Computing The Protein Angles That Characterize Backbone Structure. Creating Ribbon Models.

James D Emery

9/2/2001

Contents

0.1 Introduction 2
0.2 Computing the Dihedral Angles from the Sequence. The Inverse Transformation. 7
0.3 Application to A PDB File 11
0.4 An Alternate Method for Computing The Dihedral Angle Between The Planes That Are Defined By A Sequence Of Four Points 13
0.5 The Definition of a Backbone Sequence With Angles And Bond Lengths 29
0.6 A Reverse Property. 29
0.7 Protein Description 31
0.8 Protein Backbone 32
0.9 Angles Characterizing Protein Structure 33
0.10 Specifying a Sequence of Points in Three Space With Dihedral Angles, Bond Angles, and Bond Distances. 34
0.11 The Ribbon Model of a Protein 35
0.12 X-Ray Diffraction 36
0.13 Locating the Hydrogen Atom in the Polypeptide Plane 37
0.14 Making a Physical Model of the Polypeptide Plane 38
0.15 A Maple Program For The Polypeptide Plane Computations . 40
0.16 The Protein Folding Problem 42
0.17 A Program List 43
0.18 Bibliography 43

0.1 Introduction

We shall define and compute the bond lengths, the bond angles, and the dihedral angles that characterize protein backbone structure. We shall give algorithms for doing the computations on PDB (Protein Data Bank) files. We shall create techniques for making computer graphics VRML models of proteins, where the alpha helices may be modeled as helical ribbons, and selected amino acid residues may be modeled as ball and stick models.

But before we go into the details of these things, here is a taste of some of the programs that we have created for extracting the angle and bond information from PDB files, and a taste of the graphics programs. The PDB files that we use have the ".ent" file extension.

The file **proteinbb.cpp** contains a C++ program that computes the bond lengths and dihedral angles for a backbone sequence of a protein specified in a Protein Data Bank file. This program also creates various VRML graphics. This includes a spline curve through the protein backbone. The user may add ribbon models of any alpha helix that occurs in the pdb file. A Ball and stick model can be added for any amino acid residue. The ball and stick models are read from files previously created by another program, such as **aminowrl.cpp**. There are four programs for creating ball and stick models. **getamino.cpp** takes as parameters, the amino acid number, the pdb file and the output file. The output file specifies the atoms and the bonds of the molecule. **readmol.cpp** reads the molecule definition created by **getamin.cpp**, or perhaps such a molecule file created by hand, and produces a VRML file (extension wrl), which can be displayed by a VRML program such as Cosmo Player, or Microsoft VRML viewer. The extraction and VRML creation can be done in one step with the program **aminowrl.cpp**. Program **aminowrlh.cpp** is similar to **aminowrl.cpp**, but adds the hydrogen atom of the polypeptide plane, which in general is not present in the PDB file (X-Ray Crystallography is not able to locate a hydrogen atom). These graphics files are written as VRML files (Virtual Reality Modeling Language). They may be viewed with a VRML viewer such as Microsoft VRML viewer, or Cosmo Player. These viewers work with a browser such as Netscape or Internet Explorer. One can travel about the model, or one can examine the model by rotating it and so on. This rotation allows one to understand the three dimensional nature of the model. Shading can help us visualize 3d surfaces. Space curves are very hard to understand in a two dimensional

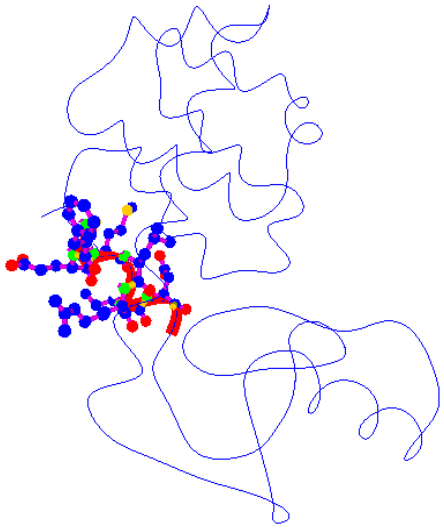

Figure 1: The elements of the a chain of la.ent. The figure shows the first alpha helix and some of the amino acid residues forming the helix. Also shown is the curve through the entire backbone.

projection. By dynamically rotating the curve, the shape is easily comprehended. Some help information is given when the program name is typed without parameters. For example, typing **proteinbb** results in the following output:

```
proteinbb.cpp, Jim Emery, Version ******
Reads backbone atoms from a pdb ent file.
Calculates the angles and the bond lengths.
Generates a VRML file for the protein, with ribbon and ball and stick options.
Reference: protein.tex
The default name for the output anglesfile is angles.txt
The default name for the output VRML file is ribbon.wrl
Usage: proteinbb inputfile.ent [anglesfile.txt ribbonfile.wrl]
```

proteinbb reads and extracts the backbone of a pdb file. It computes the bond lengths, bond angles, and dihedral angles of the backbone chain. Here is a portion of the output file for pdb3fxc.ent. It lists just the first few of the 98 atoms in the backbone chain.

Figure 2: All of the alpha helices of the first chain in the protein contained in file la.ent. The helices are presented as ribbon models.

```
 N ALA 1
29.851 -11.261 54.580
 CA ALA 1
28.916 -11.062 53.392
 C ALA 1
27.545 -11.661 53.824
 N THR 2
26.673 -11.632 52.851
 C-N =      1.3068871 Angstroms    (bond length)
 CA-C-N =      111.86095 Degrees   (bond angle)
 N-CA-C-N =      175.73775 Degrees (dihedral angle)
 CA THR 2
25.364 -12.202 52.959
 N-CA =      1.4317978 Angstroms
 C-N-CA =       123.0258 Degrees
 CA-C-N-CA =     -175.75436 Degrees
```

```
 C THR 2
25.053 -13.229 51.878
 CA-C =      1.5231582 Angstroms
 N-CA-C =      113.67565 Degrees
 C-N-CA-C =      124.12872 Degrees
 N TYR 3
24.367 -12.944 50.798
 C-N =      1.3108093 Angstroms
 CA-C-N =      123.02468 Degrees
 N-CA-C-N =       98.30426 Degrees
```

The last element listed here shows a nitrogen atom in the 3rd amino group. It lists the atom's xyz coordinates, the bond length before the nitrogen, the bond angle of CA-C-N, and the dihedral angle of N-CA-C-N.

We can find the average values of bond lengths, bond angles, and dihedral angles, by generating a batch file with the program called **grepprot**. For example if we wanted to find the average bond angle for N-CA-C, which is given in the books and in the original Pauling paper as 110 degrees, we would type:

```
grepprot outfile N-CA-C
```

We get the batch file **doit.bat**, which contains:

```
echo off
grep3 " CA-C-N " outfile > tmp1.txt
cutl -f3 tmp1.txt > tmp2.txt
echo CA-C-N
meanmm tmp2.txt
We run the script doit and get output:
 CA-C-N
 Mean=     113.9305835 Min=        104.5974 Max=       129.78559
```

We see that there is quite a bit of deviation from the standard angle of 110 degrees.

In the same way if we look at the average of the dihedral angle at the polypeptide bond CA-C-N-CA, we see that it deviates a bit from the standard value of 180 degrees. Thus the atoms of the polypeptide plane are sometimes

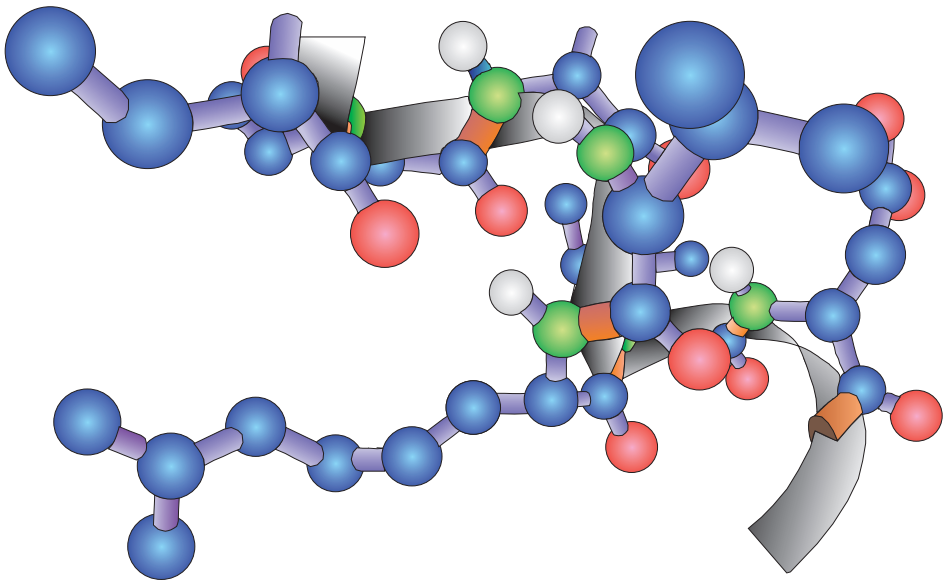

Figure 3: The figure shows the first alpha helix of the protein and some of the amino acid residues forming the helix. The white spheres are hydrogen, the red oxygen. The ribbon passes through the polypeptide bonds, which are the thicker bonds in the figure. In the middle of the picture we see a hydrogen atom near an oxygen atom. This is one of the hydrogen bonds. The hydrogen bonds are responsible for the shape of the alpha helix.

outside of the plane. Some of this deviation is no doubt experimental error in the X-Ray crystallography and the assignment of atoms to electron density centers.

The extracted backbone chain can be used to create a ribbon model. We can then triangulate it, and display it in 3d as a VRML file. Some versions of Internet Explorer have a built in VRML Browser. Cosmo Player is a better VRML viewer for large files. It is free for download from SGI and will install itself into the local browser. VRML viewers allow the model to be examined and rotated dynamically. We could in theory create our own protein backbone by assigning arbitrary dihedral angles. Hence we could attempt to "fold" or "unfold" the protein.

0.2 Computing the Dihedral Angles from the Sequence. The Inverse Transformation.

Given points p_1, p_2, p_3, and p_4, let us define

$$r_1 = p_1 - p_2$$

$$r_2 = p_3 - p_2$$

$$r_3 = p_4 - p_3.$$

We define a unit vector u_2 in the direction of r_2 as

$$u_2 = \frac{r_2}{\|r_2\|}.$$

Let P_{u_2} be the projection operator in the direction of u_2, which is defined by

$$P_{u_2}(x) = x - (x \cdot u_2)u_2$$

This is the projection to the plane orthogonal to u_2.

Define

$$u_1 = \frac{P_{u_2}(r_1)}{\|P_{u_2}(r_1)\|}$$

$$u_3 = \frac{P_{u_2}(r_3)}{\|P_{u_2}(r_3)\|}$$

Let s be the sign of $(u_1 \times u_3) \cdot u_2$. Then the dihedral angle is defined to be

$$\theta = f_1(p_1, p_2, p_3, p_4) = \text{angle } (u_1, u_3)s$$

The bond angle is defined as the angle between edge p_2p_3 and p_3p_4, and is equal to $\pi - \alpha$ where

$$\alpha = f_2(p_1, p_2, p_3, p_4) = \text{angle}(r_2, r_3).$$

The bond length is defined to be

$$b = f_3(p_1, p_2, p_3, p_4) = \|r_3\|.$$

The angles-length function f is defined as

$$(\theta, \alpha, b) = f(p_1, p_2, p_3, p_4),$$

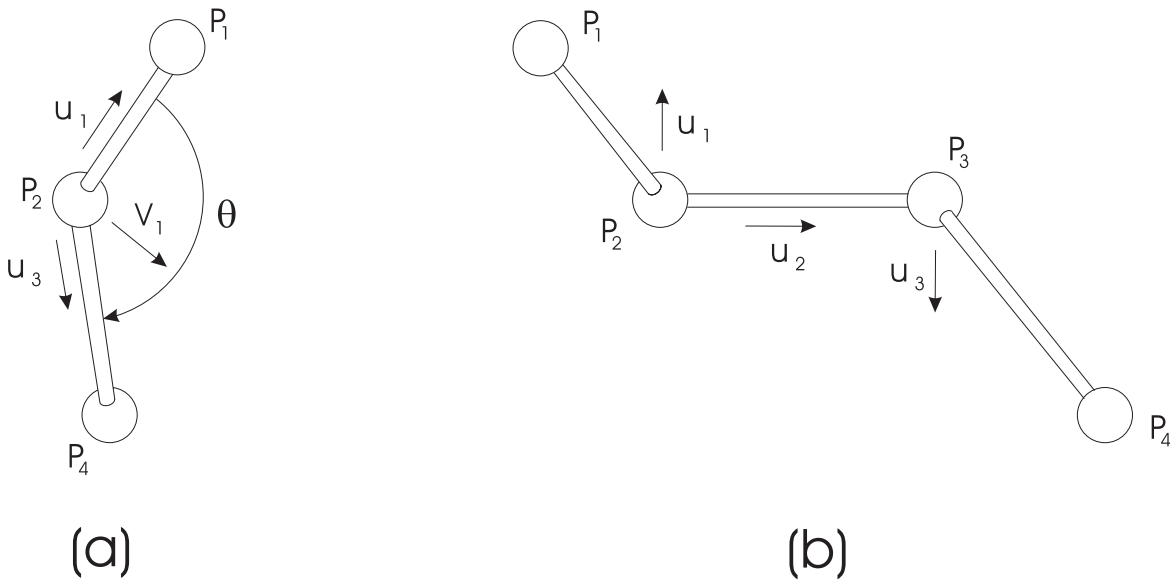

Figure 4: The Projection Method For Dihedral Angle Definition And Calculation. (a)A view in the projection direction of edge p_2p_3. (b) An orthogonal view showing edge p_2p_3 at full scale. u_1 is the unit vector in the direction of the projection of p_2p_1, u_2 is the unit vector in the direction of p_2p_3, and u_3 is the unit vector in the direction of the projection of p_3p_4. The dihedral angle θ is the angle between the planes $p_1p_2p_3$ and $p_2p_3p_4$. This is the clockwise angle, while viewing in direction p_2p_3, that vector u_1 must be rotated to make it coincide with u_3.

where the component functions of f are f_1, f_2, and f_3.

Ideally the angle ϕ between two vectors could be obtained from the dot product. Given two unit vectors w_1 and w_2, the angle ϕ between them can be obtained mathematically from the equation.

$$w_1 \cdot w_2 = \cos(\phi).$$

But near $\phi = 0$ the curve of $\cos(\phi)$ is nearly flat. So an accurate inverse can not be computed in this neighborhood. We may handle this computational difficulty using the dot product and the cross product together. We have

$$\|w_1 \times w_2\| = |sin(\phi)|.$$

Thus

$$\phi = \arg(w_1 \cdot w_2, \|w_1 \times w_2\|),$$

where arg is the argument function. The argument function is essentially the library function atan2. We take the angle between vectors to be the smallest angle between them, and so the angle is between 0 and π. This angle computation is done in function **veca3**.

The Inverse of f.

We shall define the inverse function

$$p_4 = g(p_1, p_2, p_3, \theta, \alpha, b).$$

By visualizing the geometry we arrive at the following calculations for defining g. Let

$$v_2 = u_2 \times u_1$$

and

$$v_1 = \cos(\theta)u_1 + \sin(\theta)v_2.$$

Then v_1 lies in the $p_2p_3p_4$ plane and is orthogonal to u_2 (actually v_1 is the u_3 of the projective calculation for the dihedral angle). Then u_2 and v_1 serve as a basis for locating the vector r_3 using the angle α and the length b. We have

$$r_3 = b(\cos(\alpha)u_2 + \sin(\alpha)v_1).$$

Then we get

$$p_4 = g(p_1, p_2, p_3, \theta, \alpha, b) = p_3 + r_3.$$

Let us now prove formally that we have constructed the inverse. Namely, if

$$\theta' = f_1(p_1, p_2, p_3, g(p_1, p_2, p_3, \theta, \alpha, b),$$

$$\alpha' = f_2(p_1, p_2, p_3, g(p_1, p_2, p_3, \theta, \alpha, b),$$

$$b' = f_3(p_1, p_2, p_3, g(p_1, p_2, p_3, \theta, \alpha, b),$$

then

$$\theta' = \theta$$

$$\alpha' = \alpha$$

$$b' = b.$$

We shall show that $\theta' = \theta$. The other two equations are obvious. Let us use primes for the quantities in the algorithm for f. We have

$$u_1' = u_1$$

$$u_2' = u_2$$

and

$$u_3' = \frac{P_{u_2}(r_3)}{\|P_{u_2}(r_3)\|},$$

where

$$r_3 = b(\cos(\alpha)u_2 + \sin(\alpha)v_1).$$

We have

$$P_{u_2}(r_3) = b\sin(\alpha)v_1.$$

Then because $\sin(\alpha)$ is positive, we have

$$u_3' = \frac{P_{u_2}(r_3)}{\|P_{u_2}(r_3)\|} = v_1.$$

Then

$$|\theta'| = \text{angle}(u_1, u_3') = \text{angle}(u_1, v_1) = |\theta|.$$

It remains to show that θ' and θ have the same sign. Let s' be the sign of θ', which is defined as the sign of

$$(u_1 \times u_3') \cdot u_2 = (u_1 \times v_1) \cdot u_2.$$

We have

$$(u_1 \times v_1) \cdot u_2 = (u_1 \times (\cos(\theta)u_1 + \sin(\theta v_2)) \cdot u_2$$
$$= (u_1 \times \sin(\theta)v_2) \cdot u_2$$
$$= \sin(\theta)(u_1 \times (u_2 \times u_1)) \cdot u_2)$$

Now

$$u_1 \times (u_2 \times u_1) = u_2(u_1 \cdot u_1) - u_1(u_1 \cdot u_2),$$

so

$$(u_1 \times (u_2 \times u_1)) \cdot u_2 =$$
$$\|u_1\|^2\|u_2\|^2 - (u_1 \cdot u_2)^2 > 0.$$

The last expression is the Cauchy-Schwartz inequality. Therefore the sign of

$$(u_1 \times u_3') \cdot u_2)$$

is the sign of $\sin(\theta)$, which is the sign of θ. Therefore θ' and θ have the same sign, and so they are equal,

$$\theta' = \theta.$$

0.3 Application to A PDB File

The program **proteinbb.cpp** extracts coordinate information from a PDB file and computes angles. As an example we consider the file **137L.ent**. Here is a table of computed angles that was captured from the Protein Data Bank internet site for this file:

```
Chain 137L:A
angle          total#   average stddev  min      at       max    at
Phi             147     280.25  36.222  -43.30   THR 115 54.37  ASN 55
Phi (PRO)       3       -65.87  10.223  -80.33   PRO 37  -58.59 PRO 86
Phi (GLY)       11      136.85  95.702  -62.80   GLY 77  61.39  GLY 28
Phi helix       100     -64.84  10.082  -112.88 MET 106 -43.30 THR 115
Psi             150     265.97  96.914  -1.27    LEU 15  2.46   PRO 37
Psi (GLY)       11      151.30  129.268 -19.76   GLY 113 7.55   GLY 23
Psi helix       100     -40.58  8.836   -59.59   THR 115 -2.72  MET 106
```

```
Omega           161     179.69  1.918   -175.81 THR 54  174.28 LEU 99
Chi1 g(-)       16      67.21   14.139  36.95   MET 106 87.79  ARG 80
CHI1 trans      45      180.93  19.090  126.20  GLU 108 223.62 ASN 132
 Chi1 g(+)      71      -65.64  15.930  -110.54 ASN 53  -11.46 LYS 135

Chain 137L:B
angle          total#   average stddev  min     at      max    at
Phi             149     280.49  34.792  -33.03  THR 115 56.62  ASN 55
Phi (PRO)       3       -65.23  4.722   -70.57  PRO 37  -59.09 PRO 86
Phi (GLY)       11      135.90  94.542  -64.12  GLY 110 60.13  GLY 28
Phi helix       105     -65.60  10.936  -114.81 MET 106 -33.03 THR 115
Psi             152     268.94  94.802  -7.87   LYS 162 0.36   LYS 135
Psi (GLY)       11      150.54  130.429 -3.73   GLY 113 10.45  GLY 56
Psi helix       105     -39.21  9.766   -58.71  TRP 126 1.01   MET 106
Omega           163     179.95  2.224   -172.96 THR 142 175.31 LEU 91
Chi1 g(-)       16      62.67   17.559  25.95   ASP 127 103.39 SER 90
CHI1 trans      47      182.46  14.715  156.97  TRP 138 230.49 LYS 147
Chi1 g(+)       70      -66.25  13.904  -111.24 ARG 96  -36.30 LYS 124
RCSB
```

The angle ϕ is the dihedral angle of the sequence $C - N - C_\alpha - C$. The angle ψ is the dihedral angle of the sequence $N - C_\alpha - C - N$. From the table we see that the minimum value of ϕ occurs at residue 115, and has value -43.30. A relevant portion of the output file **137LAOUT.txt** is

```
Last read: C from residue PHE 114
 x=41.120 y=43.273 z=26.694
 CA-C =      1.5229737 Angstroms
 N-CA-C =      111.75185 Degrees
 phi(114)=  C-N-CA-C =      -80.65172 Degrees
 inversion: p4=           41.12          43.273          26.694
Last read: N from residue THR 115
 x=42.123 y=44.152 z=26.633
 C-N =      1.3350547 Angstroms
 CA-C-N =      115.72171 Degrees
 psi(114)=  N-CA-C-N =      54.185894 Degrees
 inversion: p4=          42.123          44.152          26.633
```

```
Last read: CA from residue THR 115
 x=41.998 y=45.581 z=26.386
 N-CA =      1.4555669 Angstroms
 C-N-CA =      126.12995 Degrees
 CA-C-N-CA =     -179.13683 Degrees
 inversion: p4=          41.998          45.581          26.386
Last read: C from residue THR 115
 x=41.027 y=45.975 z=25.281
 CA-C =      1.5228598 Angstroms
 N-CA-C =      115.58751 Degrees
 phi(115)=  C-N-CA-C =     -43.297684 Degrees
 inversion: p4=          41.027          45.975          25.281
Last read: N from residue ASN 116
 x=41.300 y=45.482 z=24.062
 C-N =       1.342959 Angstroms
 CA-C-N =      115.72417 Degrees
 psi(115)=  N-CA-C-N =     -59.590956 Degrees
 inversion: p4=            41.3          45.482          24.062
```

The line for $\phi(115)$ gives the dihedral angle as -43.297684 Degrees, which agrees with the PDB table value of -43.30. All other values that can be compared also agree. The output file was generated from the command

```
proteinbb 137la.ent 137laout.txt
```

File **137la.ent** contains the portion of **137L.ent** for the first chain A.

0.4 An Alternate Method for Computing The Dihedral Angle Between The Planes That Are Defined By A Sequence Of Four Points

This section gives a variation on the dihedral angle calculation. It uses plane normals, rather than projections. The projection method was given above. The two methods will give the same result, including the sign of the angle.

Both methods give a result that agrees with the biochemical standard for protein dihedral angles.

The structure of a chain of atoms in a molecule is partially specified by dihedral angles, which are also called torsion angles. This calculation as well as the previous calculation given above defines the sign of the angles and so eliminates the ambiguity of sign.

Let p_1, p_2, p_3 and p_4 be four points in space. If the first three points do not lie on a straight line then they define a plane. If the last three points do not lie on a straight line also define a plane. These two planes have a common edge p_2p_3. These two planes form a dihedral angle. There are three edges joining the points: p_1p_2, p_2p_3, and p_3p_4.

Then the first set of three points define a plane, and the last set of three points define a plane. These planes intersect along the edge p_2p_3. The two planes form a dihedral angle at the edge. The dihedral angle is defined to be the angle through which the first plane must be rotated about the edge in order to bring it into coincidence with the second plane. The positive angle corresponds to the right hand rule: with the thumb of the right hand in the direction of the edge, the fingers curl around in the positive direction. This is equivalent to the idea of sitting in back of point p_2 and rotating plane $p_1p_2p_3$ clockwise about edge p_2p_3. We shall define a function

$$\theta = f(p_1, p_2, p_3, p_4),$$

so that θ is the dihedral angle between the planes. We take this angle to be the clockwise angle measured from plane 1 to plane 2 while viewing in the direction from p_2 to p_3. We have chosen our dihedral angle convention to agree with that of the standard for polypeptide chains. The standard is contained in the paper **IUPAC-IUB Commission on Biochemical Nomenclature,1969**, which is listed in the references. We also refer to the book **Principles of Protein Structure** by G. E. Schultz and R.H. Schirmer, Springer-Verlag, 1979, p 19. The book is more vague than the paper, and in fact did mislead me as to the sign of the dihedral angle. This section was originally written when I was mislead. But I have made changes so that the correct sign is now calculated.

The Biochemical Nomenclature paper notes that the 1969 standard reverses some previous dihedral angle conventions. Thus some references in earlier books (before 1969) may have dihedral angles given that are supplements of the angles that follow the 1969 standard.

We shall devise an analytic method for calculating the dihedral angle θ. Refer to the Figure for what follows. In that figure, as we sit on point p_2 looking at point p_3, the dihedral angle is the clockwise angle through which edge p_1p_2 must be rotated about the p_2p_3 axis, in order to make it lie in plane $p_2p_3p_4$. Let q_1, q_2, q_3 be the vectors between the points,

$$q_1 = p_2 - p_1,$$

$$q_2 = p_3 - p_2,$$

$$q_3 = p_4 - p_3.$$

The direction of angle θ is defined by the right hand rule. While grasping the vector q_2 with the thumb in the q_2 direction, the positive angle is in the direction that the fingers curl around q_2. To visualize the angle refer to the Figure.

The normals to the pair of planes are

$$\pm\frac{q_1 \times q_2}{\|q_1 \times q_2\|},$$

and

$$\pm\frac{q_2 \times q_3}{\|q_2 \times q_3\|}.$$

The angle should be zero when the planes coincide, and when at the same time p_1 and p_4 lie on the same side of the edge. That is when p_1 is cis to p_4. In this case the unit normals should coincide. This will occur if we choose the same sign for the normals. So we define the plane normals n_1 and n_2 as

$$n_1 = \frac{q_1 \times q_2}{\|q_1 \times q_2\|},$$

and

$$n_2 = \frac{q_2 \times q_3}{\|q_2 \times q_3\|}.$$

We shall create an orthogonal coordinate system for measuring the angle. The system is defined by three unit orthogonal vectors u_1, u_2, and u_3. We take u_1 to be the normal of the second plane

$$u_1 = n_2.$$

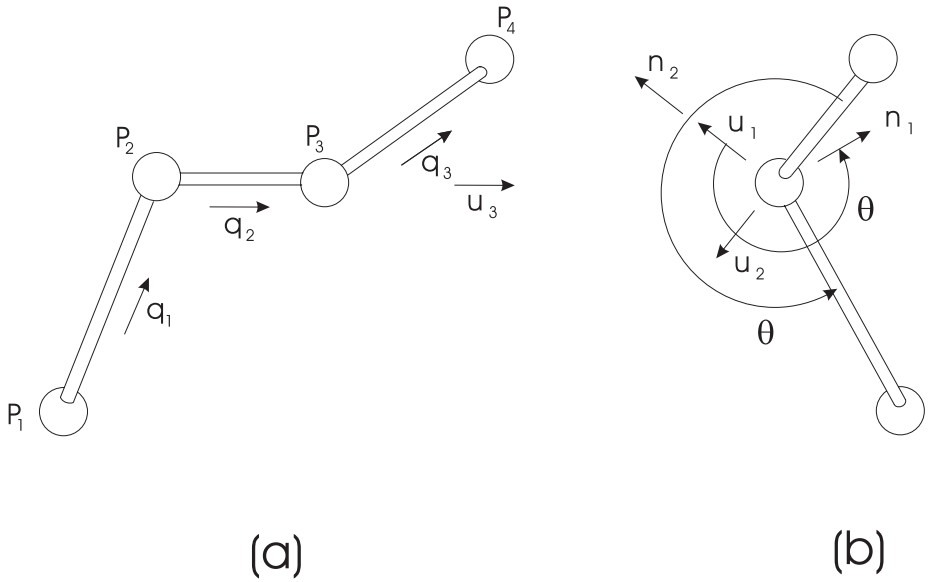

Figure 5: The second alternative dihedral angle definition. (a)Front view with edge q_2 parallel to paper. (b) View orthogonal to the intersection edge q_2 with unit vector u_3, which is in the direction of q_2, directed outward. The plane normals are n_1 and n_2. The orthogonal view is toward the edge view of the planes. The dihedral angle θ is the angle between the planes, or the angle between the normals. Angle θ is the negative of the angle of the vector normal n_1 in the u_1u_2 coordinate system.

We define n_3 to be the unit vector in the direction of the intersection edge

$$u_3 = \frac{q_2}{\|q_2\|}.$$

We define

$$u_2 = u_3 \times u_1.$$

This defines a right handed orthogonal system of unit vectors u_1, u_2, and u_3. We consider a projection, which is orthogonal to the intersection edge, that is orthogonal to q_2 and to u_3. This is a projection to the plane containing u_1, and u_2. This projection plane is the subspace with basis vectors u_1, and u_2. It contains the two plane normals n_1 and n_2. Now n_2 is u_1, therefore the dihedral angle is the negative of the counterclockwise angle θ measured from u_1 to the first plane normal n_1. The angle is counterclockwise here because we are now looking in the opposite direction to q_2. Because $\{u_1, u_2\}$ is a basis of the subspace, the angle θ is defined by the equation

$$\begin{aligned} n_1 &= (n_1 \cdot u_1)u_1 + (n_1 \cdot u_2)u_2 \\ &= \cos(\theta)u_1 + \sin(\theta)u_2. \end{aligned}$$

The dihedral angle $\theta = f(p_1, p_2, p_3, p_4)$ is calculated as

$$\theta = -\arg(k \cdot n_1, k \cdot n_2),$$

where arg is the argument function. In the computer languages Fortran or C, θ can be calculated as

```
theta=-atan2(dotpr(n1,u1),dotpr(n1,u2))
```

Note that if p_1 is cis to p_4, across edge p_2p_3, then $\theta = 0$, and if p_1 is trans to p_4, then $\theta = \pi$. Note also that the dihedral should be thought of as being directly associated with the edge, or the bond p_2p_3, rather than with any point or atom position. This is true because θ is the rotation angle about the edge taking plane 2 to plane 1.

Let us consider an example. Let the points of the sequence be

$$p_1 = (1, 0, 0),$$

$$p_2 = (0, 0, 0),$$
$$p_3 = (0, 1, 0),$$

and

$$p_4 = (0, 1, 1).$$

The first point is on the x axis, the second at the origin, the third on the y axis, and the fourth directly above the third in the z direction. The points are shown in Figure 2. It is clear that according to our convention the dihedral angle should be $-\pi/2$. This is because, viewing in the direction from p_2 to p_3, we must rotate point p_1 -90 degrees clockwise, to bring it into plane $p_2p_3p_4$. Let us do the analytic calculation to verify this. We have

$$q_1 = p_2 - p_1 = (-1, 0, 0),$$

$$q_2 = p_3 - p_2 = (0, 1, 0),$$

and

$$q_3 = p_4 - p_3 = (0, 0, 1).$$

The first plane normal is

$$n_1 = q_1 \times q_2 = (0, 0, -1).$$

The second plane normal is

$$n_2 = q_2 \times q_3 = (1, 0, 0).$$

We have

$$u_1 = n_2 = (1, 0, 0).$$
$$u_3 = q_2 = (0, 1, 0).$$
$$u_2 = u_3 \times u_1 = (0, 0, -1).$$

Then

$$\cos(-\theta) = n_1 \cdot u_1 = 0$$

and

$$\sin(-\theta) = n_1 \cdot u_2 = 1.$$

Therefore,

$$\theta = -\frac{\pi}{2}.$$

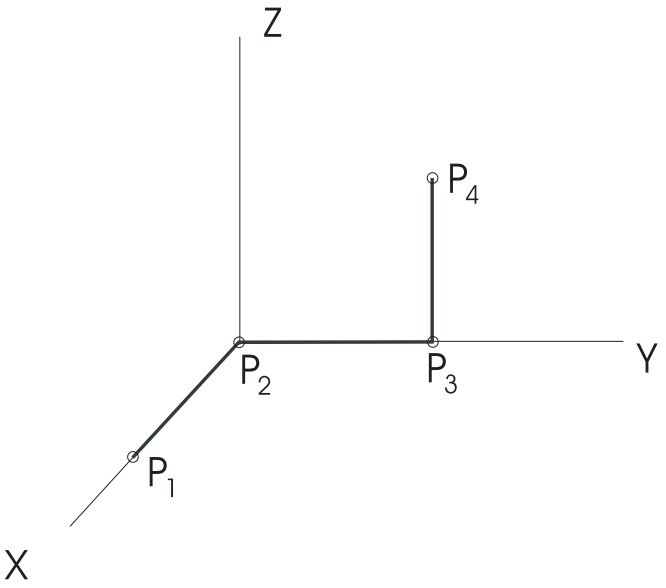

Figure 6: Example calculation of the dihedral angle. The edge p_1p_2 is rotated clockwise by dihedral angle $\theta = -90$ degrees about axis p_2p_3.

The angle is -90 degrees. Here this is the angle one rotates edge p_1p_2 in a clockwise manner, as viewed from point p_2, to make vector p_1p_2 parallel to vector p_3p_4 and thus lie in plane $p_2p_3p_4$. Thus our calculation agrees with the description of the dihedral angle as given in the standard.

Now let us consider the problem of reconstructing a sequence from angles and bond lengths. Recall that we delt with this inversion process before in connection with the projection method. To reconstruct an entire sequence it is sufficient to solve the problem of reconstructing a fourth point from the previous three. Given the points p_1, p_2, p_3, we can reconstruct the next point p_4 from the dihedral angle θ, the bond angle ρ, and the bond length s. In the projection method, we used different symbols for the bond angle and bond length. When the original four points p_1, p_2, p_3, p_4 are known, the latter two values ρ and s are defined as follows. The bond angle ρ is defined to be the angle between the vectors q_2 and $q_3 = p_4 - p_3$. This is a positive angle with magnitude between 0 and π. The calculation contained in the code below is in function **veca3**. The angle ρ is the supplement of the angle $p_2p_3p_4$. So, for example, in the case of the protein backbone sequence, if p_3 is the alpha

carbon, then angle $p_2p_3p_4$ is 110 degrees, so $\rho = 70$ degrees, or in radians

$$\rho = \pi - \frac{110 * \pi}{180}.$$

The bond length s is the length of the vector q_3.

Here we shall use θ, ρ, and s, to reconstruct p_4. We are given, p_1, p_2, p_3 and thus q_1 and q_2. The normal n_1, of the first plane, is the normalized $q_2 \times q_2$. To get n_2, which is the normal to the second plane, and which is defined by points p_2, p_3, p_4, we can use the dihedral angle θ. By the definition of θ, n_2 is n_1 rotated by the dihedral angle θ about axis q_2. We can do this calculation to get n_2.

For information on this calculation, see the paper in the Journal IEEE Computer Graphics and Applications, **A Note on Rotation Matrices,** Jay Fillmore, Feb. 1984. The function in the code below that does this is called **orthgm**. It computes the required orthogonal rotation matrix defined by an axis and angle.

We shall introduce a set of unit vectors v_1 and v_2, which will allow the computation of p_4. v_1 is in the direction of q_2. v_2 will be in the plane $p_2p_3p_4$ and will be perpendicular to v_1. Because, by definition of the dihedral angle θ, n_2 is the normalized value of the cross product $q_2 \times q_3$, it follows that if we define $v_2 = n_2 \times v_1$, then our reconstruction is completed by

$$p_4 = p_3 + s(\cos(\rho)v_1 + \sin(\rho)v_2).$$

Here is the code that exercises the sequence-to-angles calculation and the angles-to-sequence reconstruction (similar functions exist for the projection method, and are actually the ones used in program **proteinbb.cpp**):

```
//seqangs.c computations of angles of a point sequence and inversion
//6/5/01
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int vectorcopy(int n,double* a,double* b);
int vectordiff(int n,double* a,double* b,double* c);
int vectorsum(int n,double* a,double* b,double* c);
```

```
int vectormult(int n,double* a,double s,double* c);
double vectornorm(int n,double* a);
int vectorrowprint(int n,double* a);
int crsspr(double* a,double* b,double* c);
double dotpr(double* a,double* b);
double veca3(double* a,double* b);
double dihedral(double* p1,double* p2,double* p3,double* p4);
int seq2angs(double* p1,double* p2,double* p3,double* p4,double* bl,double* ba,dou
int angs2seq(double* p1,double* p2,double* p3,double bl,double ba,double da,double
int orthgm(double* x,double t,double* a,double* a4);
int matrixm(double* a,double* b,int m,int n,int l,double* c);
int main(){
 // double p1[]={1.,0.,0.};
 // double p2[]={0.,0.,0.};
 // double p3[]={0.,1.,0.};
 // double p4[]={0.,1.,1.};
double p1[]={5.,0.,1.};
double p2[]={0.,1.,0.};
double p3[]={0.,4.,1.};
double p4[]={-5.,4.,11.};
 double theta;
 double bl,ba,da;
 double pi = 3.14159265358979;
 //seq2angs(p4,p3,p2,p1,&bl,&ba,&da);
 //printf(" dihedral angle of p4,p3,p2,p1 = %15.8g \n",180.*da/pi);
 seq2angs(p1,p2,p3,p4,&bl,&ba,&da);
 printf(" bond length = %15.8g \n",bl);
 printf(" bond angle = %15.8g \n",180.*ba/pi);
 printf(" dihedral angle = %15.8g \n",180.*da/pi);
 printf(" p4 = ");vectorrowprint(3,p4);
 angs2seq(p1,p2,p3,bl,ba,da,p4);
 printf("  p4 (inverted) = ");vectorrowprint(3,p4);
 return(0);
}
// angs2seq computes the 4th point of a sequence from angles and length
int angs2seq(double* p1,double* p2,double* p3,double bl,double ba,double da,double
 //Input:
```

```
// p1,p2,p3 3d points of a sequence
// bl       length of vector p3p4
// ba       bond angle between vectors p2p3 and p3p4
// da       dihedral angle between the planes p1p2p3 and p2p3p4
//Output:
// p4       fourth point of the sequence
int i;
double q1[3];
double q2[3];
double q3[3];
double n1[3];
double n2[3];
double v[3];
double s;
double v1[3];
double v2[3];
double v3[3];
double a[9],a4[16];
vectordiff(3,p2,p1,q1);
printf(" q1 = ");vectorrowprint(3,q1);
vectordiff(3,p3,p2,q2);
printf(" q2 = ");vectorrowprint(3,q2);
s=vectornorm(3,q2);
if(s > 0.)vectormult(3,q2,1./s,v1);
printf(" v1 = ");vectorrowprint(3,v1);
crsspr(q1,q2,v);
s=vectornorm(3,v);
if(s > 0.)vectormult(3,v,1./s,n1);
printf(" n1 = ");vectorrowprint(3,n1);
//compute n2 as n1 rotated about q2 by angle da
orthgm(q2,da,a,a4);
matrixm(a,n1,3,3,1,n2);
printf(" n2 = ");vectorrowprint(3,n2);
crsspr(n2,v1,v2);
printf(" v2 = ");vectorrowprint(3,v2);
for(i=0;i<3;i++){
 p4[i] = p3[i]+bl*(cos(ba)*v1[i]+sin(ba)*v2[i]);
```

```
 }
 return(0);
}
//c+ seq2angs angles and a length that define the fourth point of a sequence
 int seq2angs(double* p1,double* p2,double* p3,double* p4,double* bl,double* ba,do
 //Input:
 // p1,p2,p3,p4  3d points of a sequence
 //Output:
 // bl           length of vector p3p4
 // ba           bond angle between vectors p2p3 and p3p4
 // da           dihedral angle between the planes p1p2p3 and p2p3p4
 double q1[3];
 double q2[3];
 double q3[3];
 double n1[3];
 double n2[3];
 double v[3];
 double s;
 double u1[3];
 double u2[3];
 double u3[3];
 vectordiff(3,p2,p1,q1);
 printf(" q1 = ");vectorrowprint(3,q1);
 vectordiff(3,p3,p2,q2);
 printf(" q2 = ");vectorrowprint(3,q2);
 vectordiff(3,p4,p3,q3);
 printf(" q3 = ");vectorrowprint(3,q3);
 *bl=vectornorm(3,q3);
 //printf(" ||q3|| = %15.8g\n",*bl);
 *ba=veca3(q2,q3);
 crsspr(q1,q2,v);
 s=vectornorm(3,v);
 if(s > 0.)vectormult(3,v,1./s,n1);
 printf(" n1 = ");vectorrowprint(3,n1);
 crsspr(q2,q3,v);
 s=vectornorm(3,v);
 if(s > 0.)vectormult(3,v,1./s,n2);
```

```
 printf(" n2 = ");vectorrowprint(3,n2);
 vectorcopy(3,n2,u1);
 printf(" u1 = ");vectorrowprint(3,u1);
 s=vectornorm(3,q2);
 if(s > 0.)vectormult(3,q2,1./s,u3);
 printf(" u3 = ");vectorrowprint(3,u3);
 crsspr(u3,u1,u2);
 printf(" u2 = ");vectorrowprint(3,u2);
 *da = -atan2(dotpr(u2,n1),dotpr(u1,n1));
 return(0);
}
//c+ vectorcopy copy a vector from a to b
int vectorcopy(int n,double* a,double* b){
 // the n vector a, is copied to b
 int i;
 for (i=0; i<n; i++){
  b[i] = a[i];
 }
 return(0);
}
//c+ vectordiff vector difference: a-b=c
int vectordiff(int n,double* a,double* b,double* c){
 // The n vector b is subtracted from a, and the difference stored in c
 int i;
 for (i=0; i<n; i++){
  c[i]=  a[i] - b[i];
 }
 return(0);
}
//c+ vectorsum vector sum: a+b=c
int vectorsum(int n,double* a,double* b,double* c){
 // the n vector a is added to b and stored in c
 int i;
 for (i=0; i<n; i++){
  c[i]=  a[i] + b[i];
 }
 return(0);
```

```
}
//c+ vectormult vector multiplied by scalar: a*s=c
int vectormult(int n,double* a,double s,double* c){
 // the n vector a is multiplied by s and stored in c
 int i;
 for (i=0; i<n; i++){
  c[i]=  s*a[i];
 }
 return(0);
}
//c+ vectornorm norm of vector
 double vectornorm(int n,double* a){
 // norm = sqrt(sum a[i]^2, for i=0 to i-1)
 // must include math.h
 int i;
 double s;
 s=0.;
 for (i=0; i<n; i++){
  s = s + a[i]*a[i] ;
 }
 s=sqrt(s);
 return(s);
}
//c+ vectorrowprint vector printed as a row
int vectorrowprint(int n,double* a){
 // the n vector is printed as a row
 int i;
 for (i=0; i<n; i++){
  printf(" %15.8g ",a[i]);
 }
 printf("\n");
 return(0);
}
//c+ crsspr  vector cross product.
int crsspr(double* a,double* b,double* c){
 //prototype int crsspr(double*,double*,double*);
 //Version 01/17/99
```

```
 //c=product of a and b
 c[0] = a[1]*b[2]-a[2]*b[1];
 c[1] = a[2]*b[0]-a[0]*b[2];
 c[2] = a[0]*b[1]-a[1]*b[0];
 return 0;
}
//c+ dotpr scalar product.
double dotpr(double* a,double* b){
 //prototype double dotpr(double*,double*);
 double s;
 int i;
 s = 0.0;
 for(i=0; i<3; i++){
  s = s + a[i]*b[i];
 }
 return s;
}
//c+ veca3   angle between 3d vectors
double veca3(double* a,double* b){
//prototype double veca3(double*,double*);
//Version 2/21/99
 double c[3];
 double cn;
 double sn;
 double ang;
 //External Functions:
 //double sqrt();
 //int crsspr();
 //double atan2();
 //Note: requires #include <math.h>
 crsspr(a,b,c);
 sn=sqrt(c[0]*c[0]+c[1]*c[1]+c[2]*c[2]);
 cn=a[0]*b[0]+a[1]*b[1]+a[2]*b[2];
 ang=atan2(sn,cn);
 return(ang);
}
//c+ orthgm orthogonal matrix (rotation matrix) from axis and angle
```

```
int orthgm(double* x,double t,double* a,double* a4){
 //prototype int orthgm(double*,double,double*,double*);
 // c version modified 6/16/95
 // input:
 // x  three dimensional axis vector
 // t  rotation angle
 //output:
 // a  3 by 3 rotation matrix
 // a4 4 by 4 version of matrix (homogeneous matrix)
 // see:  IEEE Computer Graphics and Applications,
 //Feb. 1984, A Note on Rotation Matrices, Jay Fillmore
 static double id[9] =
 { 1.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,1.0 };
 double l[9],l2[9],lambda,c1,c2;
 int i,j;
 lambda = sqrt(x[0]*x[0]+x[1]*x[1]+x[2]*x[2]);
 for(i=0; i<3; i++){
  l[4*i] = 0.0;
 }
 l[3] = -x[2];
 l[6] =  x[1];
 l[7] = -x[0];
 l[1] = -l[3];
 l[2] = -l[6];
 l[5] = -l[7];
 matrixm(l,l,3,3,3,l2);
 c1 = sin(t)/lambda;
 c2 = (1.0 - cos(t))/(lambda*lambda);
 for(i=0; i<9; i++){
  a[i] = id[i] + c1*l[i] + c2*l2[i];
 }
 for(i=0;i<3;i++){
  for(j=0;j<3;j++){
   a4[i+j*4]=a[i+ j*3];
  }
 }
 a4[3]=0.;
```

```
 a4[7]=0.;
 a4[11]=0.;
 a4[12]=0.;
 a4[13]=0.;
 a4[14]=0.;
 a4[15]=1.;
 return(0);
}
//c+ matml   matrix multiplication (row offset equals row size).
int matml(double *a,double *b,double *c,int m,int n,int l){
//  function        -product of matrices, c=a*b
//  parameters     a-m rows by n columns
//                 b-n rows by l columns
//                 c-m rows by l columns
 int i,j,k;
 for(i=0; i<m; i++) {
  for(j=0; j<l; j++) {
   *(c+i+j*m) = 0.0;
   for(k=0; k<n; k++) *(c+i+j*m) += (*(a+i+k*m)**(b+k+j*n));
  }
 }
 return(0);
}
//c+ matrixm the product of matrices.
int matrixm(double* a,double* b,int m,int n,int l,double* c){
 //prototype int matrixm(double*,double*,int,int,int,double*);
 // c=a*b,
 // a: m rows by n columns
 // b: n rows by l columns
 // c: m rows by l columns
 //fortran addressing: a_{i,j} = a[i+j*m-1]
 int i,j,k;
 for(i=0; i<m; i++){
  for(j=0; j<l; j++){
   *(c+i+j*m) = 0.0;
   for(k=0; k<n; k++){
    *(c+i+j*m) += *(a+i+k*m) * (*(b+k+j*n));
```

```
   }
  }
 }
 return(0);
}
```

0.5 The Definition of a Backbone Sequence With Angles And Bond Lengths

The previous section shows that a sequence

$$\{p_n\}_1^\infty$$

may be completely specified with dihedral angles, bond angles, and bond lengths. That is by the sequence

$$p_1, p_2, p_3, (\theta_4, \rho_4, s_4), (\theta_5, \rho_5, s_5), (\theta_6, \rho_6, s_6), ...$$

Here θ_i is the dihedral angle of $p_{i-3}, p_{i-2}, p_{i-1}, p_i$, bond angle ρ_i is the angle between q_{i-2} and q_{i-1}, and bond length s_i is the length of q_{i-1}.

The first three points of the sequence are needed to specify the absolute location of the sequence in space. The shape independent of absolute location could be specified as

$$(s_2), (\rho_3, s_3), (\theta_4, \rho_4, s_4), (\theta_5, \rho_5, s_5), (\theta_6, \rho_6, s_6), ...$$

In the case of proteins, the ρ_i and s_i are known, so the sequence is defined by the dihedral angles only. In addition every third dihedral angle is constant, so only the dihedral angles for the two bonds to the alpha carbon atoms must be specified. This is discussed more fully below.

0.6 A Reverse Property.

Proposition The dihedral angle $\theta = d(p_1, p_2, p_3, p_4)$ is not changed by reversing the sequence. That is,

$$d(p_1, p_2, p_3, p_4) = d(p_4, p_3, p_2, p_1)$$

Proof. Let

$$p'_1 = p_4, p'_2 = p_3, p'_3 = p_2, p'_4 = p_1.$$

Then

$$q'_1 = p'_2 - p'_1 = p_3 - p_4 = -q_3$$
$$q'_2 = p'_3 - p'_2 = p_2 - p_3 = -q_2$$
$$q'_3 = p'_4 - p'_3 = p_1 - p_2 = -q_1$$

We have

$$q'_1 \times q'_2 = (-q_3) \times (-q_2) = -q_2 \times q_3.$$

Therefore

$$n'_1 = -n_2.$$

Similarly

$$n'_2 = -n_1.$$

The orthogonal basis u'_1, u'_2, u'_3 becomes

$$u'_1 = n'_2 = -n_1$$

$$u'_3 = \frac{q'_2}{\|q'_2\|} = -u_3.$$

$$u'_2 = u'_3 \times u'_1 = (-u_3) \times (-n_1) = u_3 \times n_1.$$

So the components of n'_1 in the u'_1, u'_2 basis are

$$n'_1 \cdot u'_1 = (-n_2) \cdot (-n_1) = u_1 \cdot n_1,$$

and

$$n_1 \cdot u'_2 = (-n_2) \cdot (u_3 \times n_1) =$$

$$-n_2 \times u_3 \cdot n_1 = -u_1 \times u_3 \cdot n_1 = u_3 \times u_1 \cdot n_1 = u_2 \cdot n_1.$$

Therefore

$$\theta' = \text{atan2}(n'_1 \cdot u'_2, n_1 \cdot u'_1)$$
$$= \text{atan2}(n_1 \cdot u_2, n_1 \cdot u_1) = \theta.$$

This result is not especially amazing, because it simply says that a screw thread looks the same from either end of the screw.

0.7 Protein Description

A protein is a chain of amino acids, each of which consists of an amino group NH_2, a carboxyl group $COOH$, and a side chain R. The amino acid is written in symbolic form as

```
 NH2
  |
H-C-COOH
  |
  R
```

There are 20 amino acids making up the proteins of animals. They are:

```
alanine
arginine
asparagine
aspartic acid
cysteine
glutamine
gultamic acid
glycine
histidine
isoleucine
leucine
lysine
methionine
phenylalanine
proline
serine
threonine
tryptophan
tyrosine
valine
```

One can extract a VRML ball and stick model of an amino acid residue by using the program **aminowrl.cpp** on a PDB file that contains the desired residue. Many other amino acids exist. Some of these occur in plants.

A free amino acid is added to a polypeptide chain when the nitrogen of its amino group forms a double bond with the carbon of the last carboxyl group of the original polypeptide chain. This bond is called the polypeptide bond. The carbon forming the bond will be written as a primed carbon, C', in what follows. The nitrogen loses two hydrogen atoms, the primed carbon atom loses an oxygen atom. A water molecule is formed in the polymerization process.

0.8 Protein Backbone

Here is the structure of the polypeptide backbone chain:

$$... = N_n - C_n^\alpha - C'_n = N_{n+1} - C_{n+1}^\alpha - C'_{n+1} = N_{n+2} - C_{n+2}^\alpha - C'_{n+2} = ...$$

There is a double bond between C' and N. This is the polypeptide bond. This bond causes each group

$$C_\alpha - C' = N - C_\alpha.$$

to lie in a polypeptide plane. The bond angles in degrees are

$$\theta_{C'=N-C^\alpha} = 123,$$

$$\theta_{C^\alpha - C' - N} = 114,$$

and

$$\theta_{N-C^\alpha - C'} = 110.$$

The bond lengths in Angstroms are

$$\ell_{C=N} = 1.32,$$

$$\ell_{N-C^\alpha} = 1.47,$$

and

$$\ell_{C^\alpha - C} = 1.53.$$

0.9 Angles Characterizing Protein Structure

The dihedral angle for the bond before the nth alpha carbon is called ϕ_n and is defined by

$$\phi_n = f(C'_{n-1}, N_n, C^\alpha_n, C'_n),$$

where the function f gives the dihedral angle between the plane defined by the first three atom centers, and the plane defined by the last three atom centers.

The dihedral angle for the bond after the nth alpha carbon is called ψ_n, and is defined by

$$\psi_n = f(N_n, C^\alpha_n, C'_n, N'_{n+1}).$$

As we have shown above this chain or backbone sequence of atoms is defined by the bond lengths, the bond angles, and the dihedral angles. The nth residue has three atoms in the chain namely N_n, C^α_n, C'_n and there are three bonds per residue. But we only need two angles ϕ and ψ per residue, because the polypeptide bond is planar and always has a dihedral angle of π. There is a third angle that specifies a dihedral angle at the alpha carbon that determines how each side chain is attached, but we do not deal with that here. See the references for this information.

The primary structure of the protein is the amino acid sequence. The angles ϕ and ψ characterize the secondary structure of the protein. Not all values of these angles occur, because for some values of ϕ and ψ, the atoms of the molecules will interfere. This is called steric hindrance. A plot of the angles is called a Ramachandran map. By plotting these angles for a large sample of proteins the Ramachandran map identifies empty regions, which may be considerd forbidden angles. It thus describes certain forbidden secondary structures.

The tertiary structure consists of substructures such as the α-helix, and the β-sheet. For an α-helix substructure, the sequence of angles pairs (ϕ_n, ψ_n) will be constant and is one method of identifying the α-helix.

The quaternary structure is the folded globular structure. An example is the protein shell of a virus, which often has icosahedral symmetry.

0.10 Specifying a Sequence of Points in Three Space With Dihedral Angles, Bond Angles, and Bond Distances.

Simply as a mathematical problem we see that a sequence of points in three space, which is essentially a piecewise linear curve, can be specified by angles and distances. Suppose we are given a sequence of points $\{p_i\}$ such that no two points p_k, p_{k+1} of the sequence are collinear. Each three points p_k, p_{k+1}, p_{k+2} then determines. a plane. Suppose the distances $d_k = d(p_k, p_{k+1})$ are known for each k, as well as the angles formed by each contiguous pair of edges. Consider the position of three points, $p_n, pn + 1, p_{n+2}$ known. Then from knowledge of the dihedral angle, we have some information about the location of the next point p_{n+3}. We know the plane on which it must lie and we know its distance from p_3. But this is not sufficient to uniquely locate it. But if we also know the angle p_2, p_3, p_4, then p_4 is determined. In the case of the protein backbone, we shall have this information because the angles in the peptide plane are constant. So we can essentially specify the shape of a protein backbone with the sequence by dihedral angles $\theta_3, \theta_4, \theta_5, ...$ together with bond distances $d_3, d_4, ...$, and the known angles of the polypeptide plane. We would need to specify the coordinates of the first three points to locate the sequence in space. In the case of a polypeptide chain, because the atoms in the polypeptide plane always lie in a plane and in fixed orientation, one of every sequence of three in the chain have fixed dihedral angles, so we only need angles at the two bonds of the chain before and after the C^α. This sequence of angles specifies the shape of the protein because the bond lengths are constant. We can also "invent" trial proteins by specifying a set of angles, and then study, calculate energy, and view the shape using computer graphics. Given the angles of a protein we can examine whether we are at a state of minimum free energy by altering one or more of the angles. We would of course require a method of specifying the orientation of the R's attached to the alpha carbons, because they would have to be involved in the energy calculation.

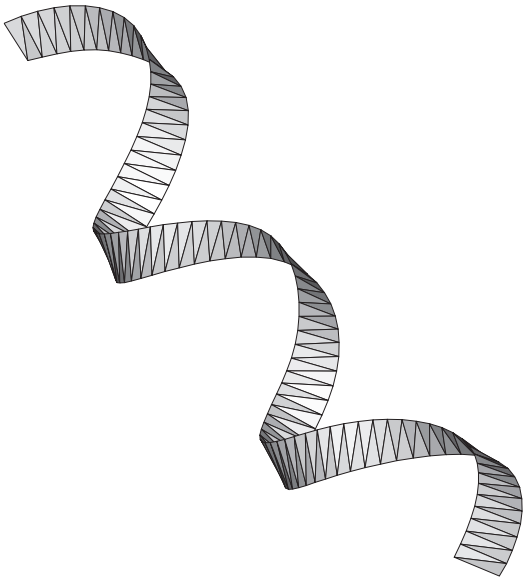

Figure 7: The first helix of the ribbon model of la.net with flat shading.

0.11 The Ribbon Model of a Protein

Proteins can be modeled in several ways, especially in terms of computer graphics. A protein has a one dimensional model defined by the lines joining the alpha carbon atoms. A two dimensional model could consist of the sequence of polypeptide planes. We can get a continuous ribbon model by taking a line segment lying in the polypeptide plane and passing through the center of the plane. We can interpolate lines between each pair of lines to get a ribbon generated by sweeping these interpolated lines. We could use the plane normals to get these lines and then interpolate between the lines using smooth interpolation functions such as splines.

More specifically we can specify a two dimensional ribbon that has the property of being tangent to each polypeptide plane. Suppose we specify a center and path direction at each polypeptide plane in the following way. The direction is specified by the line segment lying in the polypeptide plane that joins C_n^α to C_{n+1}^α. The center is taken to be the midpoint of the line. We can define a generating path $p(t)$ as a piecewise cubic curve that has continuous tangent so that it passes through each center and has a velocity

in the direction of the line segment. We may give it unit velocity at each center interpolation point. Such a curve is known as a Hermite cubic. Also we can create a cubic spline curve $n(t)$ to smoothly interpolate polypeptide plane normals. These two curves can be parameterized by knot count. Let t be a parameter. Then $p(t)$ specifies a point on the center of the ribbon at parameter t and a tangent direction $p'(t)$, and $n(t)$ gives us a ribbon normal. From these points and vectors, together with a ribbon width, we can obtain a generating line for the ribbon. We would normally triangulate the ribbon for graphic display. One can examine the code for the program **proteinbb.cpp** to see how the ribbon model has been computed in practice, and how the VRML data has been generated. That program can also write triangles for the ribbons, which can be processed by a Z-buffer algorithm to create a traditional graphics surface file.

0.12 X-Ray Diffraction

Protein structure is determined by first crystallizing the protein and then using either X-Ray diffraction, or NMR (Nuclear Magnetic Resonance) methods. The protein must be crystallized so that the crystal has a regularly repeating structure. In the case of X-Ray diffraction, the regularly repeating patterns of electron density scatter the X-Rays in well defined directions to produce diffraction patterns. These diffraction patterns are inverted, using Fourier techniques, to reconstruct the electron densities. High electron densities correlate to atomic positions. The electron densities obtained are compared with the electron densities due to an assumed model. The model is adjusted until there is good correlation. In the early days the model was a physical model and the comparison was done physically by plotting the electron densities on transparencies and viewing the model through the transparencies. Now computer models are used for this purpose.

The electron density of a hydrogen atom is too small to produce observable diffraction. The position of hydrogen atoms can not be established by X-Ray diffraction methods.

The position of the hydrogen atom in the polypeptide plane may be computed from the positions of the other atoms in the plane. The relative position of the hydrogen atom has been be determined by a different experimental method also. However, NMR can locate hydrogen positions.

The computation of the position of the hydrogen atom in the polypeptide plane is given in the next section. We need the position of the hydrogen atom H_n, because it forms the hydrogen bond that results in the secondary structures of the alpha helix and the beta sheet.

0.13 Locating the Hydrogen Atom in the Polypeptide Plane

Suppose we are given information about the $n-1$th and the nth residues of a protein Specifically, suppose we are given three atoms of the protein backbone, namely C_{n-1}, N_n, and C_n^α. We shall reconstruct the position of the hydrogen H_n, which is bound to N_n. Let p_1 the position of C_{n-1}, p_2 be the position of N_n, p_3 the position of C_n^α, and p_4 the position of H_n. Let $r_1 = p_2 - p_1$ and $r_2 = p_3 - p_2$. Let us construct an orthonormal basis u_1, u_2, u_3, where u_1 is parallel to r_1, u_2 is perpendicular to u_1 and lies in the polypeptide plane, and u_3 is normal to the polypeptide plane. To do this, let

$$u_1 = \frac{r_1}{\|r_1\|}.$$

Let

$$u_3 = -\frac{r_1 \times r_2}{\|r_1 \times r_2\|}$$

and

$$u_2 = u_3 \times u_1.$$

Notice that we take the minus sign in the definition of u_3 because H_n is trains to C_n^α across bond $C_{n-1} - N_n$, and we want u_2 to point towards H_n.

Then after Pauling, taking the bond length $N_n - H_n$ to be 1, and the bond angle $C_{n-1} - N_n - H_n$ to be 126 , we can locate the position p_4 of hydrogen H_n. Thus taking θ to be the supplement of the bond angle, namely $\theta = 54$ degrees, we find that

$$p_4 = p_2 + r(\cos(\theta)u_1 + \sin(\theta)u_2),$$

where r is the bond length, which is taken to be 1 Angstrom here.

0.14 Making a Physical Model of the Polypeptide Plane

To make a physical model of each polypeptide plane (as in the olden days), we need the x, y, z coordinates of the centers of the six atoms that lie in each polypeptide plane. We would of course use units like centimeters in place of Angstroms. So we could model each polypeptide plane with an index card, and link the cards together with pivots of some kind. Let
p_1 be the position of the first alpha carbon atom C_n^α,
p_2 the position of the primed carbon atom C_n',
p_3 the position of the nitrogen N_{n+1},
p_4 the position of the second alpha carbon atom C_{n+1}^α,
p_5 the position of the oxygen atom O_n, and
p_6 the position of the hydrogen atom H_{n+1}.

We may define vectors connecting the atoms of the polypeptide plane. Let

$$q_1 = p_2 - p_1,$$

$$q_2 = p_3 - p_2,$$

$$q_3 = p_4 - p_3,$$

$$q_4 = p_5 - p_2,$$

$$q_5 = p_6 - p_3.$$

Then

$$p_1 = p_2 - q_1$$

$$p_3 = p_2 + q_2$$

$$p_4 = p_3 + q_3$$

$$p_5 = p_2 + q_4$$

$$p_6 = p_3 + q_5$$

The atomic position vectors lie in a two-dimensional vector space, so from knowledge of their relative positions we give them the following coordinates

$$p_2 = [0, 0],$$

$$q_1 = p_2 - p_1 = 1.53[\cos(-66\,\mathrm{deg}), \sin(-66\,\mathrm{deg})],$$

$$q_2 = p_3 - p_2 = 1.32[1, 0],$$
$$q_3 = p_4 - p_3 = 1.47[\cos(-57\,\text{deg}), \sin(-57\,\text{deg})],$$
$$q_4 = p_5 - p_2 = 1.24[\cos(-125\,\text{deg}), \sin(-125\,\text{deg})],$$
$$q_5 = p_6 - p_3 = 1.0[\cos(57\,\text{deg}), \sin(57\,\text{deg})].$$

We shall write each vector p_i in terms of p_2, q_2, and q_3. Suppose we are given vectors r_1 and r_2 which are linearly independent. Then given vector r_3 in the space spanned by these linearly independent vectors, we may write it as

$$r_3 = ar_1 + br_2.$$

We have

$$ar_1[1] + br_2[1] = r_3[1],$$
$$ar_1[2] + br_2[2] = r_3[2].$$

So that

$$a = \frac{r_3[1]r_2[2] - r_2[1]r_3[2]}{r_1[1]r_2[2] - r_2[1]r_1[2]}$$
$$b = \frac{r_1[1]r_3[2] - r_3[1]r_1[2]}{r_1[1]r_2[2] - r_2[1]r_1[2]}.$$

Hence we find that

$$q_1 = aq_2 + bq_3 = .7527224378q_2 + .5047728570q_3$$
$$q_4 = aq_2 + bq_3 = -1.038537570q_2 + .8239056243q_3$$
$$q_5 = aq_2 + bq_3 = .8252106593q_2 - .6802721091q_3.$$

Then

$$p_1 = p_2 - q_1$$
$$p_3 = p_2 + q_2$$
$$p_4 = p_2 + q_2 + q_3$$
$$p_5 = p_2 + q_4$$

and

$$p_6 = p_2 + q_2 + q_5.$$

p_6 is the position of the hydrogen atom, which we see is defined by the other atoms in the polypeptide plane.

In the polypeptide plane we can give the atoms the following coordinates.

$$p_1 = [\,-.6223070630\, 1.397724551\,]$$

$$p_2 = [\,0\, 0\,]$$

$$p_3 = [\,1.32\, 0\,]$$

$$p_4 = [\,2.120619381\, -1.232845735\,]$$

$$p_5 = [\,-.7112347811\, -1.015748535\,]$$

$$p_6 = [\,1.864639035\, .8386705681\,]$$

0.15 A Maple Program For The Polypeptide Plane Computations

Here is an example of using the computer algebra program Maple to verify a simple calculation. This example is a verification of the calculation of the previous section. That is by running this program in Maple, we may verify the numbers of the previous section.

```
with(linalg):
pi:=evalf(Pi);
p2:=array([0,0]);
q1:=array([1.53*cos(-66*pi/180),1.53*sin(-66*pi/180)]);
q2:=array([1.32*cos(0),1.32*sin(0)]);
q3:=array([1.47*cos(-57*pi/180),1.47*sin(-57*pi/180)]);
q4:=array([1.24*cos(-125*pi/180),1.24*sin(-125*pi/180)]);
q5:=array([cos(57*pi/180),sin(57*pi/180)]);
cf:=proc(r1,r2,r3)
global a,b;
a:=evalf((r3[1]*r2[2] - r2[1]*r3[2])/(r1[1]*r2[2] - r2[1]*r1[2]));
b:=evalf((r1[1]*r3[2] - r3[1]*r1[2])/(r1[1]*r2[2] - r2[1]*r1[2]));
end;
```

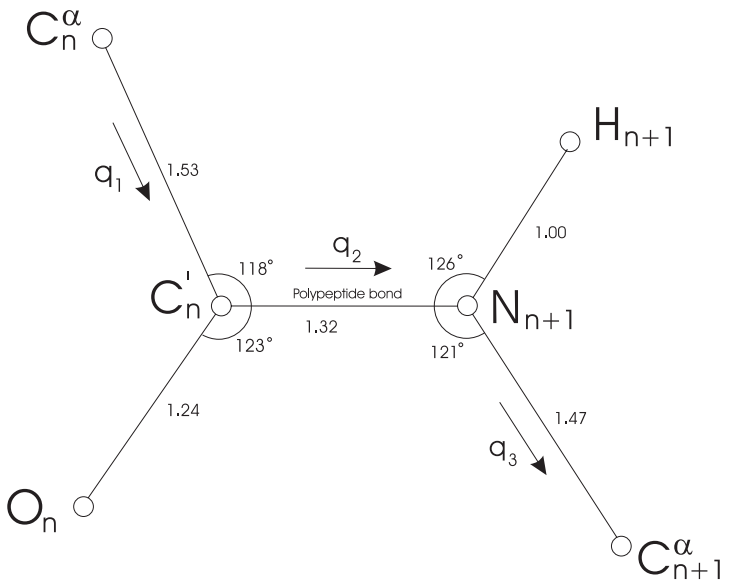

Figure 8: The polypeptide plane. The backbone sequence includes the nth alpha carbon C_n^α, the nth primed carbon C'_n, the $n+1$ nitrogen N_{n+1}, and the $n+1$ alpha carbon C_{n+1}^α. These form the point sequence p_1, p_2, p_3, and p_4. The atoms lie in a plane. The bond lengths are given in Angstroms, and the bond angles in degrees. Because the atoms lie in a plane, the dihedral angle for the polypeptide bond is equal to π.

```
cf(q2,q3,q1):
#q1 = a q2 + b q3
'a'=a;
'b'=b;
cf(q2,q3,q4):
#q4 = a q2 + b q3
'a'=a;
'b'=b;
cf(q2,q3,q5):
#q5 = a q2 + b q3
'a'=a;
'b'=b;
'p1'= evalm(p2 - q1);
'p3'= evalm(p2 + q2);
'p4'= evalm(p2 + q2 + q3);
'p5'= evalm(p2 + q4);
'p6'= evalm(p2 + q2 + q5);
```

0.16 The Protein Folding Problem

Given an amino acid sequence, the protein folding problem is this: to find the conformation or shape of the protein from only the amino acid sequence. Each protein appears to fold in one and only one way. Otherwise I suppose biology would not be possible. Yet this is very strange, because it suggests that there are no local minimum states in folding. One possible technique of finding the folded shape, would be to vary values of the angles $\{(\phi_n, \psi_n) : n = 1, 2, 3, ...\}$ and find a position of minimum electrostatic energy $f(\phi_1, \psi_1, \phi_2, \psi_2, ...)$. This could be attempted with a supercomputer. However, it is not easy to distinguish a local minimum from a global minimum. Also the possible number of shapes is huge. The charge distributions are not easy to calculate. In practice such a calculation does not necessarily lead to the observed shape of the protein. The protein folding problem is an open research problem. For an interesting historical discussion of the problem as of 1994 (which is several years ago), see the paper by Baldwin in the bibliography. There are no doubt several good current research summaries of protein folding.

0.17 A Program List

Programs for calculating angles and creating VRML graphics for ribbon and ball and stick models of a protein are

proteinbb.cpp, molecule.cpp, cubicspline.cpp, parametriccubic.cpp, readmol.cpp, getamino.cpp, aminowrl.cpp. Some of these are class definitions. See the code or run the executables for information on them. A slick windows program might be constructed for the computations described in this document. This would probably be done in Borland C++ builder. VRML files may be viewed with Cosmo Player a VRML plug in for a browser such as Internet Explorer. The Microsoft VRML viewer can also be used, but it does not function well with very big models. The controls of Cosmo player are not very obvious, and documentation is lacking. To go to the examination mode, one must click the little circular button to the left of center. Then the center button rotates nicely. There is a bigger button to the left of the little circular button, which when clicked allows one to use the mouse to bring a point in the image up close and make it the examination center of rotation.

0.18 Bibliography

[1]**IUPAC-IUB Commission on Biochemical Nomenclature,1969** Biochemistry 9, 3471-3479, 1970. A Description of the standards for conformation and angle definitions of polypeptide chains.
[1]Kabsch Wofgang, Sander Christian, **Dictionary of Protein Secondary Structure: Pattern Recognition of Hydrogen-Bonded and Geometrical Features**, Bipolymers, v22, 2577-2637 (1983). HELIX records are now being generated, automatically by the PDB using the Kabsch and Sander algorithms.
[3]Pauling L, Corey R. B., Branson H R, **The Structure of Proteins: Two Hydrogen Bonded Helical Configurations of The polypeptide Chain**, Proceedings of the National Academy of Sciences, volume 37, 1951, pp205-211. The first announcement of the alpha helix and the solution to the structure of proteins.
[4]Hager Thomas, **Force of Nature: The Life Of Linus Pauling**, Simon and Schuster, 1995, Chapter 16 gives a history of the discovery of the alpha helix..

[5]Schulz G E, Schirmer R H, **Principles of Protein Structure**, Springer-Verlag, 1975.
[6]Baldwin Robert L, **Pathways and Mechanisms of Protein Folding**, Contained in, **Statistical Mechanics, Protein Structure, and Protein Substrate**, Doniach Sebastian (Editor), Plenum Press, 1994.
[7]Flory P J, **Principles of Polymer Chemistry**, Cornell University Press, 1971.
[8]Flory P J, **Statistical Mechanics of Chain Molecules**, Interscience, 1969