

# **Using AutoDock with AutoDockTools: A Tutorial**

*Written by Ruth Huey and Garrett M. Morris*

*The Scripps Research Institute  
Molecular Graphics Laboratory  
10550 N. Torrey Pines Rd.  
La Jolla, California 92037-1000  
USA*

*22 January 2003*

---

## **Contents**

<b>Introduction</b> .....	<b>4</b>
Before We Start.....	4
<b>FAQ – Frequently Asked Questions</b> .....	<b>5</b>
<b>Exercise One: PDB Files are Not Perfect: Editing a PDB file</b> .....	<b>7</b>
Procedure: .....	7
<b>Exercise Two: Preparing a ligand file for AutoDock</b> .....	<b>12</b>
Procedure: .....	12
<b>Exercise Three: Preparing the macromolecule file</b> .....	<b>16</b>
Procedure: .....	16
<b>Exercise Four: Preparing the grid parameter file</b> .....	<b>18</b>
Procedure: .....	18
<b>Exercise Five: Starting AutoGrid</b> .....	<b>21</b>
Procedure: .....	21
<b>Exercise Six: Preparing the docking parameter file</b> .....	<b>23</b>
Procedure: .....	23
<b>Exercise Seven: Starting AutoDock</b> .....	<b>25</b>
Procedure: .....	25
<b>Exercise Eight: Analyzing AutoDock Results-Reading Docking Logs</b> .....	<b>27</b>
Procedure: .....	27
<b>Exercise Nine: Analyzing AutoDock Results-Visualizing Docked Conformations</b> .....	<b>29</b>
Procedure: .....	29
<b>Exercise Ten: Analyzing AutoDock Results -Clustering Conformations</b> .....	<b>31</b>
Procedure: .....	31
<b>Exercise Eleven: Analyzing AutoDock Results-Visualizing Conformations in Context</b> .....	<b>34</b>
Procedure: .....	34
<b>Files for exercises:</b> .....	<b>37</b>
Input Files:.....	37
Results Files .....	37

Useful Scripts .....	37
Customization Options for ADT.....	37
<b>Appendix 1 .....</b>	<b>38</b>
<b>Appendix 2: Docking Parameters .....</b>	<b>64</b>
Parameters common to SA, GA, GALS: .....	64
Simulated Annealing Specific Parameters:.....	66
Genetic Algorithm Specific Parameters: .....	67
Local Search Specific Parameters:.....	68
Clustering keywords: .....	69

---

## ***Introduction***

This tutorial will introduce you to docking using the **AutoDock** suite of programs. We will use a Graphical User Interface called **AutoDockTools**, or **ADT**, that helps a user easily set up the two molecules for docking, launches the external number crunching jobs in **AutoDock**, and when the dockings are completed also lets the user interactively visualize the docking results in 3D.

### **Before We Start...**

And only if you are at The Scripps Research Institute... These commands are for people attending the tutorial given at Scripps. We will be starting the graphical user interface to AutoDock from the command line. To do this, you need to open a Terminal window and then type this at the UNIX, Mac OS X or Linux prompt:

```
% source /tsri/python/share/bin/initadtcsh  
% cd tutorial  
% adt1
```

---

## **FAQ – Frequently Asked Questions**

1. *Where should I start **ADT**?*

You should always start **ADT** in the same directory as the macromolecule and ligand files. You can start **ADT** from the command line in a Terminal by typing “**adt**” and pressing <Return> or <Enter>.

2. *Should I always add polar hydrogens?*

Yes, for the macromolecule you should always add polar hydrogens, and then assign Kollman United Atom charges.

For the ligand, you should add all hydrogens before computing Gasteiger charges, and then you must merge the non-polar hydrogens. If your ligand is a peptide, then you can also just add polar hydrogens and assign Kollman United Atom charges.

Polar hydrogens are hydrogens that are bonded to electronegative atoms like oxygen and nitrogen. Non-polar hydrogens are those bonded to carbon atoms.

3. *How many **AutoGrid** grid maps do I need?*

You need one **AutoGrid** map for every atom type in the ligand. *E.g.*: in ethanol, C<sub>2</sub>H<sub>5</sub>OH, you would need C, O and H maps.

4. *Why should all the total charges on the residues be an integer?*

This is because it is assumed that the residue is interchangeable with others, and that no electrons are withdrawn or received by adjacent residues. In proteins, *e.g.*, arginines should have a total charge of +1.000 if they are protonated, or 0.000 if they are neutral.

5. *How easy will it be to get good docking results?*

In general, the more rotatable bonds in the ligand, the more difficult it will be to find good binding modes in repeated docking experiments.

6. *How big should the **AutoGrid** grid box be?*

The grid volume should be large enough to at least allow the ligand to rotate freely, even when the ligand is in its most fully-extended conformation.

7. *Can I identify potential binding sites of a ligand on a protein with **AutoDock**?*

Yes, if you do not know where the ligand binds, you can build a grid volume that is big enough to cover the entire surface of the protein, using a larger grid spacing than the default value of 0.375Å, and more grid points in each dimension. Then you can perform preliminary docking experiments with **AutoDock** to see if there are particular regions of the protein that are preferred by the ligand. This is sometimes referred to as “*blind docking*”.

Then, in a second round of docking experiments, you can build smaller grids around these potential binding sites and dock in these smaller grids.

If the protein is very large, then you can break it up into overlapping grids and dock into each of these grid sets, *e.g.* one covering the top half, one covering the lower half, and one covering the middle half.

---

## Exercise One: PDB Files are Not Perfect: Editing a PDB file

Protein Data Bank (PDB) files can have a variety of potential problems that need to be corrected before they can be used in **AutoDock**. These potential problems include missing atoms, added waters, more than one molecule, chain breaks, alternate locations *etc.*

AutoDockTools (ADT) is built on the Python Molecule Viewer (PMV), and has an evolving set of tools designed to solve these kinds of problems. In particular, two modules, **editCommands** and **repairCommands**, contain many useful tools which permit you to add or remove hydrogens, repair residues by adding missing atoms, modify histidine protonation, modify protonation of intrachain breaks, *etc.*

**Note:** any problems in your ligand must be corrected also.

In this exercise, you will work on the macromolecule, the molecule we want to dock to and which will be kept fixed during the dockings. You will learn how to remove waters, how to add the polar hydrogens that AutoDock expects, and how to save the modified result.

### Procedure:

1. **File** → **Read molecule**

This will open a file browser, showing all the files in the current directory. Select **hsg1.pdb** and click on **Open**. Alternatively, instead of using the mouse to click on the button in the GUI, you could also press the **<Enter>** key on the keyboard while the cursor is still in the entry. This is true for many parts of the GUI in both PMV and ADT.

This loads a **Molecule** named 'hsg1' into ADT. If the molecule appears in the ADT viewer, you can skip to Step 3.

You might have a three-button mouse. If so, the mouse buttons can be used alone or with a modifier key to perform different operations. To zoom the molecule (make the molecule look bigger or smaller) in the viewer window, press and hold down the **<Shift>** key and then click and drag with the middle mouse button. To rotate the molecule, just click and drag with the middle mouse button. To summarize what the mouse buttons do:

Modifier	Left	Middle	Right
None	<i>Pick</i>	<i>Rotate</i>	<i>Translate left/right (X) and up/down (Y)</i>
Shift	<i>Set center of rotation</i>	<i>Scale or Zoom</i>	<i>Translate in/out (Z)</i>

You can also press the following keys in the viewer window to change the view of the molecule:

Key	Action
R	<i>Reset view</i>
N	<i>Normalize – scale molecule(s) so all visible molecules fit in the viewer</i>
C	<i>Center on the center of gravity of all the molecules</i>
D	<i>Toggle on/off <u>Depth-cueing</u> (blends molecule into background farther away)</i>

## 2. Edit → Bonds → Build By Distance

**Note:** You can get this behavior automatically either by setting the **File** → **setOnAddObjectCommands** checkbox for **buildBondsByDistance** or writing a **userpref** in your **.pmvrc** file.

This command builds bonds between atoms based on their van der Waals (vdW) radii. It has the side effect of calling **Un/Display** → **Lines**, so you should be able to see the molecule at this point.

The bonds between bonded atoms are represented as lines; while non-bonded atoms, such as metal ions and oxygen atoms of water molecules, are shown as small squares. The non-bonded atoms you see here in 'hsg1' are the oxygen atoms of waters that were present in the crystal structure. We will remove these waters later.

3. **Color** → **by Atom Type**

Click on **All Geometries** and then click **OK**. All of the displayed objects will be colored according to the chemical element, as follows:

- Carbons that are aliphatic (C) - white,
- Carbons that are aromatic (A) - green,
- Nitrogens (N) - blue,
- Oxygens (O) - red,
- Sulfurs (S) - yellow,
- Hydrogens (H) - cyan.

This makes the display more informative.

4. **File** → **LoadModule**

Highlight **deleteCommands** in the list of available Pmv modules. Click on **Load Module** to load it.

Now scroll down the list to see the **selectionCommands** option. Highlight **selectionCommands** in the list by clicking on it, and then click on **Load Module** to load it.

You have just loaded a set of selection and delete commands that we will use to remove waters from hsg1. Click on **Dismiss** to close the widget.

5. **Select** → **Select From String**

**Select From String** lets you build a selection based on strings you enter for the Molecule, Chain, Residue and/or Atom level. These strings can be names, numbers, ranges of numbers, or lambda expressions which are evaluated to build a set. The strings can contain regular expressions including wild cards such as \* which matches anything. For our example, we want to select all atoms (\*) in residues named HOH\*. Type **HOH\*** in the Residue entry, press the <Tab> key to move to the next entry field, and type \* in the Atom entry. Now click **Select**. You get a warning asking you if you want to “change selection level to Atom”: click **Yes**.

**PCOM** stands for “picking command” (earlier versions used ICOM for “interactive command”—they are the same thing), and it tells you what will happen when you click with the mouse and some modifier key, such as <Shift>, <Ctrl>, or <Alt>. It also depends on which level of the structure you are currently working. There are four levels in this hierarchy, and each level can include many instances of the structures at the next lowest level: in order from highest to lowest, these levels are:

- Molecule - red
- Chain - cyan
- Residue - green
- Atom - yellow

If the **PCOM** Level in your viewer is not set to **Atom**, you will be asked if you want to set the selection level to Atom. Click on **Yes**. You will see **Selected: 127 Atom(s)** with a yellow background in the center of the message-bar at the bottom of the ADT window. Click **Dismiss** to close the **Select From String** widget.

**Note:** if there is no current selection, ADT expands the selection to include all atoms in the viewer. If the userpref, ‘warnOnEmptySelection’ is set to 1, ADT will ask you if it should “expand empty selection to all molecules.” The default behavior is to not ask you if you want the empty selection to be expanded to include every molecule in the viewer. For ADT, make sure to leave the warnOnEmptySelection set to 0.

6. **Edit** → **Delete** → **Delete AtomSet**

If there is a current selection, it is deleted by this command. You will be asked if you really want to do this, because deleting an AtomSet (or a molecule) cannot be undone. Click on **CONTINUE**. The selected oxygens will disappear from the viewer.

**Note:** The added hydrogens are automatically saved as a set “hsg1\_addedH” which you could select with the sequence:

**Select** → **Select a set**,  
**hsg1\_addedH**, **OK**

If you try this, be sure to **clear** **selection** before you go on.

7. **Edit** → **Hydrogens** → **Add**

Choose to add **Polar Only** using Method **noBondOrder** with **yes** to renumbering. Click **OK** to add the polar hydrogens. 330 hydrogen atoms are added to hsg1.

Step 8 is optional because we are going to add charges and atomic solvation parameters to hsg1. However, if you had no further plans to modify this molecule, you would save your modified result at this point:

[8. **File** → **Save** → **Write Pdb**

This opens a file browser that lets you enter the filename. Type in **hsg1.pdb**. You can **Save** or **Cancel**. **Save** opens a **Write Options** widget where you can choose to Sort Nodes and which if any CONECT records to write. Choose **Yes** and leave all the check-buttons off so that no CONECT records are written. Click on **OK** to write the file.]

---

## Exercise Two: Preparing a ligand file for AutoDock.

AutoDock ligands have partial atomic charges for each atom. We also distinguish between aliphatic and aromatic carbons: names for aromatic carbons start with 'A' instead of 'C'. AutoDock ligands are written in files with special keywords recognized by AutoDock. The keywords ROOT, ENDROOT, BRANCH, and ENDBRANCH establish a "torsion tree" object or **torTree** that has a **root** and **branches**. The root is a rigid set of atoms, while the branches are rotatable groups of atoms connected to the rigid root. The keyword TORSDOF signals the number of torsional degrees of freedom in the ligand. The TORSDOF for a ligand is the total number of possible torsions in the ligand minus the number of torsions that only rotate hydrogens. TORSDOF is used in calculating the change in free energy caused by the loss of torsional degrees of freedom upon binding.

You can follow what happens with the ligand more easily if you undisplay the macromolecule first. To do this, click on the check-button labelled  **show/hide molecule** to open the hide show molecule widget. Click on the check-button labelled  **hsg1:ON/OFF** to undisplay hsg1. Close the widget by clicking on the  **show/hide molecule** check-button again.

### Procedure:

1. **Ligand** → **Input Molecule** → **Read Molecule**

Opens a file browser. Click on the **PDBQ files: (\*.pdbq)** menu button to display file type choices and click on **PDB files: (\*.pdb)** files. Choose **ind.pdb**. Click on **Open**.

After the ligand is loaded in the viewer, ADT initializes it. This process involves a number of steps. Then, ADT reports its findings.

- ADT checks for and merges non-polar hydrogens, unless you have set a **userpref** 'adt\_automergeNPHS' not to do so.

**Choose Molecule** and **Rigid Molecule** are discussed in the Appendix.

- ADT detects whether the ligand already has charges or not. If not, ADT determines whether the ligand is a peptide, by checking whether all of its residues' names appear in the standard set of the 20 commonly occurring amino acids. If all the residues are amino acids, ADT adds **Kollman** charges to the ligand.\*\* If not, it computes **Gasteiger** charges; remember that for the Gasteiger calculation to work correctly, the ligand *must* have all hydrogen atoms added, including both polar and non-polar ones. If the charges are all zero, ADT will try to add charges. It checks whether the total charge per residue is an integer. \*\*. Kollman charges are added using a look-up dictionary based on the names of the atoms in the ligand. If the name is not found, a charge of 0.0 is assigned.
- ADT renames planar carbons unless you have set a user preference, '**autotors\_autoCtoA**' not to do so. For peptide ligands, ADT uses a look-up dictionary for planar cyclic carbons (unless you set another userpref, '**autotors\_useProteinAromaticList**', not to do so). For other ligands, ADT determines which are planar cyclic carbons by calculating the angle between adjacent carbons in the ring. If the angle is less than the cut-off of 7.5° (the default value) for all the atoms in the ring, the first letter of the ring carbons' atom names will be renamed "A".

Note: it is also possible to edit which planar, cyclic carbons are renamed 'A' but we are not going to do that in this example. This is discussed in the Appendix. Also you can adjust the aromaticity cut-off if a ring is more warped. See **Ligand** → **Aromatic Carbons** → **Change Aromaticity Criteria** in the Appendix.

## 2. **Ligand** → **Define Rigid Root** → **Automatically**

ADT determines its idea of the best root and marks it with a green sphere.

This best root is the atom in the ligand with the smallest largest subtree. In the case of a tie, if either atom is in a cycle, it is picked to be root. If neither atom is in a cycle, the first found is picked. (If both are in a cycle, the first found is picked). As you might imagine, this can be a slow process for large ligands.

The rigid portion of the molecule includes this root atom *and* all atoms connected to it by non-rotatable bonds (which we will examine in the next section.) You can visualize the current root portion with **Ligand** → **Define Rigid Root** → **Show Root Atoms** (and hide this with **Ligand** → **Show/hide sphere marking root**) However, at this point in our example, the root portion includes only the best

Note: **Add a chain to root** and **Remove a chain from root** are discussed in the Appendix

root atom, atom C11, because all its bonds to other atoms are rotatable.

3. **Ligand** → **Rotatable Bonds** → **Define Rotatable Bonds**

Opens the Torsion Count widget. The widget displays the number of currently active bonds. Bonds which cannot be rotated are colored red. Bonds which could be rotated but are currently marked as inactive are colored purple. Bonds which are currently active are colored green.

Bonds in cycles cannot be rotated. Bonds to leaf atoms cannot be meaningfully rotated. Only single bonds can be rotated (not double or aromatic etc...). ADT determines which bonds could be rotated ('possibleTors'). You set which of these are to be rotatable ('activeTors') by inactivating the others

You can toggle the activity of a bond or group of bonds by picking them in the viewer. Alternatively, buttons on this widget let you toggle the activity of a type of bonds such as 'peptide bonds', 'amide bonds', 'bonds between selected atoms' or 'all rotatable bonds'. One way to do this is to click on **Make all active bonds non-rotatable** then click on **Make all rotatable bonds rotatable**.

Amide bonds should not be rotatable. You must enforce that piece of chemical knowledge by turning off the activity of amide bonds. Do this by clicking on **Make all amide bonds non-rotatable**. You can see that two bonds have been inactivated, the bond between atoms **N2;6** and **C3;4** and that between atoms **C21;26** and **N4;28**. Notice that the current total number of rotatable bonds is **14**.

Before you close this widget with **Done**, leave all the bonds except the two amide bonds active.

4. **Ligand** → **Rotatable Bonds** → **Set Number of Active Torsions**

This brand-new feature allows you to set the total number of active bonds while specifying whether you want active bonds which move the fewest atoms or those which move the most. To see this distinction, set the radiobutton for **fewest atoms**, type '**6**' in the entry and then press <**Enter**> on your keyboard.

Note: This step, using **Set Number of Active Torsions** is optional. We are including it so that your ligand will always have the same torsion tree for this tutorial

ADT will turn off all but six torsions, leaving active the torsions which move the fewest atoms.

Set the radiobutton to **most atoms** and type <Enter> in the entry window. You will see a very different set of 6 rotatable bonds.

For our exercise, **leave the 6 torsions** that move the **fewest** atoms **active**. Click **Dismiss** to close the widget.

5. **Ligand** → **Write PDBQ...**

Opens a file browser allowing you to enter a name. Type in 'ind.out.pdbq' and click **Save**.

You must write a **pdbq** file, which is an AutoDock specific file format, **pdb** augmented by 'q', a charge. Our convention is to name the ligand output files '\*.out.pdbq', but this is not required.

---

## Exercise Three: Preparing the macromolecule file.

The receptor file used by AutoDock must be in **pdbs** format which is pdb plus ‘q’ charge and ‘s’ solvation parameters: **AtVol**, the atomic fragmental volume, and **AtSolPar**, the atomic solvation parameter which are used to calculate the energy contributions of desolvation of the macromolecule by ligand binding.

If the molecule from Exercise One is not still in your viewer, repeat Exercise One. Undisplay the ligand using the show/hide checkbox.

### Procedure:

1. **Grid** → **Macromolecule** → **Choose Macromolecule**.

Choose **hsg1**.

Selecting the macromolecule in this way causes the following sequence of initialization steps to be carried out automatically:

- ADT checks that the molecule has charges. If not, ADT determines whether it is a peptide. If so, ADT adds Kollman charges; if not, it adds gasteiger charges. ADT checks that the total charge per residue is an integer. If not, a list of residues with non-integral charges pops up.\* ADT also adds **solvation parameters**. This process, like that of adding Kollman charges, depends on a look-up dictionary based on the names of the atoms and the name of their parent residues. If a name is not found, 0.0 is assigned for each parameter.
- ADT merges non-polar hydrogens unless the userpref **adt\_automergeNPHS** is set not to do so.
- ADT also determines the types of atoms in the macromolecule. AutoDock can accommodate up to 7 atom types in the macromolecule. It uses a standard set with two customizable types, ‘X’ and ‘M’. If your macromolecule has a non-standard atom type, ADT will prompt you to set up a customizable type X or M for it by entering energy parameters. For example, Zn is not in the standard set. If your macromolecule has Zn, for AutoDock you have to rename the

\* **Note:** the most likely cause of non-integral charges using our tutorial example **hsg1** is that it lacks polar hydrogens (see Exercise One). With other molecules, it is also possible that some atoms are missing from the pdb file. You can repair these missing atoms with a command in the repairCommands module. To do so, first load that module then **Edit** → **Misc** → **Repair Missing Atoms**. Obviously, you have to repeat Exercise One: adding polar hydrogens etc if any residues have been repaired.

'Zn' as 'M' and provide energy coefficients for Zn. 'X' can be used as a second customizable type. It is not possible to have more than 7 types in the macromolecule.

Note: **Grid**  
→ **Macromolecule** → **Read**  
**Macromolecule** and  
**Grid** → **Macromolecule** → **Add**  
**Solvent Parameters** are discussed in the

The macromolecule must be written in a pdbqs file for use by AutoGrid. Since the molecule you chose has been modified by ADT, a file browser opens for you to specify a file name. Type **hsg1.pdbqs** in the input entry and click **Save**.

---

## Exercise Four: Preparing the grid parameter file.

The grid parameter file tells **AutoGrid** the types of maps to compute, the location and extent of those maps and specifies pair-wise potential energy parameters. In general, one map is calculated for each element in the ligand plus an electrostatics map. Self-consistent 12-6 Lennard-Jones energy parameters - **Rij**, equilibrium internuclear separation and **epsij**, energy well depth - are specified for each map based on types of atoms in the macromolecule. If you want to model hydrogen bonding, this is done by specifying 12-10 instead of 12-6 parameters in the gpf.

### Procedure:

1. **Grid** → **Set Map Types**

The types of maps depend on the types of atoms in the ligand. Thus one way to specify the types of maps is by choosing a ligand. If the ligand you formatted in Exercise Two is still in the viewer, choose **Grid** → **Set Map Types** → **Choose Ligand**. If not, use **Grid** → **Set Map Types** → **Read Formatted Ligand File**.

Choosing the ligand opens the AutoGpf Ligand widget that allows you to modify the types of maps to be calculated, and to choose whether to model possible hydrogen bonding. In our example, the ligand has Nitrogen, Oxygen and Hydrogen atoms so we can model N-H and O-H hydrogen bonds but not S-H hydrogen bonds.

Close this widget with the **Accept** button.

2. **Grid** → **Set Grid**

Opens the **Grid Options Widget**. First a brief tour of this widget:

- This has menu buttons at the top: **File**, **Center**, **View** and **Help**.

→ **File**

**Note:** Alternatively, if you plan to use the same macromolecule with a variety of different ligands, you might choose to calculate all the maps you would eventually need via **Set Map Types Directly**. This, along with **Set Params For New Atom Type** and **Set Up Covalent Map**, is discussed in the Appendix.

This menu lets you close the Grid Options Widget, which also causes the grid box to disappear. You can **Close saving current values** to keep your changes or **Close w/out saving** to forget your changes.

→ **Center**

This menu lets you set the center of the grid box in four ways: → **Pick an atom**, → **Center on ligand**, → **Center on macromolecule** or → **On a named atom**.

→ **View**

This menu allows you to change the visibility of the box using **Show box**, and whether it is displayed as lines or faces, using **Show box as lines**. This menu also allows you to show or hide the center marker using **Show center marker** and to adjust its size using **Adjust marker size**.

- The “Grid Options Widget” displays the Current Total Grid Points per map. This tells you how big each grid map will be:  $(n_x + 1) \times (n_y + 1) \times (n_z + 1)$ , where  $n_x$  is the number of grid points in the  $x$ -dimension, *etc.*
- This has 3 **thumbwheel** widgets which let you change the number of points in the  $x$ ,  $y$  and  $z$  dimensions. The default settings are 40, 40, 40 which makes the total number of grid pts per map 68921 because AutoGrid always adds one in each dimension.
- You will also notice it has a **thumbwheel** that lets you adjust the spacing between the grid points.
- There are also entries and thumbwheels that let you change where the center of the grid is.

The **number of points** in each dimension can be adjusted up to 126. AutoGrid requires that the input number of grid points be an even number. It then actually adds one point in each dimension, since AutoGrid and AutoDock need a central grid point.

The **spacing** between grid points can be adjusted with another thumbwheel. The default value is 0.375 Å between grid points, which is about a quarter of the length of a carbon-carbon single bond. Grid

**Note:** clicking with the right mouse button on a **thumbwheel** widget opens a box that allows you to type in the desired value directly. Like many other entry fields in ADT, this updates only when you press <Enter>.

spacing values of up to 1.0 Å can be used when a large volume is to be investigated. If you need grid spacing values larger than this, you can edit the GPF in a text editor later, and before running AutoGrid.

For this exercise:

Adjust the number of points in each dimension to **60**. Notice that each map will have 226,981 points.

Type in **2.5**, **6.5** and **7.5** in the *x* center, *y* center and *z* center entries. This will center the grid box on the active site of the HIV-1 protease, **hsg1**.

Close this widget by clicking **File** → **Close saving current**.

3. **Grid** → **Write GPF**

Opens a file browser allowing the user to specify the name of the grid parameter file. The convention is to use ‘.gpf’ as the extension.

Write the gpf as ‘**hsg1.gpf**’

[4. **Grid** → **Edit GPF**

If you have written a grid parameter file, it opens in an editing window. If not, you can pick one to read in and edit via the **Read** button. If you make any changes to the content of the grid parameter file, you can save the changes via the **Write** button. Edit GPF will open the file we wrote in step 3. Either **OK** or **Cancel** close this widget]

**Grid** → **Set Other Options** and  
**Grid** → **Get values from a GPF** are  
discussed in the Appendix.

---

## Exercise Five: Starting AutoGrid

In general you should know:

- **AutoGrid** (and **AutoDock**) must be run in the directories where the macromolecule, ligand and parameter files are to be found.
- The named files in the parameter file must not include pathnames.
- Currently, it is not possible to run either program on a WINDOWS platform.

### Procedure:

**Run** → **Start AutoGrid**

Opens the Run AutoGrid widget. Here is a brief tour:

- The first two entries in the widget are used to specify which machine to use. By default the local machine is named in the **Macro Name:** entry and in the **Host name:** entry. It is possible to define macros to specify other machines and this is described in the Appendix.
- **Program Pathname:** entry specifies the location of the autogrid3 executable. If it is not in your path, you can use the Browse button to locate it.
- **Parameter Filename:** entry specifies the gpf file. If you have just written a gpf file, opening this widget will automatically load the gpf filename in the **Parameter Filename:** entry. If not, you can use the **Browse** button to the right of the entry to locate the gpf you want to use.
- **Log Filename:** entry specifies the log file. Selecting a gpf creates a possible related name for the glg.

- **Nice Level:** entry used to specify a nice level for remote jobs.
- **Cmd:** entry shows you the command that will be invoked when you click on **Launch**.

1. **Launch**

Starts the AutoGrid job. On non-Linux platforms, this opens a AutodockProcess Manager Widget which allows you to see specifics about current AutoGrid and AutoDock jobs. It is a limited process manager which you can use to terminate an autoxxxx process by selecting its entry. You are asked if you really want to kill it.

**Note:** When you source `/tsri/python/share/bin/initadtcsh`, the directories containing the executables for AutoGrid and AutoDock are added to your path. If you want to start a job from the command line in a different terminal window, you must also source the set-up file in that other window.

**Edit Hosts Dictionary** is discussed in the Appendix.

Please note that you can easily start a job from the command line:

```
% autogrid3 -p hsg1.gpf -l hsg1.glg &
```

---

## Exercise Six: Preparing the docking parameter file.

The docking parameter file tells AutoDock which map files to use, the ligand molecule to move, what its center and number of torsions are, where to start the ligand, which docking algorithm to use and how many runs to do. It usually has the file extension, “.dpf”. Four different docking algorithms are currently available in AutoDock: SA, the original *Monte Carlo* simulated annealing; GA, a traditional Darwinian genetic algorithm; LS, local search; and GA-LS, which is a hybrid genetic algorithm with local search. The GA-LS is also known as a Lamarckian genetic algorithm, or LGA, because children are allowed to inherit the local search adaptations of their parents.

Each search method has its own set of parameters, and these must be set before running the docking experiment itself. These parameters include what kind of random number generator to use, step sizes, *etc.* The most important parameters affect how long each docking will run. In simulated annealing, the number of temperature cycles, the number of accepted moves and the number of rejected moves determine how long a docking will take. In the GA and GA-LS, the number of energy evaluations and the number of generations affect how long a docking will run. ADT lets you change all of these parameters, and others not mentioned here. See **Appendix 2: Docking Parameters**.

### Procedure:

1. **Docking** → **Set Macromolecule** → **Select Macromolecule Filename**

Note: **Docking** → **Set Macromolecule** → **Choose Macromolecule** is discussed in the Appendix.

Select the file you wrote in Exercise Three: **hsg1.pdbqs**. Click **Open**. This doesn't result in a read operation, because AutoDock only needs the filename.

2. **Docking** → **Set Ligand Parameters** → **Choose Ligand**

Choose **ind**. Click **Select Ligand**.

This opens a panel that tells you the name of the current ligand, its atom types, its center, its number of active torsions and its

Note: You can only choose a ligand if you have previously written it to an output file because AutoDock requires the filename of the formatted ligand.

Note: **Read Autotors-Formatted Ligand File** and **Adjust Ligand Parameters** are discussed in the Appendix.

number of torsional degrees of freedom. You can set a specific initial position of the ligand and initial relative dihedral offsets and values for its active torsions. For our exercise we will use the defaults. Click **Close** to close this widget.

3. **Docking** → **Set Search Parameters** → **Genetic Algorithm Parameters**

This lets you change the genetic algorithm specific parameters. It is a good idea to do a trial run with fewer energy evaluations maybe 25 000 evals. For our exercise, we will use the defaults. Click **Close** to continue.

4. **Docking** → **Set Docking Run Parameters**

Here you can choose which random number generator to use, the random number generator seeds, the energy outside the grid, the maximum allowable initial energy, the maximum number of retries, the step size parameters, output format specification and whether or not to do a cluster analysis of the results. For today, use the defaults and just click **Close**.

5. **Docking** → **Write DPF** → **GALS.dpf**

We specify the name of the DPF we are about to write out here. This file will contain docking parameters and instructions for a Genetic Algorithm-Local Search (GA-LS) docking, also known as the Lamarckian Genetic Algorithm (LGA).

Type in **ind.dpf** and click on **Save**.

[6. **Docking** → **Edit DPF**

If you want to look at the contents of the file we just wrote in step 5, use this menu option. You can check that the outputfilename, "**ind.out.pdbq**" appears after the keyword 'move', that ndihe is "6", and torsdof is set to "14 0.3113". You can click either **OK** or **Cancel** to continue.]

**Note:** ADT allows you to change the parameters for any of the four possible docking algorithms at any time. You commit to a specific algorithm only at the Write DPF stage.

---

## **Exercise Seven: Starting AutoDock.**

In general you should know:

- **AutoGrid** and **AutoDock** must be run in the directories where the macromolecule, ligand, gpf and dpf files are to be found.
- The named files in the parameter file must not include pathnames.
- Currently, it is not possible to run either program on a WINDOWS platform

NOTE: To run AutoDock on Linux and Mac OS X machines AND here at Scripps, you must put the following line in your “.cshrc” or “.login” file:

```
limit stacksize unlimited
```

### **Procedure:**

**Run** → **Start AutoDock**

This opens the “Run AutoDock” widget. Here is a brief tour:

- The first two entries in the widget are used to specify which machine to use. By default the local machine is named in the **Macro Name:** entry and in the **Host Name:** entry. It is possible to define macros to specify other machines and this is described in the appendix.
- **Program Pathname:** entry specifies the location of the **autodock3** executable. If it is not in your path, you can use the Browse button to locate it.
- **Parameter Filename:** entry specifies the dpf file. If you have just written a dpf file, opening this widget will automatically load the dpf filename in the **Parameter Filename:** entry. If not, you can use

the **Browse** button to the right of the entry to locate the dpf you want to use.

- **Log Filename:** entry specifies the log file. Selecting a dpf creates a possible related name for the dlg.
- **Nice Level:** entry used to specify a nice level for remote jobs.
- **Cmd:** entry shows you the command that will be invoked when you click on **Launch**.

1. **Launch**

Starts the AutoDock job. On non-Linux platforms, this opens a AutodockProcess Manager Widget which allows you to see specifics about current AutoGrid and AutoDock jobs. It is a limited process manager which you can use to terminate an autodock process by selecting its entry. You are asked if you really want to kill it.

**Note:** When you source `/tsri/python/share/bin/initadtsh`, the directories containing the executables for AutoGrid and AutoDock are added to your path. If you want to start a job from the command line in a different terminal window, you must also source the set-up file in that other window.

**Edit Hosts Dictionary** is discussed in the Appendix.

Please note that you can easily start a job from the command line:

```
% autodock3 -p ind.dpf -l ind.dlg &
```

---

## **Exercise Eight: Analyzing AutoDock Results- Reading Docking Logs**

Reading a docking log or a set of docking logs is the first step in analyzing the results of docking experiments. (By convention, these results files have the extension “.dlg”.)

During its automated docking procedure, AutoDock outputs a detailed record to the file specified after the `-l` parameter. In our example, this log was written to the file ‘ind.dlg’. The output includes many details about the docking which are output as AutoDock parses the input files and reports what it finds. For example, for each AutoGrid map, it reports opening the map file and how many data points it read in. When it parses the input ligand file, it reports building various internal data structures. After the input phase, AutoDock begins the specified number of runs. It reports which run number it is starting; it may report specifics about each generation. After completing the runs, AutoDock begins an analysis phase and records details of that process. At the very end, it reports a summary of the amount of time taken and the words ‘Successful Completion’. The level of output detail is controlled by the parameter “outlev” in the docking parameter file. For dockings using the GA-LS algorithm, outlev 0 is recommended.

The key results in a docking log are the docked structures found at the end of each run, the energies of these docked structures and their similarities to each other. The similarity of docked structures is measured by computing the root-mean-square-deviation, **rmsd**, between the coordinates of the atoms. The docking results consist of the PDBQ of the Cartesian coordinates of the atoms in the docked molecule, along with the state variables that describe this docked conformation and position.

Before starting this exercise, you should undisplay any molecules in the viewer using the **show/hide molecule** check-button.

### **Procedure:**

1. **Analyze** → **Docking Logs** → **Read Docking Log**

First, you need to choose the AutoDock log file you would like to Analyze. This command opens a file browser that lets you choose a file with the extension **.dlg**

Choose **ind.dlg**.

\* If there is a previous **Docking** instance in the viewer, you are asked whether you want to add this dlg to the previous **Docking** instance. This can be done when the same macromolecule, ligand and dpf files were used for both docking experiments. In this case the total number of docked conformations is reported.

Reading a docking log creates a **Docking** instance in the viewer\*. A **Conformation** instance is created for each docked result found in the docking log. A **Conformation** represents a specific state of the ligand and has *either* a particular set of state variables from which all the ligand atoms' coordinates can be computed, *or* the coordinates themselves. **Conformations** also have energies: docked energy, binding energy, and possibly per atom electrostatic and vdw energies. AutoDock computes intermolecular energy, internal energy and torsional energy. The first two of these combined give 'docking energy' while the first and third give 'binding energy.'

ADT reports how many docked conformations were read in from the dlg and tells you to how to visualize the docked conformations or 'states'.

**Delete Docking Log**, **Select Docking Log** and **Read all DLGs** in directory are discussed in the Appendix.

### ***Scripps Research Institute Tutorials Only***

If time permits, we will attempt to cluster all the output files from every computer in the class. Please copy your 'ind.dlg' into the directory we write on the white board (spdir) as 'ind\_XX.dlg' where XX is your machine number. For instance, if you are working on machine 'training15', type this in a terminal window:

```
% cp ind.dlg (spdir)/ind_15.dlg
```

and press <Enter> to continue.

---

## Exercise Nine: Analyzing AutoDock Results- Visualizing Docked Conformations

This exercise lets you visualize the docked conformations of the current **Docking** instance, which was created in the last exercise by reading ind.dlg. The ‘best’ docking result can be considered to be the conformation with the lowest (docked) energy. Alternatively, it can be selected based on its rms deviation from a reference structure.

At the end of each docking run, AutoDock outputs a result which is the lowest energy conformation of the ligand it found during that run. This conformation is a combination of translation, quaternion and torsion angles and is characterized by intermolecular energy, internal energy and torsional energy. The first two of these combined give the ‘**docking energy**’ while the first and third give ‘**binding energy**.’ AutoDock also breaks down the total energy into a vdW energy and an electrostatic energy for each atom.

For this exercise, you may want to hide the macromolecule and input ligand using  **show/hide molecule** and zoom in on the docked ligand using **Shift-Button2**.

### Procedure:

1. **Analyze** → **Conformations** → **Show Conformations**

This command opens a **StatesPlayerWidget** (SPW) for ind.out.pdbq. The SPW has a current list of conformations (its sequence) and a current ID list. These two lists vary depending on the last sequence of menu buttons. The sequence list is all of the docked conformations, ordered by run. The ID list is [0,1,2,3...10]. “0” is reserved for the original, input conformation.

First, a brief tour of the StatesPlayerWidget:

- **Show Conformation List** check-button displays the current list of the current sequence of conformations.

- Arrows at right and left of **state:** entry let you step through the sequence, forwards or backwards.
- **Play Sequence** animates changing the conformation of the ligand through all the docked conformations in the current sequence from the current conformation to the end of the sequence list.
- **Play in Reverse** plays from the current conformation backwards.
- **Stop** resets the conformation of the ligand back to conformation **0**, which is the input conformation.
- **Pause** halts the animation at the current conformation. You can resume play after Pause, either forwards or backwards.
- **Make rms refcoords** makes the current conformation the reference for the rmsd calculation displayed in the upper right corner. By default the input ligand conformation, conformation **0**, is the reference.
- **Build** constructs a new molecule with the current conformation's coordinates (unless you have already added it).

At this point, the sequence linked to the spw is all of the docked conformations from the docking log, ordered by run number. The idList is ['0','1','2','3','4','5','6','7','8','9','10'] (Note: "0" is the first one in this list. This sounds odd at first, but this is because we use Python inside and this has the "0=first" convention.)

Try playing the sequence of conformations, changing the coloring scheme to **vdw** or **elect\_stat**. In the next exercise, we will use the **Build** button to add new molecules to the Viewer.

---

## Exercise Ten: Analyzing AutoDock Results - Clustering Conformations

An AutoDock docking experiment usually has several solutions. The reliability of a docking result depends on the similarity of its final docked conformations. One way to measure the reliability of a result is to compare the rmsd of the lowest energy conformations and their rmsd to one another, to group them into families of similar conformations or “clusters.”

The `dpf` keyword, **analysis**, determines whether clustering is done by AutoDock. As you will see below, it is also possible to cluster conformations with ADT. By default, AutoDock clusters docked results at 0.5Å rmsd. This process involves ordering all of the conformations by docked energy, from lowest to highest. The lowest energy conformation is used as the seed for the first cluster. Next, the second conformation is compared to the first. If it is within the rmsd tolerance, it is added to the first cluster. If not, it becomes the first member of a new cluster. This process is repeated with the rest of the docked results, grouping them into families of similar conformations.

**Note:** lower energies are “better” and in the genetic algorithm, “fitter.”

First we will examine the AutoDock clustering that we read in from `ind.dlg`. Next we will make new clusterings at different rms values.

### Procedure:

1. **Analyze** → **Clusterings** → **Show Clustering**

Opens an instance of a Python object, an **interactive histogram chart** labeled ‘ind\_out\_1:rms = 0.5 clustering’. This chart has bars which represent the clusters computed at the specified rmsd. The bars are sorted by energy of the lowest-energy conformation in that cluster and start off colored blue.

For example, the lowest energy conformation in the second bar is **2\_1**. The height of the bar represents how many conformations are in that cluster. Clicking on a bar makes that cluster the current sequence for the ligand’s **SPW**, and its color changes to red.

Note: If you have read in more than one docking log into the current Docking or if the results did not include clustering, you must **Make Clusterings** before you can show them.

The label in the top left of the SPW widget shows you the rmsd between the current reference and the displayed conformation. As described above in the tour of the SPW, you can set the reference coordinates to that of any of the docked conformations when it is the current conformation. When viewing clustering results, this is especially useful because it allows you to examine the rmsd between cluster members. To do this, choose a cluster and use the arrow key to step forward to its lowest energy conformation, *e.g.* 1-1. Set the rms reference to this conformation. Now, stepping through the cluster will show you the rms difference between the lowest energy member of this cluster, *i.e.* 1-1, and the rest of the conformations in this cluster.

**Note:** if clusterings were performed using several different rms tolerance values, the menu option,

**Analyze** → **Clusterings** → **Show**

**Clustering** would open a widget containing a list of the available rms values. Be sure to click only once and to click delicately on this list to open a new interactive histogram. (Otherwise, you may get several identical windows.)

You can change clusters by picking a different bar in the interactive histogram chart. You can save this histogram as a PostScript file: from the **interactive histogram**'s menu select **Edit** → **Write** to open a file browser for you to enter a filename. Make sure to use “.ps” extension. Select **File** → **Exit** to close.

The active-site of the hiv protease has C2 symmetry. You can probably see evidence of this by examining the clusters of docked indinavir molecules.\* Step 1 is to build a copy of the lowest energy conformation: cluster 1, conformation 1. First display it via the spw, then click the **Build** button. Try clicking on the second bar in the histogram and display the lowest energy member of the second cluster by using the arrow keys next to the entry. If this result doesn't show C2 symmetry, try another cluster bar. You should see the symmetry related docked conformations.

\* **Note:** To facilitate comparing the docked conformations, type

**File** → **Preferences** → **Set**

**Commands to be Applied on**

**Objects** then

select  **colorByMolecules** When this is on, each time a new molecule is added to the viewer (up to a current

## 2. **Analyze** → **Clusterings** → **Make Clusterings**

Opens a widget which lets you enter a series of new **rms tolerances** as floating point number separated by spaces. These will be used to perform new clustering operations on the docked results. The time consuming step in clustering is computing a difference matrix between conformations to be compared. Larger rms values require fewer comparisons; conformations which are more similar require fewer comparisons. If you type a name in the **OutputfileName:** entry, a clustering output file will be written. Our convention is to use the

extension “.clust” for these files. Their format is described below in the Appendix.

It is important to set the ligand to the original, input conformation (numbered 0) before clustering.

Type in a list of RMSD tolerances separated by spaces thus **1.0 2.0 3.0** and click on **OK**. For our example, this should be very fast. You can visualize the new clusterings by repeating Step 1.

---

## **Exercise Eleven: Analyzing AutoDock Results- Visualizing Conformations in Context**

Ultimately, the goal of a docking experiment is to illustrate the docked result in the context of the macromolecule, explaining the docking in terms of the overall energy landscape. The interactions between the ligand and the macromolecule are driven by energy composed of van der Waals(vdW), electrostatic, hydrogen bonding and desolvation component energies.

The first step is to visualize the docked conformations while showing the macromolecule. If hsg1 is still in your viewer, skip Step 1. Instead use **show/hide molecule** to display it. Also, undisplay any docked conformations you may have already built.

### **Procedure:**

1. **Analyze** → **Molecules** → **Show Macromolecule**

This command loads the macromolecule used for the docking experiment. If it is not found in the current directory, a file browser will ask you to specify where it can be found.

Alternatively, you may want to visualize the docked conformations in the context of the energy grids. This may be useful for computer-aided drug design.

2. **Analyze** → **Grids** → **Show Grids Used for Calc**

This opens a list chooser of the grids used in this docking. Select **hsg1.O.map**.

**Note:** The grid isocontours are colored by atom type. (See Exercise One for a list of these colors.)

The AutoGrid map file is read into the viewer, creating an instance of a **Grid**. This map is visualized as an **isocontour** in 3D. This means that every point in the grid box that is equal to the Isocontour level will be connected together by lines or polygons. You can change the isocontour level, which is an energy in Kcal/mol; the step between grid points for sampling the grid values; and whether to show the isocontoured regions

as lines or filled (solid) polygons. You can also toggle the visibility of the Grid and its bounding box.

To illustrate the kind of information you can obtain from the atomic affinity grid maps, try this:

1. Set the IsoValue to **-0.15**; if you type into the slider entry, remember to press **<Return>** or **<Enter>**.
2. Set the Sampling to 1 and press **<Return>** or **<Enter>**.
3. Display **hsg1.pdbqs**; if it is not present in the viewer, use **Analyze** → **Molecules** → **Show Macromolecule**.
4. Choose **Select** → **Select From String** and type in **ASP25** into the Residue field and then click **Select**. Click **Yes** to change selection level and **Dismiss** to close **Select From String** widget.
5. Choose a low-energy docked conformation using the SPW, and hide it using the **show/hide molecule** check-button.
6. Next, **Un/Display** → **Lines** and click the **display only** radio button. If you are asked to change the selection level, then click **Yes**.
7. You can remove the yellow selection highlights by clicking on **Clear Selection**, and then clear all the entry fields in the panel by clicking **Clear Form**.
8. Show the ligand, then once again in the **Select** → **Select From String** panel, type in **IND201** into the Residue field and **O2** into the Atom field. Then click **Select**.

Now you can rotate the objects in the viewer. You will see that the single selected atom in the inhibitor IND201:O2, is buried in a pocket of Oxygen-affinity. If you **Build** (see below) other low-energy docked conformations, you should be able to see the same O2 atom sitting in this region.

**Show Extra Grid** is discussed in the Appendix.

Click **Display Map** and **Show Box** to undisplay the isocontour and its bounding box before you **Dismiss** this panel.

It is maybe useful to visualize all the docked conformations at once by placing spheres, one for each docking, at the center of each docked conformation.

3. **Analyze** → **Molecules** → **Visualize Dockings as Spheres**

This command represents each docked conformation by a sphere. A sphere is placed at the average position of the coordinates of all the atoms in each conformation. Clicking on the name of a docking log in the list makes the spheres representing its results visible only if the associated ligand is visible.

Click on **ind.dlg** in the list. You can change the **radii** of the spheres, their **color** and their smoothness (or “**quality**”).

This command gives you a nice overview of the distribution of the docked results.

---

## ***Files for exercises:***

### **Input Files:**

hsg1.pdb  
ind.pdb

### **Results Files**

#### ***Ligand***

ind.out.pdbq (6 torsions moving fewest atoms)

#### ***Macromolecule***

hsg1.pdb  
hsg1.pdbqs

#### ***AutoGrid***

hsg1.gpf  
hsg1.glg  
hsg1.\*.map  
hsg1.maps.fld,hsg1.maps.xyz

#### ***AutoDock***

hsg1.dpf  
ind.dlg

### **Useful Scripts**

extract.py (file.dlg out - pares down dlg)  
extractDir.py (for all dlgs in this dir)  
submit.py (launch many dockings on a queue)  
recluster.py (cluster dlgs)

### **Customization Options for ADT**

adt\_automergeNPHS: default is 1  
adt\_autoCtoA: default is 1  
adt\_editHISprotonation: default is 'No Change'  
autotors\_userProteinAromaticList

---

## Appendix 1

**Ligand** → **Input Molecule** → **Read Molecule**

Opens a file browser. Clicking on **PDBQ files: (\*.pdbq)** menubutton displays file type choices which include pdbq, pdb, and mol2. Clicking on a file in the file browser selects that file as the ligand and loads it into the viewer. After the ligand is loaded in the viewer, ADT initializes it. This process involves a number of steps.

- ADT checks for and merges non-polar hydrogens, unless a **userpref adt\_automergeNPHS** has been set not to do so, i.e. set to 0.
- ADT detects whether the ligand already has charges or not. If not, ADT determines whether the ligand is a peptide (by checking whether all of its residues' names appear in the standard set). If so, ADT adds **Kollman** charges to the ligand.\*\* If not, it adds **gasteiger** charges. (If the charges are all zero, ADT will try to add charges). It checks whether the total charge per residue is an integer. (If not, a list of residues with charge errors appears in a msg box.)
- ADT renames planar carbons unless a user preference **autotors\_autoCtoA** has been set not to do so. For peptide ligands, ADT uses a look-up dictionary for planar cyclic carbons (unless the userpref **autotors\_useProteinAromaticList** is set to 0). For other ligands, ADT determines which are planar cyclic carbons by calculating the angle between adjacent carbons in the ring. If the angle is less than 7.5 degrees for all the atoms in the cycle, the carbons are renamed A... This cut-off can be adjusted.

\*\* . Kollman charges are added using a look-up dictionary based on the names of the atoms in the ligand. If the name is not found, a charge of 0.0 is assigned.

**Ligand** → **Input Molecule** → **Choose Molecule**

Opens a list chooser showing names of all molecules present in the viewer. Clicking on the name of the ligand molecule sets it to be the ligand. After the ligand is chosen, ADT initializes it as described above.

**Ligand**->**Input Molecule**->**Rigid Molecule**

Opens a file browser. The user chooses the molecule and clicks **Open**. Please note, this command does NOT load the molecule into the viewer. Instead this process involves simply adding 4 lines to the pdbq file so that it is an acceptable ligand file for AutoDock. This is particularly useful for protein-protein docking. In the case of a protein ligand, detecting possible torsions is very time consuming. Moreover, it is pointless to detect possible torsions if the user intends to turn off all torsions. The 4 lines include a line setting the active torsions to 0, a line marking the beginning of ROOT portion of the molecule which includes all the atoms in the molecule, a line marking the end of the ROOT and finally a line setting TORSDOF to 0.

**Ligand**->**Define Rigid Root**->**By Picking**

Makes picking a root atom the command currently bound to picking in the viewer.

The first atom picked by the user becomes the root atom and picking in the viewer is restored to what it was previously. To pick a different root, follow the same sequence.

**Ligand**->**Define Rigid Root**->**Automatically**

ADT determines its idea of the best root and marks it with a green sphere.

This best root is an atom in the middle of the ligand, the atom with the smallest 'largest subtree.' In the case of a tie, if either atom is in a cycle, it is picked to be root. If neither atom is in a cycle, the first found is picked. (If both are in a cycle, the first found is picked). As the user might imagine, this can be a slow process for large ligands.

**Ligand**->**Define Rigid Root**->**Add a chain to root**

Allows the user to pick an entire chain to be in the root portion of the molecule.

**Ligand**->**Define Rigid Root**->**Remove a chain from root**

Allows the user to remove an entire chain from the root portion of the molecule.

**Ligand**->**Define Rigid Root**->**Show Root Atoms**

The rigid portion of the molecule includes the root atom AND all atoms connected to it by non-rotatable bonds. This command toggles the visibility of small green spheres marking atoms in the expanded root portion of the molecule.

**Ligand**->**Rotatable Bonds**->**Define Rotatable Bonds**

Opens the **Torsion Count** widget. The Torsion Count widget displays the number of currently active bonds. The user can toggle the activity of a possibly rotatable bond or group of possibly rotatable bonds by picking them in the viewer.

Bonds in cycles cannot be rotated. Bonds to leaf atoms cannot be meaningfully rotated. Only single bonds can be rotated (not double or aromatic etc...). ADT determines which bonds could be rotated ('possibleTors'). The user sets which of these are to be rotatable ('activeTors') by inactivating the others. Bonds which cannot be rotated are colored red. Bonds which could be rotated but are currently marked as inactive are colored purple. Bonds which are currently active are colored green.

**Ligand**->**Rotatable Bonds**->**Set Number of Active Torsions**

This feature allows the user to set the total number of active bonds while specifying whether he wants to activate/deactivate the bonds which move the fewest atoms or those which move the least. Please note a ROOT atom must be specified **before** using this command because the number of atoms moved by a rotatable bond depends on the torsion tree specified by the current root.

**Ligand**->**Aromatic Carbons**->**Rename Aromatic Carbons**

Changes the first character of the names of the set of carbon atoms which were found to be aromatic from **C** to **A**. This happens automatically when ADT initializes a ligand molecule unless the userpref **autotors\_autoCtoA** is set to 0.

**Ligand**->**Aromatic Carbons**->**Restore Aliphatic Carbons**

Changes the first character of the set of carbon atoms which were found to be aromatic from **A** to **C**.

**Ligand**->**Aromatic Carbons**->**Set Carbon Names**

Opens a STOP widget and makes changing the names of carbons the current ICOM in the viewer. The user can tell which command is the current ICOM by examining the **ICOM:** entry in the third menu bar. It should say **ADtors\_setCarbonNames**. This means that currently picking in the viewer invokes ADtors\_setCarbonNames. While this command is active, picking on a carbon atom changes it from being considered aromatic to being considered aliphatic (ie changes its name from A\* to C\*) and the converse, picking on an aliphatic carbon changes it to aromatic (ie changes its name from C\* to A\*). Please note that AutoDock considers carbons in rings which meet the flatness criteria as aromatic.

ADT renames carbons in rings only if all the carbons in the ring meet the flatness criteria.

NB: Until recently ADT considered fused-rings as 1 ring. If a ring which met the flatness criteria was fused to one which is not, ADT did not rename the carbons of the flat ring. In this case, the user had to change the names of the carbons in the flat portion of the fused ring via this command. THIS HAS BEEN FIXED IN THE LATEST ADT VERSION.

**Ligand**->**Aromatic Carbons**->**Change Aromaticity Criteria**

Opens an entry widget which lets the user change the aromaticity criteria from its current value to a new value. The initial default value is 7.5 degrees. This is the maximum allowable angle between normals to adjacent atoms in a ring. The ring is judged flat and the carbon atoms aromatic if the angles between all pairs of adjacent atoms in the ring is less than or equal to the aromaticity criteria.

**Ligand**->**Write PDBQ ...**

Opens a file browser allowing the user to enter a name. The user must specify a **pdbq** filename, which is an AutoDock specific file format: **pdb** augmented by 'q', a charge. Our convention is to name the ligand output files '\*.out.pdbq', but this is not required.

**Ligand**->**Show/hide sphere marking root**

Allows the user to toggle the visibility of the sphere used to mark the root atom and the smaller spheres which mark all atoms adjacent to the root atom by non-rotatable bonds. Together the root atom and these adjacent atoms constitute the root portion of the ligand molecule.

**Ligand**->**Automatic autotors setup**

Allows the user to set up the ligand with one command. The command initializes the ligand which includes adding charges, merging non-polar hydrogens and lone pairs and detecting and renaming aromatic carbons. It sets the root automatically. It turns off amide and peptide-backbone bonds. Finally, it writes an output file.

**Grid**->**Macromolecule**->**Read Macromolecule**

Opens a file browser allowing the user to select the macromolecule for the docking experiment. The selected file is read into the viewer and macromolecule is initialized as described below.

**Grid**->**Macromolecule**->**Choose Macromolecule**

Opens a list chooser allowing the user to choose the name of the macromolecule from the list of molecules present in the viewer.

Selecting the macromolecule causes this process of initialization:

- ADT checks that the molecule has charges. If not, ADT determines whether it is a peptide. If so, ADT adds Kollman charges; if not, it adds gasteiger charges. ADT checks that the total charge per residue is an integer.

- ADT also adds **solvation parameters AtVol and AtSolPar**. This process, like that of adding Kollman charges, depends on a look-up dictionary based on the names of the atoms and the names of their parent residues. If a name is not found, 0.0 is assigned for each parameter.
- ADT merges non-polar hydrogens unless the userpref **adt\_automergeNPHS** is set not to do so.
- ADT also determines the types of atoms in the macromolecule. AutoDock can accommodate up to 7 atom types in the macromolecule. It uses a standard set with two customizable types, 'X' and 'M'. If your macromolecule has a non-standard atom type, ADT will prompt you to set up a customizable type X or M for it by entering energy parameters. For example, Zn is not in the standard set. If the macromolecule has Zn, for AutoDock the user must rename the 'Zn' as 'M' and provide energy coefficients for Zn. 'X' can be used as a second customizable type. It is not possible to have more than 7 types in the macromolecule.

The macromolecule must be written in a pdbqs file for use by AutoGrid.

**Grid->Macromolecule->Add Solvent Parameters**

Opens a dialog box asking whether the macromolecule is already in the viewer. If so, a list chooser lets the user select its name. If not, a file browser lets the user select the filename. After the molecule is selected, solvation parameters are added based on a dictionary look-up. If the residue name-atom name string is not in the keys of the dictionary, the values 0.0 0.0 are assigned for AtVol and AtSolPar. The macromolecule must be written in a pdbqs file for use by AutoGrid. A file browser opens for the user to enter the filename. The user can choose to cancel here, but he will be warned that AutoGrid requires a written pdbqs file.

Please Note: If the user cancels at this step, the chosen molecule's name won't be written in gpf file.

**Grid->Set Map Types->Set Map Types Directly**

The types of maps generated by AutoGrid depend on the types of atoms in the ligand molecule. It is possible to use one set of grids for many different ligand molecules. One way to do this is to calculate a set of grids for all probable ligand atom types. This command lets the user enter a list of atom types for the grid calculation.

**Grid->Set Map Types->By Choosing Ligand**

Another way to determine which grids to calculate is by choosing a ligand. This command sets the types of maps to be generated by AutoGrid to the types of atoms present in the chosen ligand molecule.

**Grid->Set Map Types->By Reading Formatted File**

The final way to determine which grids to calculate is by reading in a formatted-ligand file. This command sets the types of maps to be generated by AutoGrid to the types of atoms present in the ligand molecule

**Grid->Set Map Types->Set Params For New Atom Type**

The standard set of atom types for the macromolecule are CNOSH. If the macromolecule has atoms of other types, the user has to set Rij and epij energy parameters for the unusual types. This command prompts the user through this process.

**Grid->Set Map Types->Set Up Covalent Map**

Opens the Covalent Grid Parameters widget in which the user must specify the energy barrier height, the half-width in Angstrom and the attachment atom for a covalent grid map.

**Grid** -> **Set Grid**

Opens the **GridOptions Widget**. First a brief tour of this widget:

- It has menubuttons at the top: **File**, **Center**, **View** and **Help**.

-> **File**

Lets the user close the Grid Options Widget, which also causes the grid box to disappear. The user can **Close saving current values** or **Close w/out saving**

-> **Center**

The center of the grid box can be set four ways: -> **Pick an atom**, -> **Center on ligand**, -> **Center on macromolecule** or -> **On a named atom**.

-> **View**

Allows the user to change the visibility of the box **Show box** and whether it is displayed as lines or faces **Show box as lines**. Also allows the user to change the visibility of the center marker **Show center marker** and to adjust its size **Adjust marker size**.

- Grid Options Widget displays the Current Total Grid Points per map which tells you how big each grid map will be:  $(nxpts+1)*(nypts+1)*(nzpts+1)$
- It has 3 **thumbwheel**\*\* widgets which let the user change the number of points in the x, y and z dimensions. The default settings are 40, 40, 40. ADT attempts to center the box on the ligand. It adjusts the size of the box to fit the ligand.

- It has a thumbwheel which lets you adjust the spacing between the grid points.
- It also has entries and thumbwheels which let you offset the center of the grid.

\*\*Please note that right-clicking on a thumbwheel widget opens a control box allowing the user access to various options including one which lets the user type in the desired value. Like many other widgets in ADT, this widget updates its value to the typed entry **only** when the user types 'Enter'.

The **number of points** in each dimension can be adjusted up to 126. AutoGrid requires that the each specified number of grid points be an even number. It adds one point in each dimension and requires that the total then be an odd number.

The **spacing** between grid points can be adjusted with another thumbwheel. By default this value is .375 Angstrom between grid points, which is about a quarter of the length of a carbon-carbon single bond. Spacings of up to 1.0 Angstrom can be used when a large volume is to be investigated.

**Grid**->**Set Other Options**

Opens the Autogpf Options widget which lets the user: (1)change the smooth factor, (2) tell AutoGrid whether or not to calculate a floating point map and (3)set what dielectric constant to be used. The smooth factor, the radius within which to store minimum energy, is 0.5 by default and should not be changed. Generally, floating point maps are not necessary. The default is to use distance dependent dielectric, but the user may specify a constant dielectric in this widget.

**Grid**->**Write GPF**

Opens a file browser allowing the user to specify the name of the grid parameter file. The convention is to use '.gpf' as the extension.

**Grid->Edit GPF**

If the user has written a grid parameter file, it opens in an editing window. If not, the user can pick one to read in and edit via the **Read** button. If he makes any changes to the content of the grid parameter file, the user can save the changes via the **Write** button. Either **OK** or **Cancel** closes this widget but neither writes a file.

**Grid->Get values from a GPF**

Opens a file browser allowing the user to specify the name of the grid parameter file. The specific parameter values in the gpf replace the general default values. Please note that this command does not set the receptor or ligand but does set other parameters such as the center of the grid and the number of points in each dimension.

**Docking->Get values from a DPF**

Opens a file browser allowing the user to specify the name of the docking parameter file. The specific parameter values in the dpf replace the general default values. Please note that this command sets the receptor\_stem field but does not set the receptor\_filename or ligand\_filename.

**Docking->Set Macromolecule->Choose Macromolecule**

Opens a list chooser allowing the user to choose from a list of molecules present in the viewer. The file from which the chosen molecule was read becomes the stem of certain keys words in the docking parameter file including 'map' and 'fld'.

**Docking** -> **Set Macromolecule** -> **Select Macromolecule Filename**

Opens a file browser allowing the user to select the macromolecule filename. Unlike other file browser choices, choosing in this file browser doesn't result in the molecule being added to the viewer. Instead, as described in the last entry, the filename becomes the stem of certain key words in the docking parameter file including 'map' and 'fld'.

**Docking** -> **Set Ligand Parameters** -> **Choose Ligand**

Opens a list chooser allowing the user to choose from a list of molecules present in the viewer. Opens the AutoDpf Ligand Parameters widget which tells you the name of the current ligand, its atom types, its center, its number of active torsions, and its torsdof. This widget lets the user set a specific initial position of the ligand and initial relative dihedral offsets and values for its active torsions. The user can only choose a ligand if it was read in from an autotors-formatted file or if in this session of ADT it was previously written to an outputfile. This is required because AutoDock must have the filename of the formatted ligand.

**Docking** -> **Set Ligand Parameters** -> **Read Autotors-Formatted Ligand File**

Opens a file browser allowing the user to select a ligand filename. This file must be the result of formatting a ligand with ADT. The file is read and the ligand added to the viewer. Then the AutoDpf Ligand Parameters widget opens displaying the name of the current ligand, its atom types, its center, its number of active torsions, and its torsdof. This widget lets the user set a specific initial position of the ligand and initial relative dihedral offsets for its active torsions.

**Docking** -> **Set Ligand Parameters** -> **Adjust Ligand Parameters**

Reopens the AutoDpf Ligand Parameters widget for the current ligand. This command may be useful if the user wants to review or change the ligand's current information.

## Docking Parameters are defined in Appendix 2

**Docking** -> **Set Search Parameters** -> **Simulated Annealing Parameters**

The Simulated Annealing Parameters widget lets the user adjust these values from their defaults (which are shown with the corresponding AutoDock key word in parentheses here):

Number of:

Runs(**runs** 10)

Cycles(**cycles** 50)

Accepted steps per cycle(**accs** 100)

Rejected steps per cycle(**rejs** 100)

To Begin the next cycle, use:

minimum state(**select m**) ON or last state(**select l**) OFF

Reduction schedule type:

Linear (**linear\_schedule**) ON Geometric  
(**#linear\_schedule**) OFF;

Reduction Factors per cycle for:

Translation(**trnrf** 1.0) Quaternion(**quarf** 1.0)

Dihedral(**dihrf** 1.0) Temperature(**rtrf** 0.95)

Initial Temperature(**rt0** 1000.)

## Docking Parameters are defined in Appendix 2

**Docking** -> **Set Search Parameters** -> **Genetic Algorithm Parameters**

The Genetic Algorithm Parameters widget lets the user adjust these values from their defaults (which are shown with the corresponding AutoDock key word in parentheses here):

Number of GA Runs(**ga\_run** 10)

Population Size(**ga\_pop\_size** 50)

Maximum Number of energy evaluations(**ga\_num\_evals** 2500000)

Maximum Number of generations(**ga\_num\_generations** 27000)

Number of top individuals who automatically survive(**ga\_elitism** 1)

Rate of Gene Mutation(**ga\_mutation\_rate** 0.02)

Rate of Crossover(**ga\_crossover\_rate** 0.8)

Mean of Cauchy distribution for gene mutation(**ga\_cauchy\_alpha** 0.0)

Variance of Cauchy distribution for gene mutation(**ga\_cauchy\_beta** 1.0)

Number of generations for picking worst individual(**ga\_window\_size** 10)

## **Docking Parameters are defined in Appendix 2**

**Docking** -> **Set Search Parameters** -> **Local Search Parameters**

The Local Search Parameters widget lets the user adjust these values from their defaults (which are shown with the corresponding AutoDock key word in parentheses here):

Number of LS Runs(**do\_local\_only** 50)

Maximum Number of iterations(**sw\_max\_its** 300)

Maximum Number of successes in a row before changing  
rho(**sw\_max\_succ** 4)

Maximum Number of failures in a row before changing  
rho(**sw\_max\_fail** 4)

Solis&Wets parameter defining initial variance and size of  
local space to

sample(**sw\_rho** 1.0)

Lower bound on rho(**sw\_lb\_rho** 0.01)

Probability of any particular phenotype being subjected to local  
search

(**ls\_search\_freq** 0.06)

For Local Search, Use:

Solis&Wets with uniform variance(**set\_sw1**) OFF or

pseudo-Solis&Wets with relative  
variance(**set\_psw1**) ON

## **Docking Parameters are defined in Appendix 2**

**Docking** -> **Docking Run Parameters**

The Set Docking Run Options widget lets the user adjust these  
values from their defaults (which are shown with the corresponding  
AutoDock key word in parentheses here):

**RANDOM NUMBER SEEDS:**

Built-in library(OFF) or the Platform-independent  
Library from the

University of Texas(ON)

Select Two Random Numbers Generator Seeds:

time(ON) pid(ON) user defined(OFF).

Please note, the seeds must be different so neither time nor pid can be selected for both choices. If the user elects to specify his own seeds, they must be different.

#### ENERGY PARAMETERS:

External Grid Energy(**extnrg** 1000.0)

Maximum allowable initial energy(**e0max** 0.0)

Maximum Number of Retries(**e0max** 10000)

Please note, e0max is specific to the SA algorithm and requires two parameters

Calculate internal electrostatic energy(**intelec**) OFF

#### STEP SIZE PARAMETERS:

Translation(Angstrom/step)(**trnrf** 2.0) Please note you can enter values for the first and last cycles and AutoDock will calculate the required translation step size. Quaternion(**quarf** 50.0)

Torsion(**dihrf** 50.0)

#### OUTPUT FORMAT PARAMETERS:

Level of detail for output(**outlev** 1.0)

Rms cluster tolerance(**rmstol** 0.5)

Reference structure file for RMS calc:(**rmsref** )

Perform cluster analysis(**analysis**) OFF

Write all conformations in a cluster(**write\_all\_cluster\_members**) OFF

The next 5 commands write docking parameter files with appropriate parameters. Each opens a file browser allowing the user to specify the filename for the docking parameter file and writes appropriate parameters with their current values.

Docking->Write DPF->SA.dpf

Docking->Write DPF->GA.dpf

Docking->Write DPF->LS.dpf

Docking->Write DPF->GALS.dpf

Docking->Write DPF->Cluster.dpf

Docking->Edit DPF

If the user has written a docking parameter file, it opens in an editing window. If not, he can pick one to read in and edit via the **Read** button. If he makes any changes to the content of the docking parameter file, he can save the changes via the **Write** button. Either **OK** or **Cancel** close this widget but neither writes a file.

**Run**-> **Start AutoGrid** and **Run**->**Start AutoDock** have the same structure and so are described together here. **xpf** refers to the parameter file and **xlg** to the log file. autoxxxx refers to either autogrid or autodock. Either sequence opens the Run AutoXxxx widget. A brief tour:

- The first two entries in the widget are used to specify which machine to use. By default the local machine is named in the **Macro Name:** entry and in the **Host name:** entry. (FYI. It is possible to define macros to specify other

machines and other executables such as autodock4.0. This is described in the next section.)

- **Program Pathname:** entry specifies the location of the autoxxxx3 executable. If it is not in your path, you can use the **Browse** button to locate it.
- **Parameter Filename:** entry specifies the xpf file. If you have just written a xpf file, opening this widget will automatically load the xpf filename in the **Parameter Filename:** entry. If not, you can use the **Browse** button to the right of the entry to locate the xpf you want to use. Click **Enter** in this entry to automatically suggest a log filename.
- **Log Filename:** entry specifies the log file. Usually xlg filename is based on the xpf filename.
- **Nice Level:** entry used to specify a nice level for remote jobs.
- **Cmd:** entry shows you the command which will be invoked when you click **Launch**.

1. **Launch**

Starts the AutoXxxx job. Opens **AutodockProcess Manager** Widget described in the next section.

**Run**->**Process Manager**

Opens **AutodockProcess Manager** Widget which allows the user to see specifics about current AutoGrid and AutoDock jobs. It is a limited process manager which the user can use to terminate an autoxxxx process by selecting its entry. The user is asked if he really wants to kill it. Please note this may not be available on all flavors of Unix.

**Run**->**Edit Hosts Dictionary**

Opens **Add Host Manager** Widget which allows the user to see a list of macros currently defined, select and edit or delete one of them or define a new macro. An adtHost macro has a name, a string identifier and values for the following fields: host name which is the name of the computer to use, the Autogrid Program Pathname, the Autodock Program Pathname, the Queue Type which can be 'int' for interactive or 'nqe' for machine supporting nqe and a flag indicating whether the macro was defined in the user's .adtrc or not. By default the current machine is always present as a macro of its own name which is not from the user's .adtrc. The **Add** button adds the defined macro to the current session's adtHosts dictionary. The **Write** button adds code defining the macro to the user's .adtrc. Please note that **Delete** removes the macro from the current session but does not modify the written file. Also please note that the pbs option is not implemented.

**Analyze**->**Docking Logs**->**Read Docking Log**

Opens a file browser allowing the user to select a docking log file with extension .dlg. This file is parsed and a Docking Log Object is created. If there is a previous **Docking** instance in the viewer, the user is asked whether he wants to add this dlo to the previous Docking instance. This can be done when the same macromolecule, ligand and dpf files were used for both docking experiments. In this case the total number of docked conformations is reported. A **Conformation** instance is created for each docked result found in the docking log. A **Conformation** represents a specific state of the ligand and has either a particular set of state variables from which coordinates can be computed or the coordinates themselves. Conformations have energies: docked energy, binding energy, and possibly per atom electrostatic and vdw energies.

ADT reports how many docked conformations were read from the dlg and tells you to how to visualize the docked conformations.

**Analyze->Docking Logs->Delete Docking Log**

Opens a list chooser allowing the user to select a Docking to delete. Deleting a Docking removes its ligand molecule from the viewer.

**Analyze->Docking Logs->Select Docking Log**

Opens a list chooser allowing the user to select a Docking to be the current Docking. Selecting a Docking links its items to menu buttons under Analyze such as Molecule-> Show Macromolecule, Results-> Show Chart etc.

**Analyze->Docking Logs->Read all DLGs in directory**

Opens a file browser allowing the user to select a docking log file with extension .dlg. All the files in the directory of the selected file are read into a single Docking.

**Analyze->Molecules->Show Macromolecule**

Displays current Docking's macromolecule. If it is not present in the viewer it is read in from a file in the current directory. If its file is not in the current directory, a file browser opens allowing the user to specify its location.

**Analyze->Molecules->Eval Molecule in Grids**

Opens a file browser allowing the user to select a docking parameter file. Then a second file browser opens, allowing the user to select a file to be evaluated in the grids referred to in the dpf. Via a pipe, Autodock is started in the command mode and the command

**epdb filename** is invoked. ADT parses the result, loads the molecule and attaches the per-atom electrostatic, vdw and total energies to each atom. The atoms in the molecule are colored by vdw\_energy.

**Analyze**->**Molecules**->**Choose a Docked Conformation**

Opens a list chooser allowing the user to select a docked conformation of the current Docking. Selecting a conformation displays information about its docked energies in the top of the list chooser. Double clicking on the conformation changes the coordinates of the atoms of the ligand to those of that docked structure. It is possible to write a pdbq file with the current coordinates via File->Save.

**Analyze**->**Results**->**Show Chart**

Opens a widget showing results of ligands vs macromolecules dockings. This is useful if the same ligand has been docked against many macromolecules and conversely.

**Analyze**->**Results**->**Get Output**

Opens a text widget showing the output lines detailing the Clustering Histogram Results. Please note this is not available for a Docking formed from many dlg files.

**Analyze**->**Results**->**Show Histogram**

Opens a widget showing a primitive histogram corresponding to the clustering found in a dlg. Each column represents a cluster and each box within the column a specific docked result. Clicking on the box sets the ligand to the corresponding conformation. Middle-button clicking displays information about the docked result. Please note this is not available for Dockings formed from multiple dlgs.

**Analyze** -> **Grids** -> **Show Grids Used For Calc**

Opens a list chooser of the grids used in the current Docking. Selecting a file causes it to be read into the viewer, creating an instance of a Grid visualized as an **isocontour**.

A Visualize AutoGrids: <filename> widget opens, allowing the user to change the isocontour interval, the step between grid points and the representation as lines or fill. The user can toggle the visibility of the Grid and its bounding box.

**Analyze** -> **Grids** -> **Show Extra Grid**

Opens a file browser allowing the user to select a grid besides the grids used in the current Docking. Selecting a file causes it to be read into the viewer, creating an instance of a Grid visualized as an **isocontour**. This may be useful for some purposes of comparison.

**Analyze** -> **Conformations** -> **Show Conformations**

If there are more than one molecule with conformations in the viewer, this opens a list chooser allowing the user to select which molecule's conformations to visualize. Selecting a molecule from this list chooser opens its StatePlayerWidget (**spw**) which lets the user step through the different conformations starting at conformation **0** which is the original input ligand. . (If there is only one molecule with conformations, its spw opens immediately). For this command, the sequence displayed by the molecule's spw is the sequence of docked conformations found in the docking log which are identified by run number, eg 1, 2, 3....

The current conformation can be changed by clicking on the **arrows** at the end of the entry which shows the current conformation's id. This sequence can be played forward or backward from the current conformation. The **Stop** button returns the ligand to its original coordinates. **Pause** stops the play process. **Build** adds a new molecule

with the current conformation's coordinates and its name. The **rms** label at the top displays the rms between the current reference conformation (which is initially 0 but can be set to any conformation via the **Make rms refcoords** button) and the current conformation. The **color by** widget lets the user color the current conformation according to each atom's element '**atom**', or van der Waal energy for docked conformations '**vdw**', or electrostatic energy '**elec\_stat**' or the sum of these two, '**total**'. Please note that a blue to red color ramp is used when coloring by individual atom energy where blue is lowest energy and red highest. Color by **molecule** can be helpful if many docked conformations have been built. The label at the top of the widget displays the current conformation's **binding** energy which is the final intermolecular energy plus the ligand's torsional free energy as well as displaying the conformation's **docked** energy which is the final intermolecular energy plus the ligand's final internal energy. The user can also display a list of ids and change the current conformation by double clicking on entries in this list (**Show IdList** button). **Close** closes this widget, leaving the ligand in the current conformation.

**Analyze**->**Conformations**->**Show Energy HISTOGRAM**

Opens a list chooser allowing the user to select which molecule's conformations to visualize via an energy histogram if there are more than one molecule with conformations in the viewer. Selecting a molecule from this list chooser opens a widget which lets the user specify how many energy bins, the energy minimum and the energy maximum for the energy histogram. Build histogram builds the histogram and opens an interactive histogram chart displaying it. The x-axis in this histogram is the docking energy of the lowest energy in the bin and its height the number of conformations in this bin. Clicking on a bar makes that energy bin the current sequence for the molecule's spw and opens the spw widget. The Show Conformation List shows the run numbers of the docked results in the current energy bin. The rest of the buttons are described in the preceding description. Please note that the interactive histogram chart can be saved as a postscript file via its Edit->Write command and that it is closed via its File->Exit command.

**Analyze**->**Conformations**->**Read Conformations From File**

Opens a file browser allowing the user to select the docking parameter file for the docking result. (Please note it is necessary to read the docking parameter file because it contains the about keyword which is not present in the result file and which is required for setting the conformation from the state variables.) After selecting the dpf, the user is presented with another file browser allowing him to select the result file. If there is already a Docking in the viewer, the user is asked if he wants to add these results to the previous Docking. This is appropriate to do if the same molecules and dpf have been used. If the ligand specified by the 'move' keyword is not in the current directory, a file browser opens so the user can indicate its location. Reading conformations from file makes their Docking the viewer's current Docking.

**Analyze**->**Conformations**->**Write Result File**

Opens a file browser allowing the user to specify a filename. If there is only one Docking in the viewer, its docked conformations are written, one per line, to this file in the following format:

An invariant string (required to conform to Entropia results format):

```
17 18 19 1/23/2001 7:27:42 AM 1/23/2001 7:27:24 AM
1.00 3.00 3.05
```

which are place holders for: output\_id, data\_run\_id, dpf\_id, creation\_dtime, last\_update\_dtime, ei\_version, ag\_version, ad\_version,

Followed by a variant string:

```
%d %d %d %d %d %d %d %f %f %f 3%f 4%f %d n%f
```

whose values are: run\_rank, run\_number, cluster\_rank, cluster\_size, run\_size, rseed1, rseed2, rmsd, binding\_energy, docking\_energy, translation, quaternion, number of torsions followed by nvalues, one per torsion.

**Analyze**->**Clusterings**->**Show Clustering**

Opens an interactive histogram chart displaying the clusters in the current clustering. (If there is more than one molecule with conformations in the viewer, this opens a list chooser allowing the user to select which molecule's clusterings to visualize. If the selected molecule has clusterings both on binding energy and on docking energy, a list chooser opens allowing the user to specify which clusterer's results to show. If selected clusterer has clusterings at more than one rms, a list chooser opens allowing the user to specify which rms clustering to display. Any of these choices having only one possibility are skipped). Selecting a bar in the interactive histogram makes the sequence of conformations in that cluster the current sequence for the molecule's spw. The corresponding idList is formed of cluster-rank\_cluster-number. For instance, 1-1 is the id of the first ranked cluster's first member and 1-2 the id of its second member. Clusters are ranked from lowest energy to highest. Within a cluster, conformations increase in energy from the first conformation. The spw provides various ways to interact with the current sequence which are described in detail under **Analyze->Conformations->Show Conformations** above. Please note that the interactive histogram chart can be saved as a postscript file via its Edit->Write command and that it is closed via its File->Exit command.

**Analyze->Clusterings->Make Clusterings**

Opens a form allowing the user to enter a list of rms tolerances for the clustering, allowing him to choose whether to cluster on docking energy or on binding energy and to enter an optional filename for a written output file. The clustering is done on the current Docking. If it already has a clusterer of the specified energy type, that clusterer is used to perform clusterings at each of the specified rms tolerances. If not, a new clusterer is created and then the clusterings are performed. New clusterers are added to the Docking's clusterer\_dict under the key of the energy type ('docking' or 'binding'). If an output filename is specified, the clusterer writes its data to that file.

**Analyze->Clusterings->Make Clusterings on Subset**

Opens a form allowing the user to choose a set from the previously saved subsets of the ligand molecule AND to enter a list of rms tolerances for the clustering, allowing him to choose whether to cluster on docking energy or on binding energy and to enter an optional filename for a written output file. The user must have previously selected a subset of the atoms in the ligand molecule and saved it via the Select->Save Set command. The clustering is done on the specified subset of atoms of the current Docking. A new clusterer of the specified type is created and the clusterings are performed at the specified rms tolerances. The new clusterer is added to the Docking's clusterer\_dict under the key built from the subset's name and from the energy type. For instance, if the subset were name 'mysubset1', the key in the clusterer\_dict for a clustering done using that subset on docking energy would be 'mysubset1\_docking' and for a clustering done using binding energies: 'mysubset1\_binding'. If an output filename is specified, the clusterer writes its data to that file.

**Analyze->Clusterings->Write Clustering File**

Opens a filebrowser allowing the user to specify the filename for the written

clustering. (If there is more than one molecule with conformations in the viewer, this command opens a list chooser allowing the user to select which molecule's clusterings to write. If the selected molecule has clusterings both on binding energy and on docking energy, a list chooser opens allowing the user to specify which clusterer's results to write. Either of these choices having only one possibility is skipped). The specified clusterer writes its data to the specified filename. By convention, these files have **.clust** extensions. The format for a .clust file is a one-line header which lists the rms tolerances at which clusterings have been performed plus a one-letter flag denoting which energy was used for the clustering-'d' or 'b'. The header is followed by one line per conformation, sorted from lowest energy to highest. The lines are made up of pairs of integers, one pair for each rms. The first number of the pair is the cluster rank of the cluster to which a conformation belongs when clustered at the column's rms and the second the rank of the conformation within that cluster.

---

## **Appendix 2: Docking Parameters**

### **Parameters common to SA, GA, GALS:**

**'seed'**: The number of arguments following this keyword determines which random number generator is to be used. One argument causes AutoDock to use the system's implementation of the random number generator and a corresponding system seed call. One argument is required for the simulated annealing algorithm. Two arguments tells AutoDock to use the platform-independent library for random number generation. Two arguments are required for the genetic algorithm. The arguments themselves can be any combination of explicit long integers, the key word 'time' or the keyword 'pid'. 'time' is the number of seconds since the epoch, referenced to 00:00:00CUT 1 Jan 1970. 'pid' gives the UNIX process ID of the currently executing AutoDock process.

**'types'**: Atom names for all atom types present in ligand.

**'fld'**: grid data field file created by AutoGrid and readable by AVS.

**'map'**: filename for the first AutoGrid affinity grid map of the 1<sup>st</sup> atom type. Repeated for all atom types specified in 'types' plus 'e' for required electrostatics map.

**'move'**: filename for the ligand to be docked.

**'about'**: x y z center of ligand about which rotations will be made. Coordinate frame of reference inside AutoDock. That is, internally the ligand's coordinates become centered at the origin.

**'tran0'**: initial coordinates for the center of the ligand or random. Each new run starts the ligand from this location. Please note: the user should ensure that the ligand, when translated to these coordinates still fits inside the volume of the grid maps. If there are some atoms which do lie outside this volume, AutoDock will automatically move the ligand until the ligand is completely inside the box.

**'quat0'**: initial quaternion Qx, Qy, Qz, Qw or random.

**'ndihe'**: number of rotatable bonds in the ligand.

**'dihe0'**: initial relative dihedral angles or random. There must be `ndihe` number of values specified if the user decides to explicitly set the initial relative dihedrals.

**'tstep'**: if one argument, the maximum translation jump per step. If single argument and less than one, the reduction factor is multiplied with the `tstep` at the end of each cycle to get next value. Alternatively, if there are two arguments, the user specifies the value for the first and last cycle and AutoDock calculates the reduction factor that satisfies these constraints. Default is 2.0 Angstrom

**'qstep'**: maximum orientation step size for the angular component `w` of quaternion. Default is 50.0 degrees.

**'dstep'**: maximum dihedral step size. Default is 50.0 degrees.

**'torsdof'**: number and coefficient of the torsional degrees of freedom for the estimation of the change of free energy upon binding. Here, the number of possible rotatable bonds in the ligand excluding any torsions that only rotate hydrogens: eg hydroxyls, amines... The coefficient is 0.3113 kcal/mol

**'intnbp\_r\_eps'**: internal pairwise non-bonded energy parameters for the flexible ligand: equilibrium distance and well depth followed by integer exponents `n` and `m`.

**'intelec'**: Optional: whether to calculate internal ligand electrostatic energies. Default is no.

**'outlev'**: diagnostic output level. For simulated annealing 0= no output, 1=minimal output, 2 = full state output at end of each cycle, 3=detailed output for each step. For GA and GA-LS: 0=minimal output, 1= write minimum, mean and maximum of each state variable at the end of every generation. Use `outlev 1` for SA and `outlev 0` for GA and GA-LS.

**'rmstol'**: the rms deviation tolerance for cluster analysis, carried out after multiple docking runs. If two conformations have an rms less than this tolerance, they will be placed in the same cluster. The structures are ranked by energy, as are the clusters.

**'rmsref'**: the root mean square deviation of the docked conformations calculated with respect to the coordinates in the `pdbq` file or `pdb` file specified here. Particularly useful for comparing a docked result to a known crystal structure.

'**extnrg**': external grid energy assigned to any atoms that stray outside the volume of the grid during a docking. Default is 1000. kcal/mole.

'**analysis**': perform a cluster analysis on results of a docking and output results to the log file. The docked conformations are sorted in order of increasing energy, then compared by root mean square deviation. If the docked result is within the 'rmstol' threshold, it is placed into the same cluster.

### **Simulated Annealing Specific Parameters:**

'**rt0**': initial annealing temperature-actually absolute temperature multiplied by the gas constant. Default is 500. cal/mole.

'**e0max**': two floats-used only in SA. Keyword stipulates that the ligand's initial state cannot have an energy greater than the first value, nor can there be more than the second value's number of retries. Default is 0, 10000.

'**linear\_schedule**': instructs AutoDock to use a linear temperature reduction schedule during Monte Carlo simulated annealing. Unless given, a geometric reduction schedule is used according to rtrf described below. Default is to use linear\_schedule.

'**rtrf**': annealing temperature reduction factor. At the end of each cycle the annealing temperature is multiplied by this factor to give that of the next cycle. Must be positive and less than one. Default is 0.95

'**trnrf**': per cycle reduction factor for translations. Default is 1.0.

'**quarf**': per cycle reduction factor for quaternions. Default is 1.0.

'**dihrf**': per cycle reduction factor for dihedrals. Default is 1.0

'**runs**': number of automated docking runs to carry out. Default is 10.

'**cycles**': number of temperature reduction cycles. Default is 50.

'**accs**': Maximum number of accepted steps per cycle. Default is 100.

'**rejs**': Maximum number of rejected steps per cycle. Default is 100.

'**select**': State selection flag. 'm' minimum state is selected or 'l' last state. Default is m.

'**simanneal**': instructs AutoDock to do specified number of docking runs using the simulated annealing SA search engine.

### **Genetic Algorithm Specific Parameters:**

'**ga\_pop\_size**': number of individuals in the population. Each individual is a coupling of a genotype and its associated phenotype. Typical values range from 50 to 200. Default is 50.

'**ga\_num\_evals**': Maximum number of energy evaluations that a GA run should make. Default is 250000.

'**ga\_num\_generations**': Maximum number of generations that a GA or LGA run should last. Default is 27000.

'**ga\_elitism**': Number of top individuals that are guaranteed to survive into the next generation. Default is 1.

'**ga\_mutation\_rate**': The probability that a particular gene is mutated. Default is 0.02

'**ga\_crossover\_rate**': Crossover rate is the expected number of pairs in the population that will exchange genetic material. Setting this value to 0 turns the GA into the evolutionary programming method (EP) but that requires a concomitant increase in the `ga_mutation_rate` in order to be effective. Default is 0.80.

'**ga\_window\_size**': Number of preceding generations to take into consideration when deciding the threshold for the worst individual in the current population. Default is 10.

'**ga\_cauchy\_alpha**': Alpha parameter in a Cauchy distribution which corresponds roughly to the mean of the distribution. Default is 0.

'**ga\_cauchy\_beta**': Beta parameter in a Cauchy distribution which corresponds roughly to something like the variance of the distribution. However, the Cauchy distribution doesn't have finite variance. Default is 1.

'**set\_ga**': sets the global optimizer to be a genetic algorithm. PLEASE note all ga\_ parameters must be specified before this line in order to be used in the docking.

'**do\_global\_only**': instructs AutoDock to carry out docking using only a global search. Local search parameters in the dpf are ignored when this is present.

### **Local Search Specific Parameters:**

'**sw\_max\_its**': Maximum number of iterations that the local search procedure applies to the phenotype of any given individual. Default is 50.

'**sw\_max\_succ**': Number of successes in a row before a change is made to the rho parameter in Solis & Wets algorithm. Default is 4.

'**sw\_max\_fail**': Number of failures in a row before Solis & Wets algorithms adjust rho. Default is 4.

'**sw\_rho**': Parameter defining the initial variance and specifying the size of the local space to sample. Default is 1.0.

'**sw\_lb\_rho**': Lower bound on rho, the variance for making changes to genes. Rho can never be modified to a value smaller than sw\_lb\_rho. Default is 0.01.

'**ls\_search\_freq**': The probability of any particular phenotype being subjected to local search. Default is 0.06.

'**set\_psw1**': Instructs AutoDock to use the pseudo-Solis and Wets local searcher. This method maintains the relative proportions of variances for the translations in Angstrom and the rotations in radians. These are typically 0.2 Angstrom and 0.087 radians to start with so the variance for translations will always be 2.3 times larger than that for the rotations (i.e. the orientation and torsions).

'**do\_local\_only**': Instructs AutoDock to carry out only the local search of a global-local search. The genetic algorithm parameters are ignored, except for the population size. This is an ideal way of carrying out a minimization using the same force field as is used during the dockings. The ga\_run keyword should not be given. The integer following the keyword determines how many dockings will be performed. Default is 50.

GALS parameters include all Genetic Algorithm and Local Search parameters listed above except **do\_global\_only** and **do\_local\_only**.

### **Clustering keywords:**

**'cluster'**: keyword specific to reclustering jobs. Parameter which follows is filename for concatenated docking logs. Script 'recluster.py' produces this kind of file.

**'rmstol'**: root mean square tolerance for reclustering.

**'types'**: types of atoms present in ligand

**'write\_all\_cluster\_members'**: option causes printed output of all docked structures in each cluster. Default is to write the first docked structure in each cluster only. (The first docked structure has the lowest energy of all the docked structures in the cluster.)