Computing Forward Kinematics for Protein-like linear systems using Denavit-Hartenberg Local Frames

Hernan Stamati, Amarda Shehu, Lydia Kavraki
Department of Computer Science

June 2007

Introduction

The purpose of this work is to derive a mathematical framework for manipulating protein structures in a particular coordinate set that includes the angles between the protein's bonds. The ability to predict the spatial location of every atom in the protein upon changes on its bond angles is a crucial part of any algorithmic procedure that selectively manipulates these angles. We start by introducing some key concepts such as protein structure and conformation, and the application of geometric methods to precisely define the relationship between the most useful coordinate sets that describe a protein's constituent elements. Then we focus on the derivation of an expression to convert between different coordinate representations of a protein-like linear system.

Protein structure and conformation

Two important terms to understand and differentiate in computational molecular biology are the *structure* and *conformation* of a protein.

The structure of a protein is the set of atoms and bonds that join them; in other words, its inherent connectivity. There are many possible structure models for a single protein, which usually depend on the granularity or level of detail required by the application. For example, one may consider backbone-only structures, or even simpler bead models where each aminoacid residue is modelled as a single entity (in these last two cases, the protein becomes a linear sequence of connected beads). A protein structure can also be divided into substructures, such as its backbone, sidechains, etc.

The conformation of a protein is a particular realization of its geometry, an actual arrangement in 3-dimensional space of its constituent atoms. At a more abstract level, one could also have other coordinate sets -other than the actual (x,y,z) coordinates of atoms- describing a protein conformation, such as the angles between consecutive bonds; in this case we could very well have an angular conformational space, with as many dimensions as the number of angles (consecutive bond pairs). Many other parameter spaces are possible to specify a conformation, including the protein's *internal coordinates* discussed below.

Alternative parameter spaces, other than the intuitive Cartesian coordinates, are of interest in algorithmic computational biology since:

a) Physical and/or chemical constraints on the angular displacement of bonds and their length may provide a significant decrease in the total number of independent degrees of freedom (DOFs) of the protein, greatly reducing the complexity of geometric algorithms that operate on them.

b) A reduced number of DOFs not only reduces the dimensionality of the space to explore algorithmically, it also improves the memory requirements of such algorithms (due to a more compact representation).

c) The use of certain angular measurements (such as the angle between bonds) may provide a better geometric picture that describes certain phenomena.

d) Many interesting problems in protein geometry can be readily defined in terms of internal coordinates (discussed next).

One of the most useful representations for protein conformations considers the bond lengths, bond angles, and bond torsions (otherwise known as dihedral angles) as the inherent degrees of freedom (DOFs) in such conformations. These are known as *internal coordinates*, in contrast with the Cartesian (x,y,z) coordinates of every atom in the protein. It is clear that a protein conformation can be completely specified by either its Cartesian coordinates or, alternatively, all of its internal coordinates (the relative position and orientation of the protein may be considered arbitrary if it does not interact with its environment).

We outlined above some of the most important reasons for working with internal coordinates. In later sections, we will be interested in the problem of recomputing all Cartesian (x,y,z) atom positions when changes in internal coordinates are introduced. Why would we be interested in this conversion? Most real applications require such conversion because:

a) A protein's POTENTIAL ENERGY (a quantity that indicates the feasibility of a particular conformation) is generally defined in Cartesian coordinates.

b) Energy minimization algorithms rely heavily on the efficient computation of the potential energy, and usually work in Cartesian space.

c) A small change in one of the angular DOFs may actually produce important deviations in the Cartesian coordinates of atoms further along the protein.

d) Protein similarity measures, such as RMSD, are defined precisely for Cartesian coordinates.

e) Visualization programs always convert to cartesian coordinates before displaying a protein on the screen.

Since we have established that both coordinate representations of a protein -cartesian and internal- are equally useful, we need, at the very least, a procedure to obtain the former in terms of the latter.

Forward Kinematics

Kinematics is the description of motion, disregarding the ultimate physical interactions that produce motion (for example, kinematics can be used to describe the trajectory of a falling object, without the need to introduce the concept of gravity). For a protein, we are interested in knowing what conformations are ”allowable” while respecting the protein’s structure, without the need to resort to the covalent binding forces to explain HOW the structure is kept together. This definition suggests that the study of kinematics is a purely geometrical problem.

The term *forward kinematics* was introduced in the study of robotics to denote the computation of the cartesian (x,y,z) coordinates of the robot’s *end-effector* (see figure), or any of its intermediate joints, based on the robot’s internal coordinates (the joint angles and the link lengths). This procedure is of special interest to us since it embodies the conversion mentioned in the previous section; we will derive a mathematical expression to solve the forward kinematics problem for a simple linear protein model in a later section. For now, you should convince yourself that a kinematic protein model is no different from a general articulated mechanism or robot.

Inverse Kinematics

Imagine you are sitting at your table and you want to grasp a glass of water that is standing on the table. You will, of course, automatically modify the angles at your shoulder, elbow and wrist to position your hand around the glass. You will do so by taking into account the lengths of your arm and forearm, and the effect of different angles at your joints, until your visual senses inform you that your hand is close enough to the glass. This is an instance of an inverse kinematics problem: given a target position of the end-effector, what are the (possibly many) values of the internal coordinates that satisfy it? One particularly useful instance of the inverse kinematics problem for proteins is the *loop closure* problem: given a flexible section of a protein (commonly referred to as a loop), are there any values for its internal coordinates that ensure the loop endpoints connect two other protein domains? Problems such as this usually require the definition and understanding of the protein’s forward kinematics first, for which we develop a model next.

Performing Forward Kinematics on a protein model. The Denavit-Hartenberg local frames

To perform forward kinematics on a protein model (as on any articulate mechanism or robot) requires the INTEGRATION of a series of *local constraints*, starting from an *initial condition* (the absolute location of the protein’s ”origin” or ”base”) as in any integration problem, either discrete or differential. In our case, we have a discrete problem characterized by:

1. The specification of an *anchor* atom. This will typically be at an endpoint along the protein, and will serve as the boundary condition on which to propagate the following placement constraints.

2. The description of an atom's location, relative to a neighboring atom, in terms of the local internal coordinates for the bond that joins them.

It should be clear that these two points define a recursive discrete integration problem on the internal coordinates. The first point is trivially solvable (simply specify the cartesian coordinates of the anchor atom). We proceed next to define the mathematical characterization of the second point.

In order to describe the location of an atom relative to its neighbors, we define a coordinate system, or *frame*, attached to each atom, following the *Denavit-Hartenberg (DH)* method.

In the DH local frames method, there is a local coordinate frame attached to every atom A_i (see figure 1). The idea is to express the cartesian coordinates of atom A_i with respect to the coordinate frame attached to atom A_{i-1}; this will serve as the definition of our recursion step in the integration procedure outlined above, and obviously this conversion will involve bond parameters (i.e. angles and bond length). Once we have such a coordinate-frame conversion, we can propagate it towards the anchor atom A_0, for which we know its exact position. We will adopt the following convention to define local frames: frame $F_{i-1} = \{A_{i-1}, \mathbf{x_{i-1}}, \mathbf{y_{i-1}}, \mathbf{z_{i-1}}\}$, where A_{i-1} indicates the origin of the frame, and $\mathbf{x}, \mathbf{y}, \mathbf{z}$ the three orthogonal axes. Similarly, the local frame F_i attached to atom A_i is defined as $F_i = \{A_i, \mathbf{x_i}, \mathbf{y_i}, \mathbf{z_i}\}$. By convention axis $\mathbf{z_i}$ lies along bond b_i in the direction from A_{i-1} to A_i; this will simplify the math shortly. Axis $\mathbf{x_{i-1}}$ is chosen perpendicular to both $\mathbf{z_{i-1}}$ and $\mathbf{z_i}$ (so that the bond angle α_i can be defined around it) and can be obtained by their cross-product as in $\mathbf{x_i} = \mathbf{z_{i-1}} \times \mathbf{z_i}$. Finally, to complete a coordinate frame, $\mathbf{y_i}$ is perpendicular to both $\mathbf{x_i}$ and $\mathbf{z_i}$ and can be obtained by their cross-product $\mathbf{y_i} = \mathbf{x_i} \times \mathbf{z_i}$ (in that order) to define a right-handed coordinate system. Keep in mind that this construction is only theoretical and does not actually need to be performed in a program; its sole purpose is to obtain a mathematical formula for the conversion, as will be shown.

As depicted in Figure 1, the distance between atom A_{i-1} and atom A_i, the *bond length* for bond b_i, is denoted by d_i. The angle between $\mathbf{z_{i-1}}$ and $\mathbf{z_i}$, the *bond angle*, is denoted by α_{i-1}. Finally, the *dihedral angle* on bond b_i is denoted by θ_i. The formal definition of the dihedral angle θ_i is the angle between the plane defined by atoms A_{i-2}, A_{i-1} and A_i and the plane defined by atoms A_{i-1}, A_i and A_{i+1}. Intuitively, θ_i defines the *torsion* of bonds b_{i-1} and b_{i+1} around bond b_i.

For the purpose of deriving one coordinate frame conversion, assume P is an arbitrary point in space, with coordinates (x, y, z) in frame F_i. We want to express these coordinates with respect to frame F_{i-1} (to obtain our recursion formula). Then,

$$(x', y', z', 1)^t = R_i \cdot (x, y, z, 1)^t$$

where (x', y', z') are the coordinates of P with respect to frame F_{i-1}, and R_i is the matrix that transforms frame F_{i-1} to frame F_i. During the following discussion, you may want to keep inspecting figure 1 to validate the results (this may require several reads of this section).

To obtain the above frame conversion, one approach is to apply succesive transformations to F_{i-1} until it coincides with F_i (the reader should convice him/herself that this is equivalent to obtaining the transformations that convert the coordinates of P from F_i to F_{i-1}; try creating some simple examples, by choosing simple P's, and following the conversion of coordinates for this point). First we need to remove the displacement between the origins of the frames, and then we need to align the frames,

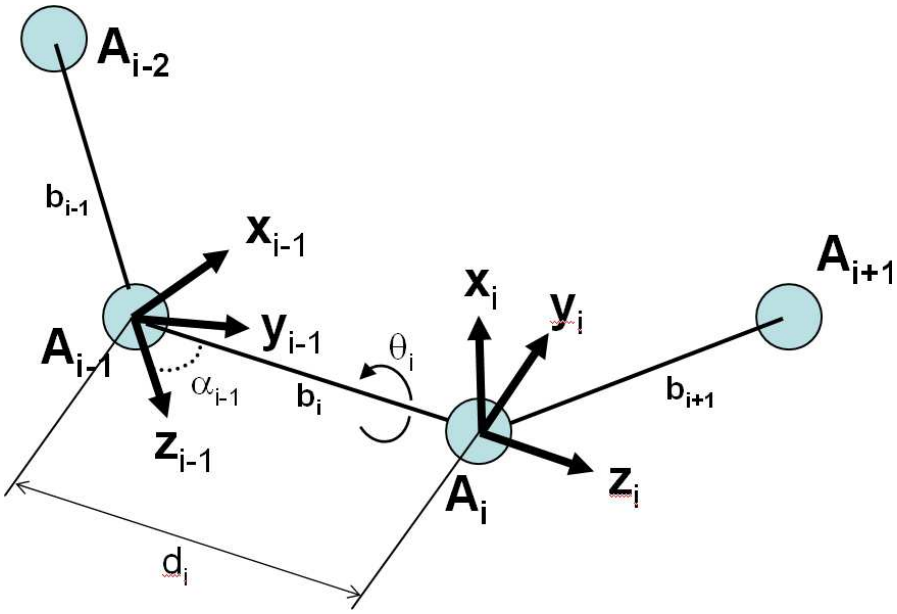

Figure 1: Local frames F_{i-1} and F_i attached to atoms A_{i-1} and A_i, respectively.

so that their axes coincide. To remove the displacement between the origins of the frames, one needs to translate frame F_{i-1} along z_i by the distance between the two frame origins, d_i. This translation can be expressed by the following matrix in homogeneous coordinates (recall that in homogeneous coordinates, translation, which is not a linear transformation, can be composed with other linear transformations by simple matrix multiplication):

$$T_{z_i}(d_i) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1}$$

After removing the displacement, one needs to align the two frames to make their axes coincide. Axis $\mathbf{x_{i-1}}$ does not coincide with axis $\mathbf{x_i}$ due to the dihedral angle θ_i between them. Axis $\mathbf{z_{i-1}}$ does not coincide with axis $\mathbf{z_i}$ due to the bond angle α_{i-1} between them. We first align $\mathbf{x_{i-1}}$ with $\mathbf{x_i}$, and then align $\mathbf{z_{i-1}}$ with $\mathbf{z_i}$.

To align $\mathbf{x_{i-1}}$ with $\mathbf{x_i}$, one needs to perform a rotation by the dihedral angle θ_i around axis $\mathbf{z_i}$. This rotation is given by the following matrix:

$$R_{z_i}(\theta_i) = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2}$$

To align $\mathbf{z_{i-1}}$ with $\mathbf{z_i}$, one can perform the rotation by bond angle α_{i-1} around axis $\mathbf{x_i}$. This rotation is given by the following matrix:

$$R_{x_i}(\alpha_{i-1}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & 0 \\ 0 & \sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

Note that the order of rotations is important. One cannot perform the rotation of frame F_{i-1} by α_{i-1} first and then the dihedral rotation, because a rotation by α_{i-1} changes the axis x_{i-1} to x'_{i-1}. The value for the dihedral θ_i is defined with respect to x_{i-1} and x_i, not $x_{i-1}{}'$ and x_i. To do the rotation by the dihedral second, you would need to update the definition of θ_i with respect to x'_{i-1} since the axis x_{i-1} changed. This is not convenient since for every transformation you would need to recompute the dihedral angle; this is why a rotation by the bond angle first and the dihedral angle second is necessary.

Now we can finally transform F_{i-1} to F_i by combining the above matrices as follows:

$$R_i = R_{x_i}(\alpha_{i-1}) \cdot R_{z_i}(\theta_i) \cdot T_{z_i}(d_i)$$

where:

$$R_i = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \sin(\theta_i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4}$$

This formula provides all we need to integrate the atom coordinates from any starting point. As we said before, a generic step in the integration is to get the cartesian coordinates (x, y, z) of any atom A_i with respect to frame F_{i-1} as follows:

$$(x, y, z, 1)^t = R_i \cdot (0, 0, 0, 1)^t$$

Recall that the coordinates of atom A_i in its own frame F_i are $(0, 0, 0)$ since we defined each local frame as resting on the atom centers, so the origin of frame F_i *is* the atom A_i.

When using internal coordinates, the position of atom A_0 (the anchor atom) is the only one known, so every other atom's cartesian coordinates can be obtained by chaining matrices R_i. Since the local coordinates of atom A_i in its own frame are $(0, 0, 0)$ (each local frame is attached to the center of an atom), its coordinates (x, y, z) with respect to the frame attached to the anchor atom are computed as follows:

$$(x, y, z, 1)^t = R_1 \cdot R_2 \cdot \ldots \cdot R_i \cdot (0, 0, 0, 1)^t$$

Thus, for any value of i, we can retrieve A_i's position in terms of A_0 and all the bond parameters between them (encoded in all the matrices). The only thing left to specify to globally place our protein is its *orientation*. 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.

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 $\mathbf{z}$ by angle α; rotation around $\mathbf{y}$ by the angle β; rotation around $\mathbf{x}$ by the angle γ. In the X-Y-Z convention, the order is as follows:

$$E = R_{\mathbf{z}}(\alpha) \cdot R_{\mathbf{y}}(\beta) \cdot R_{\mathbf{x}}(\gamma)$$

This definition yields as the Euler matrix the one below:

$$E = \begin{pmatrix} \cos(\alpha)\cos(\beta) & \cos(\alpha)\sin(\beta)\sin(\gamma) - \sin(\alpha)\cos(\gamma) & \cos(\alpha)\sin(\beta)\cos(\gamma) + \sin(\alpha)\sin(\gamma) & 0 \\ \sin(\alpha)\cos(\beta) & \sin(\alpha)\sin(\beta)\sin(\gamma) + \cos(\alpha)\cos(\gamma) & \sin(\alpha)\sin(\beta)\cos(\gamma) - \cos(\alpha)sin(\gamma) & 0 \\ -\sin(\beta) & \cos(\beta)\sin(\gamma) & \cos(\beta)\cos(\gamma) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (5)$$

Propagation of the rotation by the Euler matrix to the entire polypeptide chain can be done as follows:

$$(x, y, z, 1)^t = E \cdot R_1 \cdot R_2 \cdot \ldots \cdot R_i \cdot (x_0, y_0, z_0, 1)^t$$

Remember that all the protein's internal coordinates (bond angles, bond lengths and dihedral angles) are encoded in the R_i matrices, so this formula provides the conversion between internal and cartesian coordinates we set out to find.

For more information on the original application of Denavit-Hartenberg local frames to robotic mechanisms, see [1].

[1] J. Craig. *Introduction to robotics: manipulation and control.* Addison-Wesley, 2 edition, 1989.