REPORT

SmartPATCH

A PROTEIN PATCHING AND CLASH REMOVAL PROGRAM

Made by: Kanishk Asthana. 4th Semester, B.E. Biotechnology. Delhi College of Engineering, Bawana Road, Shahbad Daulatpur, New Delhi 110042.
Email: kanishk_asthana@yahoo.com
Roll no. 2K8/BT/10
Summer Project: Summer 2010. SCFBIO IIT Delhi, Hauz Khas, New Delhi 110016
Ms. Priyanka Dhingra (PhD Student): Ms. Dhingra played a pivotal role in development and testing of this algorithm. I wish to thank her ,for her contribution.
Email: priyanka@scfbio-iitd.res.in
Working under Prof. B. Jayaram.
Email: bjayaram@scfbio-iitd.res.in

1. ABSTRACT: SmartPATCH was developed to patch protein fragments generated by BHAGEERATH (protein structure prediction software developed at SCFBIO). The main aim of the program is to assemble the predicted protein. Two fragments are taken at a time and patched together, in sequence. Patching is followed by clash reduction.

Clash reduction is achieved by rotating, bending and swinging the second fragment to a suitable orientation with respect to the first fragment. The patched structure then undergoes a process called minimization.

In this process, the energy of the patched structure is minimized by changing the bond lengths, bond angles and the general orientation of the protein itself, to obtain the final structure. Patching is a requirement as; BHAGEERATH predicts smaller structures better than larger ones. In theory predicting smaller structures and then joining them together should give a better structure for a larger protein, rather than predicting the entire structure in one go.

Minimization is computationally a very expensive process, thus it is unable to check for structures with least energy beyond a certain range. This made it necessary for the patching program to take care of both patching and orientation of the two fragments under consideration.

2. THEORY:

It is suggested that the reader go through Chapter 4 (Kinematics of Rigid Body motion), of Classical Mechanics (2nd edition) by Herbert Goldstein, before reading the following sections.

2.1 ROTATION OF RIGID BODY ABOUT A FIXED AXIS

Rotation of a rigid body about the X, Y, or Z axis in three dimensional Cartesian space can be described by the following rotation matrices or their inverse.

REPORT

$$\mathbf{A}=\begin{bmatrix}1 & 0 & 0\\ 0 & cos\alpha & sin\alpha\\ 0 & -sin\alpha & cos\alpha\end{bmatrix}$$ for rotation about X axis.

$$\mathbf{B}=\begin{bmatrix}cos\beta & 0 & sin\beta\\ 0 & 1 & 0\\ -sin\beta & 0 & cos\beta\end{bmatrix}$$ for rotation about Y axis.

$$\mathbf{C}=\begin{bmatrix}cos\gamma & sin\gamma & 0\\ -sin\gamma & cos\gamma & 0\\ 0 & 0 & 1\end{bmatrix}$$ for rotation about Z axis.

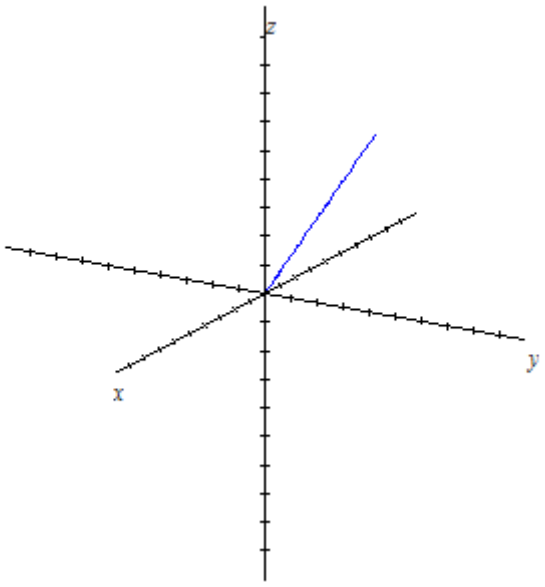
Multiplication by the inverse of the above matrices results in a rotation of the same magnitude but in the opposite direction.

The rotation of any rigid body about an axis along the unit vector $\boldsymbol{v} = l\boldsymbol{i} + m\boldsymbol{j} + n\boldsymbol{k}$, by a known angle ϕ can be derived using the above matrices.

(NOTE: l, m and n are the direction cosines. So $l^2 + m^2 + n^2 = 1$)

Since we know the transformed coordinates of a rotation only about the X, Y, or Z axis, we will have to orient our system, such that the axis of rotation coincides with the X, Y, or Z axis. We will consider the Z axis in our discussion. The transformation matrices applied to the axis of rotation to orient it along the Z axis, when applied to every coordinate of the rigid body, will orient the entire system such that, the body is oriented with its axis pointing along the Z axis.

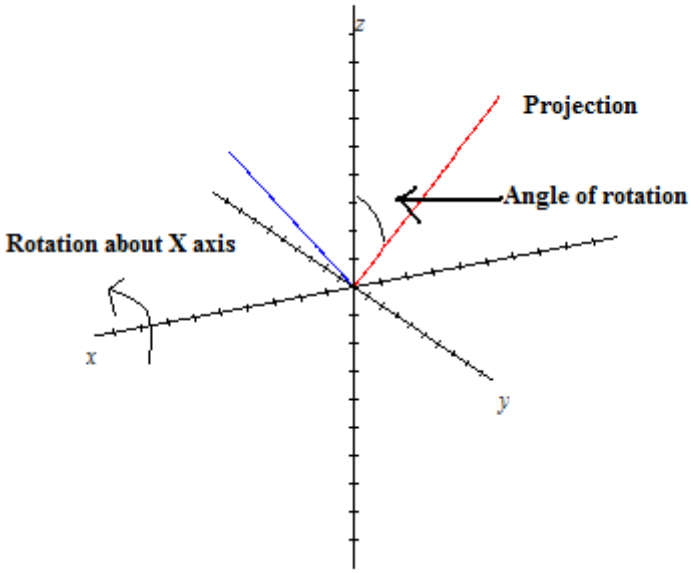
FIGURE 1: Initial axis



REPORT

The axis of rotation can be oriented along the Z axis by two elementary rotations. The first rotation should be such that, the axis of rotation moves into the X-Z plane. This can be achieved by a single rotation about the X axis. The magnitude of the angle by which the rotation is to be performed can be found out by calculating the angle between the projection of the axis of rotation on the Y-Z plane and the Z axis. The direction can be found out by observing the direction in which the y component of the new axis becomes zero.

FIGURE 2: First Rotation



The projection of the original unit vector on the Y-Z plane is $m\boldsymbol{j} + n\boldsymbol{k}$.

Suppose the angle of rotation is θ. Then:

$$\cos\theta = \frac{n}{\sqrt{m^2+n^2}}$$

$$\sin\theta = \frac{m}{\sqrt{m^2+n^2}}$$

Multiplication by inverse of the matrix defined for rotation about the X axis reduces the y component to zero.

(NOTE: Multiplication by the original matrix defined would have resulted in a similar rotation but in the opposite direction.)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{n}{\sqrt{m^2+n^2}} & \frac{-m}{\sqrt{m^2+n^2}} \\ 0 & \frac{m}{\sqrt{m^2+n^2}} & \frac{n}{\sqrt{m^2+n^2}} \end{bmatrix} \begin{bmatrix} l \\ m \\ n \end{bmatrix} = \begin{bmatrix} l \\ 0 \\ \sqrt{1-l^2} \end{bmatrix}$$

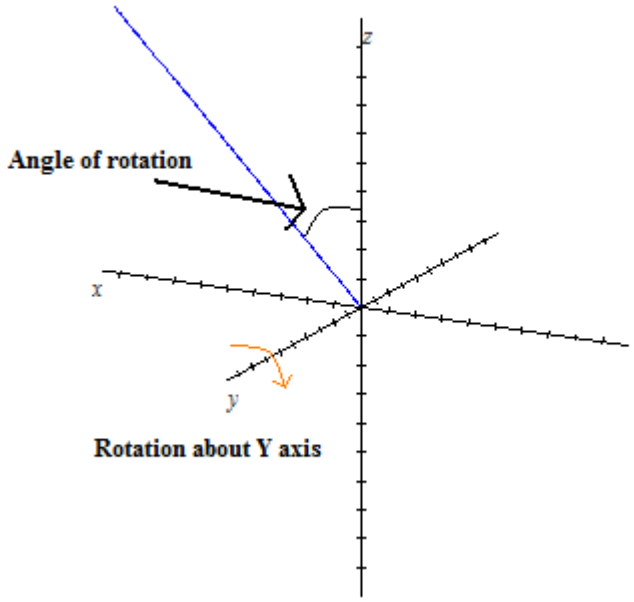
Hence the first matrix multiplication is

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{n}{\sqrt{m^2+n^2}} & \frac{-m}{\sqrt{m^2+n^2}} \\ 0 & \frac{m}{\sqrt{m^2+n^2}} & \frac{n}{\sqrt{m^2+n^2}} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad (1)$$

Here x', y' and z' represent the transformed coordinates of any point lying on the rigid body.

The second rotation should be such that the axis of rotation becomes the Z axis. This can be achieved by a single rotation about the Y axis. The magnitude of the angle can be calculated by finding the angle between the axis of rotation after first transformation and the Z axis. The direction can be obtained by observing the direction in which the x component becomes zero and z component becomes 1.

FIGURE 3: Second Rotation



Suppose the angle by which rotation is to be carried out is θ. Then:

$$\cos\theta = \sqrt{1-l^2}$$

$$\sin\theta = l$$

Multiplication by inverse of the matrix defined for rotation about the Y axis, reduces the x component to zero and z component to 1:

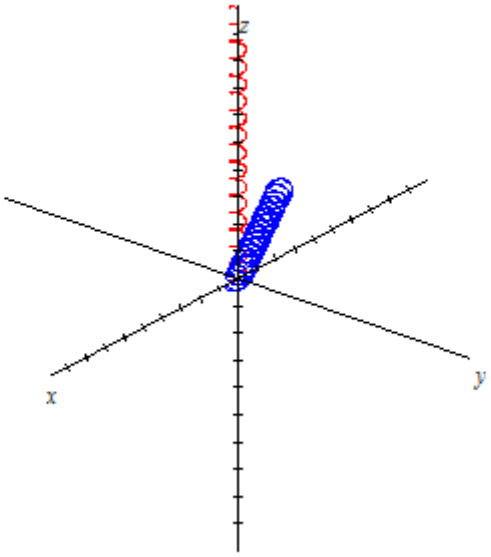
$$\begin{bmatrix} \sqrt{1-l^2} & 0 & -l \\ 0 & 1 & 0 \\ l & 0 & \sqrt{1-l^2} \end{bmatrix} \begin{bmatrix} l \\ 0 \\ \sqrt{1-l^2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Hence the second matrix multiplication is

$$\begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} = \begin{bmatrix} \sqrt{1-l^2} & 0 & -l \\ 0 & 1 & 0 \\ l & 0 & \sqrt{1-l^2} \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \quad (2)$$

Here x'', y'' and z'' represent the transformed coordinates of any point lying on the rigid body.

FIGURE 4: Example of both matrices applied to helical axis as well as helix.



Using the above two matrices we can obtain a relationship between the initial x, y, and z coordinates (with the axis of rotation along $\boldsymbol{v} = l\boldsymbol{i} + m\boldsymbol{j} + n\boldsymbol{k}$) and the final x'', y'' and z'' coordinates (with the axis of rotation along Z axis). If we multiply another matrix to the x'', y'' and z'' coordinates such that the body rotates about the Z axis by the desired angle ϕ, and then solve for the new values of x, y and z, we will get the new coordinates for the rigid body rotated about the axis $\boldsymbol{v} = l\boldsymbol{i} + m\boldsymbol{j} + n\boldsymbol{k}$ by an angle ϕ.

The following equations will illustrate the procedure.

Relation between (x'', y', z'') and (x, y, z):

$$x'' = x\left(\sqrt{1-l^2}\right) - l\left(\frac{my+nz}{\sqrt{m^2+n^2}}\right) \quad (3)$$

$$y'' = \frac{ny-mz}{\sqrt{m^2+n^2}} \quad (4)$$

$$z'' = lx + my + nz \quad (5)$$

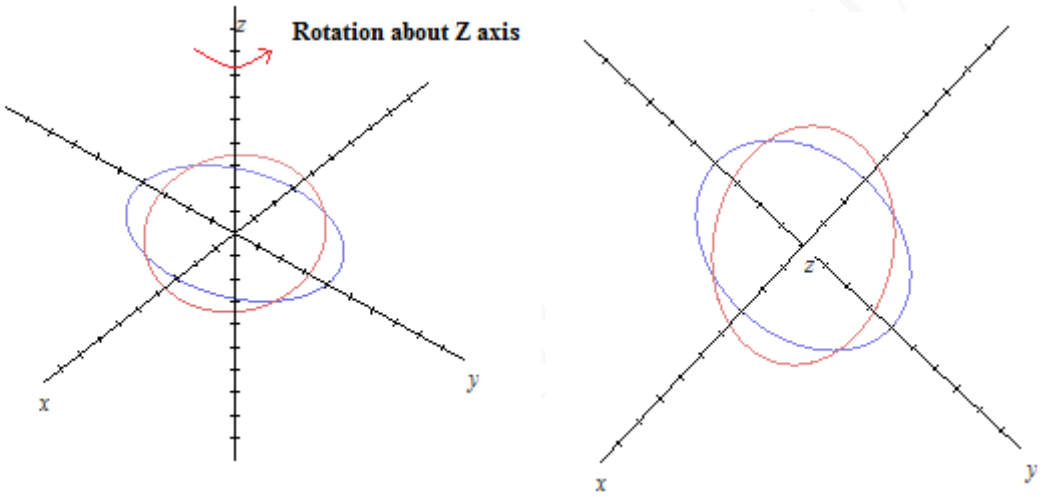
REPORT

Here (x'', y'', z'') represents the transformed coordinates of the rigid body with its axis of rotation along the Z axis, and (x, y, z) are the original coordinates with the axis of rotation along $= l\boldsymbol{i} + m\boldsymbol{j} + n\boldsymbol{k}$ **.**

Let (x''', y''', z''') represent the transformed coordinates of the rigid body (with axis of rotation along Z axis) after a rotation about the Z axis by an angle ϕ.

$$\begin{bmatrix} x''' \\ y''' \\ z''' \end{bmatrix} = \begin{bmatrix} cos\varphi & sin\varphi & 0 \\ -sin\varphi & cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} \quad (6)$$

FIGURE 5: Ellipse rotated along Z axis.



Using equations **3**, **4** and **5,** we can find the relation between the transformed coordinates (x''', y''', z''') and the coordinates of the same rigid body with the axis of rotation, along $\boldsymbol{v} = l\boldsymbol{i} + m\boldsymbol{j} + n\boldsymbol{k}$ **.**

Let the coordinates for the new position of the rigid body with the original axis of rotation be, (x~, y~, z~).

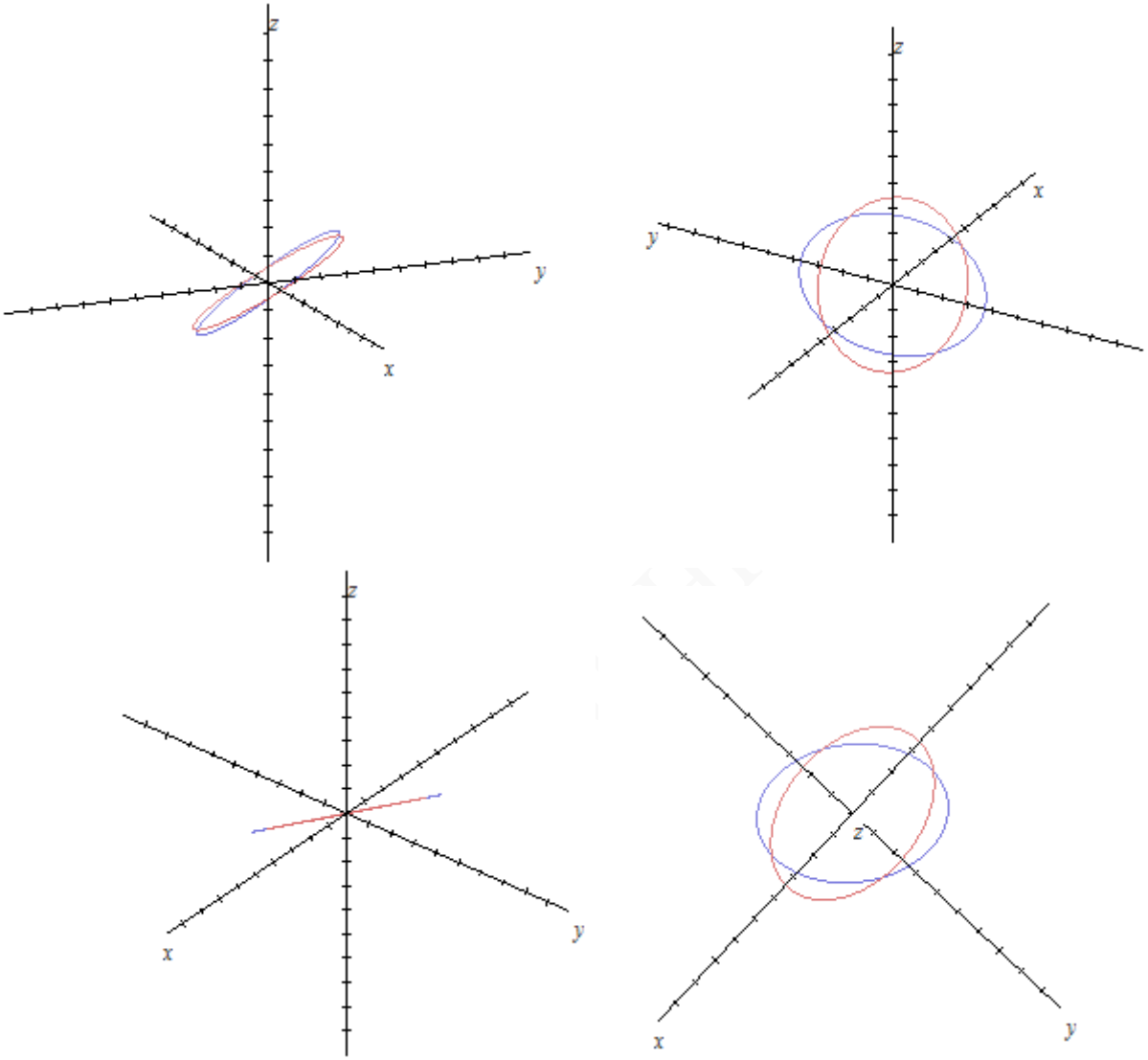
Solving for x~, y~, z~ gives us the new position of the rigid body rotated by an angle ϕ, about the unit vector $= l\boldsymbol{i} + m\boldsymbol{j} + n\boldsymbol{k}$ **.**

$$\begin{bmatrix} x^{\sim}\left(\sqrt{1-l^2}\right) - l\left(\frac{my^{\sim}+nz^{\sim}}{\sqrt{m^2+n^2}}\right) \\ \frac{ny^{\sim}-mz^{\sim}}{\sqrt{m^2+n^2}} \\ lx^{\sim} + my^{\sim} + nz^{\sim} \end{bmatrix} = \begin{bmatrix} x''' \\ y''' \\ z''' \end{bmatrix} \quad (7)$$

Solving the above 3 simultaneous linear equations for x~, y~, and z~ gives us the desired coordinates.

So using the above equations, we can rotate a rigid body about any axis, by the desired angle.

FIGURE 6: Example of ellipse in figure 5 reoriented along its original axis using equation **7**, resulting in equivalent rotation by same angle along the original axis.



2.2 USAGE IN PROGRAM

Using the above equations, if the unit vector along the axis of rotation and the angle by which the rotation is to be performed is known then, we can patch, bend, rotate, or swing any protein fragment we desire. A rotation about the desired unit vector without taking a point on the protein as the origin will lead to a general rotation about the origin of the system of coordinates. Therefore before every rotation an atom in the protein is taken as reference and the position vectors of the rest of the atoms are obtained with respect to the reference. The rotation is then performed about the specified axis and the new coordinates are obtained. The summation of the new coordinates and the reference atom vector, repositions the protein to the correct position.

(NOTE: A common region has been included at the ending of the first and starting of the next fragment, for every pair of protein fragments taken.)

REPORT

For rotating the second fragment, I have taken the bond between, N and C_α of the first amino acid after the common region, as the axis of rotation and the angle by which the rotation is to be performed as a variable. C_α of the same amino acid, has been chosen as reference.

For bending the second fragment, I have selected the region around the first amino acid after the common region and bent the protein by a fraction of the specified angle at each backbone atom along the axis perpendicular to the peptide bond and the bond between, N and C_α of the first amino acid after the common region. The backbone atoms have been chosen as reference for each small rotation.

An algorithm similar to bending has been used for swinging the two fragments apart, but the axis specified is the one perpendicular to the resultants of the sum of distances of each atom of both fragments with respect to the first atom of the first amino acid after the common region.

2.3 PATCHING

For patching to be performed a common region is available in both protein fragments. A reference atom is chosen as well as two other atoms from each fragment. N, C and C_α from the same amino acid have been chosen from both fragments. Both the axis as well as the angle of rotation is unknown.

Here we will use Euler's rotation theorem to determine the axis of rotation, which states:

In three-dimensional space, any displacement of a rigid body such that a point on the rigid body remains fixed, is equivalent to a single rotation about a fixed axis that runs through the fixed point.

This means, if we take the reference as the fixed point, then the position vectors of the other two atoms with respect to the reference of their respective fragments are related to each other by a single rotation about an axis passing through the reference. Once the axis and the angle by which the rotation is to be performed has been established, the second fragment can be rotated about the calculated axis by using the procedure established in section **2.1**, with N of the second fragment taken as the origin (with respect to which the position vectors are to be taken). The fragment can then be translated to the correct position by the summation of the new coordinates with the position vector of the reference atom taken on the first fragment (position vector of N of the first fragment).

Assuming the unit vector along the axis of rotation to be $= l\boldsymbol{i} + m\boldsymbol{j} + n\boldsymbol{k}$.

We will require three equations to calculate the values of l, m and n.

The first being:

$$l^2 + m^2 + n^2 = 1 \qquad (8)$$

The other two equations can be obtained with the help of the other two points taken.

(Note: N is reference, and reference has coordinates (0, 0, 0), C and C_α are the other two points on each fragment)

REPORT

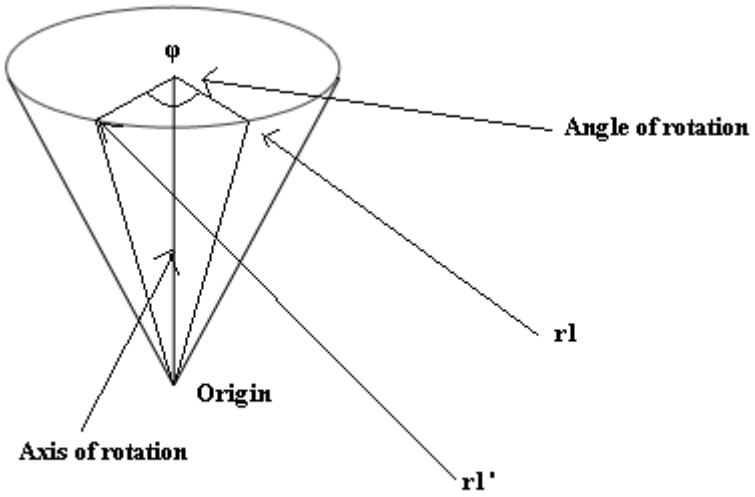
Let us assume that the coordinates of the two points with respect to their references be represented by points **r1**, **r2**, and **r1'**, **r2'**.(**r1**, **r2** from fragment 1and **r1'**, **r2'** from fragment 2).

Then the projection of **r1** on the axis of rotation should be equal to the projection of **r1'**on the axis of rotation.

Because:

1. Both **r1** and **r1'** have the same distance from their respective references.
2. The angle between **r1** and the axis as well as, **r1'** and the axis will remain same.
3. The rotation of any point about a fixed axis is always along the exterior of a cone with the tip centred at the origin.
4. The perpendicular distance of a point remains same from the axis of rotation.

FIGURE 7 Illustration of rotation about axis of rotation



Note: In the figure given above, projection of both **r1** and **r1'** on the axis of rotation is equal to height of the cone.

Therefore: Projection of **r1** on the line λ***v*** = Projection of **r1'**on the line λ***v***, or

$$\frac{\boldsymbol{r1}.\lambda\boldsymbol{v}}{\lambda|\boldsymbol{v}|} = \frac{\boldsymbol{r1}'.\lambda\boldsymbol{v}}{\lambda|\boldsymbol{v}|}$$

OR,

$$\boldsymbol{r1}.\boldsymbol{v} = \boldsymbol{r1}'.\boldsymbol{v}$$

Here ***v*** is the unit vector along the axis of rotation, and λ is an arbitrary constant.

Let (x1, y1, z1) and (x1', y1', z1') be the x, y, and z components of **r1** and **r1'** respectively. Then using the equation above, we get:

$$lx1 + my1 + nz1 = lx1' + my1' + nz1'$$

OR

$$l\Delta x1 + m\Delta y1 + n\Delta z1 = 0 \quad (9)$$

Where $\Delta x1 = x1 - x1'$, $\Delta y1 = y1 - y1'$, $\Delta z1 = z1 - z1'$

Similarly

$$l\Delta x2 + m\Delta y2 + n\Delta z2 = 0 \quad (10)$$

Using equations 8,9 and 10 we can find the values of l, m and n.

To find the value of the angle of rotation we will apply the procedure used in section 2.1 to orient the axis of rotation along the z axis. Once the axis has been aligned along the Z axis, the angle between the projections of either C or C_α of both fragments, on the X-Y plane can be calculated. The calculated angle is the angle of rotation. Another method is to solve the matrix equation for rotation about the Z axis, between the C or C_α (either atom can be used, as both will give the same result) of the second and first fragments, and obtain the values of $cos\varphi$ and $sin\varphi$.

(NOTE: Remember, the angle between the atoms of second and first fragment is to be calculated, as the program has been written such that it orients the second fragment towards the first one.)

3. ALGORITHM:

3.1 CODE FOR PATCHING

Explanation of the code has been done with appropriately placed comments within the *cpp* file of the program. A description of the algorithm used for orienting the fragments is, as follows.

3.2 ALGORITHM FOR ORIENTATION

Once the patching is complete, the second fragment is rotated about the selected bond, to find the rotational position of minimum clashes. The minimum number of clashes is stored in the variables *bendclashes1* and *bendclashes2.* Two variables are required as bending is in both directions (anticlockwise and clockwise). The rotational position of minimum clashes is designated as *centre* for the first bending operation. The rotational positions on both sides of *centre* are checked for clashes (after bending the second fragment by one position in either direction). If either side has equal or lower number of clashes with respect to the *centre*, then the program continues checking for clashes by rotating in the direction the number of clashes is lower, till the lowest possible value for clashes in that direction is obtained. These values are then designated as *leftclashes* and *rightclashes* for anticlockwise and clockwise directions respectively. The rotational position having lower number of clashes is then designated as *centre* for the next bending operation. If number of clashes is same then the position closer to the *centre* is designated as *centre* for the next bending operation. If the number of clashes at the new *centre* is lower than *bendclashes* , then the bending position and

number of clashes at *centre* are stored in *benpos* and *bendclashes* respectively. The rotational position is stored in *rotpos*.

This process continues till, either the loop for bending completes or the value of *bendclashes* is not replaced for more than three times in a row.

Once the values for *bendclashes1* and *bendclashes2* have been obtained, the one with the lower value is chosen as *finalbendclashes*. The rotational and bending positions are also recorded in *finalrotpos* and *finalbendpos* respectively. The second protein fragment is oriented to *finalrotpos* and *finalbendpos*.

One final operation is also performed. The second fragment is swung along the axis perpendicular to the resultants of the vector positions of all coordinates of each fragment, with respect to the first atom of amino acid no. *res_dif + end_patch*.

The aim is to find the position of minimum clashes when the second fragment is swung about the above axis. The second fragment is then oriented to the position obtained. The reoriented protein is now written to a *pdb* file.

REFERENCES[i]

CALCULUS by Thomas & Finney 9th Edition

CLASSICAL MECHANICS by Herbert Goldstein 2nd Edition