CS612 - Algorithms in Bioinformatics

Fall 2017 – Structural Manipulation

November 22, 2017
Emergence of large structural databases which do not allow manual (visual) analysis and require efficient 3-D search and classification methods.

Structure is much better preserved than sequence – proteins may have similar structures but dissimilar sequences.

Structural motifs may predict similar biological function

Getting insight into protein folding. Recovering the limited (?) number of protein folds.

Comparing proteins of not necessarily the same family.
Implementing structural algorithms (folding, docking, alignment) requires geometric manipulation of protein structures.

A 3-D protein structure is represented as a set of $x, y, z$ coordinates (vectors).

Structural manipulation is done via geometric transformations (translation, rotation) of some or all the coordinates.

Transformations can be represented using matrices applied on the coordinate vectors.
Definition (Degrees of Freedom)

The degree of freedom (DOF) is the set of independent parameters that can be varied to define the state of the system.

Examples:

- The location of a point in a 2-D cartesian system has two independent parameters – its \((x, y)\) coordinates.
- An alternative representation – \((r, \theta)\), distance from the origin and rotation about the origin, respectively.
- A molecule with \(n\) atoms can be represented by a set of \(3 \times N\) cartesian coordinates, so it has \(3 \times N\) DOFs...
- ...or does it?
The atoms are mutually restricted by bond lengths and angles.
Each such bond/angle poses a constraint on the system.
We effectively assume that bond lengths and planar angles don’t change.
For example – if we know that the bond length (distance) between two atoms is 1.5Å, then they are no longer independent!
So the “real” number of DOFs is much smaller than $3 \times N$. 
When trying to manipulate the structure internal coordinates may be easier to work with.

The internal coordinates represent bond length, angles and dihedrals.

Remember that we treat bond lengths and planar angles as fixed, but we still need them.

They help us infer the connectivity of the structure and switch between representations.
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.
As a reminder – there are two freely rotatable backbone dihedral angles per amino acid residue in the protein chain: $\phi$ is a consequence of the rotation about the bond between N and $C\alpha$, and $\psi$, which is a consequence of the rotation about the bond between $C\alpha$ 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 (except the first and last), a total of 2N-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 side chain, to 5 in the case of arginine.
One can generate different three dimensional structures of the same protein by varying the dihedral angles.

There are $2N-2$ 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 (why?)
An example of a Z-matrix representing the internal coordinates of methane (CH4)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>1.089</th>
<th>2</th>
<th>109.471</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>1</td>
<td>1.089</td>
<td>2</td>
<td>109.471</td>
</tr>
<tr>
<td>H</td>
<td>1</td>
<td>1.089</td>
<td>2</td>
<td>109.471</td>
</tr>
<tr>
<td>H</td>
<td>1</td>
<td>1.089</td>
<td>2</td>
<td>109.471</td>
</tr>
</tbody>
</table>
The Cartesian coordinate representation of Methane (CH4)

<table>
<thead>
<tr>
<th>Atom</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>H</td>
<td>0.000</td>
<td>0.000</td>
<td>1.089</td>
</tr>
<tr>
<td>H</td>
<td>1.027</td>
<td>0.000</td>
<td>-0.363</td>
</tr>
<tr>
<td>H</td>
<td>-0.513</td>
<td>-0.889</td>
<td>-0.363</td>
</tr>
<tr>
<td>H</td>
<td>-0.513</td>
<td>0.889</td>
<td>-0.363</td>
</tr>
</tbody>
</table>

We can switch back and forth between different representations, up to an arbitrary rigid transformation (absolute position and orientation in space). To move from internal to cartesian coordinates we need the first three atoms, $a$, $b$, $c$. 
The first atom, $a$, represents the origin of the coordinate systems. Set its three cartesian coordinates to zero.

The second atom, $b$, is at a fixed distance from the first one, which is their bond distance. Fix the $z$ axis as the axis lying on the bond between the two atoms. $b$'s $z$ value is the distance and its $x$ and $y$ coordinates are set to zero.

The third atom, $c$, makes an angle with the other two and a bond with the second atom.

We can define the $x−z$ plane as the plane defined by the C atom and the first two hydrogens (every three non-collinear points define a plane).

These two constraints and set the $y$ coordinate to zero. Let $r_{b,c}$ be the distance between atoms $b$ and $c$. 
The $x, z$ coordinates can be inferred by converting from polar to cartesian coordinates using the following formula:

$$x = r_{bc} \times \cos(180 - \theta) = -r_{bc} \cos(\theta)$$

$$z = \sin(180 - \theta) = r_{bc} \sin(\theta)$$

Now that we have the $x, z$ plane defined, the $y$ axis can be extracted by a cross product, for example, between the two vectors $||b - a||$ and $||c - b||$, after normalization.
If H3 were on the $x - z$ plane, it would make a $109.471^\circ$ angle with C-H1 in the opposite direction, so its projection on the $x - z$ axis would be $\{1.027, 0.000, -0.363\}$. However, it has a dihedral angle of $120^\circ$ with the C-H1-H2 plane, so we should rotate it by $60^\circ$ around the $z$ axis. This would leave its $z$ coordinate unchanged, and its $x, y$ values would be as follows:

\[
x = -1.027 \times \cos(60) = -0.513 \\
y = -1.027 \times \sin(60) = -0.889
\]

Similarly, H4 creates a dihedral angle of $-120^\circ$ with the C-H1-H2 plane, so we should rotate it by $-60^\circ$ around the $z$ axis:

\[
x = -1.027 \times \cos(-60) = -0.513 \\
y = -1.027 \times \sin(-60) = 0.889
\]
Re-orientating the molecule leads to Cartesian coordinates that make the symmetry more obvious:

<table>
<thead>
<tr>
<th>Atom</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>H</td>
<td>0.629</td>
<td>0.629</td>
<td>0.629</td>
</tr>
<tr>
<td>H</td>
<td>-0.629</td>
<td>-0.629</td>
<td>0.629</td>
</tr>
<tr>
<td>H</td>
<td>-0.629</td>
<td>0.629</td>
<td>-0.629</td>
</tr>
<tr>
<td>H</td>
<td>0.629</td>
<td>-0.629</td>
<td>-0.629</td>
</tr>
</tbody>
</table>
Manipulation of Molecular Structures

When generating new conformations by manipulating the dihedral angles, we 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.

Therefore we will probably need to keep the two representations simultaneously.
An $N \times N$ matrix $R$ is a rotation matrix in dimension $N$ if it is orthonormal (its columns are pairwise orthogonal and normalized) and $det(R) = 1$.

Such matrix has the property $R^T = R^{-1}$.

A vector $t = \{t_1, t_2, ..., t_N\}$ is a translation vector in dimension $N$.

A rigid transformation on a vector $v$ (rotation + translation) has the form $Rv + t$.

Rotation matrices are a group under matrix multiplication.
Rotation Matrices and Groups

- Group – A set G with an operation defined on it.
- 4 defining axioms:
  1. Closure: \( \forall a, b \in G, a \circ b \in G \)
  2. Associativity: \( \forall a, b, c \in G, (a \circ b) \circ c = a \circ (b \circ c) \)
  3. Identity: \( \exists e \in G \text{s.t} \forall a \in G, a \circ e = e \circ a = a \)
  4. Inverse: \( \forall a \in G \exists a^{-1} \in G \text{s.t} \forall a \in G, a \circ e = e \circ a = a \)
- Rotation matrices are groups under matrix multiplication.
  1. Closure: The multiplication of every two rotation matrices is a rotation matrix.
  2. Associativity: True for every matrix.
  3. Identity: The identity matrix, which is a rotation by 0 degrees.
  4. Inverse: Rotation in the inverse direction.
Objects undergo transformations in space – translation, rotation in 2D or 3D.

Matrices can encode transformations.

Translation vectors, rotation matrices.

Example – Rotation in 2D:

\[
A = \begin{bmatrix}
\cos(\theta) & -\sin(\theta) \\
\sin(\theta) & \cos(\theta)
\end{bmatrix}
\]

\(\theta\) is the rotation angle in the 2D plane.
Example – Rotation in 2D
Matrix multiplication $\leftrightarrow$ unit complex multiplication.

$$R(\Theta_1) \cdot R(\Theta_2) \downarrow e^{i\Theta_1} \cdot e^{i\Theta_2} \rightarrow e^{i(\Theta_1 + \Theta_2)}$$

$$a + bi \leftrightarrow \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \sim S^1$$
Combining rigid body translations and rotations.

SE(n) – special Euclidean groups:

\[ SE(n) = \begin{bmatrix} R & V \\ 0 & 1 \end{bmatrix} \]

Where R is a rotation matrix and V is a translation vector.
Representing Rotations in 2D – SO(2)

$$R = \begin{bmatrix}
\cos(\theta) & -\sin(\theta) \\
\sin(\theta) & \cos(\theta)
\end{bmatrix}$$
**Representing Rotations in 3D**

- $SO(3)$ – special orthogonal group in 3D, rigid body rotation in 3D.
- Not a simple extension of 2D rotation.
- How many degrees of freedom are there in $SO3$?
- $O(3) \rightarrow AA^T = 1$.
- 3 constraints on unit row vectors, 3 constraints on orthogonality.
- $SO(3) \rightarrow det(A) = 1$. 
Representing Rotations in 3D

- 3x3 matrix
- Euler angles (phi,theta,psi)
- Yaw, pitch, roll angles
- Axis-angle representation
- Quaternions
$3 \times 3$ Matrix

$R = \begin{bmatrix}
\tilde{x}_1 & \tilde{y}_1 & \tilde{z}_1 \\
\tilde{x}_2 & \tilde{y}_2 & \tilde{z}_2 \\
\tilde{x}_3 & \tilde{y}_3 & \tilde{z}_3 \\
\end{bmatrix} = \begin{bmatrix}
R_{11} & R_{12} & R_{13} \\
R_{21} & R_{22} & R_{23} \\
R_{31} & R_{32} & R_{33} \\
\end{bmatrix} \in SO(3)$
Performing a Rotation on a Vector

\[
\begin{bmatrix}
R_{11} & R_{12} & R_{13} \\
R_{21} & R_{22} & R_{23} \\
R_{31} & R_{32} & R_{33}
\end{bmatrix}
\begin{bmatrix}
v_x \\
v_y \\
v_z
\end{bmatrix}
= 
\begin{bmatrix}
R_{11}v_x + R_{12}v_y + R_{13}v_z \\
R_{21}v_x + R_{22}v_y + R_{23}v_z \\
R_{31}v_x + R_{32}v_y + R_{33}v_z
\end{bmatrix}
\]
CCW Rotations Around Axes

- To combine rotations around axes multiply matrices
- Notice that matrix multiplication is not commutative (order matters!)
- Look at order from right to left. For example – 
\[ R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \gamma & \sin \gamma \\ 0 & -\sin \gamma & \cos \gamma \end{bmatrix} \]
\[ R_y(\beta) = \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix} \]
\[ R_z(\alpha) = \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \]
Rotation by 30 degrees around $z$

Rotation Sequence – $xzy$

Rotation by 45 degrees around $y$

Rotation sequence – $xyz$

Rotation by -15 degrees around $x$

Rotation Sequence – $yzx$
Roll, Pitch, Yaw

The rotation matrix $R(\alpha, \beta, \gamma)$ is given by:

$$R(\alpha, \beta, \gamma) = \begin{bmatrix}
\cos \alpha \cos \beta & \cos \alpha \sin \beta \sin \gamma - \sin \alpha \cos \gamma & \cos \alpha \sin \beta \cos \gamma + \sin \alpha \sin \gamma \\
\sin \alpha \cos \beta & \sin \alpha \sin \beta \sin \gamma + \cos \alpha \cos \gamma & \sin \alpha \sin \beta \cos \gamma \cos \alpha \sin \gamma \\
-\sin \beta & \cos \beta \sin \gamma & \cos \beta \cos \gamma
\end{bmatrix}$$
$R = \begin{bmatrix} \cos \gamma & \sin \gamma & 0 \\ -\sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \beta & \sin \beta \\ 0 & -\sin \beta & \cos \beta \end{bmatrix} \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Euler Angles – Example: $-60^\circ, 30^\circ, 45^\circ$
Two major problems with yaw, pitch, roll and Euler angles:

- Cases where a continuum of values yield the same rotation matrix (no unique solution in certain cases).
- Cases where non-zero angles yield the identity rotation matrix which is equivalent to zero angles.
Gimbal Lock

Example – when $\beta = 0$, Euler angle representation becomes:

$$R = \begin{bmatrix}
\cos \alpha & -\sin \alpha & 0 \\
\sin \alpha & \cos \alpha & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\cos \gamma & -\sin \gamma & 0 \\
\sin \gamma & \cos \gamma & 0 \\
0 & 0 & 1
\end{bmatrix}

= \begin{bmatrix}
\cos(\alpha + \gamma) & -\sin(\alpha + \gamma) & 0 \\
\sin(\alpha + \gamma) & \cos(\alpha + \gamma) & 0 \\
0 & 0 & 1
\end{bmatrix}$$
Axis-Angle Representations

\[ \theta \circ \mathbf{v} \]
Quaternions are an extension of complex numbers.

\[ h = a + bi + cj + dk, \ a,b,c,d \ \text{real numbers.} \]

\( i,j,k \) : imaginary components s.t.:

- \( i^2 = j^2 = k^2 = -1 \)
- \( ij = k, \ jk = i, \ ki = j \)
- \( ij = -ji, \ jk = -kj, \ ki = -ik \)

Magnitude of a quaternion: \( ||h|| = \sqrt{a^2 + b^2 + c^2 + d^2} \)

a unit quaternion: \( ||h|| = 1 \)
Axis-Angle Representations Through Quaternions

\[ h = \cos \frac{\theta}{2} + (v_1 \sin \frac{\theta}{2})i + (v_2 \sin \frac{\theta}{2})j + (v_3 \sin \frac{\theta}{2})k \]

\[ h = \cos \frac{\theta}{2} + v \sin \frac{\theta}{2} \]

\[ -h = -\cos \frac{\theta}{2} - v \sin \frac{\theta}{2} \]

We assume that \( v \) is a unit vector!
Operations on Quaternions – Multiplication

- Given two quaternions – \( h_1 = a_1 + ib_1 + jc_1 + kd_1 \), \( h_2 = a_2 + ib_2 + jc_2 + kd_2 \)
- Assume \( v = [b, c, d] \), like a 3-D vector.
- \( h_1 \cdot h_2 = (a_1 * a_2 - v_1 \cdot v_2, a_1 v_2 + a_2 v_1 + v_1 \times v_2) \)
- \( v_1 \cdot v_2 \) is the dot product of \( v_1 \) and \( v_2 \), \( v_1 \times v_2 \) is the cross product.
- \( h_1 \cdot h_2 = a_3 + ib_3 + jc_3 + kd_3 \) Where:
  \[
  a_3 = a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2 \\
  b_3 = a_1 b_2 + a_2 b_1 + c_1 d_2 - c_2 d_1 \\
  c_3 = a_1 c_2 + a_2 c_1 + b_2 d_1 - b_1 d_2 \\
  d_3 = a_1 d_2 + a_2 d_1 + b_1 c_2 - b_2 c_1
  \]
Given a unit quaternion \( h = a + bi + cj + dk \), define its conjugate quaternion \( h^* = a - bi - cj - dk \):

- Transform point \( p(x, y, z) \) by sandwiching: \( h \cdot p \cdot h^* \)
- Treat \( p \) as a quaternion with no real component (\( a=0 \)).
- The rotated point \( p'(x', y', z') \) is obtained by the \( i,j,k \) components of the result
- To multiply a vector and a quaternion, see matrix representation above.
- Don’t forget to translate the vector to the origin and translate back.
Lemma: \((pq)^* = q^* p^*\).

Sandwiching: \(S_h(v) = h \cdot v \cdot h^*\)

\((S_{h_1} \circ S_{h_2})(v) = S_{h_1}(S_{h_2}(v)) = S_{h_1}(h_2v h_2^*) = h_1(h_2v h_2^*)h_1^* = (h_1 h_2)v(h_2^* h_1^*) = S_{h_1 h_2}(v)\)
### Operations on Quaternions – Useful Examples

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Identity, no rotation</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>180° turn around X axis</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>180° turn around Y axis</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>180° turn around Z axis</td>
</tr>
<tr>
<td>$\sqrt{0.5}$</td>
<td>$\sqrt{0.5}$</td>
<td>0</td>
<td>0</td>
<td>90° rotation around X axis</td>
</tr>
<tr>
<td>$\sqrt{0.5}$</td>
<td>0</td>
<td>$\sqrt{0.5}$</td>
<td>0</td>
<td>90° rotation around Y axis</td>
</tr>
<tr>
<td>$\sqrt{0.5}$</td>
<td>0</td>
<td>0</td>
<td>$\sqrt{0.5}$</td>
<td>90° rotation around Z axis</td>
</tr>
<tr>
<td>$\sqrt{0.5}$</td>
<td>$-\sqrt{0.5}$</td>
<td>0</td>
<td>0</td>
<td>-90° rotation around X axis</td>
</tr>
<tr>
<td>$\sqrt{0.5}$</td>
<td>0</td>
<td>$-\sqrt{0.5}$</td>
<td>0</td>
<td>-90° rotation around Y axis</td>
</tr>
<tr>
<td>$\sqrt{0.5}$</td>
<td>0</td>
<td>0</td>
<td>$-\sqrt{0.5}$</td>
<td>-90° rotation around Z axis</td>
</tr>
</tbody>
</table>

Example – Rotation by 90° around Y axis

- \( \mathbf{v} = [0, 1, 0] \) (the rotation axis, which is the Y axis).
- \( \theta = 90° \).
- \( h = \cos \frac{\theta}{2} + (v_1 \sin \frac{\theta}{2})i + (v_2 \sin \frac{\theta}{2})j + (v_3 \sin \frac{\theta}{2})k = \sqrt{0.5} + 0 \cdot i + \sqrt{0.5} \cdot j + 0 \cdot k \)
- \( h = \sqrt{0.5} + \sqrt{0.5} \cdot j \)
- \( h^* = \sqrt{0.5} - \sqrt{0.5} \cdot j \)
- Say \( \mathbf{p} = [1, 2, 3] = 1 \cdot i + 2 \cdot j + 3 \cdot k \)
- Transforming \( \mathbf{p} \):
  \[ p' = h \cdot p \cdot h^* = (\sqrt{0.5} + \sqrt{0.5} \cdot j) \cdot (i + 2 \cdot j + 3 \cdot k) \cdot (\sqrt{0.5} - \sqrt{0.5} \cdot j) \]
Example – Rotation by 90° around Y axis

- Transforming p:
  \[ p' = h \cdot p \cdot h^* = (\sqrt{0.5} + \sqrt{0.5}i \cdot j) \cdot (i + 2j + 3k) \cdot (\sqrt{0.5} - \sqrt{0.5}i \cdot j) \]
- \[ h_1 \cdot h_2 = (a_1 \cdot a_2 - v_1 \cdot v_2, a_1 v_2 + a_2 v_1 + v_1 \times v_2) \]
- \[ p \cdot h^* = -[1, 2, 3] \cdot [0, -\sqrt{0.5}, 0], \sqrt{0.5} \cdot [1, 2, 3] + [1, 2, 3] \times [0, -\sqrt{0.5}, 0] \ldots \]
- \[ h \cdot p \cdot h^* = 0, 3, 2, -1 \]
Quaternions Vs. Matrices

- A quaternion needs 4 doubles instead of 9
- Sandwiching takes 28 multiplications while matrices need 9
- Composing rotations takes 16 multiplications with quaternions and 27 for matrices
- When composing matrices, numerical inaccuracies lead to distortions. Vectors are no longer orthonormal and angles are distorted.
- Quaternions do not distort angles and renormalization is just a division by the quaternion magnitude: $q = q/|q|$
- In interpolation with matrices $R(t) = (1 - t)R0 + tR1$, $R(t)$ does not represent a rotation.
- With $q(t) = (1 - t)q0 + tq1$, $q(t)/|q(t)|$ is a valid rotation
Some Resources