.09×10^1	3.62×10^2	2.33×10^3
CH	7.28×10^9	1.22×10^{-6}	7.86×10^2

For reference, our UMC gives as a function of the control parameter τ the following results.

τ of UMC	$\widetilde{M}$ condition number	$\widetilde{M}$ minimum eigenvalue	$\|E\|_\infty$
40	546.157	-874.032	40.0
120	198.716	-98.475	120.0
200	491.541	-18.475	200.0
240	423.753	21.524	240.0
280	148.903	61.524	280.0

largest magnitude of an element of M. From (6) we see that an element e_j of E has the following expression

$$
e_j = d_j - \overline{d_j} = \begin{cases} \delta - \overline{d_j} & \text{when } \delta \geq \max\{|\overline{d_j}|, \frac{\theta^2}{\beta^2}\}, \\ \frac{\theta^2}{\beta^2} - \overline{d_j} & \text{when } \frac{\theta^2}{\beta^2} \geq \max\{|\overline{d_j}|, \delta\}, \\ |\overline{d_j}| - \bar{d_j} & \text{when } |\overline{d_j}| \geq \max\{\frac{\theta^2}{\beta^2}, \delta\}. \end{cases} \tag{7}
$$

For a negative $\overline{d_j}$, e_j may thus be $2|\overline{d_j}|$ or $\frac{\theta^2}{\beta^2} + |\overline{d_j}|$. If $\overline{d_j}$ is a large negative number or $\frac{\theta^2}{\beta^2}$ a large positive number, $\|E\|_\infty$ may be much larger than the value of $|\lambda_{min}(M)|$.

The second author has studied performance of the GMW [8] versus the SE MC [26] for difficult computational chemistry problems in the context of TN [22]. That study showed that no factorization is clearly superior to another. We have retained the former in TNPACK since it is simple to implement in the context of YSMP. The CH MC [3] is another possibility worth examining in our context since it is easily implemented in existing software. Still, Box 2 suggests that all MC algorithms may

TABLE 2

Performance of TNPACK based on the GMW MC with pivoting (GMW_P) and without pivoting (GMW) and our UMC with $\tau = 10$

Butane (42 variables)					
MC	Final E	Final $\|\|g\|\|$	Outer (Inner) Iter.	E & g Evals	CPU Time
GMW_P	4.7531	3.57×10^{-10}	35 (1329)	44	6.3 sec.
GMW	3.9039	2.07×10^{-8}	52 (78)	93	0.28
UMC	3.9039	1.12×10^{-8}	21 (62)	26	0.17
Alanine dipeptide (66 variables)					
GMW_P	-15.245	1.45×10^{-8}	31 (5641)	39	78
GMW	-15.245	1.85×10^{-11}	6387 (7108)	15801	44
UMC	-15.245	3.65×10^{-9}	27 (186)	39	1.3

exhibit poor performance for a highly indefinite matrix M.

In our TN applications, while we find that pivoting strategies can improve the performance of MC and even reduce the total number of outer (Newton) iterations, the total number of inner (PCG) iterations may increase significantly. This may result from the large modification made to M. Consequently, the CPU time of TN is large even when pivoting is used in standard MC schemes. See Table 2 for examples on two small molecular systems. Note that pivoting strategies (for example, Bunch-Kaufman [2] and that used in [3]) require at least $O(n)$ comparisons as well as substantial data movement.

The objective of allowing an indefinite preconditioner in the context of TN is to produce an efficient preconditioner for the inner loop, that is, one that leads to the smallest number of PCG iterations. The original indefinite M_k is a good approximation to H_k, so we do not want to make excessively large (and perhaps artificial) perturbations, as often required by standard MC methods we have experimented with. Since the PCG with an indefinite preconditioner can still generate directions of descent (Theorem 2 of Appendix A), using the UMC to solve the linear system involving M_k in the context of YSMP is one feasible efficient strategy.

In formulating the UMC, we were also guided by the observation that the Hessian matrix itself in our applications is often positive-definite near a solution (minimum energy). This suggested to us to construct preconditioners that also exhibit this trend. This can be accomplished by adding a constant matrix τI to M_k, where τ is a problem-size independent small positive number found by experimentation (e.g., $\tau = 10$).

Intuitively, the UMC can be interpreted as follows. When $\tau > |\lambda_{min}(M_k)|$, $M_k + \tau I$ is positive-definite and has the standard (stable) LDL^T factorization. To ensure a numerically stable factorization when $M_k + \tau I$ is indefinite, we modify it further by adding a diagonal matrix as in GMW, so as to impose an upper bound on the factors L and D. The difference in our treatment from the standard GMW MC is that our diagonal candidates can be negative (the third situation in eq. (9) below), and thus the resulting UMC matrix may still be indefinite. Certainly, other procedures for solving linear systems involving indefinite matrices exist, but the simple UMC strategy above is most easily incorporated into our current software and is found to work well.

3.2. The UMC factorization. Our UMC effectively applies a standard LDL^T factorization for matrix $M + \tau I$ for a given nonnegative number τ. The simple approach of adding a multiple of the identity matrix to the indefinite matrix has been discussed in Dennis and Schnabel [10]; however, the scalar τ is chosen to make $\widetilde{M}$ safely positive definite on the basis of a diagonal dominance estimate and thus can be much larger than necessary. Our approach effectively sets τ to be a small nonnegative number like 10 (through numerical experiments) that ensures that $M_k + \tau I$ is positive definite at the final steps of the TN minimization process. At other steps, $M_k + \tau I$ may be indefinite, but the modification to the original M is relatively small, and this produces faster convergence overall.

Since $M + \tau I$ may not be positive definite, a similar strategy to the standard GMW strategy [7] (i.e., the use of two bound parameters δ and β in (8) and the dependence of the entries d_j of the factor D on the elements of M as shown in (9)) is employed in UMC to guarantee numerical stability. The following scheme describes our numerically stable process for factoring a symmetric matrix M with small perturbations, with the resultant matrix not necessarily positive definite.

In the jth step of the UMC factorization, suppose that the first $j-1$ columns have been computed, and satisfy

$$|d_k| > \delta, \quad \text{and} \quad |l_{ik}|\sqrt{|d_k|} \le \beta, \quad i > k, \tag{8}$$

for $k = 1, 2, \ldots, j-1$. Here δ is a small positive number used to avoid numerical difficulties when $|d_k|$ is too small, and β is a positive number satisfying $\beta^2 = \xi/\sqrt{n(n-1)}$, where ξ is the largest magnitude of an element of M.

We define

$$\overline{d_j} = m_{jj} - \sum_{k=1}^{j-1} l_{jk} c_{jk} \quad \text{and} \quad \theta = \max_{j+1 \le i \le n} |c_{ij}|,$$

where $c_{ij} = l_{ij} d_j$ is computed by using

$$c_{ij} = m_{ij} - \sum_{k=1}^{j-1} l_{jk} c_{ik}, \qquad i = j+1, \ldots, n.$$

We then set $\widetilde{d_j} = \overline{d_j} + \tau$, and define

$$d_j = \begin{cases} \max\{\widetilde{d_j}, \frac{\theta^2}{\beta^2}\} & \text{when } \widetilde{d_j} > \delta, \\ \delta & \text{when } |\widetilde{d_j}| \le \delta, \\ \min\{\widetilde{d_j}, -\frac{\theta^2}{\beta^2}\} & \text{when } \widetilde{d_j} < -\delta, \end{cases} \tag{9}$$

where δ and β are given in (8). Note that the above d_j is negative when the third possibility in (9) occurs, resulting in an indefinite matrix.

This definition of d_j implies by induction that the relation (8) holds for all $k = 1, 2, \ldots, n$, and hence the factorization is numerically stable.

The effective $\widetilde{M}$ produced by our UMC satisfies

$$\widetilde{M} = LDL^T = M + E,$$

where E is a diagonal matrix. In particular, $\widetilde{M}$ becomes positive definite with $E = \tau I$ if $\tau > |\lambda_{min}(M)|$; otherwise, the jth element e_j of E can be expressed in the form

$$e_j = d_j - \overline{d_j} = \begin{cases} \tau & \text{when } |\overline{d_j} + \tau| > \max\{\delta, \frac{\theta^2}{\beta^2}\}, \\ \delta - \overline{d_j} & \text{when } \delta \ge \max\{|\overline{d_j} + \tau|, \frac{\theta^2}{\beta^2}\}, \\ \frac{\theta^2}{\beta^2} - \overline{d_j} & \text{when } \frac{\theta^2}{\beta^2} \ge \overline{d_j} + \tau > \delta, \\ -\frac{\theta^2}{\beta^2} - \overline{d_j} & \text{when } -\delta > \overline{d_j} + \tau > -\frac{\theta^2}{\beta^2}, \end{cases} \tag{10}$$

for $j = 1, 2, \ldots, n$. By arguments similar to those used in [7], it can be shown that

$$|e_j| \leq \frac{\theta^2}{\beta^2} + |\overline{d_j}| + \delta + \tau,$$

along with $|\overline{d_j}| < \gamma + (n-1)\beta^2$ and $\theta \leq \xi + (n-1)\beta^2$, where

$$\gamma = \max_{1\leq i\leq n} |m_{ii}| \qquad \text{and} \qquad \xi = \max_{1\leq j\leq n} \max_{j+1\leq i\leq n} |m_{ij}|.$$

Therefore, a worst-case bound of $\|E\|_\infty$ is obtained:

$$\|E\|_\infty \leq \left[\frac{\xi}{\beta} + (n-1)\beta\right]^2 + \gamma + (n-1)\beta^2 + \tau + \delta \tag{11}$$

for a τ satisfying $\tau \leq |\lambda_{min}(M)|$.

If we denote the above upper bound of $\|E\|_\infty$ as a function $\phi(\beta)$, it can be shown that $\phi(\beta)$ has a minimum at $\beta^2 = \xi/\sqrt{n(n-1)}$, an *a priori* choice of β in our UMC method.

The upper bound of $\|E\|_\infty$ in (11) is similar to that for GMW [7]. Hence, like the GMW factorization, our UMC can lead to large perturbations when $\tau \leq |\lambda_{min}(M)|$. In our numerical experiments, we rarely observe this; instead, we often have $\|E\|_\infty = \tau$ even when $\tau \leq |\lambda_{min}(M)|$ (see `Figure` 4, for example). Note that a large τ satisfying $\tau > |\lambda_{min}(M)|$ reduces UMC to the standard Cholesky factorization.

To avoid perturbing a positive-definite matrix, our algorithm can be divided into two phases (in the spirit of the SE MC [26]). We first apply the standard LDL^T factorization to matrix M, stopping at the first occasion that a diagonal element d_j of D becomes negative or very small. We then switch to the second phase, where the modified matrix $M + \tau I$ is applied.

The performance of our UMC on the 42×42 indefinite matrix M is shown in Box 2, following results of other factorizations. Clearly, as τ increases, $\widetilde{M}$ approaches positive definiteness. This is accompanied by a monotonic reduction of the condition number of $\widetilde{M}$. The error bound $\|E\|_\infty$ is equal to τ.

4. Numerical results. We consider four molecular systems for our tests: butane, alanine dipeptide, BPTI, and lysozyme. Butane is a small, 14-atom hydrocarbon molecule with the chemical formula C_4H_{10}. Alanine dipeptide, a blocked alanine residue, consists of 22 atoms. The 58-residue protein BPTI has 568 atoms and thus 1704 Cartesian variables. It is considered "small" by computational chemists. The larger protein lysozyme has 130 residues, 2030 atoms, and 6090 variables.

All computations were performed in double precision in serial mode on an SGI Power Challenge L computer with R10000 processors of speed 195 MHz at New York University. Parameter files were used from CHARMM version 19, but the TNPACK code was implemented into CHARMM version 23 with default TNPACK parameters of [23], unless otherwise stated. These parameters are also listed in the TN algorithm of the appendix. No cutoffs were used for the nonbonded interactions of the potential energy function, and a distance-dependent dielectric function was used. The vector norm $\|\cdot\|$ in all tables and figures is the standard Euclidean norm divided by $\sqrt{n}$, where n is the number of independent variables of a potential energy function. The inner loop of TN is followed as outlined in Algorithm 1, with $\tau = 10$ and $\mathrm{IT}_{\mathrm{PCG}} = 40$ unless otherwise stated.

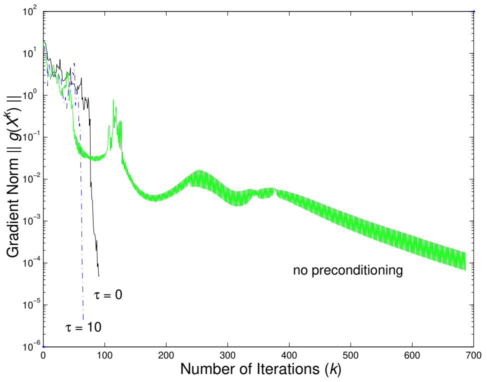


FIG. 1. *TN based on PCG with an indefinite preconditioner performs much better than without preconditioning (even when $\tau = 0$ in UMC) for the minimization of the BPTI potential function.*

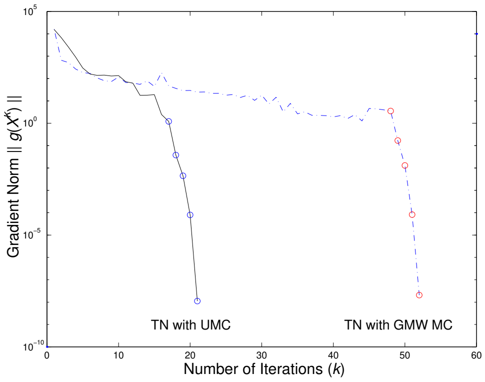


FIG. 2. *The gradient norms generated by TN based on GMW MC vs. UMC for butane minimization. Here circular markers indicate values at the last few steps.*

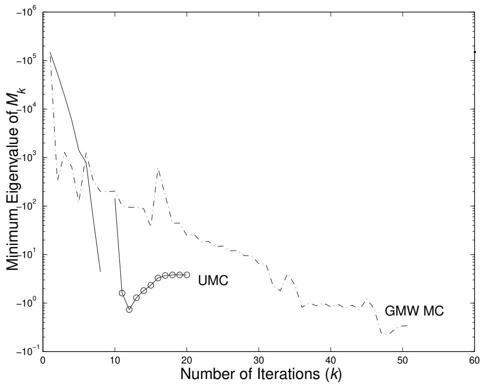


FIG. 3. *The minimum eigenvalue of $\{M_k\}$ resulting from GMW MC vs. UMC for the minimization of a butane molecular system. Circular marks indicate that the modified matrices $\{\widetilde{M_k}\}$ are positive-definite. For $k = 9$, M_k is also positive definite for UMC.*

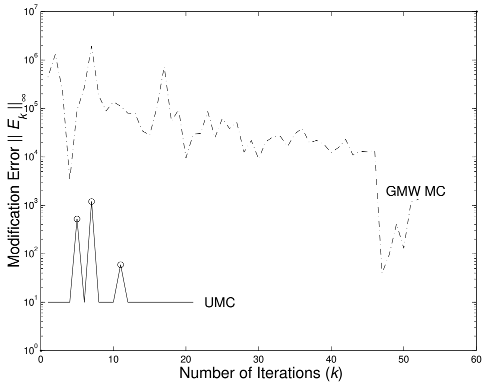


FIG. 4. *The error norms $\{\|E_k\|_\infty\}$ ($E_k = \widetilde{M_k} - M_k$) generated by GMW MC vs. UMC for butane minimization. With $\tau = 10$ for our UMC, $\|E_k\|_\infty = 10$ for all k except the three points indicated by circles.*

4.1. No preconditioning vs. indefinite preconditioning. Figure 1 shows that our TN based on UMC uses far fewer outer iterations than the minimizer without preconditioning for BPTI. We experimented with both $\tau = 0$ and also $\tau = 10$ for the UMC. Since all $\{M_k\}$ were indefinite throughout the TN process, the preconditioner $\widetilde{M_k}$ used by TN was indefinite.

Even better, the total CPU time is much smaller for the indefinite preconditioner version. Namely, the indefinite preconditioner variant required only 8 minutes (for 92 TN iterations and a total of 2390 inner PCG iterations) to find a local minimizer. In contrast, without preconditioning, 80 minutes were required for 687 TN iterations and 27347 CG iterations. This behavior is typical for the molecular systems examined.

TABLE 3

Performance of TNPACK on BPTI based on PCG with an indefinite preconditioner at different values of IT_{PCG}, the maximum number of allowable PCG iterations at each inner loop. For $IT_{PCG} = 300$, the truncation test was used throughout the TN process

IT_{PCG}	Final energy	Final $\|\|g\|\|$	TN outer loops	Total PCG iterations	CPU time (min.)	
					Total	PCG
2	−2780.47	5.9×10^{-4}	1402	2801	27.24	9.81
5	−2769.33	6.8×10^{-5}	233	1139	6.54	3.49
10	−2755.07	5.7×10^{-5}	92	778	3.65	2.31
20	−2756.40	3.2×10^{-5}	73	1114	4.32	3.24
40	−2769.25	2.1×10^{-5}	71	1456	5.13	4.07
120	−2769.25	1.4×10^{-6}	72	2221	7.32	6.25
200	−2775.14	1.0×10^{-6}	143	4640	15.26	13.13
250	−2775.14	1.3×10^{-6}	151	5937	18.86	16.62
300	−2775.14	3.8×10^{-6}	150	6049	18.96	16.74

4.2. Standard MC vs. our UMC. We next compare the performance of TNPACK based on GMW without pivoting [7] and our UMC for butane minimization. Pivoting in GMW was discussed in section 3.1; see Table 2. Efficiency argues for sparsity-based factorization in our context. We further compare our UMC vs. GMW in Figures 2, 3, and 4 for the minimization of the butane potential function.

Figure 2 shows that TN based on the UMC strategy performs favorably in terms of Newton iterations. It also requires less CPU time (0.17 vs. 0.28 sec.; see Table 2). Further, it has a quadratic convergence rate at the last few iterations, as shown by the circles in the figure.

Figures 3 and 4 plot the minimum eigenvalues of $\{M_k\}$ and the values of $\{\|E_k\|_\infty\}$, respectively, where $E_k = \widetilde{M_k} - M_k$. The UMC leads to much smaller modifications of $\{M_k\}$ than the standard MC. Since the minimum eigenvalue of M_k is less than 10 for $k \geq 12$ (circles in Figure 3), our effective preconditioner $\widetilde{M_k}$ is positive definite for $k \geq 12$ with $\|E_k\|_\infty = 10$.

4.3. The importance of the maximum limit on PCG iterations. Table 3 illustrates how TN performs with different values of IT_{PCG} (see Algorithm 1) for BPTI minimization. With $IT_{PCG} = 300$ (last row), the truncation test (step 5 of Algorithm 1) was satisfied throughout TN process. These results can also be visualized in Figures 5 and 6, which show the CPU time and the total number of TN iterations as functions of IT_{PCG}, respectively. The evolution of the gradient norm from TN minimization, corresponding to $IT_{PCG} =$ 40 (leading to the fewest outer iterations) and 300, as a function of the number of TN iterations, is shown in Figure 7. Note the quadratic convergence in the last few steps.

There are several interesting observations from the data of Table 3. As Figure 5 shows, an optimal value for IT_{PCG} can be associated with the smallest CPU time. Here, about 4 minutes resulted from $IT_{PCG} = 10$, much less than about 19 minutes required when $IT_{PCG} = 300$.

Figure 6, however, shows that a somewhat larger value of IT_{PCG} (namely 40) leads to a minimal value of the total number of TN iterations, 71. In contrast, the IT_{PCG} value for optimal CPU time (namely 10) is associated with 92 TN iterations. For reference, a small value, $IT_{PCG} = 2$, gives

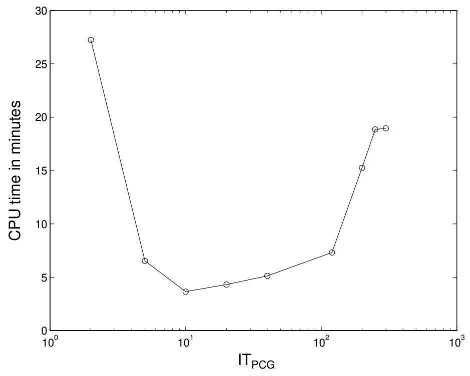


FIG. 5. *An optimal choice for the maximum number of allowable PCG iterations (per TN inner loop) in terms of total CPU time can be seen when TN uses an indefinite preconditioner.*

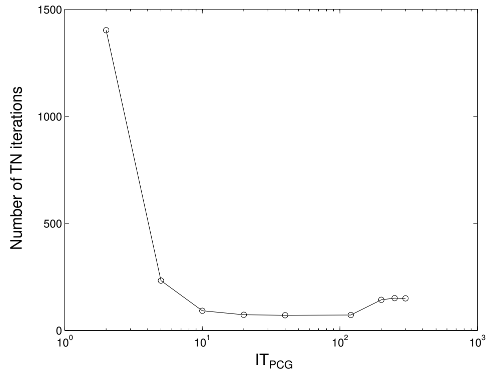


FIG. 6. *An optimal choice for the maximum number of allowable PCG iterations (per TN inner loop) in terms of total number of TN iterations can be seen when TN uses an indefinite preconditioner.*

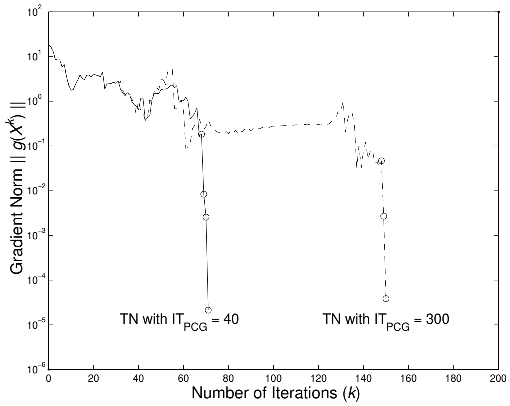


FIG. 7. *TN with a maximum allowable number of 40 PCG iterations per inner loop can more efficiently find a local minimum than when IT_{PCG} is 300. Similar convergence rates are still seen in both paths, as indicated by the circular markers at the last few iterations.*

1402 TN iterations, and a very large $\mathrm{IT_{PCG}}$ gives 150.

In terms of the final energy value obtained for the different variants, we clearly see that several local minima are reached by varying the minimization procedure (six different energy values noted for the nine runs). This multiple-minima problem is beyond the scope of this work. However, we suggest that a larger $\mathrm{IT_{PCG}}$ value might be preferred over a lower one (within a small optimal range) in an attempt to reach lower energy values.

Figures 5 and 6 also show that the range of $\mathrm{IT_{PCG}}$ for which TN performs better than the minimizer based on the truncation test alone is fairly large, here between 10 to 120. Based on this observation and the point above regarding larger $\mathrm{IT_{PCG}}$ for smaller final energy values, we set $\mathrm{IT_{PCG}} = 40$ for the numerical experiments presented below.

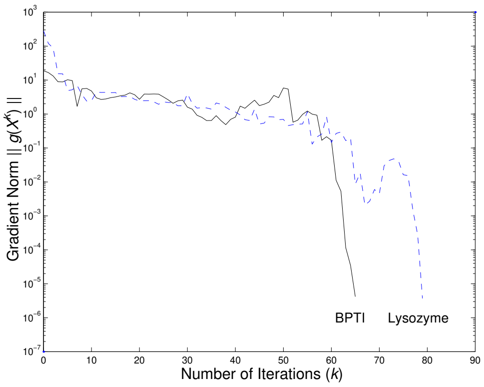


FIG. 8. *Evolution of the gradient norms generated by TNPACK for BPTI (solid) and lysozyme (dashed).*

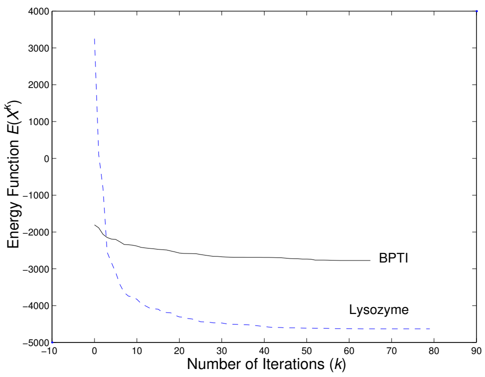


FIG. 9. *The decrease of the potential energy function for BPTI (solid) and lysozyme (dashed) by TNPACK.*

TABLE 4
Comparison of TNPACK with two other CHARMM minimizers and LM-BFGS

Minimizer	Iterations	Final E	Final $\|\|g\|\|$	E & g evals.	CPU time
Butane (42 variables)					
TN	21 (62)*	3.904	1.1×10^{-8}	26	0.17 sec.
LM-BFGS	93	3.904	7.7×10^{-7}	101	0.14
LM-BFGS (P)	133	4.753	9.4×10^{-7}	148	0.33
ABNR	368	3.904	9.8×10^{-8}	368	0.32
CONJ	127	3.904	9.1×10^{-7}	307	0.17
Alanine dipeptide (66 variables)					
TN	29 (210)	−15.25	7.67×10^{-11}	44	1.12 sec.
LM-BFGS	711	−15.25	1.4×10^{-6}	740	1.39
LM-BFGS (P)	367	−15.25	1.3×10^{-6}	378	1.97
ABNR	16466	−15.25	9.9×10^{-8}	16467	7.47
CONJ	882	−15.25	9.83×10^{-7}	2507	2.34
BPTI (1704 variables)					
TN	65 (1335)	−2773.70	4.2×10^{-6}	240	5.21 min.
LM-BFGS	4486	−2792.96	6.3×10^{-5}	4622	12.61
LM-BFGS (P)	3929	−2792.92	5.9×10^{-5}	3946	64.2
ABNR	8329	−2792.96	8.9×10^{-6}	8330	25.17
CONJ	12469	−2792.93	9.9×10^{-6}	32661	97.8
Lysozyme (6090 variables)					
TN	79 (1841)	−4631.38	3.7×10^{-6}	244	1.54 hrs.
LM-BFGS	5546	−4617.21	1.4×10^{-4}	5711	4.26
LM-BFGS (P)	3331	−4620.27	1.6×10^{-4}	3374	12.92
ABNR	7637	−4605.94	9.9×10^{-6}	7638	6.11
CONJ	9231	−4628.36	9.9×10^{-5}	24064	19.63

* The number in parentheses is the total number of PCG iterations.

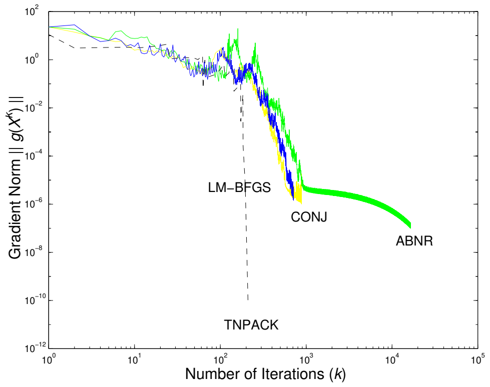


FIG. 10. *A comparison of gradient norms generated by TNPACK, ABNR, CONJ, and LM-BFGS for alanine dipeptide minimization.*

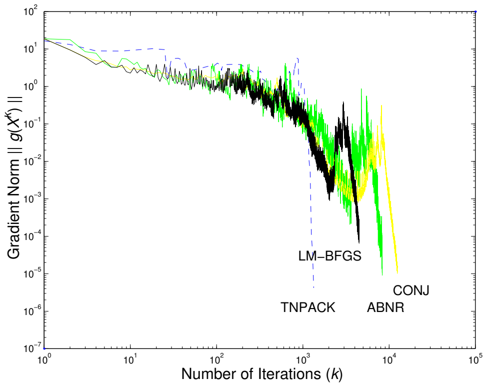


FIG. 11. *A comparison of gradient norms generated by TNPACK, ABNR, CONJ, and LM-BFGS for BPTI minimization.*

4.4. Comparison to other minimization schemes. We now compare, in Table 4, the minimization performance of TNPACK with two other CHARMM minimizers, ABNR (an adopted basis Newton-Raphson method) and CONJ (a nonlinear conjugate gradient method), as well as with LM-BFGS, with $u = 5$ stored updates [11]. For LM-BFGS we test no-preconditioning as well as preconditioning options. The preconditioning strategy used for LM-BFGS was described by Schlick [21]. Briefly, the initial search vector in each sequence of LM-BFGS updates is set as the solution p_k to the system

$$M_k p_k = -g_k, \tag{12}$$

where M_k is defined as before, so that M_k replaces the initial approximation to the Hessian. To solve (12) in LM-BFGS we use the standard GMW MC. We expect preconditioning in LM-BFGS to reduce the number of function evaluations significantly, but this must be balanced with the added cost involved in evaluating and factoring the preconditioner.

In all computations, we used the default parameters in CHARMM for the minimizers. No cutoffs for the nonbonded terms were used to avoid formation of artificial minima that result when the nonbonded terms are turned off at some distance, even when this is done smoothly. We also used the same convergence test (i.e., inequality (B3d) in the Appendix B with $\epsilon_g = 10^{-6}$) for TNPACK, ABNR, CONJ, and LM-BFGS. Both TNPACK and ABNR can reach much lower gradient norms than CONJ.

For butane and alanine dipeptide, all minimizers (except for one case: LM-BFGS with preconditioning for butane[2]) find the same minimum value, while for BPTI and lysozyme different minima are obtained. This is a consequence of different paths taken toward a local minimum in each case. The results in Table 4 show that TNPACK requires less CPU time than the other methods

[2] For butane, the global minimum corresponds to an open chain configuration ("trans-staggered"), with the central dihedral angle φ, defining the relative orientation of the four carbons, adopting the value $-180°$; the higher-energy minimum corresponds to a more compact configuration, with φ about $-65°$.

and reaches very low gradient norms. The results for LM-BFGS show how preconditioning tends to reduce the total number of iterations but to increase the CPU time. For the proteins, the CPU time of TNPACK is less than that of the best LM-BFGS variant by a factor of 2 to 3.

In Figure 8 we illustrate the evolution of the gradient norm for BPTI and lysozyme molecular systems for TNPACK, along with their energy decreases in Figure 9.

In Figures. 10 and 11, we compare the gradient norm evolution for TNPACK, ABNR, CONJ, and LM-BFGS (no preconditioning) for the dipeptide and BPTI. For TNPACK, the ''iteration'' value in the abscissa corresponds to the accumulated number of PCG iterations.

The relative importance of updating and preconditioning in LM-BFGS was discussed in [21] by testing preconditioning with various numbers of stored updates (i.e., $u = 0, 1, 2, 3, 4, 5$). It was found that the relative importance of these factors in generating performance improvement depends on the initial guess for the minimum---preconditioning is more important when the initial guess is better. Our experiments here with different numbers of updates for the LM-BFGS version without preconditioning revealed that $u = 4$ or 5 is optimal in terms of CPU time (data not shown); when preconditioning is used, the optimal u tends to be lower (e.g., $u = 2$ or 3).

5. Conclusions. We have suggested the use of an indefinite rather than a positive-definite preconditioner in the TN optimization method applied to large-scale, highly nonlinear functions with problem-formulated preconditioners. With the UMC applied to solve a linear system involving the preconditioner, we guarantee that the resulting search vectors are directions of descent. Thus, convergence to a local minimum can be derived as in classic TN methods.

An indefinite preconditioner makes sense in our applications for efficiency considerations. Namely, the sparse preconditioner generated from the local chemical interactions [25] can have large negative eigenvalues, and other MC schemes [3, 7, 8, 26] (when used with PCG for solving such preconditioned linear systems) tend to exhibit poor numerical behavior when very large modifications are permitted. This leads to many PCG iterations and large CPU times for the overall minimization method. We overcome this difficulty by proposing the UMC to prescribe matrix modifications τI in a numerically stable manner. The parameter τ is chosen heuristically, so as to lead to positive-definite preconditions near a minimum. This bound appears insensitive to the problem size, and in our application we use $\tau = 10$. Undoubtedly, there are other ways to factor a symmetric matrix M in this way.

The numerical experiments reported here highlight that the unconventional use of an indefinite preconditioner works better than the minimizer without preconditioning, as well as other minimizers available in CHARMM (ABNR and CONJ). A competitive method tested is also LM-BFGS, where we examined both with and without preconditioning. Although preconditioning reduces the total number of iterations in LM-BFGS, it increases the CPU time because of the added cost of the linear system. Results show that TNPACK requires less CPU time than the other methods tested for large potential energy problems. Very recently we have updated the program routines of TNPACK/CHARMM to significantly reduce memory requirements by using a specified sparsity pattern for the preconditioner and finite-differences for Hessian/vector multiplication.

These developments, including applications to problems with up to 35,000 variables, will be reported separately.

These algorithmic suggestions may open new opportunities for other large-scale optimization problems in which partial second-derivative information might be exploited in the TN framework. Particularly interesting is the possibility of using TNPACK as a local minimizer in the context of a stochastic global optimization method. The different local minima reached for the proteins in this work suggest that even a simple global aspect added to the local minimizer can be of practical importance.

Acknowledgments. We are indebted to the referees and the reviewing editor for valuable comments.

Appendix A. Analyses for the PCG inner loop of TN

We consider the PCG algorithm of Algorithm 1 for solving Newton equation $H_k p = -g_k$ with a preconditioner M_k, where both H_k and M_k are nonsingular but not necessarily positive definite. We assume that there exists a positive integer l such that $d_j^T H_k d_j \neq 0$ and $r_j^T M_k^{-1} r_j \neq 0$ for $j = 1, 2, \ldots, l$, and thus the PCG iterates $\{p_j\}_{j=1}^{l}$ are well defined. As for the standard case (i.e., M_k is positive definite) [9], it follows that the PCG residual vectors $\{r_j\}_{j=1}^{l}$ $(r_j \equiv -g_k - H_k p_j)$ satisfy in exact arithmetic

$$r_i^T M_k^{-1} r_j = 0 \quad \text{for} \quad 1 \leq i < j \leq l. \tag{13}$$

Theorem .1 (motivation for not using d_j as exit search direction). *Let M_k be nonsingular and the initial guess $p_1 = 0$. Then, if $g_k^T M_k^{-1} r_j = 0$ for $1 < j \leq l$, all vectors d_j for $1 \leq j \leq l$ satisfy*

$$g_k^T d_j = -r_j^T M_k^{-1} r_j. \tag{14}$$

Proof. Since $r_1 = -g_k$, from (13) it follows that

$$g_k^T M_k^{-1} r_{j+1} = -r_1^T M_k^{-1} r_{j+1} = 0 \quad \text{for } 1 \leq j \leq l-1.$$

Thus, for all $j = 1, 2, \ldots, l-1$,

$$\begin{aligned} g_k^T d_{j+1} &= g_k^T (M_k^{-1} r_{j+1} + \beta_j d_j) \\ &= g_k^T M_k^{-1} r_{j+1} + \beta_j g_k^T d_j \\ &= \beta_j g_k^T d_j. \end{aligned}$$

Noting that $\beta_j = r_{j+1}^T M_k^{-1} r_{j+1} / r_j^T M_k^{-1} r_j, r_1 = -g_k$, and $d_1 = -M_k^{-1} g_k$, we obtain

$$g_k^T d_{j+1} = \beta_j \beta_{j-1} \cdots \beta_1 g_k^T d_1$$

$$
\begin{aligned}
&= -\frac{r_{j+1}^T M_k^{-1} r_{j+1}}{r_1^T M_k^{-1} r_1} g_k^T M_k^{-1} g_k \\
&= -r_{j+1}^T M_k^{-1} r_{j+1}. \qquad \square
\end{aligned}
$$

From Theorem 1 it follows that d_j may be an ascent direction for the case of an indefinite M_k.

THEOREM .2 (MOTIVATION FOR USING AN INDEFINITE PRECONDITIONER IN TN). *Let M_k be nonsingular, and the initial guess $p_1 = 0$. Then, if $d_j^T H_k d_j > 0$ and $r_j^T M_k^{-1} r_j \neq 0$ for $j = 1, 2, \ldots, l$, all p_j with $2 \leq j \leq l+1$ are descent directions and satisfy*

$$
g_k^T p_{l+1} < \cdots < g_k^T p_j < g_k^T p_{j-1} < \cdots < g_k^T p_2 < 0. \tag{15}
$$

Proof. Using $\alpha_j = r_j^T M_k^{-1} r_j / d_j^T H_k d_j$ and (14), we have

$$
g_k^T p_{j+1} = g_k^T p_j + \alpha_j g_k^T d_j = g_k^T p_j - \frac{(r_j^T M_k^{-1} r_j)^2}{d_j^T H_k d_j}. \tag{16}
$$

Relation (16) together with $d_j^T H_k d_j > 0$ and $(r_j^T M_k^{-1} r_j)^2 > 0$ give $g_k^T p_{j+1} < g_k^T p_j$ for $j = 1, 2, \ldots, l$. In particular, $g_k^T p_2 < g_k^T p_1 = 0$ as $p_1 = 0$. Therefore, it follows that all p_j with $2 \leq j \leq l+1$ satisfy (15). $\square$

From Theorem 2 it follows that the PCG directions of Algorithm 1 with an indefinite preconditioner M_k are directions of descent.

THEOREM .3 (EQUIVALENCE OF THE DESCENT DIRECTION AND NEGATIVE CURVATURE TESTS). *Let M_k be nonsingular and $r_j^T M_k^{-1} r_j \neq 0$. Then $g_k^T p_{j+1} > g_k^T p_j$ if and only if $d_j^T H_k d_j < 0$.*

Proof. From (14) and $\alpha_j = r_j^T M_k^{-1} r_j / d_j^T H_k d_j$, we have

$$
\alpha_j g_k^T d_j = -(r_j^T M_k^{-1} r_j)^2 / d_j^T H_k d_j.
$$

Thus, if the denominator $d_j^T H_k d_j < 0$, then the left-hand side $\alpha_j g_k^T d_j > 0$, implying that

$$
g_k^T p_{j+1} = g_k^T p_j + \alpha_j g_k^T d_j > g_k^T p_j.
$$

On the other hand, if $g_k^T p_{j+1} > g_k^T p_j$, then

$$
-(r_j^T M_k^{-1} r_j)^2 / d_j^T H_k d_j = \alpha_j g_k^T d_j = g_k^T p_{j+1} - g_k^T p_j > 0,
$$

which implies that $d_j^T H_k d_j < 0$. $\square$

From Theorem 3 it follows that the descent direction test of Algorithm 1 is equivalent to the negative curvature test.

THEOREM .4 (ANOTHER MOTIVATION FOR USING p_j RATHER THAN d_j AS EXIT SEARCH DIRECTION). *Let both H_k and M_k be positive definite. Then there exists an index $j_0 > 0$ such that for all $i \geq 2$*

$$g_k^T p_i < g_k^T d_j < 0 \quad \text{whenever} \quad j > j_0. \tag{17}$$

Proof. When both H_k and M_k are positive definite, from PCG theory we know that $r_j^T M_k^{-1} r_j$ approaches zero as j increases. Subsequently, there exists $j_0 > 0$ such that $r_j^T M_k^{-1} r_j < |g_k^T p_2|$ whenever $j > j_0$. Together with (14) and (15), we have $g_k^T p_2 < 0$, and

$$g_k^T d_j = -r_j^T M_k^{-1} r_j > g_k^T p_2 > g_k^T p_i \quad \text{for all } i > 2. \qquad \square$$

The steplength λ can have a larger range of feasibility to satisfy

$$E(X^k + \lambda P^k) < E(X^k)$$

with a larger value of $|g_k^T P^k|$, where $g_k^T P^k$ is negative. Thus, the objective function value may be reduced more on a larger range of λ. In this sense, Theorem 4 suggests that p_j is a ''better'' search direction than d_j because choosing the search direction $P^k = p_j$ for $j \geq 2$ can lead to further reduction than using $P^k = d_j$ for a sufficiently large j. Similarly, Theorem 2 suggests that a PCG iterate p_j is better than p_i when $j > i$.

APPENDIX B. The TN algorithm

The TN algorithm based on the PCG method consists of an outer and an inner loop. We present these two loops in turn, listing the parameter values used in the numerical examples reported in this paper (unless specified otherwise in text). The new algorithmic components introduced in this paper are marked by asterisks. We denote the objective function to be minimized by E; the gradient vector and Hessian matrix of E by g and H, respectively; and the preconditioner for PCG by M. We omit the subscript k from g, H, and M for clarity.

OUTER LOOP OF THE TN METHOD

1. *Initialization*
- Set $k = 0$ and evaluate $E(X^0)$ and $g(X^0)$ for a given initial guess X^0.
- If $||g(X^0)|| < 10^{-8} \max(1, ||X^0||)$, exit algorithm, where $||\cdot||$ is the standard Euclidean norm divided by $\sqrt{n}$.

2. *Preparation for UMC*
- Evaluate the preconditioner M at X^0 by assembling only the local potential energy terms (bond length, bond angle and dihedral angle components).
- Determine the sparsity pattern of M. The upper triangle of M is stored in a compressed row format, and the pattern is specified by two integer arrays that serve as row and column pointers [23].
- Compute the *symbolic factorization* LDL^T of M, that is, the sparsity structure of the factor L.

- Evaluate the Hessian matrix H at X^0.

3. *Inner loop*

Compute a search vector P^k by solving the Newton equation $HP = -g$ *approximately* using PCG with preconditioner M based on the UMC method (see below).

4*. *Line search*

- Compute a steplength λ by safeguarded cubic and quadratic interpolation [12] (see also [27] for a minor modification that avoids a too small acceptable steplength λ) so that the update $X^{k+1} = X^k + \lambda P^k$ satisfies

$$E(X^{k+1}) \le E(X^k) + \alpha \lambda g(X^k)^T P^k \quad \text{and} \quad |g(X^{k+1})^T P^k| \le \beta |g(X^k)^T P^k|,$$

where $\alpha = 10^{-4}$ and $\beta = 0.9$.

5. *Convergence tests*

- Check the following inequalities:

$$E(X^{k+1}) - E(X^k) < \epsilon_f (1 + |E(X^{k+1})|) \ , \qquad (B1a)$$

$$||X^{k+1} - X^k|| < \sqrt{\epsilon_f}\ (1 + ||X^{k+1}||)/100 \ , \qquad (B1b)$$

$$||g(X^{k+1})|| < \epsilon_f^{1/3} (1 + |E(X^{k+1})|) \ , \qquad (B1c)$$

$$||g(X^{k+1})|| < \epsilon_g (1 + |E(X^{k+1})|) \ , \qquad (B1d)$$

where $\epsilon_f = 10^{-10}$ and $\epsilon_g = 10^{-8}$.
If conditions (B1a), (B1b), (B1c), or (B1d) are satisfied, exit algorithm.

6. *Preparation for the next Newton step*

- Compute the preconditioner M at X^{k+1} by using the pattern determined originally.
- Evaluate the Hessian matrix H at X^{k+1}.
- Set $k \to k+1$, and go to step 3.

INNER LOOP OF THE TN METHOD (Step 3 of Outer Loop)

The sequence $\{p_j\}$ below represents the PCG vectors used to construct P^k in step 3 of **Outer loop**.

1. *Initialization*
   - Set $j = 1$, $p_1 = 0$, and $r_1 = -g$.
   - Set the parameters $\eta_k = \min\{c_r/k\ , ||g||\}$ and IT_{PCG} for the truncation test in step 5. We use $c_r = 0.5$ and $\text{IT}_{\text{PCG}} = 40$.

2*. *The UMC factorization*
   - Perform the UMC of M so that the resulting effective preconditioner is $\widetilde{M} = LDL^T$ with a chosen parameter τ (we use $\tau = 10$). The factor L is stored in the same sparse row format used for M.

- `Solve for` z_j `in` $\widetilde{M} z_j = r_j$ `by using the triangular systems`

$$Lx = r_j \quad \texttt{and} \quad L^T z_j = D^{-1} x.$$

- `Set` $d_j = z_j$.

3*. *Singularity test*
`Compute the matrix-vector product` $q_j = H d_j$.
`If either` $|r_j^T z_j| \le \delta$ `or` $|d_j^T q_j| \le \delta$ `(e.g.,` $\delta = 10^{-10}$),
`exit PCG loop with` $P^k = p_j$ `(for` $j = 1$`, set` $P^k = -g_k$`).`

4*. *Implement one of the following two tests*

4a. [THE DESCENT DIRECTION TEST]
`Update the quantities`

$$\alpha_j = r_j^T z_j \,/\, d_j^T q_j \quad \texttt{and} \quad p_{j+1} = p_j + \alpha_j d_j. \qquad (B2)$$

If $g^T p_{j+1} \ge g^T p_j + \delta$,
exit `inner loop with` $P^k = p_j$ `(for` $j = 1$`, set` $P^k = -g$`).`

4b. [THE STANDARD NEGATIVE CURVATURE TEST]
`if` $d_j^T q_j \le \delta(d_j^T d_j)$,
exit `inner loop with` $P^k = p_j$ `(for` $j = 1$`, set` $P^k = -g$`);`
else `update` α_j `and` p_{j+1} `as in (B2).`

5*. *Truncation test*
- `Compute` $r_{j+1} = r_j - \alpha_j q_j$.
- *If* $\|r_{j+1}\| \le \eta_k \|g\|$ `or` $j + 1 > \mathtt{IT}_{\mathtt{PCG}}$,
exit `inner loop with search direction` $P^k = p_{j+1}$.

6*. *Continuation of PCG*
- `Solve for` z_{j+1} `as in step 2 in` $\widetilde{M} z_{j+1} = r_{j+1}$.
- `Update the quantities`

$$\beta_j = r_{j+1}^T z_{j+1} \,/\, r_j^T z_j \quad \texttt{and} \quad d_{j+1} = z_{j+1} + \beta_j d_j. \qquad (B3)$$

- `Set` $j \leftarrow j + 1$, `and go to step 3`

REFERENCES

[1] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus, *CHARMM: A program for macromolecular energy, minimization, and dynamics calculations*, J. Comp. Chem., 4 (1983), pp. 187–217.

[2] J. R. Bunch and L. Kaufman, *Some stable methods for calculating inertia and solving symmetric linear systems*, Math. Comp., 31 (1977), pp. 163–655.

[3] S. H. Cheng and N. J. Higham, *A modified Cholesky algorithm based on a symmetric indefinite factorization*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 1097–1110, also available as Numerical Analysis Report 289, University of Manchester, England, April 1996.

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

[5] P. Derreumaux, G. Zhang, B. Brooks, and T. Schlick, *A truncated-Newton method adapted for CHARMM and biomolecular applications,* J. Comp. Chem., 15 (1994), pp. 532–552.

[6] A. Forsgren, P. E. Gill and W. Murray, *Computing modified Newton directions using a partial Cholesky factorization*, SIAM J. Sci. Comput., 16 (1995), pp. 139–150.

[7] P. E. Gill and W. Murray, *Newton-type methods for unconstrained and linearly constrained optimization,* Math. Prog., 28 (1974), pp. 311–350.

[8] P. E. Gill, W. Murray, and M. H. Wright, *Practical optimization*, Academic Press, London, 1983.
[9] G. H. Golub and C. F. van Loan, *Matrix Computations*, 2nd ed., John Hopkins University Press, Baltimore, MD, 1986.
[10] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.
[11] D. C. Liu and J. Nocedal, *On the limited memory BFGS method for large-scale optimization*, Math. Prog., 45 (1989), pp. 503–528.
[12] J. J. Moré and D. J. Thuente, *Line search algorithms with guaranteed sufficient decrease*, ACM Trans. Math. Software, 20 (1994), pp. 286–307.
[13] S. G. Nash, *Newton-type minimization via the Lanczos method*, SIAM J. Numer. Anal., 21 (1984), pp. 770–788.
[14] S. G. Nash, *Preconditioning of truncated-Newton methods,* SIAM J. Sci. Statist. Comput., 6 (1985), pp. 599–616.
[15] 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.
[16] A. Nyberg and T. Schlick, *A computational investigation of dynamic properties with the implicit-Euler scheme for molecular dynamics simulations*, J. Chem. Phys., 95 (1991), pp. 4986–4996.
[17] P.M. Pardalos, D. Shalloway and G. Xue, Eds., *Global minimization of nonconvex energy function: Molecular Conformation and Protein Folding*, DIMACS ser. Discrete Math. Theoret. Comput. Sci., 23, AMS, Providence, RI, 1996.
[18] P. M. Pardalos and G. Xue, Eds, *Special issue on computer simulations in molecular and protein conformations*, J. Global Optim., 11 (1997).
[19] M. H. Schultz, S. C. Eisenstat and A. H. Sherman, *Algorithms and data structures for sparse symmetric Gaussian elimination,* SIAM J. Sci. Statist. Comput., 2 (1981), pp. 225–237.
[20] M. H. Schultz, S. C. Eisenstat, M. C. Gursky and A. H. Sherman, *Yale sparse matrix package, I. The symmetric codes,* Internat. J. Numer. Meth. Engrg., 18 (1982), pp. 1145–1151.
[21] T. Schlick, *Optimization methods in computational chemistry*, in Reviews in Computational Chemistry III, pp. 1–71, K. B. Lipkowitz and D. B. Boyd, eds., VCH Publishers, New York, New York, 1992.
[22] T. Schlick, *Modified Cholesky factorizations for sparse preconditioners,* SIAM J. Sci. Comput., 14 (1993), pp. 424–445.
[23] T. Schlick and A. Fogelson, *TNPACK — A truncated Newton minimization package for large-scale problems:* I. *Algorithm and usage*, ACM Trans. Math. Software., 18 (1992), pp. 46–70.
[24] 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.
[25] T. Schlick and M. L. Overton, *A powerful truncated Newton method for potential energy functions.* J. Comp. Chem., 8 (1987), pp. 1025–1039.
[26] R. B. Schnabel and E. Eskow, *A new modified Cholesky factorization*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 1136–1158.
[27] D. Xie and T. Schlick, *A more lenient stopping rule for line search algorithms*, 1998, submitted.
[28] D. Xie and T. Schlick, *Remark on the updated truncated Newton minimization package, Algorithm 702*, ACM Trans. Math. Software., 25 (March 1999).
[29] X. Zou, I. M. Navon, M. Berger, K. H. Phua, T. Schlick, F. X. Le Dimet, *Numerical experience with limited-memory quasi-Newton and truncated Newton methods*, SIAM J. Opt., 3 (1993), pp. 582–608.