Efficiency of Rotational Operators for Geometric Manipulation of Chain Molecules

Chaok Seok* and Evangelos A. Coutsias†

Department of Chemistry, College of Natural Sciences, Seoul National University, Seoul 151-747, Korea
*E-mail: chaok@snu.ac.kr
†Department of Mathematics and Statistics, University of New Mexico, Albuquerque, New Mexico 87131, USA
*E-mail: vageli@math.unm.edu

Received July 23, 2007

Geometric manipulation of molecules is an essential elementary component in computational modeling programs for molecular structure, stability, dynamics, and design. The computational complexity of transformation of internal coordinates to Cartesian coordinates was discussed before. The use of rotation matrices was found to be slightly more efficient than that of quaternions although quaternion operators have been widely advertised for rotational operations, especially in molecular dynamics simulations of liquids where the orientation is a dynamical variable. The discussion on computational efficiency is extended here to a more general case in which bond angles and side chain torsion angles are allowed to vary. The algorithm of Thompson is derived again in terms of quaternions as well as rotation matrices, and an algorithm with optimal efficiency is described. The algorithm based on rotation matrices is again found to be slightly more efficient than that based on quaternions.

Key Words: Rotation operator, Rotation matrix, Quaternion, Internal coordinates

Introduction

Cartesian coordinate modeling is popular in molecular studies, for example, in molecular dynamics simulations of biomolecules. In many other situations, internal coordinates are a natural choice when constraints on internal coordinates such as bond lengths and bond angles are to be implemented to reduce the number of degrees of freedom. Additional dihedral angle constraints from experiments or structures of homologous protein sequences can be easily incorporated in internal coordinates.

If the dihedral angles are taken from the actual structure, and the bond lengths and bond angles are fixed at ideal values, the degree of deviation from the actual structure increases with the chain length. For example, a protein structure with 800 amino acids can show RMSD (Root-Mean-Square Deviation) as large as 20 Å if all the bond lengths and bond angles are replaced with ideal values. However, changes in the bond lengths and angles can be absorbed into small variations in dihedral angles to accurately represent the structure. In Ref. 6, it is shown that protein structures can be represented very accurately (within 0.3 Å) with ideal bond lengths and bond angles if dihedral angles are slightly modified to minimize RMSD. For this reason, strategies to take a reduced number of degrees of freedom in internal coordinates are popular in protein structure prediction and design studies.

When internal coordinates are employed for geometry representation, conversion of the internal coordinates to Cartesian coordinates is necessary because most of the realistic energy functions or scoring functions involve terms that are conveniently evaluated in Cartesian coordinates. The transformation requires rotation operators: quaternions, rotation matrices, and other transformations have been used. (See Ref. 1 for more detailed discussions.)

Alvarado and Kazeronian showed that operation in rotation matrix is the most efficient for typical protein chains, and operation using quaternion in the matrix form is almost equivalent. A more general system than that in Ref. [1] is considered here in that bond angles are not required to be kept constant and side chains are not rigid. An efficient algorithm, which is equivalent to that of Thompson [3], is derived by employing a simple reference state and performing rotations in the reverse order. It is shown again that the method using rotation matrix is slightly more efficient than quaternion.

A method that requires minimal number of rotation operations is described below. The method can be described in abstract terms, considering rotations as building blocks. The method is first described in the context of quaternion operators for convenience, and the same method is stated in terms of rotation matrices later. Efficiencies of the two methods using quaternions and rotation matrices are then compared.

Method of Constructing a Chain Molecule with Minimal Number of Operations

A linear chain involving N atoms is considered. Branches can be easily added once the backbone is constructed. This method can be combined with an analytical loop closure algorithm to manipulate chains with internal loops, for example, polypeptides with disulfide bonds or cyclic chains. A linear chain can be fully described by N sets of bond lengths $b$, bond angles $\theta$, and dihedral angles $\phi$ as follows:

$$P_N = \{(b_i, \theta_i, \phi_i), i = 1, \Lambda, N\}, \quad (1)$$
where the backbone atoms of the chain are numbered from 1 to $N$, $b_i$ is the bond length between the nodes at $i$ and $i+1$, $\theta_i$ is the bond angle formed by the three nodes $i-2$, $i$, and $i+1$, and $\phi_i$ is the dihedral angle defined by the four nodes $i-3$, $i-2$, $i$, and $i+1$. Six degrees of freedom, $b_i$, $\theta_i$, $\theta_{i+1}$, $\phi_i$, and $\phi_{i+1}$, do not affect the internal geometry of the chain, but define the absolute location and orientation of the chain in space with respect to ‘virtual’ nodes $i-2$, $i+1$, and 0. Therefore, the internal geometry of the chain can be fully described by $N-1$ bond lengths, $N-2$ bond angles, and $N-3$ dihedral angles.

To construct the molecule, simple rotations are applied starting from a simple initial arrangement. A straight, linear conformation which is laid along the positive $x$-axis is taken as the initial geometry. The initial arrangement is defined by

$$P_i^0 = \{ (b_i, \pi, 0), i = 1, 2, N \} .$$

Next, Cartesian coordinates for the nodes are determined successively. Placing successive atoms requires a rotation at each step, to the desirable $\theta_i$ and $\phi_i$ values. There is a “best” way for performing these rotations.

Recall that rotating a vector $v$ about axis $u$ by angle $w$ is affected through conjugation by the quaternion $q$

$$v \rightarrow v' = q * v * q^{-1} ,$$

with $q$ given by

$$q = \left( \cos \frac{w}{2}, u \sin \frac{w}{2} \right) .$$

Here, ‘*’ is the operator of quaternion multiplication, defined below (see Eq. 20). The $i$-th link in the chain is placed by performing, in succession, all indicated rotations along the chain about the local axes $u_k$ by $\omega_k$, where $k = 1, 2$, $\Lambda$, $i$.

The net quaternion describing this rotation, $q_i$, is constructed by the product of individual quaternions as

$$q_i = p_i * \Lambda * p_2 * p_1, \quad i = 1, 2, N ,$$

where $p_1$ and $p_2$ can be set to unit quaternions ($p*p = 1$) if the node 1 and 2 are to be placed on the $x$-axis for simplicity. It is important to note that in this construction, the axes of each successive rotation, $u_k$, are determined by the effect of all previous rotations on the initial vector, $u_k$ as

$$u_k = q_{k-1} * u_k^0 * q_{k-1}^c .$$

However, by carrying out the rotations in the reverse order, the same net rotation can be affected by utilizing the initial axes as

$$q_i = p_i * \Lambda * p_2 * p_1 = p_1 * p_2 * \Lambda * p_1 .$$

The reason for this, obvious geometrically, can be also seen as follows:

$$q_i = p_1 * q_{i-1}$$

$$= \left( \cos \frac{w_i}{2}, u_i \sin \frac{w_i}{2} \right) * q_{i-1} .$$

Comparison of Rotation Matrix and Quaternion Methods

Rotation matrix method. One can construct a $3 \times 3$ rotation matrix $U_i$ for each $(\theta_i, \phi_i)$ pair, and multiply by a vector to get

$$R_i = R_i - 1 + b_i U_i(\theta_i, \phi_i, \Lambda, \theta_i, \phi_i)$$

$$U_i(\theta_i, \phi_i, \Lambda, \theta_i, \phi_i) = U_1(\theta_1, \phi_1, \Lambda, \theta_1, \phi_1) U_0(\theta, \phi, \Lambda)_0 .$$

where $r_0$ is an arbitrary unit vector. The $i$-th rotation matrix $U_i$ depends on all the angles indexed 1 through $i$ because the axis of rotation changes with previous angles. This can be avoided if rotations are performed in the reverse order:

$$R_i = R_i - 1 + b_i U_i^0(\theta_i, \phi_i, \Lambda, \theta_i, \phi_i) U_0^0(\theta, \phi, \Lambda)_0 U_0^0(\theta, \phi, \Lambda)_0 \hat{e}_x .$$

where the superscript 0 implies rotation about fixed axes. The rotation matrix $U_i^0$ can be expressed as a multiplication of two rotation matrices $U_i(\theta_i)$ and $U_i(\phi_i)$, where $U_0$ is matrix for rotation by $\pi$-$\theta$ around the $z$-axis, and $U_z$ for rotation by $\phi_i$ around the $x$-axis. This simplification is made possible by setting $r_0$ equal to a unit vector $\hat{e}_x$ along the $x$-axis. The resulting $U_i^0$ is

$$- \left( \cos \frac{w_i}{2}, [q_{i-1} * u_i * q_{i-1}^c] * \sin \frac{w_i}{2} \right) * q_{i-1} .$$
Efficiency of Rotational Operators for Geometric Manipulation


and the rotation matrix corresponding to quaternion \( q = (q_0, q_1, q_2, q_3) \) is

\[
U(q) = \begin{pmatrix}
q_0^2 + q_1^2 - 1/2 & q_1 q_2 - q_0 q_3 & q_1 q_3 + q_0 q_2 \\
q_1 q_2 + q_0 q_3 & q_0^2 + q_2^2 - 1/2 & q_2 q_3 - q_1 q_0 \\
q_1 q_3 - q_0 q_2 & q_2 q_0 + q_1 q_3 & q_0^2 + q_2^2 - 1/2
\end{pmatrix}.
\] (21)

the cost of computation for adding atom \( i \) is counted as follows:

1. Division of the angles \( \phi \) and \( \theta \) by 2 (2 divisions).
2. Calculation of cosines and sines, \( \cos(\phi/2) \) and \( \cos(\theta/2) \).
3. Quaternion update, \( U_i U_{i-1} q_i \) (8 multiplications, 4 additions).
4. Quaternion update, \( U_i U_{i-1} q_i \) (8 multiplications, 4 additions).
5. First column of \( U_i \) (no cost).
6. Scaling by \( b \) and shift by \( R_{i-1} \) (3 multiplications and 3 additions).

The computational cost of adding a terminal atom is the same as above because \( q_i \) needs to be calculated to get the first column of \( U(q)/2 \).

Comparison of computational efficiency. The minimum number of mathematical operations required for calculation of the position of one atom to be added to a growing chain is summarized in Table 1.

The total number of operations for constructing the whole chain is approximately the number of atoms times the above numbers. Note that this linear dependency on the number of atoms comes from the reverse rotation. If the operations \( \cos \) and \( \sin \) are assumed to be 20 times more expensive than \( +, -, \times, \text{and} \div \), which are assumed to be equal in computational time, the relative computational costs are 122 and 123 for rotation matrix and quaternion methods, respectively. If, alternatively, the computation of \( \cos \) and \( \sin \)

Table 1. Comparison of the number of operations for the two methods

<table>
<thead>
<tr>
<th>Method</th>
<th>( + ) or (-)</th>
<th>( \times )</th>
<th>( \div )</th>
<th>( \cos )</th>
<th>( \sin )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotation matrix</td>
<td>15</td>
<td>27</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Quaternion</td>
<td>15</td>
<td>26</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 2. Comparison of the number of operations for the two methods when applied to two proteins of different sizes

<table>
<thead>
<tr>
<th>Method</th>
<th>Kinase</th>
<th>Keratin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotation matrix</td>
<td>105 831</td>
<td>269 971</td>
</tr>
<tr>
<td>Quaternions</td>
<td>11 5989</td>
<td>296 799</td>
</tr>
<tr>
<td>Rotation matrix with half-tangent formula</td>
<td>81 313</td>
<td>207 233</td>
</tr>
<tr>
<td>Quaternions with half-tangent formula</td>
<td>89 585</td>
<td>229 235</td>
</tr>
</tbody>
</table>
is performed using the familiar half-tangent formulas
\[
\cos \theta = \frac{1 - u^2}{1 + u^2}, \quad \sin \theta = \frac{2u}{1 + u^2}, \quad \text{where} \quad u = \tan \left( \frac{\theta}{2} \right), \quad (22)
\]
the costs of the two methods then become 96 and 95 FLOPs, respectively (again assuming the cost of the tangent as 20 operations).

Using rotation matrices or quaternions for accumulating rotations for non-terminal atoms are therefore almost equivalent in terms of computational cost, but quaternions require more operations for terminal atoms. Overall, rotation matrix calculation is more efficient, the exact cost depending on the number of terminal atoms. Table 2 shows a comparison of the number of FLOPs when the two methods are applied to two proteins, Abl tyrosine kinase and heparin-binding growth factor, following Ref. 1. The kinase has 62 amino acids: MNDPNLFVALYDFVASGNTLSITKGEKLRVLGYNHNGECWAEQTKNGQGWVPSNYITPVMNS. Keratin has 154 amino acids: MAAGSITTPALPDGSGAFLPPHGKDPKRLYCKINGFFLRHPGRVDGVERKSDPHIKLQQAERGGVSIGVSANRYLANKEDG RLLASKSVTDECFFERLESNNYNTYSRKYTSWVA LKRTGQYKLGSKTGPQKAILFLPMSAKS. According to Table 2, the rotation matrix method is about 10% more efficient whether the half-tangent formula of Eq. (22) is used or not, because of the differences in FLOPs required to invert terminal atoms.

**Discussion**

An algorithm with minimal rotation operations for interconversions between internal and Cartesian coordinate systems for a molecular chain is described, starting from a simple, straight reference state and applying rotations on the chain backwards. The rotation matrix formula is equivalent to that of Thompson. The number of algebraic operations is reduced compared to the methods described in Ref. 1 due to the use of a simpler reference. In Ref. 1, the cost is 70 operations (+, −, ×, and ÷) for calculation of operator and 15 for placement of atom when rotation matrices are used, and 74 and 15, respectively, when quaternions are used. In the algorithms described here, the cost is 36 and 6 with rotation matrices, and 37 and 6 with quaternions. (All methods require equal numbers of sin/cos evaluations). The conclusion that the use of rotation matrices results in slightly more efficient computation than the use of quaternions is maintained. Efficient transformation of coordinate systems can facilitate studies on protein structure especially when local constraints are incorporated in natural manners.

**Acknowledgement.** EAC and CS would like to acknowledge the hospitality of the Dill group at UCSF during their stay. CS acknowledges grants from the Seoul R&BD program, and a grant from MarineBio21, Ministry of Maritime Affairs and Fisheries, Korea.

**References**