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.[1] The use of rotation matrices was found to be slightly more efficient than that of quaternion although quaternion operators have been widely advertised for rotational operations, especially in molecular dynamics simulations of liquids where the orientation is a dynamical variable.[2] The discussion on computational efficiency is extended here to a more general case in which bond angles and sidechain torsion angles are allowed to vary. The algorithm of Thompson[3] 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.[4,5] 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,[7,8] rotation matrices,[9] and other transformations[10] have been used. (See Ref. 1 for more detailed discussions.)

Alvarado and Kazerounian[1] 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[11] 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 θ_i, and dihedral angles ϕ_i 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–1 and i, θ_i is the bond angle formed by the three nodes i–2, i–1, and i, and ϕ_i is the dihedral angle defined by the four nodes i–3, i–2, i–1, and i. Six degrees of freedom, b_1, θ_1, θ_2, ϕ_1, ϕ_2, and ϕ_3, 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 $i = -2$, -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

$$P_N^o = \{(b_i,\ \pi,\ 0),\ i = 1,\ \Lambda,\ N\}\,. \tag{2}$$

Next, Cartesian coordinates for the nodes are determined successively. Placing successive atoms requires a rotation at each step, to the desirable θ_i and ϕ_i values. There is a "best" way for performing these rotations.

Recall that[2,12] rotating a vector $\mathbf{v}$ about axis $\mathbf{u}$ by angle w is affected through conjugation by the quaternion q

$$\mathbf{v} \rightarrow \mathbf{v}' = q*\mathbf{v}*q^c\,, \tag{3}$$

with q given by

$$q = \left(\cos\frac{w}{2},\ \mathbf{u}\,\sin\frac{w}{2}\right). \tag{4}$$

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 $\mathbf{u}_k$ by w_k, where k = 1, 2, Λ, i.

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

$$q_i = p_i*\Lambda*p_2*p_1,\quad i = 1,\ 2,\ \Lambda,\ N, \tag{5}$$

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, $\mathbf{u}_k$ are determined by the effect of all previous rotations on the initial vector, $\mathbf{u}_k^0$ as

$$\mathbf{u}_k = q_{k-1}*\mathbf{u}_k^0*q_{k-1}^c\,. \tag{6}$$

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

$$q_i = p_i*\Lambda*p_2*p_1 = p_1^0*p_2^0\Lambda*p_i^0\,. \tag{7}$$

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

$$\begin{aligned} q_i &= p_i*q_{i-1} \\ &= \left(\cos\frac{w_i}{2},\ \mathbf{u}_i\sin\frac{w_i}{2}\right)*q_{i-1} \\ &= \left(\cos\frac{w_i}{2},\ [q_{i-1}*\mathbf{u}_i*q_{i-1}^c]\sin\frac{w_i}{2}\right)*q_{i-1} \\ &= q_{i-1}*\left(\cos\frac{w_i}{2},\ \mathbf{u}_i^0\,\sin\frac{w_i}{2}\right)*q_{i-1}^c*q_{i-1} \\ &= q_{i-1}*p_i^0\,. \end{aligned} \tag{8}$$

p_i^0 is the multiplication of $p_x(\phi_i)$, rotation about the x-axis by ϕ_i, and $p_z(\theta_i)$, rotation about the z-axis by π–θ_i:

$$\begin{aligned} p_i^0 &= p_x(\phi_i)p_z(\theta_i) \\ &= \left(\cos\frac{\phi_i}{2},\ \hat{\mathbf{e}}_x\sin\frac{\phi_i}{2}\right)*\left(\cos\frac{\pi-\theta_i}{2},\ \hat{\mathbf{e}}_z\sin\frac{\pi-\theta_i}{2}\right). \end{aligned} \tag{9}$$

This construction is understood as follows: start morphing the chain from its end, node 1; rotate the bond vector (b_i, 0, 0) about the positive z-axis by angle π–θ_i (this enforces correct bond angle at node i); rotate about the positive x-axis by angle ϕ_i to create the proper dihedral angle; then apply the rotation accumulated by placement of all the previous atoms. The rotations are simple, either around the x- or z-axis, and always applied to a vector on the x-axis. The algorithm can thus be implemented very efficiently. The calculation of the Cartesian coordinate $\mathbf{R}_i$ for the ith node is now collected as

$$q_i = q_{i-1}*\left(\cos\frac{\phi_i}{2},\ \hat{\mathbf{e}}_x\sin\frac{\phi_i}{2}\right)*\left(\cos\frac{\pi-\theta_i}{2},\ \hat{\mathbf{e}}_z\sin\frac{\pi-\theta_i}{2}\right) \tag{10}$$

$$\mathbf{R}_i = \mathbf{R}_{i-1} + b_i q_i\hat{\mathbf{e}}_x q_i^c\,, \tag{11}$$

where i = 1, 2, Λ, N. $\mathbf{R}_0$ and first few quaternions can be chosen to give simple arrangement for the first few atoms when only internal arrangement is of interest, or be chosen to place the chain in a precise location and orientation in space depending on the purposes.

Comparison of Rotation Matrix and Quaternion Methods

Rotation matrix method. One can construct a 3×3 rotation matrix U_i for each (θ_i, ϕ_i) pair, and multiply by a vector to get

$$\begin{aligned} \mathbf{R}_i = \mathbf{R}_{i-1} &+ b_i U_i(\theta_i,\phi_i,\Lambda,\theta_1,\phi_1) \\ &U_{i-1}(\theta_{i-1},\phi_{i-1},\Lambda,\theta_1,\phi_1)\Lambda\,U_1(\theta_i,\phi_i)\mathbf{r}_0\,, \end{aligned} \tag{12}$$

where $\mathbf{r}_0$ 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:

$$\mathbf{R}_i = \mathbf{R}_{i-1} + b_i U_1^0(\theta_1,\phi_1)U_2^0(\theta_2,\phi_2)\Lambda\,U_i^0(\theta_i,\phi_i)\hat{\mathbf{e}}_x\,, \tag{13}$$

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 $U_z(\theta_i)$ and $U_x(\phi_i)$, where U_z is matrix for rotation by π–θ_i around the z-axis, and U_x for rotation by ϕ_i around the x-axis. This simplification is made possible by setting $\mathbf{r}_0$ equal to a unit vector $\hat{\mathbf{e}}_x$ along the x-axis. The resulting U_i^0 is

$$U_i^0(\theta_i, \phi_i) = U_x(\phi_i)U_z(\theta_i) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi_i & -\sin\phi_i \\ 0 & \sin\phi_i & \cos\phi_i \end{pmatrix} \begin{pmatrix} -\cos\theta_i & -\sin\theta_i & 0 \\ \sin\theta_i & -\cos\theta_i & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (14)$$

Each node i is added to a growing chain with i–1 nodes by updating the accumulated rotation,

$$V_i = U_1^0 U_2^0 \Lambda U_{i-1}^0 U_i^0 = V_{i-1} U_i^0, \quad (15)$$

applying the rotation on he vector (b_i, 0, 0), and translating the resulting vector by $\mathbf{R}_{i-1}$.

The computational cost of this procedure is counted as follows:

1. Calculation of cosines and sines, $\cos\phi_i$, $\sin\phi_i$, $\cos\theta_i$, and $\sin\theta_i$.

2. Rotation update, $V_i' = V_{i-1}U_x(\phi_i)$ (12 multiplications and 6 additions).

3. Rotation update, $V_i = V_i' U_z(\theta_i)$ (12 multiplications and 6 additions).

4. First column of U_i^0 (no cost).

5. Scaling by b_i and shift by $\mathbf{R}_{i-1}$ (3 multiplications and 3 additions).

In fact, the above algorithm is equivalent to the algorithm described by Thompson[3] in 1967 using rotation matrices in a local coordinate system. A physical procedure of chain construction is provided here, and an algorithm using quaternions can be easily derived using this procedure, as shown above.

The computational cost of adding an atom at a branching point is the same as the above if the rotation matrix V_i has to be updated for later atoms connected to it. If an atom is terminal, only the first column of that rotation matrix is required, which is

$$V_{i-1} \begin{pmatrix} -\cos\theta_i \\ \sin\theta_i \cos\phi_i \\ \sin\theta_i \sin\phi_i \end{pmatrix}. \quad (16)$$

The cost of adding a terminal node is therefore 11 multiplications and 6 additions for Eq. (16) and 3 multiplications and 3 additions for scaling by b_i and shift by $\mathbf{R}_{i-1}$, in addition to the cost of calculating cosines and sines.

Quaternion method. Quaternions, instead of rotation matrices, can be used to accumulate successive rotations as follows:

$$\mathbf{R}_i = \mathbf{R}_{i-1} + (b_i p_1^0 p_2^0 \Lambda p_i^0)\hat{\mathbf{e}}_x (b_i p_1^0 p_2^0 \Lambda p_i^0)^c, \quad (17)$$

where p_i^0 is given by Eq. (9). The four components of the quaternions $p_x(\phi_i)$ and $p_z(\phi_i)$ are

$$p_x(\phi_i) = \left(\cos\frac{\phi_i}{2}, \sin\frac{\phi_i}{2}, 0, 0\right), \quad (18)$$

$$p_z(\theta_i) = \left(\cos\frac{\pi - \theta_i}{2}, 0, 0, \sin\frac{\pi - \theta_i}{2}\right), \quad (19)$$

Noting[12] that multiplication of two quaternions $a = (a_0, \mathbf{a})$ and $b = (b_0, \mathbf{b})$ is given by

$$ab = (a_0 b_0 - \mathbf{a} \cdot \mathbf{b}, a_0 \mathbf{b} + b_0 \mathbf{a} + \mathbf{a} \times \mathbf{b}), \quad (20)$$

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

$$U(q) = 2\begin{pmatrix} q_0^2 + q_1^2 - 1/2 & q_1q_2 - q_0q_3 & q_1q_3 - q_0q_2 \\ q_1q_2 + q_0q_3 & q_0^2 + q_2^2 - 1/2 & q_2q_3 - q_0q_1 \\ q_1q_3 - q_0q_2 & q_2q_3 + q_0q_1 & q_0^2 + q_3^2 - 1/2 \end{pmatrix}, \quad (21)$$

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

1. Division of the angles θ_i and ϕ_i by 2 (2 divisions).

2. Calculation of cosines and sines, $\cos\frac{\phi_i}{2}$, $\sin\frac{\phi_i}{2}$, $\cos\frac{\theta_i}{2}$, and $\sin\frac{\theta_i}{2}$.

3. Quaternion update, $q_i' = q_{i-1}p_x(\phi_i)$ (8 multiplications, 4 additions).

4. Quaternion update, $q_i = q_i' p_z(\theta_i)$ (8 multiplications, 4 additions).

5. First column of $U(q_i)/2$ (6 multiplications, 4 additions).

6. Scaling by $2b_i$ and shift by $\mathbf{R}_{i-1}$ (4 multiplications, 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 $U(q_i)/2$.

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 dependence 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 +, –, ×, and ÷, 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

Method	+ or –	×	÷	cos	sin
Rotation matrix	15	27	0	2	2
Quaternion	15	26	2	2	2

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

Method	Kinase	keratin
Rotation matrix	105 831	269 971
Quaternion	115 989	296 799
Rotation matrix with half-tangent formula	81 313	207 233
Quaternion with half-tangent formula	89 585	229 235

is performed using the familiar half-tangent formulas

$$\cos\theta = \frac{1-u^2}{1+u^2}, \quad \sin\theta = \frac{2u}{1+u^2}, \quad \text{where } u = \tan\left(\frac{\theta}{2}\right), \quad (22)$$

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: MNDPNLFVALYDFVASGDNTLSITKGEKLRVLGYNHNGEWCEAQTKNGQGWVPSNYITPVNS. Keratin has 154 amino acids: MAAGSITTLPALPEDGGSGAFPPGHFKDPKRLYCKNGGFFLRIHPDGRVDGVREKSDPHIKLQLQAEERGVVSIKGVSANRYLAMKEDGRLLASKSVTDECFFFERLESNNYNTYRSRKYTSWVALKRTGQYKLGSKTGPGQKAILFLPMSAKS. 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.[3] 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

1. Alvarado, C.; Kazerounian, *Prot. Eng.* **2003**, *16*, 717.
2. Rappaport, D. C. *The Art of Molecular Dynamics Simulation*; Cambridge Univ. Press: New York, 1995.
3. Thompson, H. B. *J. Chem. Phys.* **1967**, *47*, 3407.
4. Lee, S.; Kim, Y. *Bull. Korean Chem. Soc.* **2004**, *25*, 838.
5. Lee, S. H.; Pak, Y. *Bull. Korean Chem. Soc.* **2004**, *25*, 533.
6. Coutsias, E. A.; Seok, C.; Dill, K. A. *J. Comput. Chem.* **2004**, *25*, 1849.
7. Jain, A.; Vaidehi, N.; Rodriguesz, G. *J. Comput. Phys.* **1993**, *106*, 258.
8. Kneller, G.; Hinsen, K. *Phys. Rev. E* **1994**, *50*, 1559.
9. Dullweber, A.; Leimkuhler, B.; McLachlan, R. *J. Chem. Phys.* **1997**, *107*, 5840.
10. Manocha, D.; Zhu, Y.; Wright, W. *Comput. Appl. Biol. Sci.* **1995**, *11*, 71.
11. Coutsias, E. A.; Seok, C.; Jacobson, M. P.; Dill, K. A. *J. Comput. Chem.* **2004**, *25*, 510.
12. Coutsias, E. A.; Romero, L. *The Quaternions with Applications to Rigid Body Dynamics*; Sandia National Laboratories, Technical Report SAND2004-0153, 2004.