Representing Proteins in Silico and Protein Forward Kinematics[*]

Lydia E. Kavraki

Abstract

This module discusses how to represent proteins in terms of the Cartesian coordinates of their atoms and in terms of the angle values of their rotatable bonds. It then discusses Forward Kinematics, which allows the computation of Cartesian coordinates when the torsional angle values are known.

1

Topics in this Module

2 Modeling Proteins on a Computer

In order to construct efficient, maintainable software to deal with and manipulate protein structures, a suitable way to store these structures has to be adopted. Depending on the ultimate application, different representations may have advantages and disadvantages from a software perspective. For example, when designing a simple visualization software, the Cartesian (x,y,z) coordinates of each atom are useful and simple to render on the screen. However, if the program is to manipulate bond angles and bond lengths for

[*]Version 1.15: Jun 11, 2007 6:50 pm -0500
[†]http://creativecommons.org/licenses/by/1.0

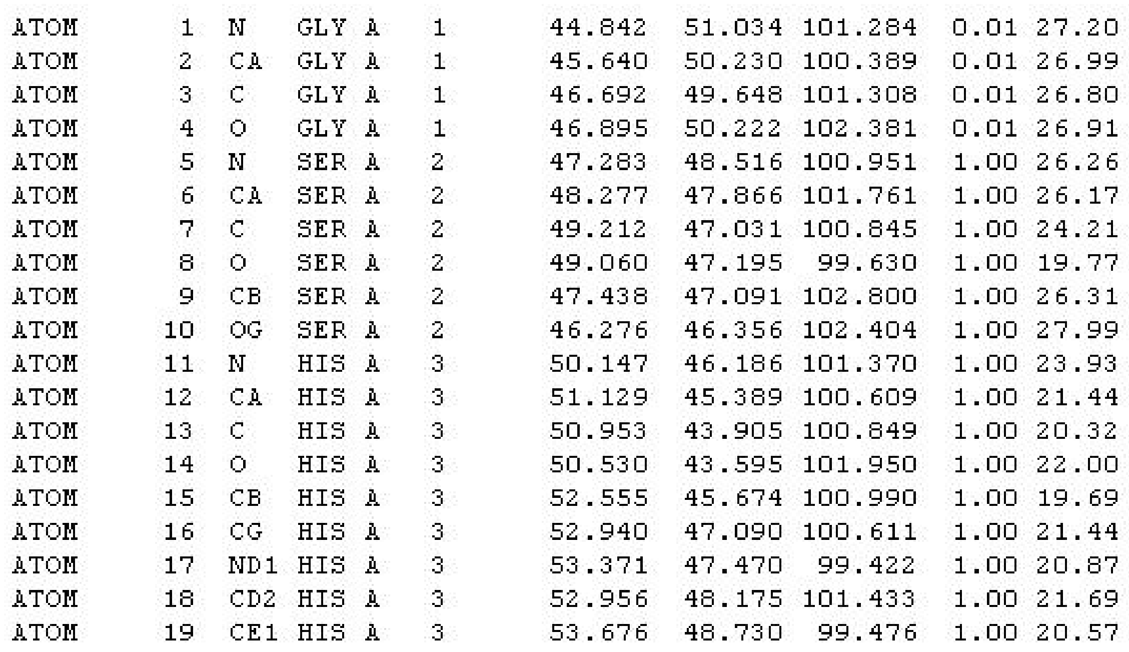
example, a representation based on the internal degrees of freedom (see below) may be more appropriate. Some applications may even need to store more than one representation at a time; for example a simulation program that needs to compute a protein's Potential Energy, which is a function of both Cartesian and Internal coordinates, would benefit from keeping both representations at the same time.

The **structure** of a protein is the set of atoms it contains, and the bonds that join them, that is, its inherent connectivity. A particular geometric shape of a protein (that is, the spatial arrangement of the atoms in the molecule) is called its **conformation**. Thus, a given protein structure can have many different conformations. Next, we discuss the two most common ways to model protein structures and conformations for software applications: Cartesian and Dihedral representations.

2.1 Cartesian Representation of Protein Conformations

The most essential information for modeling a protein structure is the relative position of each atom, given as (x,y,z) Cartesian coordinates. Popular imaging methods such as X-Ray Crystallography, Nuclear Magnetic Resonance (NMR) and Cryogenic Electron Microscopy (Cryo-EM) are used to experimentally obtain relative atom positions from protein crystals and solutions. This is precisely the information provided by Protein Databank (PDB) format coordinate files:

First 19 atom coordinate records of PDB entry 2HLA



```
ATOM      1  N   GLY A   1      44.842  51.034 101.284  0.01 27.20
ATOM      2  CA  GLY A   1      45.640  50.230 100.389  0.01 26.99
ATOM      3  C   GLY A   1      46.692  49.648 101.308  0.01 26.80
ATOM      4  O   GLY A   1      46.895  50.222 102.381  0.01 26.91
ATOM      5  N   SER A   2      47.283  48.516 100.951  1.00 26.26
ATOM      6  CA  SER A   2      48.277  47.866 101.761  1.00 26.17
ATOM      7  C   SER A   2      49.212  47.031 100.845  1.00 24.21
ATOM      8  O   SER A   2      49.060  47.195  99.630  1.00 19.77
ATOM      9  CB  SER A   2      47.438  47.091 102.800  1.00 26.31
ATOM     10  OG  SER A   2      46.276  46.356 102.404  1.00 27.99
ATOM     11  N   HIS A   3      50.147  46.186 101.370  1.00 23.93
ATOM     12  CA  HIS A   3      51.129  45.389 100.609  1.00 21.44
ATOM     13  C   HIS A   3      50.953  43.905 100.849  1.00 20.32
ATOM     14  O   HIS A   3      50.530  43.595 101.950  1.00 22.00
ATOM     15  CB  HIS A   3      52.555  45.674 100.990  1.00 19.69
ATOM     16  CG  HIS A   3      52.940  47.090 100.611  1.00 21.44
ATOM     17  ND1 HIS A   3      53.371  47.470  99.422  1.00 20.87
ATOM     18  CD2 HIS A   3      52.956  48.175 101.433  1.00 21.69
ATOM     19  CE1 HIS A   3      53.676  48.730  99.476  1.00 20.57
```

Figure 1: The third column lists the atom type and the seventh, eighth, and ninth columns contain the x, y, and z coordinates of each atom. These Cartesian coordinates are given in relation to some reference frame determined by the experimental imaging technique, which is not important. The conformation is uniquely specified by the relative positioning of the atoms.

The coordinates and type of each atom, together with the amino acid type they belong to, are sufficient information to reconstruct the connectivity (bonding) of a protein, and therefore sufficient to render an image of the protein. If one wishes to allow the protein to move in a realistic fashion, however, more information may be necessary.

2.2 The Internal Degrees of Freedom of a Protein

The **degrees of freedom** of a system are a set of parameters that may be varied independently to define the state of the system. For example, the location of a point in the Cartesian 2D plane may be defined as a displacement along the x-axis and a displacement along the y-axis, given as a (x,y) pair. It may also be given as a rotation about the origin by θ degrees and a distance r from the origin, given as a (r,θ) pair. In either case, a point moving freely in a plane has exactly two degrees of freedom.

As mentioned before, the spatial arrangement of the atoms in a protein constitute its conformation. In the PDB coordinate file above, we can see that one obvious way to define a protein conformation is by giving x, y, and z coordinates for each atom, relative to some arbitrary origin. These are not independent degrees of freedom, however, because atoms within a molecule are not allowed to leave the vicinity of their neighboring atoms (if no chemical reaction takes place). Pairs of atoms bonded to each other, for example, are constrained to remain close, so moving one atom causes others connected to it to move in a dependent fashion. In the kinematics terminology, this means that the true, effective or independent number of degrees of freedom is much less than the input space parameters -an (x,y,z) tuple for each atom-. The remainder of this section defines a set of independent degrees of freedom that more readily model how proteins and other organic molecules can actually move.

2.2.1 Bonds and Bond Length

The atoms in proteins are connected to one another through covalent bonds. Each pair of bonded atoms has a preferred separation distance called the **bond length**. The bond length can vary slightly with a spring-like vibration, and is thus a degree of freedom, but realistic variations in bond length are so small that most simulations assume it is fixed for any pair of atoms. This is a very common assumption in the literature and reduces the effective degrees of freedom of a protein; the remainder of this module makes this assumption.

Although bond lengths will not be allowed to vary in this work, the presence of bonds is still important because it allows us to represent the connectivity of the protein as an undirected graph data structure, where the atoms are the nodes and the bonds between them are undirected edges. In some cases, it is helpful to artificially break any cycles in the graph, and choose an atom from the interior as an anchor atom. The graph can then be treated as a tree data structure, with the anchor atom as the root.

A Protein as a Graph Data Structure

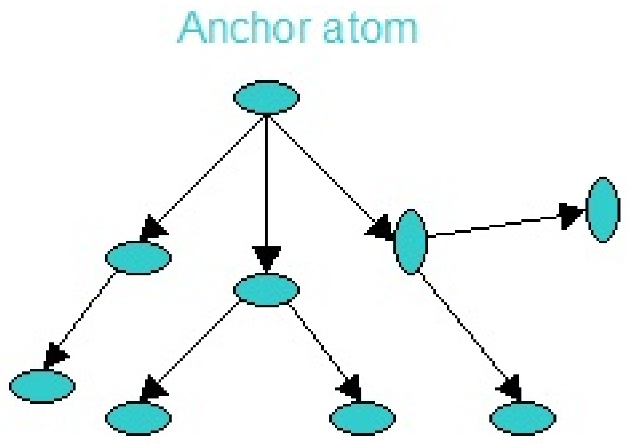


Figure 2: A tree-like representation of protein connectivity, for a very small molecule. Cycles are broken by ignoring one bond in each.

2.2.2 Bond Angles

Bond length is an independent degree of freedom given two connected atoms. A set of three atoms bonded in sequence defines another degree of freedom: the angle between the two adjacent bonds. This is, appropriately, referred to as the **bond angle**. The bond angle can be calculated as the angle between the two vectors corresponding to the bonds from the central atom to each of its neighbors. As a reminder, the angle between two vectors is the inverse cosine of the ratio of the dot product of the vectors to the product of their lengths. Like bond lengths, bond angles tend to be characteristic of the atom types involved, and, with few exceptions, vary little. Thus, like bond lengths, this module considers all bond angles as fixed (again, this is a common assumption).

2.2.3 Dihedral Angles

In most organic molecules, including proteins, the most important internal degree of freedom is rotation about **dihedral (torsional) angles**. A dihedral angle is defined by four consecutively bonded atoms. Given four consecutive atoms A_{i-2}, A_{i-1}, A_i, and A_{i+1}, the dihedral angle is defined as the smallest angle between the planes π_1 and π_2, as shown in the figure. Variation of the dihedral angle is a consequence of rotation of the two outer bonds about the central bond.

A Dihedral Angle

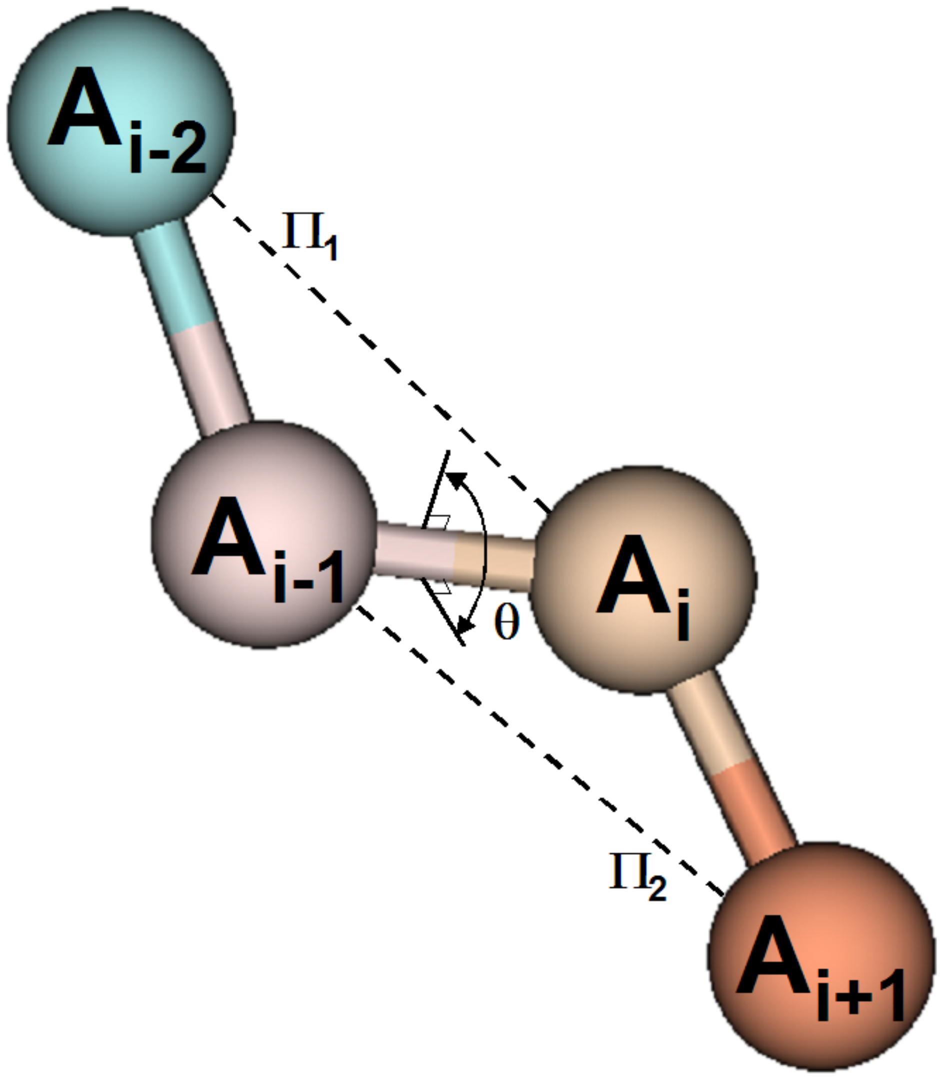


Figure 3: π_1 is the plane uniquely defined by the first three atoms A_{i-2}, A_{i-1}, and A_i. Similarly, π_2 is the plane uniquely defined by the last three atoms A_{i-1}, and A_i, and A_{i+1}. The dihedral angle, θ, is defined as the smallest angle between these two planes. You can read more about the angle between two intersecting planes here[1].

In this module, because bond lengths and bond angles are being ignored as underlying degrees of freedom of a protein, the only remaining degrees of freedom are the dihedral rotations. Representing protein conformations with the dihedral angles as the only underlying degrees of freedom is known as the **idealized** or **rigid geometry model**. Ignoring bond lengths and bond angles greatly reduces the number of degrees of freedom and therefore the computational complexity of representing and manipulating protein structures. Even more efficient representations which reduce the number of degrees of freedom even further exist [2], but these are beyond the scope of this introduction.

[1]http://mathworld.wolfram.com/Plane.html

2.3 Dihedral Representation of Protein Conformations

All amino acids share the same core of one nitrogen, two carbon, and one oxygen atoms. This shared core makes up the backbone of the protein. There are two freely rotatable backbone dihedral angles per amino acid residue in the protein chain: the first, designated ϕ, is a consequence of the rotation about the bond between N and C_α, and the other, ψ, which is a consequence of the rotation about the bond between C_α and C. The peptide bond between C of one residue and N of the adjacent residue is not rotatable.

The number of backbone dihedrals per amino acid is 2, but the number of side chain dihedrals varies with the length of the side chain. Its value ranges from 0, in the case of glycine, which has no sidechain, to 5 in the case of arginine.

Dihedral Angles in Arginine

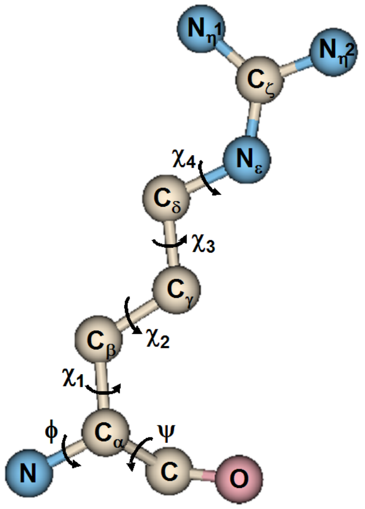


Figure 4: The backbone atoms appear at the bottom of the illustration (the peptide bond is not rotatable). The sidechain dihedrals are conventionally designated by χ and a subscript.

One can generate different three dimensional structures of the same protein by varying the dihedral angles. There are 2N backbone dihedral DOFs for a protein with N amino acids, and up to 4N side chain dihedrals that one can vary to generate new protein conformations. Changes in backbone dihedral angles generally have a greater effect on the overall shape of the protein than changes in side chain dihedral angles. Think about why.

3 Protein Forward Kinematics

Kinematics is a branch of mechanics concerned with how objects move in the absence of mass (inertia) and forces. You can imagine that varying the dihedral angles will move a protein's atoms relative to each other in space. The problem of computing the new spatial locations of the atoms given a set of dihedral rotations is known as the **forward kinematics** problem.

The importance of this problem to protein modeling and simulation should be clear: as stated earlier, the only internal degrees of freedom usually considered for a protein are its dihedral angles. Thus, moving a protein will be achieved by setting some of its dihedral angles to new values. For some applications, such as the rendering of an image of the protein and the computation of its Energy, however, the Cartesian (x,y,z) coordinates for each atom are needed. These are obtained by forward kinematics.

3.1 Mathematical Background: Matrices and Transformations

The math involved in solving forward kinematics requires some background in linear algebra, specifically in the anatomy and application of transformation matrices. The links provided in this section should provide enough mathematical background to understand the rest of this module and eventually write a simple protein manipulation program.

Background on Transformations

- **Transformation Matrices:** The main transformations you will apply to polypeptide chains will be a combination of **translations** and **rotations**. Please see introduction to translations [2] and introduction to two- and three-dimensional rotations[3] . One special rotation matrix is the Euler matrix [4] . Please pay particular attention to the different conventions used for defining the Euler matrix. The one adopted for this module is the XYZ convention (there is also the ZXZ convention). Now that you know what an Euler matrix looks like, you need to get familiar with rotations about an arbitrary vector or line. Please read more on rotations around an arbitrary vector [5] .
- **Homogeneous Transformations:** The use of homogenous coordinates and transformations can simplify some of the calculations involved in using three-dimensional transformations. In particular, they allow **translation**, which is not a linear operator in 3D, to become a linear operator in the 3D subspace (x,y,z,1) of a 4D space. The advantage of this representation is that translation becomes achievable by multiplying a vector by a matrix, and so becomes composable. A direct benefit from this is the ability to express, as a matrix, a rotation around an **arbitrary point**, not just the origin as in the pure 3D case. See homogenous transformations [6] .
- **Quaternions** Quaternions are an efficient, robust method of representing three-dimensional rotations. In particular, they are not subject to the undesirable singularities and numerical instability of rotations represented by orthonormal matrices and Euler angles. Please visit this introduction to quaternions [7] to see how they relate to homogenous transformations. In this class quaternions will be used for the optimal structural alignment of two proteins and it is recommended that the reader familiarizes him/herself with the concept of quaternions as soon as possible.

A more detailed discussion of spatial descriptions and transformations can be found in chapter 2 of [1]. The most widely used transformations to manipulate protein chains are rotations. Several representations are possible for rotations:

- **Euler angles**: The orientation of an object is given as three rotations about set axes. For example, in the ZXZ convention, the angles specify a rotation about the global z-axis, followed by one about the global x-axis, and finally, one more about the global z-axis. The use of Euler angles is subject to an undesirable phenomenon called **gimbal lock**, in which two of the rotational axes become aligned in such a way that a degree of freedom is lost.
- **Cardan angles**: The orientation is specified as a set of three rotations about axes defined by the object. The typical example is the pitch-roll-yaw set of rotations for an aircraft. Pitch corresponds to a rotation about the axis from wingtip to wingtip. Roll corresponds to a rotation about an axis from the nose to the tail, and yaw corresponds to rotation about a third "vertical" axis through the center of the plane, and roughly corresponds to a notion of horizontal heading. This method is also subject to gimbal lock.
- **Axis-angle representation**: It can be proven that any three-dimensional rotation can be represented as a single rotation about an axis, represented by a unit vector.

[2]http://mathworld.wolfram.com/Translation.html
[3]http://mathworld.wolfram.com/RotationMatrix.html
[4]http://mathworld.wolfram.com/EulerAngles.html
[5]http://mathworld.wolfram.com/EulerParameters.html
[6]http://bishopw.loni.ucla.edu/AIR5/homogenous.html
[7]http://mathworld.wolfram.com/Quaternion.html

- **Rotation matrices**: A rotation matrix is an orthonormal matrix that represents a rotation. Rotation matrices are discussed later in the module. Applying the matrix to a vector yields the rotated vector. Given two rotations represented by matrices A and B, the result of applying both rotations in sequences is given by the matrix product AB.
- **Unit quaternions**: A rotation of angle theta about the axis represented by the unit vector v = [x, y, z] is represented by a unit quaternion. Quaternions are described in this module[8].

3.2 Forward Kinematics

As stated earlier, a common operation when manipulating proteins in silico is to retrieve the Cartesian coordinates of each atom in the protein from our knowledge of its dihedral angles and rotations applied to them. For simplicity, assume we have an anchor atom and we are modeling the protein backbone only, that is, the protein consists of a serial linkage composed of consecutive backbone atoms, as shown in Figure 5.

3.2.1 A Simple Approach

The simplest way to represent a protein chain is to store the Cartesian (x,y,z) coordinates of each atom at all times. These coordinates are relative to some global coordinate frame which is unimportant, for example that in which the atomic positions were obtained by X-Ray crystallography and which are typically read from the PDB files. These coordinates can be changed if so desired. Common changes are to remove the center of mass (thus centering the protein at the global origin), subtract the position of the anchor atom (to center the protein at this atom), etc.

But it was discussed earlier that the "natural" degrees of freedom for kinematic manipulations are usually the dihedral angles alone. This means that algorithms that operate on dihedral angles to achieve their goals will normally require a way to modify the Cartesian coordinates when dihedral rotations are performed, to reflect the new atomic positions. This can be easily done with rotation matrices as follows.

[8]"Molecular Distance Measures" <http://cnx.org/content/m11608/latest/>

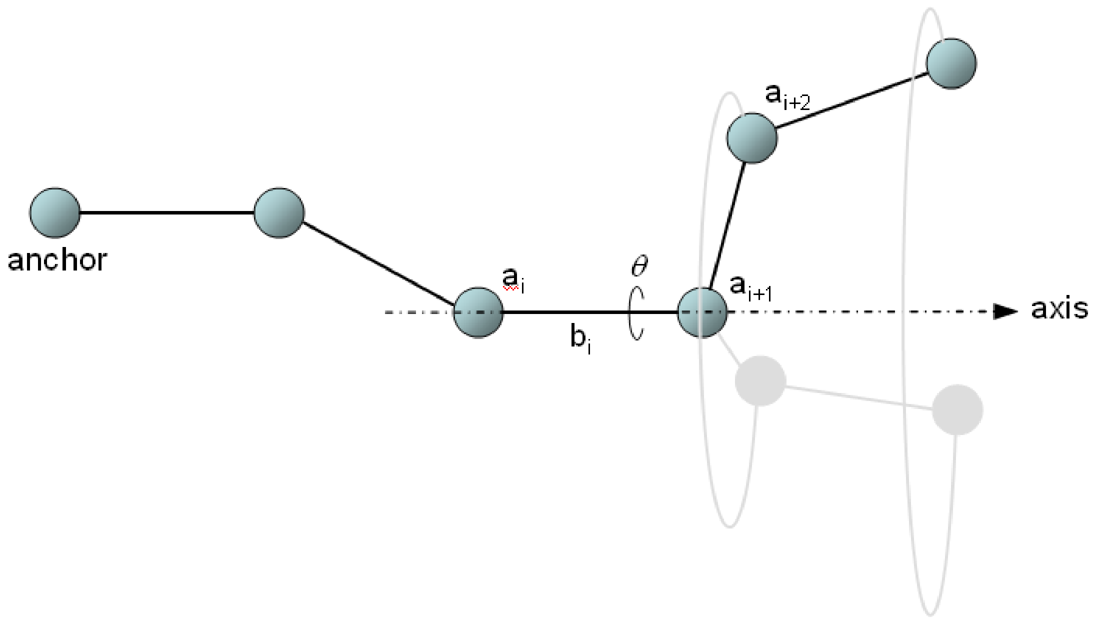


Figure 5: A protein backbone as a serial linkage.

When a rotation of θ degrees around bond i is performed, one can think of all atom positions starting at i+2 rotating around the axis defined by bond i, and all other atoms (from anchor to atom i+1 inclusive) remaining stationary. Thus, upon such a rotation, the Cartesian coordinates of the atoms after the bond need to be updated, and their new values are given by:

$$[x', y', z', 1]^T = R(i, \theta) \cdot [x, y, z, 1]^T$$

Where [x,y,z,1] is the position of a generic atom in homogeneous form, [x',y',z',1] is its position after the rotation (T is the transpose operator), and R(i,θ) is a 4x4 matrix that encodes a rotation of θ degrees around an axis coinciding with bond i that passes through atom a_i, and is given in homogeneous form as:

$$R(i, \theta) = T(\mathbf{a}_i) \cdot R_0(\mathbf{axis}, \theta) \cdot T(-\mathbf{a}_i)$$

In the above formula, T(x) is a translation by the vector x and R_0(axis,θ) is a rotation around an axis that goes through the origin of the specified coordinate system. As can be seen, this rotation around an arbitrary point is realized by translating the point to the origin, rotating the target atom around the axis through the origin, and then translating it back (the composition of these 3 transformations yields a unique 4x4 homogeneous matrix that achieves the same effect). The axis of rotation can easily be computed from the positions of atoms i and i+1 and must have unit norm. To perform successive rotations about different bonds, this procedure can be repeated, updating the Cartesian coordinates for each rotation. Note that the convention used for matrix-vector multiplication is to multiply **column** vectors by matrices on the **left**, so the rightmost transformation gets applied first, and so on. This is the convention used in most of the literature, but the alternate convention is possible (multiplying **row** vectors with matrices on the **right**; these matrices are the transpose of the column-vector convention).

Alternatively, if many rotations need to be performed at the same time (and the intermediate Cartesian coordinates are not needed), these rotations could be sorted by bond number and applied simultaneously, by noting that rotations can be performed in a cumulative way as the backbone is traversed from anchor to end atom. The ability to chain rotations around arbitrary vectors in space (i.e. not through the origin) is one of the main benefits of homogeneous transformations. For example, if two rotations need to be applied at the same time, one around bond 3 by 30 degrees and another around bond 7 by 15 degrees, the atoms between bonds 3 and 7 get updated by:

$$[x', y', z', 1]^T = R(\mathbf{bond}_3, 30) \cdot [x, y, z, 1]^T$$

But the atoms after bond 7 are updated by:

$$[x', y', z', 1]^T = R(\mathbf{bond}_7, 15) \cdot R(\mathbf{bond}_3, 30) \cdot [x, y, z, 1]^T$$

In the above, **bond n** is the unit vector defined along bond n, easily computed by subtracting the coordinates of atoms n+1 and n, and then dividing by its norm. The chaining of transformations as explained above is very useful to achieve arbitrary rotations of bonds within a protein. Sections of the protein (i.e. atoms belonging to certain residues) can be updated when a dihedral rotation is performed simply by constructing the overall matrix that should affect them.

3.2.2 Denavit-Hartenberg Local Frames

The previous approach, while simple and intuitive, has some shortcomings:

- The accumulation of math operations in the rotation matrices is prone to numerical instability. After only a couple of hundred rotations of a point, each accumulating on the other, the final position of the point may start differing significantly from its actual, intended position. As a consequence, the relative position and orientation of atoms in the protein chain will no longer be in agreement with the protein structure. In particular, bond lengths and angles will begin stretching and deviating from their physically acceptable values.
- The actual values of the Cartesian coordinates are always stored in a particular, arbitrarily chosen frame of reference. For example, if we wanted to translate the protein, we would need to modify the Cartesian coordinates stored.
- Once a rotation is applied, the method "forgets" the current values of the dihedral angles, which would need to be re-computed if needed. What is stored is a snapshot of the current Cartesian coordinates of each atom.

The original definition of Forward Kinematics, however, is a method to obtain the Cartesian coordinates of each atom from the current values of the internal degrees of freedom (dihedral angles in our case) at any time. In such an approach, the Cartesian coordinates need not be recomputed after every change in the dihedral angles; rather, the idea is to store the current values of the dihedral angles, and to have a procedure to reconstruct the atomic positions when needed. The advantages of this approach are:

- A more compact representation of the variables of the problem, since the dihedral angles require less space than the (x,y,z) coordinates of each atom (the protein topology requires the values of the bond lengths and angles anyway, so the total amount of numbers to store is comparable).

- It is not prone to numerical instability since the number of rotations performed to position an atom is always its sequence number in the chain. (Actually if the chain is thousands of residues long, some uncertainty could arise in the position of atoms far along the chain, but the relative position of consecutive atoms can still be kept under control, avoiding bond stretching).
- Performing a dihedral rotation consists simply of adding/subtracting the rotation angle from the stored value for each angle. In particular, simultaneous rotations (i.e. rotating more than one dihedral angle at a time) which consists of multiplying many 4x4 matrices in the global method, reduces to modifying the angle values.
- There is no explicit global coordinate frame for the protein. It can be positioned arbitrarily by prepending a position/orientation matrix to the forward kinematics computation.

The only preprocessing step that is necessary to start working with this method, however, is to perform an initial pass on the protein to extract the initial values of the dihedral angles and the constant bond lengths and angles, from the Cartesian coordinates available from PDB files, if the intention is to start from the protein's native state. This is easily done; bond lengths can be obtained by computing the distance between the bonded atoms, and bond angles by computing the angle between the vectors formed by two consecutive bonds (recall that the dot product of two vectors yields the product of their lengths times the cosine of the angle between them). Next, we present the transformations required for the Denavit-Hartenberg method.

Consider three consecutive bonds as in the figure below. Suppose that a local coordinate frame is attached at the beginning of each bond. For example, local coordinate system x_{i-1},y_{i-1},z_{i-1} is centered at atom A_{i-1}. Therefore, imagine that the position of each atom in three-dimensional space is specified in terms of a frame that is centered at the previous atom. Given the frames at atom A_{i-2}, and atom A_{i-1}, one can determine how the frames at atoms A_i and atom A_{i-1} will change in space as a consequence of a rotation around the bond that connects atoms A_{i-1} and A_i with the dihedral angle [2]. The correct transformation can be computed in terms of three primitive operations: two rotations and one translation. The two rotations are a rotation around the dihedral bond by the dihedral angle and a rotation around an axis perpendicular to the bond angle, by the bond angle. The translation refers to the fact that the origins of the frames are on the respective centers of the atoms connected by the bond, thus separated by bond lengths.

The Denavit-Hartenberg convention

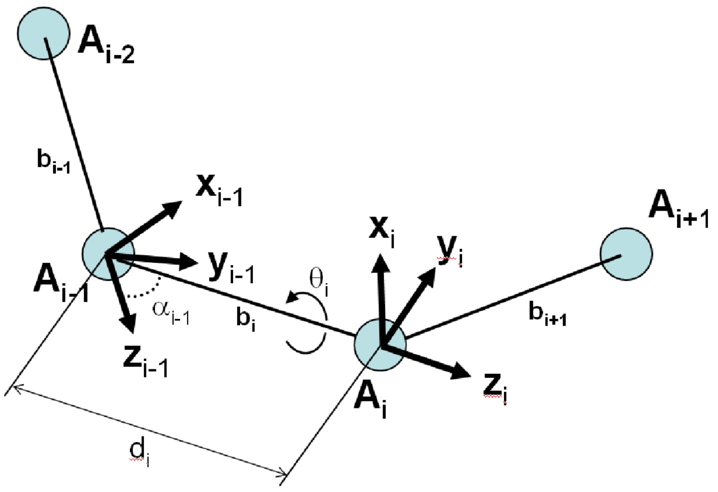


Figure 6: To describe the position of atom i in terms of the coordinate frame centered at atom i-1, two rotations and a translation are composed.

The order in which to compose these 3 transformations, to obtain the total transformation that expresses the position of atom i in terms of frame i-1, is the following: $R(\mathbf{x}, \alpha_{i-1}) \cdot R(\mathbf{z}, \theta_i) \cdot T(0,0,d_i$
where the rotation axes are the usual x (1,0,0) and z (0,0,1), not to be confused with the DH Local Frames. The resulting homogenous transformation is shown below.

Transformation

$$T = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \sin(\theta_i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 7: Homogeneous transformation to express the coordinates of atom i in terms of the frame centered at i-1

Note that θ_i is the dihedral angle on bond b_i and α_{i-1} is the bond angle between bonds b_{i-1} and b_i. d_i is the length of bond b_i. For a more detailed derivation of this transformation, please read the included material in required readings. The position of any atom in the molecule can be determined by chaining matrices of the form given above. For example, suppose that b_i, b_{i-1}, ..., b_1, represents the sequence of bonds on the path from a particular atom a to the anchor atom a_{anch}. Then, for atom a, its Cartesian coordinates with respect to the frame attached to the anchor atom is given by:

Equation 1

$$[x_i \ y_i \ z_i \ 1]^t = T_1 T_2 \ldots T_i \ [0 \ 0 \ 0 \ 1]^t$$

Figure 8: The coordinates of atom *a* with respect to the local frame attached to it are 0, 0, 0.

To complete the description, one can allow for rotations or translations of the local frame attached to the anchor atom with respect to some global frame. Rotations of the anchor atom with respect to a global frame cause a rigid rotation of the entire polypeptide chain. To do so, one can define the rotation frame as the Euler matrix defined by the Euler angles of the local frame of the anchor atom to the global frame. As discussed before, there are many conventions to define the Euler matrix. One of them, the X-Y-Z convention, defines the Euler matrix as the product of three rotation matrices: rotation around the z axis by angle α; rotation around the y axis by angle β; rotation around the x axis by the angle γ. The order of performing these three rotations in the X-Y-Z conventions is: rotation around x axis first, then around y axis second, and around z axis last. The resulting Euler matrix according to this convention is given below:

Euler Matrix

$$E = \begin{pmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma & 0 \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma & 0 \\ -s\beta & c\beta s\gamma & c\beta c\gamma & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 9: α, β, γ are the so-called Euler angles - the angles with respect to each of the Cartesian axes. The convention used here is the XYZ convention. cα and sα denote cos(α) and sin(α) respectively.

The Euler matrix can be applied last to the accumulating dihedral rotations in order to allow the anchor atom to move with respect to a global frame. For a more detailed explanation, please read the included material in required readings.

3.3

Required Reading

- Zhang-Kavraki 2002 [PDF] [9] Zhang, M. and L. E. Kavraki, "A New Method for Fast and Accurate Derivation of Molecular Conformations". Journal of Chemical Information and Computer Sciences, 42:64-70, 2002.
- Stamati-Shehu-Kavraki. Computing Forward Kinematics for Protein-like linear systems using Denavit-Hartenberg Local Frames [PDF][10]

[9] http://pubs.acs.org/cgi-bin/article.cgi/jcisd8/2002/42/i01/pdf/ci010327z.pdf
[10] http://cnx.org/content/m11621/latest/DH-deriv.pdf

Resources

1. **VMD** Visual Molecular Dynamics, is an excellent tool for visualization and scripted manipulation of protein structures that uses Tcl scripting - Humphrey, W., Dalke, A. and Schulten, K., "VMD - Visual Molecular Dynamics", J. Molec. Graphics, 1996, vol. 14, pp. 33-38.
2. **RasMol** is mostly a viewer, but has some built-in tools. - Roger Sayle and E. James Milner-White. "RasMol: Biomolecular graphics for all", Trends in Biochemical Sciences (TIBS), September 1995, Vol. 20, No. 9, p. 374.
3. **Chimera** is a very powerful visualizer that handles huge structures easily. - Pettersen, E.F., Goddard, T.D., Huang, C.C., Couch, G.S., Greenblatt, D.M., Meng, E.C., and Ferrin, T.E. "UCSF Chimera - A Visualization System for Exploratory Research and Analysis." J. Comput. Chem. 25(13):1605-1612 (2004).
4. **InsightII**, **Cerius2** and **Catalyst** are products for simulation, discovery and analysis that recently became commercial, and can be found here[11] .
5. **CHARMM** is a simulation package based on the CHARMM force field. Brooks BR, Bruccoleri RE, Olafson BD, States DJ, Swaminathan S, Karplus M (1983). "CHARMM: A program for macromolecular energy, minimization, and dynamics calculations". J Comp Chem 4: 187217.
6. **NAMD** is another popular simulation package and can be obtained here[12] . James C. Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D. Skeel, Laxmikant Kale, and Klaus Schulten. Scalable molecular dynamics with NAMD. Journal of Computational Chemistry, 26:1781-1802, 2005.
7. **Amber** is one of the most widely used molecular dynamics simulators due to its speed. Duan et al. A point-charge force field for molecular mechanics simulations of proteins based on condensed-phase quantum mechanical calculations Journal of Computational Chemistry Vol. 24, Issue 16. Pages 1999-2012 (2003).

References

[1] J. J. Craig. *Introduction to Robotics: Mechanics and Control (chapter 2).* Addison-Wesley, Reading, MA, 2003.

[2] M. Zhang and L. E. Kavraki. A new method for fast and accurate derivation of molecular conformations. *Journal of Chemical Information and Computer Sciences*, 42:64–70, 2002. http://pubs.acs.org/cgi-bin/article.cgi/jcisd8/2002/42/i01/pdf/ci010327z.pdf.

[11]http://www.accelrys.com/products/
[12]http://www.ks.uiuc.edu/Research/namd/