An Effective Compressed Sparse Preconditioner for Large Scale Biomolecular Simulations*

Dexuan Xie

Department of Mathematical Sciences, University of Wisconsin,
Milwaukee, WI 53201-0413, USA
dxie@uwm.edu

Abstract. The natural preconditioner defined by local potentials is effective in the truncated Newton method for minimizing large scale biomolecular potential energy functions. This paper extends its definition, and proposes an algorithm for generating the sparse pattern of the preconditioner from the primary structure of a molecular system with N atoms. It shows that the implementation of the new compressed sparse preconditioner requires only a linear order of N memory locations.

1 Introduction

The natural preconditioner M defined by local potentials [7] can significantly improve the convergence and performance of the CHARMM version of TNPACK (the truncated Newton program package [6]) for minimizing the potential energy function of a biomolecular system with N atoms [3, 8–10]. Here CHARMM is one widely-used molecular mechanics and dynamics program [1]. While the truncated Newton (TN) method is a second derivative minimization method [2], it can be simply modified into a first derivative method by using a finite difference approximation to the product of the Hessian matrix (i.e., the second derivative of the potential function) with a vector. Thus, the total memory requirement of TNPACK can be reduced to a linear order of N if the preconditioner M can be formulated in a compress format by a linear order of N memory locations. If so, the size of a biomolecular system that TNPACK can be applied to will be sharply increased. However, since the sparse pattern of M is extremely irregular, which varies with molecular systems, it is difficult to compress M without employing a full $3N \times 3N$ matrix. How to formulate a compressed M by a linear order of N memory locations has been an open problem for a long time.

This paper solves this problem and extends the definition of the preconditioner using the spherical cutoff approach to improve the efficiency of the preconditioner for a biomolecular system containing a large number of water molecules. Note that a sparse Hessian matrix has been programmed in CHARMM in a

* This work was supported by the National Science Foundation through grant DMS-0241236, and, in part, by the Graduate School Research Committee Award (343267-101-4) of the University of Wisconsin-Milwaukee.

J. Zhang, J.-H. He, and Y. Fu (Eds.): CIS 2004, LNCS 3314, pp. 64–70, 2004.

special compress format (which will be referred to as the CHARMM compress format for clarity) [4]. Hence, a new scheme is first developed to generate the sparse pattern of the preconditioner from the primary structure of a biomolecular system, along with a given spherical cutoff condition, and express it in the CHARMM compress format. With such a sparse pattern, the preconditioner is then defined as a compressed sparse Hessian matrix so that the program routines for evaluating the preconditioner can be easily created by adapting the current CHARMM program routines. Finally, the compressed preconditioner is converted from the CHARMM compress format to the YSMP (Yale Sparse Matrix Package [5]) compress format, resulting in the compressed sparse preconditioner suitable for TNPACK. It is shown that the new compressed preconditioner can be formulated and evaluated in a linear order of N memory locations.

The remainder of this paper is organized as follows. Section 2 defines the new natural preconditioner. Section 3 expresses the compressed preconditioner in both CHARMM and YSMP formats. Section 4 describes a scheme for formulating the sparse pattern of the preconditioner directly from the primary structure of a molecular system. Section 5 presents a scheme for converting the CHARMM compress format to the YSMP compress format. Numerical results will be reported in the subsequent paper.

2 The New Definition of the Natural Preconditioner

Let $E(X)$ be a biomolecular potential energy function with $X = (X_1, X_2, \ldots, X_N)$, where $X_i = (x_i, y_i, z_i)$ denotes the position of atom i for $i = 1, 2, \ldots, N$. An important task in biomolecular simulations is to find a minimum point of $E(X)$ over the $3N$ dimensional space, which often leads to a feasible conformation of the molecular system. In general, E includes the local potential E_{local}, the van der Waals potential energy, and the Coulomb potential energy. E_{local} is defined as the sum of the bond length, bond angle, and torsional potential terms. See [1] for a detail description of the biomolecular potential energy function.

With a given E_{local} and a given nonnegative value of η, the sparse pattern set P is introduced as below:

$$P = \{(i,j) | \frac{\partial^2 E_{local}(X)}{\partial X_i \partial X_j} \neq 0 \text{ or } \|X_i - X_j\|_2 \leq \eta \text{ for } i < j, i, j = 1, 2, \ldots, N\}, \tag{1}$$

where $\|X_i - X_j\|_2 = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$.

The natural preconditioner is a sparse $N \times N$ symmetric block matrix M defined by $M = (M_{ij})_{i,j=1}^N$ with block entry M_{ij} as given below:

$$M_{ij} = \begin{cases} \frac{\partial^2 E(X)}{\partial X_i \partial X_j} & \text{if } (i,j) \in P \\ \mathbf{0} & \text{otherwise} \end{cases} \text{ for } i \leq j, \quad \text{and} \quad M_{ij} = M_{ji} \quad \text{for } i > j, \tag{2}$$

where $i, j = 1, 2, \ldots, N$, $\mathbf{0}$ denotes a 3×3 matrix of zero, and

$$\frac{\partial^2 E(X)}{\partial X_i \partial X_j} = \begin{bmatrix} \frac{\partial^2 E}{\partial x_i \partial x_j} & \frac{\partial^2 E}{\partial x_i \partial y_j} & \frac{\partial^2 E}{\partial x_i \partial z_j} \\ \frac{\partial^2 E}{\partial y_i \partial x_j} & \frac{\partial^2 E}{\partial y_i \partial y_j} & \frac{\partial^2 E}{\partial y_i \partial z_j} \\ \frac{\partial^2 E}{\partial z_i \partial x_j} & \frac{\partial^2 E}{\partial z_i \partial y_j} & \frac{\partial^2 E}{\partial z_i \partial z_j} \end{bmatrix}. \tag{3}$$

In [7], M was defined as the Hessian matrix of E_{local} . Clearly, replacing E_{local} by E and adding the cutoff condition $\|X_i - X_j\|_2 \leq \eta$ in the definition of M in (1) and (2) can significantly improve the approximation of the preconditioner to the Hessian matrix of E. Hence, the new preconditioner is expected to be more effective than the traditional one in [7] for TNPACK.

The preconditioner M is very sparse. It has only a linear order of N nonzero entries. In fact, each local potential term of E_{local} involves at most three bond connections, and the value of η is small (a default value of η in TNPACK is 4 Å). Hence, the total number of nonzero block entries on each row of M is a small fixed number. Thus, M has only a linear order of N nonzero entries.

3 The Preconditioner in Two Compress Formats

In this section, the preconditioner is compressed in the CHARMM and YSMP formats, respectively. For clarity, the ith row of the upper triangle part of M is assumed to have $n_i + 1$ nonzero block entries M_{ij} with column indices $j = i, \nu_{i1}, \nu_{i2}, \ldots, \nu_{i,n_i}$ satisfying

$$i < \nu_{i1} < \nu_{i2} < \ldots < \nu_{in_i} \text{ and } n_i \geq 0 \text{ for } i = 1, 2, \ldots, N. \tag{4}$$

Thus, the sparse pattern set P can be rewritten as

$$P = \{(i, j) | j = i, \nu_{i1}, \nu_{i2}, \ldots, \nu_{i,n_i} \text{ satisfying (4) for } i = 1, 2, \ldots, N\}. \tag{5}$$

The block entry M_{ij} is also denoted by

$$M_{ij} = \begin{bmatrix} m_{\mu,\nu} & m_{\mu,\nu+1} & m_{\mu,\nu+2} \\ m_{\mu+1,\nu} & m_{\mu+1,\nu+1} & m_{\mu+1,\nu+2} \\ m_{\mu+2,\nu} & m_{\mu+2,,\nu+1} & m_{\mu+2,\nu+2} \end{bmatrix},$$

where $\mu = 3i - 2$ and $\nu = 3j - 2$ for $i, j = 1, 2, \ldots, N$. It is only needed to store the upper triangle part of M since M is symmetric.

In the CHARMM compress format, a sparse matrix is stored into two integer arrays $\mathcal{C}$ and $\mathcal{P}$ and one real array $\mathcal{M}$, where $\mathcal{M}$ holds the nonzero entries of M, $\mathcal{C}$ the column indices of the nonzero block entries in $\mathcal{M}$, and $\mathcal{P}$ the position of the column index ν_{i1} in $\mathcal{C}$ for $i = 1, 2, \ldots, N$. For the preconditioner M with the sparse pattern set P given in (5), the three arrays $\mathcal{M}$, $\mathcal{C}$ and $\mathcal{P}$ can be found as below:

$$\mathcal{C} = (\nu_{11}, \nu_{12}, \ldots, \nu_{1,n_1}, \nu_{21}, \nu_{22}, \ldots, \nu_{2,n_2}, \ldots, \nu_{N1}, \nu_{N2}, \ldots, \nu_{N,n_N}), \tag{6}$$

$\mathcal{P} = (\mathcal{P}(1), \mathcal{P}(2), \ldots, \mathcal{P}(N))$ with the ith component $\mathcal{P}(i)$ being defined by

$$\mathcal{P}(1) = 1 \quad \text{and} \quad \mathcal{P}(i) = 1 + \sum_{k=1}^{i-1} n_k \quad \text{for } i = 2, 3, \ldots, N, \tag{7}$$

and $\mathcal{M} = (\mathcal{R}(1), \mathcal{R}(2), \ldots, \mathcal{R}(N))$ with $\mathcal{R}(i)$ holding the $n_i + 1$ nonzero block entries of row i in the form

$$\mathcal{R}(i) = (\mathcal{D}(i), \mathcal{U}(i, \nu_{i1}), \mathcal{U}(i, \nu_{i2}), \ldots, \mathcal{U}(i, \nu_{in_i})).$$

Here $\mathcal{D}(i)$ holds the upper triangle part of M_{ii}, and $\mathcal{U}(i, j)$ the 9 entries of M_{ij} with $i < j$ in a row by row pattern.

In terms of $\mathcal{C}$ and $\mathcal{P}$, the column index ν_{ij} and number n_i are expressed as

$$\nu_{i,j} = \mathcal{C}(\mathcal{P}(i) + j - 1) \quad \text{and} \quad n_i = \mathcal{P}(i + 1) - \mathcal{P}(i), \tag{8}$$

where $j = 1, 2, \ldots, n_i$, and $i = 1, 2, \ldots, N$.

In the YSMP compress format, a sparse matrix is stored into two integer arrays IA and JA and one real array A, where A holds the nonzero entries of the matrix in a row-wise fashion, JA the column indices of the entries of A, and IA is a pointer vector in which the ith entry $IA(i)$ points to the location of the first nonzero entry of the i-th row of the matrix in vector A. If $A(k) = m_{ij}$, where m_{ij} is the first nonzero entry of the i-th row, then $IA(i)$ and $JA(k)$ are defined by $IA(i) = k$ and $JA(k) = j$. Clearly, the difference $IA(i + 1) - IA(i)$ indicates the total number of nonzero entries in the ith row of M, and the nonzero entries of the ith row of M can be expressed by

$$A(IA(i)), A(IA(i) + 1), \ldots, A(IA(i + 1) - 1),$$

while their column indices by

$$JA(IA(i)), JA(IA(i) + 1), \ldots, JA(IA(i + 1) - 1),$$

where $i = 1, 2, \ldots, n$, and $IA(n + 1) = IA(n) + 1$.

For the natural preconditioner M, there are $n_i + 1$ nonzero block entries in the ith row of the upper triangle part of M, and each block entry is a 3×3 matrix. Hence, for each value of i for $i = 1, 2, \ldots, N$, the corresponding three rows of M have labelling indices $3i - 2, 3i - 1$ and $3i$, and contain $3n_i + 3$, $3n_i + 2$ and $3n_i + 1$ nonzero entries, respectively. Thus, the compressed M by the YSMP compress format can be easily obtained.

4 Formulation of the Sparse Pattern Arrays $\mathcal{C}$ and $\mathcal{P}$

The primary structure of a molecular system is described as a list of pairs (ζ_k, η_k) for $k = 1, 2, \ldots, N_b$ in CHARMM, where ζ_k and η_k denote the labelling indices of the two end atoms of bond k, and N_b the total number of bonds. For a given

bond list, the sparse pattern arrays $\mathcal{C}$ and $\mathcal{P}$ can be generated by using the following scheme:

It is clear that comparing the two atomic indices of each bond with the index of a given atom can identify all the atoms that share one bond with the given atom. Suppose that for $i = 1, 2, \ldots, N$, there exist l_i such atoms with labelling indices $\mu_{i,1}, \mu_{i,2}, \ldots, \mu_{i,l_i}$. Then, these indices can be held by vector $\mathcal{J}$ as below:

$$\mathcal{J}(\mathcal{I}(i) + k - 1) = \mu_{i,k} \quad \text{for} \quad k = 1, 2, \ldots, l_i, \text{ and } i = 1, 2, \ldots, N, \tag{9}$$

where $\mathcal{I}(1) = 1$, and $\mathcal{I}(i) = \sum_{k=1}^{i-1} l_k$ for $i = 2, 3, \ldots, N$. Since $\mathcal{I}(i)$ points to the position of $\mu_{i,1}$ in vector $\mathcal{J}$, l_i can be expressed by

$$l_i = \mathcal{I}(i) - \mathcal{I}(i-1) \quad \text{for } i = 1, 2, \ldots, N.$$

By using $\mathcal{J}$ and $\mathcal{I}$, the sparse pattern arrays $\mathcal{C}$ and $\mathcal{P}$ are formulated in four steps for each given atom, say atom i, where $1 \le i \le N$. In *Step 1*, find l_i atoms with indices $\mu_{i,1}, \mu_{i,2}, \ldots, \mu_{i,l_i}$ that share one bond with atom i from $\mathcal{J}$ and $\mathcal{I}$. If there exists $\mu_{i,j} > i$ with $1 \le j \le l_i$, then $\mu_{i,j}$ is selected as a new entry of $\mathcal{C}$. In *Step 2*, for each selected atom from Step 1, say atom k, find l_k atoms with indices $\mu_{k,1}, \mu_{k,2}, \ldots, \mu_{k,l_k}$ that share one bond with atom k from $\mathcal{J}$ and $\mathcal{I}$. If there exists $\mu_{k,j} > i$ with $1 \le j \le l_k$, then $\mu_{k,j}$ is selected as a new entry of $\mathcal{C}$. In *Step 3*, for each selected atom in Step 2, say atom t, similarly find the atoms that share one bond with atom t and have indices more than i from $\mathcal{J}$ and $\mathcal{I}$. Clearly, such selected atoms connect to atom i through three bonds. Hence, their indices become new entries of $\mathcal{C}$. Finally, in *Step 4*, the indices of atoms satisfying the cutoff condition are selected as new entries of $\mathcal{C}$ if $\eta \neq 0$. To ensure that the indices $\{\nu_{ij}\}$ satisfy (4), in each step of selecting the ith entry of $\mathcal{C}$, the repeated indices between $\mathcal{P}(i)$ and $\mathcal{P}(i+1) - 1$ are excluded.

The above scheme can be implemented in a linear order of N memory locations. In fact, both $\mathcal{I}$ and $\mathcal{P}$ are vectors of size $N + 1$, and the sizes of $\mathcal{J}$ and $\mathcal{C}$ can be found to be $\sum_{i=1}^{N} l_i$ and $\sum_{i=1}^{N} n_i$, respectively. Set $C_{max} = \max\{\max_{1 \le i \le N} n_i, \max_{1 \le i \le N} l_i\}$. Clearly, C_{max} is a constant much less than N. A default value of C_{max} is set as 180 in TNPACK. Hence, the sizes of $\mathcal{J}$ and $\mathcal{C}$ are estimated as $C_{max}N$. Therefore, the total number of integers required to formulate the sparse pattern arrays $\mathcal{C}$ and $\mathcal{P}$ are estimated as

$$2(N + 1 + C_{max}N) \approx 2(C_{max} + 1)N = O(N).$$

5 Switch the CHARMM Format to the YSMP Format

Since TNPACK uses the YSMP compress format to store the preconditioner M, a scheme is needed to switch the CHARMM format $(\mathcal{M}, \mathcal{C}, \mathcal{P})$ to the YSMP format (A, IA, JA). It is easy to obtain IA and JA from $\mathcal{C}$ and $\mathcal{P}$ since the involved numbers $\{\mu_{i,j}\}$ and $\{n_i\}$ can be produced directly from formula (8) when the sparse pattern arrays $\mathcal{C}$ and $\mathcal{P}$ are available.

Clearly, the sparse pattern of M remains the same in the whole minimization process because it is defined according to the bond list. Hence, the formulation of the sparse pattern (IA, JA) or $(\mathcal{C}, \mathcal{P})$ is carried out only at the initial step of TN iterations. After the pattern (IA, JA) is produced, the memory locations of the four integer arrays $\mathcal{I}$, $\mathcal{J}$, $\mathcal{C}$, and $\mathcal{P}$ can be released immediately.

To switch $\mathcal{M}$ to A, two additional pointer arrays, ξ and ζ, are introduced as below. Set their ith components $\xi(i)$ and $\zeta(i)$ to store the positions of $m_{\mu\mu}$ and $m_{\mu\nu}$ in the vector $\mathcal{M}$, respectively, where $\mu = 3i - 2$ and $\nu = 3\nu_{i,\tau} - 2$ for $\tau = 1, 2, \ldots, n_i$, and $i = 1, 2, \ldots, N$. $\xi(i)$ and $\zeta(i)$ can be found as below:

$$\xi(1) = 1, \quad \text{and} \quad \xi(i) = 9n_{i-1} + \xi(i-1) + 6 \quad \text{for} \quad i = 2, 3, \ldots, N, \tag{10}$$

$$\zeta(i) = \xi(i) + 9(\tau - 1) + 6 \quad \text{for} \quad \tau = 1, 2, \ldots, n_i, i = 1, 2, \ldots, N. \tag{11}$$

In terms of ξ and ζ, the nonzero entry array A can be produced directly from $\mathcal{M}$ by the following expressions:

$$A(IA(\mu) + k) = \mathcal{M}(\xi(i) + k), \quad A(IA(\mu + 1)) = \mathcal{M}(\xi(i) + 3),$$

$$A(IA(\mu + 1) + 1) = \mathcal{M}(\xi(i) + 4), \quad A(IA(\mu + 2)) = \mathcal{M}(\xi(i) + 5),$$

$$A(IA(\mu) + 3j + k + 1) = \mathcal{M}(\zeta(i) + k),$$

$$A(IA(\mu + 1) + 3j + k) = \mathcal{M}(\zeta(i) + k + 3),$$

$$A(IA(\mu + 2) + 3j + k - 1) = \mathcal{M}(\zeta(i) + k + 6),$$

where $k = 0, 1, 2$, $\mu = 3i - 2$, $j = 1, 2, \ldots, n_i$, and $i = 1, 2, \ldots, N$.

Both $\mathcal{M}$ and A have the same size, which can be found by $6N + 9\sum_{i=1}^{N} n_i$, and estimated by

$$(9 \max_{1 \le i \le N} n_i + 6)N \le 9C_{max}N = O(N).$$

Hence, the formulation of A from $\mathcal{M}$ requires only a linear order of N memory locations.

References

1. Brooks, B. R., Bruccoleri, R. E., Olafson, B. D., States, D. J., Swaminathan, S. Karplus, M.: CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. J. Comp. Chem., **4** (1983) 187-217.
2. Dembo, R. S., Steihaug, T.: Truncated-Newton algorithms for large-scale unconstrained optimization. Math. Prog., **26** (1983) 190-212.
3. Derreumaux, P., Zhang, G., Brooks, B., Schlick, T.: A truncated-Newton method adapted for CHARMM and biomolecular applications. J. Comp. Chem., **15** (1994) 532-552.
4. Perahia, D., Mouawad, L.: Computation of low-frequency normal modes in macromolecules: improvements to the method of diagonalization in a mixed basis and application to hemoglobin. Computers & Chemistry, **19** (1995) 241-245.

5. Schultz, M. H., Eisenstat, S. C., Sherman, A. H.: Algorithms and data structures for sparse symmetric Gaussian elimination. SIAM J. Sci. Statist. Comput., **2** (1981) 225-237.
6. Schlick, T., Fogelson, A.: TNPACK — A truncated Newton minimization package for large-scale problems: I. Algorithm and usage. ACM Trans. Math. Softw., **14** (1992) 46-70.
7. Schlick, T., Overton, M. L.: A powerful truncated Newton method for potential energy functions. J. Comp. Chem., **8** (1987) 1025-1039.
8. Xie, D., Schlick, T.: Efficient implementation of the truncated-Newton algorithm for large-scale chemistry applications. SIAM J. OPT., **10** (1999) 132-154.
9. Xie, D., Schlick, T.: Remark on Algorithm 702—The updated truncated Newton minimization package. ACM Trans. on Math. Software, **25** (1999) 108-122.
10. Xie, D., Schlick, T.: A more lenient stopping rule for line search algorithms. Optimization Methods and Software, **17** (2002) 683-700.