MMM- Multiscale modeling of macromolecular systems

G. Jeschke
ETH Zürich, Lab. Phys. Chem.

Gunnar.Jeschke@phys.chem.ethz.ch

12. März 2009

Inhaltsverzeichnis

1 Introduction

1.1 Why yet another program for protein modeling?

There are many good and even free software packages for visualization of biomacromolecules and biomacromolecule complexes. Some of them are open source and could be extended to use information from newly developed techniques, such as distance measurements by pulsed electron paramagnetic resonance (EPR) spectroscopy [1, 2]. However, our experience with rotamer libraries for the description of spin label conformations has shown that rather complex extensions may be required that would be difficult and inefficient to introduce into software packages designed for work with atomistically resolved x-ray and NMR structures. Our previous workaround were home-written programs for modeling that can read and write PDB files [1]. This approach has now run into the following problems:

- The PDB format is designed for static atomistic coordinates, as obtained by x-ray or NMR techniques. In many cases experimental information is available that is insufficient to build an atomistic model, but is still useful.
- The PDB format- and programs based on it- do not allow for introducing low-resolution objects, such as a lipid bilayer, without specifying atomic coordinates.
- The PDB format- and programs based on it- consider structures as generally certain, if possibly incomplete. In fact, different parts of a structural model may be known with different degrees of certainty and with different resolution.

These problems have led to an unsatisfying situation in modeling of biomacromolecular systems. Atomistic structures are generally considered as reliable and as a finished, static model of the system. In reality they are snapshots that describe only part of the behavior of the system. Non-atomistic models are generated by *ad hoc* methods geared to the type and quantity of information accessible at the time of modeling. They often exist only as pictures, not mathematically formalized models, which makes systematic improvement by new experimental information impossible. As an alternative they

[1] For definition of the PDB format, see http://www.wwpdb.org/docs.html

can be specified in atomistic coordinates, but with the caveat that much of the information content is based on mere assumptions and there is no systematic way to specify what part of the information that is. For this reason non-atomistic models are seen as unreliable- and they often are!
In fact, a protein or, more generally, a complex of biomacromolecules and cofactors is a dynamic system. Information on this system is not obtained by a single group who solves "the structureänd it is *never complete*. In other words, the structure file cannot be closed, as a data base with unchangeable files suggests. A protein model is an evolving object that becomes more and more detailed- and more and more dynamic- over time. Information of several studies from different groups should be incorporated and it should be possible to extend the model by information from newly developed experimental approaches, even if these were unknown when the first version of the model was built. In other words, a protein model is not like a Shakespeare play, once written by a single author and never changed after that, but rather like a software package that has many contributors and goes through many versions.
This view of protein models is at odds with the basic data structure assumed in the PDB format and by established modeling programs. The problem cannot be solved by extension of these programs, as the architecture of a program depends very much on the concept of the processed data. For that reason a complete redesign of the modeling approach is required. This approach should encompass atomistic structures, as these are by far the most reliable models that we currently have. However, it should allow for models that have different resolution and different degrees of certainty in different domains and that can be specified as dynamic with respect to the conformation of certain domains or their relative arrangement.
The basic idea of multiscale coarse-grained modeling, which can form the mathematical basis for such a program, has emerged in the molecular dynamics (MD) community as a way to overcome severe limitations in the time scale of simulations of protein dynamics and folding [3, 4]. However, our approach is concerned with modeling based on *experimental* data, supported by simulations. Hence, our primary concern is not a realistic description of the energy of the system, as it is sought by the MD community. The aim of our approach is to build models that are consistent with a set of experimental restraints from different techniques and with established theoretical knowledge on protein structure. Furthermore, the uncertainty of such models

should be specified as far as possible.
As not all the mathematics and not all the experimental techniques or even types of restraints are yet known (or will ever be known) the whole approach in general and the program in particular have to be extensible. This starts with the data format for encoding the model. Object-oriented programming [5] appears to be best suited for such a program.

1.2 What the program should do and how the user interface should look like

For many basic tasks in computational structural biology powerful approaches do already exist. Reprogramming everything would be inefficient and would probably not result in a program that matches the performance of the best available software for that task. The program should thus be open not only with respect to future extensions, but also with respect to communication with third-party software. Examples are

- Optimization of sidegroup packing, for instance after mutations, by SCWRL3 [6].
- Molecular dynamics, based on the AMBER, CHARMM, or GROMOS force fields, for instance using Tinker[2] with AMBER96 (Ponder:03).
- Use of NMR or x-ray data and work with analogous restraints by interfacing to CNS [8].

The program should implement new algorithms for working with experimental techniques that are not yet used for structure determination in a systematic way, foremost EPR. It should also implement modeling for structural transitions (function-related structural dynamics). Furthermore, the program needs to be able to define restraints with respect to coarse-grained objects, for instance the immersion depth of a residue in a lipid bilayer that is defined only by its center plane and thickness.
Although very pleasing visualization programs for proteins exist, our program needs to implement visualization, mainly because systematic coarse-graining is not possible in the existing programs. Here we do not strive for the aesthetically most pleasing visualization, as in many cases it will be possible to visualize a finished version of the model with other software. Rather

[2] see: http://dasher.wustl.edu/tinker/

the visualization approach should allow to find the wanted information in the model with the least effort. It should thus be possible to guide the display by certain questions about the structure or structural transitions, rather than searching the information by trial-and-error rotations with the mouse.
This calls for a user interface that is at the same time precise and easy to work with. As the models are complex- and will be even more complex in the future- the tasks will be complex. The user will soon be annoyed with the need to perform the same 100 mouse clicks again and again. Hence, the program needs a scripting language and must be based on a command interpreter. Experience shows that most users (initially) do not like command control of programs. Hence, the whole functionality of the program should be accessible by mouse clicks. This apparent contradiction can be solved by assigning commands to each menu function, tool button and push button and by echoing the commands initiated with the mouse in the command line. Thus users can learn the command language conveniently and at their own pace.
Complex tasks call for an 'undo' function. In fact, it should be possible to undo several actions and redo them if necessary. This is easy to implement in a command-based approach by defining an inverse command for each command and storing it in an undo list. At the same time the command history is stored. Undone commands are stored in a redo list. Some tasks, in particular if they involve external programs, may not be undoable or only at considerable expense in computation time, storage, and programming effort. In such cases the user should be warned that the command cannot be taken back after execution. The user should be given the possibility to veto command execution.
Although windows of most current graphical user interfaces (GUIs) can be resized, this actually does not make sense. In a good GUI everything is at the same place every time, as most people have some degree of visual memory and can navigate more easily when it looks the same. On the other hand, the window that displays the model should be resizeable. This is necessary as one may want to work on computers with different screen resolution and one wants to see as much detail as possible with given screen resolution. Hence, the model display window should be detachable from the GUI, so that it can be resized indepedently. It should also be reattachable, as some actions require only a small picture of the structure and are easier to perform with this picture in the GUI.

Written information is nowadays underestimated. Some results are best displayed as words and numbers. The GUI should contain a large message board to display such information routinely in the same place, where users really look for it, rather than in a hidden tab that the uesers first have to find. Special tasks may require special forms of display that are not suited for the model display window or the message board. In this case additional windows should be opened.

2 Hierachical description of macromolecular models

cite CNS

3 Display algorithms

3.1 Ribbon models

Coarse-grained graphical representations were utilized in an early comprehensive review on structural features of proteins [9] and shortly thereafter computer-generated schematic diagrams were proposed [10]. Here we start from the formal description of the ribbon model [11], which uses six parameters per residue k, the coordinates of the $C^{\alpha}{}_k$ atom and the carbonyl oxygen atom O_k. The local frame $\vec{a}_k, \vec{c}_k, \vec{d}_k$ at each residue is derived starting with a unit vector $\vec{a}_k$ pointing along the backbone from C^{α}_k to C^{α}_{k+1}. The auxiliary unit vector $\vec{b}_k$ points from C^{α}_k to the carbonyl oxygen O_k. The unit vector $\vec{c}_k$ perpendicular to the approximate peptide plane is generated by the cross product

$$\vec{c}_k = \vec{a}_k \times \vec{b}_k \ , \tag{1}$$

and the Cartesian frame is completed by unit vector $\vec{d}_k$ within the peptide plane

$$\vec{d}_k = \vec{c}_k \times \vec{a}_k \ . \tag{2}$$

The flat face of the ribbon is the approximate peptide plane, i.e., it contains $\vec{d}_k$. For the last residue C^{α}_{k+1} is not defined, so that we keep $\vec{d}_k$ from the previous residue. For the typical right-handed α-helix vector $\vec{d}_k$ points to the right of the backbone. For β-sheets the complication arises that $\vec{d}_k$ flips from the right to the left backbone side or *vice versa* between consecutive

residues. Hence vector $\vec{d}_k$ has to be inverted each other residue in sheets [11].

The original implementation of ribbon models constructs guide points between residues and interpolates between guide points by B-splines, which were convenient as they could be handled by graphics devices at that time [11]. Here we depart from these conventions. First, we assign the guide points to the C_k^α, so that the parameters of the residue in the coarse-grained representation are the coordinates of C_k^α and unit vector $\vec{f}_k$. This assigns all parameters to the local frame of residue k and reduces the number of free parameters per residue to five. Second, we specify the interpolating splines as cardinal splines with zero tension (Catmull-Rom splines). Past implementations use different interpolation techniques, as can be seen by comparing the representations of loops in different programs, for instance PyMOL [12][3] and Swiss-Pdb Viewer [13]. Our implementation is similar to the one in PyMOL, however, it does not apply additional smoothing to the ribbons. The effect of smoothing is particularly strong for β-sheets, but can also be seen in α-helices.

Abb. 1: Screenshot of a ribbon model of the sodium/proton antiporter NhaA of *Escherichia coli* (PDB identifier 1ZCD).

Helical segments are defined with a rectangular cross-section of the ribbon with width 3 Åalong $\vec{d}_k$ and height 0.2 Åalong $\vec{c}_k$. For sheets we also use a rectangular cross-section of the ribbon with width 2 Åand height 0.2 Åalong $\vec{c}_k$. The last residue is marked by an arrow that starts at 1.75 times the original width and narrows to zero at the end of the sheet. For sheets the backbone curve is smoothed using a piecewise fit of quadratic functions with continuous first derivative with knots defined at the C^α atoms. This fit is performed separately for the x, y, and z coordinates. The fitted backbone

[3] http://www.pymol.org

curve passes through the first and last C^{α} atom but may deviate from other C^{α} atoms in the sheet, as is usual in ribbon models. Loops and turns are defined by a circular cross section with radius 0.2 Å, which is approximated by a decagon. By spline interpolation we define five straight segments per residue, which provides a satisfying compromise between aesthetic quality and speed of Matlab graphics. For loops, turns, and helices the center of the ribbon passes through all C^{α} atoms. As an example a representation of the crystal structure of the sodium/proton antiporter NhaA of *Escherichia coli* (PDB identifier 1ZCD) is shown in Fig. **??**.

Literatur

[1] G. Jeschke, Ye. Polyhach, *Phys. Chem. Chem. Phys.*, 2007, **9**, 1895–1910.

[2] O. Schiemann, T. F. Prisner, *Quater. Rev. Biophys.*, 2007, **40**, 1–53.

[3] P. Doruker, R. L. Jernigan, I. Bahar, *J. Comp. Chem.*, 2002, **23**, 119–127.

[4] P. J. Bond, M. S. P. Sansom, *J. Am. Chem. Soc.*, 2006, **128**, 2697–2704.

[5] B. Meyer, *Object-oriented Software Construction*, Prentice Hall, New York, 1997.

[6] A. A. Canutescu, A. A. Shelenkov, R. L. Jr. Dunbrack, 2003, *Protein Science*, **12**, 2001–2014.

[7] J.W. Ponder, D.A. Case, 2003, *Adv. Prot. Chem.*, **66**, 27–85.

[8] A. T. Brunger, P. D. Adams, G. M. Clore, P. Gros, R. W. Grosse-Kunstleve, J.-S. Jiang, J. Kuszewski, N. Nilges, N.S. Pannu, R.J. Read, L.M. Rice, T. Simonson, G.L. Warren, 1998, *Acta Cryst.D.* **54**, 905–921.

[9] J.S. Richardson, 1981, *Adv. Protein Chem.*, **34**, 167–339.

[10] A. M. Lesk, K. D. Hardman, 1982, *Science*, **216**, 539–540.

[11] M. Carson, C. E. Bugg, 1986, *J. Mol. Struct.*, **4**, 121-122.

[12] W. L. DeLano, 2002, *The PyMOL Molecular Graphics System* DeLano Scientific, San Carlos, CA, USA.

[13] N. Guex, M. C. Peitsch, 1997, *Electrophoresis*, **18**, 2714–2723.

[14] G. Jeschke, A. Koch, U. Jonas, A. Godt, *J. Magn. Reson.*, 2002, **155**, 72–82.

[15] G. Jeschke, G. Panek, A. Godt, A. Bender, H. Paulsen, *Appl. Magn. Reson.*, 2004, **26**, 223–244.

[16] Y. W. Chiang, P. P. Borbat, J. H. Freed, *J. Magn. Reson.*, 2005, **172**, 279–295.

[17] G. Jeschke, V. Chechik, P. Ionita, A. Godt, H. Zimmermann, J. Banham, C. R. Timmel, D. Hilger, H. Jung, *Appl. Magn. Reson.*, 2006, **30**, 473–498.