Desmond Tutorial

Desmond Version 2.0 / Document Version 0.1

June 2008

D. E. Shaw Research

Notice

The *Desmond Tutorial* and the information it contains is offered solely for educational purposes, as a service to users. It is subject to change without notice, as is the software it describes. D. E. Shaw Research assumes no responsibility or liability regarding the correctness or completeness of the information provided herein, nor for damages or loss suffered as a result of actions taken in accordance with said information.

No part of this guide may be reproduced, displayed, transmitted or otherwise copied in any form without written authorization from D. E. Shaw Research.

The software described in this guide is copyrighted and licensed by D. E. Shaw Research under separate agreement. This software may be used only according to the terms and conditions of such agreement.

Copyright

Trademarks

Ethernet is a trademark of Xerox Corporation.

InfiniBand is a registered trademark of systemI/O Inc.

Intel and Pentium are trademarks of Intel Corporation in the U.S. and other countries.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

All other trademarks are the property of their respective owners.

June 2008

Preface

Intended Audience

This tutorial is intended for computational chemists using Desmond in setting up and running molecular dynamics simulations. It assumes a broad familiarity with the concepts and techniques of molecular dynamics simulation.

Prerequisites

Desmond runs on Intel™-based Linux® systems with Pentium™ 4 or more recent processors; running CentOS 3 (RHEL3) or later. Linux clusters can be networked with either Ethernet™ or Infiniband®.

Viparr requires a recent version of Python; we recommend Version 2.5.1.

This tutorial assumes that someone has prepared the Desmond-Maestro environment for you, either by installing the packages available from Schrödinger LLC or the Academic release of Desmond available from D. E. Shaw Research. Where noted, the procedures described in the tutorial make use of software that is not included in the Academic version. In those cases the software is available from Schrödinger.

About this Guide

This manual contains the following sections:

Chapter 1 describes Desmond and outlines the steps to perform a simulation on a simple protein.

Chapter 2 describes the Desmond-Maestro environment, focusing on the tools and facilities available in Maestro for manipulating molecular systems.

Chapter 3 describes how to use Schrödinger's Prime utility for repairing proteins with missing residues or side chains.

Chapter 4 describes the System Builder, the tool for setting up molecular systems for simulation using Desmond

Chapter 5 describes force field parameter assignment program ***Viparr*** and making changes to the configuration file output by the system builder.

Chapter 6 describes how to run Desmond within the Maestro environment and from the command line.

Chapter 7 describes Free Energy Perturbation simulation using Maestro workflow.

Chapter 8 describes how to view the results of Desmond simulations in Maestro.

Chapter 9 describes VMD, an alternate workflow available separately from the University of Illinois, that can also be used to view trajectories and analyze the results of simulations.

Chapter 10 provides pointers to additional documentation on Desmond-Maestro system.

Format Conventions

Command lines appear in a monospaced font; in some cases, bolding draws your attention to a particular part of the command:

```
mdsim --include equil.cfg
```

Placeholders intended to be replaced by actual values are obliqued:

```
mdsim --tpp 4 --restore checkpoint_file
```

Configuration file examples also appear in a monospaced font:

```
mdsim = {     title = w
              last_time = t1
              checkpt = { ... }
              plugins = [ ... ]
              ... }
```

Configuration files are divided into sections, which can in some cases contain other sections; parameters occur at all levels. When discussed in the context of their particular section, configuration parameters appear by name in a monospace font, thus: `plugins`.

When discussed outside of the context of their sections, however, configuration parameters appear as a *keypath*, in which the name of each enclosing section appears in order

from outermost to innermost, separated by periods. For example, `force.nonbonded.far.sigma` refers to the `sigma` configuration parameter in the `far` subsection of the `nonbonded` subsection of the `force` section of the configuration file.

About the equations

The equations in this document are concerned with scalars, vectors, and matrices of various sorts. To help clarify the type of a quantity, equations in this manual use the following conventions:

- An upper- or lowercase letter without bolding or arrows, such as A or a, is a scalar.
- An arrow over a variable, such as $\vec{A}$ or $\vec{a}$, indicates three variables as a three-dimensional vector.
- A boldfaced lowercase letter, such as $\boldsymbol{a}$, is a vector of unspecified dimension, with a_i indicating the i^{th} element of the vector.
- A boldfaced uppercase letter is a matrix of unspecified dimensions, though usually 3×3, with A_{ij} being the element of row i and column j in matrix $\boldsymbol{A}$.

Certain quantities that are $3n$-dimensional vectors, such as $\boldsymbol{r}$, the positions of n particles, are indexed differently. The manual does not use r_i to refer to one of its $3n$ components, but instead $\vec{r}_i$ denotes the i^{th} three-dimensional vector in $\boldsymbol{r}$, which is the position of the i^{th} particle in this case.

Contents

List of Figures

1 Desmond Tutorial

Introducing Desmond

Desmond is an advanced classical molecular dynamics simulation system, which has an intuitive graphical user interface integrated in the Maestro molecular modeling environment. Using Desmond, you can perform in silico simulations of small molecules, proteins, nucleic acids, and membrane systems to study the interactions of complex molecular assemblies. With Desmond, you can generate high-quality simulation trajectories of long time scales as well as compute a variety of thermodynamic quantities. Desmond trajectories can be visualized and analyzed using numerous tools in Maestro and VMD.

This tutorial assumes a basic knowledge of molecular mechanics and molecular dynamics. For those users who have not yet used Maestro for molecular modeling, an overview of the Maestro environment is included.

Beyond basic molecular modeling, there are a number of steps that must be performed on a structure to make it viable for simulation with Desmond. Each of these steps mandates an understanding of certain concepts and differing approaches that can be taken.

To enable novices and advanced users alike to process the information in this tutorial, the organization of the tutorial includes:

- A step-by-step example that describes, at a high level, how to perform a simulation on a simple protein.
- In-depth concept sections that cover the detailed information users will need in order to perform a simulation similar to our example.

This tutorial also includes conceptual information and steps for performing free energy perturbation (FEP) calculations. FEP is useful for performing "alchemical" ligand mutations to determine binding free energy differences, which is a crucial task for such applications as computational drug design.

Steps to Perform Simulation on a Simple Protein

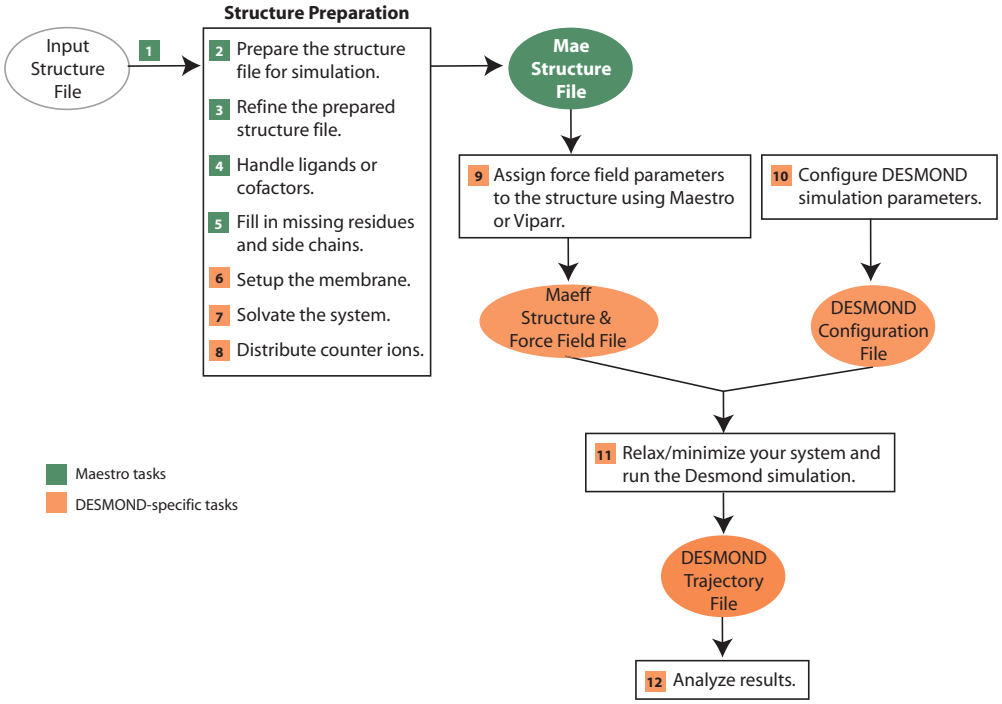
While the example described in this tutorial is a basic simulation on a single protein, the exercise is useful for learning the Maestro-Desmond environment as well as, in general, investigating the molecular properties of different proteins.

Preparing a molecular model for simulation involves these general steps:

1. Import a structure file into Maestro.
2. Prepare the structure file for simulation. During this step, you remove ions and molecules which are artifacts of crystallization, set correct bond orders, add hydrogens, detect disulfide bonds, and fill in missing side chains or whole residues as necessary.
3. Refine the prepared structure file. During this step, you analyze the choices made by Maestro's Protein Preparation Wizard and manually change protonation and tautomeric states of variable residues, and (when present) of ligands and co-factors.
4. If your system is a membrane protein, immerse the protein in the membrane.
5. Generate a solvated system for simulation.
6. Distribute positive or negative counter ions to neutralize the system, and introduce additional ions to set the desired ionic strength (when necessary).
7. Let Maestro assign OPLS-AA force field parameters to the entire molecular system; alternatively, assign force field parameters external to Maestro with Desmond's *Viparr* utility program, which provides access to a variety of widely used force fields.
8. Configure simulation parameters using the Desmond window in Maestro or edit the Desmond conguration file.
9. Run the simulation from Maestro or from the command line.
10. Analyze your results using the Trajectory Viewer and other analysis tools.

This process is illustrated in Figure 1.1.

Figure 1.1 Simulation process



The rest of this section contains step-by-step procedures for building and running a simulation of a single protein.

Tutorial Steps

1. Start Maestro from your Linux desktop by issuing the following command:

   ```
   $SCHRODINGER/maestro
   ```

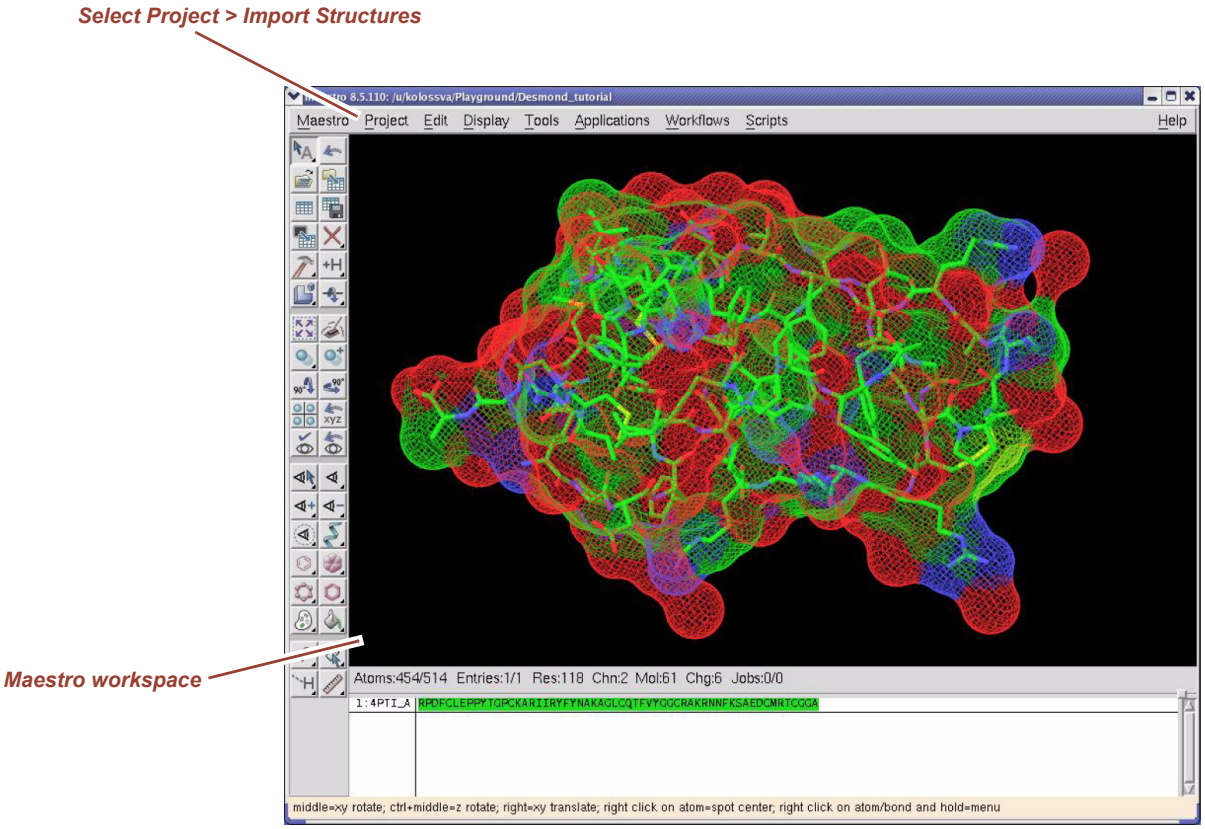
   If you access Maestro through the network (e.g., via VNC), start the program by the command:

   ```
   $SCHRODINGER/maestro -SGL
   ```

   This command will launch Maestro using Schrödinger's own software OpenGL graphics library.

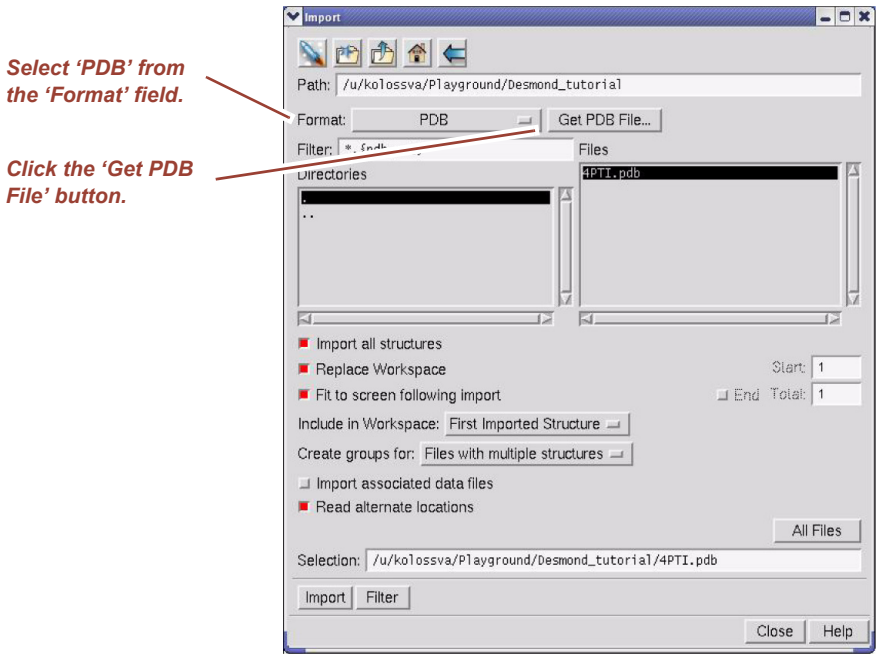
   The Maestro window appears as shown in Figure 1.2. If you are unfamiliar with Maestro, see "Learning the Maestro-Desmond Environment" on page 19.

Figure 1.2 Maestro main window



2. Import a protein structure file into your workspace by selecting **Project > Import Structures** from the menu bar. The Import panel appears as shown in Figure 1.3.

Figure 1.3 Import panel



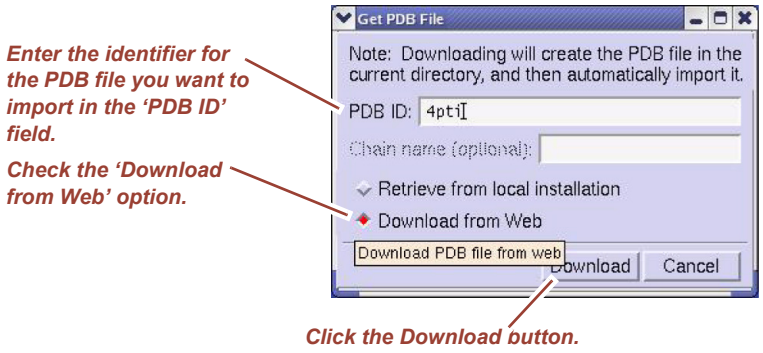
3. Select the type of structure file you will import from the Format field. For this tutorial, select **PDB** because we will import a PDB (Protein Databank File).

NOTE Maestro supports many common file formats for structural input. See the Maestro tutorial (Section 11) or click the Help button for a list of supported formats.

4. Click the Get PDB File button.

   The Get PDB File window appears as shown in Figure 1.4.

Figure 1.4 Get PDB File window



5. Enter the identifier for the PDB file you want to import into the PDB ID field.

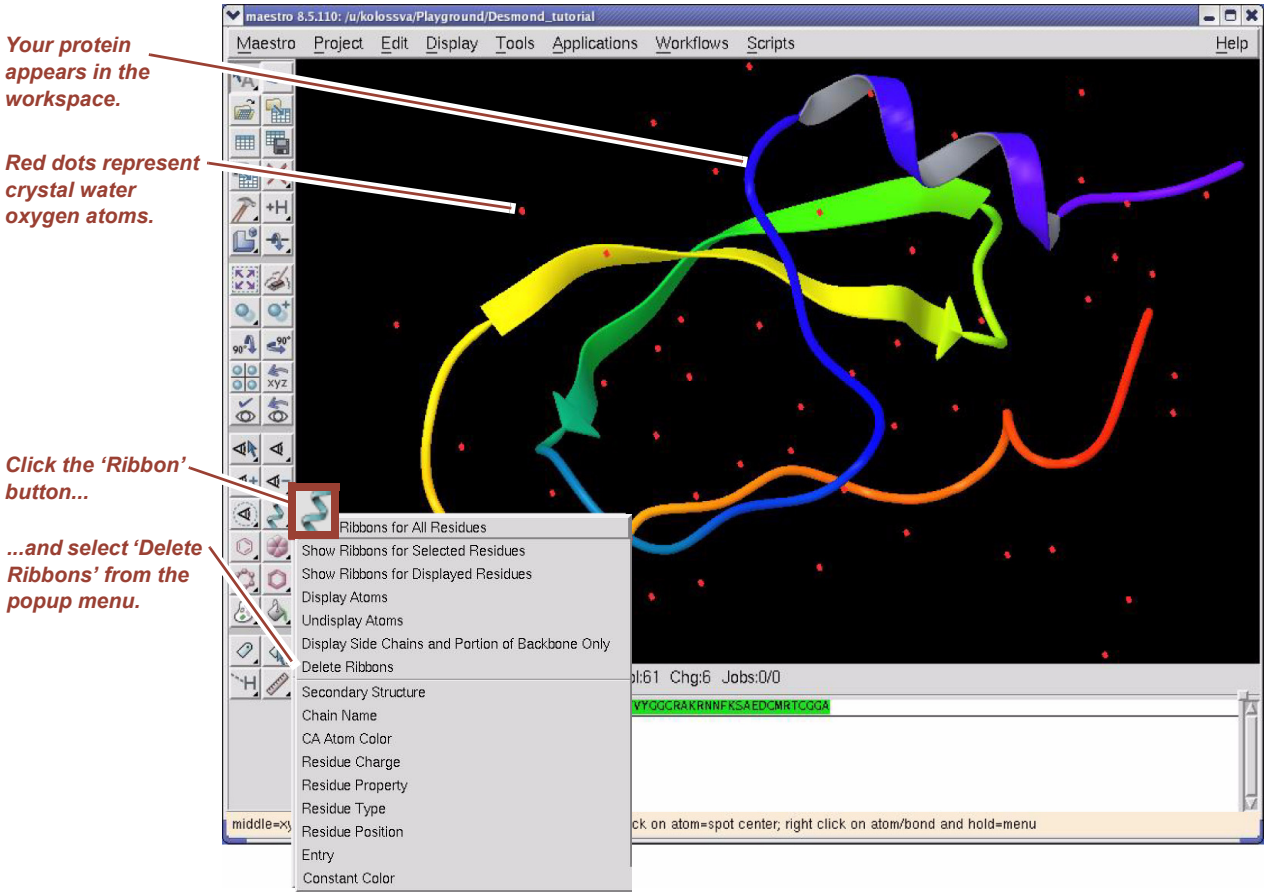
   For this example, we will import the small, proteinase (trypsin) inhibitor protein: 4pti. Therefore, enter **4pti** in the PDB ID field.

6. Check the Download from Web option to allow Maestro to download the PDB file from the PDB website.

7. Click the Download button.

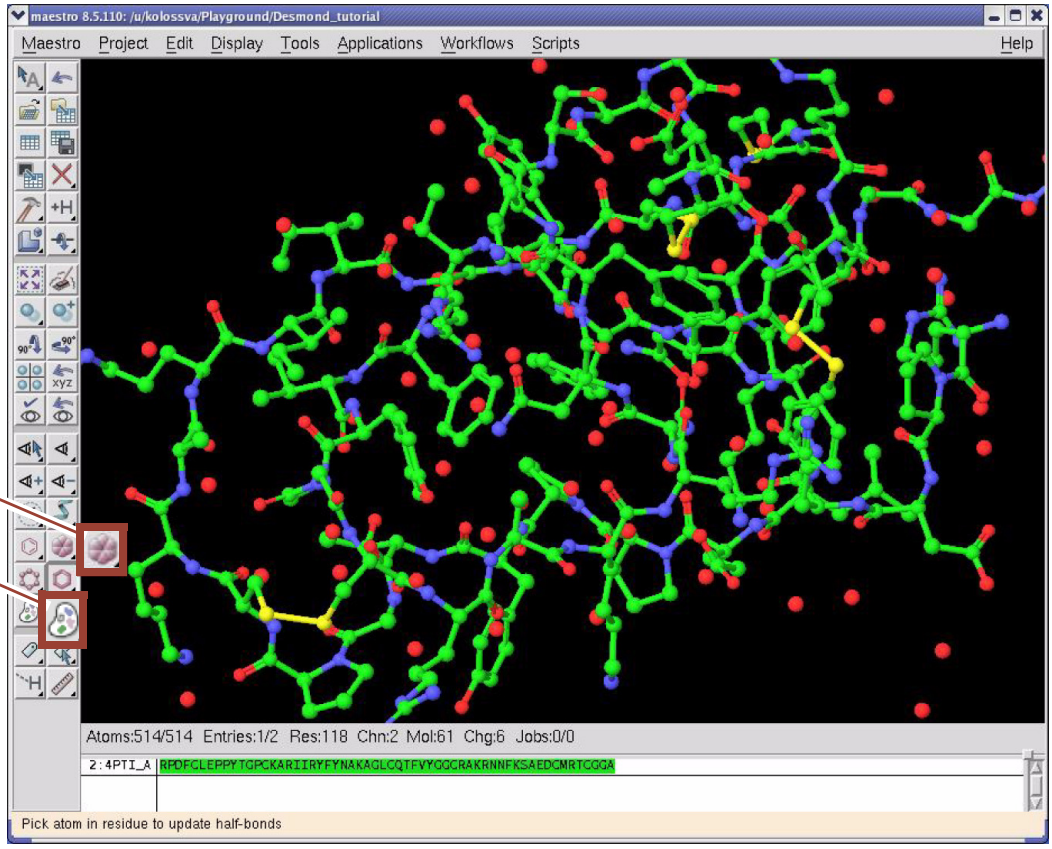
   The protein structure appears in the Maestro workspace as shown in Figure 1.5. The protein structure is displayed using a ribbon representation with colors continuously varying along the rainbow scale from the N terminus toward the C terminus. The red dots show the location of the oxygen atoms of the water molecules present in the X-ray structure.

Figure 1.5 Imported protein structure file



8. Switch from ribbon view to ball-and-stick view by clicking the Ribbons button and selecting Delete Ribbons from the pop-up menu that appears as shown in Figure 1.5. Next, as shown in Figure 1.6, double-click the Draw Atoms in Ball & Stick button and, finally, click the Color by Elements button to color the carbons in the structure green.

Figure 1.6 Changing from ribbon to ball-and-stick view



Once you have arrived at this point, you are ready to prepare the structure for simulation. Just having a molecular model in the workspace is not enough to perform molecular mechanics/dynamics calculations. You must prepare the protein model so that it corresponds as closely as possible to the biological system you are studying.

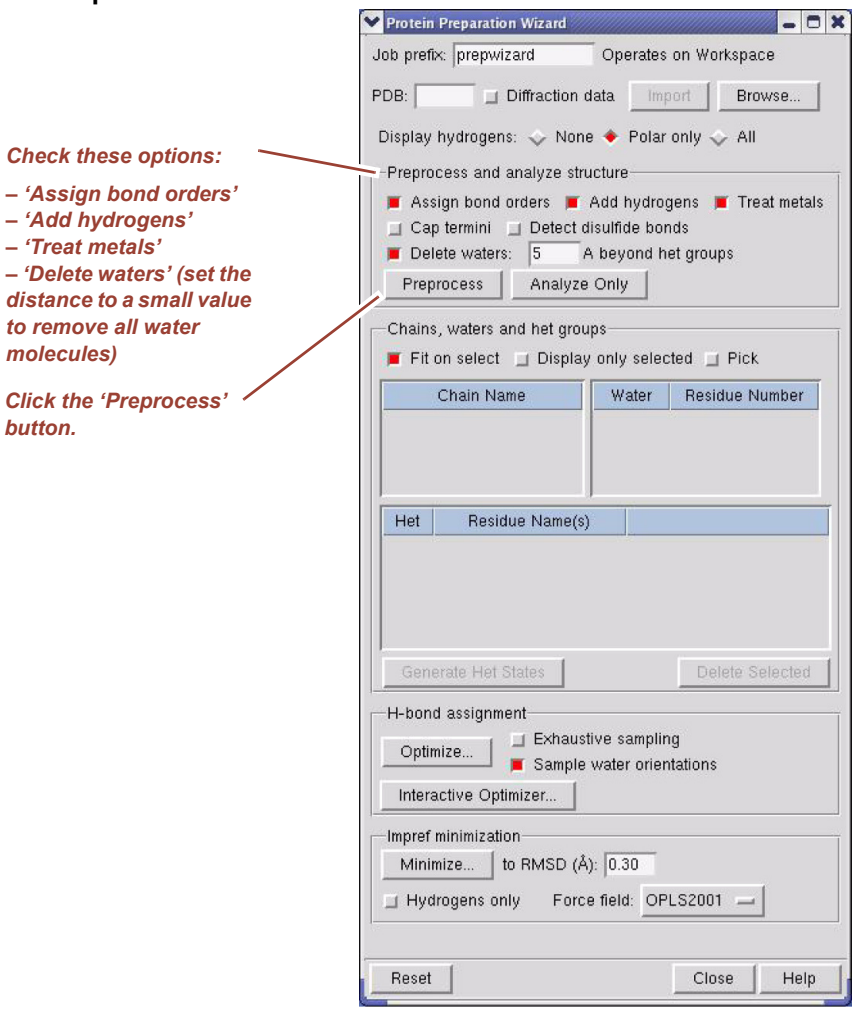
From this view, it is clear that there are no hydrogen atoms in the protein structure. Moreover, there might be ill-defined bond orders, protonation states, formal charges, tautomerization states, disulfide bonds, and so on. The red dots shown in the workspace represent crystal waters. All of these issues must be resolved before we can perform simulation calculations.

There are two methods of correcting these issues: the Build Panel opens Maestro's general purpose molecular structure editor and the Protein Preparation Wizard is a very useful, automated tool for protein structure preparation. For this tutorial, we will use the Protein Preparation Wizard.

9. Select **Workflows > Protein Preparation Wizard** from the menu bar.

   The Protein Preparation Wizard appears as shown in Figure 1.7.

Figure 1.7 Protein Preparation Wizard



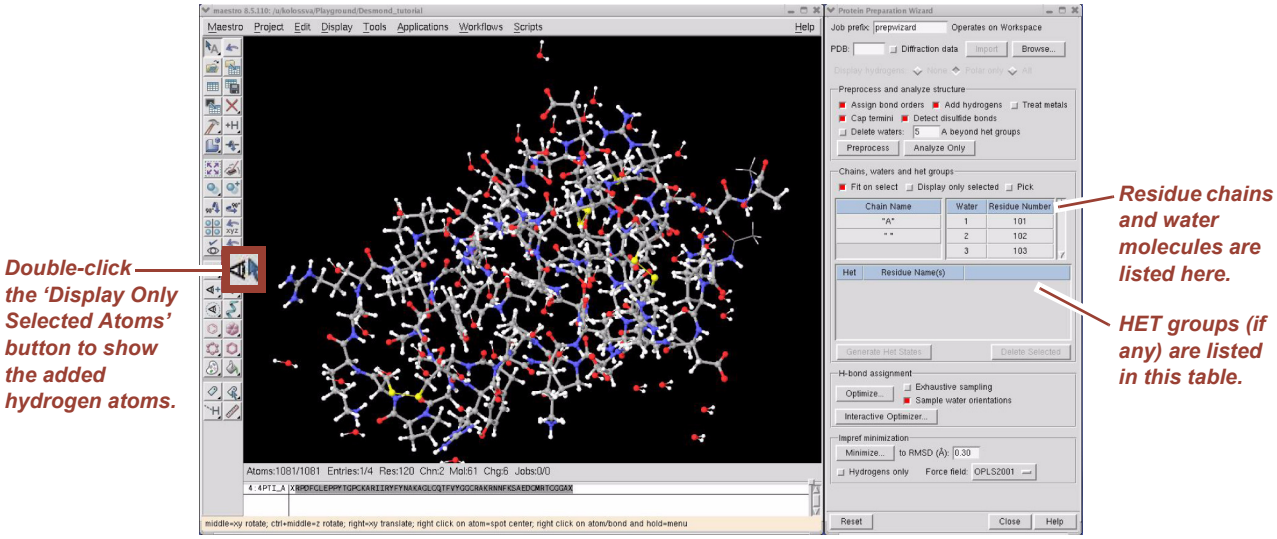
10. In the Preprocess and analyze structure area, check these options:
    - Assign bond orders
    - Add hydrogens
    - Cap termini. Set this option if you want the N or C termini capped. There are many capping groups available; the most common are ACE on the N terminus and NMA on the C terminus.
    - Detect disulfide bonds. The 4pti structure has three disulfide bonds, which are all correctly recognized by the Protein Preparation Wizard.
    - Delete waters. Set this option if you have a reason to remove crystal waters. Set the distance to a small value to remove all water molecules. "Het groups" refers to atoms that are labeled HETATM in a PDB file (anything that is not a protein residue or water).

NOTE In most cases crystal waters should not be removed from the interior of the protein molecule, or near the protein surface. These water molecules are crucial structural components of the protein and cannot be adequately reproduced by the algorithm used to solvate the system later on.

11. Click Preprocess to execute the first phase of the protein preparation tasks. The Analyze only button simply fills the tables in the wizard panel, but changes are not applied to the structure.

    Setup takes only a few seconds to complete. The Protein Preparation Wizard displays the chains, waters, and het groups as shown in Figure 1.8.

Figure 1.8 Protein Preparation Wizard - Preprocessing stage

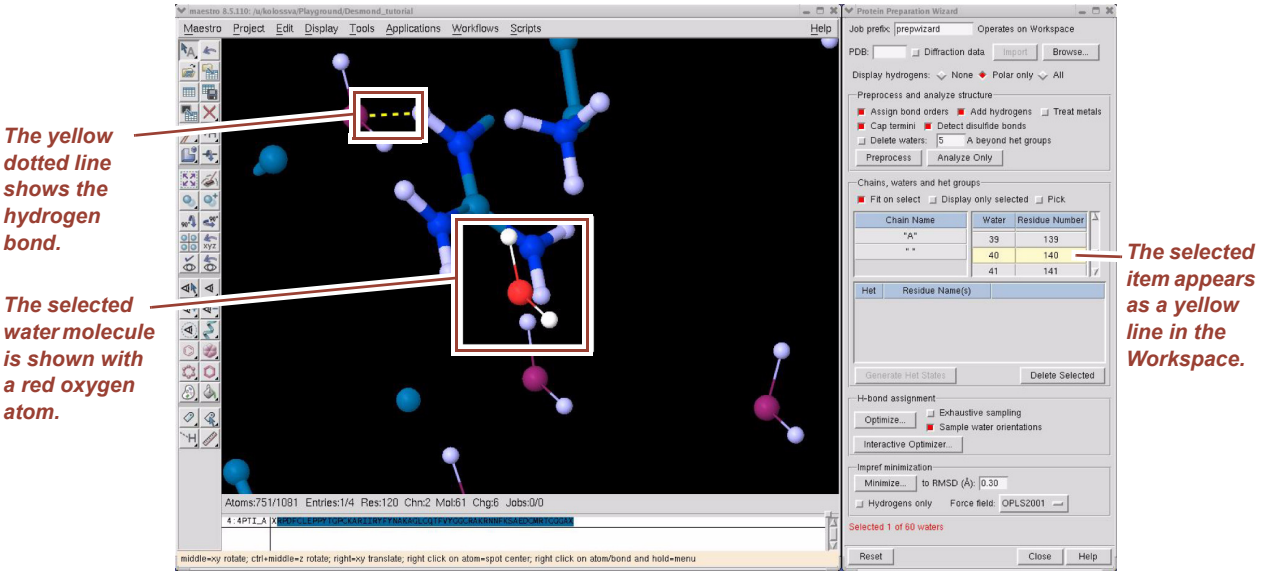


As shown, the top table lists residue chains and water molecules, and the bottom table lists HET groups such as metal ions, enzyme cofactors, ligands, and so on (in this example this table is empty).

The added hydrogen atoms are not shown by default. To display them, double-click the Display Only Selected Atoms button as shown in Figure 1.8. Also note that the added capping groups can be seen in the circled area, shown in wire frame representation.

You can select any item in the table by clicking its row as shown in Figure 1.9. Notice that when you select an item in the table, the Workspace view focuses on the selection and highlights it with a contrasting color scheme; hydrogen bonds are also displayed. Use CTRL-Click to return to the original view. On Figure 1.9 a single water molecule is selected, which is shown with the red oxygen atom in the middle of the Maestro Workspace.

Figure 1.9 Protein Preparation Wizard - Selection display



12. The Generate HET States button is used to automatically generate alternative protonation and tautomerization states for the HET groups. This is especially useful when ligands or cofactors are present in the system, but the current example does not have any such items.

    Moreover, you will need a license from Schrödinger, LLC to use their Epik tool to perform automatic HET state generation. If you do not have an Epik license, you will have to prepare your ligands and cofactors outside the Maestro environment before you import the system into Maestro and process it with the Protein Preparation Wizard.

13. At this point you should have a topologically correct molecular system in the Workspace, which can be subjected to molecular mechanics calculations. However, since hydrogen atoms were added by the Protein Preparation Wizard using simple geometric templates, the hydrogen bond network should be optimized.

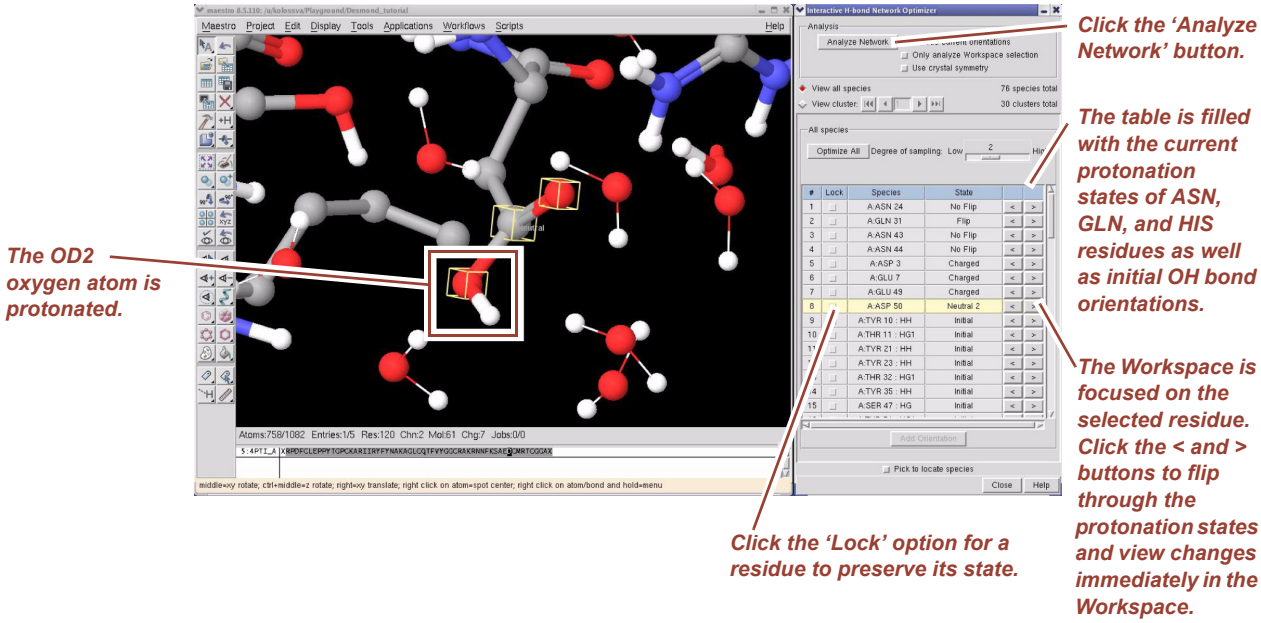
    There are two alternatives in the H-bond Assignment subpanel:

    — The Optimize button launches a comprehensive Monte Carlo search: different protonation states of ASN, GLN, and HIS residues are sampled and OH bonds are flipped to optimize hydrogen bond geometry. As part of this procedure you can switch to exhaustive search mode, as well as decide whether the orientation of crystal water molecules should be sampled.

    — The Interactive Optimizer button launches an interactive tool for hydrogen bond geometry optimization. The interactive optimizer window is shown in Figure 1.10.

    When you click the Analyze Network button, the Protein Preparation Wizard fills in the table with the current protonation states of ASN, GLN, and HIS residues as well as initial OH bond orientations. The table on Figure 1.10 reflects the current state after clicking the Optimize option.

Figure 1.10 Protein Preparation Wizard - Interactive H-bond optimizer



Select an item in the table to focus the Workspace on the selected residue. For example, selecting item 8 in the table focuses the workspace on ASP50 as shown in Figure 1.10. Originally, this ASP residue was charged. By pressing the < and > buttons next to ASP50 in the table you can flip through all the different protonation states and immediately view the result in the Workspace. Figure 1.10 shows the Workspace when the OD2 oxygen atom is protonated (see the selected area). Similarly, residue states and OH orientations of other residues can be set manually and, if desired, locked by checking the Lock option.

Finally, by clicking the Optimize All button you can re-optimize the hydrogen bonding network. For further information about the Interactive H-bond Optimizer click the Help button.

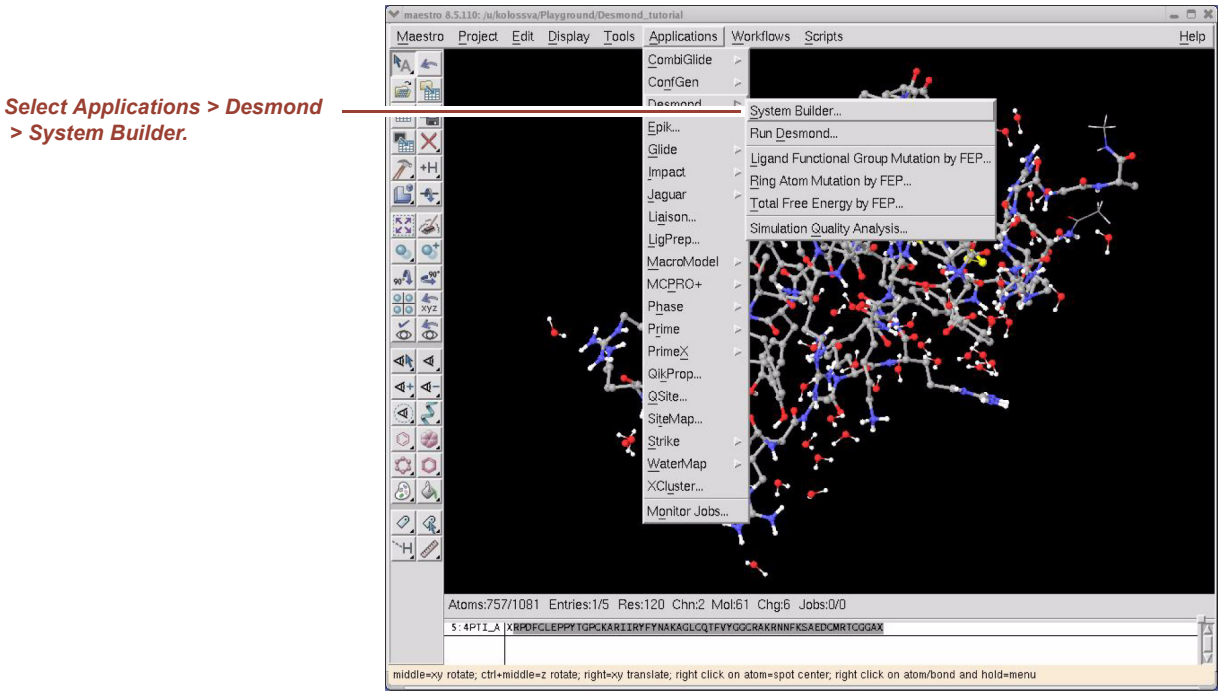
14. Finally, provided that you have a Glide license from Schrödinger, the whole protein structure in the Workspace can be subjected to constrained renement by clicking the Minimize button in the Impref minimization subpanel.

NOTE The preferred renement procedure for MD simulations is to use Desmond's own minimization/relaxation protocol (see below).

NOTE The Protein Preparation Wizard cannot deal with the common case of missing side chains and missing residues that often plagues PDB structures. Before using the Protein Preparation Wizard, make sure that any missing parts of the protein structure are filled in. This tutorial describes a method using Schrödinger's Prime application. See "Preparing Proteins with Missing Residues or Side Chains" on page 29 for useful information on this subject. However, you will need a license from Schrödinger to use Prime. In lieu of Prime, you should use one of several tools available outside the Maestro environment.

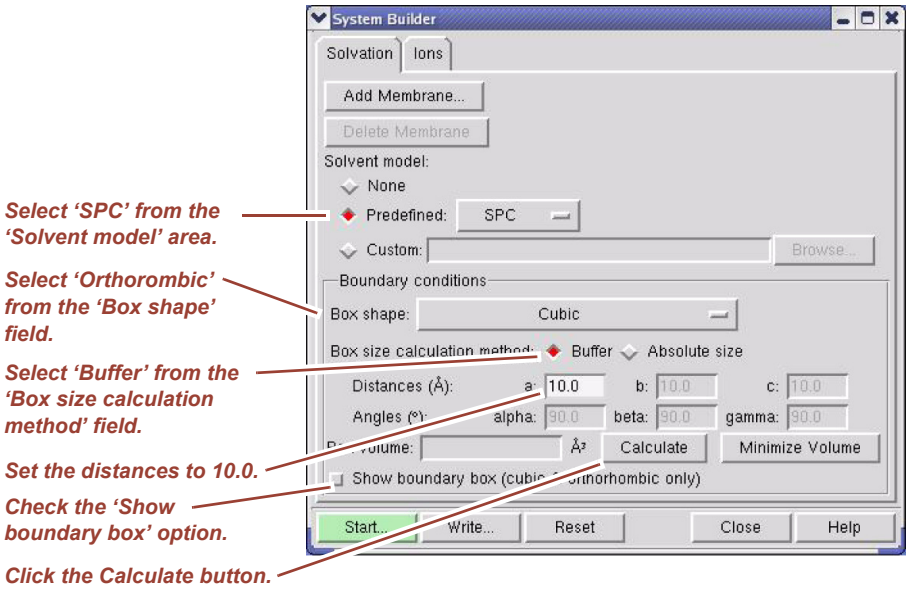
15. The 4pti structure is now ready for preparing a Desmond simulation. Select **Applications > Desmond > System Builder** as shown in Figure 1.11.

Figure 1.11 Launching Desmond System Builder



This launches the Desmond System Builder panel shown in Figure 1.12. The System Builder generates a solvated system that includes the solute (protein, protein complex, protein-ligand complex, protein immersed in a membrane bilayer, etc.) and the solvent water molecules with counter ions.

Figure 1.12 Desmond System Builder panel

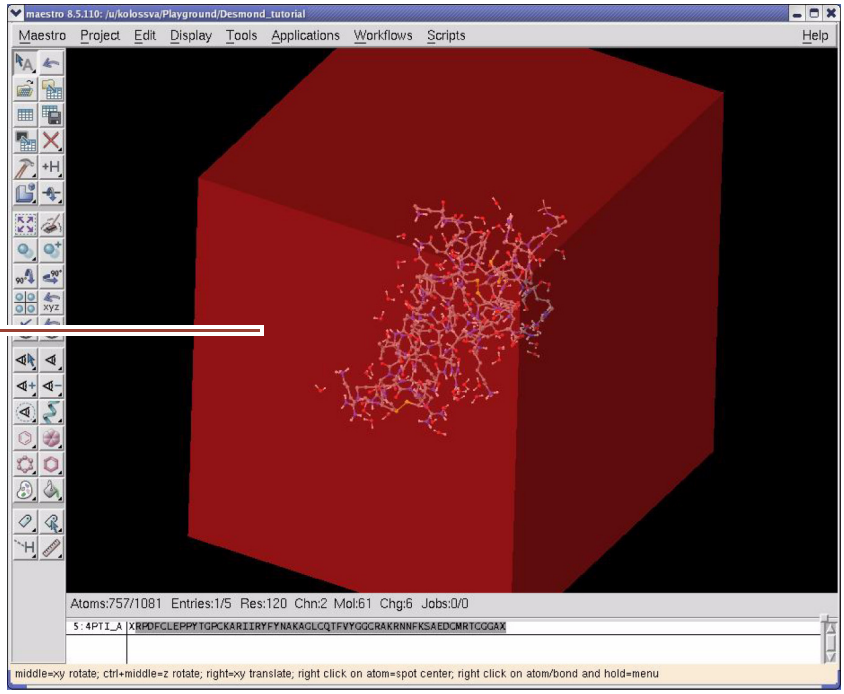


16. Select **SPC** from the Solvent model field.
17. Select **Orthorombic** from the Box shape field.

18. Select **Buffer** from the Box size calculation method field.
19. Enter **10.0** in the Distances (Å) field.
20. Check the Show boundary box option and click the Calculate button. The System Builder can also be instructed to minimize the volume of the simulation box by aligning the principal axes of the solute along the box vectors or the diagonal. This can save computational time if the solute is not allowed to rotate in the simulation box.

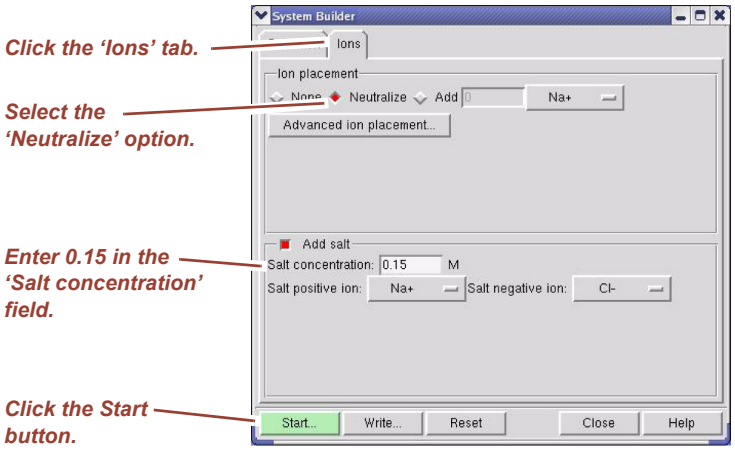
    After setting all these options, the Workspace displays the solute and the boundary box as shown in Figure 1.13.

Figure 1.13 Solute and boundary box in the Maestro Workspace



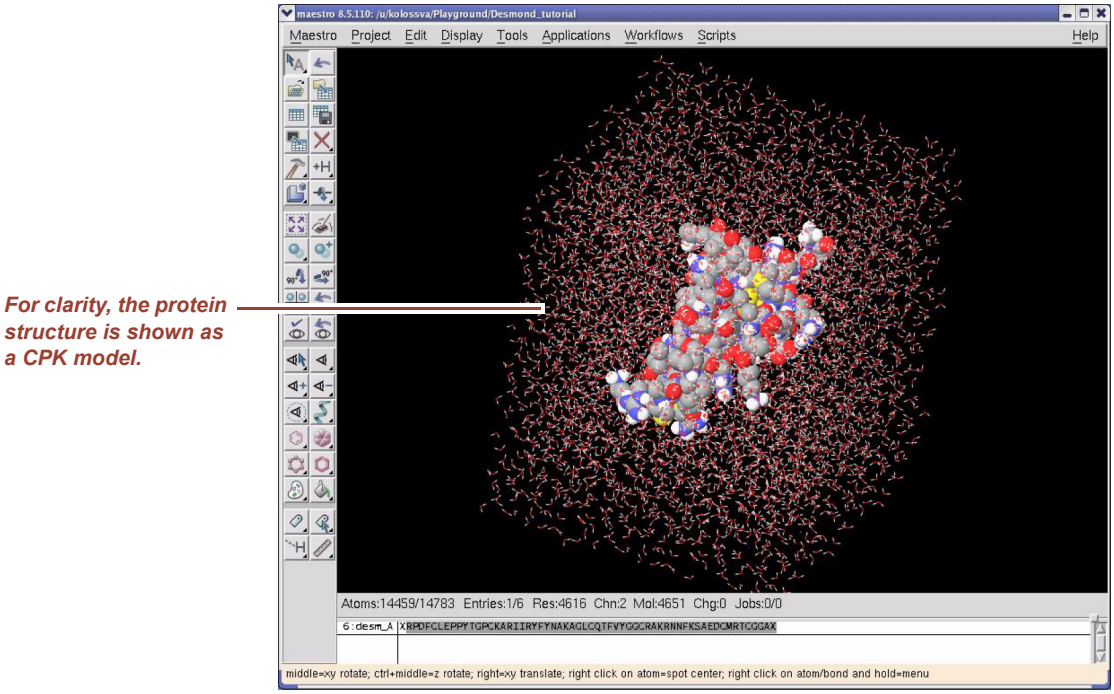
21. Click the Ions tab. The Ions window in the System Builder appears as shown in Figure 1.14.

Figure 1.14 Ions tab in Desmond System Builder



22. In the Ion placement field, select the Neutralize option.
23. In the Salt concentration field, enter **0.150**. This will add ions to the simulation box that represent background salt at physiological conditions.
24. Membrane systems can also be built with the System Builder, but this is deferred to "Setting Up Membrane Systems" on page 47, which covers membrane setup steps in detail.
25. Click the **Start** button to start building the solvated structure. When complete, the solvated structure in its simulation box appears in the Workspace as shown in Figure 1.15. For better clarity the 4pti structure is shown as a CPK model.

Figure 1.15 Solvated protein structure in the Workspace



System Builder saves the whole simulation system in a composite model system file with the extension **.cms**. `.cms` files are essentially multi-structure Maestro files that are suitable for initiating Desmond simulations.

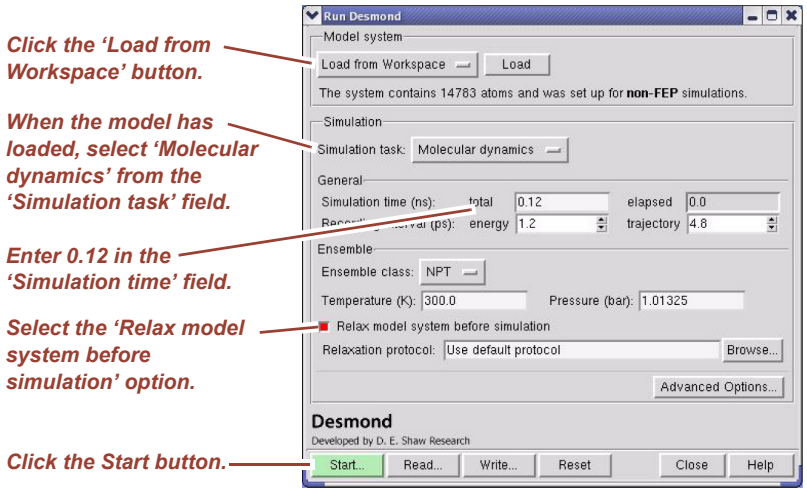
NOTE Maestro automatically assigns the latest OPLS-AA force field parameters available in the Schrödinger Suite to the entire system. If you would rather apply a Desmond provided force field (such as Amber or Charmm force fields, TIP5P water model, or Schrödinger's PFF polarizable force field), you need to process the `.cms` files using the external *Viparr* program (see "Generating Force Field Parameters with Viparr" on page 57).

Now we are ready to perform the Desmond simulation.

Expert users will typically want to start a simulation from the command-line (see "Running Simulations from the Command Line" on page 66). However, for this tutorial we will run it from the Run Desmond window in Maestro.

26. Select **Applications > Desmond > Run Desmond** from the menu bar. The Desmond window appears as shown in Figure 1.16.

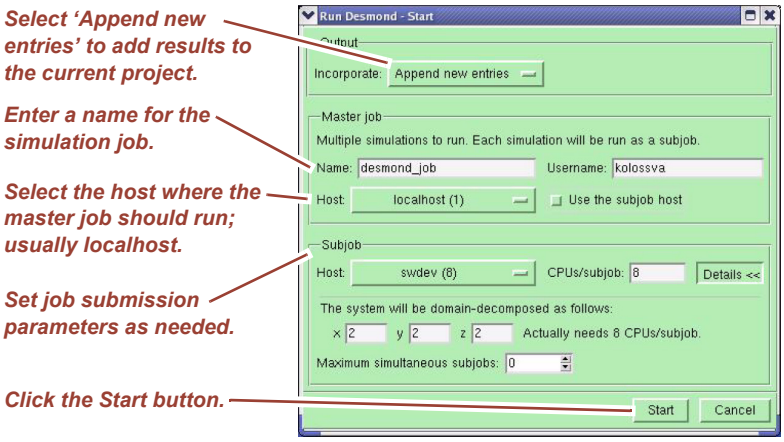
Figure 1.16 The Desmond window in Maestro



27. Import the model system into the Run Desmond environment: select either Load from Workspace or Import from file (and select a .cms file), and then click Load. The import process may take several minutes for large systems. For this example, select Load from Workspace.
28. Select **Molecular dynamics** from the Simulation task field.
29. In the Simulation time field, set the total simulation time to **0.12** ns.
30. Select the **Relax model system before simulation** option. This is a vital step to prepare a molecular system for production-quality MD simulation. Maestro's default relaxation protocol includes two stages of minimization (restrained and unrestrained) followed by four stages of MD runs with gradually diminishing restraints. This is adequate for most simple systems; you may also apply your own relaxation protocols by selecting a customized command script using the Browse button. For more details see "Running MultiSim jobs from the Command Line" on page 67.
31. Select Advanced Options to set parameters for the simulation. Advanced options are covered in "Specifying Desmond Simulation Parameters" on page 60.

32. Click the green **Start** button. The Desmond-Start window appears as shown in Figure 1.17.

Figure 1.17 The Desmond-Start window



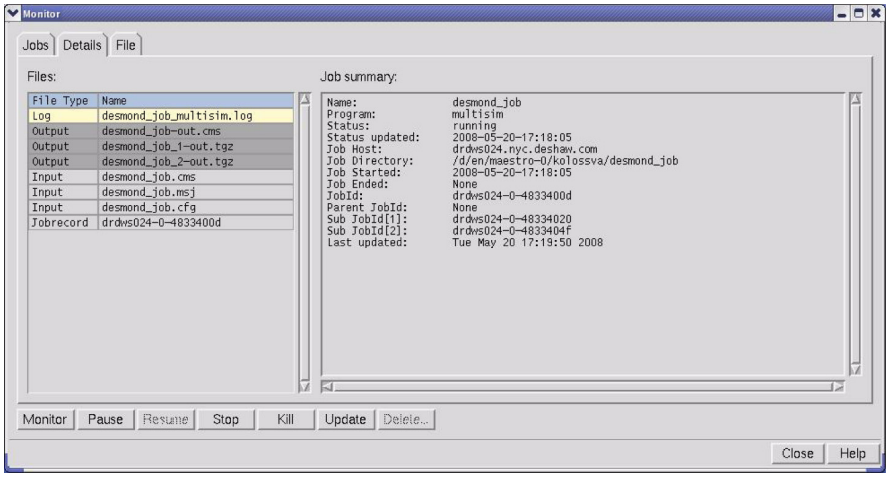
33. Select **Append new entries** from the Incorporate field in the Output area to indicate that results of the Desmond simulation should be added to the current Maestro project (see "Maestro Projects" on page 24).

34. In the Master job area you can give the job a name and specify the username for job submission. Normally, the master job should run on *localhost*, which is your workstation where you launched Maestro. Use the options in the Subjob area to control job submission parameters including the host on which the simulation subjobs will run and the domain decomposition (the number of blocks into which the simulation box will be split in the X, Y, and Z directions) that is compatible with the given number of CPUs. In this example a cluster queue called *swdev* will be used, which allows a maximum of 8 CPUs to be used by any one subjob. The domain-decomposition is 2x2x2. In this particular case only a single subjob will be running at any time; a sequence of relaxation jobs followed by the production run. However, there are other types of simulations (for example, FEP simulations) where multiple subjobs can run simultaneously, in which case you may want to limit the number of simultaneous subjobs submitted to a queue. This can be set in the Maximum simultaneous subjobs field; zero means no limit and any nonzero value limits the number of simultaneously submitted jobs to that value.

35. Click the **Start** button. The Desmond simulation process begins execution.

    Job progress appears in the Maestro Monitor window similar to that shown in Figure 1.18.

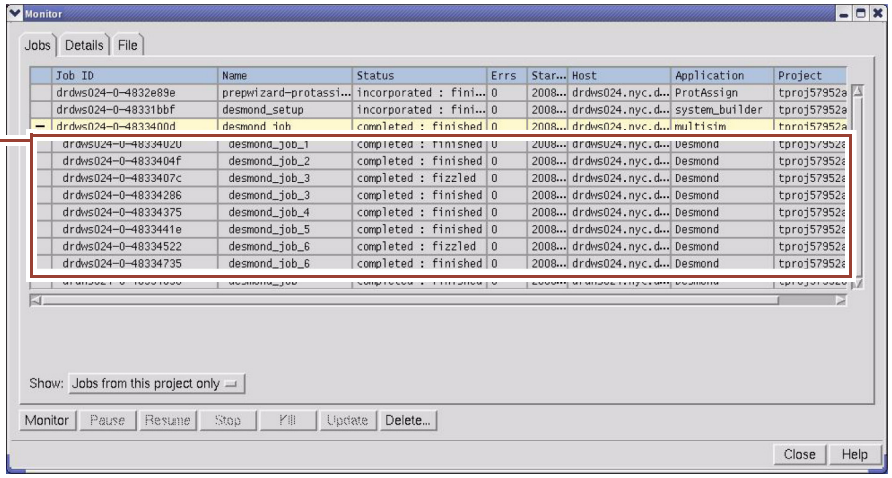
Figure 1.18 The Maestro Job Monitor window



The job (relaxation/equilibration and the production run) will finish in about half an hour and at that point the Jobs tab in the Monitor window will display a list similar to that shown in Figure 1.19.

Figure 1.19 List of Desmond jobs in the Maestro Job Monitor window

Jobs 1-6 represent the relaxation phase.



Jobs numbered 1-6 represent the relaxation phase. Note that two jobs (3 and 6) failed initially (marked as “fizzled”), but the job monitoring system automatically restarted them.

The results of the simulation are saved in multiple files. In this case the file list is as shown in the following table.

Table 1.1 Simulation files

File	Purpose
`desmond_job.cfg`	The Desmond conguration file, which has the simulation parameters.
`desmond_job.cms`	The input file for simulation.
`desmond_job.msj`	The “multisim” command script.

Table 1.1 Simulation files

File	Purpose
desmond_job_multisim.log	
desmond_job-out.cms	The output file (the last frame of the trajectory).
desmond_job_1-out.tgz desmond_job_2-out.tgz desmond_job_3-out.tgz desmond_job_4-out.tgz desmond_job_5-out.tgz desmond_job_6-out.tgz	Compressed tar files containing all the information about the relaxation/equilibration stages.
desmond_job.log	A general log of the simulation.
desmond_job_mon.maegz	A monitoring snapshot file (gzip compressed) that is used to periodically update the Workspace view in Maestro.
desmond_job.ene	The results of the production run are stored in the .ene file (energy terms).
desmond_job_trj	The .trj files store the trajectory snapshots in a _trj directory along with the simbox information.
desmond_job_trj.idx	
desmond_job_simbox.dat	Simbox information.
desmond_job.cpt	A checkpoint file that allows for the bitwise accurate restart of Desmond jobs that crashed for some reason.

You can find the pertinent documentation in the *Desmond User Guide* and the *Schrödinger Desmond User Manual* listed in "Documentation Resources" on page 97. Trajectory analysis is covered in "Visualization and Analysis using Maestro" on page 81 and "Analyzing Trajectories Using VMD" on page 85.

2 Learning the Maestro-Desmond Environment

Overview

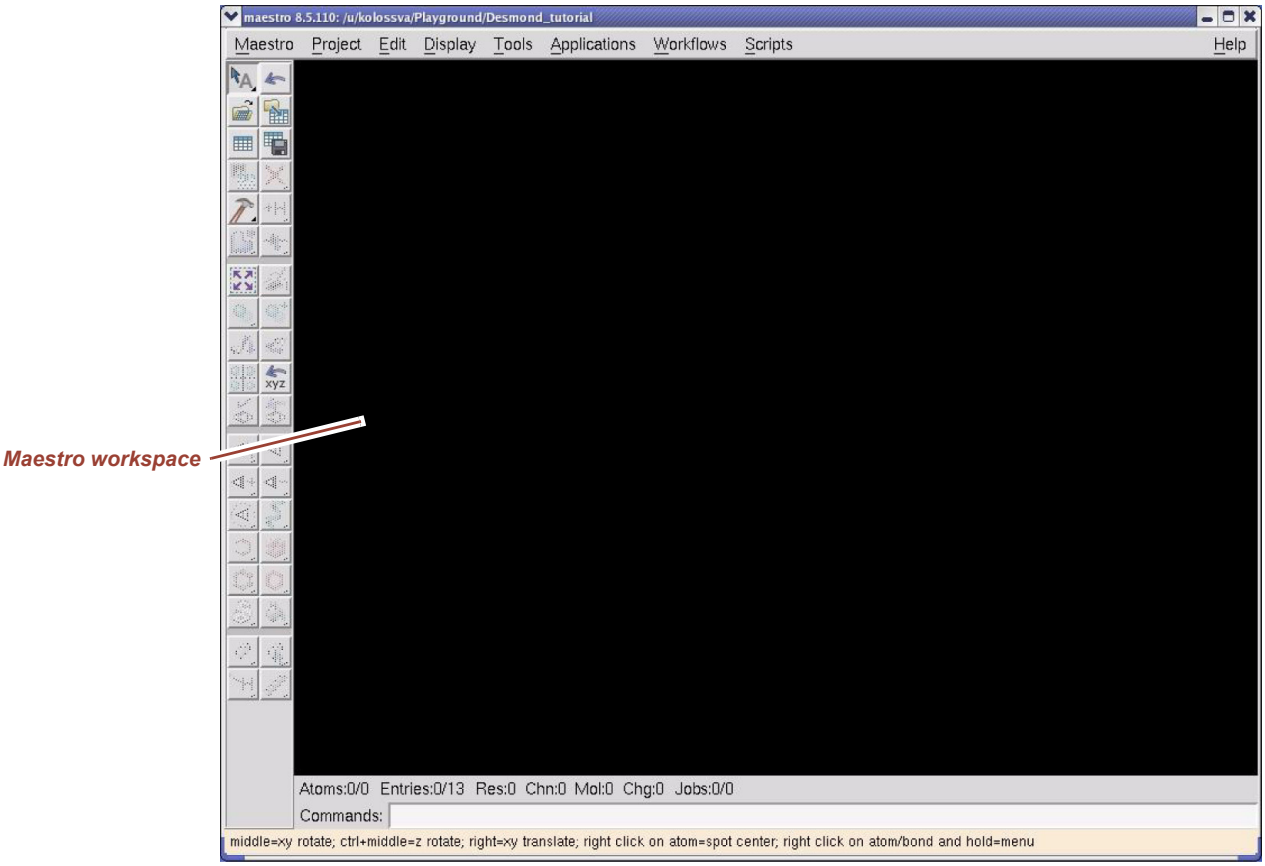
The main Maestro window (shown in Figure 2.1) has a large workspace area, a vertical toolbar along the left side of the window, a menu bar at the top of the window, and a command line at the bottom of the window. The workspace is where you build and manipulate molecular models.

An information bar at the bottom of the window displays information about the molecular model currently in the workspace. The yellow cursor in the workspace is always labeled according to the current context.

When a molecular structure is present in the workspace, all of the buttons in the vertical toolbar are active.

If you move the cursor over any of the buttons, Maestro will display help text within a bubble. Most buttons have a small arrow in the lower right corner. Pressing and holding the left mouse button over this arrow displays more options.

Figure 2.1 Maestro window

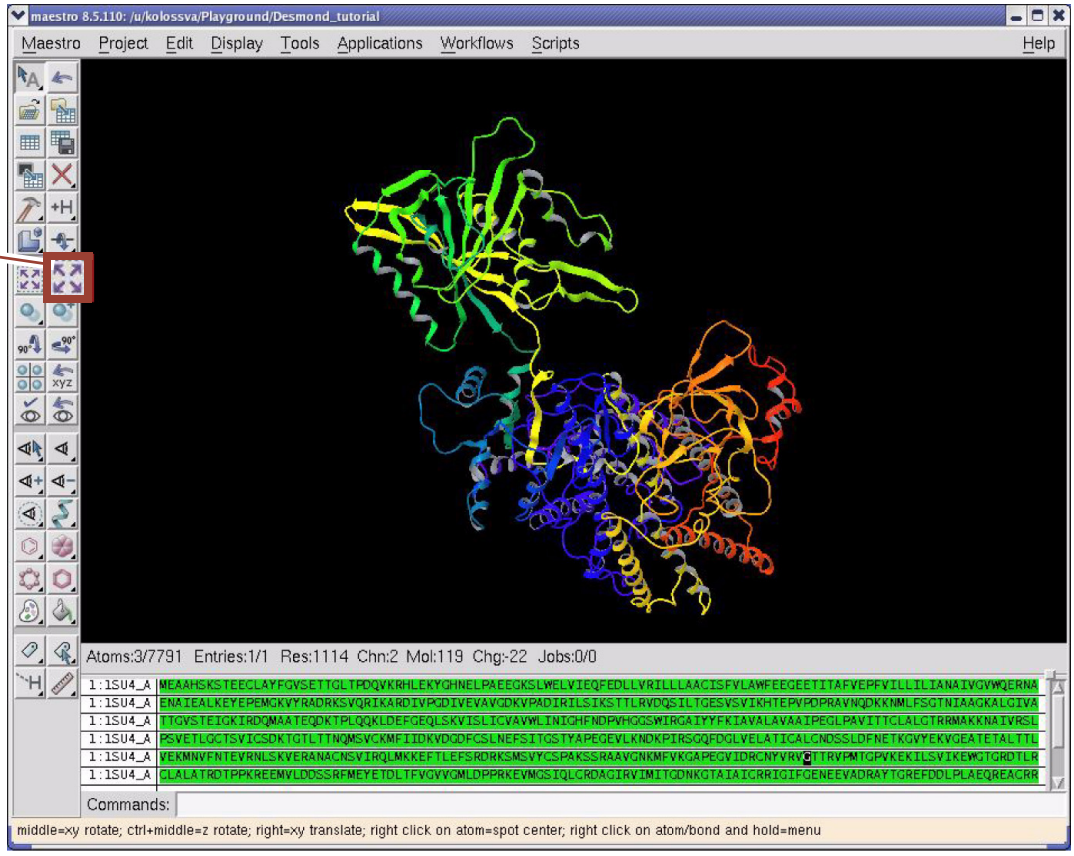


Zooming In and Restoring Full View

When you want to take a closer look at a portion of a molecular model, you can zoom in by right-clicking a particular atom to center the view in that area and then scrolling the mouse wheel to zoom in or out. You can also zoom by pressing the middle and right buttons at the same time.

Return to a full size view of the molecular model by clicking the Fit button in the vertical toolbar as shown in Figure 2.2.

Figure 2.2 Returning to full screen view



Visually Distinguishing Residues and Displaying Atom Information

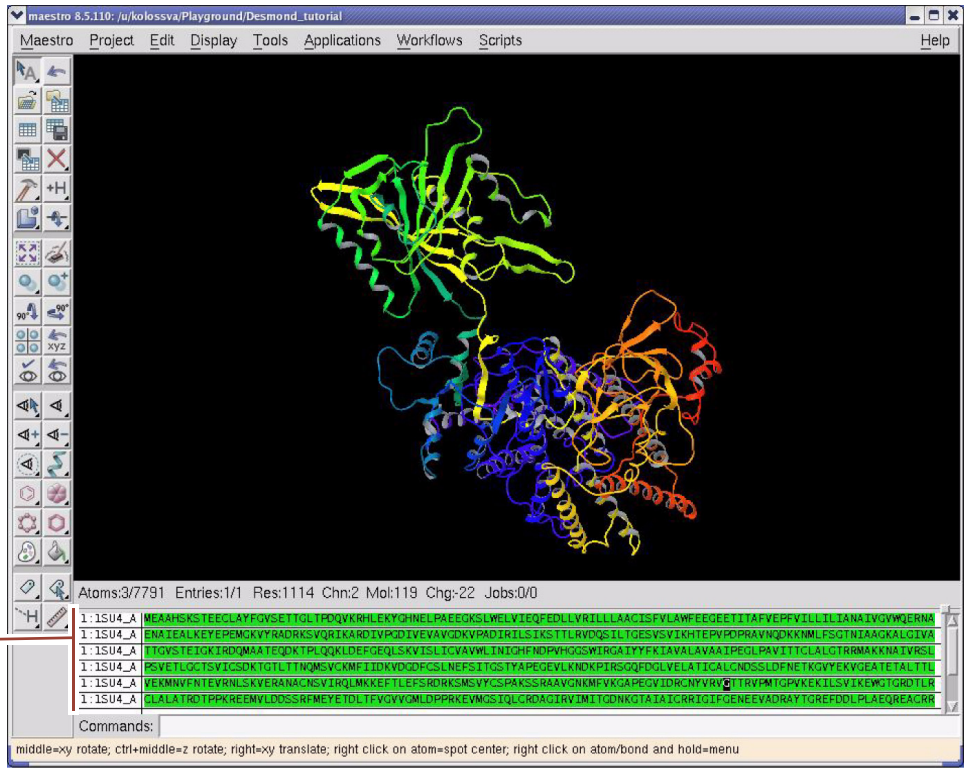
You can visually distinguish different residues, or display residue and atom information quickly, using the Sequence Viewer in Maestro. To use the Sequence Viewer, select **Display > Sequence Viewer** from the menu bar. The Sequence Viewer appears at the bottom of the window as shown in Figure 2.3.

To highlight certain residues in the workspace, click the residue in the Sequence Viewer. To select more than one residue, use SHIFT-click or CTRL-click. The selected residues appear highlighted in yellow within the Workspace.

To display information about an atom within the workspace, move the cursor over the atom. The corresponding residue and atom information is displayed just above the Sequence Viewer.

Double-click in any empty space within the workspace to deselect your selections.

Figure 2.3 Sequence Viewer



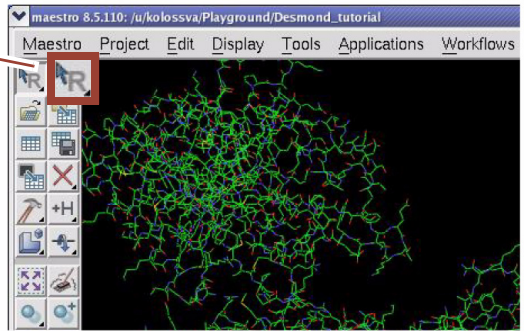
The Sequence Viewer appears at the bottom of the window when you select Display > Sequence Viewer from the menu bar.

Changing Selection Mode in Maestro

You can change the selection mode within the workspace to allow selections based on atom, residue, molecule, and so on. To quickly switch selection mode, click and hold the Select button located in the upper left corner of the vertical toolbar as shown in Figure 2.4. New options appear allowing you to change selection mode. The letter on the Select button indicates the current selection mode.

Figure 2.4 Selecting mode display

Click the 'Select' button to change the selection mode in the Workspace.

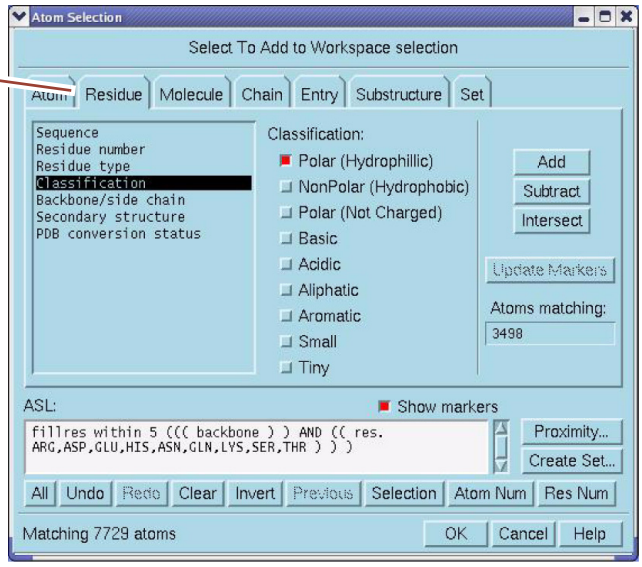


Maestro also offers a powerful tool for selecting arbitrary sets of atoms—the Selection Tool. To open the Selection Tool, choose the Select option from the Select button. From the Selection Tool you can make complex selections on the different tabs of the tool,

including SMARTS. Configurations within the Selection Tool are translated to an ASL (atom specification language) expression. The settings shown in the Selection Tool in Figure 2.5 constrain the workspace to show residues that are at least 5 Å away from the backbone of polar residues.

Figure 2.5 Selection Tool window

Select arbitrary sets of atoms from the tabs of this window to show the desired residues in the Workspace.



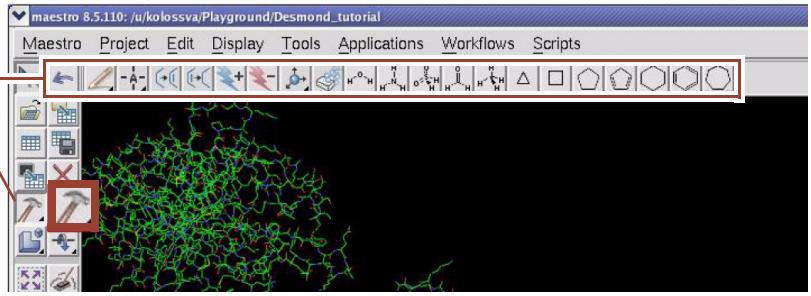
Editing a Molecular Structure

Maestro provides a general structure building and editing tool. It can be accessed either from the vertical toolbar by clicking the `Hammer' button as shown in Figure 2.6, or by selecting different editing tools from the Edit menu.

When you click the `Hammer' button, the most common editing options appear as icons in the top horizontal toolbar. These tools allow you to hand-draw structures, set chemical elements, increment/decrement bond order or formal charge, adjust the position of atoms, clean up 3D geometry, and add common molecular fragments. View help for each toolbar icon by slowly moving the cursor over the icons.

Figure 2.6 Opening the Build/Edit Icon Menu

General tools for building and editing molecular structures appear when you click the 'Hammer' button.



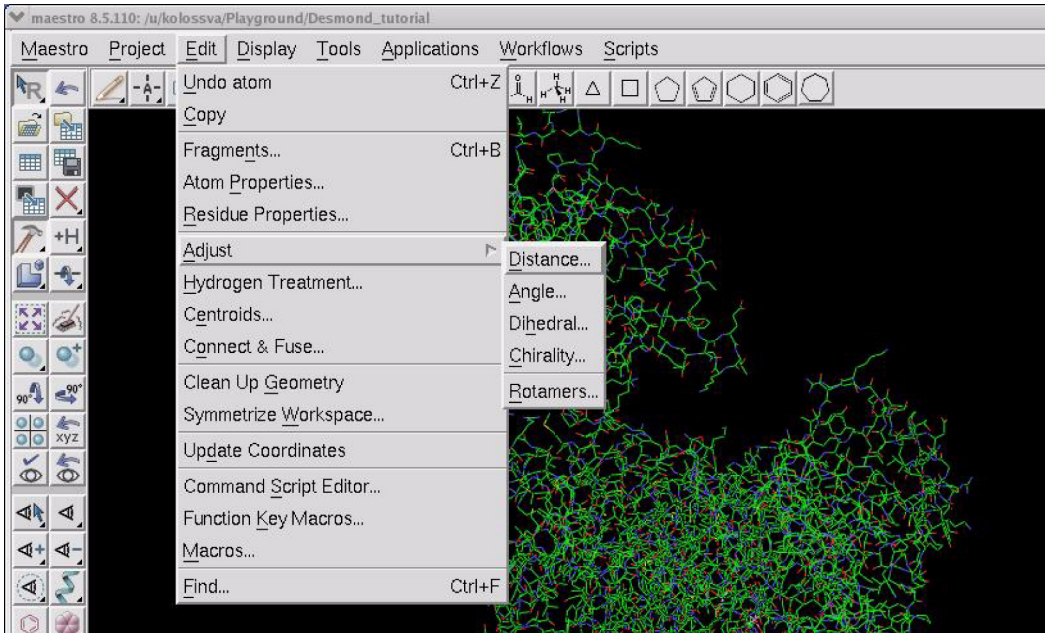
As shown in Figure 2.7, the Build/Edit menu includes:

- Fragments for adding molecular fragments to your structure.

- Atom Properties and Residue Properties for fine-tuning the "chemistry" of your molecule.
- Hydrogen Treatment for adding hydrogen atoms.
- Adjust for making interactive adjustments in local geometry, setting chirality, and generating side chain rotamers.
- Connect & Fuse for connecting fragments and fusing ring structures.
- Clean Up Geometry for producing a rudimentary 3D model of a structure via quick minimization.

Expert use of the Build/Edit menu requires practice; refer to Maestro documentation for detailed information.

Figure 2.7 The Build/Edit Menu List

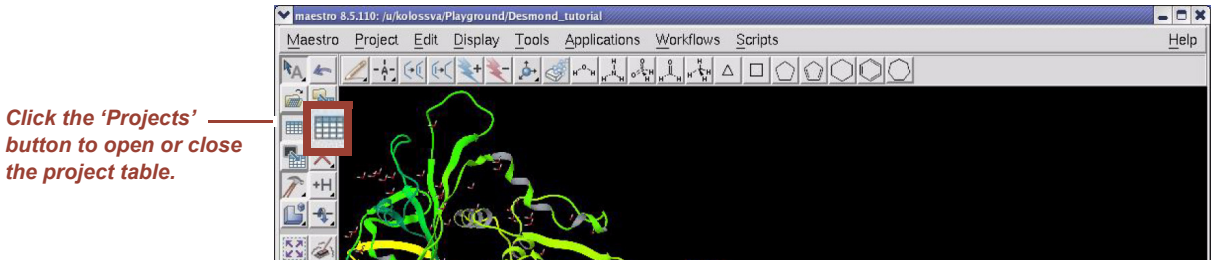


Maestro Projects

Every user-workflow in Maestro is organized into projects. Projects allow you to store all experiments for a particular molecular structure together, keeping your daisy-chained computational and analysis tasks for a structure in one place. A project in Maestro is like a spreadsheet where each and every change to molecular structures (including visual changes, such as coloring or rendering modes) is recorded. The results of computational tasks run in batch mode (such as minimization or conformational searching) are also incorporated in the project.

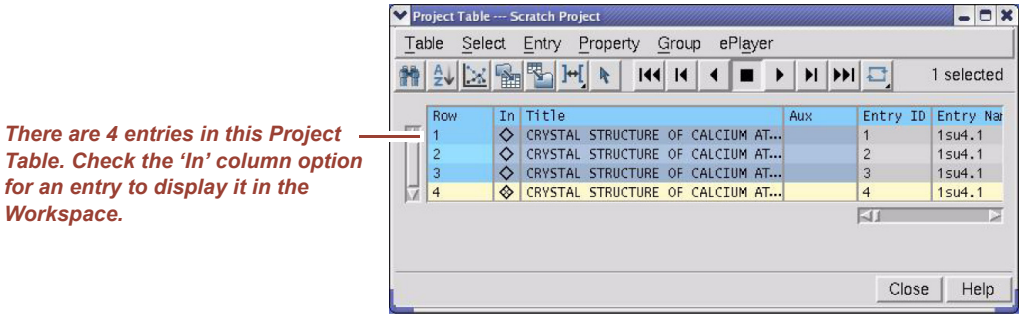
Projects can be opened and closed from the Project pull-down menu or the shortcut buttons on the vertical toolbar. Projects are stored in subdirectories under the current working directory. The project spreadsheet is called the Project Table and can be turned on and off using the Project button in the vertical toolbar as shown in Figure 2.8. The name of the project directory (with extension **.prj**) is shown at the top of the open Project Table.

Figure 2.8 Opening or closing projects



Each row in the Project Table is associated with one or more molecular structures and a number of properties shown in the columns. There is a horizontal scroll bar at the bottom of the table for scrolling through the entire table. Each row of the Project Table is an entry. Entries in the Project Table are created sequentially as the user proceeds with a workflow. The pull-down menus and the action buttons in the Project Table allow you to apply to the project a great variety of organizational tasks (such as sorting, copying, deleting entries), computational tasks (e.g., computing the molecular weight), and display options. Figure 2.9 shows four entries in a Project Table.

Figure 2.9 Project Table in Maestro



The entries in the Project Table shown in Figure 2.9 represent the steps from the import of the original 1su4 structure (Calcium ATPase, which will be used to build a membrane system later on in this tutorial), to the H-bond optimized, final structure obtained by the Protein Preparation Wizard. Selecting an entry allows you to apply computational tasks to the applicable state of the selected entry; for example, you can compute the energy of selected entries, or export selected entries into one or more structure files to be read by other program packages.

To display an entry in the workspace, click the option in the In column of the Project Table for that entry's row. You can display multiple entries in the workspace at a time which is useful, for example, if you wanted to visually analyze the binding modes for several ligands bound to the active site of a protein receptor. You can also apply a rendering mode to multiple entries at once by selecting the option in the In column of the Project Table for every entry you want included, and then applying the rendering mode on the whole conglomorate of entries at once.

If you want to create a new entry from the currently selected entry, right-click in an empty space in the workspace and select Create Entry From Workspace from the popup menu.

There is no need to manually 'save' a project because every action you perform on a structure in Maestro is automatically recorded in its Project Table. Project Tables can be exported in tab- or comma-delimited file format (.csv) to be read by spreadsheet programs. Likewise, you can import tables created elsewhere into Maestro as Project Tables.

The project directory has a compressed, back-up copy of the entire project, but it is good practice to keep around a separate back-up copy of important project directories, because sometimes files in the project directory can get corrupted rendering the project unreadable.

Useful Tools in Maestro

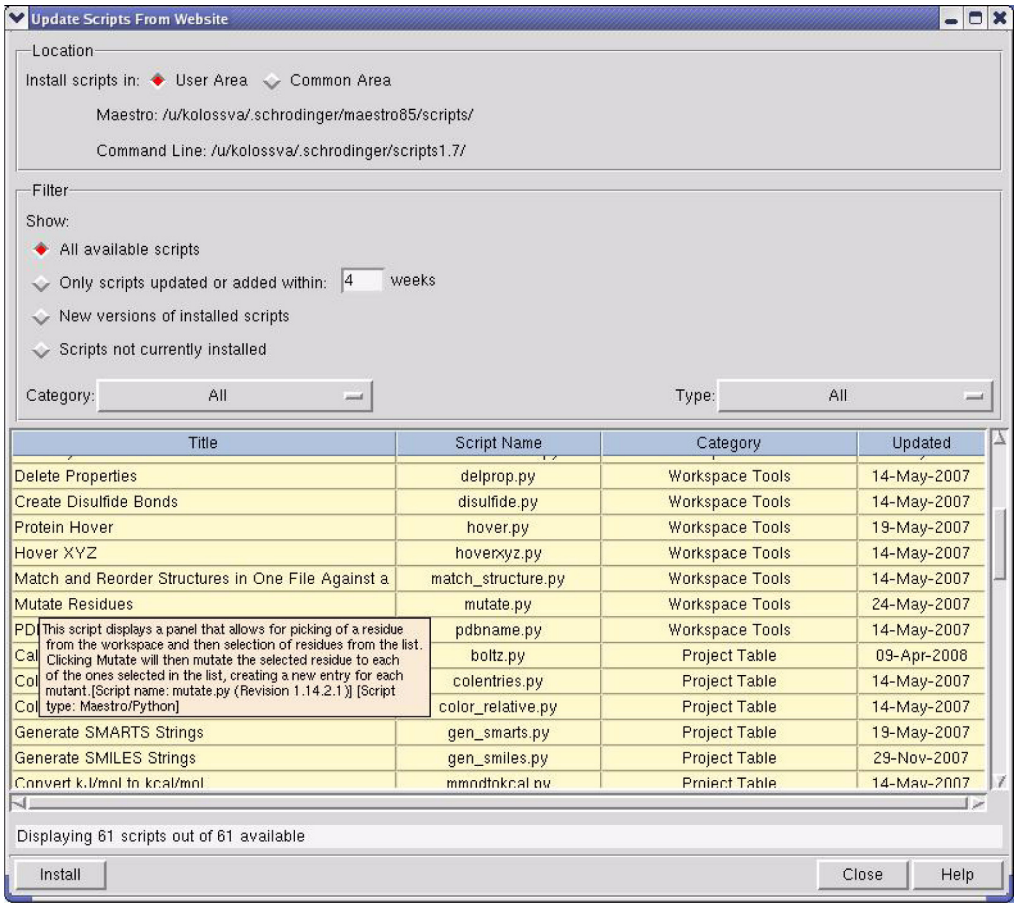
With no aim at providing a complete list, here are some useful tools you might want to explore. Note, however, that the tools under the Applications menu require separate licenses from Schrödinger, LLC.

- **Maestro > Save Image** saves a high quality screen capture of the workspace.
- **Project > Make Snapshot** saves the current state of a project so it can be recovered later if necessary.
- **Edit > Clean Up Geometry** quickly generates a reasonable 3D-model of a structure that was drawn by hand.
- **Edit > Command Script Editor** is a powerful yet simple scripting tool. Each Maestro command you execute (including graphics tasks) is saved during a Maestro session. The Command Script Editor allows you to cut-and-paste selected commands and save them in an executable script. For example, if you apply a sophisticated rendering scheme on a protein complex structure and want to repeat the same rendering on some other system at a later time, the corresponding sequence of commands can be saved in a script and applied to the new system. Use of the Command Script Editor does not require any knowledge of the Maestro command language.
- **Display > Surfaces** generates a variety of molecular surfaces that can be added to the Project Table.
- **Tools > Sets** defines arbitrary sets of atoms to be used for quick rendering or different kinds of computations.
- **Tools > Protein Structure Alignment** automatically superposes homologue protein structures. This is useful when you want to bring multiple X-ray structures from different labs into a common coordinate frame using protein structure alignment.
- **Tools > Protein Reports** generates an extensive table of useful structural information for the selected entry.
- **Applications > Glide** is a versatile, fast docking tool.
- **Applications > Ligprep** prepares small molecules by fixing bond orders, protonation and tautomerization states, stereo chemistry, and so on.
- **Applications > MacroModel** provides a plethora of molecular mechanics tools including large scale conformational searches, flexible docking, free energy calculations, and so on. MacroModel can be used with a variety of different force fields and implicit solvent models.
- **Scripts menu.** Virtually all of Maestro's functionalities can be accessed directly from Python scripts. Maestro provides a straightforward mechanism for implementing these scripts so that they become permanent, menu-driven options in Maestro. There is an extensive library of Maestro scripts that can be downloaded from the Schrödinger website (www.schrodinger.com) and that is continuously updated and expanded. To install scripts, select **Scripts > Update** and select scripts to include, as

shown in Figure 2.10. Move the cursor over the title of a script to display the function of the script. Click the title of a script to select it, and click the Install button. The scripts are automatically downloaded and installed.

NOTE You only need to install a script once. Maestro remembers it when it is requested later. If a new version of a script becomes available, installation will replace the old version with the new script.

Figure 2.10 Installing or updating Maestro scripts from the Schrödinger website



3 Preparing Proteins with Missing Residues or Side Chains

Overview

Most protein structures in the PDB database have missing residues or side chains with missing atoms, mainly as a result of unresolved electron density. The Protein Preparation Wizard does not address these issues, and so more work needs to be performed on the model to prepare it for simulation. In many cases, the missing atoms must be filled in. In other cases, you might have a complete experimental structure of a protein available, but you want to simulate a mutant.

Although the Edit menu provides tools which can be used to fill in missing parts of a protein structure, or replace one amino acid side chain with another for generating a mutant, manually filling in entire residues—or even just a side chain—can be an extremely challenging task that requires a lot of hands-on experience with Maestro. Maestro offers a tool called Prime that can be used to automatically build a three-dimensional model of a complete amino acid sequence. Note, however, that you need a license from Schrödinger, LLC to use Prime. Of course, other tools outside the Maestro environment can also be used to fill in missing residues or side chains.

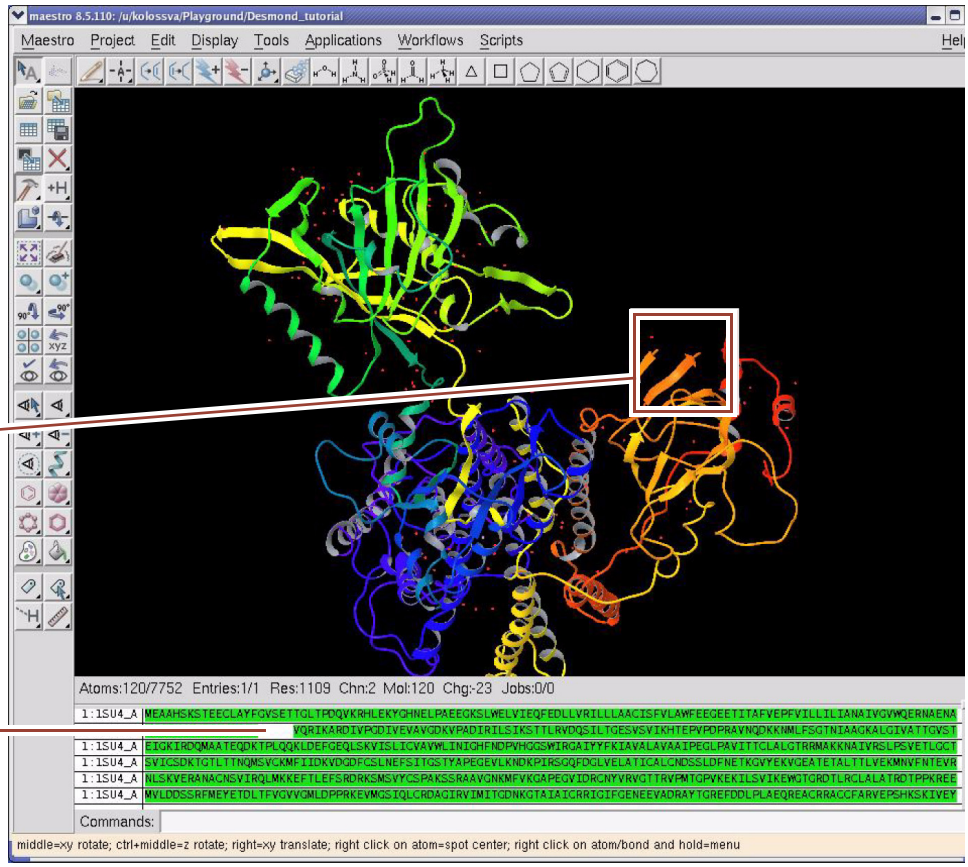
Running Prime

While Prime is typically used as a three-dimensional structure prediction tool because of its comparative modeling and sophisticated ab-initio loop prediction, it can also be 'fooled' into filling in missing residues and side chains in an incomplete protein structure, and building a three-dimensional model of a complete amino acid sequence. This section explains the protein preparation aspects of using Prime.

To run Prime:

1. Download the 1su4 structure as shown in Figure 1.4 on page 5. This structure has all residues present in the PDB file, but as a working example, we will delete a small loop by hand-editing the PDB file.
2. Delete residues 132-136 from the 1su4 PDB file by hand (the original PDB file should be in the working directory after the download) and save it as **1su4_missing_loop.pdb**.
3. Import the modied PDB structure into Maestro as shown on Figure 1.3 on page 4. The truncated structure replaces the full structure in the Workspace where the missing loop is called out in a red box in Figure 3.1.

Figure 3.1 Missing loop in the 1su4 protein



The loop between the two orange ribbons is missing.

The white gaps in the residues (shown in green) represents the missing loop.

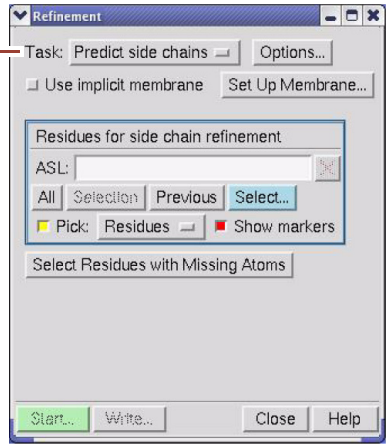
4. Apply Prime to the truncated 1su4 protein structure with the missing loop.

NOTE Running Prime can be computationally intensive. To save time, you can distribute Prime jobs among multiple CPUs.

NOTE If your protein only has missing side chains (no missing residues), you can select the Predict side chains task under Applications > Prime > Refinement to automatically fill in the missing residues without running the entire Prime structure prediction protocol as shown in Figure 3.2.

Figure 3.2 Automatic side chain prediction in Prime

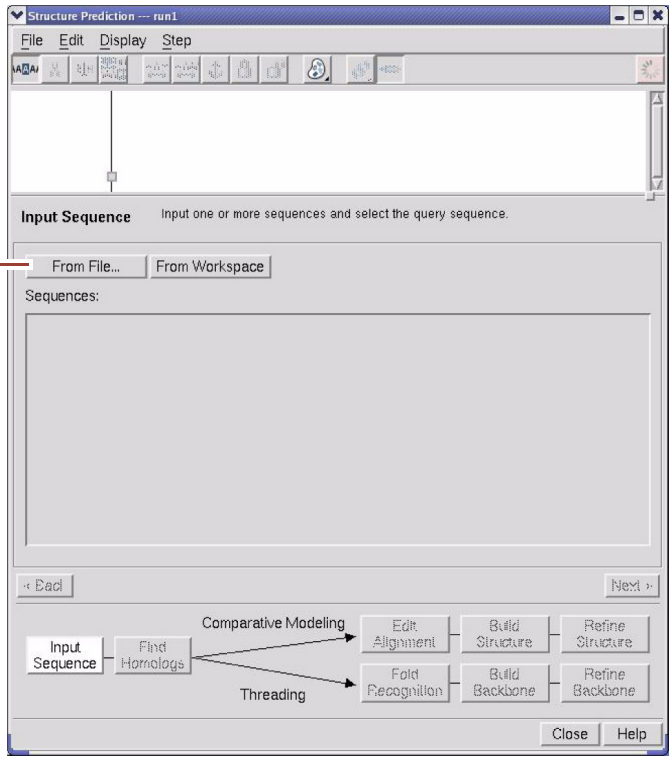
Select 'Predict side chains' from the 'Task' field to automatically fill in missing side chains.



a. Select **Applications > Prime > Structure Prediction** from the menu bar. The Structure Prediction window appears as shown in Figure 3.3.

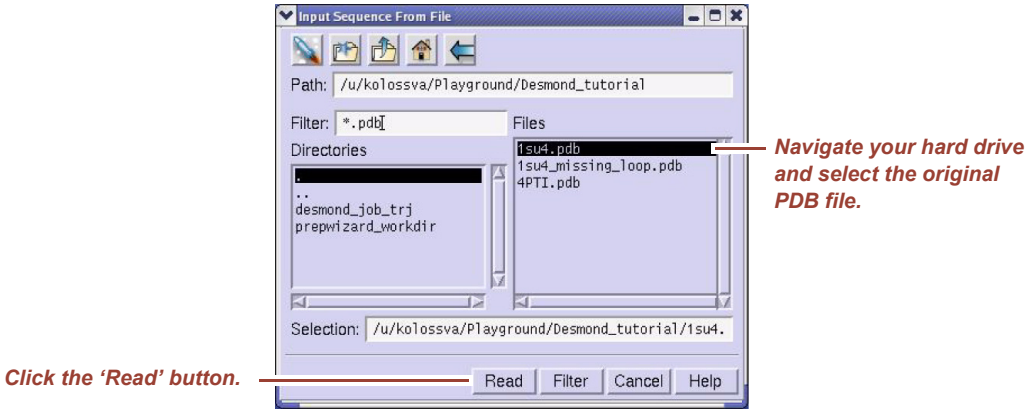
Figure 3.3 Prime Structure Prediction window

Click 'From File...' to input a sequence from a file.



b. Click **From File** under Input Sequence. The Input Sequence From File window appears as shown in Figure 3.4.

Figure 3.4 Input Sequence From File window in Prime

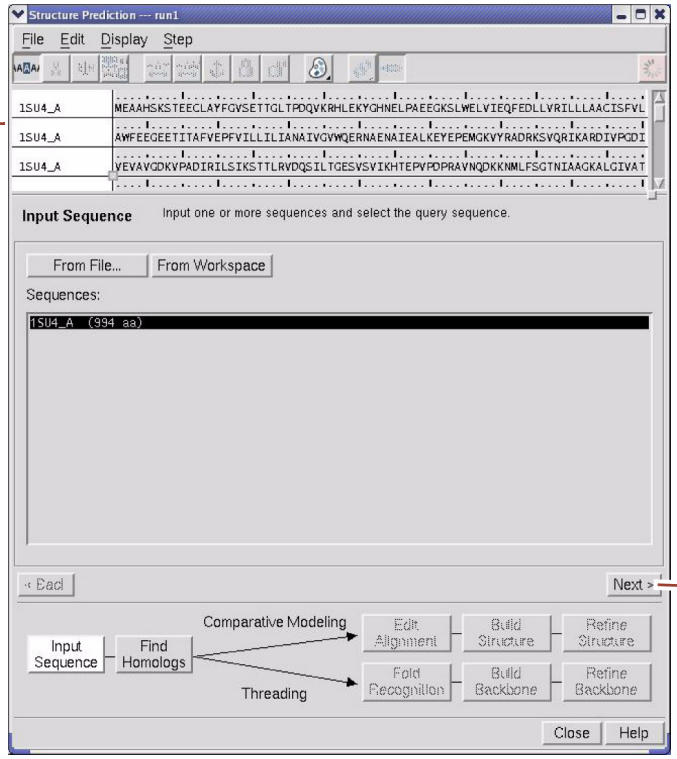


c. Navigate your directory structure and select the original 1su4 PDB file from the Files area of the window, then click Read.

   The full sequence (994 residues) is read from the SEQRES fields and can be seen at the top of the Structure Prediction window as shown in Figure 3.5.

NOTE Reading in the sequence from a PDB file is generally not a good idea. Often a few residues are missing in the protein structure from either or both termini. Since we do not want Prime to build the head or tail of the structure, we need to truncate the sequence by hand, editing the SEQRES fields to ensure that the SEQRES fields start at column 20 in the PDB file and have a total of 13 residues (except for the last field). The number of residues should also be decreased accordingly. Since, however, Prime can read all kinds of protein sequence file formats (FASTA, NEXUS, and so on), you should use one of these formats.

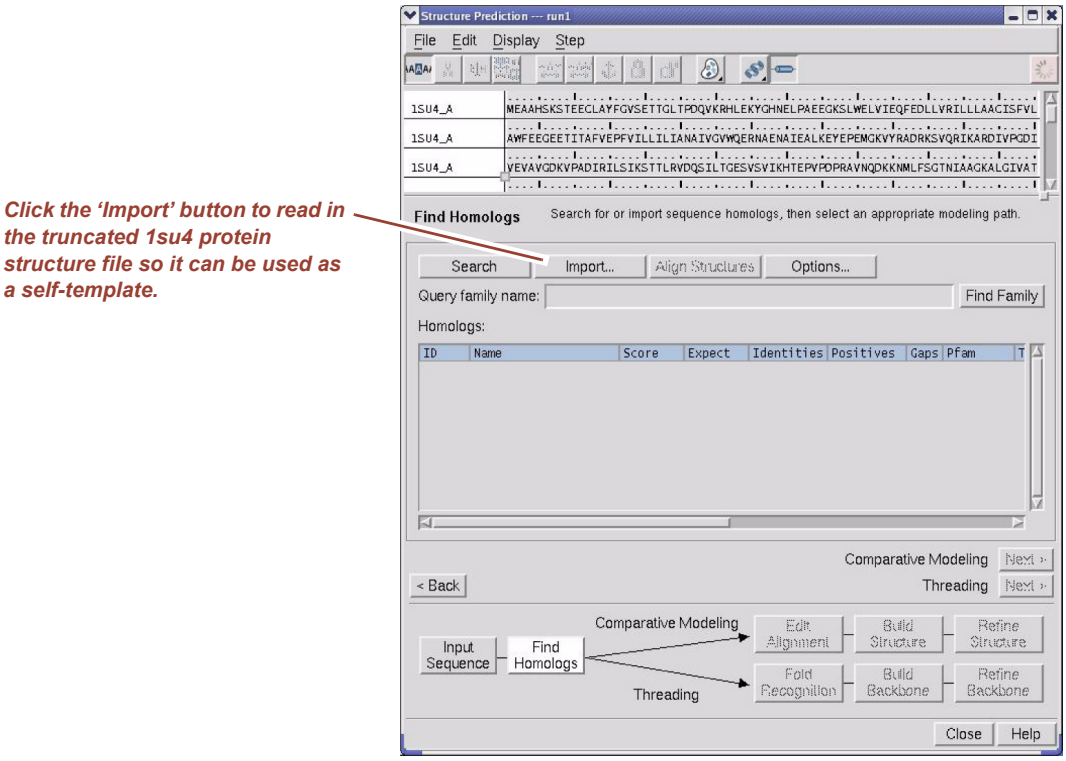
Figure 3.5 Full sequence in the Prime Structure Prediction window



d. When you are finished reading the input sequence, click **Next**.

At this point Find Homologs is highlighted in the workflow diagram at the bottom of the window as shown in Figure 3.6.

Figure 3.6 Prime Find Homologs window

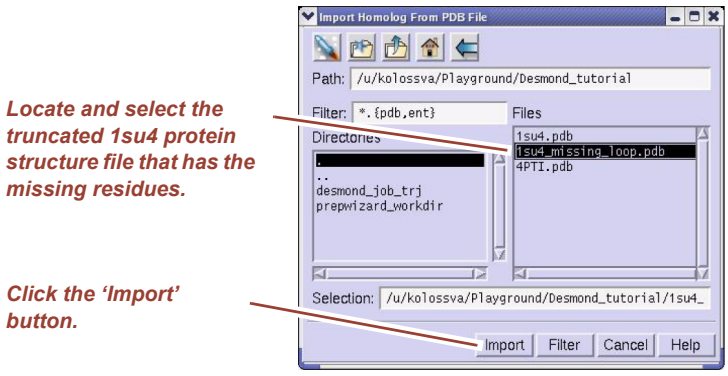


Now it is time to 'fool' Prime. Instead of reading the structures of sequence homologues to be used as structural templates—as we would do for comparative modeling—we will read in the truncated 1su4 protein structure file so it can be used as a self-template.

e. Click **Import** under Find Homologs.

The Import Homolog From PDB File window appears as shown in Figure 3.7.

Figure 3.7 Import Homolog From PDB File window in Prime



f. Select the truncated 1su4 protein structure file that has the missing residues from the Files area. In this example, it is the **1su4_missing_loop.pdb** file which was manually altered to remove residues 132-136. Click **Import**.

The Structure Prediction window reappears as shown in Figure 3.8. There is a single yellow line in the Homologs area of the window that shows quality measures for the sequence alignment. Since this is essentially a self-alignment, the sequence identity is 100%.

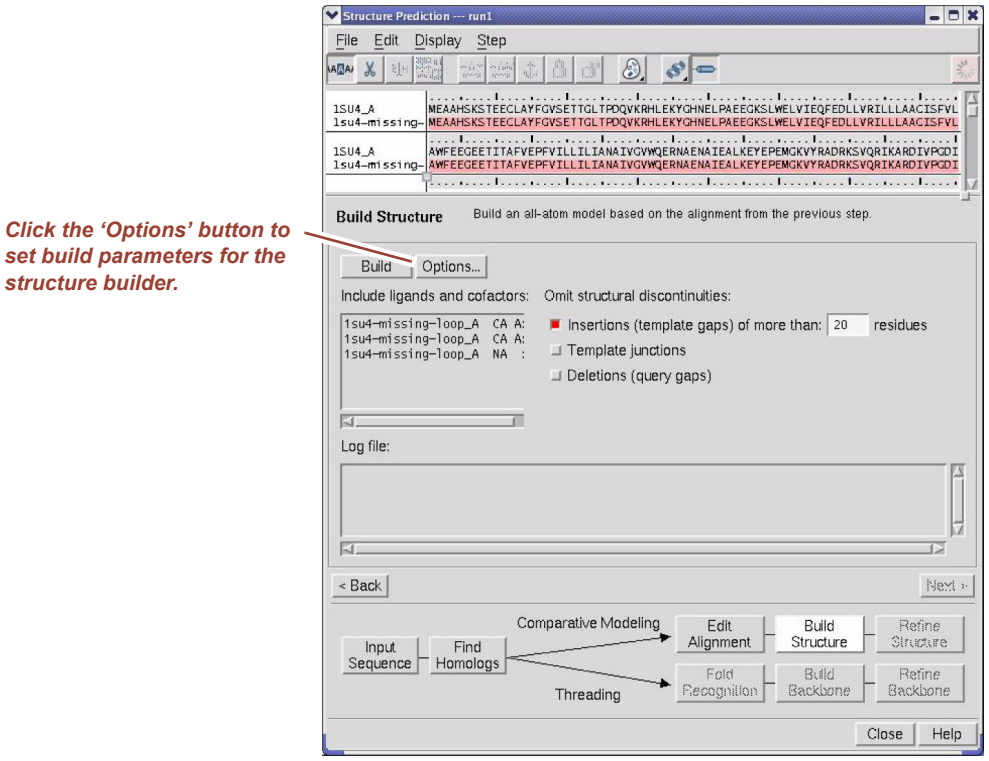
If you click the left tip of the yellow line (which is in the ID column), you see the sequence alignment at the top of the panel. The sequence alignment and the incomplete structure in the workspace are color coded by residue type.

Figure 3.8 Sequence alignment in the Prime Structure Prediction window

g. Click **Next**. The next phase in the process is Edit Alignment. Since this is self-alignment, we do not need to perform this step. Click **Next** again. Note that Prime may pop up a warning message, but ignore it.

At this point Build Structure is highlighted in the workflow diagram at the bottom of the window as shown in Figure 3.9.

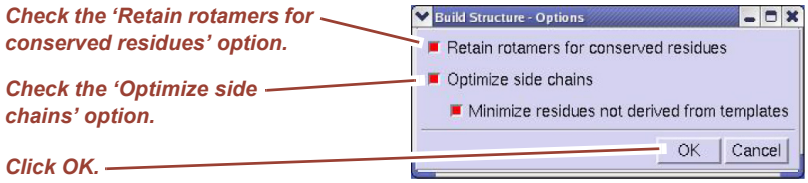
Figure 3.9 Prime Build Structure window



h. Click **Options**.

The Build Structure–Options window appears as shown in Figure 3.10.

Figure 3.10 Prime Build Structure – Options window



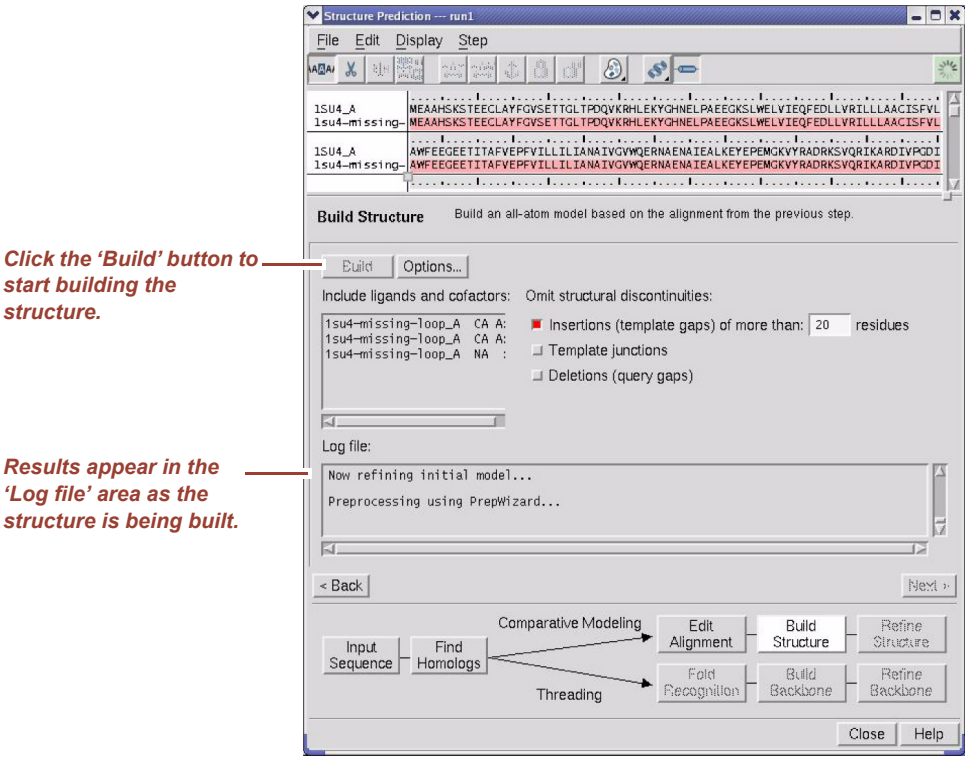
i. Check the Retain Rotamers for conserved residues option to ensure that none of the atoms in the X-ray structure (the self-template structure) are allowed to move since the self-template residues are considered to be conserved residues by definition.

j. Check the Optimize side chains option to allow the structure builder to optimize rotamers of side chains as it builds them.

k. Click **OK**.

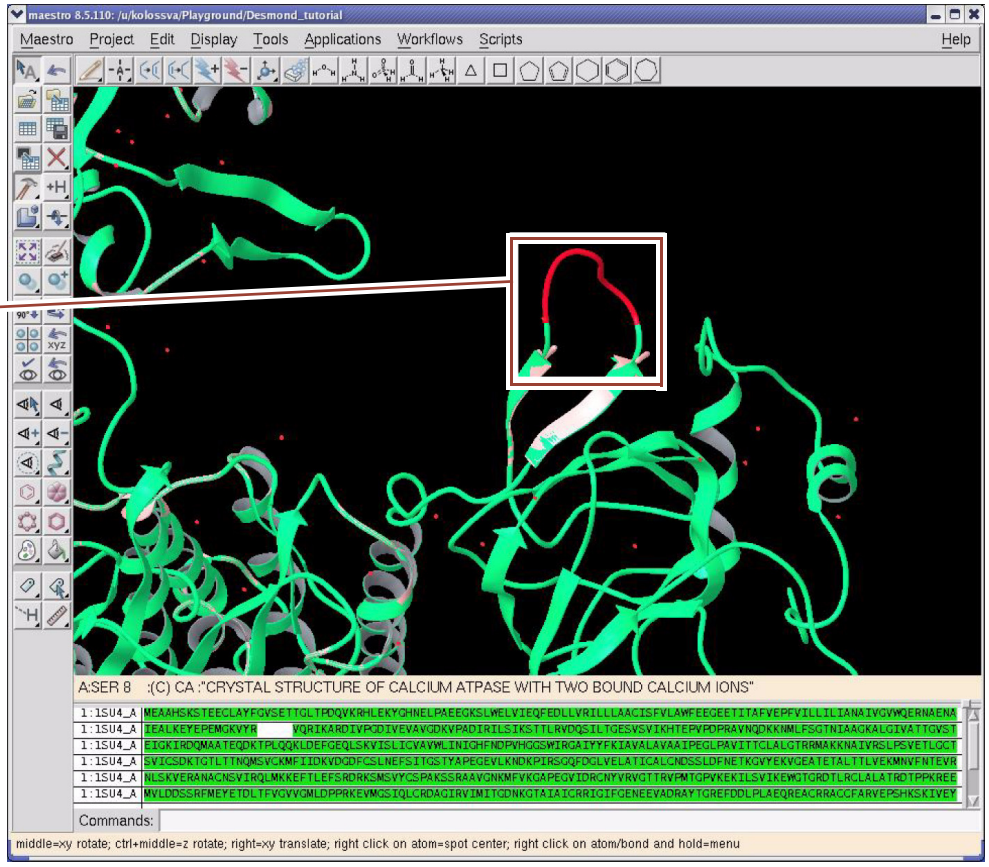
l. Back in the Structure Prediction window, click the **Build** button as shown in Figure 3.11.

Figure 3.11 Prime loop building in progress



Prime begins building an all-atom structure. The procedure takes a few minutes during which a green icon rotates at the top right corner of the window to indicate computations are underway. Log information appears in the Log File window. When the build procedure is successfully completed, the built-in loop appears in the Workspace as shown (colored red) in Figure 3.12.

Figure 3.12 Built in loop shown in the Workspace



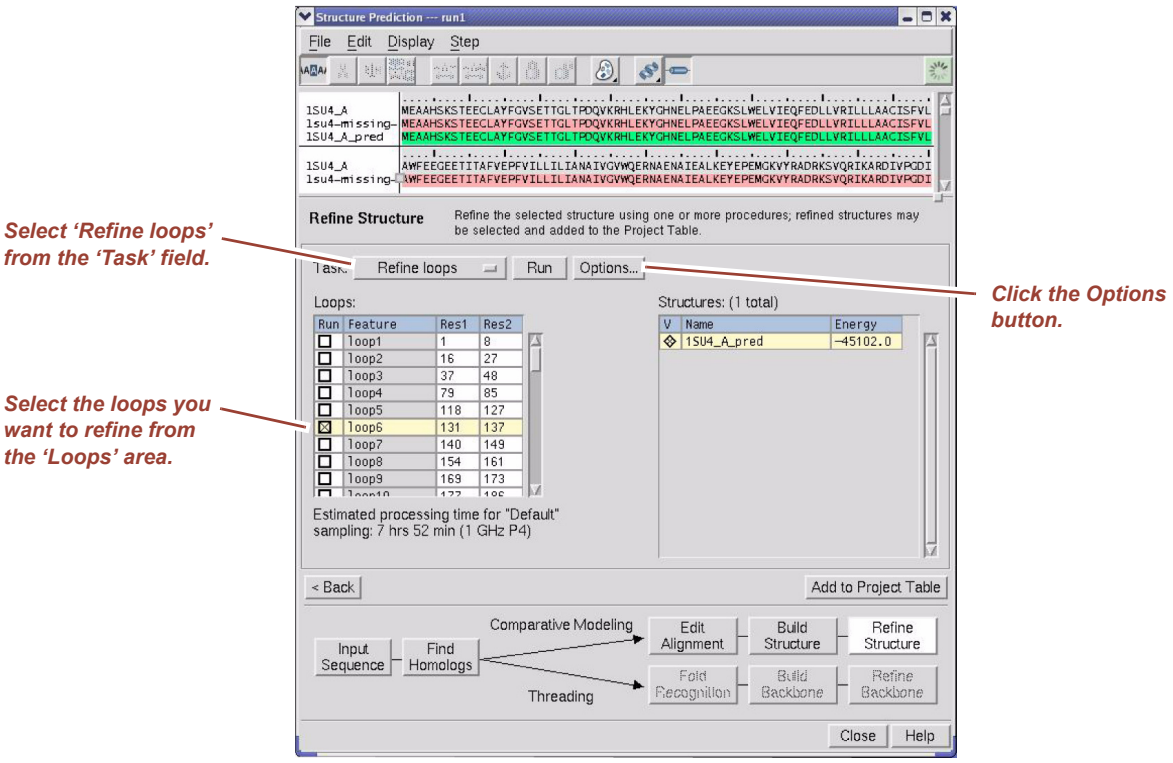
m. Click **Next** in the Structure Prediction window.

At this point Refine Structure is highlighted in the workflow diagram at the bottom of the window as shown in Figure 3.13.

Structure refinement includes loop refinement, side chain prediction, and simple minimization. Side chain prediction is generally fairly reliable and loop prediction is also reasonable up to 5-6 residues. Multiple loops can be refined simultaneously.

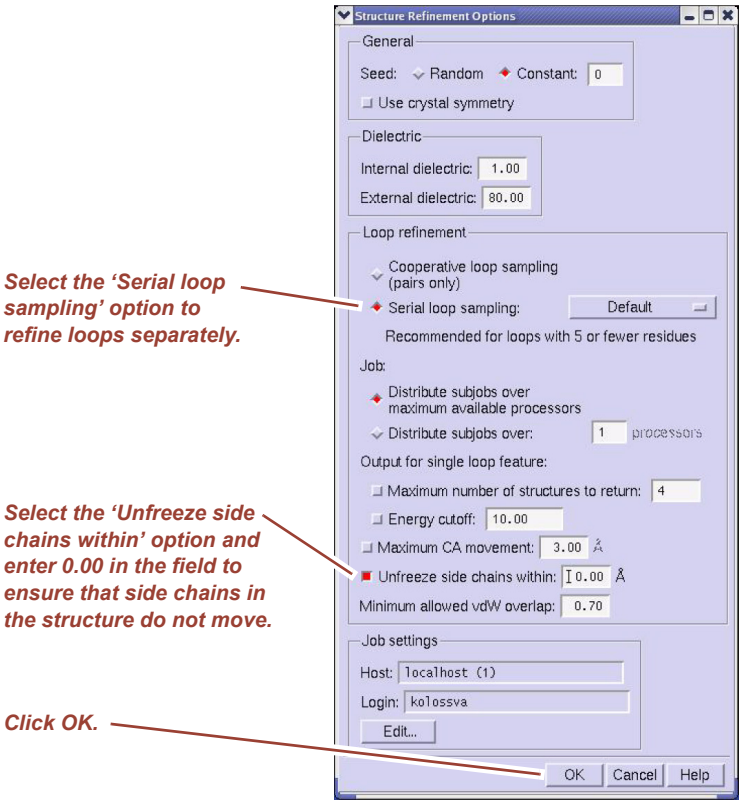
n. Select the Refine loops task and in the Loops area of the window, select one or more loops for refinement. Selected loops appear with yellow highlight as shown in Figure 3.13. For this example, we select loop6 (131-137) which contains the desired residues (132-136) as well as joining residues. You can hand-edit the first and last residue of any loop in the table by double-clicking the value and entering a new one.

Figure 3.13 Prime Refine Structure window



o. Click **Options** to set a number of search parameters. The Structure Refinement Options window appears as shown in Figure 3.14.

Figure 3.14 Structure Refinement Options window in Prime



p. Select the Serial loop sampling option to refine loops separately.

NOTE Serial loop sampling has different levels: default, extended (low, medium, high), and ultra extended. The default option is only recommended for loops up to five residues long. Our loop is a borderline case with 7 residues, but it should be adequate for this exercise. The default mode is reasonably fast: the test case takes only about 15-20 minutes to complete on a single CPU. Even the low-extended mode would run for much, much longer.

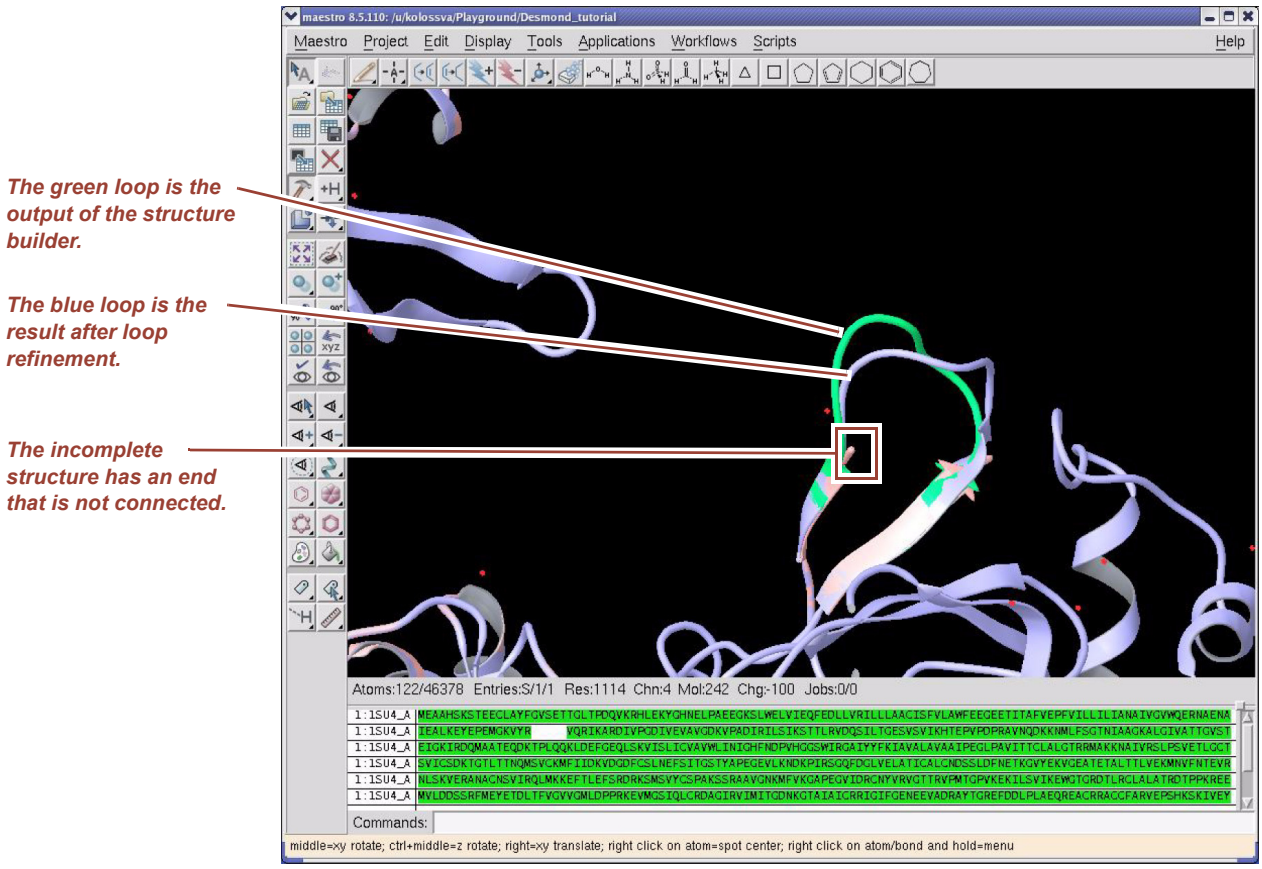
q. Enter zero (0.0) in the Unfreeze side chains within field to ensure that side chains in the X-ray structure do not move. This limits the size of the sphere within which side chains are unfrozen to facilitate better loop conformations by finding good interactions.

r. Click **OK** to store your structure refinement preferences.

s. In the Structure Prediction window, click **Run** under the Refine Structure area.

Structure refinement begins. Upon completion, the refined loop appears in the Workspace. As shown in Figure 3.15, the salmon colored structure is the incomplete structure, the green loop is the output of the structure builder, and the blue loop is the result after loop refinement (which has a slightly different conformation). For comparison, Figure 3.16 shows a superposition of all three loops including the original loop from the X-ray structure in red.

Figure 3.15 Workspace model after Prime structure refinement



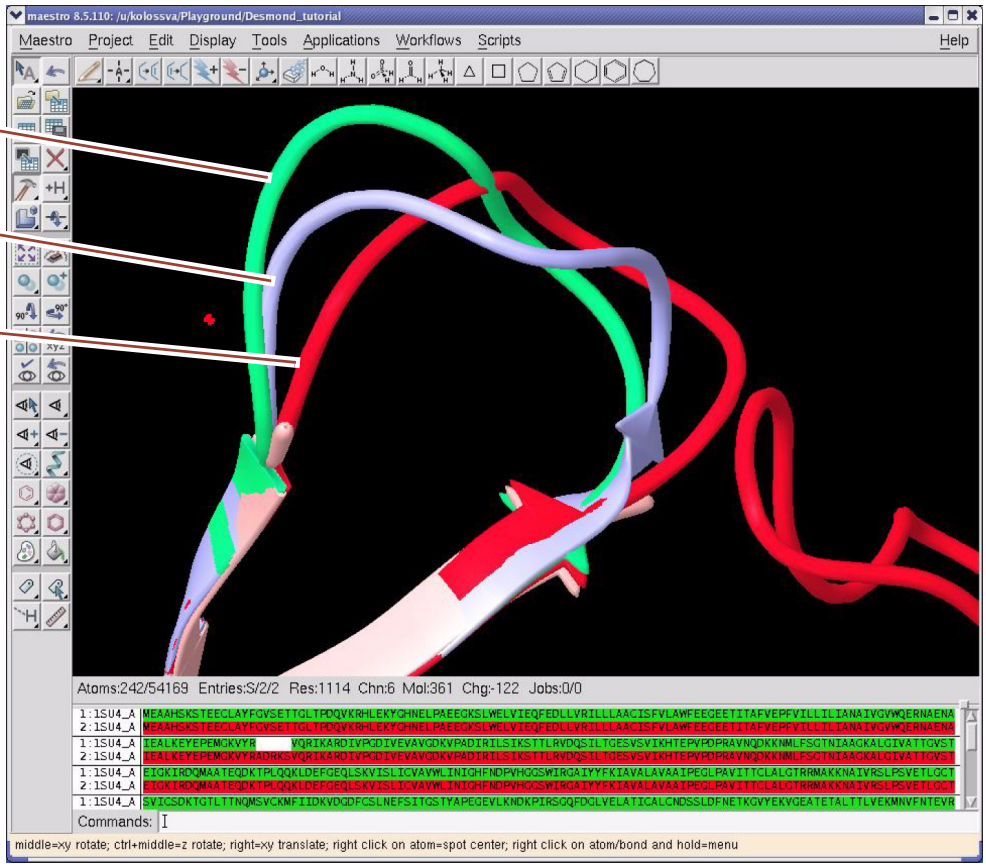
t. Close the Structure Prediction window.

Figure 3.16 Loop comparison: Prime vs. X-ray

The green loop is the output of the structure builder.

The blue loop is the result of loop refinement.

The red loop is the original loop from the X-ray structure.

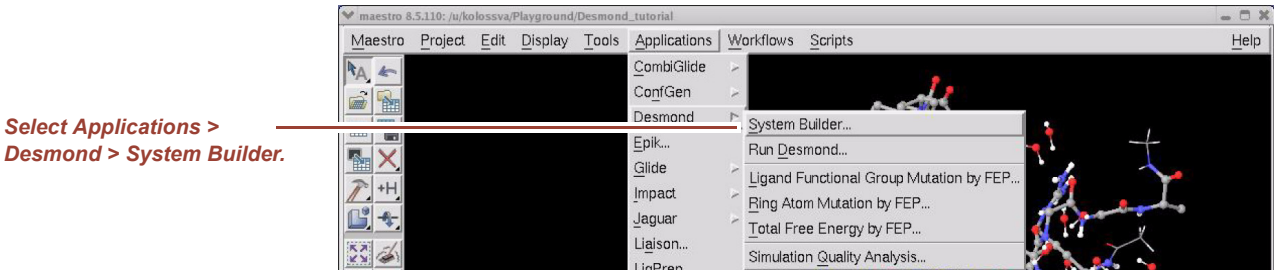


4 Preparing a Desmond simulation with the System Builder

Overview

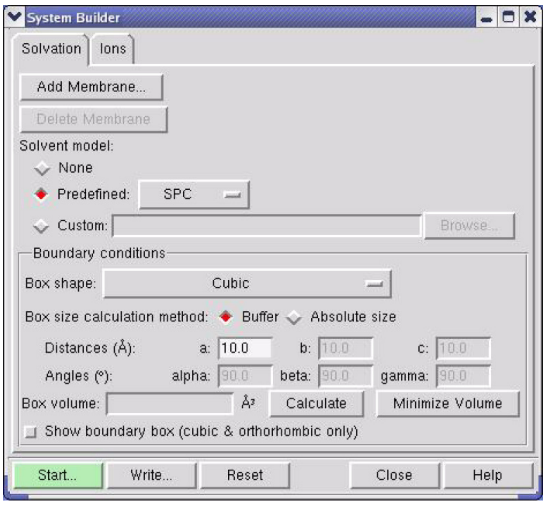
The System Builder is a graphical tool in Maestro that lets you generate a solvated system for Desmond simulations. You can launch the System Builder by selecting **Applications > Desmond > System Builder** from the menu bar as shown in Figure 4.1.

Figure 4.1 Launching Desmond System Builder



The System Builder panel appears as shown in Figure 4.2.

Figure 4.2 System Builder window



The solvated system generated by System Builder includes the solute (protein, protein complex, protein-ligand complex, or similar systems, or, a protein immersed in a membrane bilayer), solvent (currently only water), and counter ions. All structural topological information and force field parameters for the solvated system are written to a special Maestro file that is subsequently used for Desmond simulation.

Selecting Solutes and Solvents

The System Builder considers the current contents of the Workspace to constitute the solute. Note that some parts of the structure in the Workspace may not be displayed, but are still included in the solute.

Supported solvent models in the GUI include SPC, TIP3P, and TIP4P water; *Viparr* allows TIP5P as well (see "Generating Force Field Parameters with Viparr" on page 57). Custom models are also allowed if you can provide the location of a pre-equilibrated box of a different solvent molecule.

Defining the Simulation Box

When defining the simulation box, the goal is to reduce the volume of solvent while ensuring that enough solvent surrounds the solute so that the protein does not `see' a periodic image of itself during simulation. Too much solvent will unduly lengthen the computation.

One way to minimize solvent volume is to select a shape for the simulation box that is similar to the protein structure. The System Builder shown on Figure 4.2 supports all standard box shapes—cubic, orthorhombic, triclinic, truncated octahedron, and so on. Select the most appropriate shape from the Box shape option list in the Boundary conditions section of the Solvation tab and click the Calculate button if you want to compute the box volume. Besides the shape, the box size also depends on how you define the solvent buffer around the solute:

- Apply a buffer distance to each dimension of the simulation box. A typical buffer distance is 10 Å, which is equal to the usual real space Coulombic interaction cutoff for long-range electrostatic calculations.
- Set the absolute size and shape of the box.

Next, by clicking the Minimize Volume button you can instruct the System Builder to minimize the box volume by aligning the principal axes of the solute with the xyz or diagonal vectors of the simulation box—essentially, you can align the protein structure so that it fits inside its simulation box more comfortably. However, this method is only recommended if the solute is not allowed to rotate during MD simulation.

Selecting the Show boundary box option displays a helpful, translucent graphical representation of the simulation box in the Workspace.

NOTE System Builder puts the center of gravity of the solute at the center of the simulation box. As a consequence, fairly non-spherical proteins will appear to shift toward one side of the box, leaving only a thin water buffer on that side. This may not be ideal from a visual perspective, but in terms of periodic boundary conditions, this is perfectly adequate. It is not the distance between the outer surface of the protein and the near face of the simulation box that matters; rather, it is the sum of two such distances with respect to the opposite sides of the protein that is important.

System Builder Output File Format

The System Builder writes the simulation system in a composite model system file with the extension **.cms**. .cms files are essentially multi-structure Maestro files that are suitable for initiating Desmond simulations. Typically, the total solvated system is decomposed into five separate sections in the .cms file: protein (solute), counter ions, positive salt ions, negative salt ions, and water molecules. The System Builder also writes the force field parameters in the .cms file. You can find detailed documentation of the .cms file format in the *Desmond User Guide* and the *Schrödinger Desmond User Manual* listed in "Documentation Resources" on page 97.

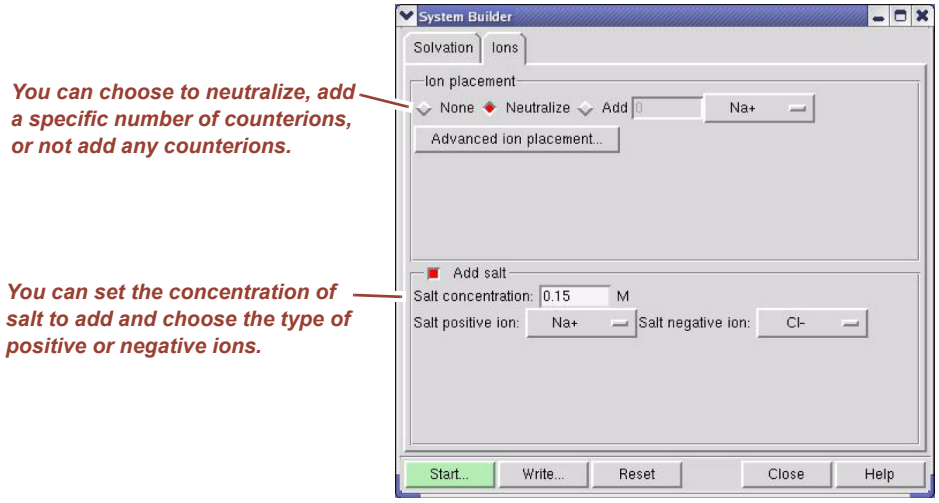
Adding Ions

Click the Ions tab in the System Builder window as shown in Figure 4.3 to add ions to your system. By default the System Builder automatically neutralizes the solute. For example, if the solute has a net charge of +N, System Builder will randomly select N residues on the surface of the protein with a positive formal charge and, respectively, place a negative counterion in the vicinity of the selected residue. For negatively charged solutes, positive counterions will be similarly positioned. The Advanced ion placement button allows you to place counterions in a more sophisticated manner, but this procedure is illustrated in "Setting Up Membrane Systems" on page 47 where membrane setup is discussed using a concrete example.

- An arbitrary number of counterions can be added; for example, if there is a reason not to neutralize the system (refer to the special note in "Setting Up Membrane Systems" on page 47). Conversely, you may choose not to place any counterions.

- Background salt can be added to the system by specifying the salt concentration in the Add salt section in the Ions tab. The salt ions will be randomly distributed in the entire volume of the simulation box excluding, of course, the volume occupied by the solute.
- The type of positive or negative ions to use can be specified in the Ions tab.

Figure 4.3 Ions tab in Desmond System Builder



Generating the Solvated System

At this point, you can generate the solvated system by clicking the green Start button in the System Builder window. After this task has completed, the solvated system will appear in the Workspace (see Figure 1.15 on page 14) and the resulting system, which is ready for Desmond simulation will be written to a .cms file (see "System Builder Output File Format" on page 45).

NOTE System Builder automatically assigns the latest OPLS-AA force field parameters available in the Schrödinger Suite to the entire system. If you would rather apply a Desmond provided force field (Amber or Charmm force fields, TIP5P water model, Schrödinger's PFF polarizable force field, etc.), you need to process the `.cms` file using the external *Viparr* program (see "Generating Force Field Parameters with Viparr" on page 57).

You can also generate the solvated system from the command line. You may decide to use this method if you want to manually edit the input file to produce an effect that cannot directly be generated by the System Builder.

To generate the solvated system from the command line:

1. Click the Write button in the System Builder window to write the job files to disk.

   There are two input files:

   — **`my_setup.mae`**. This is the Maestro structure file of the solute, which serves as the input for system setup.

— **my_setup.csb**. This is the command file, which can be hand-edited for custom setup cases. For detailed documentation of the `.csb` file see the *Schrödinger Desmond User Manual* listed in "Documentation Resources" on page 97.

And a single output file (besides a log file):

— **my_setup-out.cms**. This is the Maestro structure file of the entire simulation system including OPLS-AA force field parameters. For details on the .cms file see the *Schrödinger Desmond User Manual* listed in "Documentation Resources" on page 97.

2. Execute the following command at the command line:

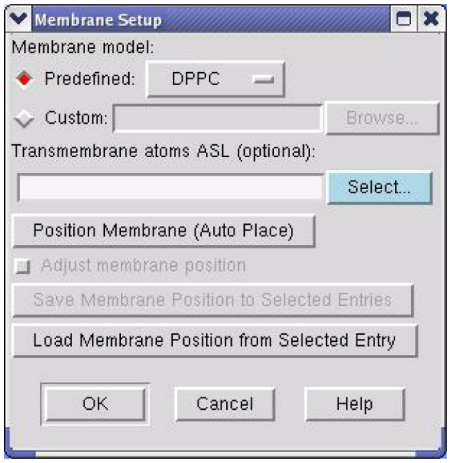
```
$ $SCHRODINGER/run $SCHRODINGER/utilities/system_builder
my_setup.csb
```

where `my_setup` is the file name given to "Write".

Setting Up Membrane Systems

The Membrane tab as shown in Figure 4.4 can be used to embed a membrane protein structure in a membrane bilayer. In this procedure a template consisting of an equilibrated membrane—including the accompanying water—is used to generate a large enough region to encompass the protein. Using the Membrane tab, you can position the protein within the membrane using a semi-automated procedure.

Figure 4.4 Membrane setup in the System Builder



As a real-life example we can consider the calcium ATPase protein, 1su4. Follow the procedure below to prepare this protein for membrane simulation in Desmond.

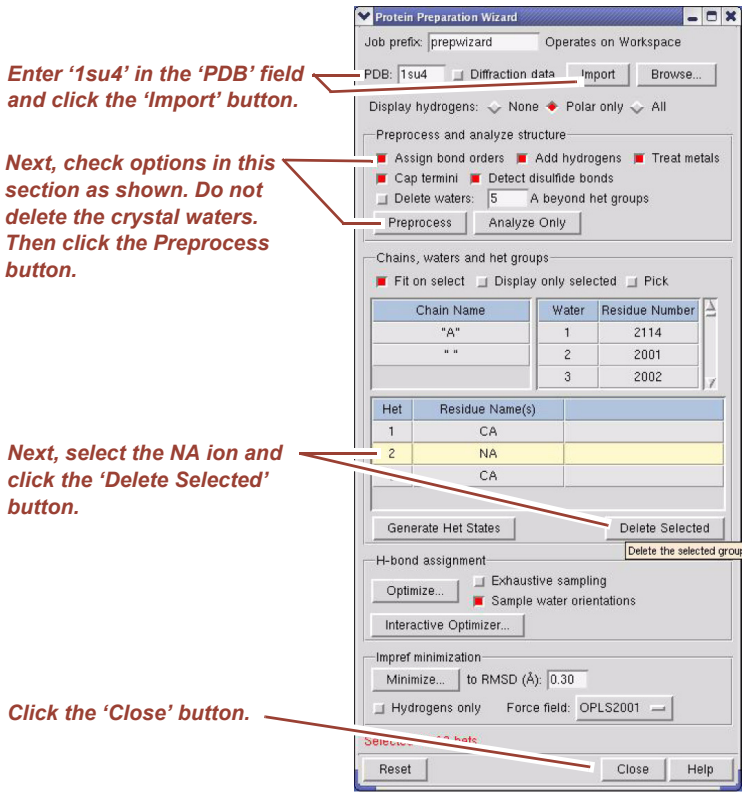
To setup the 1su4 membrane protein system for Desmond simulation:

1. Open the Protein Preparation Wizard by selecting **Workflows > Protein Preparation Wizard**. Type "**1su4**" in the PDB field and click the Import button as shown in Figure 4.5.
2. When the structure appears in the MaestroWorkspace, set the preprocessing options as shown in Figure 4.5; do not delete the crystal waters as they are integral to the pro-

tein structure. Click the Preprocess button. The tables will be filled in by the Protein Preparation Wizard.

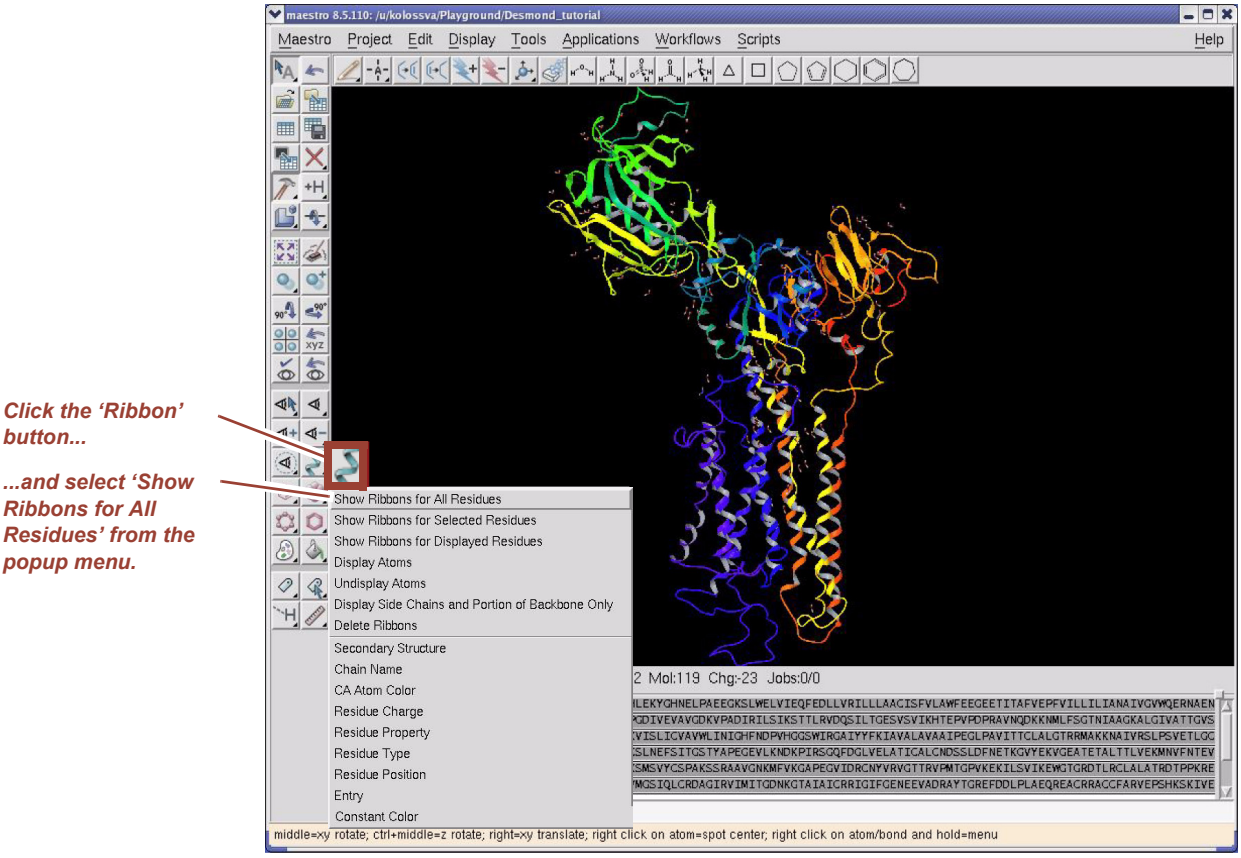
3. Select the sodium ion in the Het table (the focus of the Workspace view will center and zoom on the Na ion) and then click the Delete Selected button. The sodium ion is just an artifact of the crystallization process, only the Ca ions are integral parts of the protein.
4. You may want to experiment with H-bond assignment as described in Figure 1.10, but it is not essential for the current membrane setup exercise. Click the Close button to exit from the Protein Preparation Wizard.

Figure 4.5 Preprocessing the 1su4 structure



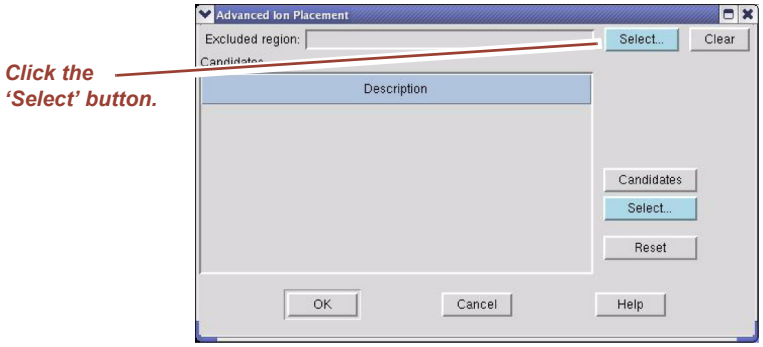
5. Change the view to ribbon view in the Workspace and orient the protein similar to what can be seen in Figure 4.6, which is the standard way of looking at membrane proteins (with the transmembrane bundle aligned along the vertical axis).

Figure 4.6 The 1su4 structure in standard orientation



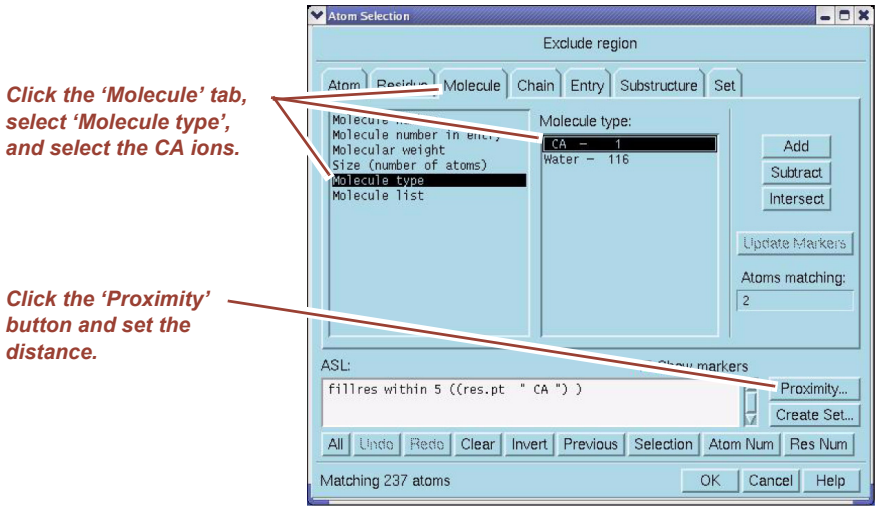
6. Launch the System Builder by selecting Applications > Desmond > System Builder. Click the Ions tab and click the Advanced ion placement button. The panel appears as shown in Figure 4.7.

Figure 4.7 The advanced ion placement window



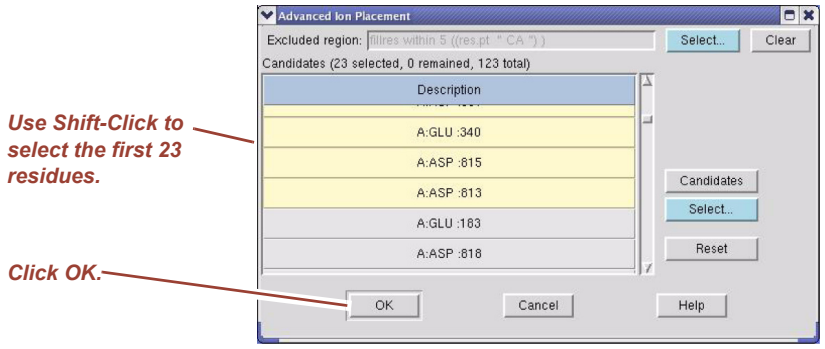
7. In this tutorial exercise we do not want to place counterions in the vicinity of the calcium ions. This can be achieved the following way. Click the blue Select button next to the Excluded region field. The selection window pops up as shown in Figure 4.8.

Figure 4.8 Selecting the excluded region



8. Select the two calcium ions by "molecule type" and use the Proximity button to select all residues within 5 Å. The resulting ASL expression at the bottom of the selection window defines the excluded region. Click the OK button.
9. Click the Candidates button in the Advanced ion placement window shown in Figure 4.7. A list of all negatively charged residues, which lie outside of the excluded region appear in the table as shown in Figure 4.9.

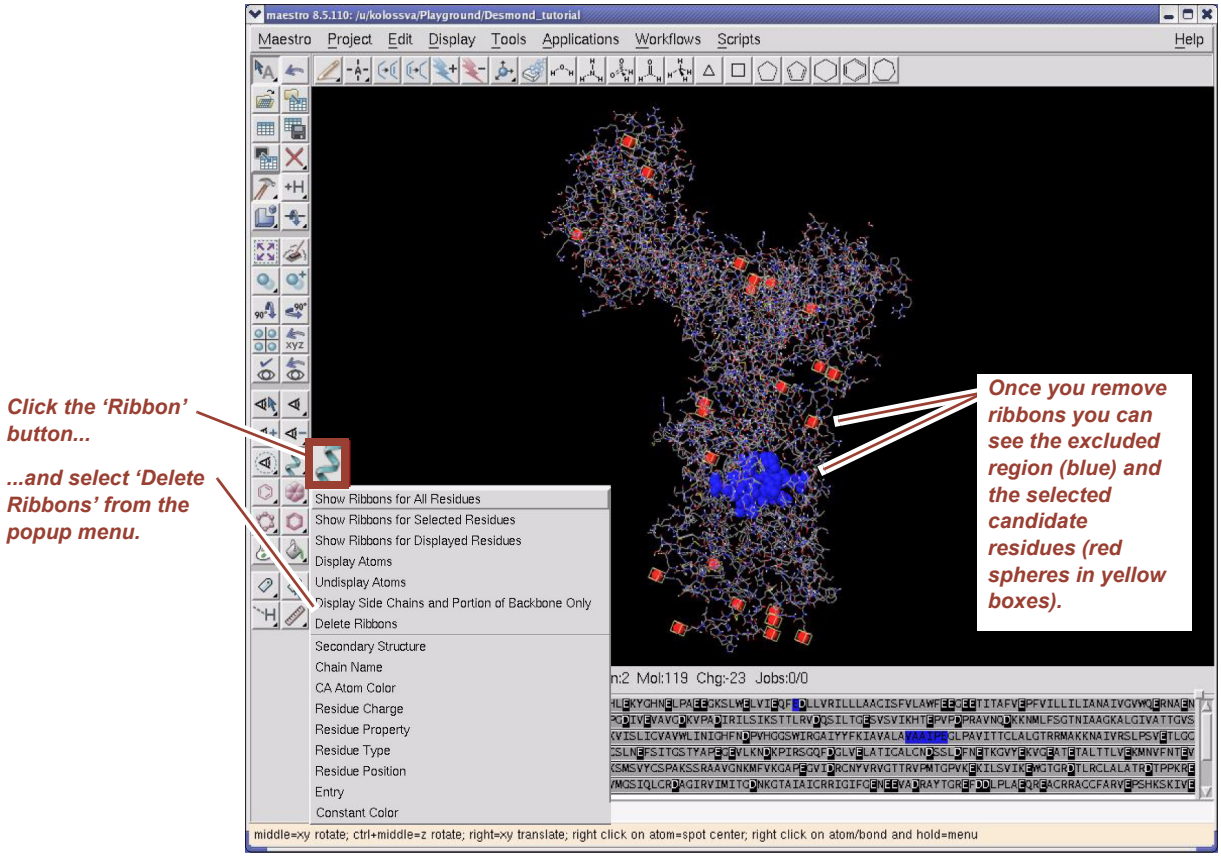
Figure 4.9 Placement of the counterions



Note that the 1su4 structure has a net total charge of -23 after being processed with the Protein Preparation Wizard. The candidate residues are listed in the table in no particular order; therefore, if you select the first 23 residues, a good spacial distribution of the positive counterions should result. Click the OK button.

10. Temporarily turn off the ribbon view so you can see the visual feedback of candidate selection in the Workspace. The 1su4 structure should appear similar to the Workspace in Figure 4.10.

Figure 4.10 Visual feedback of ion placement



The blue "blob" shows the excluded region and the selected candidate residues are highlighted by red spheres enclosed in yellow boxes (the selection markers). The positive counterions will be placed close to the red spheres, which mark the atoms within the residues that have the largest-magnitude partial charge.

In many cases, you may want to place only a few counterions at particular locations and distribute the rest of the charges in a truly random fashion. Follow the steps below to achieve this goal.

a. Set the Solvent model to None (Figure 4.2 on page 44) and instead of neutralizing, explicitly add the number of counterions to the system that you want to place at particular locations (Add option in Figure 4.3 on page 46).

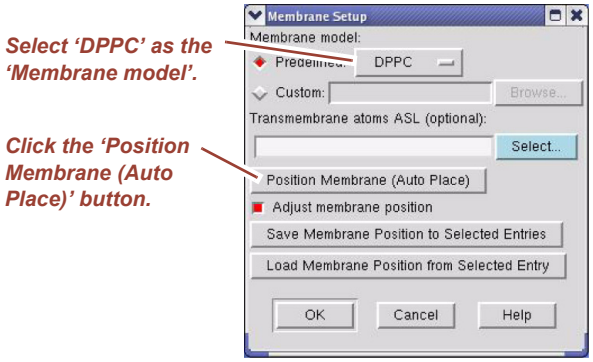
b. Apply the "advanced ion placement" procedure above to place this subset of counterions in the desired locations.

c. Run the System Builder to generate this intermediate system with no solvent and a remaining net total charge.

d. Import the resulting .cms file back to Maestro and re-run the System Builder with the appropriate solvent and the "neutralize" option, with no further application of the "advanced ion placement" procedure.

5. After setting the Solvation and Ions options, click the Add membrane button in the System Builder. The Membrane Setup window appears as shown in Figure 4.11.

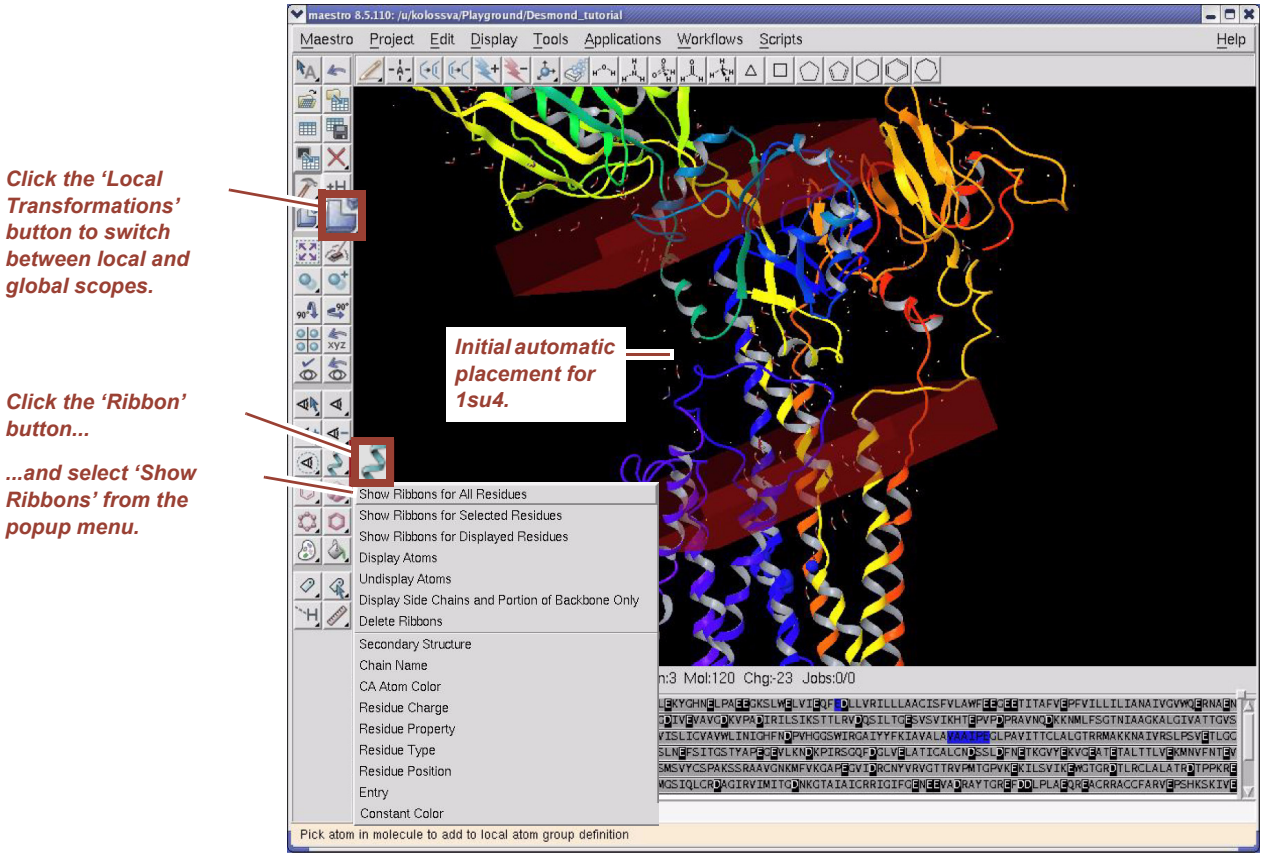
Figure 4.11 Membrane setup window



6. Set the membrane model to DPPC and click the Position Membrane (Auto Place) button. Other membrane models include POPE and POPC. The System Builder will attempt to position the membrane bilayer at a reasonable location. By specifying the set of transmembrane atoms, you can expect a highly improved auto placement. However, leave the Transmembrane atoms ASL field empty for this tutorial example. The resulting, automatic membrane placement is shown in Figure 4.12. As you can see, the membrane positioning needs adjustment.

NOTE Switch back to ribbon view for the remainder of this membrane setup exercise.

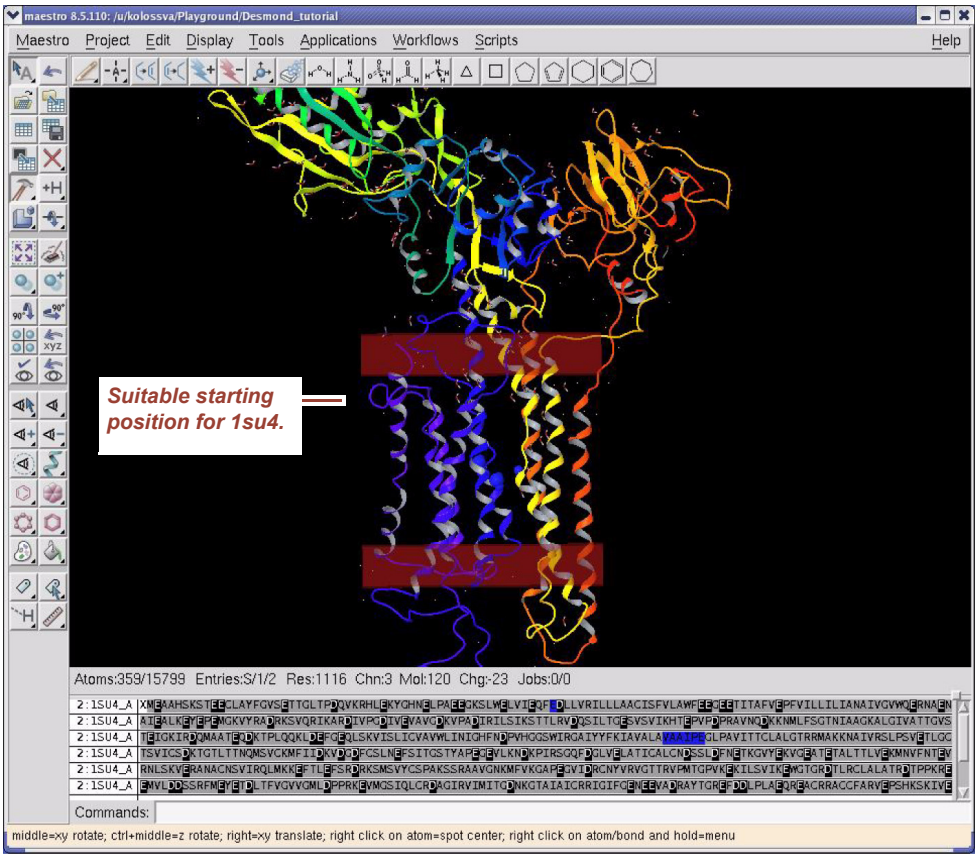
Figure 4.12 Initial automatic membrane placement for 1su4



7. Check the Adjust membrane position option in the Membrane Setup window as shown in Figure 4.11. At this point you should be able to translate and rotate the membrane model in the Workspace, relative to the protein structure. Note that the membrane model as shown in Figure 4.12 gives a schematic representation of the lipid bilayer headgroups. Moving the membrane relative to the protein structure is termed the local scope for transformations.

   However, for a better view you will probably want to translate/rotate the system as a whole as well. This is called the global scope and you can switch back and forth between the local and the global scopes by repeatedly clicking the Local transformations icon in the left icon menu (the button is highlighted in Figure 4.12). With only a few translations and rotations, switching between the local and global scopes, you should be able to place the membrane in a suitable starting position. After applying manual transformations you should place the membrane similar to that shown in Figure 4.13.

Figure 4.13 Adjusted position of the membrane for 1su4



NOTE You may want to position two or more different membrane proteins (e.g., for mutagenesis studies) in a bilayer in exactly the same way. Of course, this would be virtually impossible to do, trying to reproduce the same manual adjustments. The Membrane Setup in the System Builder provides a convenient way to automate this procedure. Any time you are satisfied with the membrane placement, the net translation/rotation transformation matrix can be saved in the project entry by simply clicking the Save Membrane Position to Selected Entries button in the Membrane Setup window (see Figure 4.11 on page 52). Then, for any subsequent protein, you can simply click the Load Membrane Position from Selected Entry

button to position the entry according to the saved transformation matrix. This way, all proteins will be placed in the membrane in exactly the same position and orientation.

8. Finally, click the green Start button to start the membrane system setup. For 1su4, the setup procedure should take 6-8 minutes. The resulting simulation system is shown in Figure 4.14 on page 55. For better visualization, the protein is shown as a CPK model and the lipid bilayer is highlighted in green.

NOTE The visible gap at the top and bottom of the lipid layer is the gap between the water layer that is part of the DPPC model and the extra water layer, which was added to the system to satisfy the boundary conditions. This is expected since the system is put together from two different, pre-equilibrated boxes of a solvated lipid bilayer and plain solvent, respectively. The gap should go away after a relatively short equilibration period. However, membrane systems should be equilibrated for an extended period of time (up to a few hundred ns) to resolve the large hole that was carved out of the lipid bilayer to accommodate the transmembrane part of the protein. The hole is shown for the 1su4 structure in Figure 4.15.

NOTE The protein in the simulation box in Figure 4.14 is positioned so that the head of the protein seems to protrude into vacuum. This is the result of a visual artifact discussed in "Defining the Simulation Box" on page 44. System Builder puts the center of gravity of the solute at the center of the simulation box. As a consequence, fairly non-spherical systems will appear to shift toward one side of the simulation box. Membrane systems are extreme cases resulting in the view on Figure 4.14. This may not be ideal from a visual perspective, but in terms of periodic boundary conditions, this is perfectly adequate. It is a periodic image of the protruding part of the protein at the bottom of the simulation box where there is plenty of water, that is used in the computation (see Figure 4.14).

Figure 4.14 Final simulation system for 1su4

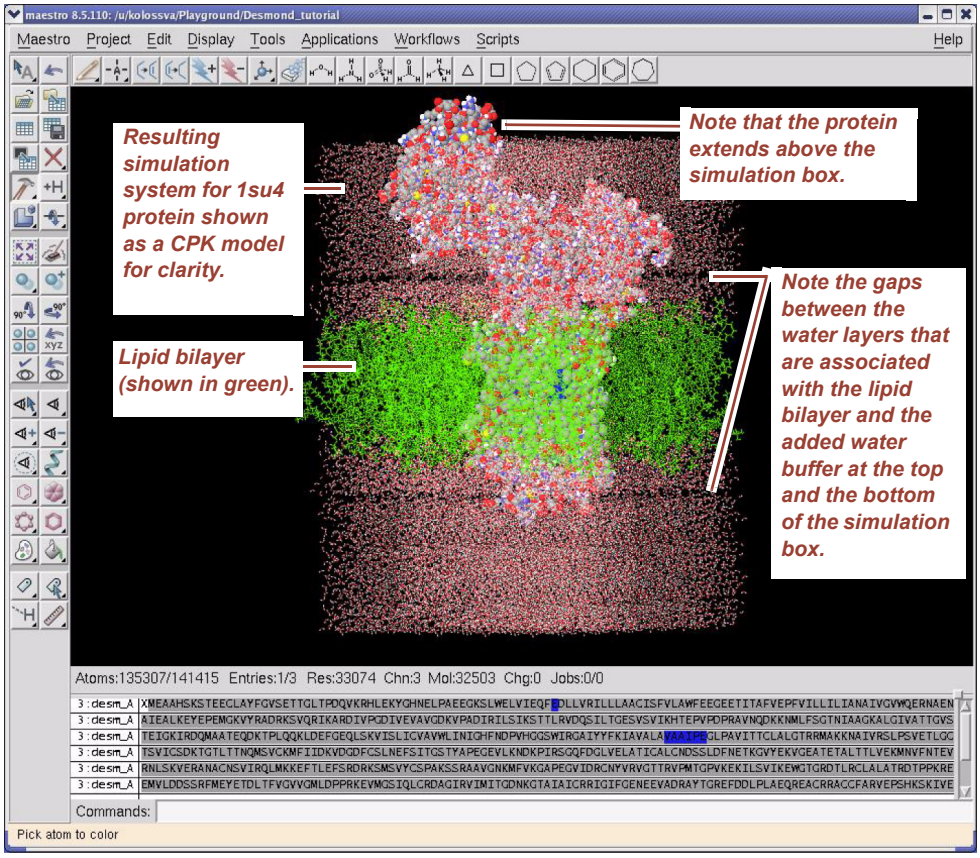
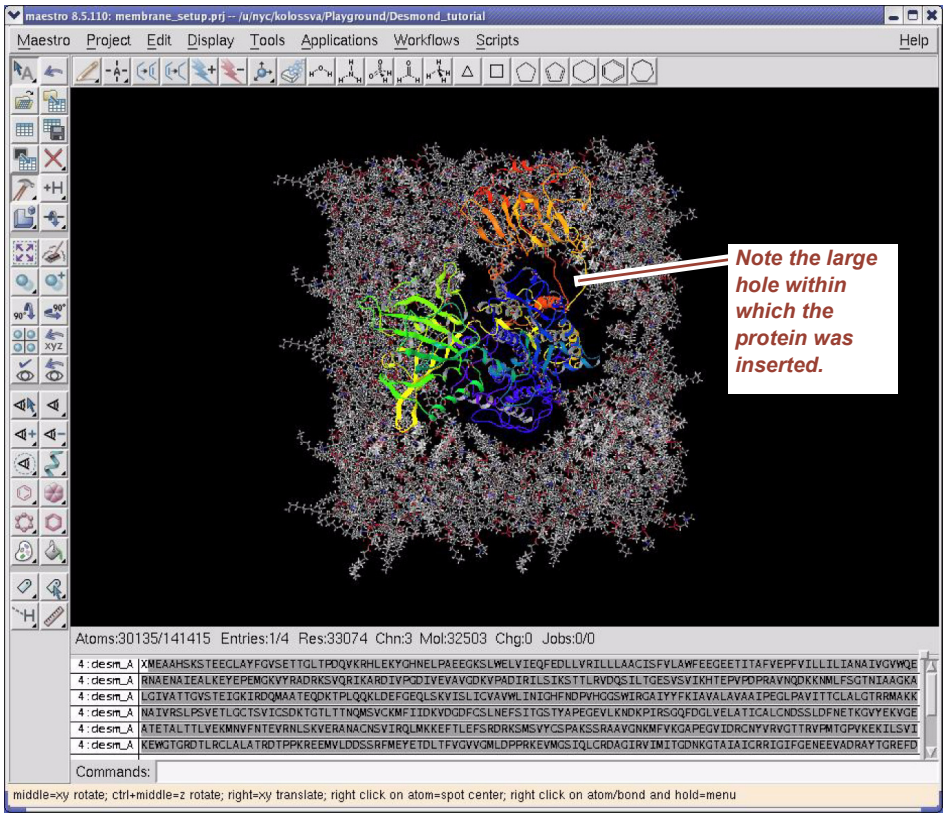


Figure 4.15 Transmembrane hole in the DPPC bilayer for 1su4



9. More experienced users may want to run the System Builder job from the command line. Instead of clicking the green Start button, click the Write button to write a command script file to disk that can be executed from the command line.

   a. Click the Write button in the System Builder window to write the job files to disk.

      There are two input files:

   — `my_setup.mae`. This is the Maestro structure file of the solute (1su4 protein, calcium ions, crystal water molecules, neutralizing ions, and counterions), which serves as the input for system setup.

   — `my_setup.csb`. This is the command file, which can be hand-edited for custom setup cases. For detailed documentation of the .csb file see the *Schrödinger Desmond User Manual* listed in "Documentation Resources" on page 97.

      And a single output file (besides a log file):

   — `my_setup-out.cms`. This is the Maestro structure file of the entire membrane simulation system including OPLS-AA force field parameters. For detailed documentation of the .cms file see the *Schrödinger Desmond User Manual* listed in "Documentation Resources" on page 97.

   b. Execute the following command at the command line:

```
$ $SCHRODINGER/run $SCHRODINGER/utilities/system_builder
my_setup.csb
```

      where `my_setup` is the file name given to "Write".

5 Finishing Preparations for Desmond Simulation

Overview

There are a few more tasks that must take place before a simulation can be run:

- The System Builder will always generate the latest OPLS-AA force field parameters for the simulation system it builds and the parameters will be included in the .cms output file. However, you may want to assign different force field parameters in case the Schrödinger provided OPLS-AA parameters are not adequate for a particular system. You can assign alternative force field parameters for your system using a companion program to Desmond called *Viparr*.
- You will also have to specify Desmond run-time simulation parameters.

These topics are addressed in the following sections.

Generating Force Field Parameters with Viparr

Viparr is a Python script that reads a `.cms` (or `.mae`) file and writes another `.cms` file, which has all the force field parameters required to run a Desmond simulation. *Viparr* can assign force field parameters from a variety of well-established force fields and water models. *Viparr* force field assignment is template based. This means that unlike force field servers, e.g., Schrödinger's OPLS-AA server that can generate parameters for various atom types derived automatically from the molecule's topological structure, *Viparr* is designed to find templates (such as an entire residue) in `.cms` files and assign the appropriate force field parameters by looking at a template database associated with a particular force field. The resulting parameters are guaranteed to be identical with those of the published, authentic force field.

The *Viparr* script is called `viparr.py` and basic help on its usage can be obtained via the command line (output for version 1.3.0):

The command line invocation of the *Viparr* script is

```
$SCHRODINGER/run $SCHRODINGER/desmond-v.../bin/Linux-x86/viparr.py
```

but it is referred to as `viparr.py` below.

```
$ viparr.py --help
usage:
     viparr.py [options] mae_file maeff_outfile

Description:
     Viparr is a force field parameter assignment program.
     -f and -d are used to specify force fields; the order of force fields
     is important: earlier ones take precedence over later ones.

Simple example:
     viparr.py -f charmm27 -f spc mae_file maeff_outfile

More complicated example:
     viparr.py -d me/myfiles/myff -f charmm27 -f spc mae_file
     maeff_outfile

Available built-in force fields:
     amber03
     amber94
     amber96
     amber99
     amber99SB
     charmm22
     charmm27
     oplsaa_impact_2001
     oplsaa_impact_2005
     oplsaa_ions_Jensen_2006
     pff
     spc
     spce
     tip3p
     tip3p_charmm
     tip4p
     tip4pew
     tip5p
options:
     --version     show program's version number and exit
     -h, --help    show this help message and exit
     -cCTNUM       run Viparr for one CT block only, e.g., for the first,
                   use "-c 1"
     -fFFNAME      built-in force field name; several can be listed, each
                   preceded by its own -f
     -dFORCEFIELD user-provided force field directory; several can be
                   listed, each preceded by its own -d
     -mMERGE_DIR   path to user-defined force field directory that is to
be
                   merged with previously specified force field; several
```

```
                can be listed, each preceded by its own -m
  -nFFIO_NAME   force field name to put into output file
  -pPLUGINDIR   override for plugins directory (advanced usage)
  -x            do not print header info (for testing purposes)
```

The built-in force fields include the latest versions of Amber, CHARMM, and OPLS-AA families of fixed-charge force fields and *pff*, which is Schrödinger's polarizable force field. *Viparr* also provides the parameters for all SPC and TIP families of solvation models. A simulation system, or *chemical system*, is described using a number of structures called *connection tables* that reside in *CT blocks* in the `.cms` file. These CT blocks are designated by the `f_m_ct{}` structure and are composed of a number of chains, which in turn are composed of a number of residues; for our purposes, residues may be something other than amino acids (for example, water molecules or ions). *Viparr* matches residues in the chemical system to templates in the force fields.

Viparr uses atomic numbers and bond structure to match residues to templates. Thus if there are non-standard atom or residue PDB names in the `.cms` file, there is no need to modify them to match the names used in the force field. You can freely modify atom and residue names for historic, idiosyncratic, or any other purposes. In particular, *Viparr* will identify the N- and C-terminal versions of the residues correctly, as well as protonated/deprotonated versions of a residue—even if they are not identified as such in the `.cms` file.

Viparr uses the atom ordering in the `.cms` file and does not alter this ordering when creating the output file. Residue numbering is also left unchanged, and can begin with any integer (including negative integers); numbering does not even need to be contiguous, which is typical to many PDB files. Residues with different chain names can have the same residue number. To aid in diagnosing problems with the input `.cms` file, messages involving residues have the form: `< "chain_name", residue_number > (residue_name)` and is usually preceded by a structure number (CT number).

Viparr outputs a compressed force field representation when all the residues in a connection table are the same. For a connection table that only contains water molecules, this means that force field parameters are output only for a single water molecule.

For the 1su4 example used in "Preparing a Desmond simulation with the System Builder" on page 43, invocation of *Viparr* is very simple:

```
$ viparr.py -f charmm27 -f spc 1su4_desmond.cms 1su4_desmond_out.cms
```

where `1su4_desmond.cms` is the output file from the Desmond Panel. The CHARMM-27 force field was used, because it has built-in parameters for calcium. The resulting `1su4_desmond_out.cms` structure file along with the `1su4_desmond.cfg` conguration file (also from the Desmond Panel) can now be used to run Desmond simulations.

This simple example already shows a very useful feature of *Viparr*; it can be used to specify multiple force fields. In this case, each residue in the chemical system matches a template in exactly one of the specified force fields. This means that the protein residues and the two calcium atoms will be assigned CHARMM-27 parameters whereas the water molecules get SPC parameters. The command line invocation for assigning the same protein parameters, but a different water model of TIP4P, appears similar to this:

```
$ viparr.py -f charmm27 -f tip4p 1su4_desmond.cms 1su4_desmond_out.cms
```

You can also use multiple force fields to combine components of two or more force fields. In this case, residues in the chemical system match templates in more than one of the specified force fields. In this operation mode, all matching force fields are applied. For example, one force field provides the angle parameters for a set of residues, while another force

field provides the dihedral parameters for the same residues. The force field components should be disjoint so there is no conflict with the parameters that are assigned to each component.

You may also use multiple force fields when parameters assigned to a particular residue by one force field override parameters assigned by another force field. This situation is similar to combining force fields; residues in the chemical system match templates in more than one of the specified force fields and all matching force fields are applied. However, in this case two or more force fields provide parameters for the same term; for example, two force fields provide parameters for the angle between atoms 1, 2, and 3 causing a conflict. The conflict is resolved by allowing the first force field (that matches the residue) to take precedence. The order is the order in which the force fields were specified on the command line.

In all multiple force field assignment cases, *Viparr* prints an error message and aborts if a bond exists between two residues that are not matched by the same force field.

The selected force fields should also have consistent van der Waals combining rules. Otherwise, *Viparr* will abort with an error message.

Viparr will print the following warning and error messages when matching residues to templates:

- If any residue matches more than one template in a force field, then *Viparr* exits with an error. (No *Viparr* force field should contain identical templates.)
- If any residue name is matched to a force field template with a different name, a message is printed. A maximum of 5 messages are printed per residue-template name pair.
- *Viparr* will check that all residues are matched by one of the selected force fields. If there are any unmatched residues, *Viparr* will print all unmatched residues and exit with an error. A maximum of 5 messages are printed per unmatched residue name.
- If any residue is matched by more than one of the selected force fields, *Viparr* will print a warning message. The user should be cautious and be responsible for any intended use of multiple force fields with multiple matches.

Besides the `-f` option, there are two more basic options. The `-c` option allows the user to assign parameters for a particular structure (CT block) in the input `.cms` file and the `-n` option provides a means to include a user-defined comment in the output `.cms` file for annotation purposes.

Viparr also supports user-defined, special force fields (the `-d`, `-m`, and `-p` options), but that is beyond the scope of this tutorial. Please refer to the *Desmond User Guide* for information.

Specifying Desmond Simulation Parameters

Before it can perform simulations, Desmond needs certain simulation parameters to be set. These parameters can be set in two ways:

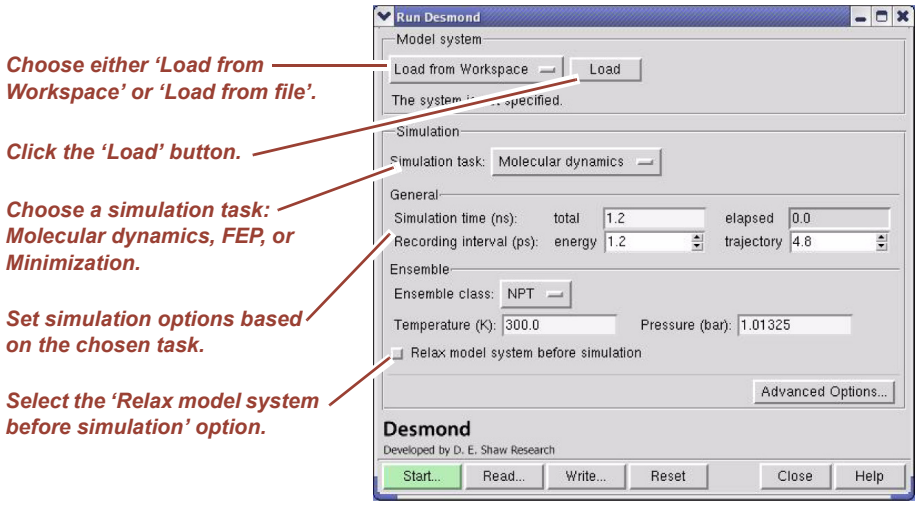
- Using the Run Desmond facility in Maestro.
- Editing the Desmond conguration file directly.

The following sections provide more details on each option.

Using the Run Desmond facility in Maestro

To set parameters from Maestro, select **Applications > Desmond > Run Desmond**. The Desmond window appears as shown in Figure 5.1.

Figure 5.1 Setting up a Desmond simulation



Import the model system into the Desmond window by clicking the Load button: select either Load from the Workspace or Import from file (and select a `.cms` file), and then click Load. The import process may take several minutes for large systems.

The basic settings shown in Figure 5.1 include:

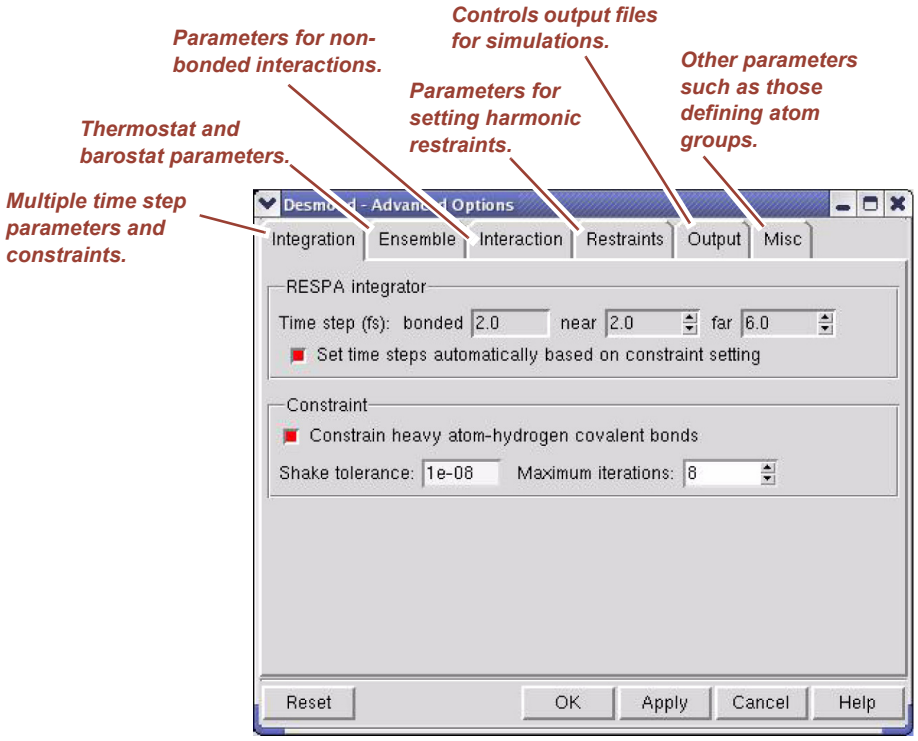
- **Simulation task.** Choose between MD, FEP, or Minimization.
- **Task-specific options.** For the MD task, these include simulation time, simulation intervals (time intervals by which different energy terms are recorded and trajectory snapshots are saved), and ensemble class (NVE, NVT, NPT).
- **Model relaxation.** Select the Relax the model system before simulation option.

When you click Advanced Options, the Advanced Options window appears as shown in Figure 5.2.

The options that appear on the Advanced Options window depend on the task you selected in the main window. Options in the Integration tab for the MD task include the multiple time step (RESPA) schedule and constraints.

Some parameters are interdependent; for example, spinboxes for setting the time step schedule. If you set the innermost time step and hit **Return** the midrange and longrange time steps are adjusted automatically. Of course, you can always manually edit the values or use the spinbox up/down arrows to adjust the values.

Figure 5.2 Advanced Options for simulation jobs



The Advanced Options window has several tabs:

- Ensemble—Contains thermostat and barostat parameters.
- Interaction—Contains parameters for computing non-bonded interactions.
- Restraint—Provides a table in which multiple sets of atoms can be subjected to harmonic restraint with a user-defined force constant. Atom selection can be directly input as an ASL expression or you can click the Select button to bring up the Selection Tool.
- Output—Includes the names of various Desmond output files, and allows you to specify how often these files are updated during the simulation. For example, different energy terms are recorded in the energy sequence file in user-defined intervals. Snapshots of the simulation are periodically saved in the trajectory file and the entire simulation state is also saved at user-defined intervals in a checkpoint file. Desmond simulations can be restarted from checkpoint files with bitwise accuracy.
- Misc—Provides control over miscellaneous parameters, including the definition of atom groups. Currently, two different atom groups are recognized by Desmond. A `frozen' atom group means a set of atoms that move during the entire MD simulation as a rigid body. (**NOTE:** in Desmond, unfrozen atoms cannot be connected to frozen atoms). For example, an entire protein molecule can be frozen (during equilibration), but one cannot freeze only the heavy atoms, or just certain residues. In that case, restraints should be used. Also note that frozen atoms cannot be used with NPT simulations, because the Virial will be incorrect. The other atom group termed `cm_moi` is used to define parts of the system whose center of mass motion is set to zero. In most protein simulations the protein center of mass motion is set to zero to prevent the protein from translating and rotating in the simulation box. Atom

groups can be selected the same way as restraint groups, using ASL syntax or the Selection Tool.

The Desmond window is filled in automatically with default parameter values. To configure your own default values so they are automatically presented in the Desmond window, set the desired values in the Desmond window and save it by clicking the Write button. Then, copy the resulting `.cfg` file to **~/.schrodinger/desmond10/desmond_default.cfg**. The new default values take effect the next time you open the Desmond window.

Detailed documentation of the Run Desmond facility can be found in the *Schrödinger Desmond User Manual* listed in "Documentation Resources" on page 97.

Once you have set values for relevant parameters, you are ready to execute a simulation.

Editing the Desmond Conguration File Directly

The Desmond conguration file syntax is described in detail in the *Desmond User Guide* (see "Documentation Resources" on page 97). You can edit the Desmond conguration file directly. It can be useful to edit existing configuration files that were used in other projects.

6 Running Desmond Simulations

Overview

You can run a Desmond simulation from the Desmond window or the command line. Before running a simulation you should check the Relax the model system before simulation option in the Desmond window. As discussed earlier, Desmond should perform structure relaxation to prepare a molecular system for production-quality MD simulation. Maestro's built-in relaxation protocol includes two stages of minimization (restrained and unrestrained) followed by four stages of MD runs with gradually diminishing restraints.

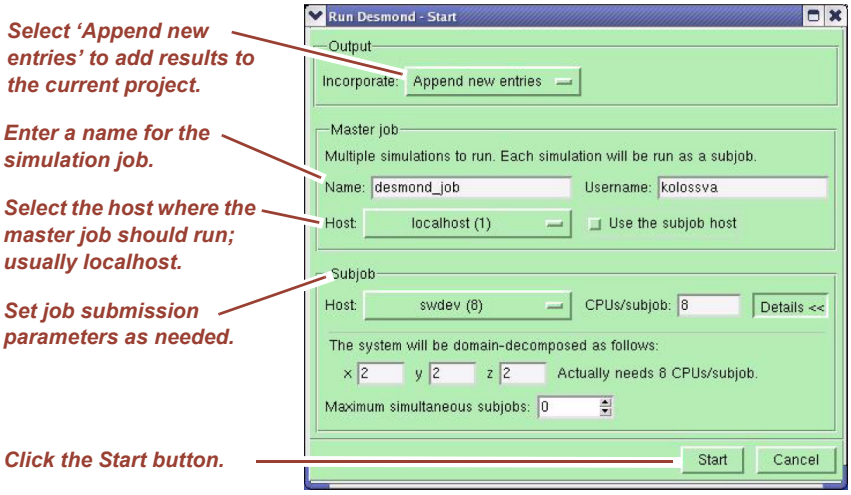
The relaxation parameters may also require edits. The relaxation protocol is written to a `.cmj` command file (Chorus multi-job file) which can be run from the command line (see "Running MultiSim jobs from the Command Line" on page 67). You can manually adjust the relaxation parameters by hand-editing the `.cmj` file.

NOTE If you decide to run the simulation from the Desmond window, you will be limited to use the built-in OPLS-AA force field for relaxation. Ideally, Desmond simulations should be run using the same force field for relaxation that is used for equilibration and simulation. Currently, if you run the simulation using *Viparr* you will need to run it from the command line.

Running Simulations from the Desmond Window

1. Click the Start button at the bottom of the Desmond window to launch Desmond simulation jobs directly from the Maestro interface. The Desmond-Start window appears as shown in Figure 6.1.

Figure 6.1 Running a Desmond simulation



2. Select Append new entries from the Incorporate field in the Output area to indicate that results of the Desmond simulation should be added to the current Maestro project (see "Maestro Projects" on page 24).
3. In the Master job area you can give the job a name and specify the username for job submission. Normally, the master job should run on *localhost*, which is your workstation where you launched Maestro. Use the options in the Subjob area to control job submission parameters including the host on which the simulation subjobs will run and the domain decomposition (the number of blocks into which the simulation box will be split in the X, Y, and Z directions) that is compatible with the given number of CPUs. On Figure 6.1 a cluster queue called **swdev** is shown, which allows a maximum of 8 CPUs to be used by any one subjob and the domain-decomposition is set to 2x2x2. Note that for certain types of simulations, e.g., FEP simulations, multiple subjobs can run simultaneously in which case the user may want to limit the number of simultaneous subjobs submitted to a queue. This can be set in the Maximum simultaneous subjobs field; zero means no limit and any nonzero value limits the number of simultaneously submitted jobs to that value.
4. Click the Start button. The Desmond simulation process begins.

Running Simulations from the Command Line

Given a valid `.cms` structure file and corresponding `.cfg` conguration file, the command line syntax for running Desmond jobs is as follows.

```
$SCHRODINGER/desmond -HOST <hostname> -exec <desmond_task> -P <cpu>
-c <jobname>.cfg -in <jobname>.cms
```

In case a job has to be restarted from a checkpoint file, replace the `-c` and `-in` options with `-restore` and the name of the checkpoint file:

```
$SCHRODINGER/desmond -HOST <hostname> -exec <desmond_task> -P <cpu>
-restore <jobname>.cpt
```

where:

- <desmond task> can be either minimize to run a simple minimization or mdsim to run an MD simulation (including FEP jobs described in "Preparing Free Energy Perturbation" on page 69), or vrun to run the vrun trajectory analysis tool.
- <jobname> is the name of the job.
- <hostname> is the name of the compute host, which can be the name of a queue on a cluster.
- <cpu> species the number of CPUs you want each Desmond simulation subjob to use. This should be 1 for running the serial version of Desmond (for example, to quickly test on your workstation whether your job will start at all), or a power of 2.

Detailed documentation of several more options for running Desmond on the command line can be found in the *Schrödinger Desmond User Manual* listed in "Documentation Resources" on page 97.

Running MultiSim jobs from the Command Line

It is often necessary to run multiple jobs, either sequentially or simultaneously, to complete a particular computational task. The two most common cases are system relaxation and FEP simulations. System relaxation was discussed on Figure 1.16 on page 15 and FEP simulations are introduced in "Preparing Free Energy Perturbation" on page 69.

Multiple jobs are handled by the MultiSim facility. MultiSim reads a .msj command script file and runs multiple simulations defined within it. For instance, to apply a custom relaxation protocol the following command can be used:

```
$SCHRODINGER/utilities/multisim -JOBNAME <jobname> -maxjob <maxjob> -cpu
<cpu> -HOST localhost -host <hostname> -i <jobname>.cms -m <jobname>.msj
-o <jobname>-out.cms
```

where:

- <jobname> is the name of the job.
- <maxjob> is the maximum number of subjobs that can be submitted to a host or a queue simultaneously. A value of zero means that there is no limit.
- <cpu> is the number of CPUs per subjob, which can also be specified as, e.g., "2 2 2" (note the quotes) indicating 8 CPUs with a spatial decomposition of 2x2x2.
- -HOST localhost means that the master job runs on your local workstation.
- <hostname> given with the -host option is the name of the host or queue where the simulation jobs will be running.

The input files include a .cms file, which is the output file of the System Builder, and the

.msj command script file, which defines the custom relaxation protocol. The custom-relaxed system will be saved in the <jobname>-out.cms file.

The .msj syntax as well as a complete list of MultiSim options can be found in the *Schrödinger Desmond User Manual* listed in "Documentation Resources" on page 97.

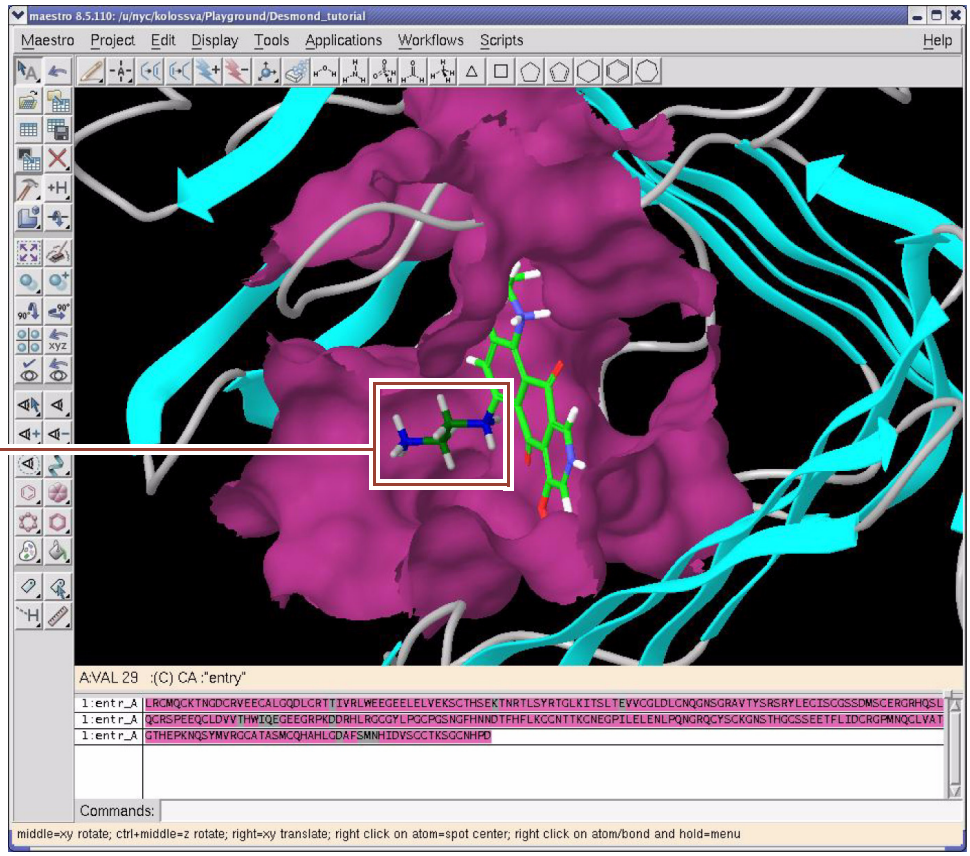
7 Preparing Free Energy Perturbation

Overview

You can use Desmond to run free energy perturbation (FEP) calculations using the dual topology approach to perform ligand mutation, annihilation, and single-atom mutation in a ring. From Maestro, you can visually specify the fixed and variable parts of a molecule that are subjected to FEP calculations and then store the atom correspondence map (which defines dual topologies) in the Maestro `.mae` file.

Ligand mutation—the most common FEP calculation—is illustrated in Figure 7.1 using a typical scenario. The figure shows a graphical representation of the soluble form of the human urokinase plasminogen activator receptor (sUPAR, PDB code 2fd6) and the ZINC-01538934 ligand. The protein structure has been processed using the Protein Preparation Wizard and Prime as described in "Tutorial Steps" on page 3 and "Running Prime" on page 29, respectively, and the ligand was docked using Schrödinger's Glide application as part of a virtual screening experiment on the ZINC database (http://zinc.docking.org/). The ligand is rendered in a tube representation in the sUPAR active site with green carbon atoms and ball-and-stick representation.

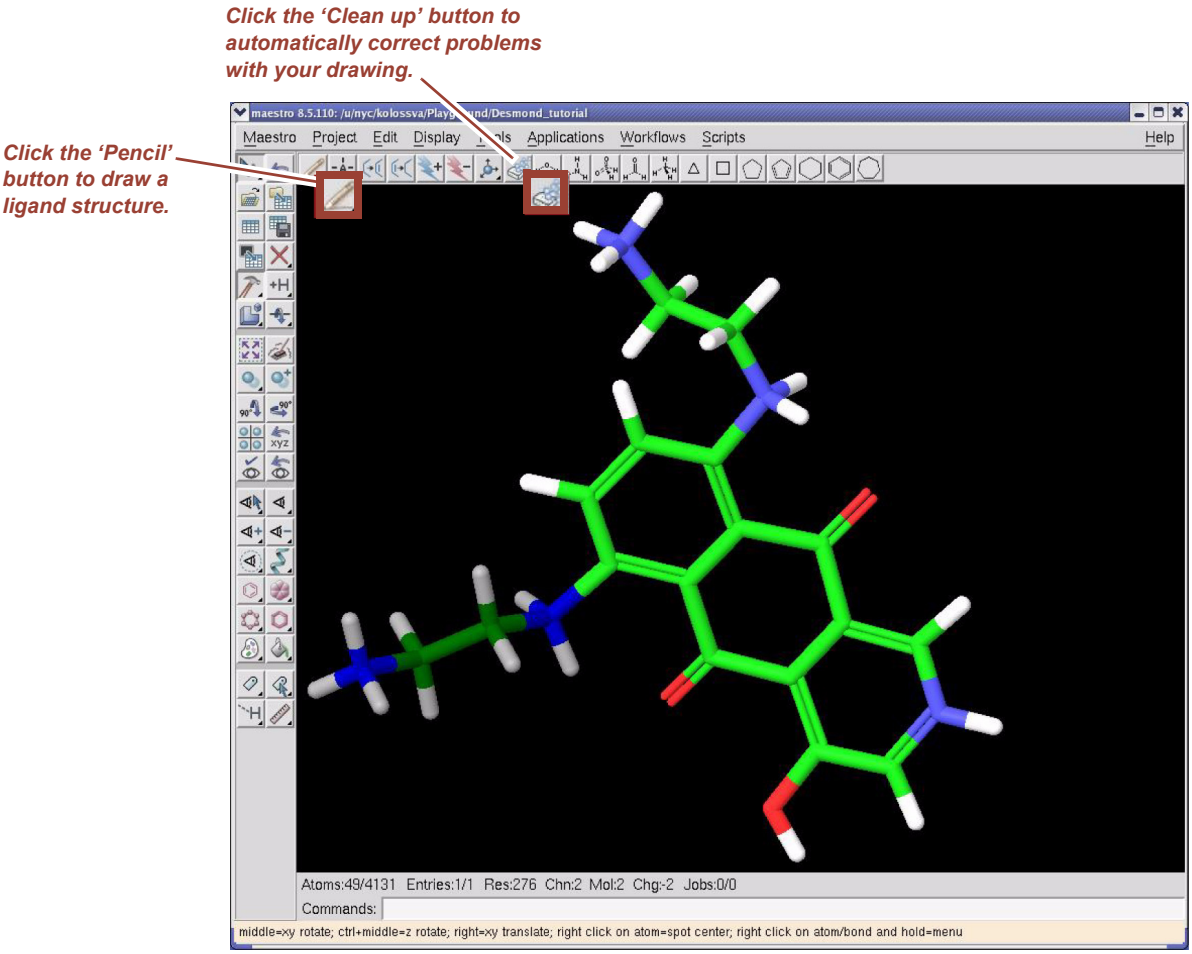
Figure 7.1 FEP Example—Ligand mutation



As shown in Figure 7.1, contact residues in the active site are displayed with a matching molecular surface to emphasize the 3D layout of the active site. One of the two $-N^+H_2-CH_2-N^+H_2-CH_3$ side chains is colored darker (see at the center of the Workspace in Figure 7.1); this is the side chain that will be mutated to a `butyl` group by FEP.

Note that the setup in Figure 7.1 is only an illustration showing a common real-life example for FEP calculations, however, you will not need to perform any docking calculations for this tutorial exercise. Instead, just build the ligand structure using the hand-drawing and geometry-cleanup options as shown in Figure 7.2.

Figure 7.2 The ZINC-01538934 ligand structure



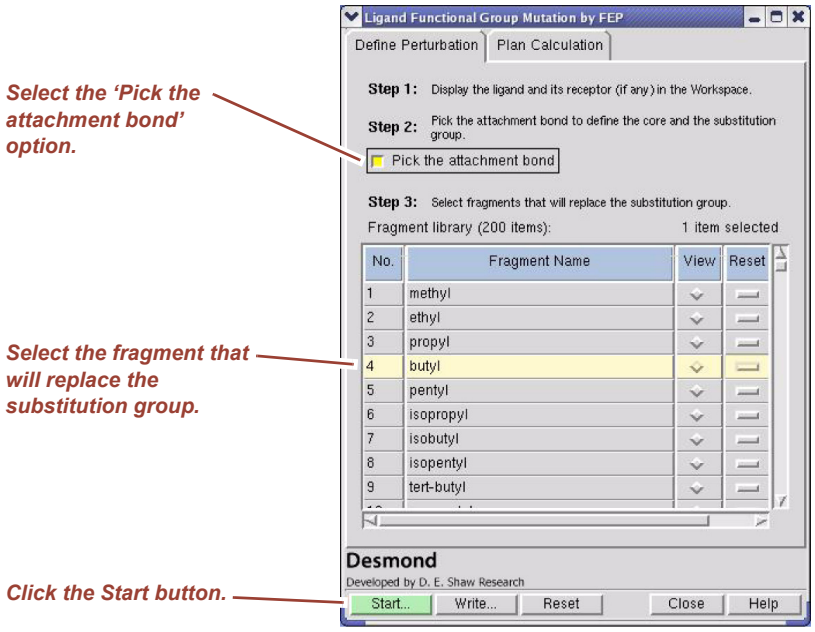
Setting Up an FEP Calculation

To setup a Free Energy Perturbation calculation for ligand mutation:

1. Select **Applications > Desmond > Ligand Functional Group Mutation by FEP**.

   The FEP Setup panel appears as shown in Figure 7.3.

Figure 7.3 FEP Setup panel for ligand mutation

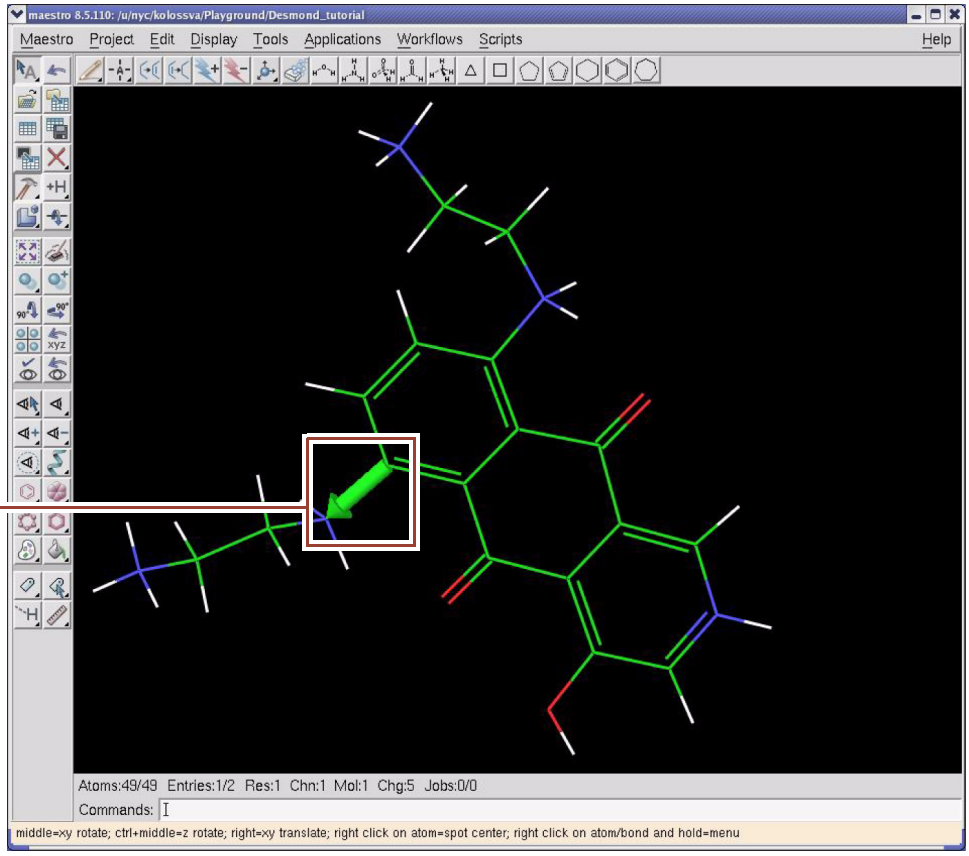


2. Select the 'Pick the attachment bond' option (the yellow button in Figure 7.3).

   When you define a structural perturbation for FEP, the ligand molecule is separated into two parts: a fixed part (called the *core*) and a variable part (called the *substitution group*). The substitution group will be mutated during simulation, and the core will remain intact. As shown in Figure 7.4, the core and the substitution groups are connected by a single bond termed the *attachment bond* (green arrow), which has to be picked by hand. Currently, the attachment bond can only be a single, covalent bond in the molecule.

NOTE When selecting the attachment bond, you determine the direction of the arrow by which half of the attachment bond you select. The attachment bond arrow should point from the core to the substitution group.

Figure 7.4 Defining the mutation



The attachment bond is shown as a green arrow in the Workspace. The arrow should point from the core to the substitution group.

3. Define the mutation to be performed by selecting the fragment(s) that will replace the substitution group from the Fragment Library. For this example, select the butyl row from the list of fragments. The selected row is highlighted in yellow as shown in Figure 7.3.

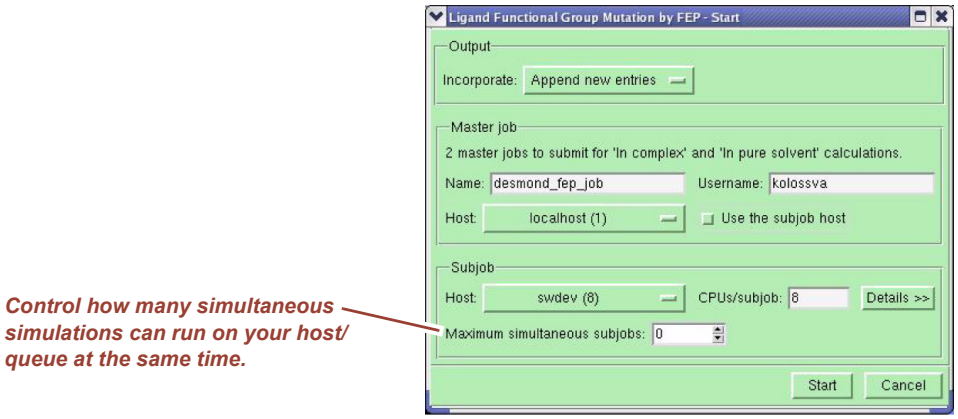
   You can apply multiple mutants at a time by using CTRL-click and SHIFT-click to select the mutants from the Fragment Library. When you select multiple fragments, FEP simulation creates different systems for each mutant, and runs multiple thermodynamic cycles to compute, respectively, the binding free energy difference between the reference ligand and each of the selected mutants, with respect to the receptor.

NOTE You can also create arbitrary groups rather than selecting mutations from the Fragment Library. See "Creating a Custom Fragment Group" on page 80 for details.

4. Click the green Start button. FEP simulation starts execution.

NOTE Desmond uses Bennett's acceptance ratio method for FEP calculations, which means that numerous independent simulations will be run for a single FEP calculation. Since the independent simulations can be run simultaneously, you should be careful as to how many simulations can be allowed to run on your host/queue at the same time. You can control this parameter in the green Start window by setting the maximum number of subjobs as shown in Figure 7.5. The value zero means no limit.

Figure 7.5 FEP Startup window



5. Alternatively, click the Write button. A total of four files are written:
   - **Two Maestro files:** `my-fep-job_lig.mae` and `my-fep-job_cmp.mae`. The first file contains the original ligand and one or multiple mutants. The second file contains the receptor and the ligand structures. Note that both files are regular Maestro files with no force field information included at this point. The only information about the FEP setup in these files is the definition of the attachment bond.
   - **Two MultiSim command script files:** `my-fep-job_solvent.msj` and `my-fep-job_complex.msj`. The first script defines a series of simulations involving the ligands in the solvent and the second script defines corresponding simulations involving the ligand-receptor complexes.

     Both `.msj` scripts can be run from the command line as described in "Running MultiSim jobs from the Command Line" on page 67.

NOTE Mutants do not need to be stoichiometric equivalents as in this example. Atoms that have no match in the mutant (or vice versa) will be annihilated into, or grown out of, dummy atoms during the course of the FEP simulation in accordance with the double topology technique.

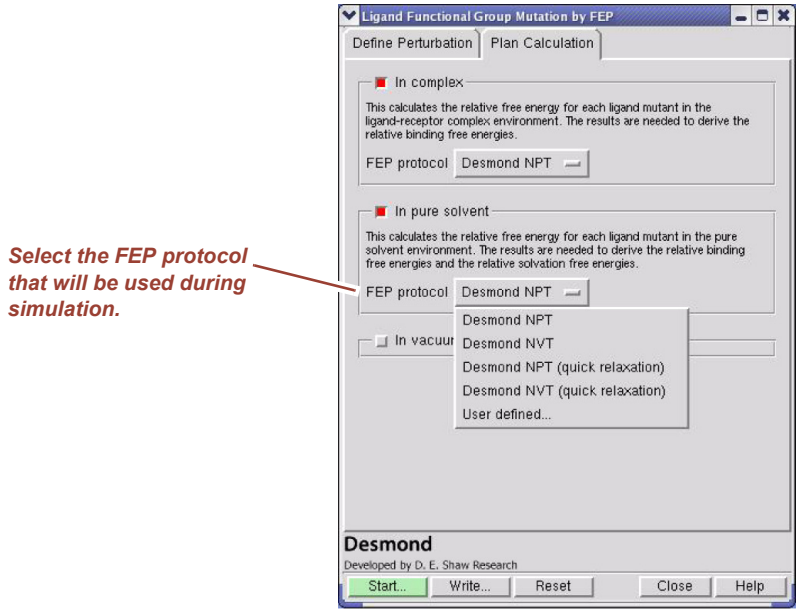
NOTE For more realistic simulations, you will need to adjust the conformation of the mutant. See "Adjusting the Conformation of the Mutant" on page 77 for details.

NOTE This tutorial does not go into details about the configuration settings for FEP simulations, see the *Desmond User Guide* and the *Schrödinger Desmond User Manual* listed in "Documentation Resources" on page 97 for a full account of this topic.

6. There are several ways to run the FEP simulation:
   - Write your own Desmond configuration file and run the FEP simulation from the command line. See "Running FEP Simulations from the Command Line" on page 76 for instructions to run the FEP simulation from the command line. This is the most flexible, but least automated approach.
   - Use Maestro to generate a Desmond configuration file and run the FEP simulation from the command line. See "Using Maestro To Generate A Desmond Configuration File" on page 75 for instructions on using Maestro to generate the configuration file. See "Running FEP Simulations from the Command Line" on page 76 for instructions to run the FEP simulation from the command line.

— Run the FEP simulation directly from the FEP Setup window by clicking the Start button. The workflow defined in the `.msj` files is automatically executed. This workflow has a short description shown under the Plan Calculation tab shown in Figure 7.6; essentially, system setup with System Builder is automatically performed, and all required Desmond simulations (NPT or NVT) are run with reasonable default settings, but user-defined workflows can also be imported. This option is the least flexible, but most convenient solution.

Figure 7.6 FEP workflow control



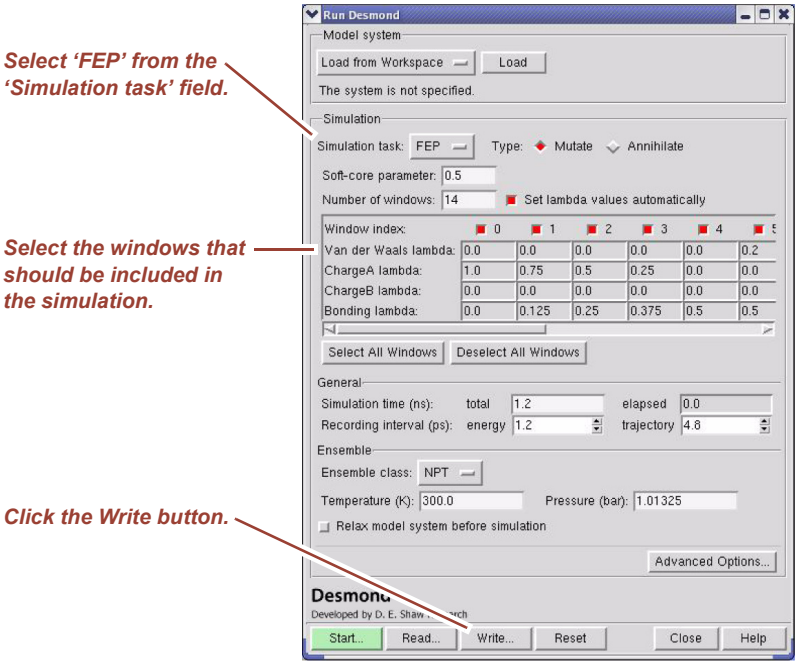
NOTE All of these options use the Schrödinger OPLS-AA force field server. Of course, you can use *Viparr* to generate force field parameters for other force fields, however, *Viparr* has limited coverage of the chemical space of organic compounds, which means that for most ligand mutations OPLS-AA is currently the only viable force field.

Using Maestro To Generate A Desmond Configuration File

To create a Desmond conguration file for FEP:

1. Select **Applications > Desmond > Run Desmond**. The Desmond window appears as shown in Figure 7.7.

Figure 7.7 Setting FEP parameters from the Run Desmond panel



2. Select FEP from the Simulation task field. The Simulation section of the panel allows you to set a number of FEP related parameters. For details see the *Desmond User Guide* and the *Schrödinger Desmond User Manual* listed in "Documentation Resources" on page 97.
3. Select the λ windows that should be included in FEP simulation.

NOTE FEP simulations involve many separate calculations with different values of λ and it is quite possible that some of them will crash at the first try. When this happens, you should rerun the FEP simulation for the failed λ windows. The GUI lets you select the failed λ windows for a re-run.

4. Write out the Desmond conguration file by clicking the Write button.

Running FEP Simulations from the Command Line

FEP simulations can be run from the command line using a `.cmj` file in exactly the same way as described in "Running MultiSim jobs from the Command Line" on page 67:

```
$SCHRODINGER/utilities/multisim -JOBNAME <jobname> -maxjob <maxjob>
-cpu <cpu> -HOST localhost -host <hostname> -i <jobname>.cms
-m <jobname>.msj -o <jobname>-out.cms
```

where:

- <jobname> is the name of the job.
- <maxjob> is the maximum number of subjobs that can be submitted to a host or a queue simultaneously. A value of zero means that there is no limit.
- <cpu> is the number of CPUs per subjob, which can also be specified as, e.g., "2 2 2" (note the quotes) indicating 8 CPUs with a spatial decomposition of 2x2x2.
- -HOST localhost means that the master job runs on your local workstation whereas
- <hostname> given with the -host option is the name of the host or queue where the simulation jobs will be running.

To apply non-default FEP λ parameters, add the '-cfg config_file' option to the command syntax where config_file is a Desmond conguration file.

Note that this conguration file will be used throughout the .cmj workflow to make sure that all minimization and MD runs use the same λ settings for consistency.

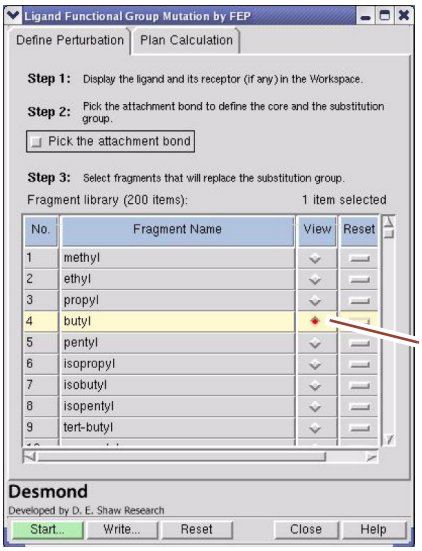
NOTE There is a Python script available, which can be used directly to compute the total free energy difference associated with a FEP job. $SCHRODINGER/desmond-v.../bin/Linux-x86/bennett.py will process the "energy" file output of the different λ runs and compute the free energy. Use the --help option to learn about the use of this script.

Adjusting the Conformation of the Mutant

The initial conformation of the mutant must be as close as possible to that of the original ligand for FEP simulations. Using Maestro, you can make manual adjustments to the conformation or configuration of small molecules.

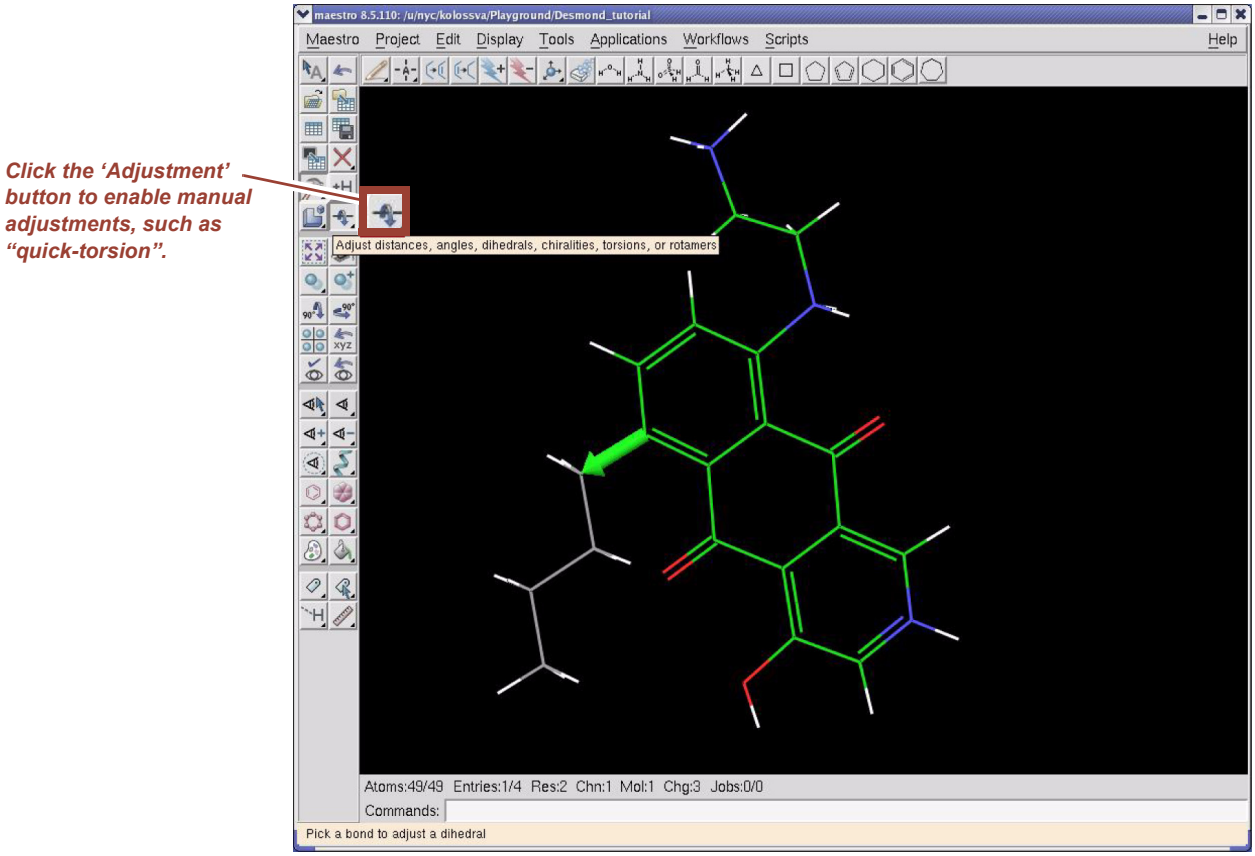
To show the mutant in its default conformation, click the button in the 'View' column of the Fragment Library for the fragment's row in the FEP Setup window. In Figure 7.8, the 'View' option is selected for the butyl fragment.

Figure 7.8 Showing a mutant in its default conformation



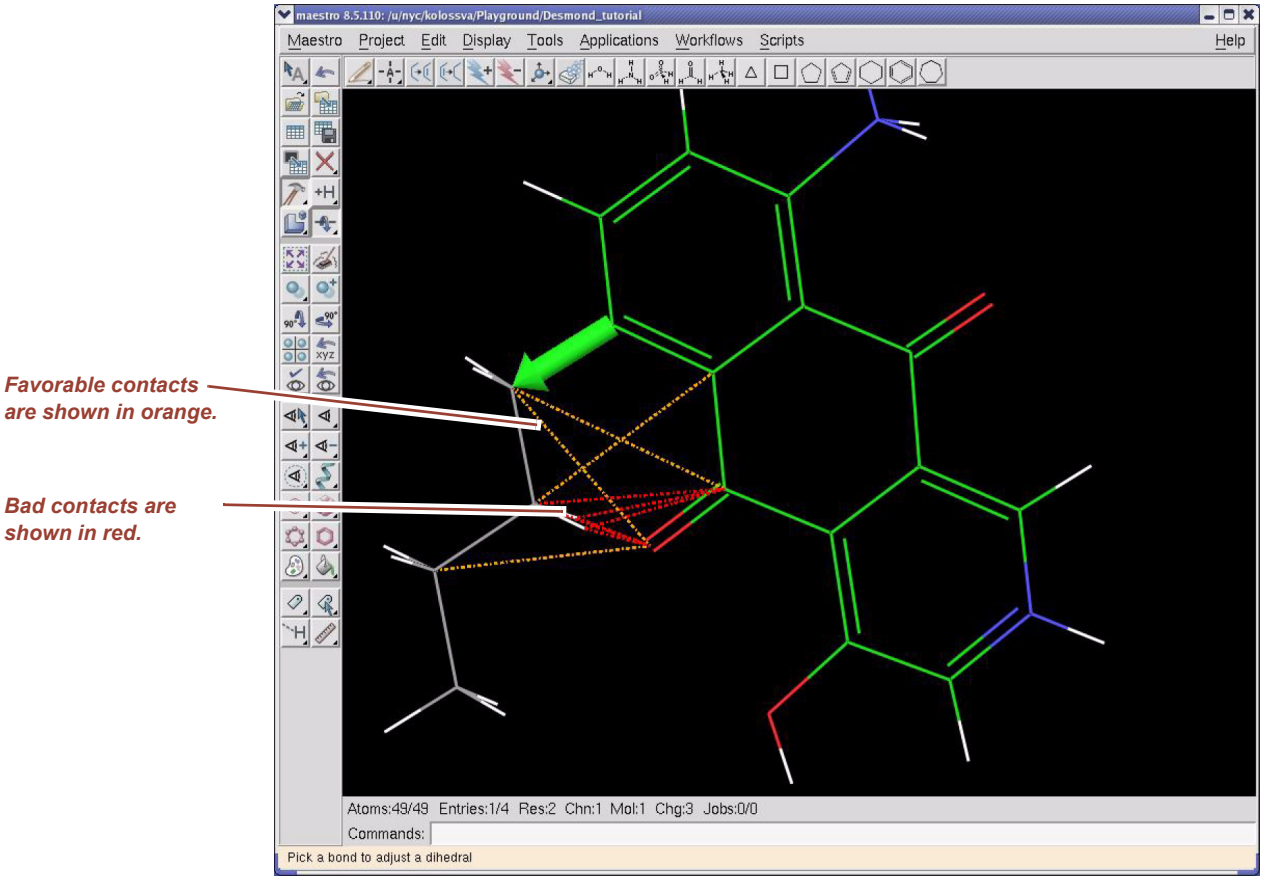
Now click Maestro's Adjustment tool on the left icon menu as shown in Figure 7.9 to turn on the manual adjustment option. You can click this button to switch between manual adjustment mode and global translation/rotation mode.

Figure 7.9 Manual adjustment of the substitution group conformation



Click the attachment bond (shown in green) to display the non-bonded contacts between the `butyl` fragment and the ligand core. As shown in Figure 7.10, there are favorable contacts (displayed in orange) and bad contacts (displayed in red).

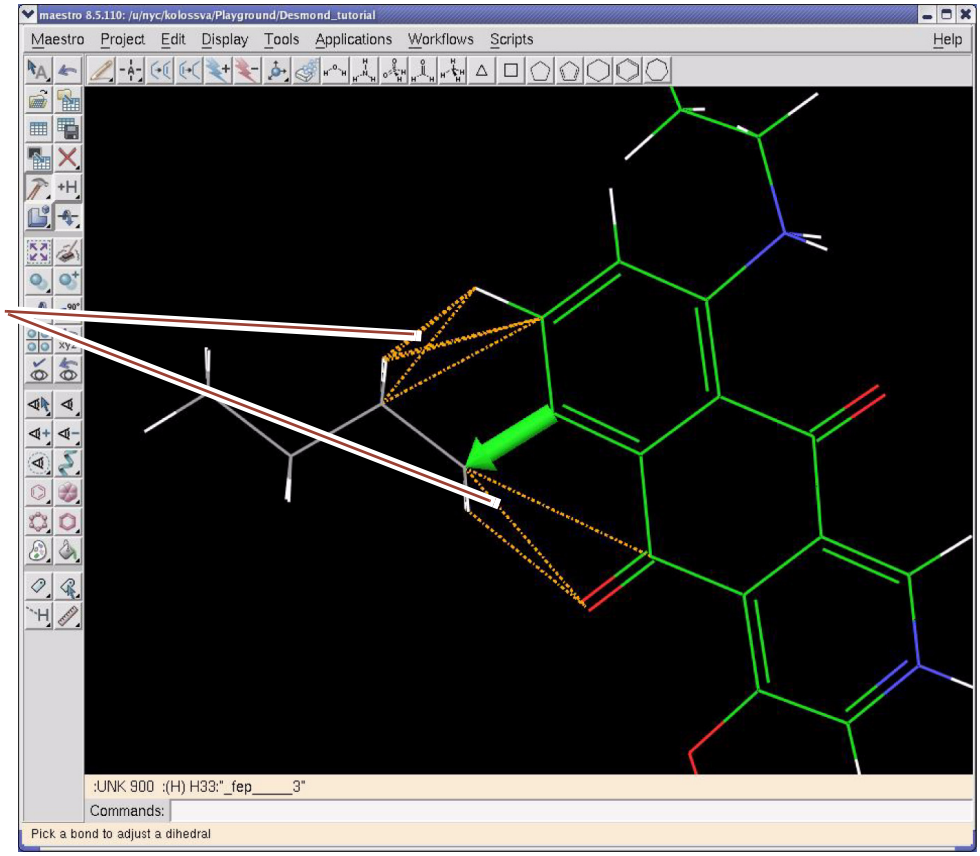
Figure 7.10 Displaying the non-bonded contacts



To arrive at a suitable conformation of the substitution group, you must adjust the torsion angle (and often, multiple angles). You can alter the torsion angle manually by holding down the left mouse button and moving the mouse horizontally, or by rotating the mouse wheel. For our example, the attachment bond was rotated 120 degrees to remove bad contacts. As shown in Figure 7.11, the altered `butyl` conformation is now fairly close to that of the original $-N^+H_2 - CH_2 - N^+H_2 - CH_3$ group.

To restore the default conformation, you can click the button in the 'Reset' column in the FEP Setup panel.

Figure 7.11 Bad contacts removed



Favorable contacts are the only contacts that remain. The bad contacts have been removed.

Creating a Custom Fragment Group

If the Fragment Library does not have the mutant structure you want to use, you can edit any pre-defined substitution group manually using the Build/Edit tools. The only caveat is that the customized substitution group will retain the name of the fragment that you modified. However, the FEP simulation will use the modified fragment.

8 Visualization and Analysis using Maestro

Overview

Maestro provides two basic visualization and analysis tools. Maestro's Trajectory Player is similar to the ePlayer found in the Project Table, which was mentioned in "Maestro Projects" on page 24, but it is specifically designed to nicely animate Desmond trajectories much faster than the ePlayer could, and it has a number of specific visualization options.

Maestro can also perform basic quality analysis using the Simulation Quality Analysis window.

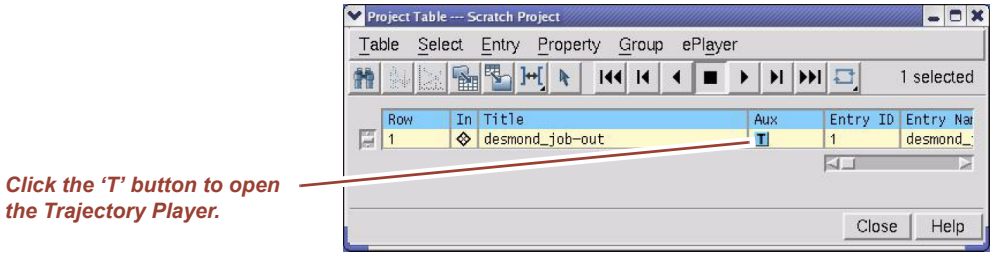
The following sections cover the basics of these tools.

Animating Desmond Trajectories with the Trajectory Player

Assuming that you have completed the first exercise in this tutorial in "Tutorial Steps" on page 3 you should have a completed Desmond simulation of the 4pti structure in your working directory with the base name of `desmond_job` by default (unless you gave it a different name).

1. Select **Project > Import Structures**: Read in the `desmond_job-out.cms` file. You will have to set the file format selector in the Import panel to CMS to display .cms files in the file browser window.
2. Select **Project > Show Table**: Open the Project Table. The Trajectory Player can be launched from the Project Table. Note the blue **T** button in the Aux column in the Project Table, as shown in Figure 8.1.

Figure 8.1 Launching the Trajectory Player



3. Click the **T** button to open the Trajectory Player.

   The Trajectory Player is shown in Figure 8.2 and the associated Maestro Workspace view can be seen in Figure 8.3.

   The Trajectory Player has two sets of options for frame control and display properties; the best way to learn about these options is to experiment with them. The Trajectory Player is fully documented in the *Schrödinger Desmond User Manual* listed in "Documentation Resources" on page 97.

NOTE In most cases you do not want the entire system to be displayed when, for example, visually inspecting the conformational changes of a protein molecule along the trajectory. The Trajectory Player recognizes Maestro's current display settings, which means that you can select/deselect any set of atoms for display using the Display > Display > Undisplay Atoms menu options and you can also render the atoms in any way you want. The Trajectory Player will generate a smooth animation with your current display settings and you can even save it in an MPEG movie file for incorporation in any presentation material.

Figure 8.2 The Trajectory Player

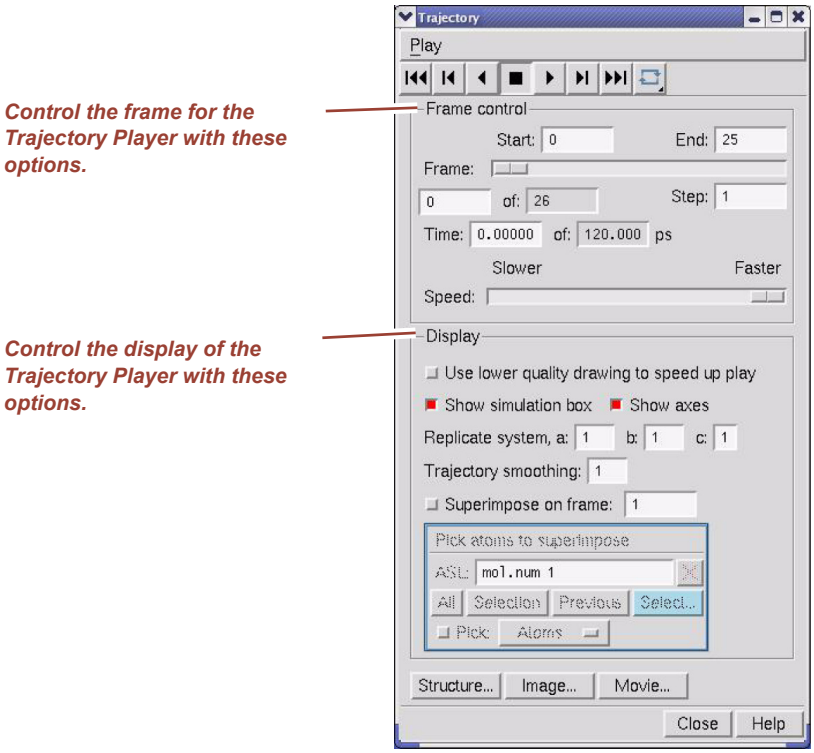
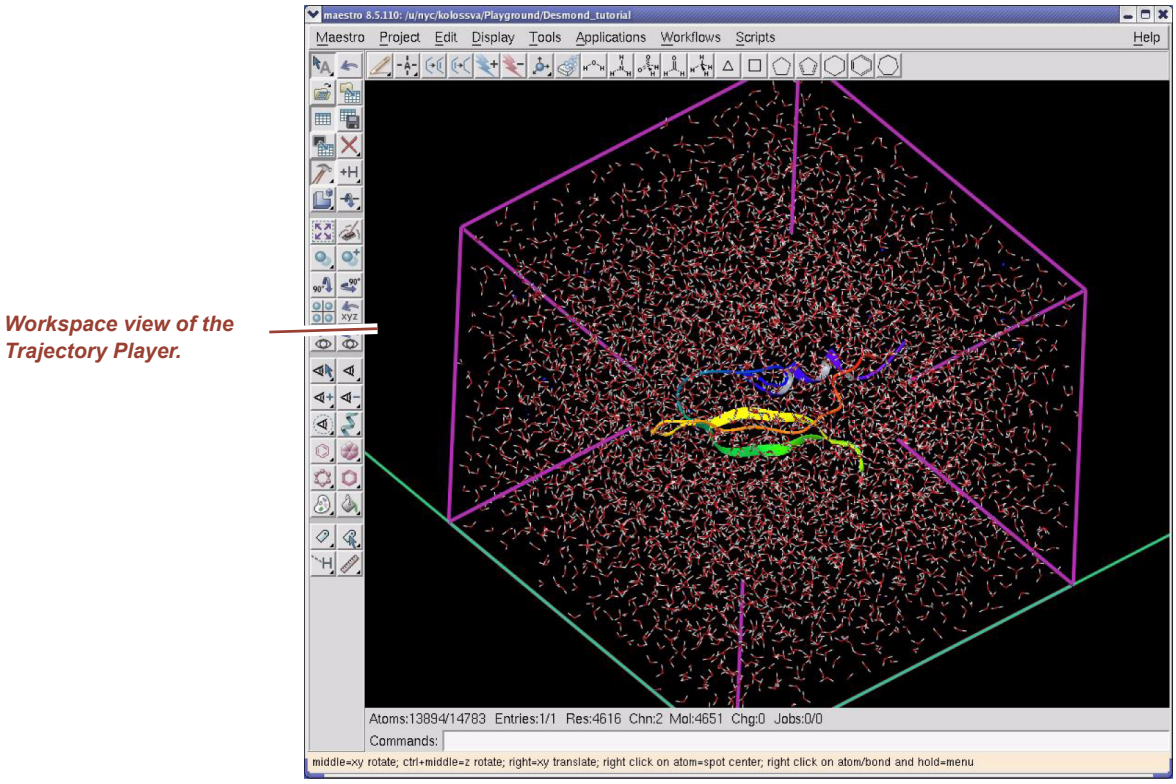


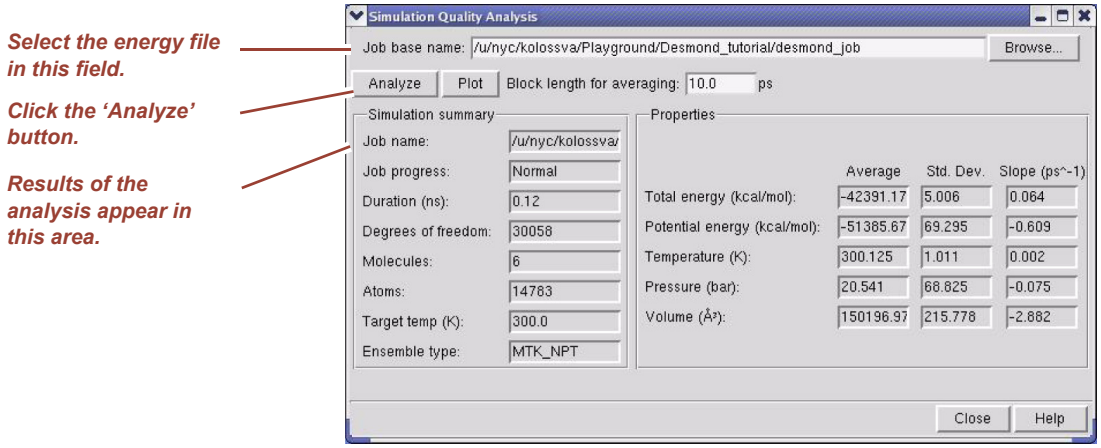
Figure 8.3 Workspace view for trajectory visualization



Performing Simulation Quality Analysis

Maestro can also perform basic quality analysis. Launch this tool from the **Applications > Desmond > Simulation Quality Analysis** menu. The quality analysis window appears as shown in Figure 8.4.

Figure 8.4 Simulation Quality Analysis window



Using the Browse button, read in the "energy" file `desmond_job.ene` and press the `Analyze` button. You should see similar output to that shown in Figure 8.4. The tables display useful information about simulation parameters and show the statistical properties of basic thermodynamic quantities based on block averages. You can find more details about simulation quality analysis including plotting options in the *Schrödinger Desmond User Manual* listed in "Documentation Resources" on page 97.

NOTE More sophisticated visualization and analysis tools can be applied to Desmond trajectories using an extension that allows you to import Desmond trajectories in the widely used VMD visualization package.

9 Analyzing Trajectories Using VMD

Overview

Desmond provides extensions to VMD that facilitate trajectory analysis. VMD is primarily a molecular visualization program, but it is a natural environment for analysis with its strong support for atom selections, the ability to efficiently animate trajectories that contain thousands of snapshots, and support for Tcl and Python scripting languages.

This section assumes a Unix/Linux environment and basic familiarity with VMD. If you're just getting started with VMD, a number of excellent tutorials already exist and can be found on the VMD home page:

http://www.ks.uiuc.edu/Research/vmd/current/docs.html#tutorials

The VMD Python Interface

The Python interface can be accessed from either the console (the terminal from which you launched VMD), or from an Idle window that runs inside of VMD (preferred). To initiate the Python interface in the console, simply type `gopython` at the `vmd>` prompt. To launch the built-in Idle window, you'll need to set your `PYTHONPATH` properly as described in the Desmond installation instructions.

The console command to launch the Idle window is

```
gopython -command "import vmdidle; vmdidle.start()"
```

You may wish to copy this command into your `.vmdrc` so that you open this window every time you launch VMD. Once you have a Python prompt at either the console or the Idle window, you're ready to start using the VMD Python interface.

In Python, classes and functions are organized into modules. Some modules are provided by the Python interpreter when you launch Python, while others are discovered at runtime and loaded from separate files. When you access Python from within VMD, the VMD executable provides several built-in modules for loading data into VMD, as well as querying and modifying VMD state. The most important of these are the `molecule` module, the `atomsel` module, and the `vmdnumpy` module. Use the built in `help()` command to get the most up-to-date documentation for these modules.

Loading and Viewing Trajectories

Before you begin, follow the instructions provided with the Desmond distribution to ensure that VMD can locate the plug-ins necessary to load CMS and DTR files.

Thanks to its file plugin architecture, VMD can load many different kinds of files. To simplify the discussion, we're going to assume that you're working with only two kinds of files:

- **Structure files:** `.mae` or `.cms` files created by Maestro or Desmond. These contain topology, mass, charge and possibly force field information about a single molecular system, including any solvent or ions that might be present. They also contain a single snapshot of the positions of the atoms and of any virtual sites in that system.

  In VMD, there are three different **file types** associated with CMS files. The default, `mae`, loads both topology and positions as described above. A second type, `mae0`, loads only topology but not coordinates; this may simplify some analysis tasks. The third type, `maev`, loads topology, positions, and velocities into VMD, if they are present in the file.

- **Trajectory files:** `.dtr` files created by Desmond. Although we will always refer to a trajectory as a file, the on-disk representation of a dtr is in fact a directory containing metadata files as well as frame files holding the actual snapshots. Resist the temptation to think of the frame files as being somehow independent of the rest of trajectory; they are not, and renaming or moving them around will cause issues.

  Like structure files, trajectory files can be loaded in any of three different modes, as specified by the file type. The default, `dtr`, loads just the positions. A second type, `dtrs`, also loads mass and atomic number information. Be sure the trajectory in fact contains that information, because if it doesn't, it will override any existing mass and atomic number that might have come from the structure file with zeros. Finally, the third type, `dtrv`, loads positions as well as velocities, if they are present in the frames.

Knowing the file type will be necessary for loading structure and trajectory files from either the command line or the scripting interface. Both structure files and trajectory files can be loaded in three different ways, with varying degrees of flexibility and ease of use.

Loading Files from the Command Line

If you are comfortable using the shell and the command line, loading files into VMD using command line arguments is the easiest way to go. The syntax is:

```
vmd -mae structure.cms -dtr trajectory.dtr
```

Launching VMD in this fashion will create a new molecule, initialize its structure from the `cms` file, create an initial set of coordinates from that same file, and finally append all the snapshots from the `dtr` file to that molecule.

Multiple structure and trajectory files can be loaded into separate molecules using combinations of the `-m` and `-f` options. The effect of `-m` is to cause a new molecule to be created for each subsequent file. The `-f` option causes subsequent files to be loaded into the same molecule. For example:

```
vmd -m -mae mol1.mae mol2.mae -f mol3.mae -dtr trajectory.dtr
```

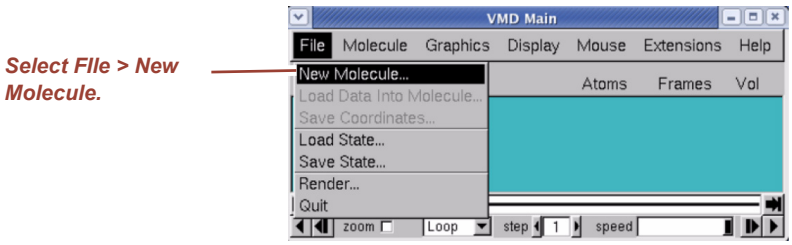
In this example, three molecules will be created. The first molecule will take data from `mol1.mae`, the second will take data from `mol2.mae`, and the third will take data from both `mol3.mae` and `trajectory.dtr`. Think of `-m` for loading multiple molecules, while `-f` is for loading multiple files.

Note that the command line loads all the snapshots from all the specified files before giving control back to the user. If your trajectories are so large that not all snapshots will fit into memory, you'll need to use either the GUI or the scripting interface to load a subset of snapshots into memory instead.

Loading Files from the GUI

VMD's File menu lets you load CMS and DTR files using a graphical user interface (GUI). Select **File > New Molecule** in the Main menu (Figure 9.1). Click the Browse button, navigate to your structure file, and click **OK** or press **Enter** to select the file. You should now see your file in the Filename field to the left of the Browse button, and underneath the filename the file type chooser should show Maestro File as its selected item (Fig. 80). If you want to load the CMS file as an `mae0` or `maev` file, corresponding to topology or to positions and velocities, choose either Maestro File (no positions) or Maestro File (with velocities) in the file type chooser. When the file and its type are to your satisfaction, click Load to proceed.

Figure 9.1 Loading files from the GUI



Once you've loaded a molecule, you'll notice that the molecule chooser at the top of the Molecule File Browser now points to your newly created molecule. If you create more new molecules using different CMS files, select New Molecule from the molecule chooser and repeat the steps above. Otherwise, to append snapshots to this molecule from a trajectory, click Browse again and navigate to the location of the DTR file you wish to load. Since DTR trajectories are stored as directories rather than ordinary files, clicking on a DTR trajectory will take you inside of a directory (Fig. 81). You should see a file called `clickme.dtr`. Click on it, and VMD will figure out that you wanted to load the frames within the directory containing the `clickme` file.

Figure 9.2 Loading files from the GUI

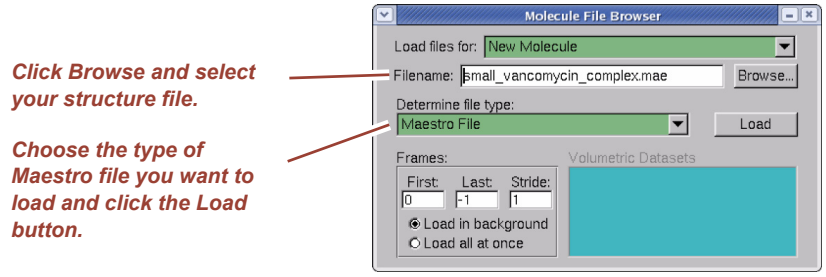
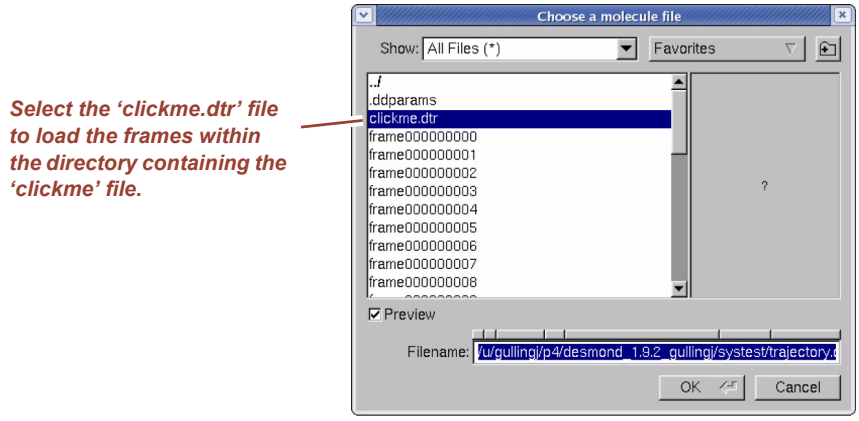


Figure 9.3 Loading files from the GUI



At this point, you may wish to limit the number of snapshots that are loaded from the trajectory. If you haven't done so already, choose Desmond Trajectory from the file type chooser. The Frames box underneath should now be active. Type in values for the snapshots you want to load. The default First value of 0 corresponds to the first snapshot in the trajectory; the default Last value of −1 corresponds to the last snapshot; and a Stride of 1 means load every snapshot in the selected range. Choosing a Stride of 10 would load every 10th snapshot from the trajectory.

You also have the choice of Load in background, which is slower but keeps VMD responsive during the load and gives you the option of canceling the load before it completes, and Load all at once, which is quite a bit faster but leaves VMD in an unresponsive state until the load has completed. Use Load all at once whenever you're reasonable sure that all the frames you're about to load will fit in memory.

Loading files from the scripting interface

Files can also be loaded into VMD using the Tcl or Python scripting interface. We'll confine ourselves here to the Python interface, although you may find using the Tcl interface with VMD TkCon window more convenient for simple tasks.

Load the molecule module by typing `import molecule` at the Python prompt. To load a structure file and all frames of a trajectory file, enter the following commands:

```
import molecule
molid = molecule.read(molid=-1, filetype='mae', filename='struc-
ture.cms')
```

```
molecule.read(molid, filetype='dtr', filename='trajectory,dtr', waitfor=-
1)
```

The −1 argument to `molecule.read` indicates that a new molecule should be created for the file. We specify filetype and filename to load the structure. Finally, we use the `molid` returned by the first read command as the first argument to the second read command so that data from the trajectory gets appended to the original structure file. The `waitfor=-1` means to load all frames in the trajectory before returning from the call. This corresponds to the 'Load all at once' option in the File menu.

You can also specify a range of frames to be loaded:

```
molecule.read(molid, 'dtr', 'trajectory.dtr',
beg=5, end=55, skip=10, waitfor=-1)
```

Assuming there were at least 55 snapshots in `trajectory.dtr`, this command would load snapshots 5, 15, 25, 35, 45, and 55 into the molecule with the given `molid`.

Getting information about snapshots

Data in VMD is organized into *Molecules*, which represent entire molecular systems as well any number of snapshots associated with those systems. As molecules are loaded, they are assigned a integer key called a `molid`. Exactly one molecule at any given time is tagged as the `top` molecule; unless you set the top molecule yourself, this will always be the most recently loaded molecule.

Each molecule in VMD has a fixed number of atoms. To each atom, VMD associates various attributes such as name, type, mass, charge, etc. This data is usually obtained from a structure file, but missing values not supplied by the structure file are guessed, and the stored values can be overridden using scripting commands as described below.

The snapshots associated with each molecule are referred to as frames. Each frame holds one set of coordinates for each atom in the molecule, as well as unit cell dimensions for that frame. Velocities can also be stored if the trajectory file supplies them and the user requests it using the appropriate file type. Table 1 summarizes the functions in the molecule module for querying the high level structure of molecules in VMD.

In the rest of this section, we outline the Python interface provided by VMD to extract atom and frame information from a molecule.

Atom selections

An atom selection consists of all the atoms in a particular molecule and frame that satisfy some predicate, or set of conditions. We often refer to the predicate as the selection, but it should be kept in mind that the set of atoms satisfying the predicate could be different for different frames. Atom selections are used in the GUI to select which atoms to draw with each style. The same selection language is used in the scripting interface to specify a predicate for an atom selection object.

Atom selection objects are created in the VMD Python interface using the `atomsel` module:

```
# Import the atomsel type
from atomsel import atomsel
# Create a selection corresponding to all atoms in the top molecule,
```

```
# pointing to the current frame.
all = atomsel()
```

Table 9.1 Summary of functions from VMD built-in molecule module

function	result
`new(name)`	`molid` of newly created molecule
`listall()`	list of available `molid` values
`numatoms(m)`	number of atoms in molecule `m`
`numframes(m)`	number of frames (snapshots) in molecule `m`
`get_top()`	`molid` of top molecule; -1 if none
`set_top(m)`	make molecule `m` the top molecule
`get_frame(m)`	get current frame of molecule `m`
`set_frame(m, frame)`	set current frame for molecule `m` to `frame`
`get_periodic(m=-1, frame=-1)`	dictionary of periodic cell data; default top molecule, current timestep
`set_periodic(m=-1, frame=-1, )`	set periodic cell keywords `a`, `b`, `c`, `alpha`, `beta`, `gamma`
`get_physical_time(m=-1, frame=-1)`	time value associated with timestep; default top molecule, current timestep
`read(molid, filetype, filename, ...)`	load molecule files
`write(molid, filetype, filename, ...)`	save molecule files

```
# Select just the atoms named 'CA' (i.e., the alpha carbons), in the
# top molecule in the current frame
ca = atomsel(selection='name CA')

# Select the protein atoms in frame 10 of the top molecule; if the
# current frame changes, the selection will still point to the same
# frame.
pro10 = atomsel('protein', frame=10)

# Select the waters near the protein from molecules 0 and 1.
wat0 = atomsel('water and same residue as (within 5 of protein)',
molid=0)
wat1 = atomsel('water and same residue as (within 5 of protein)',
molid=1)
```

As illustrated above, the `atomsel` type takes three arguments, all of which are optional:

- `selection`; default `all`. This is the selection predicate. The selection language is described in full in the *VMD User's Guide*.
- `molid`; default `-1`. The `molid` is the molecule associated with the selection, and never changes. The default `molid` of `-1` means the top molecule at the time that the selection was created. If the top molecule happens to change after the selection is created, for instance, when a new molecule is loaded, the `molid` does not change. If a molecule gets deleted from VMD's workspace, all associated atom selections are invalidated.

- `frame`; default `-1`. The `frame` is the index of the snapshot referenced by the selection. The default value of `-1` means that the selection should reference whatever the current frame happens to be. A `frame` value of 0 or higher means the selection always refers to the same frame, even if the current frame changes. If the frame referenced by selection does not exist, then the last frame in the molecule is used.

You can get and set the frame attribute in the usual Python way:

```
n1 = len(wat0)    # number of atoms in the selection
f = wat0.frame    # returns -1
wat0.frame = 20   # makes wat0 reference frame 20
n2 = len(wat0)    # will be equal to n1!
wat0.update()     # recompute the selection based on frame 20
n3 = len(wat0)    # might be different from n1
```

Note, however, that changing the frame does *not* cause the selection to be recomputed. Only the `update()` method changes the set of atoms referenced by an existing selection. However, changing the frame *does* change the result of calculations that depend on the coordinates, such as `center()` and `rmsd()`. The reason things are done this way is because one might want to select a set of atoms based on a distance criterion, such as the `wat0` selection in the example above, and then see howthe positions of those atoms change over time. There would be no way to do this if changing the frame automatically updated the selection.

With an atom selection object in hand, you can extract data about a molecule for the atoms in the selection. Continuing from the first example:

```
# Get the number of alpha carbons
num_ca = len(ca)

# get the residue name of each alpha carbon
resnames = ca.get('resname')

# Set the 'type' attribute of all atoms to 'X'
all.set('type', 'X')

# Compute the center of mass the protein atoms in frame 10
mass = pro10.get('mass')
com = pro10.center(mass)

# Align pro10 to the corresponding atoms in frame 0, and compute
# the RMSD relative to frame 0.
pro0 = atomsel('protein', frame=0)
m = pro10.fit(pro0, weight=mass)
pro10.move(m)
rms = pro10.fit(pro0)
```

Snapshots

Time-varying data in VMD includes positions, velocities, and periodic cell parameters. The atom selection interface can be used to extract positions and velocities from a given frame in the same way as other attributes:

```
# get the x, y, and z coordinates for the atoms in wat1 as Python lists
x = wat1.get('x')
y = wat1.get('y')
```

```
z = wat1.get('z')
# get the z component of the velocity
vz = wat1.get('vz')
```

However, this method comes with significant overhead if a large number of snapshots are to be processed, and the Python lists returned by the `get()` are not as efficient or convenient to work with as NumPy arrays.

For this reason, VMD implements a built-in module called `vmdnumpy` that returns positions and velocities as NumPy arrays, rather than lists. The syntax is

```
vmdnumpy.positions(molid=-1, frame=-1) # Return reference to coordinates
vmdnumpy.velocities(molid=-1, frame=-1) # Return reference to velocities
```

The arrays returned by `positions()` and `velocities()` are zero-copy references to the snapshot data stored in VMD. They will be sized as N x 3 arrays, where N is the number of atoms in the molecule. The x, y, and z components of the position or velocity are stored in columns 0, 1, and 2 of the respective arrays. If a frame holds no velocities, then the `velocities()` method will return `None`. The `molid` argument defaults to the top molecule, just like atom selections, and the frame defaults to the current frame at the time the array is returned. Since the arrays reference snapshot data without making a copy, they must not be used after the frame to which they refer is deleted; any attempt to do so will probably result in a program crash. However, accessing positions with `vmdnumpy` is far more efficient than extracting coordinates from atom selections. The most efficient way to extract a subset of the coordinates corresponding to a selection is to use the atom indices as a slice argument to the positions array:

```
pos = vmdnumpy.positions() # all positions in top molecule's current
                           frame
inds = ca.get('index')     # index of all CA atoms
ca_pos = pos[inds]         # M x 3 array, where M = len(ca)
```

As summarized in Table 9.1, the periodic cell data is returned by `molecule.get_periodic()` as a dictionary whose keys are `a`, `b`, `c`, `alpha`, `beta`, and `gamma`, corresponding to the lengths of the three unit cell shift vectors and the angle between `b` and `c`, `a` and `c`, and `a` and `b`, respectively. To change the unit cell by hand, use `molecule.set_periodic()` with the same keyword arguments as the dictionary to change the desired attributes.

Writing structures and trajectories

Modied structures or trajectories can be written back out to disk for further analysis or as input to simulation preparation. There are two distinct methods for writing out coordinates. The first method for writing coordinates and structure is the `molecule.write` command:

```
molecule.write(molid, 'dtr', 'trajectory.dtr', waitfor=-1)
```

The `molid`, filetype, and filename are required parameters. The `waitfor` option indicates how many frames should be written before returning from the command; you will almost want to use -1. You can also specify a range of frames to be written, just like the `molecule.read` command:

```
molecule.write(molid, 'dtr', 'trajectory.dtr', beg=100, end=-1, skip=10,
waitfor=-1)
```

This command would write every 10th frame starting from frame 100. Coordinates can also be written using an atom selection:

```
sel = atomsel('name CA')
sel.frame = 45
sel.write('mae', 'ca45.mae')
```

This command would create write a Maestro file containing just the CA atoms whose coordinates were taken from frame 45.

Analyzing whole trajectories

We can now give a nontrivial trajectory analysis example that illustrates the use of `atomsel`, `vmdnumpy`, and `molecule`. The script shown in Figure 9.4 runs through all the frames in a molecule, aligns each frame to the first frame, and computes the aligned RMSD for each frame and the averaged position of the alpha carbons. To use this script, it's necessary to load the reference structure and any trajectories to be processed into VMD, using any of the methods outlined in "Loading and Viewing Trajectories" on page 86.

Once that's done, you can launch the script as described in "The VMD Python Interface" on page 85.

Figure 9.4 Analysis script example

```
from atomsel import atomsel
import molecule, vmdnumpy
import numpy

avg = None                # will hold averaged coordinates
all = atomsel()           # all atoms in current frame
ref = atomsel('name CA', frame=0) # reference frame is frame 0
sel = atomsel('name CA')  # alpha carbons in current frame
inds = sel.get('index')   # indexes of alpha carbons don't change
mass = sel.get('mass')    # masses don't change, either
rms = [0.]                # will hold RMSD to reference frame

def processFrame():
    global avg
    all.move( sel.fit(ref, mass) ) # Align current frame w/ref frame
    rms.append( sel.rmsd(ref, mass) ) # Append new RMSD value
    # Get positions of CA atoms
    pos = vmdnumpy.positions()
    ca_pos = pos[inds]

    # Accumulate positions into average
    if avg is None: avg = ca_pos[:] # make a copy of ca_pos
    else: avg += ca_pos # add this frame's contribution

# Loop over frames
n = molecule.numframes(molecule.get_top()
for i in range(1,n ):
    molecule.set_frame(molecule.get_top(), i)
    processFrame()
```

```
# Scale the average
avg /= n
```

Big trajectories

It is often the case that not all frames of a trajectory can fit into memory at once. The `BigTrajectory` framework provided by the Desmond distribution makes it simple to circumvent this limitation for analysis scripts that need to process just one frame at a time.

In order to use `BigTrajectory`, the user writes one or more analysis tasks that implement a `processFrame()` method like the one described in the previous section. `BigTrajectory` then takes care of looping through snapshots one at time and calling each task in turn to process the data.

Any number of reference frames can be kept in memory while the trajectory frames are processed, so that tasks like RMSD alignment can be implemented. Organizing your analysis code into tasks encourages module code development and reuse. Figure 9.5 shows how, with only trivial changes, the script in Figure 9.4 on page 93 can be adapted to be used in `BigTrajectory`.

Figure 9.5 Analysis task using BigTrajectory

```
##
## file: Analysis.py
##

from atomsel import atomsel
import molecule, vmdnumpy
import numpy
import BigTrajectory

class AnalysisTask:
     def __init__(self, rmsfile):
       self.avg = None            # will hold averaged coordinates
       self.count = 0             # number of processed frames
       self.all = atomsel()       # all atoms in current frame
       self.ref = atomsel('name CA', frame=0) # reference frame is
                                                frame 0
       self.sel = atomsel('name CA') # alpha carbons in current frame
       self.inds = sel.get('index')
       self.mass = sel.get('mass')

       # write initial 0 to rmsfile output
       self.rmsfile.write("%f\n" % 0)

     def processFrame(self):
       # Align current frame with reference frame
       self.all.move( self.sel.fit(self.ref, self.mass) )

       # Write out next value of RMSD
       self.rmsfile.write( "%f\n" % self.sel.rmsd(self.ref, self.mass) )
```

```
    # Get positions of CA atoms
    pos = vmdnumpy.positions()
    ca_pos = pos[self.inds]

    # Accumulate positions into average and increment count
    if self.avg is None: self.avg = ca_pos[:]
    else: self.avg += ca_pos
    self.count += 1

  def finish(self):
    self.avg /= self.count
    self.rmsfile.flush()
```

The most obvious difference between the `BigTrajectory` version of the script and the original version is that we've packaged the data for the task into a Python class. This means we have to access our data structures through the self reference, rather than as global variables. Another difference is that, rather than simply appending the RMSD values to a list, we write them to a file; this keeps us from running ourselves out of memory if we end up processing a huge number of frames.

To use this task in our own analysis script, let's assume that the contents of Figure 9.5 are stored in a Python module called `Analysis`. Rather than copying and pasting the task into our script, we import the definition of the task into our workspace, then call `BigTrajectory.analyze` to apply the task to our own files. Figure 9.6 illustrates how this might look.

Figure 9.6 Analysis script using Figure 9.5 and BigTrajectory

```
import Analysis
import BigTrajectory
import sys

inputfiles = sys.argv[1:]
rmsfile = open('rms.dat', 'w')
task = Analysis.AnalysisTask(rmsfile)
BigTrajectory.analyze( inputfiles, task )
exit()
```

10 Documentation Resources

Detailed Desmond documentation can be found in the D. E. Shaw Research *Desmond User Guide*.

Schrödinger's Desmond documentation includes the *Quick Start Guide* and the *Desmond User Manual*:

```
$SCHRODINGER/docs/desmond/quick_start/des20_quick_start.pdf
```

and

```
$SCHRODINGER/docs/desmond/user_manual/des20_user_manual.pdf
```

Maestro documentation can be found online at **www.schrodinger.com** and context specific online help can be brought up in any Maestro panel, which has a Help button. The complete Maestro documentation set includes:

```
$SCHRODINGER/docs/maestro/overview/mae85_overview.pdf,
$SCHRODINGER/docs/maestro/ref_manual/mae85_command_reference.pdf,
$SCHRODINGER/docs/maestro/tutorial/mae85_tutorial.pdf,
$SCHRODINGER/docs/maestro/user_manual/mae85_user_manual.pdf.
```

The full ASL (atom specification language) documentation can be found in:

```
$SCHRODINGER/docs/maestro/help/mae85_help/misc/asl.html
```

The documentation of Prime can be found at

```
$SCHRODINGER/docs/prime/quick_start/pri20_quick_start.pdf
```

and

```
$SCHRODINGER/docs/prime/user_manual/pri20_user_manual.pdf
```