[2] Modeling and Studying Proteins with Molecular Dynamics

By ROBERT SCHLEIF

Introduction

Structure determines function, but it is not easy to deduce many of the activities or properties of proteins by mere visual inspection of their structures. We do not even know which questions to ask about structure so as to gain the most insight into the behavior of a protein. It does seem likely, though, that answering questions involving geometry, energetics, and atomic motions will be important in understanding protein function. Typical questions that might be asked are as follows: Which residues interact with each other, and how strongly? How exposed to solvent is a particular residue? Is a part of the protein held rigidly or loosely in position with respect to the rest of the protein? How much change in the structure will be generated by a particular mutation? If a mechanism has been postulated for the function of a protein, how would a particular mutation alter that function?

Quite a number of graphical display computer programs are capable of answering geometric questions about proteins, using as input the atomic coordinates as determined by X-ray diffraction, nuclear magnetic resonance (NMR), or model building. Some of these programs can, in addition, perform energetics calculations, using the atomic coordinates and the identities of the atoms. Finally, still more powerful programs can perform geometric and energetics calculations as well as allow movement of atoms in response to interatomic forces. In this third group of programs, atoms may be allowed to move in the course of energy minimization, or all the atoms in a system may be given velocities characteristic of a chosen temperature, and the trajectory of each atom in response to all the forces acting on it may be calculated for a large number of short time steps. This latter technique simulates what we think are the atomic motions in a system, although we call it molecular dynamics. The value of molecular dynamics simulations in understanding the activities of proteins remains to be determined, but model building and calculations concerning proteins of known structure or proteins not too distantly related to proteins of known structure that involve geometry, energetics, and that allow movement of atoms, seem almost certain to be indispensable. Therefore, it seems likely that growing numbers of biochemists, molecular biologists, and biophysicists

0076-6879/04 $35.00

will choose to include in their arsenal of research tools the capability of using a molecular dynamics program. With one program, they will be able to handle a wide variety of calculations.

Several commercial programs for model building, structure analysis, and molecular dynamics that run on workstations are available.* These programs smoothly combine excellent graphical display with model-building facilities, powerful protein structure analysis tools, and computational facilities. The core of much of the computational work done by both programs is the molecular dynamics program CHARMM (Chemistry at Harvard, Molecular Modeling).[1,2] Some scientists in need of these facilities may, however, prefer to do molecular dynamics calculations on a personal computer and without the front end that is provided by the commercial programs. Using a molecular dynamics program in this way does require the writing of scripts, which are primitive programs, but in return the total costs are low (just the personal computer and the nominal the cost of the CHARMM program) and computational versatility is high, and considerable safety is gained because the program is directly controlled rather than being controlled through an interface that freely allows the performance of meaningless calculations. By the time the user can make CHARMM itself do a calculation, the user, more likely than not, knows enough to direct it to perform sensible calculations.

The molecular dynamics programs Amber,[3] CHARMM,[1,2] Gromos,[4] and NAMD[5] all appear suitable for the types of calculations outlined above, although the strengths of each program are slightly different. Because the four programs share much in philosophy, functions, features, and some of their file structures, much of what is true of one program is likely to be true of the others. The remainder of this chapter focuses on CHARMM. This program has been continuously upgraded and expanded by scores of experts over the years, and its availability, accuracy, and

* Quanta and Insight are available from Accelrys (San Diego, CA).

[1] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus, *J. Comput. Chem.* **4,** 187 (1983).

[2] A. D. MacKerell, Jr., B. Brooks, C. L. Brooks III, L. Nilsson, B. Roux, Y. Won, and M. Karplus, *in* "The Encyclopedia of Computational Chemistry" (P. V. R. Schleyer, N. L. Allinger, T. Clark, J. Gasteiger, P. A. Kollman, H. F. Schaefer III, and P. R. Schreiner, eds.), Vol. 1, p. 271. John Wiley & Sons, Chichester, 1998.

[3] D. A. Pearlman, D. A. Case, J. W. Caldwell, W. S. Ross, T. E. Cheatham III, S. DeBolt, D. Ferguson, G. Seibel, and P. Kollman, *Comput. Phys. Commun.* **91,** 1 (1995).

[4] W. R. P. Scott, P. H. Hünenberger, I. G. Tironi, A. E. Mark, S. R. Billeter, J. Fennen, A. E. Torda, T. Huber, P. Krüger, and W. F. van Gunsteren, *J. Phys. Chem. A* **103,** 3596 (1999).

[5] L. Kalé, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, and K. Schulten, *J. Comput. Phys.* **151,** 283 (1999).

reputation are all well known. Because it runs on a wide variety of Unix computers, as well as on inexpensive personal computers running Linux, it is widely used.

Most of the authors of chapters in this volume are producers rather than consumers. That is, they originated and wrote the computer programs they write about. The author of this chapter, however, is solely a consumer. For many years research in the author's laboratory has been centered on regulation of the L-arabinose operon in *Escherichia coli,* and lately it has been directed at understanding the atomic details in its regulatory protein, AraC. It became apparent that computational abilities were needed to help analyze and guide the biochemical and molecular genetics approaches chosen. This chapter describes the author's experience in using CHARMM for this purpose.

Sampling of CHARMM Capabilities

Below are listed a few examples of protein structure–function questions that can be addressed with a program such as CHARMM.

Simple Structural Questions

- Simple geometric quantities, such as distance between two atoms, the angle between three atoms, the dihedral angle between four atoms, locations of holes in a protein, or the surface area of a protein, can be determined.
- The interaction energies between any atoms or residues and any other atoms or residues can be calculated.
- Complex selections can be included in quantities to be calculated. For example, it is possible to calculate the strength of the electrostatic interactions between the backbone atoms of a portion of a protein and the solvent.
- The data for residue–residue contact maps and residue–residue interaction energy maps can easily be calculated.
- The solvent exposure of each residue with and without ligand present or the relative solvent accessibility of each residue in the protein can easily be determined.

Model-Building Operations Possible

- Simple molecules such as a carbohydrate or complex molecules such as a protein can be ''constructed'' *de novo* and studied computationally in the same way as molecules whose representation in the computer is based on a real structure.

- It is possible to ''mutate'' one or more of the residues in a protein and to determine the subsequent structural accommodation of the rest of the protein to the change.
- The rotameric state of the side chain of a residue can be adjusted.
- A peptide or protein or ligand can be moved around in space.
- Any set of atoms can be fixed in space and another part of the peptide or protein can be pulled to any desired position, all the while allowing all the unfixed atoms to move subject to forces exerted by existing bonds and nonbonding interactions.
- Two peptides or proteins can be fused.
- The homologous regions of two different proteins can be overlaid through RMS fitting.

Operations That Involve Molecular Dynamics

- Dynamics of a protein can be run in a shell of water or in an infinite array with periodic boundary conditions. Any temperature can be simulated. When periodic boundary conditions are used, simulations can be done at constant pressure, constant temperature, or constant volume.
- From a molecular dynamics trajectory, it is possible to determine average structure and determine RMS fluctuations in atom positions. This can locate flexible and rigid portions of a protein. The fluctuations can be assigned to the *B* value or the crystallographic temperature factor, written to an output file in the Protein Data Bank[6] (PDB) format, and the flexibility of the protein can be visualized with a molecular display program that colors the backbone according to the *B* value.
- It is possible to see how the structure and interactions in a protein respond to a mutational change in a residue or multiple residues at physiologically meaningful temperatures in the presence of water.
- The energy of the system or any interaction energy can be determined for any or all of the frames of a dynamics trajectory.
- During a molecular dynamics run, it is possible to gently pull on a part of the protein so as to generate a desired conformational change. The work done in generating the conformational change can then be calculated.
- The CHARMM program can also carry out more advanced operations such as Monte Carlo studies, Poisson–Boltzman

[6] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, *Nucleic Acids Res.* **28,** 235 (2000).

calculations, and normal mode analysis; determine free energy differences by thermodynamic integration or perturbation techniques; and examine issues of protein folding and denaturation.

Program Operation Basics

Molecular Dynamics: Theory

We deal with atoms that have reasonably high mass compared with electrons, at reasonably high temperatures, and we consider reactions that do not include the making or breaking of chemical bonds. Therefore, reasonably accurate results can be obtained by classic mechanics rather than quantum mechanics and by considering the locations of the atomic nuclei rather than the densities of electron clouds. The basis of the calculations can then be Newton's equation of motion as shown in Eq. (1):

$$F_i = m_i a_i \tag{1}$$

where F_i is the sum of forces acting on the ith particle, m_i is the mass of the ith particle, and a_i is the acceleration of the ith particle. A molecular dynamics program is said to integrate these equations, one for each particle. The word *integrate* as used here does not mean to find the mathematical expressions for particle motions that solve the differential equations; rather, it means to determine the numerical values of the coordinates of each particle at successive, closely spaced, time points, all the while obeying Newton's equation of motion. In practice, these time points are about 10^{-15} s apart.

The principle of numerical integration is that given the position $x_i(t_0)$ and velocity $v_i(t_0)$ of the ith particle at the starting time t_0, the position and velocity of the particle a short time later, at t_1, are given by

$$x_i(t_1) = x_i(t_0) + v_i(t_0) \times (t_1 - t_0) \tag{2}$$

$$v_i(t_1) = v_i(t_0) + \left.\frac{dv_i(t_0)}{dt}\right|_{t_0} \times (t_1 - t_0) \tag{3}$$

(3) becomes the following from Newton's equation of motion,

$$v_i(t_1) = v_i(t_0) + \frac{F_i(t_0)}{m_i} \times (t_1 - t_0) \tag{4}$$

Hence, the positions and velocities of all the particles at time t_1 can be calculated from their values at time t_0. The total force acting on each particle, F_i, depends on the positions of all the other particles in the system and, in addition, may include externally imposed forces. In actual practice, the

equations used by molecular dynamics programs are slightly different but are still based on the principles outlined above.

A crucial component in the trajectory calculations is the force that is felt by each atom. A precise representation of the force may yield simulations that closely approximate nature, but they take impossibly long to compute. Alternatively, representations of the forces that lead to rapid computation may be poor approximations of the real forces. Different molecular dynamics programs choose different compromises to this difficult tradeoff. The potential function used by CHARMM, from which energies and the forces needed for trajectory calculations are derived, approximates the forces on an atom as the sum of its pairwise interactions with the other atoms in the system.[7] The energy of an atom is approximated as the sum of bond stretching, bond bending, bond twisting, improper torsions (which act to maintain planar bonds), and van der Waals, and electrostatic interactions. The parameters describing these terms are tabulated and read in when CHARMM starts. Much of the work of developing a molecular dynamics program goes into determining the best possible values for the many terms in a parameter table. These potential functions used by CHARMM are approximated as follows.[7]

Bond-stretching energy is approximated as

$$V_{\text{bond}} = K_{\text{b}}(b - b_0)^2 \tag{5}$$

where K_{b} is a constant that depends on the two atoms sharing the bond, b is the length of the bond, and b_0 is the unstrained bond length. The energy in the bond or the force exerted by the bond is approximated as depending on the coordinates of only the two atoms sharing the bond and on the values of the constant K_{b}, which depend on the types of atoms that are involved. These and other constants used to describe the other atomic interactions are obtained by a combination of *ab initio* calculations and hand fitting of results obtained by simulation of small model compounds to physical measurements.[7]

Bond-bending energy is approximated as

$$V_{\text{angle}} = K_{\theta}(\theta - \theta_0)^2 \tag{6}$$

where K_{θ} is a constant that depends on the three atoms defining the angle, θ is the angle, and θ_0 is the unstrained angle.

[7] A. D. MacKerell, Jr., D. Bashford, M. Bellott, R. L. Dunbrack, Jr., J. D. Evanseck, M. J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F. T. K. Lau, C. Mattos, S. Michnick, T. Ngo, D. T. Nguyen, B. Prodhom, W. E. Reiher III, B. Roux, M. Schlenkrich, J. K. C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiórkiewicz-Kuczera, D. Yin, and M. Karplus, *J. Phys. Chem. B* **102,** 3586 (1998).

Bond twisting requires four atoms, A, B, C, and D, to define the rotation or torsion angle of the B–C bond. This term is also known as the dihedral energy, and it is approximated as

$$V_{\text{dihedral}} = K_\chi[1 + \cos(n\chi - \delta)] \tag{7}$$

where K_χ and δ are constants that depend on the adjacent atoms, n is an integer (2, 3, 4, or 6) that depends on the four atoms, and χ is the value of the dihedral angle between the planes defined by atoms A, B, and C, and by atoms B, C, and D. Thermal energies are usually sufficient to shift some of these bonds from one energy minimum to another. The term provides a preference for the torsion angle to place the atoms bonded to atoms B and C in noneclipsed positions, and describes the different rotameric states of the side chains of amino acids.

A group consisting of one central atom, A, that is bonded to three other atoms, B, C, and D, can be held in a particular configuration by what is called an improper energy term,

$$V_{\text{improper}} = K_\psi(\psi - \psi_0)^2 \tag{8}$$

where K_ψ and ψ_0 are constants and ψ is the angle between the plane containing atoms A, B, and C and the plane containing atoms B, C, and D. Most often an improper dihedral term is used to hold a set of four atoms in a plane. In contrast to the regular dihedrals, the energy constants for improper dihedrals are large enough to prevent substantial deviation of ψ from ψ_0.

The van der Waals interaction between two atoms is approximated with a Lennard–Jones potential as

$$V_{\text{Lennard–Jones}} = \varepsilon_{i,j}\left[\left(\frac{R_{\text{mi},i,j}}{r}\right)^{12} - 2\left(\frac{R_{\text{mi},i,j}}{r}\right)^{6}\right] \tag{9}$$

where

$$\varepsilon_{i,j} = \sqrt{\varepsilon_i \times \varepsilon_j} \tag{10}$$

and where ε_i and ε_j are constants characteristic of the two atoms,

$$R_{\text{mi},i,j} = \frac{R_{\text{mi},i}}{2} + \frac{R_{\text{mi},j}}{2} \tag{11}$$

where $R_{\text{mi},i}$ and $R_{\text{mi},j}$ are also constants characteristic of the two atoms, and r is the distance between the centers of the two atoms.

Finally, the electrostatic interaction between two atoms is

$$V_{\text{electrostatic}} = \frac{q_i q_j}{4\pi\varepsilon r} \tag{12}$$

where q_i and q_j are the charges of the two atoms, r is their separation, and ε is the dielectric constant. Ordinarily, energy calculations and simulations are performed with the protein immersed in water molecules. The parameter tables in CHARMM have been developed for this condition and with ε set equal to 1.

Program Operation: Practice

Because a molecular dynamics program is capable of generating a path of motion for every atom of a protein and a surrounding bath of water molecules, it needs much input information. Some of this information comes from a file of atomic coordinates that have been determined by X-ray crystallography, NMR, or model building. Such files provide the IUPAC names of the atoms in the residues and provide the positions of the atoms in space. Of course, coordinates that are determined by X-ray crystallography lack positions for the hydrogen atoms, and the program must place such atoms on its own.

In addition to positional information, a program needs to know every chemical bond in a structure, the bending, stretching, and twisting strengths of these bonds, the strengths of the improper restoration forces, the radius of each type of atom, and the partial electric charge on each atom. These latter classes of information are made available to the CHARMM program in the form of two types of table. One table is called the residue topology file. This contains a list of the amino acid residues and/or nucleotide bases and commonly encountered small molecules. It lists the bonding pattern for every atom in a residue or base, the partial charge of every atom, the IUPAC name of each atom, and the CHARMM atom type for the same atom. The topology file also lists the geometric position of each atom with respect to other atoms to which it is bonded. This local coordinate system is called the internal coordinate system. It allows deduction of the likely positions of atoms whose coordinates may be missing from the input coordinate files and greatly simplifies some coordinate manipulation operations. The second table of information used for input of information to CHARMM is the parameter table. This contains the masses of the atoms, the stretching, bending, and twisting strengths of the chemical bonds, the strength of improper forces, the parameters recommended for determining how the interaction force between two atoms will be calculated, and the strengths of van der Waals forces and radii for all atom types.

From the input coordinates, the residue topology file, and the parameter file, the program can construct two files specific to a protein. These are used as the program performs further operations on the protein such as energy determination, energy minimization, alteration of residues, structure

modification, or the running of dynamics simulations. One of the protein-specific files is a list of the coordinates of every atom in the system. The second is known as the principal structure file. It lists every residue in the protein, lists each atom, provides the type of each atom, and gives the mass and partial charge of each atom. It also lists every pair of atoms connected by a covalent bond, each atom triplet that defines an angle energy term, each set of four atoms defining a dihedral energy term, each set of four atoms that define an improper angle, the hydrogen bond donors, and the hydrogen bond acceptors. Splitting the files that describe a protein like this is logical, as the information in the principal structure file of a protein almost never changes, whereas that in the coordinate file changes with many operations and during a dynamics simulation.

In addition to the forces that are generated by the bonds between atoms, electrostatic and van der Waals forces also act on atoms. In principle, every atom in a system interacts with every other atom through these two types of forces. Because calculating every one of these forces would be far too time consuming and because these forces become negligible at longer separation distances, atom pairs are considered for calculation of these forces only if their separation is less than a user-set cutoff distance. Molecular dynamics programs maintain and constantly update this nonbonded list and then calculate the interactions between only those pairs on the list.

CHARMM is run by commands that are entered at a command prompt. For a reasonably complex calculation, the number of commands required is too great for practical keyboard entry. Instead, commands are written into command files called scripts. These are directed to the program by the Linux or Unix redirect command <. Similarly, CHARMM produces voluminous output, and usually this is directed into another file that is examined after the run with a text editor. Hence, CHARMM is typically invoked at the command prompt as charmm <script.inp>script.out. Data are often extracted from the output files with the Linux and Unix commands grep and awk. The resulting output can then be imported into spreadsheet programs and analyzed. Files containing coordinates and coordinate trajectories that are produced by CHARMM often are also used as input to molecular display programs.

Example Analysis

To communicate some of the flavor of using CHARMM, the steps necessary to run a dynamics simulation of AraC protein, a transcriptional regulator of the L-arabinose operon in *Escherichia coli,* are provided and described. Suppose we have just obtained the structure of its arabinose-binding

domain in the presence of arabinose and this is available in a file in PDB format. Examination of the protein with a molecular display program such as Rasmol or VMD shows that arabinose is bound inside the protein and that the N-terminal 15 amino acids of the protein close over the arabinose. It seems of interest to learn the strength of interaction of each residue with arabinose to compare with those of mutated sites leaving the protein unable to respond to arabinose, but still capable of folding. Because protein structures fluctuate and the energetics of a static structure *in vacuo* might be highly deceiving, it seems best to determine the average strength of interaction of each residue with the arabinose at 300 K while the protein is surrounded with water. The following sections show the steps required to perform such a calculation.

Preparation of Input Coordinate Files

CHARMM usually cannot directly use PDB files containing atomic coordinates, and several small modifications are necessary. These can be made with an editor, or with a word processor, and saved in ASCII format. When many files must be modified, it is convenient to write a file that directs the actions of the Linux line-editing program called awk to make the modifications. A word of warning concerning the transfer of files between Windows and Linux–Unix: on occasion a problem is caused by the fact that Windows files end lines with two characters, a carriage return, and a new line character, whereas Linux and Unix files end lines with just the new line character. Sometimes a file that originated on a Windows system and has been transferred to a Linux system looks perfectly good, but the nonprinting and nonapparent carriage return characters present at the ends of lines cause problems for CHARMM. These characters can be removed from a file named *dirty* and written to a file named *clean* with the Linux command translate, tr -d '\r' <dirty> clean. The option -d instructs translate to delete, and '\r' is the trouble-making return character that is to be deleted.

We begin with the PDB file for AraC, named 2ARC.[8] This contains the coordinates of both subunits of the dimeric protein, two molecules of arabinose, and tightly bound water molecules. Although coordinates are given for residues 7 through 167 of one subunit and through residue 170 of the other subunit, this example uses only residues 7 through 120, as the remainder of the residues form a subdomain that is well separated from arabinose. It proves simplest to split the input PDB file into separate files,

[8] S. M. Soisson, B. MacDougall-Shackleton, R. Schleif, and C. Wolberger, *Science* **276,** 421 (1997).

one for each segment of structure. Thus, in this case, one file contains the coordinates of residues 7 through 120, the second contains the coordinates of the first arabinose molecule, and the third contains the coordinates of the oxygen atoms of the crystallographic water molecules. Although much useful information is contained in the header lines, and they should be read, they need not be included with the separated files.

The header lines of 2ARC mention that alternative conformations are present for a number of the residues. The alternatives must be removed as CHARMM requires only a single coordinate value for each atom. Thus lines such as

```
ATOM  149 N AGLU A 27   14.382 47.461 27.991 0.63 14.88    N
ATOM  150 N BGLU A 27   14.433 47.417 27.971 0.39 14.03    N
```

must be replaced with single lines:

```
ATOM  149 N GLU  A 27   14.382 47.461 27.991 1.00 14.88    N
```

All residue names HIS must be changed to HSD. Even though the charge state of some residues may in reality flicker, CHARMM approximates each residue with a fixed charge. Thus, the program needs to be told which charge state the user is specifying. In the case of histidine, using the residue name HSD instead of HIS indicates that the uncharged state of histidine is to be used, and in this case all the histidine residues will be simulated as being uncharged. Perhaps for historical reasons, CHARMM does not work with the IUPAC base designation of ILE atom CD1, and these entries need to be changed to CD before reading in the coordinates. Thus, the line

```
ATOM  324 CD1 ILE A 46   10.698 35.637 40.610 1.00 12.62   C
```

becomes

```
ATOM  324 CD  ILE A 46   10.698 35.637 40.610 1.00 12.62   C
```

The C-terminal oxygen atoms need to be renamed OT1 and OT2 rather than O and OXT, as they may be in PDB files. In our case, the oxygen atom of PHE 120 becomes one of the C-terminal oxygen atoms, and will automatically be replaced with two C-terminal oxygen atoms that will be named OT1 and OT2.

In the PDB file, the segment of protein ends with the line

```
TER   1346   ILE A 167
```

and at the end of the file is a line containing the word *end*. Thus, the end of the file should be changed to the following:

```
TER   1346   PHE A 120
END
```

The atom numbers in column 2 need not be adjusted as they are ignored when CHARMM reads the input files. Two additional changes must be made to most of the lines. In column 23 of the input file is the letter A for the A subunit. This must be replaced with a blank. In this PDB file, column 73 contains symbols for the atom that is described on that line. Other PDB files contain data bank identification labels such as 2ARC in columns 73–76. CHARMM requires one label in this position, a label that must be provided later in scripts. In this example the label PROT is being used.

```
ATOM  324 CD ILE  46   10.698 35.637 40.610 1.00 12.62   PROT
```

Which of the two arabinose molecules is the one that is bound to the A subunit can be determined by close examination of the coordinate files or by the use of a molecular display program. In this file, the term HETATM must be replaced with ATOM, and the segment identifier in columns 73–76 must be provided. This example uses LIGA. Thus, the final lines of the PDB file for arabinose become the following:

```
ATOM  2705 05  ARA 100   8.802 37.405 36.359 1.00 13.87   LIGA
END
```

Changes also are made in the file containing the bound water molecules. HETATM must be changed to ATOM, columns 14–16 are changed to OH2, columns 18–21 are changed from HOH to TIP3, and SOLV must be written in columns 73–76. TIP3 specifies the highly refined model for the water molecule that is used in CHARMM simulations.[7] The topology and parameter tables contain special entries for it.

Handling Structure of Arabinose

Programs that calculate interaction energies need to know the identity and properties of every atom in a structures. As mentioned earlier, in CHARMM information about the identities of atoms contained in a residue, their partial charges, the bonding pattern of the atoms, and their spatial positions with respect to one another are all obtained from the residue topology file. CHARMM will be unable to read the PDB file of arabinose described above because its residue topology file contains no mention of arabinose. Therefore it is necessary to add an arabinose entry to the residue topology file. The *de novo* construction of a complete topology file entry for a small molecule can require substantial effort and is not described here. In the present case, however, the construction of a workable topology file entry is not difficult. Examination of the PDB file for arabinose shows that it contains coordinates for all the heavy atoms. Hence, CHARMM will not have to construct the locations of any missing atoms

through the use of the internal coordinates that may be provided in a topology file entry. Therefore, the internal coordinates can be omitted. Examination of the arabinose molecule with a graphics display program shows that the sugar in the protein is in a α-pyranose ring. Glucose also forms a pyranose ring and, therefore, much of a topology file entry for glucose can be used in writing a new entry for arabinose (Fig. 1). In practice, the entries for many carbohydrates will differ from one another only in internal coordinates of one or more oxygen and hydrogen atoms. Their bonding patterns, atom types, partial charges, and hydrogen bond donors and acceptors will all be the same.

It proves easiest to model the new arabinose topology file entry after that given for glucose given in a separate topology file, top_all22_sugar.inp, and then to place this in the top_all27_prot_na.rtf file because this file, and its partner parameter file, par_all27_prot_na_prm, can be used for calculations and simulations involving proteins, nucleic acids, and combinations of the two. Examination of the sugar file and Fig. 1 shows that the structure given for glucose becomes that of arabinose by removing all references to H61, H62, O6, and HO6 changing C6 to H51 and H5 to H52, and adjusting the stereochemistry at C4. The CHARMM type and partial charge of each atom in arabinose must also be specified. The types and partial charges are developed together, along with the parameter table. As the arabinose entry is to be added to the top_all27_prot_na.rtf file, this file should be used for examples of atom types and partial charges. The file contains entries for ribose in ribonucleotides, and thus the types and partial charges for the atoms in arabinose can be determined by example.

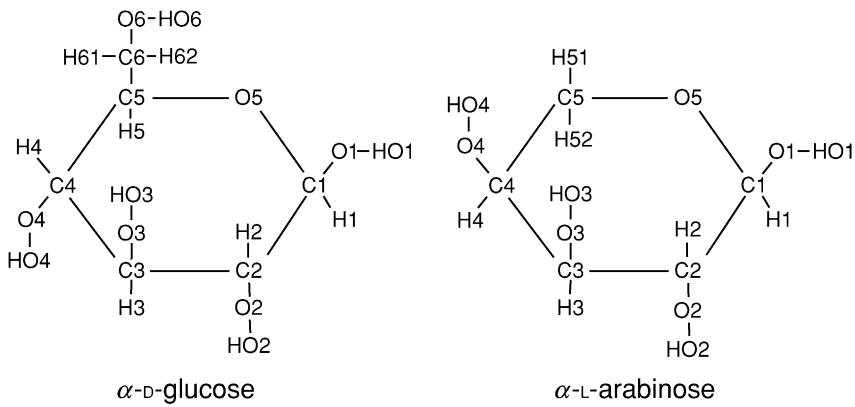

FIG. 1. Structures and Protein Data Bank identification labels of the atoms of glucose and arabinose.

Energy Determination Script for CHARMM

The script for determination of the residue–arabinose interaction energies is described below. Scripts and output files begin with header lines that can provide useful archival information about the functioning of the script. These title lines begin with an asterisk and the line following the last title line contains only an asterisk. Thus, the title lines for the energy determination script would be

```
*Example script for energy calculation
*
```

The first step in the determination of the interaction energies is the reading of the modified topology and parameter files. A file that is to be opened is named, and a label in the form of a unit number is provided so that the program can later refer to this file. The specification of card in the open statement indicates that the data are in ASCII format. This is probably a holdover from the days of punched cards. In the script, the topology and parameter files are opened and read with the following commands:

```
open read card unit 20 name top_all27_prot_na.rtf
read rtf card unit 20
close unit 20
open read card unit 22 name par_all27_prot_na.prm
read parameter card unit 22
close unit 22
```

Next, the protein to be analyzed is read in. This is done in two steps. First, the sequence of the protein is read. From the amino acid sequences, and by reference to the residue topology file, CHARMM generates arrays with a slot for each atom in the protein. These arrays include the *x, y,* and *z* coordinates of each atom; a duplicate coordinate array; an array for the internal coordinates; also arrays for the *x, y,* and *z* components of the forces on each atom, frictional forces that may be assigned to each atom, whether or not an atom is to be constrained; as well as additional arrays that can be used in computation. The generate command instructs CHARMM to set up these arrays, and the setup option instructs the program to place in the internal coordinate array of the protein the appropriate internal coordinate entries from the various residues in the residue topology file. The generate command must include a segment identifier for the molecule being read. In this case, the identifier is PROT. The relevant commands are

```
open read unit 23 card name protein.pdb
read sequence pdb unit 23
generate prot setup
rewind unit 23
```

Because the N-terminal and C-terminal amino acids of a protein possess structures different from those of the internal amino acids, special treatment is automatically given to the first and last residues of a structure. Water, of course, does not require such treatment as is reflected in the first none and last none options in the generate statement. Also, the TIP3 water molecules, which are specially devised approximations to water, are not allowed to bend or twist, hence the noangle and nodihedral options. Sequences of the arabinose and crystallographic water molecules are read in and the necessary arrays are generated as follows:

```
open read unit 24 card name arabinose.pdb
read sequence pdb unit 24
generate liga setup
rewind unit 24
open read unit 25 card name xwater.pdb
read sequence pdb unit 25
generate solv setup first none last none noangle nodihedral
rewind unit 25
```

Having generated the necessary slots, the coordinates can be read in:

```
read coordinate pdb unit 23
close unit 23
read coordinate pdb append unit 24
close unit 24
read coordinate pdb append unit 25
close unit 25
```

The side chains of some hydrophilic residues on the surface of a protein are likely to extend into the solvent and not to have fixed positions. As a result, such side chains often are not seen in structures determined by X-ray diffraction, and their coordinates will therefore be missing from the PDB files. Provision is provided in CHARMM for the generation of such missing coordinates. This same capability is also used when the program is used to construct structures *de novo*. The first of the operations necessary in the construction of the coordinates of missing atoms is to issue the ic fill preserve command. This instructs CHARMM to generate values for internal coordinates of all atoms for which Cartesian coordinates were present in the PDB file and that are now in the coordinate array. All such new internal coordinate values are to replace internal coordinate values already in the table while preserving the remainder of the internal coordinate values. At this point, then, the internal coordinate table consists mostly of entries that have been calculated from the actual coordinates of the protein and a few entries for missing atoms whose entries derive from the residue

topology file. In some cases the internal coordinate entries in the residue topology file may not have been fully specified and, therefore, at this point the internal coordinate table may still be incomplete. Therefore the ic parameter command instructs CHARMM to use the parameter table as a source of last resort and to use it for rather generic values for the internal coordinate table. Finally, when the internal coordinate table has been filled, the ic build command instructs CHARMM to reconstruct the Cartesian coordinates of all the atoms. If coordinates for an atom exist in the coordinate arrays, these values are retained, but the coordinates of all other atoms are calculated from the internal coordinate values. The final step in constructing the entire protein is the addition of any hydrogen atoms that have not been placed already. The commands used in the reconstruction process are as follows:

```
ic fill preserve
ic parameters
ic build
hbuild
```

In our example, only a part of one of the two subunits of AraC is being simulated, but crystallographic water molecules were associated with both subunits. Therefore those water molecules that were associated with the parts of the protein that are not being simulated must be deleted. This is accomplished with the delete command, by which entire water molecules are removed if their oxygen atoms are not within 8 Å of the segment we are studying. The hyphen is a line continuation symbol and is used for long lines.

```
delete atom select ( .byres. ( (.not. (segid prot .around. 8 ) )
  . and. – ( segid solv .and. type OH2) ) ) end
```

After input of the protein, arabinose, and crystallographic water molecules and removal of water molecules not associated with the protein fragment, the complex is to be centered at the origin. This is accomplished with the coordinate orient command:

```
coordinate orient select segid prot .or. segid liga .or.
  segid solv end
```

To immerse the protein in a shell of water, a box of water molecules centered at the origin is read and followed by deletion of water molecules that either overlap the complex or that lie more than 8 Å from the protein segment. The water box was prepared separately and was stored in the CHARMM coordinate format called crd. Either the Protein Data Bank format, PDB, or the CHARMM format, CRD, can be used for most purposes, although the specification of specific atoms for use in calculations

is sometimes easier if the crd format of the coordinates is available. The box is read in, the deletion operations are performed, and the designations of the two types of water, solv for the water molecules seen in the X-ray studies, and boxx for the water molecules of the box, are also coalesced for convenience as follows:

```
open read unit 26 form name box.crd
read sequence unit 26
generate boxx setup first none last none noangle nodihedral
rewind unit 26
read coordinate card append unit 26
close unit 26
delete atom select ( .byres. ( ( ( segid prot .around. 2.5 ) )
  .and. – (segid boxx .and. type OH2 ) ) ) end
delete atom select ( .byres. ( ( .not. (segid prot .around.
  8 ) ) .and. – (segid boxx .and. type OH2 ) ) ) end
join solv boxx renumber
```

At this point the psf file can be written out for later use. The commands for doing so are as follows:

```
open write unit 41 card name protwat.psf
write psf unit 41 card
*psf of protein, arabinose, crystallographic waters, and
  water box
*
```

The above-described manipulations likely have placed some atoms very near one another or abnormally stretched some bonds. Dynamics simulations cannot be done with such a system because on starting a simulation some atoms will rapidly acquire impossibly high velocities. Therefore, an energy minimization is required to reduce the net force on each atom to nearly zero. In a minimization, the coordinates of all atoms in the system will be allowed to be move. Here 1000 steps of minimization are used and the results are printed out every 100 steps. For a system containing 10,000 atoms, this operation takes about 1 h on a 1-GHz machine.

```
minimize abnr nstep 1000 nprint 100
```

The molecular dynamics can now be run. Because they are so light, the hydrogen atoms vibrate at high frequencies. As a result, small time steps are required. Longer time steps can be used if the shake command is issued first. It alters the way hydrogen atoms are considered. A restart file is also opened. This contains the atom positions and velocities and allows a dynamics run to be restarted and continued from the time point at which the file was written. A file for recording the dynamics trajectory is also

opened. In this phase of the simulation, the system will be heated from near by 0 K to about 300 K. The heating is specified to take place in 6000 time steps of 0.001×10^{-12} s, coordinates are to be saved every 100 steps, the status of the system is to be recorded every 100 steps, and atom velocities are to be increased each 100 steps so as to increase the temperature of the system by 5 K. A 1-GHz machine can process about 4000 time steps per hour of a 10,000-atom system.

```
shake bonh tolr 1e-09 mxit 500 param
open write unit 31 card name heat.rst
open write unit 32 file name heat.dcd
dynamics leap verlet start nstep 6000 timestep 0.001 nprint
  100 -
  nsavc 100 nsavv 0 inbfrq -1 iprfrq 100 ihtfrq 100 -
  iunread -1 iunwri 31 iuncrd 32 iunvel -1 kunit -1 -
  wmin 1.5 first t 0.00000 finalt 300 teminc 5 -
  iasors 1 iasvel 1 iscvel 0 ichecw 0
```

After heating, the dynamics simulation is restarted from the heating restart file. The system is allowed to equilibrate for 10,000 steps. Periodically during this time, velocities will be adjusted so as to bring the system temperature back to 300 K if it has drifted. The necessary commands for doing this are as follows:

```
  open read unit 30 card name heat.rst
  open write unit 31 card name equil.rst
  open write unit 32 file name equil.dcd
  dynamics leap verlet restart -
    nstep 6000 time .001 nprint 100 nsavc 100 -
    nsavv 0 inbfrq -1 iprfrl 100 ihtfrq 0 -
    ieqfrq 100 ntrfrq 100 -
    iunrea 30 iunwri 31 iuncrd 32 iunvel -1 kunit -1 -
    wmin 15 finalt 300.0 -
    iasors 1 iasvel 1 iscvel 0 ichecw 0
```

Once the system has been equilibrated, the simulation is restarted from the equilibration restart file and allowed to run free. The following commands specify this as well as the storing of coordinates after the run.

```
open read unit 30 card name equil.rst
open write unit 31 card name dyn.rst
open write unit 32 file name dyn.dcd
dynamics leap verlet restart -
```

```
  nstep 100000 time 0.001 nprint 1000 nsavc 1000 -
  nsavv 0 inbfrq -1 iprfrq 1000 ihtfrq 0 -
  ieqfrq 0 ntrfrq 0 -
  iunread 30 iunwri 31 iuncrd 32 iunvel 01 kunit -1 -
  wmin 1.5 final 300 iasors 1 iasvel 1 iscve10 ichecw 0
open write unit 41 card name s.crd
write coordinate unit 41 card
*Coordinates after simulation dynamics
*
```

Ordinarily the dcd file from the simulation run is analyzed later. In this example, however, the same script both generates the trajectory file and then analyzes it. The analysis portion begins by opening a file for the output of the energy data. Then the trajectory file is opened for reading. The program must be told at which time step to begin reading data and the number of time steps between data points to be read. This is specified as follows:

```
open write unit 51 card name energy.tst
open read unit 52 file name dyn.dcd
trajectory iread 52 begin 13000 skip 1000
```

Two loops will be used to calculate the energies. The outer loop controls the reading of the trajectory frames, uses the loop variable *i,* and the label *outer*. The frames will be read one by one to the end of the dcd file. The inner loop calculates the interaction energy between successive residues of the protein and the arabinose and uses the loop variable *j* and the label *inner*. Successive energy calculations will write data to the energy.tst file. Each time data are to be written, the data will be preceded by a title line that includes the current values of *j* and of *i*. This structure allows simple extraction of the energy for any particular residue. The interact command specifies the calculation of an interaction energy, and the select portion specifies that the calculation be of the interaction between residue *j* and the ligand.

```
set i 1
label outer
trajectory read
set j 1
label inner
write title unit 51
*Interaction energy of residue @j with arabinose frame @i
```

```
*
interact select segid prot .and. resid @j end select -
  segid liga end unit 51
increment j by 1
if j lt 113 goto inner
increment i by 1
if i lt 101 goto outer
stop
```

A small portion of the output file, energy.tst is as follows:

```
INTERACTION ENERGY OF RESIDUE 1 WITH ARABINOSE FRAME 1
INTE ENR: Eval#    ENERgy   Delta-E     GRMS
INTE EXTERN:      VDWaals      ELEC   HBONds      ASP     USER
-------------  ---------  --------  -------  -------  -------
INTE>       1   -0.68143   0.00000  0.02584
INTE
 EXTERN>        -0.07480  -0.60663  0.00000  0.00000  0.00000
-------------  ---------  --------  -------  -------  -------
INTERACTION ENERGY OF RESIDUE 2 WITH ARABINOSE FRAME 1
INTE ENR: Eval#  ENERgy  Delta-E  GRMS
INTE EXTERN:      VDWaals      ELEC   HBONds      ASP     USER
-------------  ---------  --------  -------  -------  -------
INTE>       2   -3.98527   3.30384  0.10683
INTE
 EXTERN>        -0.32970  -3.65557  0.00000  0.00000  0.00000
-------------  ---------  --------  -------  -------  -------
```

The following will extract the interaction energies between residue 1 and arabinose and list them in a file it creates called energy1.out. The grep command finds lines in energy.tst containing ''RESIDUE 1 WITH'' and pipes the line that matches and the next four lines to the awk command that finds the lines containing INTE> and outputs the third column to the file energy1.out.

```
grep -A 4 ''RESIDUE 1 WITH'' energy.tst|awk '/INTE>/{print
  $3}'>energy1.out
```

The energy1.out file may then easily be imported into a spreadsheet program for analysis or plotting.

Acknowledgments

The author thanks Lenny Brand, Ken Johnson, George Rose, and Michael Rodgers for comments, guidance, and suggestions on the preparation of this chapter, and acknowledges NIH Grant GM18277 for support.