Journal of Computational Physics 349 (2017) 253–264

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Distance descending ordering method: An $O(n)$ algorithm for inverting the mass matrix in simulation of macromolecules with long branches

Xiankun Xu, Peiwen Li *

Department of Aerospace and Mechanical Engineering, The University of Arizona, Tucson, AZ 85721, USA

ARTICLE INFO

Article history:
Received 1 June 2017
Received in revised form 2 August 2017
Accepted 4 August 2017
Available online 18 August 2017

Keywords:
Molecular dynamics
Internal coordinates
Mass matrix
$O(n)$ time complexity
Sparse matrix
Cholesky decomposition

ABSTRACT

Fixman's work in 1974 and the follow-up studies have developed a method that can factorize the inverse of mass matrix into an arithmetic combination of three sparse matrices—one of them is positive definite and needs to be further factorized by using the Cholesky decomposition or similar methods. When the molecule subjected to study is of serial chain structure, this method can achieve $O(n)$ time complexity. However, for molecules with long branches, Cholesky decomposition about the corresponding positive definite matrix will introduce massive fill-in due to its nonzero structure. Although there are several methods can be used to reduce the number of fill-in, none of them could strictly guarantee for zero fill-in for all molecules according to our test, and thus cannot obtain $O(n)$ time complexity by using these traditional methods. In this paper we present a new method that can guarantee for no fill-in in doing the Cholesky decomposition, which was developed based on the correlations between the mass matrix and the geometrical structure of molecules. As a result, the inverting of mass matrix will remain the $O(n)$ time complexity, no matter the molecule structure has long branches or not.

1. Introduction

Molecular dynamics simulation involves the calculation of the following equation of motion: $\dot{\mathbf{q}} = \boldsymbol{\mathcal{M}}^{-1}\mathbf{p}$. Where $\mathbf{q}$ is the generalized coordinate, $\mathbf{p}$ is the conjugate momenta, and $\boldsymbol{\mathcal{M}}$ is the mass matrix of the molecule. In Cartesian coordinate the mass matrix is diagonal and is easy to be inverted. For molecule models with geometrical constraints such as fixed angles and lengths, consideration of the molecular movements in the space of generalized internal coordinate rather than Cartesian coordinate is an efficient way to deal with the constraints. In this instance the fixed internal variables are directly excluded from the generalized coordinate, $\mathbf{q}$, and the calculation of constraint forces [1–3] can be avoided. However, in this case the corresponding mass matrix is a dense matrix. For macromolecules such as DNA, RNA, proteins, polymers, etc., the inverting of mass matrix is a difficult job, the direct calculation of $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{M}}^{-1}$ scaled as $O(n^3)$, where n is the number of atoms in the molecule.

In 1974, Fixman [4] developed a method for efficient calculation of the determinant of mass matrix for polymer chains. Lee et al. [5–7] extended this method to solve the equations of motion in $O(n)$ calculation. They applied Fixman's theorem to invert the mass matrix for structures of serial chain such as planar N-link manipulator, polymer chains, polypeptide

* Corresponding author.
E-mail address: peiwen@email.arizona.edu (P. Li).

http://dx.doi.org/10.1016/j.jcp.2017.08.006
0021-9991/

chain, and serial chain composed of rigid bodies. They found that the mass matrix inverse can be factorized as $\mathcal{M}^{-1} = \mathbf{A} - \mathbf{B}\mathbf{C}^{-1}\mathbf{B}^{\top}$, where $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are sparse matrices, the number of nonzero entries in the three matrices are all of $O(n)$, also, $\mathbf{C}$ is positive definite. The solving process of the equation of motion can then be decomposed into a series of calculations involving the three sparse matrices. However, a Cholesky decomposition $\mathbf{C} = \mathbf{L}\mathbf{L}^{\top}$ is required due to the existence of $\mathbf{C}^{-1}$. If the molecule has serial chain structure, the nonzero entries of $\mathbf{C}$ will be clustered about its diagonal. In this case the calculation of Cholesky decomposition can be of order $O(n)$. The equation of motion can therefore be solved in an $O(n)$ calculation.

While the above approaches are confined to serial chain structures, in this case the diagonal clustering property of $\mathbf{C}$ makes the Cholesky decomposition easy and efficient. However, if the molecule structures have long branches, the nonzero entries will no longer cluster to the diagonal, even though $\mathbf{C}$ is still sparse. As a result, direct Cholesky decomposition from $\mathbf{C}$ to $\mathbf{L}$ will introduce a lot of fill-in. This means that $\mathbf{L}$ is no longer sparse.

There are two major categories of numerical methods in dealing with sparse positive definite matrices problem: iterative and direct methods. Iterative methods [8,9] start with an initial approximation of the solution, and the iteration continues until the desired accuracy being achieved. Typically the number of iterations grows with the size of the matrix. On the other hand, direct methods [10] obtain the solution in a finite and fixed number of steps. Generally, iterative methods give approximate solutions, while direct methods give solutions up to the machine precision. A typical direct method often associates with finding a permutation matrix $\mathbf{P}$, and applying the Cholesky decomposition on the permuted $\mathbf{P}\mathbf{C}\mathbf{P}^{\top}$, the target is to make the corresponding $\mathbf{L}$ being as sparse as possible. Since finding a permutation that results in the minimum fill-in is an NP-hard problem [11], people proposed several heuristic methods to find permutations that give low number of fill-in. Common methods in this category include minimum degree method [12–14], approximate minimum method (an approximate derivation of minimum degree method) [15], and nested dissection method [16], etc. Some of these heuristic methods are very efficient in reducing the nonzero fills; however, according to our test, none of them can guarantee no fill-in for all types of molecule structure. Time complexity evaluation of a heuristic method should take into account of both finding permutation $\mathbf{P}$ and performing matrix decomposition, which is also depending on the nonzero structure of the sparse matrix. Therefore, the time complexities of the above-mentioned heuristic methods are generally higher than $O(n)$ [17,18]. It is necessary to develop a new method to make efficient simulation of macromolecules with long branches.

The physical aspect of the nonzero structure of $\mathbf{C}$ shows that there are correlations between the nonzero entries and the geometrical structure of the molecule. Based on this observation we proposed a new method to efficiently factorize $\mathbf{C}$, which is especially useful for long branch structures. This method is named as *distance descending ordering method*, for it is strongly associated with the distances between atoms. It is not a heuristic method, it directly gives an $O(n)$ permutation strategy about the matrix $\mathbf{C}$ based on the structure of the molecule. This method has the following features: 1) absolutely no fill-in in doing Cholesky decomposition; 2) has $O(n)$ magnitude of calculation; 3) suitable for molecules with complex structures such as branches, loops, etc., and also, constraints can be arbitrarily applied on angles and lengths; 4) simple, can easily be implemented with computer codes.

Although we will not discuss the detailed treatment of loop structure in this paper, a macromolecule in general contains not only branches, but also loop structures. The loop can be broken by simply cutting apart one bond inside the loop, and then the loop can be treated as two branches. By doing so we need to deal with the potentials associated with the broken bond. This will only have influence on the calculation of intramolecular forces, and the procedure of inverting mass matrix described in this paper will not be affected. It is expected, the distance descending ordering method can be applied to molecules with complex structures having both branches and loops.

In section 2, we will make a brief introduction about the Fixman's theorem and elaborate the difficulties in applying it on molecules with long branches. In section 3, the distance descending ordering method will be presented with detailed proof. The mechanism why this method has no fill-in in doing the Cholesky decomposition will be explained. In section 4, we apply the method on molecules with 100, 1000 and 10000 atoms to show the no fill-in property. In addition, the $O(n)$ time complexity is demonstrated by giving the average calculation time versus the molecule's number of atoms. Finally, in Appendix A we provide a complement to existing methods [19] about calculation of the relative change of internal coordinates with respect to the atoms' position vectors, which is used to calculate the entries of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$.

2. Fixman's theorem

For a molecule composed by $n+1$ atoms, the atoms are numbered from 0 to n. Where the 0-th atom is the base atom which we can choose arbitrarily. The position information of this molecule can either be represented by using the Cartesian coordinate as $\mathbf{r} = \{\vec{r}_0, \vec{r}_1, \cdots, \vec{r}_n\}$, or by the generalized internal coordinate as $\mathbf{g} = \{\vec{g}_0, \vec{g}_1, \cdots, \vec{g}_n\}$, where $\vec{r}_i = \{x_i, y_i, z_i\}$, $\vec{g}_0 = \{x_0, y_0, z_0\}$, $\vec{g}_i = \{\phi_i, \theta_i, b_i\}$, and ϕ_i, θ_i and b_i are the torsional angle, bond angle and bond length corresponding to the i-th atom, respectively. Since some internal coordinate variables are constrained as constants, the internal coordinate vector $\mathbf{g}$ is partitioned into two parts as $\mathbf{g} = \{\mathbf{q}; \mathbf{c}\}$: the soft variables $\mathbf{q} = \{q_1, q_2, \cdots, q_f\}$ and the hard variables $\mathbf{c} = \{c_1, c_2, \cdots, c_r\}$, where f is the degrees of freedom of the molecule, r is the number of constraints, and $f + r = 3(n+1)$. Generally f and r are of the same magnitude as n for real molecules, thus there is $O(n) = O(f) = O(r)$.

Since $\mathbf{r}$ and $\mathbf{g}$ are of the same length, and also the variables in $\mathbf{r}$ are mutually independent with each other (also applicable to $\mathbf{g}$), therefore, the two square Jacobians $\frac{\partial \mathbf{r}}{\partial \mathbf{g}}$ and $\frac{\partial \mathbf{g}}{\partial \mathbf{r}}$ are inverse of each other:

$$\left(\frac{\partial \mathbf{r}}{\partial \mathbf{g}}\right)^{-1} = \frac{\partial \mathbf{g}}{\partial \mathbf{r}} \tag{1}$$

where

$$\begin{aligned}
\frac{\partial \mathbf{r}}{\partial \mathbf{g}} &= \left[\begin{array}{c|c} \frac{\partial \mathbf{r}}{\partial \mathbf{q}} & \frac{\partial \mathbf{r}}{\partial \mathbf{c}} \end{array}\right] = \left[\begin{array}{ccc|ccc} \left[\frac{\partial \vec{r}_0}{\partial q_1}\right] & \left[\frac{\partial \vec{r}_0}{\partial q_2}\right] & \left[\frac{\partial \vec{r}_0}{\partial q_f}\right] & \left[\frac{\partial \vec{r}_0}{\partial c_1}\right] & \left[\frac{\partial \vec{r}_0}{\partial c_2}\right] & \left[\frac{\partial \vec{r}_0}{\partial c_r}\right] \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \left[\frac{\partial \vec{r}_n}{\partial q_1}\right] & \left[\frac{\partial \vec{r}_n}{\partial q_2}\right] & \left[\frac{\partial \vec{r}_n}{\partial q_f}\right] & \left[\frac{\partial \vec{r}_n}{\partial c_1}\right] & \left[\frac{\partial \vec{r}_n}{\partial c_2}\right] & \left[\frac{\partial \vec{r}_n}{\partial c_r}\right] \end{array}\right] \\
\frac{\partial \mathbf{g}}{\partial \mathbf{r}} &= \left[\begin{array}{c} \frac{\partial \mathbf{q}}{\partial \mathbf{r}} \\ \frac{\partial \mathbf{c}}{\partial \mathbf{r}} \end{array}\right] = \left[\begin{array}{ccc} \left[\partial q_1/\partial \vec{r}_0\right] & \left[\partial q_1/\partial \vec{r}_1\right] & \left[\partial q_1/\partial \vec{r}_n\right] \\ \vdots & \vdots & \vdots \\ \left[\partial q_f/\partial \vec{r}_0\right] & \left[\partial q_f/\partial \vec{r}_1\right] & \left[\partial q_f/\partial \vec{r}_n\right] \\ \hline \left[\partial c_1/\partial \vec{r}_0\right] & \left[\partial c_1/\partial \vec{r}_1\right] & \left[\partial c_1/\partial \vec{r}_n\right] \\ \vdots & \vdots & \vdots \\ \left[\partial c_r/\partial \vec{r}_0\right] & \left[\partial c_r/\partial \vec{r}_1\right] & \left[\partial c_r/\partial \vec{r}_n\right] \end{array}\right]
\end{aligned} \tag{2}$$

In the above equations, $\frac{\partial \mathbf{r}}{\partial \mathbf{g}}$ is a dense matrix, while $\frac{\partial \mathbf{g}}{\partial \mathbf{r}}$ is a very sparse matrix. It can be seen in Eq. (A.4) in appendix that ϕ_i, θ_i and b_i are only affected by 4, 3 and 2 atoms, respectively, and thus most of the entries in $\frac{\partial \mathbf{g}}{\partial \mathbf{r}}$ are zero. We want to use Eq. (1) together with the sparse property of $\frac{\partial \mathbf{g}}{\partial \mathbf{r}}$ to efficiently invert the mass matrix.

Define matrix $\mathbf{G}$ as follows:

$$\begin{aligned}
\mathbf{G} &= \left(\frac{\partial \mathbf{r}}{\partial \mathbf{g}}\right)^{\top} \mathbf{D} \left(\frac{\partial \mathbf{r}}{\partial \mathbf{g}}\right) \\
&= \left[\frac{(\partial \mathbf{r}/\partial \mathbf{q})^{\top}}{(\partial \mathbf{r}/\partial \mathbf{c})^{\top}}\right] \begin{bmatrix} m_0 \mathbf{I}_3 & & \\ & \ddots & \\ & & m_n \mathbf{I}_3 \end{bmatrix} \left[\begin{array}{c|c} \frac{\partial \mathbf{r}}{\partial \mathbf{q}} & \frac{\partial \mathbf{r}}{\partial \mathbf{c}} \end{array}\right] \\
&= \left[\begin{array}{c|c} \left(\frac{\partial \mathbf{r}}{\partial \mathbf{q}}\right)^{\top} \mathbf{D} \left(\frac{\partial \mathbf{r}}{\partial \mathbf{q}}\right) & \left(\frac{\partial \mathbf{r}}{\partial \mathbf{q}}\right)^{\top} \mathbf{D} \left(\frac{\partial \mathbf{r}}{\partial \mathbf{c}}\right) \\ \hline \left(\frac{\partial \mathbf{r}}{\partial \mathbf{c}}\right)^{\top} \mathbf{D} \left(\frac{\partial \mathbf{r}}{\partial \mathbf{q}}\right) & \left(\frac{\partial \mathbf{r}}{\partial \mathbf{c}}\right)^{\top} \mathbf{D} \left(\frac{\partial \mathbf{r}}{\partial \mathbf{c}}\right) \end{array}\right]
\end{aligned} \tag{3}$$

The mass matrix $\mathcal{M} = \left(\frac{\partial \mathbf{r}}{\partial \mathbf{q}}\right)^{\top} \mathbf{D} \left(\frac{\partial \mathbf{r}}{\partial \mathbf{q}}\right)$ is the first block of $\mathbf{G}$. From Eq. (1), we have

$$\begin{aligned}
\mathbf{G}^{-1} &= \left(\frac{\partial \mathbf{r}}{\partial \mathbf{g}}\right)^{-1} \mathbf{D}^{-1} \left(\frac{\partial \mathbf{r}}{\partial \mathbf{g}}\right)^{-\top} = \left(\frac{\partial \mathbf{g}}{\partial \mathbf{r}}\right) \mathbf{D}^{-1} \left(\frac{\partial \mathbf{g}}{\partial \mathbf{r}}\right)^{\top} \\
&= \left[\begin{array}{c|c} \left(\frac{\partial \mathbf{q}}{\partial \mathbf{r}}\right) \mathbf{D}^{-1} \left(\frac{\partial \mathbf{q}}{\partial \mathbf{r}}\right)^{\top} & \left(\frac{\partial \mathbf{q}}{\partial \mathbf{r}}\right) \mathbf{D}^{-1} \left(\frac{\partial \mathbf{c}}{\partial \mathbf{r}}\right)^{\top} \\ \hline \left(\frac{\partial \mathbf{c}}{\partial \mathbf{r}}\right) \mathbf{D}^{-1} \left(\frac{\partial \mathbf{q}}{\partial \mathbf{r}}\right)^{\top} & \left(\frac{\partial \mathbf{c}}{\partial \mathbf{r}}\right) \mathbf{D}^{-1} \left(\frac{\partial \mathbf{c}}{\partial \mathbf{r}}\right)^{\top} \end{array}\right] = \left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{B}^{\top} & \mathbf{C} \end{array}\right]
\end{aligned} \tag{4}$$

By comparing $\mathbf{G}$ and $\mathbf{G}^{-1}$ [5–7], we have:

$$\mathcal{M}^{-1} = \mathbf{A} - \mathbf{B}\mathbf{C}^{-1}\mathbf{B}^{\top} \tag{5}$$

The favorable feature is that $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are all sparse matrices, the number of nonzero entries in these three matrices are all of order $O(n)$. Also, both $\mathbf{A}(f \text{ by } f)$ and $\mathbf{C}(r \text{ by } r)$ are positive definite matrices.

The inverting of mass matrix problem $\dot{\mathbf{q}} = \mathcal{M}^{-1}\mathbf{p}$ can then be solved as:

$$\begin{cases}
\mathbf{y}_1 = \mathbf{A}\mathbf{p} & \text{(a)} \\
\mathbf{y}_2 = \mathbf{B}^{\top}\mathbf{p} & \text{(b)} \\
\mathbf{y}_3 = \mathbf{C}^{-1}\mathbf{y}_2 & \text{(c)} \\
\mathbf{y}_4 = \mathbf{B}\mathbf{y}_3 & \text{(d)} \\
\dot{\mathbf{q}} = \mathbf{y}_1 - \mathbf{y}_4 & \text{(e)}
\end{cases} \tag{6}$$

where calculation of Eq. (6a) can be further decomposed as $\mathbf{y}_1 = \frac{\partial \mathbf{q}}{\partial \mathbf{r}} \cdot \left\{\mathbf{D}^{-1} \cdot \left[(\partial \mathbf{q}/\partial \mathbf{r})^{\top} \cdot \mathbf{p}\right]\right\}$, and the calculation of Eqs. (6b), (6d) is similar. Because the number of nonzero entries in $\mathbf{A}$ and $\mathbf{B}$ are of the order of $O(n)$, the complexity of calculation of these three equations will also have the order of $O(n)$.

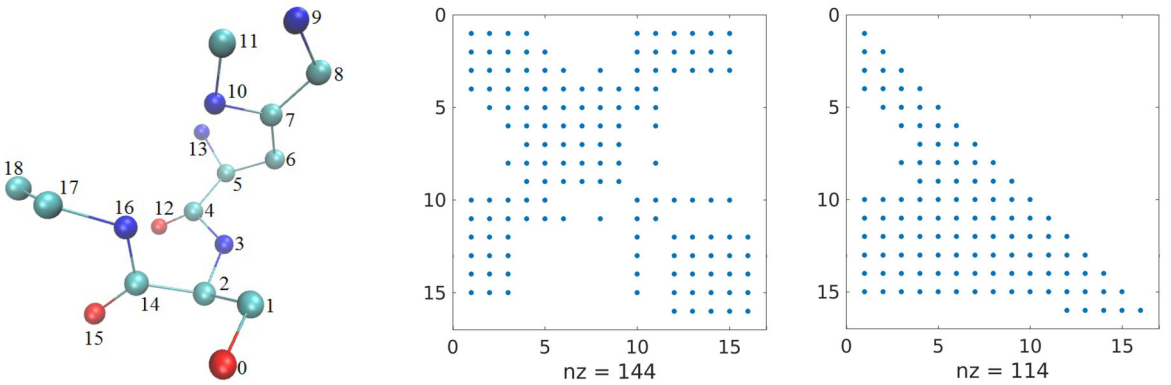

Fig. 1. A numbered molecule (left), the nonzero structures of **C** (middle) and **L** (right).

Since **C** is positive definite, we can use Cholesky decomposition to factorize it as $\mathbf{C} = \mathbf{L}\mathbf{L}^\top$, and the solving of Eq. (6c) becomes $\mathbf{L}\left(\mathbf{L}^\top \mathbf{y}_3\right) = \mathbf{y}_2$ which is backward stable. Here the critical issue is that even though **C** is a sparse matrix, the corresponding **L** may not necessarily be. If a molecule has long branches, the Cholesky decomposition from **C** to **L** may introduce fill-in, namely, **L** has nonzero entries in the positions which are zero in **C**. Take the molecule shown in Fig. 1 for demonstration, the hard variables are set as $\mathbf{c} = \{\phi_3, \phi_4, \cdots, \phi_{18}\}$, the nonzero structures of the corresponding **C** and **L** are also shown. It could be found that **L** introduces some fill-in when compared to **C**. As a result, the computation of the Cholesky decomposition will not retain to $O(r)$ time complexity, where $O(r) = O(n)$.

3. Descending distance ordering method

The most optimum condition is that no fill-in will be introduced in the Cholesky decomposition, in this situation **L** has the same nonzero structure as that of the lower diagonal part of **C**. In this section, the *distance descending ordering method* will be proposed, it can strictly guarantee no fill-in, and thus gives the $O(r)$ time complexity. Unlike most of the direct methods in dealing with sparse matrices, this method does not need to explicitly do row and column permutations to find **P**, instead it directly gives an ordering strategy about the hard variable list **c**, based on the structure of the molecule. It is especially powerful for molecules with long branches. In addition, geometrical constraints can be arbitrarily applied. This method is also very simple, making it easy to be integrated into computer codes.

In the following, we start the discussion with 2 definitions:

Correlation: For the matrix **C** defined in Eq. (4), if its ij-th entry is nonzero, we call that the two hard variables c_i and c_j are correlated, or c_i has correlation with c_j, where c_i and c_j are the i-th and j-th elements in hard variable vector **c**.

Distance: The distance of the i-th atom is defined as the number of bonds between itself and the 0-th atom, which has a unique value if there is no loops.

The following three points about the structure of $\mathbf{C}, \mathbf{L}$ and the permutation operation should be noted:

1. The number of nonzero entries in matrix **C** is of order $O(r)$, since each hard variable c_i only correlates with very few number of other hard variables. In other words, the nonzero entries in each line or column is finite and is the order of $O(10)$. This is because $\mathbf{C}_{ij} = \sum_{k=0}^{n} \frac{1}{m_k} \frac{\partial c_i}{\partial \vec{r}_k} \cdot \frac{\partial c_j}{\partial \vec{r}_k} \neq 0$ requires both c_i and c_j being affected by at least one same $\vec{r}_k$, and from Eq. (A.4) we see that the internal variables are only related to atoms that are close or nearby.
2. If no fill-in, the number of nonzero entries in matrix **L** is of the order of $O(r)$. Also, the time complexity in applying Cholesky decomposition and calculation of $\mathbf{L}\left(\mathbf{L}^\top \mathbf{y}_3\right) = \mathbf{y}_2$ are both of the order of $O(r)$.
3. A permutation on matrix **C** is actually a permutation on the hard variable list **c**, or **P** applied a reordering operation on vector **c**, which could be seen as follows:

$$\begin{aligned} \mathbf{P}\mathbf{C}\mathbf{P}^\top &= \mathbf{P}\left(\frac{\partial \mathbf{c}}{\partial \mathbf{r}}\right)\mathbf{M}^{-1}\left(\frac{\partial \mathbf{c}}{\partial \mathbf{r}}\right)^\top \mathbf{P}^\top = \left(\frac{\partial \mathbf{P}\mathbf{c}}{\partial \mathbf{r}}\right)\mathbf{M}^{-1}\left(\frac{\partial \mathbf{P}\mathbf{c}}{\partial \mathbf{r}}\right)^\top \\ &= \left(\frac{\partial \mathbf{c}'}{\partial \mathbf{r}}\right)\mathbf{M}^{-1}\left(\frac{\partial \mathbf{c}'}{\partial \mathbf{r}}\right)^\top = \mathbf{L}\mathbf{L}^\top \end{aligned} \tag{7}$$

where $\mathbf{c}' = \mathbf{P}\mathbf{c}$ is the reordered hard variable vector. Instead of directly permuting matrix **C**, we are going to find an ordering strategy about hard variable vector **c**, to make sure that there is no fill-in in the procedure of Cholesky decomposition.

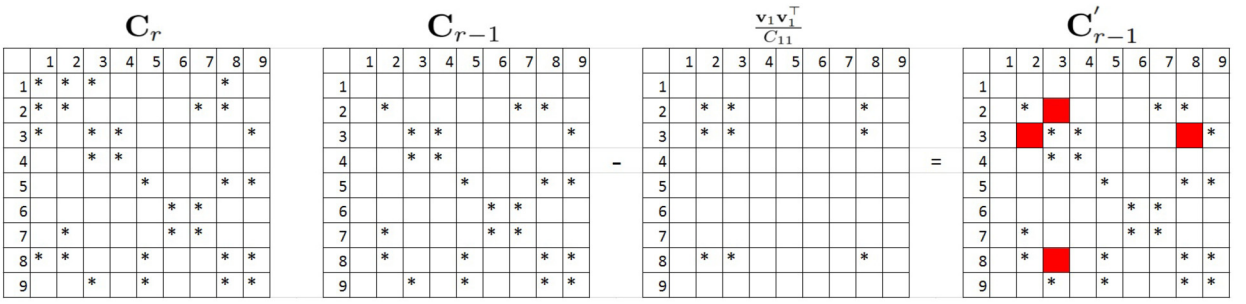

Fig. 2. Correlation determines fill-in, red boxes in $\mathbf{C}'_{r-1}$ indicate fill-in introduced in a basic factorization step. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Consider a basic factorization step of Cholesky decomposition:

$$\begin{aligned}\mathbf{C}_r &= \begin{pmatrix} C_{11} & \mathbf{v}_1^\top \\ \mathbf{v}_1 & \mathbf{C}_{r-1} \end{pmatrix} = \begin{pmatrix} \sqrt{C_{11}} & \mathbf{0} \\ \frac{\mathbf{v}_1}{\sqrt{C_{11}}} & \mathbf{I}_{r-1} \end{pmatrix} \begin{pmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{r-1} - \frac{\mathbf{v}_1\mathbf{v}_1^\top}{C_{11}} \end{pmatrix} \begin{pmatrix} \sqrt{C_{11}} & \frac{\mathbf{v}_1^\top}{\sqrt{C_{11}}} \\ \mathbf{0} & \mathbf{I}_{r-1} \end{pmatrix} \\ &= \mathbf{L}_1 \begin{pmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{r-1} - \frac{\mathbf{v}_1\mathbf{v}_1^\top}{C_{11}} \end{pmatrix} \mathbf{L}_1^\top \end{aligned} \tag{8}$$

Set $\mathbf{C}'_{r-1} = \mathbf{C}_{r-1} - \mathbf{v}_1\mathbf{v}_1^\top/\mathbf{C}_{11}$, $\mathbf{C}'_{r-1}$ is positive definite and the above basic step is recursively applied. The no fill-in requires that $\mathbf{C}'_{r-1}$ has the same nonzero structure as $\mathbf{C}_{r-1}$. Three observations can be made about the factorization step:

1. For each nonzero entry of $\mathbf{v}_1\mathbf{v}_1^\top/\mathbf{C}_{11}$, if the corresponding entry of $\mathbf{C}_{r-1}$ is also nonzero, then $\mathbf{C}'_{r-1}$ will have the same nonzero structure as $\mathbf{C}_{r-1}$.
2. The ij-th entry in $\mathbf{v}_1\mathbf{v}_1^\top/\mathbf{C}_{11}$ is nonzero, which requires the two hard variables c_i and c_j being both correlated with the eliminating hard variable c_1.
3. The ij-th entry in $\mathbf{C}$ is nonzero, which implies that the two hard variables c_i and c_j are correlated.

Therefore, in order to guarantee for no fill-in, the hard variables which have correlation with the eliminating hard variable should be mutually correlated with each other.

This statement can be demonstrated by Fig. 2, where hard variables c_2, c_3 and c_8 are correlated with c_1. The requirement of no fill-in demands that c_2, c_3 and c_8 should be mutually correlated with each other. However, c_3 is not correlated with c_2 and c_8, nonzero elements are introduced in the $(2,3)$, $(3,8)$ entries as well as their symmetric entries.

Thus the correlation between hard variables plays a critical role in determining if fill-in would happen or not. Three points may be noted about the correlation between hard variables:

1. The correlation between c_i and c_j implies that these two hard variables are affected by at least one identical atom.
2. Different types of hard variables are affected by different number of atoms. As shown in Eq. (A.4), ϕ_i, θ_i and b_i are affected by 4, 3 and 2 atoms, respectively.
3. Hard variables in different branches may also have correlation. For the molecule shown in Fig. 1, ϕ_4 and ϕ_{16} are correlated since both are affected by the 1st and 2nd atoms. And this is the reason long branches may introduce fill-in.

Point 2 suggested that the correlation depends on the types of the hard variables. To make the discussion clear, in Sec. 3.1 we will propose the distance descending ordering method for the case that only torsional angles are allowed to be hard variable. In this case each hard variable is affected by 4 atoms, making the discussion be simplified. In Sec. 3.2, the method is extended to allow all internal variables being arbitrarily constrained. Indeed, the first case is a special situation of the second one.

3.1. Only torsional angles are allowed to be hard variable

Theorem 1. *In the case that only torsional angles are allowed to be constrained, if the hard variables in vector* $\mathbf{c}$ *are arranged in the descending order according to the distances to the* 0*-th atom, then the Cholesky decomposition on matrix* $\mathbf{C}$ *will have no fill-in.*

Proof.

1. If c_1 is the eliminating hard variable and it has the largest distance to 0-th atom, then c_1 will only correlate with hard variables which have less or equal distances. The distance differences between c_1 and these hard variables are less than or equal to 4 (a limit on the difference of distance if two torsional angles need to be correlated). As a result,

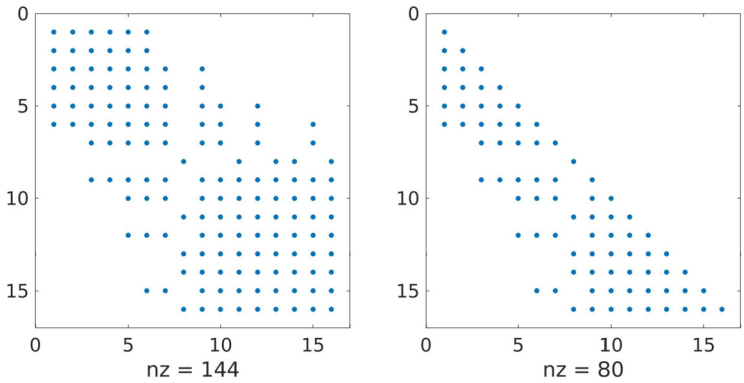

Fig. 3. Distance descending ordering method has no fill-in: **C** (left); **L** (right).

the distance differences between these hard variables are also less than or equal to 4, and all these hard variables will mutually correlate with each other. Thus the elimination of c_1 guarantees no fill-in.

2. After the elimination of c_1, its corresponding atom can be eliminated from the molecule structure, and its correlations with the remaining hard variables will no longer have influence on the following basic factorization steps. The largest distance torsional angles among the remaining are now considered as the next eliminating hard variables. The above logic is again applied, this guarantees that each basic factorization step has no fill-in. □

Theorem 1 can be best elaborated by using the molecule shown in Fig. 1 as an example. Suppose all torsional angles are constrained except ϕ_1 and ϕ_2, then the hard variable vector **c** should be ordered as $\mathbf{c} = \{\phi_9, \phi_{11}, \phi_8, \phi_{10}, \phi_7, \phi_6, \phi_{13}, \phi_{18}, \phi_5, \phi_{12}, \phi_{17}, \phi_4, \phi_{15}, \phi_{16}, \phi_3, \phi_{14}\}$. The corresponding **C** and **L** are shown in Fig. 3, it is seen that there is no fill-in introduced.

It is noted that since the values of distances are consecutive integers, the ordering of the distances takes $O(n)$ computation. As has been proven in the beginning of Sec. 3, if there is no fill-in, the time complexities of Cholesky decomposition of **C** and calculation of $\mathbf{L}\left(\mathbf{L}^{\top}\mathbf{y}\right) = \mathbf{x}$ should be both of the order of $O(n)$. Thus, the procedures of finding ordering and matrix inverting are both of linear time complexity, which indicates an $O(n)$ algorithm of distance descending ordering method.

3.2. Any internal coordinate variable can be constrained

In the case of arbitrarily constrained internal variables, because different types of internal coordinate variables are influenced by different number of atoms, the correlation between hard variables varies according to their types. As a result, simply ordering the hard variables in the descending order of distances will introduce nonzero fill-in, since it can not guarantee that the hard variables which have correlations with the eliminating hard variable are mutually correlated with each other.

In fact, the descending ordering strategy should still be applied, but with a slight modification, which can thus avoid the introducing of fill-in:

Theorem 2. *In the case that any internal variables can be considered as hard variable, the hard variables should first be ordered within the distance level l (collection of atoms have distance l to the base atom) as:* $[b_1^{(l)}, b_2^{(l)}, \cdots, b_{n_l}^{(l)}; \phi_1^{(l+1)}, \phi_2^{(l+1)}, \cdots, \phi_{n_{l+1}}^{(l+1)}; \theta_1^{(l)}, \theta_2^{(l)}, \cdots, \theta_{n_l}^{(l)}]$. *And then the distance levels are still arranged in a descending order. The corresponding Cholesky decomposition on matrix* **C** *will have no fill-in.*

The terms n_l and n_{l+1} are the number of atoms having distances l and $l+1$ to the base atom, the superscripts (l) and $(l+1)$ denote that these hard variables belong to atoms that have distances l and $l+1$, respectively. It is the rule that if any internal variable is non-constant, it should be excluded from the hard variable vector **c**.

Again, take the molecule shown in Fig. 1 as an example, the hard variables should be ordered as follows: $\mathbf{c} = \{[b_9, b_{11}; \theta_9, \theta_{11}],\ [b_8, b_{10}; \phi_9, \phi_{11}; \theta_8, \theta_{10}],\ \cdots,\ [b_2; \phi_3, \phi_{14}; \theta_2], [b_1]\}$, where x_0, y_0, z_0 and ϕ_1, θ_1, ϕ_2 are excluded from **c** because they represent the translational and rotational movements of the molecule and must be non-constant. Also, non-constant hard variables should be removed from this list.

This ordering strategy takes into account the fact that b_i, ϕ_i and θ_i are affected by different number of atoms. It can be verified that in applying this ordering method, hard variables which correlate with the eliminating hard variable are also mutually correlated with each other. As a result, the Cholesky decomposition will have no fill-in.

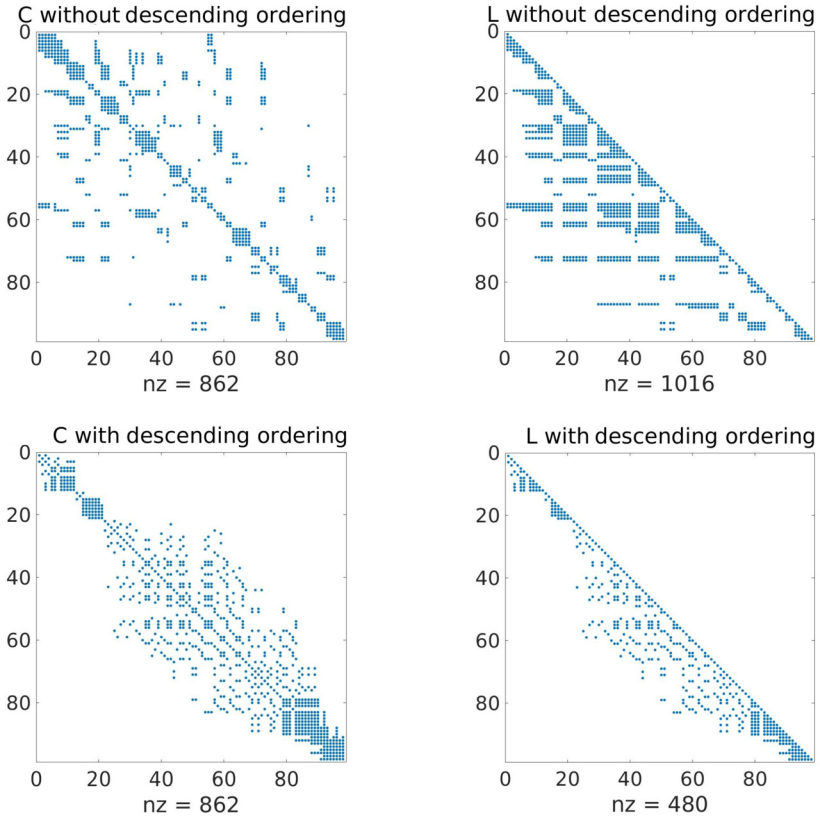

Fig. 4. Nonzero structures of **C** and **L** from algorithms with and without descending ordering, molecule with $n = 100$, $r = 98$.

4. Numerical validations

In section 3, we have proven that if no fill-in in Cholesky decomposition about matrix **C**, then the calculation complexity of Eq. (6c) will be of the order of $O(r)$. Correspondingly, the solving of the equation $\dot{\mathbf{q}} = \boldsymbol{\mathcal{M}}^{-1}\mathbf{p}$ will obtain the $O(n)$ time complexity.

In this section, three "artificial" molecules with number of atoms $n = 100, 1000$ and 10000 are used to show the no fill-in effectiveness of the distance descending ordering method. For comparison, the results calculated using direct Cholesky decomposition are also listed, in which the ordering of hard variables are made according to the numbering of atoms. Since the sparsity of matrix **C** is mainly affected by two factors: the type of the hard variables and the number of branches. To make the comparison be consistent, the hard variables are chosen by randomly fixing one third of ϕ_i, one third of θ_i, and one third of b_i for all the three molecules. Also the ratio of number of branches to number of atoms is fixed as 0.25. The nonzero structures of all the **C** and **L** are shown in Figs. 4, 5, 6. Three conclusions can be made by comparing the results:

1. For **C** and **L** obtained from the distance descending ordering method, the nonzero structure of **L** is exactly the same as that of the lower diagonal part of **C**. The number of nonzero entries in lower diagonal part of **C** is $(nz_{\mathbf{C}} - r)/2 + r$, which equals to $nz_{\mathbf{L}}$ for all the three cases. Where $nz_{\mathbf{C}}$ and $nz_{\mathbf{L}}$ are numbers of nonzero entries in **C** and **L**.
2. The numbers of nonzero entries are the same for both the two matrices **C** obtained from algorithms with and without the descending ordering, a permutation can convert one to another.
3. With the application of the distance descending ordering method, the number of nonzero entries in **L** is roughly linearly scaled with the number of atoms: $nz_{\mathbf{L}} = 480$ for $n = 100$, $nz_{\mathbf{L}} = 4960$ for $n = 1000$, and $nz_{\mathbf{L}} = 50775$ for $n = 10000$. This implies that the solving of the equation $\dot{\mathbf{q}} = \boldsymbol{\mathcal{M}}^{-1}\mathbf{p}$ will also have this linear scaling. On the other hand, without the descending ordering, the number of nonzero entries in **L** grows with a power function in the manner of $n^{(1<x\leq 3)}$.

The linear time complexity can also be demonstrated by the average calculation time versus the number of atoms. The number of atoms in a molecule is varied with values $n = 1000, 10000, 30000, 50000, 80000$ and 100000 to show the $O(n)$ performance. Furthermore, since the type of hard variables have influence on the denseness of **C**, 4 strategies in choosing the hard variables are used to study this influence. These 4 constraint strategies are:

1. All torsional angles except ϕ_1 and ϕ_2 are chosen as hard variables, $r = n - 2$.

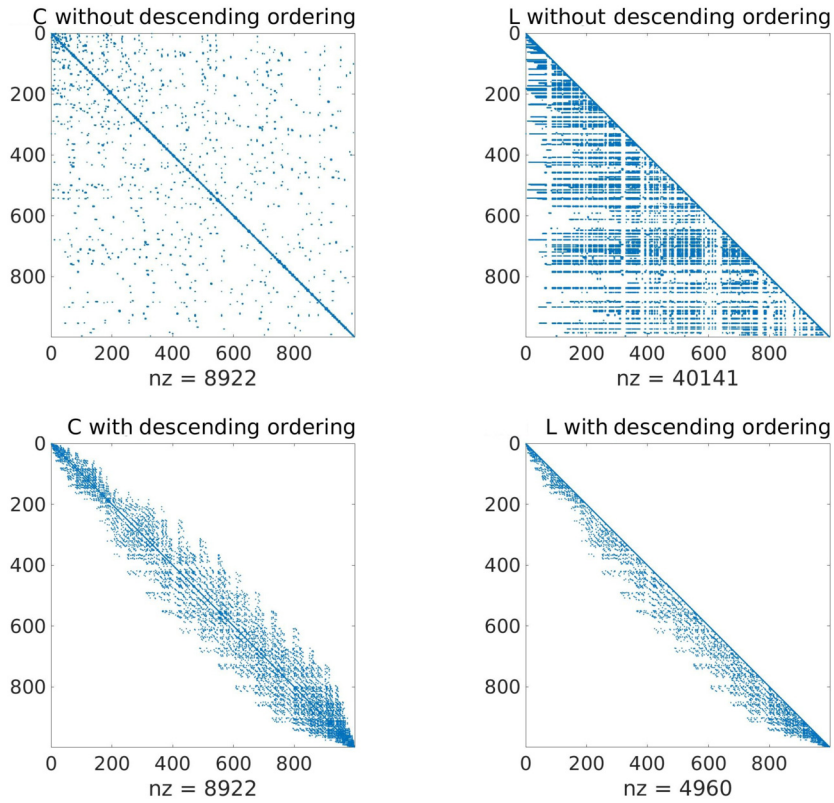

Fig. 5. Nonzero structures of **C** and **L** from algorithms with and without descending ordering, molecule with $n = 1000$, $r = 998$.

2. All bond angles except θ_1 are chosen as hard variables, $r = n - 1$.
3. All bond lengths are chosen as hard variables, $r = n$.
4. Randomly fixing one third of ϕ_i, one third of θ_i and one third of b_i, leave ϕ_1, ϕ_2, θ_1 being free, $r \approx n - 1$.

Also, the ratio of number of branches to number of atoms is fixed as 0.25 for all the four molecules. At the beginning of each calculation, the values of internal coordinates are initialized with random numbers, then the equations in Appendix A are adopted to calculate $\frac{\partial \phi_i}{\partial r_k}$, $\frac{\partial \theta_i}{\partial r_k}$ and $\frac{\partial b_i}{\partial r_k}$, and the nonzero entries in **C** can be calculated, followed by the Cholesky decomposition from **C** to **L**. Finally, Eqs. (6a)–(6e) are solved sequentially. This procedure is repeated 1000 times to give the average calculation time for each n and each constraint strategy. Fig. 7 shows the linear scaling of the distance descending ordering method, as well as the influence of the type of hard variables on the average calculation time. It could be found that all ϕ_i fixed constraint strategy has the largest average calculation time, while all b_i fixed constraint strategy has the smallest average calculation time, and the all θ_i fixed constraint strategy has the medium average calculation time. This agrees with the fact that ϕ_i, θ_i and b_i are affected by 4, 3 and 2 atoms, hence ϕ_i will correlate with more hard variables than θ_i and b_i. This indicates the more unfixed torsional angles, the faster calculation for real molecular simulations.

We also compared the time complexities between the distance descending ordering method and a conventional $O(n^3)$ method. The conventional method involves direct calculation and decomposition of the dense mass matrix $\boldsymbol{\mathcal{M}}$. In each molecule we randomly constrain every one third of the internal variables, thus the degrees of freedom $f \approx 2n + 4$ and number of constraints $r \approx n - 1$. Fig. 8 shows the averaged calculation time for each time step. When $n = 10$, the calculation time for the conventional method is lower than that of descending distance ordering method (7.7e-5 seconds vs 8.7e-5 seconds). However, the computation time of conventional method increases very fast as n gets larger, and the method quickly becomes impractical when n is over the magnitude of 100. On the contrary, the distance descending ordering method always keeps $O(n)$ time complexity.

5. Conclusions

The Fixman's theorem factorizes the inverse of mass matrix as $\boldsymbol{\mathcal{M}}^{-1} = \mathbf{A} - \mathbf{B}\mathbf{C}^{-1}\mathbf{B}^{\top}$, from which the equation of motion $\dot{\mathbf{q}} = \boldsymbol{\mathcal{M}}^{-1}\mathbf{p}$ can be decomposed into a series of $O(n)$ calculations, given that **C** can be efficiently factorized. For molecules with serial chain structure, the nonzero entries of **C** will be clustered about its diagonal, giving linear time complexity to the factorization about **C**. However, for structures with long branches, nonzero entries of **C** will no longer clustered about the

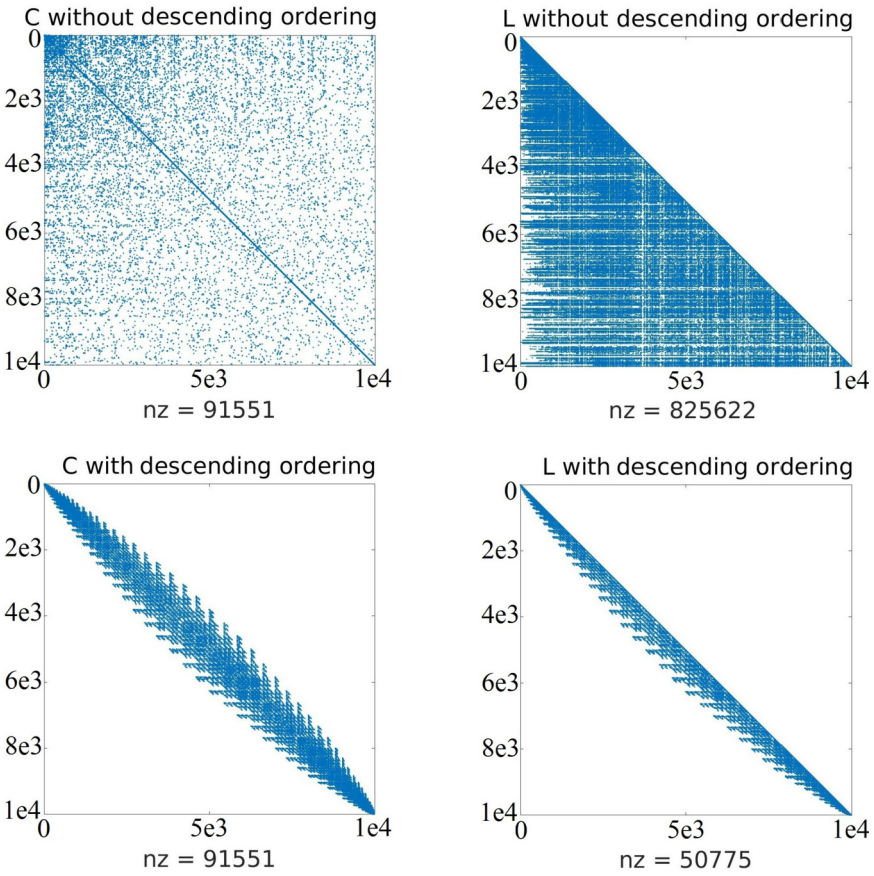

Fig. 6. Nonzero structures of **C** and **L** from algorithms with and without descending ordering, molecule with $n = 10000$, $r = 9999$.

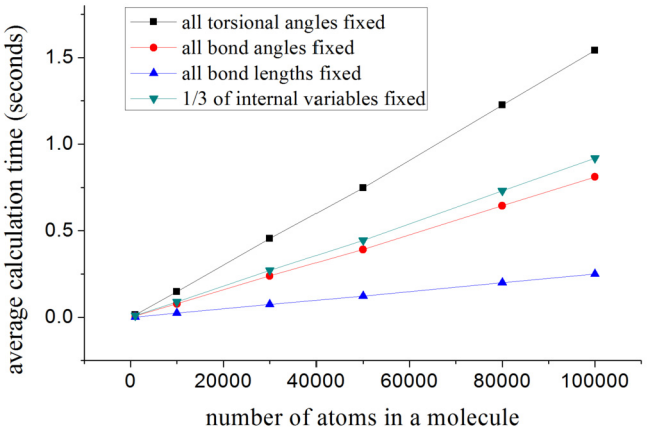

Fig. 7. Average calculation time versus number of atoms.

diagonal, and therefore direct Cholesky decomposition will have $O(n^3)$ time complexity. Although several methods may be applied in factorizing positive definite sparse matrices, none of them can strictly guarantee no fill-in for all molecule models according to our test. The distance descending ordering method considers the problem from a different perspective: it does not do direct row and column permutations about the matrix **C**, instead it applies a reordering about the hard variables. The reordering strategy is developed based on the properties of Cholesky decomposition and the geometrical structure of molecules. By applying the distance descending ordering method, the nonzero structure of **L** will be exactly the same as that of the lower diagonal part of **C**. As a result, the $O(n)$ time complexity can be remained, no matter the molecule structure has long branches or not.

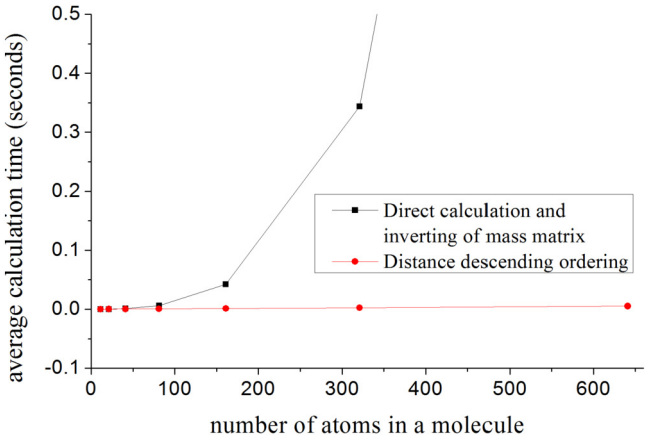

Fig. 8. Comparison with a conventional $O(n^3)$ method.

The distance descending ordering method was proposed with proper proof. The numerical validations have shown its effectiveness. The no fill-in property was demonstrated by using three molecules with number of atoms $n = 100, 1000$ and 10000. The difference between algorithms with and without applying the distance descending ordering method is huge. For results obtained from distance descending ordering method, the number of nonzero entries in $\mathbf{L}$ is roughly linearly scaled with n, this implies the $O(n)$ scaling in solving the equation of motion. However, for algorithm without distance descending ordering method, the order of calculation grows with the increase of n in the manner of $n^{(1<x\leq 3)}$.

.

The $O(n)$ time complexity of this method has also been demonstrated by the average calculation time versus the molecule's number of atoms. The authors found that the types of internal coordinates have significant influence on the average calculation time. Fixed torsional angles tend to have more calculation time than fixed bond angles and bond lengths.

Acknowledgement

The authors gratefully acknowledge the support for this research from the Department of Energy (DOE) of USA under Grant No. DE-EE0005942.

Appendix A. Calculation of $\frac{\partial \mathbf{q}}{\partial \mathbf{r}}$ and $\frac{\partial \mathbf{c}}{\partial \mathbf{r}}$

As shown in section 2, the values of $\frac{\partial \mathbf{q}}{\partial \mathbf{r}}$ and $\frac{\partial \mathbf{c}}{\partial \mathbf{r}}$ are required for the calculation. In other words, we need to calculate $\frac{\partial \phi_i}{\partial \vec{r}_k}$, $\frac{\partial \theta_i}{\partial \vec{r}_k}$ and $\frac{\partial b_i}{\partial \vec{r}_k}$. The classical book [19] has given $\nabla_{\vec{r}_k}\phi_i = \nabla_{\vec{r}_k}\cos\phi_i/(-\sin\phi_i)$ and $\nabla_{\vec{r}_k}\theta_i = \nabla_{\vec{r}_k}\cos\theta_i/(-\sin\theta_i)$. However, this procedure becomes invalid when $\sin\phi_i$ and $\sin\theta_i$ approach zero. Under this condition we need to switch to the alternative approach as $\nabla_{\vec{r}_k}\phi_i = \nabla_{\vec{r}_k}\sin\phi_i/(\cos\phi_i)$ and $\nabla_{\vec{r}_k}\theta_i = \nabla_{\vec{r}_k}\sin\theta_i/(\cos\theta_i)$.

Same as that in [19], two variables are defined to simplify the formulae:

$$\begin{aligned} C_{ij} &= C_{ji} = \vec{b}_i \cdot \vec{b}_j \\ D_{ij} &= D_{ji} = \left|\vec{b}_i \times \vec{b}_j\right|^2 = C_{ii}C_{jj} - C_{ij}^2 \end{aligned} \tag{A.1}$$

where $\vec{b}_i = \vec{r}_i - \vec{r}_{i-1}$ is the bond vector. The sines can be obtained as follows:

$$\begin{aligned} \sin\phi_i &= \frac{\vec{b}_{i-1} \times \left(\vec{b}_{i-2} \times \vec{b}_{i-1}\right)}{\left|\vec{b}_{i-1}\right|\left|\vec{b}_{i-2} \times \vec{b}_{i-1}\right|} \cdot \frac{\vec{b}_{i-1} \times \vec{b}_i}{\left|\vec{b}_{i-1} \times \vec{b}_i\right|} \\ &= \vec{b}_{i-2} \cdot \left(\vec{b}_{i-1} \times \vec{b}_i\right) \sqrt{\frac{C_{i-1i-1}}{D_{i-2i-1}D_{ii-1}}} \end{aligned} \tag{A.2}$$

and

$$\sin\theta_i = \frac{\vec{b}_i \times \left(\vec{b}_{i-1} \times \vec{b}_i\right)}{\left|\vec{b}_i\right|\left|\vec{b}_{i-1} \times \vec{b}_i\right|} \cdot \frac{\vec{b}_{i-1}}{\left|\vec{b}_{i-1}\right|} = \sqrt{\frac{D_{ii-1}}{C_{ii}C_{i-1i-1}}} \tag{A.3}$$

It is easy to verify that

$$\begin{cases} \frac{\partial \phi_i}{\partial \vec{r}_k} \neq 0 & \text{only if } k = i, i-1, i-2, i-3 \\ \frac{\partial \theta_i}{\partial \vec{r}_k} \neq 0 & \text{only if } k = i, i-1, i-2 \\ \frac{\partial b_i}{\partial \vec{r}_k} \neq 0 & \text{only if } k = i, i-1 \end{cases} \tag{A.4}$$

In the following the nonzero terms of $\nabla_{\vec{r}_k} \sin\phi_i$, $\nabla_{\vec{r}_k} \sin\theta_i$ and $\nabla_{\vec{r}_k} b_i$ are given:

$$\begin{aligned}
\nabla_{\vec{r}_i} \sin\phi_i &= C_{i-1i-1}^{1/2} \left(D_{i-2i-1} D_{i-1i}\right)^{-1/2} \left\{ \left(C_{ii-1} \vec{b}_{i-1} - C_{i-1i-1} \vec{b}_i \right) \left[\vec{b}_{i-2} \cdot \left(\vec{b}_{i-1} \times \vec{b}_i \right) \right] / D_{ii-1} \vec{b}_{i-2} \times \vec{b}_{i-1} \right\} \\
\nabla_{\vec{r}_i} \sin\phi_{i+1} &= C_{ii}^{1/2} \left(D_{ii-1} D_{ii+1}\right)^{-1/2} \left\{ \vec{b}_{i-2} \cdot \left(\vec{b}_{i-1} \times \vec{b}_i \right) \left[\vec{b}_i / C_{ii} - \left(C_{i-1i-1} \vec{b}_i - C_{ii-1} \vec{b}_{i-1} \right) / D_{ii-1} \right.\right. \\
&\quad \left.\left. - \left(C_{ii} + C_{ii+1}\right) \vec{b}_{i+1} / D_{ii+1} + \left(C_{i+1i+1} + C_{ii+1}\right) \vec{b}_i / D_{ii+1} \right] + \left(\vec{b}_{i+1} + \vec{b}_i \right) \times \vec{b}_{i-1} \right\} \\
\nabla_{\vec{r}_i} \sin\phi_{i+2} &= C_{i+1i+1}^{1/2} \left(D_{ii+1} D_{i+1i+2}\right)^{-1/2} \left\{ \vec{b}_i \cdot \left(\vec{b}_{i+1} \times \vec{b}_{i+2} \right) \left[\left(C_{i+2i+2} \vec{b}_{i+1} - C_{i+1i+2} \vec{b}_{i+2} \right) / D_{i+1i+2} \right.\right. \\
&\quad \left.\left. + \left(C_{ii} + C_{ii+1}\right) \vec{b}_{i+1} / D_{ii+1} - \left(C_{i+1i+1} + C_{ii+1}\right) \vec{b}_i / D_{ii+1} - \vec{b}_{i+1} / C_{i+1i+1} \right] + \left(\vec{b}_{i+1} + \vec{b}_i \right) \times \vec{b}_{i+2} \right\} \\
\nabla_{\vec{r}_i} \sin\phi_{i+3} &= C_{i+2i+2}^{1/2} \left(D_{i+1i+2} D_{i+2i+3}\right)^{-1/2} \left[\vec{b}_{i+1} \cdot \left(\vec{b}_{i+2} \times \vec{b}_{i+3} \right) \left(C_{i+2i+2} \vec{b}_{i+1} - C_{i+1i+2} \vec{b}_{i+2} \right) / D_{i+1i+2} \right. \\
&\quad \left. + \vec{b}_{i+3} \times \vec{b}_{i+2} \right]
\end{aligned} \tag{A.5}$$

and

$$\begin{aligned}
\nabla_{\vec{r}_i} \sin\theta_i &= D_{i-1i}^{1/2} \left(C_{i-1i-1} C_{ii}\right)^{-1/2} \left[\left(C_{i-1i-1} / D_{i-1i} - 1/C_{ii}\right) \vec{b}_i - C_{i-1i} / D_{i-1i} \vec{b}_{i-1} \right] \\
\nabla_{\vec{r}_i} \sin\theta_{i+1} &= D_{ii+1}^{1/2} \left(C_{ii} C_{i+1i+1}\right)^{-1/2} \left\{ \left[\left(C_{i+1i+1} + C_{ii+1}\right) \vec{b}_i - \left(C_{ii} + C_{ii+1}\right) \vec{b}_{i+1} \right] / D_{ii+1} \right. \\
&\quad \left. + \vec{b}_{i+1} / C_{i+1i+1} - \vec{b}_i / C_{ii} \right\} \\
\nabla_{\vec{r}_i} \sin\theta_{i+2} &= D_{i+1i+2}^{1/2} \left(C_{i+1i+1} C_{i+2i+2}\right)^{-1/2} \left[\left(C_{i+1i+2} \vec{b}_{i+2} - C_{i+2i+2} \vec{b}_{i+1} \right) / D_{i+1i+2} + \vec{b}_{i+1} / C_{i+1i+1} \right]
\end{aligned} \tag{A.6}$$

also,

$$\begin{aligned}
\nabla_{\vec{r}_i} b_i &= \vec{b}_i / b_i \\
\nabla_{\vec{r}_i} b_{i+1} &= -\vec{b}_{i+1} / b_{i+1}
\end{aligned} \tag{A.7}$$

References

[1] J.-P. Ryckaert, G. Ciccotti, H.J. Berendsen, Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes, J. Comput. Phys. 23 (3) (1977) 327–341, http://dx.doi.org/10.1016/0021-9991(77)90098-5, http://www.sciencedirect.com/science/article/pii/0021999177900985.

[2] W.F. Van Gunsteren, M. Karplus, Effect of constraints on the dynamics of macromolecules, Macromolecules 15 (6) (1982) 1528–1544, http://dx.doi.org/10.1021/ma00234a015.

[3] A.K. Mazur, Quasi-Hamiltonian equations of motion for internal coordinate molecular dynamics of polymers, J. Comput. Chem. 18 (11) (1997) 1354–1364, http://dx.doi.org/10.1002/(SICI)1096-987X(199708)18:11<1354::AID-JCC3>3.0.CO;2-K.

[4] M. Fixman, Classical statistical mechanics of constraints: a theorem and application to polymers, Proc. Natl. Acad. Sci. USA 71 (8) (1974) 3050–3053, http://dx.doi.org/10.1073/pnas.71.8.3050.

[5] K. Lee, G.S. Chirikjian, A new perspective on o(n) mass-matrix inversion for serial revolute manipulators, in: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005, pp. 4722–4726, http://ieeexplore.ieee.org/abstract/document/1570849.

[6] Y. Wang, G.S. Chirikjian, A new o(n) method for inverting the mass matrix for serial chains composed of rigid bodies, in: 5th International Conference on Multibody Systems, Nonlinear Dynamics, and Control, Parts A, B, and C, vol. 6, ASME, 2005.

[7] K. Lee, Y. Wang, G.S. Chirikjian, O(n) mass matrix inversion for serial manipulators and polypeptide chains using lie derivatives, Robotica 25 (6) (2007), http://dx.doi.org/10.1017/s0263574707003852.

[8] Y. Saad, Iterative Methods for Sparse Linear Systems, Society for Industrial and Applied Mathematics, 2003.

[9] D.M. Young, W. Rheinboldt, Iterative Solution of Large Linear Systems, Elsevier, 1971.

[10] A. George, J.W. Liu, Computer solution of large sparse positive definite systems, http://dx.doi.org/10.1016/0141-1195(83)90187-0.

[11] M. Yannakakis, Computing the minimum fill-in is NP-complete, SIAM J. Algebraic Discrete Methods 2 (1) (1981) 77–79, http://dx.doi.org/10.1137/0602010.

[12] H.M. Markowitz, The elimination form of the inverse and its application to linear programming, Manag. Sci. 3 (3) (1957) 255–269, http://dx.doi.org/10.1287/mnsc.3.3.255.

[13] W. Tinney, J. Walker, Direct solutions of sparse network equations by optimally ordered triangular factorization, Proc. IEEE 55 (11) (1967) 1801–1809, http://dx.doi.org/10.1109/proc.1967.6011.

[14] D.J. Rose, A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations, in: Graph Theory and Computing, 1972, pp. 183–217.
[15] P.R. Amestoy, T.A. Davis, I.S. Duff, An approximate minimum degree ordering algorithm, SIAM J. Matrix Anal. Appl. 17 (4) (1996) 886–905, http://dx.doi.org/10.1137/s0895479894278952.
[16] M.S. Khaira, G.L. Miller, T.J. Sheffler, Nested Dissection: A Survey and Comparison of Various Nested Dissection Algorithms, CMU-CS, Carnegie Mellon University, School of Computer Science, 1992, https://books.google.com/books?id=f69UPwAACAAJ.
[17] P. Heggernes, S. Eisestat, G. Kumfert, A. Pothen, The Computational Complexity of the Minimum Degree Algorithm, Tech. rep. Dec. 2001.
[18] K. Brandhorst, M. Head-Gordon, Fast sparse cholesky decomposition and inversion using nested dissection matrix reordering, J. Chem. Theory Comput. 7 (2) (2011) 351–368, http://dx.doi.org/10.1021/ct100618s.
[19] M.P. Allen, D.J. Tildesley, Computer Simulation of Liquids, Appendix C, Oxford University Press, 1989.