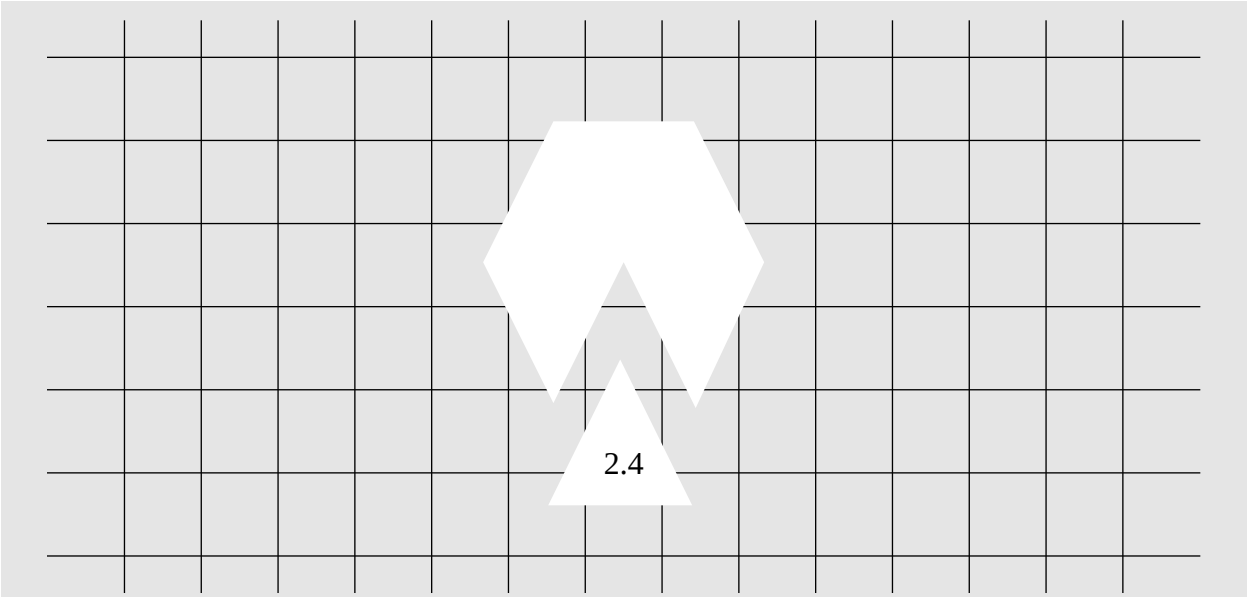
User Guide

AutoDock

Automated Docking of Flexible Ligands to Receptors



Version 2.4

Garrett M. Morris, David S. Goodsell, & Ruth Huey

Arthur J. Olson

Constructive comments, suggestions, feedback and bug reports are welcome; please e-mail me. - Garrett Morris.

E-mail addresses:

Arthur J. Olson, Ph.D.: olson@scripps.edu
Peggy Graber, *Administrative Assistant*: graber@scripps.edu
Garrett M. Morris, M.A., D.Phil. garrett@scripps.edu
David S. Goodsell, Ph.D. goodsell@scripps.edu
Ruth Huey, Ph.D. rhuey@scripps.edu

FAX: (619) 554-6860

The Scripps Research Institute,
Department of Molecular Biology,
Mail Drop MB-5,
10666 North Torrey Pines Road,
La Jolla,
CA 92037-5025, USA.

Modification date: June 3, 1996 12:45 pm

1. Contents

Introduction to AutoDock

1. Introduction

The program **AutoDock** was developed to provide a procedure for predicting the interaction of small molecules with macromolecular targets. The motivation for this work arises from problems in the design of bioactive compounds, and in particular the field of computer aided drug design. Progress in biomolecular x-ray crystallography has provided a number of important protein and nucleic acid structures that could be targets for bioactive agents in the control of disease, or as agricultural agents. The precise interaction of such agents or candidates is important in the development process. Our goal has been to provide a computational tool to aid in this process.

In any docking scheme two conflicting requirements must be balanced: the desire for a robust and accurate procedure, and the desire to keep the computational demands at a reasonable level. The ideal procedure would find the global minimum in the interaction energy between the substrate and the target protein, exploring all available degrees of freedom for the system. However, it would also run on a laboratory workstation within an amount of time comparable to other computations that a structural researcher may undertake, such as a crystallographic refinement. In order to meet these demands a number of docking techniques simplify the docking procedure. Probably the most common technique in use today is manually assisted docking. Here, the internal and orientational degrees of freedom in the substrate are under interactive control. While the energy evaluation for such techniques can be sophisticated, the exploration of configurational space is limited. At the other end of the spectrum are automated methods such as exhaustive search and distance geometry. These methods can explore configurational space, but at the cost of a much simplified model for the energetic evaluation. The procedure developed for **AutoDock** uses a Monte Carlo simulated annealing technique for configurational exploration with a rapid energy evaluation using grid based molecular affinity potentials, thus combining the advantages of a large search space and a robust energy evaluation. This has proven to be a powerful approach to the problem of docking a flexible substrate into the binding site of a static protein. Input to the procedure is minimal. The researcher specifies a volume around the protein, the rotatable bonds for the substrate, and an arbitrary starting configuration, and the procedure produces a relatively unbiased docking.

2. Overview of the Method

Rapid energy evaluation is achieved by precalculating atomic affinity potentials for each atom type in the substrate molecule in the manner described by Goodford [1]. In the **AutoGrid** procedure the protein is embedded in a three-dimensional grid and a probe atom is placed at each grid point.

1. Goodford, P.J. (1985) "A Computational Procedure for Determining Energetically Favorable Binding Sites on Biologically Important Macromolecules" *J. Med. Chem.*, **28**, 849-857.

The energy of interaction of this single atom with the protein is assigned to the grid point. An affinity grid is calculated for each type of atom in the substrate, typically carbon, oxygen, nitrogen and hydrogen, as well as a grid of electrostatic potential, either using a point charge of +1 as the probe, or using a Poisson-Boltzmann finite difference method, such as DELPHI [2,3]. The energetics of a particular substrate configuration is then found by tri-linear interpolation of affinity values of the eight grid points surrounding each of the atoms in the substrate. The electrostatic interaction is evaluated similarly, by interpolating the values of the electrostatic potential and multiplying by the charge on the atom (the electrostatic term is evaluated separately to allow finer control of the substrate atomic charges). The energy calculation using the grids is proportional only to the number of atoms in the substrate, and not to any function of the number of atoms in the protein.

The docking simulation is carried out using the *Metropolis method*, also known as *Monte Carlo simulated annealing*. With the protein static throughout the simulation, the substrate molecule performs a random walk in the space around the protein. At each step in the simulation, a small random displacement is applied to each of the degrees of freedom of the substrate: translation of its center of gravity; orientation; and rotation around each of its flexible internal dihedral angles. This displacement results in a new configuration, whose energy is evaluated using the grid interpolation procedure described above. This new energy is compared to the energy of the preceding step. If the new energy is lower, the new configuration is immediately accepted. If the new energy is higher, then the configuration is accepted or rejected based upon a probability expression dependent on a user defined temperature, *T*. The probability of acceptance is given by:

$$P(\Delta E) = e^{\left(-\frac{\Delta E}{k_B T}\right)}$$

where ΔE is the difference in energy from the previous step, and k_B is the Boltzmann constant. At high enough temperatures, almost all steps are accepted. At lower temperatures, fewer high energy structures are accepted.

The simulation proceeds as a series of cycles, each at a specified temperature. Each cycle contains a large number of individual steps, accepting or rejecting the steps based upon the current temperature. After a specified number of acceptances or rejections, the next cycle begins with a temperature lowered by a specified schedule such as:

$$T_i = gT_{i-1}$$

where T_i is the temperature at cycle i, and g is a constant between 0 and 1.

Simulated annealing allows an efficient exploration of the complex configurational space with multiple minima that is typical of a docking problem. The separation of the calculation of the molecular affinity grids from the docking simulation provides a modularity to the procedure, allowing the exploration of a range of representations of molecular interactions, from constant dielectrics to finite difference methods and from standard 12-6 potential functions to distributions

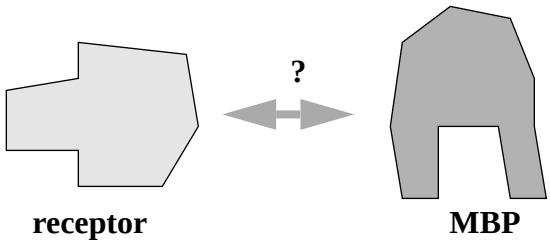
2. Sharp, K., Fine, R. & Honig, B. (1987) *Science*, **236**, 1460-1463.
3. Allison, S.A., Bacquet, R.J., & McCammon, J. (1988) *Biopolymers*, **27**, 251-269.

based on observed binding sites.

3. Applications

AutoDock was initially tested on a number of protein-substrate complexes which had been characterized by x-ray crystallography [4]. These tests included phosphocholine binding in a antibody combining site, N-formyltryptophan binding to chymotrypsin and N-acetylglucosamine binding to Lysozyme. In almost all cases the results of the **AutoDock** simulations functionally reproduced the crystallographic complexes. In further applications **AutoDock** was used to predict interactions of substrates with aconitase prior to any crystallographic structures for complexes. In this work we not only predicted the binding mode of isocitrate, but we demonstrated the utility of **AutoDock** in generating substrate models during the early stages of crystallographic proteins structure refinement [5]. Citrate docking experiments showed two binding modes, one of which approximated the experimental electron density determined for an aconitase-nitrocitrate complex. The docking simulation results provided insight into the proposed reaction mechanism of the enzyme.

The initial version of **AutoDock** has been distributed to over 35 sites around the world, and has begun to be used in universities and research labs to help predict and design bioactive agents. One novel and intriguing use of the software was reported from Koshland's laboratory[6]. These investi-



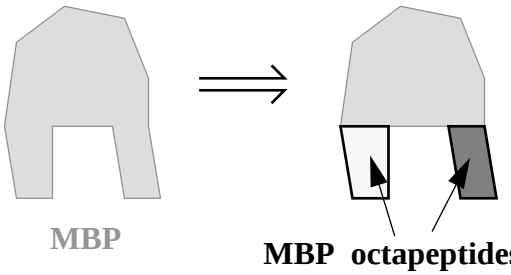
gators used the known structures of the maltose-binding protein (MBP) and the ligand binding domain of the aspartate receptor to predict the structure of the receptor-protein complex (see dia-

4. Goodsell, D.S. & Olson, A.J. (1990) "Automated Docking of Substrates to Proteins by Simulated Annealing" *Proteins: Str. Func. Genet.*, **8**, 195-202.

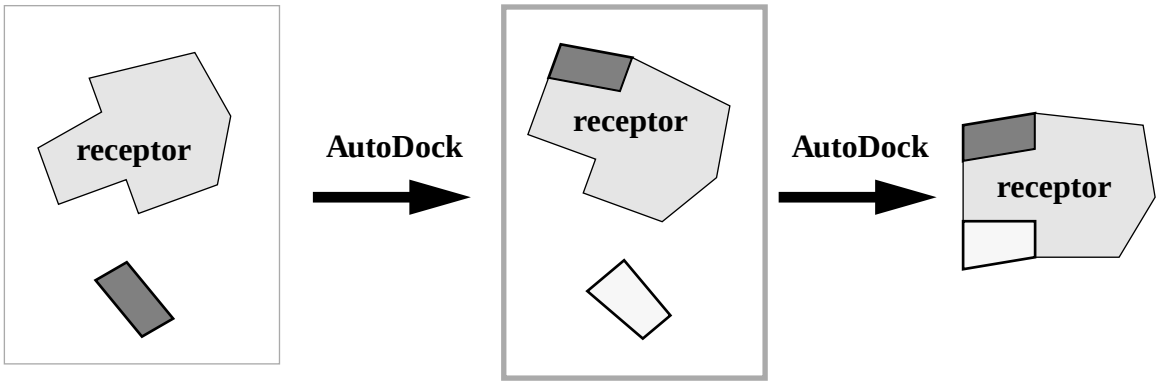
5. Goodsell, D.S., Lauble, H., Stout, C.D & Olson, A.J. (1993) "Automated Docking in Crystallography: Analysis of the Substrates of Aconitase" *Proteins: Str. Func. Genet.*, **17**, 1-10.

6. Stoddard, B.L. & Koshland, D.E. (1992) "Prediction of a receptor protein complex using a binary docking method." *Nature*, **358** (6389), 774-776.

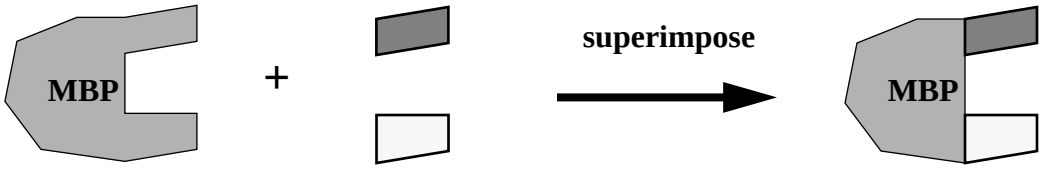
gram below). They used knowledge from mutational studies on MBP to select two octapeptides



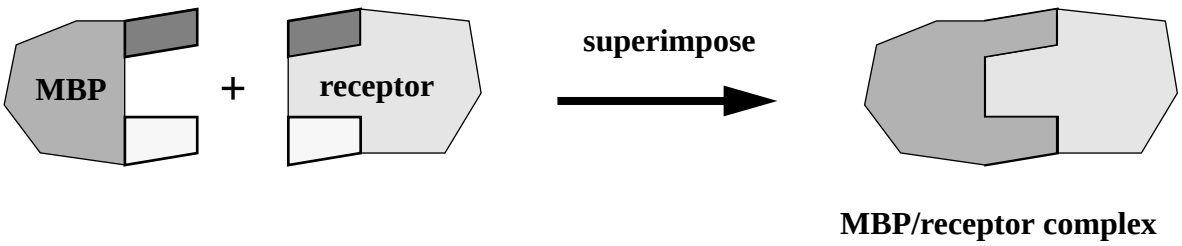
on the protein known to be involved in the binding to the aspartate receptor, which they docked



independently to the model of the receptor using our automated docking code (the backbones of the peptides were fixed, but the side-chain conformations and overall orientations were unrestrained).



The distance and orientation of the two peptides as docked to the receptor corresponded to that in the intact MBP, thus enabling a reasonable prediction of the protein-receptor complex. This technique could be generally useful in situations where there are data on multi-site interactions.

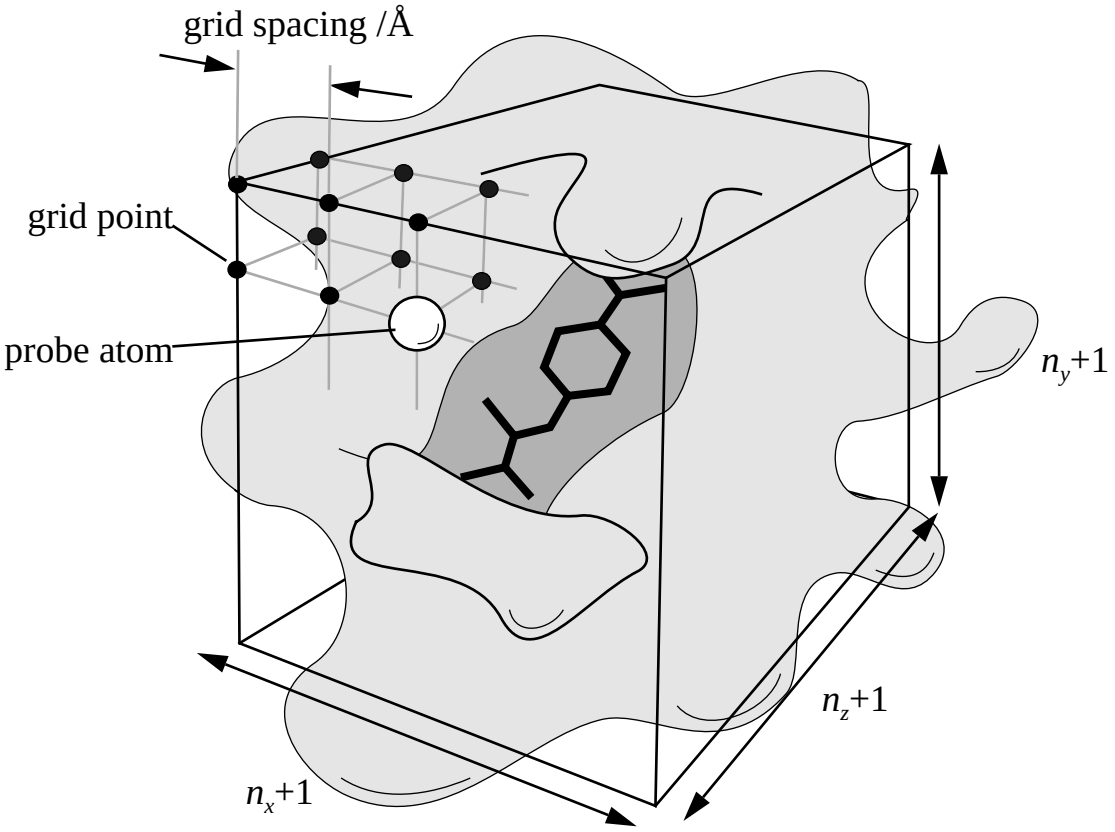


AutoDock, AutoGrid and AutoTors

This User Guide describes how to prepare, run and analyze an automated docking of a small molecule to a macromolecule, such as a protein, enzyme or oligomeric DNA, using **AutoDock**.

1. Background

AutoDock requires *grid maps* for each atom type present in the small molecule being docked. They are calculated and produced by the **AutoGrid** program. A grid map consists of a three dimensional array of regularly spaced points, centered (usually) on the active site of the protein or macromolecule under study. Each point within the grid map stores the potential energy of a '*probe*' atom or functional group at that particular position:

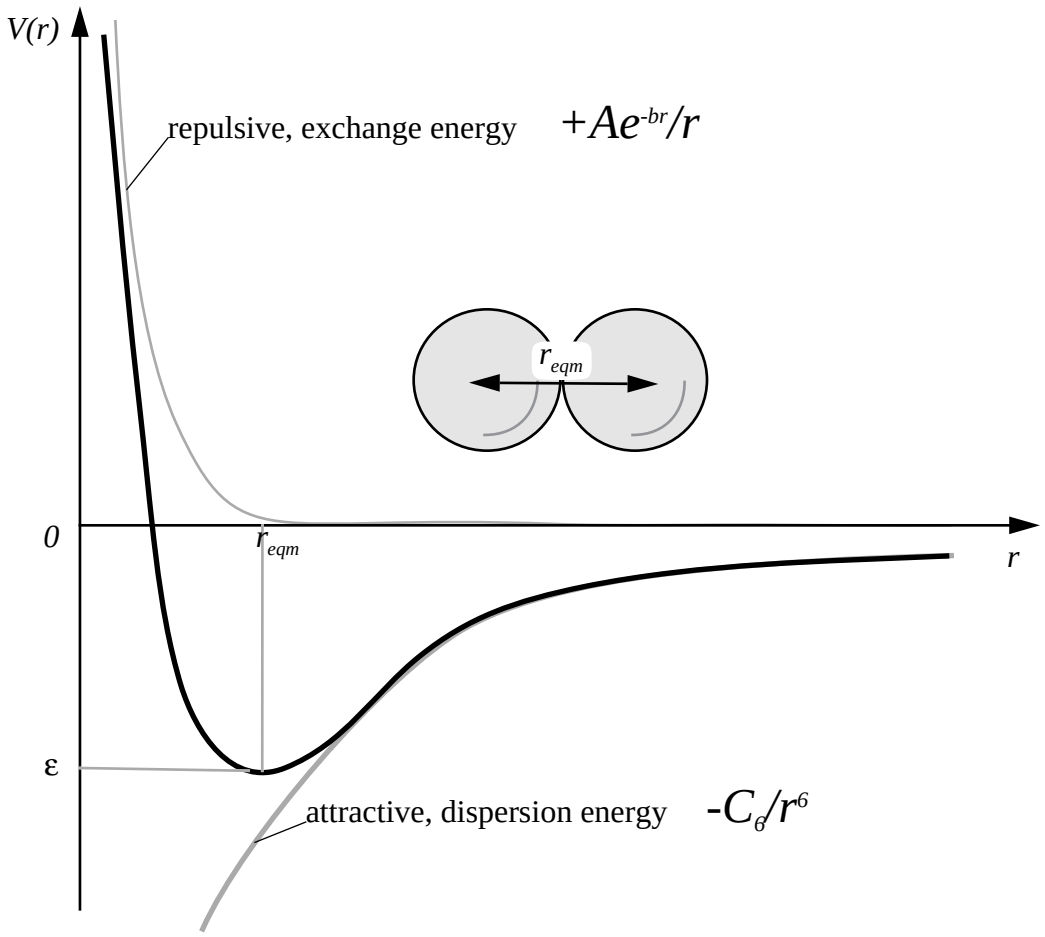


The user must specify an *even* number for n_x, n_y and n_z, since **AutoGrid** adds a central point, and **AutoDock** requires an odd number of grid points. The probe's energy at each grid point is determined by the set of parameters supplied for that particular atom type, summed over all atoms of the macromolecule, within a non-bonded radius. This potential energy, *V(r)*, can be expressed as a

function of internuclear separation, *r*, as follows,

$$V(r) = \frac{Ae^{-br}}{r} - \frac{C_6}{r^6}$$

Graphically, if r_{eqm} is the *equilibrium internuclear separation*, and ε is the *well depth* at r_{eqm}, then:



The exchange energy is often approximated thus,

$$\frac{A}{r}e^{-br} \approx \frac{C_{12}}{r^{12}}$$

Hence pairwise-atomic interaction energies can be approximated by the following general equation,

$$V(r) \approx \frac{C_n}{r^n} - \frac{C_m}{r^m} = C_n r^{-n} - C_m r^{-m}$$

where *m* and *n* are integers, and C_n and C_m are constants whose values depend on the depth of the energy well and the equilibrium separation of the two atoms' nuclei. Typically the 12-6 Lennard-Jones parameters (*n*=12, *m*=6) are used to model the Van der Waals' forces[1] experienced between two instantaneous dipoles. However, the 12-10 form of this expression (*n*=12, *m*=10) can be used to model hydrogen bonds (see *"Modelling Hydrogen Bonds"* below). Appendix 21 gives the parameters which were distributed with the first (FORTRAN-77) version of AutoDock, and which have been used in numerous published articles.

A revised set of parameters has been calculated, which are self-consistent. Here, the sum of the Van der Waals radii of any given pair of atoms is consistent with that of any other pair. Likwise, the well-depths are consistently related. Let $r_{eqm,\,XX}$ be the equilibrium separation between two like atoms *X*, and ε_{XX} be their potential energy in this configuration. The combining rules for the well depth, ε, and the Van der Waals radius, r_{eqm}, for two different atoms *X* and *Y* are:

$$\varepsilon_{XY} = \sqrt{\varepsilon_{XX}\varepsilon_{YY}}$$

$$r_{eqm,XY} = \frac{1}{2}(r_{eqm,XX} + r_{eqm,YY})$$

A derivation for the Lennard-Jones potential sometimes seen in text books invokes the parameter, σ, thus,

$$r_{eqm,\,XY} = 2^{\frac{1}{6}}\sigma$$

Then the Lennard-Jones 12-6 potential becomes:

$$V_{12\text{-}6}(r) = 4\varepsilon_{XY}\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

Hence, the coefficients C_{12} and C_6 are given by:

$$C_{12} = \varepsilon_{XY} r_{eqm,\,XY}^{12}$$

$$C_6 = 2\varepsilon_{XY} r_{eqm,\,XY}^{6}$$

We can derive a general relationship between the coefficients, equilibrium separation and well depth as follows. At the equilibrium separation, r_{eqm}, the potential energy is a minimum: in other words, $V(r_{eqm}) = -\varepsilon$. The derivative of the potential will be zero at the minimum:

1. van der Waals, J. H. (1908) Lehrbuch der Thermodynamik, Mass and Van Suchtelen, Leipzig, Part 1

$$\frac{dV}{dr} = -\frac{nC_n}{r^{n+1}} + \frac{mC_m}{r^{m+1}} = 0$$

therefore:

$$\frac{nC_n}{r^{n+1}} = \frac{mC_m}{r^{m+1}}$$

so:

$$C_m = \frac{nC_n r^{m+1}}{mr^{n+1}} = \frac{n}{m} C_n r^{(m-n)}$$

Substituting C_m into the original equation for $V(r)$, at equilibrium, we obtain,

$$-\varepsilon = \frac{C_n}{r_{eqm}^n} - \frac{nC_n r_{eqm}^{(m-n)}}{mr_{eqm}^m}$$

Rearranging:

$$C_n \left(\frac{mr_{eqm}^m - nr_{eqm}^n r_{eqm}^{(m-n)}}{mr_{eqm}^n r_{eqm}^m} \right) = -\varepsilon$$

Therefore, the coefficient C_n can be expressed in terms of n, m, ε and r_{eqm} thus:

$$\boxed{C_n = \frac{m}{n-m} \varepsilon r_{eqm}^n}$$

and, substituting into original equation for $V(r)$,

$$\boxed{C_m = \frac{n}{n-m} \varepsilon r_{eqm}^m}$$

In summary, then:

$$V(r) \approx \frac{\frac{m}{n-m} \varepsilon r_{eqm}^n}{r^n} - \frac{\frac{n}{n-m} \varepsilon r_{eqm}^m}{r^m}$$

Some example r_{eqm} and ε parameters for various AMBER atom types of carbon are shown in Table

1.

Table 1: AMBER parameters for carbon atom types.

AMBER atom type	r_{eqm} / Å	ε / kcal mol $^{-1}$
C, C*, CA, CB, CC, CD, CE, CF, CG, CH, CI, CJ, CM, CN, CP	1.85	0.12
C2	1.925	0.12
C3	2.00	0.15
CH	1.85	0.09
CT	1.80	0.06

Using the equations describing C_{12} and C_6 above, the following new set of 12-6 parameters were calculated (see Table 2). These parameters may be used with **AutoDock** version 2.4, or alternatively, you may use or derive your own.

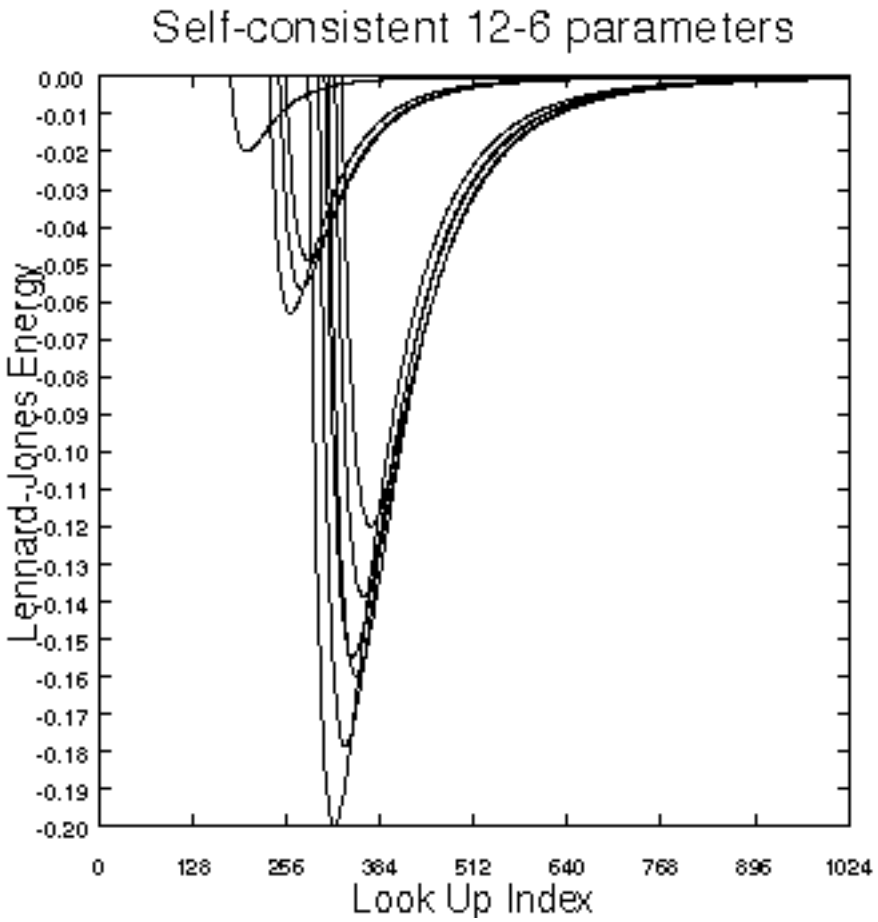
Table 2: Self-consistent Lennard-Jones 12-6 parameters.

Atoms i-j	$r_{eqm,ij}$ / Å	ε_{ij} / kcal mol^{-1}	C_{12} / kcal mol^{-1}Å^{12}	C_6 / kcal mol^{-1}Å^6
C-C	4.00	0.150	2516582.400	1228.800000
C-N	3.75	0.155	1198066.249	861.634784
C-O	3.60	0.173	820711.722	754.059521
C-S	4.00	0.173	2905899.052	1418.896022
C-H	3.00	0.055	29108.222	79.857949
N-C	3.75	0.155	1198066.249	861.634784
N-N	3.50	0.160	540675.281	588.245000
N-O	3.35	0.179	357365.541	505.677729
N-S	3.75	0.179	1383407.742	994.930149
N-H	2.75	0.057	10581.989	48.932922
O-C	3.60	0.173	820711.722	754.059521
O-N	3.35	0.179	357365.541	505.677729
O-O	3.20	0.200	230584.301	429.496730
O-S	3.60	0.200	947676.268	870.712934
O-H	2.60	0.063	6035.457	39.075098
S-C	4.00	0.173	2905899.052	1418.896022
S-N	3.75	0.179	1383407.742	994.930149

Table 2: Self-consistent Lennard-Jones 12-6 parameters.

Atoms i-j	$r_{eqm,ij}$ / Å	ε_{ij} / kcal mol^{-1}	C_{12} / kcal mol^{-1}Å^{12}	C_6 / kcal mol^{-1}Å^6
S-O	3.60	0.200	947676.268	870.712934
S-S	4.00	0.200	3355443.200	1638.400000
S-H	3.00	0.063	33611.280	92.212017
H-C	3.00	0.055	29108.222	79.857949
H-N	2.75	0.057	10581.989	48.932922
H-O	2.60	0.063	6035.457	39.075098
H-S	3.00	0.063	33611.280	92.212017
H-H	2.00	0.020	81.920	2.560000

The above parameters yield the following graphs, for C, N, O and H atom types; the curves in order of increasing well-depth are: HH << CH < NH < OH << CC < CN < CO < NN < NO < OO:-



Grid maps are required only for those atom types present in the small molecule being docked. For

example, if the small molecule being docked is a hydrocarbon, then only carbon and hydrogen grid maps would be required. In practice, however, non-polar hydrogens would not be modelled explicitly, so just the carbon grid map would be needed, for 'united atom' carbons. This saves both disk space and computational time.

2. Electrostatic Potential Grid Maps

In addition to the atomic affinity grid maps, **AutoDock** requires an electrostatic potential grid map. Polar hydrogens must be added, if hydrogen-bonds are being modelled explicitly. Partial atomic charges must be assigned to the macromolecule. The electrostatic grid can be generated by **AutoGrid**, or by other programs such as MEAD[2] or DELPHI[3], which solve the linearized Poisson-Boltzmann equation. **AutoGrid** calculates Coulombic interactions between the macromolecule and a probe of charge *e*, $+1.60219 \times 10^{-19}$ C; there is no distance cuttoff used for electrostatic interactions. A sigmoidal distance-dependent dielectric function is used to model solvent screening, based on the work of Mehler and Solmajer[4],

$$\varepsilon(r) = A + \frac{B}{1 + ke^{-\lambda B r}}$$

where: $B = \varepsilon_0 - A$; ε_0 = the dielectric constant of bulk water at 25°C = 78.4; A = -8.5525, λ = 0.003627 and k = 7.7839 are parameters.

Charges must be stored in ***PDBQ format*** in order for **AutoGrid** to read them. PDBQ is an augmented form of the standard PDB format, in which an extra column is used to store the partial atomic charges (hence the "Q" in "PDBQ"). Columns 71-76 of the PDB file hold the partial atomic charge (the older form of PDBQ contains charges in columns 55-61).

Charges can be assigned using a molecular modelling program. Unix shell scripts are provided to convert from **InsightII**[5] ".car" files ("cartopdbq") and **SYBYL**[6] ".mol2" files ("`mol2topdbq`"). See also "q.amber" and "q.kollua", in the appendices.

---

2. Bashford, D. and Gerwert, K. (1992) "Electrostatic calculations of the pK_a values of ionizable groups in bacteriorhodopsin", *J. Mol. Biol.*, **224**, 473-486; Bashford, D. and Karplus, M. (1990) "pK_as of ionizable groups in proteins - atomic detail from a continuum electrostatic model.", *Biochemistry*, **29**, 10219-10225; MEAD is available from Donald E. Bashford, Dept. Molecular Biology, Mail Drop MB1, The Scripps Research Institute, 10666 North Torrey Pines Road, La Jolla, CA 92037.
3. Gilson, M.K. and Honig, B. (1987) *Nature*, **330**, 84-86; DELPHI is available from Biosym Technologies, 9685 Scranton Road, San Diego, CA 92121-2777, USA.
4. Mehler, E.L. and Solmajer, T. (1991) "Electrostatic effects in proteins: comparison of dielectric and charge models" *Protein Engineering*, **4**, 903-910.
5. Biosym Technologies, 9685 Scranton Road, San Diego, CA 92121-2777, USA.
6. Tripos Associates, Inc., 1699 South Hanley Road, Suite 303, St. Louis, Missouri 63144-2913, USA.

3. Simulated Annealing in AutoDock

As already described in the Introduction, **AutoDock** uses *Monte Carlo* simulated annealing to explore a wide range of conformational states. A "job" consists of a number of independent runs, each of which begins with the same initial conditions. A run is a sequence of constant temperature annealing cycles. Each job can be seeded with a user-defined or a time-dependent random-number generator seed.

When using the Monte Carlo approach, it is important to consider the algorithm for random number generation. Ideally, the samples are uniform and uncorrelated in hyperdimensions; and the period is longer than the number of random values called for in a given simulation.

During each constant temperature cycle, random changes are made to the ligand's current position, orientation, and conformation if flexibiliy was defined. The resulting new state is then compared to its predecessor; if the new energy is lower than the last, this new state is accepted. However, if the new state's energy is higher than the last, this state is accepted probabilistically. This probability depends upon the energy and cycle temperature (see the first equation on page 3). Generally speaking, at high temperatures, many states will be accepted, while at low temperatures, the majority of these probabilistic moves will be rejected. The user can select the minimum energy state found during a cycle to be used as the initial state for the next cycle, or the last state can be used. The best results tend to be achieved by selecting the minimum energy state from the previous cycle.

Quaternion rotations[7] have been implemented in handling the rigid body rotation of the small molecule. It was found that this gave finer control over the movement of the small molecule, and gave better docked conformations than the alternative Eulerian rotations.

It is advisable to do a short run to check the setup first. The initial annealing temperature should be of the order of the average ΔE found during the first cycle. This ensures that the ratio of accepted to rejected steps is high near the start. A typical automated docking or 'job' may have an initial annealing temperature of 500 and a temperature reduction factor of 0.85-0.95 /cycle. Gradual cooling is recommended, so as to avoid "*simulated quenching*", which tends to trap systems in local minima. A relatively thorough search is given by 50 *Monte Carlo* cycles, and a maximum of 30,000 steps rejected or 30,000 steps accepted. 10 runs give a feel for the possible binding modes, and also an idea of their relative energies. When more than one run is carried out in a given job, cluster analysis or 'structure binning' will be performed, based on structural rms difference, ranking the resulting families of docked conformations in order of increasing energy. The default method for structure binning allows for symmetry rotations, as in a tertiary butyl which can be rotated by +/-120˚, but in other cases it may be desirable to bypass this similar atom type checking and calculate the rms on a one-for-one basis. A schedule of 100 runs, 50 cycles, 3,000 steps accepted, 3,000 steps rejected will provide more highly populated clusters, hinting at the "density of states" for a given conformation. A short test job would be: 1 run, 50 cycles, 100 accepted, 100

7. Shoemake, K. (1985) "Animating Rotation with Quaternion Curves" *SIGGRAPH '85*, **19**, 245-254.

rejected steps.

The user must specify the maximum jump a state variable can make in one *Monte Carlo* step. The default values are: translation, 0.2 Å; rigid-body rotation, 5°; and torsion angle rotation, 5°. Furthermore, these can be adjusted from cycle to cycle during each run, if the reduction factor for translations and rotations is given a (positive, non-zero) value less than one. At the start of each cycle, this constant is multiplied onto the current jump-maxima to give the new maxima.

If desired, the states visited during a docking simulation can be sampled and output to a trajectory file. This file contains all the state variables required to define each sampled conformation, position and orientation of the small molecule. The user can specify the range of cycles to be sampled. This allows the selection of the last few cycles when the docking will be nearing the final docked conformation, or the selection of the whole run.

4. Setting Up AutoGrid and AutoDock Jobs

Let us suppose that the user wishes to test **AutoDock** by trying to reproduce an x-ray crystallographic structure of a small molecule-enzyme complex taken from the Brookhaven Protein Data Bank. The first step is to create two PDB files, one containing all the heavy atoms of the enzyme, the second containing those of the small molecule. Both files should retain the extension '`.pdb`'.

Note: Care should be taken when the PDB file contains disordered residues, in which alternate location indicators (column 17) have been assigned; for each such residue, the user must select only one of the possible alternate locations (preferably that with the highest occupancy value).

We will discuss in the next sections, the steps needed to prepare the parameter files for **AutoGrid** and **AutoDock**. If desired the user can specify rotatable bonds in the small molecule (receptor flexibility is not allowed). To help in the definition of rotatable bonds in the small molecule, there is a tool called **AutoTors**, which prepares the small molecule input file for **AutoDock**, defining the 'root', 'branches' and 'torsions' automatically.

5. Preparing the Small Molecule

Initially you must add hydrogens to all the heavy atoms in the small molecule, ensuring their valences are completed. This can be done using a molecular modelling package. Make sure that the atom types are correct before adding hydrogens.

Next, assign partial atomic charges to the molecule. AMPAC or MOPAC can be used to generate partial atomic charges for the small molecule. Write these charges out in PDBQ format (see '`cartopdbq`' and '`mol2topdbq`' in Appendix 20).

6. Modelling Hydrogen Bonds

Hydrogen bonds are often important in ligand binding. These interactions can be modelled explicitly in **AutoDock**. Polar hydrogens can be allowed to rotate freely.

The partial atomic charges of *non-polar* hydrogens need to be *united* with their heavy atom's. This saves having two types of hydrogen grids, thus conserving disk space and computational time. To 'unite' a non-polar hydrogen's partial charge, the latter is added to that of the heavy atom to which it is bonded. The hydrogen can then be deleted from the small molecule. This is repeated for all non-polar hydrogens. **AutoTors** has a flag '`-h`', which will unite non-polar hydrogens for you, automatically.

The user *must* specify the appropriate 12-10 parameters in the **AutoGrid** parameter file, and on the correct lines. Pairwise atomic interaction energy parameters are always given in blocks of 7 lines, in the order: C, N, O, S, H, X, M. X and M are "spare" atom types: If there were phosphorus atoms in the receptor, X could be used as P. For example, to model *donor* hydrogens in the small molecule, 12-10 parameters would be needed in the *hydrogen* parameter block, but only for H-bond acceptors, N,O and S (second, third and fourth lines in the H-parameters). The other parameters remain as 12-6 Lennard-Jones values (C,H,X and M). In order to keep the symmetry of pairwise energetics (H-O is the same as O-H), the user must specify 12-10 parameters for H (fifth line) in the N, O and S-parameter blocks.

Table 3: Self-consistent Hydrogen bonding 12-10 parameters.

Atoms i-j	$r_{eqm,ij}$ */ Å*	ε_{ij} */ kcal mol^{-1}*	C_{12} */ kcal mol^{-1}Å^{12}*	C_{10} */ kcal mol^{-1}Å^{6}*
N-H	1.90	5.00	55332.873	18393.199
O-H	1.90	5.00	55332.873	18393.199
S-H	2.50	1.00	298023.224	57220.459

AutoGrid detects hydrogen bond parameters in the grid parameter file, if either *n* is not 12 or *m* is not 6. If so, the pairwise interaction is modulated by a function of the cosine of the hydrogen bond angle. This takes into account the directionality of hydrogen-bonds.

7. Small Molecule Flexibility and Constraints

To allow flexibility in the small molecule, it is necessary to assign the rotatable bonds. It is a good idea to have a plot of the small molecule, labelled by atom name, and a second labelled by atom serial number or "ID". **AutoDock** can handle up to `MAX_TORS` rotatable bonds. This parameter is defined in '`autodock.h`', and is ordinarily set to 32.

Torsions are defined in the PDBQ file using a number of *keywords*. These keywords use the metaphor of a tree. See the diagram below for an example. The 'root' is defined as the *fixed* portion of the small molecule, off which *rotatable* 'branches' sprout. Branches within branches are possible, and torsions are a special case of branches, where the two atoms at either end of the rotatable bond have only *two* nearest neighbours (unlike branches which can have three or more). Rotatable fragments are moved in order from leaves to root.

The PDBQ keywords must be carefully placed, and the order of the ATOM or HETATM records may need to be changed in order to fit into the correct branches. The following keywords are recognized by the **AutoDock** PDBQ file parser:

```
ROOT, ENDROOT
BRANCH, ENDBRANCH
TORSION, ENDTORSION
CONSTRAIN
```

They can be abbreviated to no less than the first 4 letters. To assist the user in correctly placing these keywords, and in re-ordering the ATOM or HETATM records in the small molecule PDBQ file, there is an interactive program called AutoTors to do this (see below).

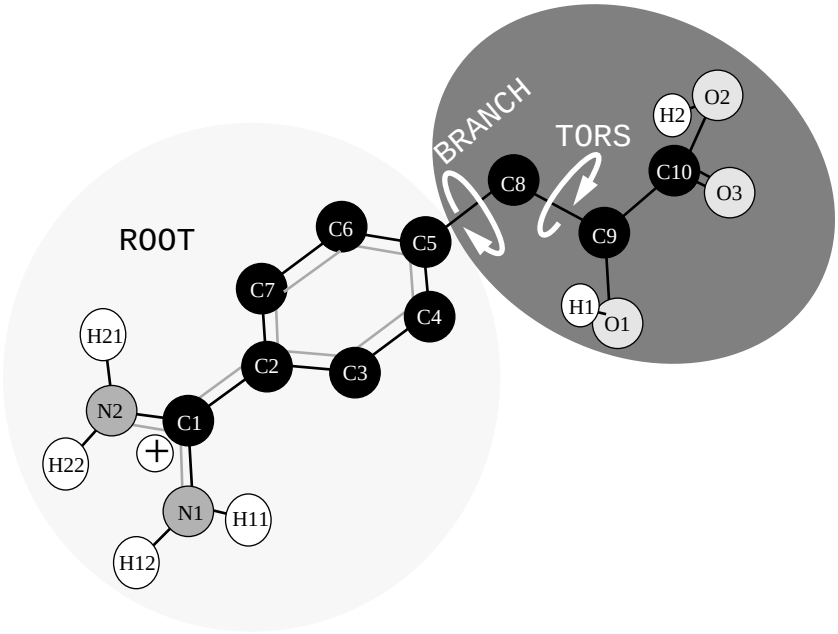
Note: **AutoTors**, **AutoGrid** and **AutoDock** do not recognize PDB "CONECT", neither do they write out "CONECT" records.

"CONSTRAIN" definesa single distance constraint, between two flexible parts of the small molecule. This retains only those conformations where this distance is within a certain range of values. In docking, a conformation which violates this constraint is instantly rejected; it does not increment the rejections-counter, its energy is not evaluated, nor is the steps-counter incremented. The command has the following syntax:

```
CONSTRAIN atom1 atom2 lower upper
```

The first two parameters are the atom serial numbers of the two atoms to be constrained, and the last two are the lower and upper bounds for this distance, in Angstroms. This can be particularly useful when docking say two proteins: a loop from one protein can be cut out and the ends con-

strained to have roughly the same value as in the original protein.



The next sections describe the input needed for **AutoTors**, and how to run it.

8. Input for AutoTors

This section describes input and output files used and generated by **AutoTors**. Input consists of one or two files, depending on whether the small molecule is in our "**AutoDock**-standard" PDBQ-format, or in Sybyl's `mol2`-format. PDBQ-format is the default; `mol2`-format is allowed with the "`-m`" flag (see below).

PDBQ-format:

When using PDBQ format, AutoTors also needs a bond file. In this example, the bond file is "oligo.bnd", and "oligo.pdbq" is the input PDBQ file; "oligo.out.pdbq" is created and contains all the ROOT, BRANCH and TORS keywords needed to define the torsions selected by the user.

`autotors oligo.bnd oligo.pdbq oligo.out.pdbq`

The ".bnd" file, contains information about the covalent bonds in the small molecule. The bonds are described by the serial numbers of the atoms in the input PDBQ file, with one line per bond. For example, if C10 is the atom appearing on the first "ATOM" line in the PDBQ file, and it is bonded to N18 which appears on the 17th line in the PDBQ file, this information appears as a discrete line in the ".bnd" file as: "1 17". The output of the script "pdbtoatm" is an "atm" file; this can be converted to a "bnd" file, using "atmtobnd". For example, to generate a "bnd" file, use something like this command:

```
pdbtoatm vitc.pdbq | atmtobnd > vitc.bnd
```

Mol2-format:

When using SYBYL-mol2 format, only one input file is required, in addition to `-m` flag. This is because the mol2 file contains both atom coordinates and bonding information. So, for eample the following command would read in the "lead.mol2" file and after interactively requesting which torsions to rotate, AutoTors would write out "lead.out.pdbq":

```
autotors -m lead.mol2 lead.out.pdbq
```

AutoTors Output:

The output filename is defined by the last **AutoTors** command-line argument. Output consists of PDBQ-formatted lines, rearranged as required by **AutoDock,** according to the user's specification of the fixed ROOT portion of the molecule, and the allowed rotatable bonds in the rest of the molecule. **AutoTors** inserts the ROOT, ENDROOT, BRANCH, ENDBRANCH, TORS, and ENDTORS lines in the necessary places.

AutoTors Flags:

`-m`

as described under "Mol2-format" above, this tells AutoTors to read in coordinates, partial charges and bond information from a SYBYL-mol2 formatted file.

`-h`

'merge' or add the charges of non-polar *hydrogen* atoms to that of the carbon atom to which they are bonded, and then delete these hydrogens from the molecule. In other words, the molecule is converted from an "all atom" representation to a "united atom" representation. The net charge on the molecule remains the same.

`-o`

read partial atomic charges from column 55 onwards (i.e. the *older* PDBQ format). The default is the new PDBQ format which has the charge data in the 71-st to 76-th columns.

`-a`

disallows torsions in *amide* bonds. This should normally be used, because amide bonds are partially conjugated and therefore cannot rotate freely.

`-c`

tells AutoTors to add a column at the end of each ATOM line, showing the number of bonded nearest neighbours to each atom. If the "-h" flag is used as well, then the atoms with non-polar hydrogens are merged first, then the resulting merged structure is used to obtain this column of atom connectivities.

The `-m`, `-h` and `-a` flag may appear in any order. The `-o` flag must be given after the `-h` flag.

Placement of these flags should follow these two examples. Square brackets denote optional flags:

```
autotors [-h][-o][-a][-c] peptid.bnd peptid.pdbq peptid.out.pdbq
```

For SYBYL-mol2 input:

```
autotors -m [-h][-o][-a][-c] drug.mol2 drug.out.pdbq
```

9. Running AutoTors

There are two interactive phases in running **AutoTors**:

1) Root selection: After all the bonding data is read in and any cycles detected in the small molecule, the user designates which (adjacent) atoms are to be considered the ROOT. If cycles are detected in the molecule, the first part of designating the ROOT is to select a cycle for the ROOT by number *or* no cycle by entering `0`. In either case the user next has the opportunity of entering specific atoms by number (after entering '`a`' on a menu provided) or quitting (selection '`q`') when all the desired root atoms are entered. (At this point it is not necessary to specify *all* of the root atoms desired because the root will be expanded to include all atoms not in a BRANCH or TORS.)

2) Torsion selection: After the root has been specified, the program goes through the data and makes a list of possible torsions. These are listed as "possible torsions" and the user next must edit this list as desired. For example, if all the possible torsions detected are to be used, the user simply selects '`q`' and quits the torsions selection section. If a small number of the possible torsions are to be eliminated, the user selects 'd' and deletes the unwanted torsions. He stops deleting by entering '`q`'. If only a few of the possible torsions are desired, it is possible to select only these few (instead of eliminating a large number of them) by selecting '`s`' from the menu. This selection process is also ended by entering '`q`'.

Once either phase is ended by entering '`q`', it is not possible to change what has been selected. Instead, the user should abort the program with `<Ctrl>-C` and start again.

`MAX_TORS`: **AutoDock** is set up to allow a maximum number of torsions. If **AutoTors** detects more torsions than are permitted, a warning to that effect is given and it is up to the user to reduce the number of torsions, either by deleting or selecting the appropriate number of torsions. `MAX_TORS` is defined in the file "`autodock.h`"; if this definition is changed, the autodock-executable must be re-made, using the appropriate Makefile.

10. Adding Polar Hydrogens to the Macromolecule

When modelling hydrogen bonds explicitly, it is necessary to add polar hydrogens to the macro-

molecule also. Then the appropriate partial atomic charges can be assigned. This can be achieved by the user's preferred method, *e.g.* using **InsightII**, **Quanta**, **Sybyl**, **AMBER** or **CHARMm**. Alternatively, one of the shell scripts described in the Appendix can be used. The charged macromolecule must be converted to PDBQ format so that **AutoGrid** can read it.

Note that most modelling systems add polar hydrogens in a default orientation, typically assuming each new torsion angle is 0˚ or 180˚. Without some form of refinement, this can lead to spurious locations for hydrogen-bonds. One option is to relax the hydrogens and perform a molecular mechanics minization on the structure. Another is to use a program like "pol_h" which takes as input the default-added polar hydrogen structure, samples favourable locations for each movable proton, and selects the best position for each. This "intelligent" placement of movable polar hydrogens can be particularly important for tyrosines, serines and threonines.

11. Getting Started...

There are several Unix scripts available to help prepare default **AutoGrid** and **AutoDock** parameter files. They are described in more detail in the Appendix; see "prepare", "prepareII", "prepare-gpf+dpf" and "prepare-dpf". The user must check these defaults, to ensure they look reasonable. The user can adjust the default parameters according to what is required. Each parameter is described in the sections "AutoGrid Parameter File Format" and "AutoDock Parameter File Format".

12. AutoGrid Parameter File Format

The input file is often referred to as a "grid parameter file" or "GPF" for short. The scripts described in the appendices give these files the extension ".gpf". In the grid parameter file, the user must specify the following spatial attributes of the grid maps:

1. the center of the grid map;
2. the number of grid points in each of the *x*-, *y*- and *z*-directions; and
3. the separation or spacing of each grid point.

In addition, the pairwise-atomic interaction energy parameters must be specified. The following lines are required for each small molecule atom type, *Y*:

4. the grid map filename for atom type *Y*;
5. seven lines containing the non-bonded parameters for each pairwise-atomic interaction, in the following order: *Y*-C, *Y*-N, *Y*-O, *Y*-S, *Y*-H, *Y-X*, (*X* is any other atom type) and *Y-M* (*M* is a metal, say).

Using coefficients C_n, C_m, n and m, the pairwise interaction energy, $V(r)$ is given by:

$$V(r) \approx \frac{C_n}{r^n} - \frac{C_m}{r^m}$$

Alternatively, the user can specify r_{eqm}, ε, n and m. By default, the *Y-X* and *Y-M* lines are copies of the *Y*-H line. But in some systems, such as receptors which consist of DNA/protein complexes, both sulphur *and* phosphorus can be present. In this scenario, the *Y-X* line can be used for modelling interactions with receptor-phosphorus atoms. A very rough approximation for phosphorus parameters is to borrow those of carbon.

The "elecmap" line in the grid parameter file is the filename of the electrostatic potential grid map. The following parameter, "dielectric", if negative, indicates that the distance-dependent dielectric function of Mehler and Solmajer[3] will be used. If positive, however, the value of that number will be used as a constant dielectric. For example, if the value were 40.0, then a constant dielectric of 40 would be used.

The **AutoGrid** parameter file format is described below. The type of each argument is described using C-style, "%s" = a character string; "%d" = a (decimal) integer; and, "%f" = a floating point or real number.

AutoGrid Keywords

`receptor %s`

Macromolecule filename, in PDBQ format.

`gridfld %s`

The grid field filename, which will be written in a format readable by **AutoDock** and AVS[8]. The filename extension *must* be '.fld'.

`npts %d %d%d`

Number of *x*-, *y*- and *z*-grid points. Each *must* be an even integer number. When added to the central grid point, there will be an odd number of points in each dimension. The number of *x*-, *y*- and *z*-grid points need not be equal.

`spacing %f`

The grid point spacing, in Å (see the diagram on page 8). Grid points must be uniformly spaced in AutoDock: this value is used in each dimension.

`gridcenter %f %f %f`
`gridcenter auto`

8. "AVS" stands for "Application Visualization System"; AVS is a trademark of Advanced Visual Systems Inc., 300 Fifth Avenue, Waltham, MA 02154.

The user can explicitly define the center of the grid maps, respectively the *x*, *y* and *z* coordinates of the center of the grid maps (units: Å, Å, Å.) Or the keyword "auto" can be given, in which case **AutoGrid** will center the grid maps on the center of mass of the macromolecule.

`types %s`

1-letter names of the atom types present in the small molecule; *e.g.* if there are carbons, nitrogens, oxygens and hydrogens, then this line will be "`CNOH`"; there are no delimiters.

`map %s`

Filename of the grid map, for ligand atom type Y; the extension is usually ".map".

`nbp_coeffs %f %f %d %d`

Either "nbp_coeffs" or "nbp_r_eps" keywords can be used to define Lennard-Jones or hydrogen bond interaction energy parameters. The keyword "nbp_coeffs" specifies coefficients and exponents, in the order "C_n C_m *n m*", delimited by spaces; *n* and *m* are integer exponents. The units of C_n and C_m must be kcal mol^{-1} Å^n and kcal mol^{-1} Å^m respectively; *n* and *m* have no units.

`nbp_r_eps %f %f %d %d`

Alternatively, the user can employ "nbp_r_eps" to specify the equilibrium distance and well depth, epsilon, for the atom pair. The equilibrium separation has units of Å and the well depth, epsilon, units of kcal mol^{-1}. The integer exponents *n* and *m* must be specified too.

In either case, the order of the parameters must be: *Y*-C, *Y*-N, *Y*-O, *Y*-S, *Y*-H, *Y*-X, and *Y*-M. Repeat 1 "map" line and the 7 "nbp_coeffs"or "nbp_r_eps" lines, for each atom type, *Y,* present in the small molecule being docked.

`elecmap %s`

Filename for the electrostatic potential energy grid map to be created; filename extension '.map'.

`dielctric %f`

Dielectric funtion flag: if negative, **AutoGrid** will use *distance-dependent* dielectric of Mehler and Solmajer[3]; if the float is positive, **AutoGrid** will use this value as the dielectric constant.

`fmap %s`

(Optional.) Filename for the so-called "floating" grid map[9]; filename extension '.map'. In such floating grids, the scalar at each grid point is the distance to the nearest atom in the receptor. These values could be used to guide the docking ligand towards the receptor's surface, thus avoiding non-interesting, empty regions.

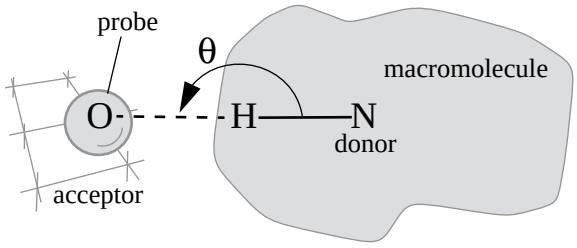
9. This grid map is not used in AutoDock 2.4; its utility is under investigation, and may be included in a later version.

Example AutoGrid Parameter File

An example **AutoGrid** parameter file is given below:

```
receptor 3ptb.pdbq         #macromolecule
gridfld 3ptb.maps.fld      #grid.data.file
npts 60 60 60              #numxyzpoints
spacing .375               #spacing/Angstroms
gridcenter -1.930 14.070 16.224#center_of_grids or auto
types CNH                  #atom.type.names
map 3ptb.C.map             #atomic.affinity.map
nbp_r_eps  4.00 0.1500  12   6#C-C non-bond Rij & epsilonij
nbp_r_eps  3.75 0.1549  12   6#C-N non-bond Rij & epsilonij
nbp_r_eps  3.60 0.1732  12   6#C-O non-bond Rij & epsilonij
nbp_r_eps  4.00 0.1732  12   6#C-S non-bond Rij & epsilonij
nbp_r_eps  3.00 0.0548  12   6#C-H non-bond Rij & epsilonij
nbp_r_eps  3.00 0.0548  12   6#C-H non-bond Rij & epsilonij
nbp_r_eps  3.00 0.0548  12   6#C-H non-bond Rij & epsilonij
map 3ptb.N.map             #atomic.affinity.map
nbp_r_eps  3.75 0.1549  12   6#N-C non-bond Rij & epsilonij
nbp_r_eps  3.50 0.1600  12   6#N-N non-bond Rij & epsilonij
nbp_r_eps  3.35 0.1789  12   6#N-O non-bond Rij & epsilonij
nbp_r_eps  3.75 0.1789  12   6#N-S non-bond Rij & epsilonij
nbp_r_eps  1.90 5.0000  12  10#N-H non-bond Rij & epsilonij
nbp_r_eps  1.90 5.0000  12  10#N-H non-bond Rij & epsilonij
nbp_r_eps  1.90 5.0000  12  10#N-H non-bond Rij & epsilonij
map 3ptb.H.map             #atomic.affinity.map
nbp_r_eps  3.00 0.0548  12   6#H-C non-bond Rij & epsilonij
nbp_r_eps  1.90 5.0000  12  10#H-N non-bond Rij & epsilonij
nbp_r_eps  1.90 5.0000  12  10#H-O non-bond Rij & epsilonij
nbp_r_eps  2.50 1.0000  12  10#H-S non-bond Rij & epsilonij
nbp_r_eps  2.00 0.0200  12   6#H-H non-bond Rij & epsilonij
nbp_r_eps  2.00 0.0200  12   6#H-H non-bond Rij & epsilonij
nbp_r_eps  2.00 0.0200  12   6#H-H non-bond Rij & epsilonij
elecmap 3ptb.e.map         #electrostatic.PE.map
dielectric -1.             #distance-dep.diel=-1,constant>0
#fmap 3ptb.f.map           #floating.grid
```

Note how hydrogen bonding is defined for oxygens. The ideal hydrogen bond would have an angle, θ, of 180˚ between the acceptor, the polar hydrogen and the donor, thus:



As θ decreases, the strength of the hydrogen bond diminishes. There are no hydrogen bonds when θ is 90˚ or less.

If a line in the parameter file contains a '10' in the fourth column, **AutoGrid** will treat this atom-pair as hydrogen bonding. So in the example above, the last 3 lines in the "mcp2_O.map" block will be treated as hydrogen bonds. **AutoGrid** scans for any polar hydrogens in the macromolecule. The vector from the hydrogen-donor, along with the vector from the probe-atom at the current grid point, are used to calculate the directional attenuation of the hydrogen bond. In this example, **AutoGrid** will calculate H-bonds between O-H, O-*X* and O-*M*.

13. Running AutoGrid

AutoGrid requires an input grid parameter file, which usually has the extension ".gpf". The command is issued as follows:

```
% autogrid -p molecule.gpf -l molecule.glg &
```

where '-p molecule.gpf' specifies the grid parameter file, and '-l molecule.glg' the log file output during the grid calculation. The '&' ensures that the this job will be run in the background. This whole line can be prefixed with the 'nice' command to ensure other processes are not unduly affected. The log file will inform the user of the maximum and minimum energies found during the grid calculations. **AutoGrid** writes out the grid maps in ASCII form, for readability and portability; **AutoDock** expects ASCII format grid maps. The first six lines of each grid map hold header information which describe the spatial features of the maps and the files used or created. These headers are checked by **AutoDock** to ensure that they are appropriate for the requested docking. The remainder of the file contains grid point energies, written as floating point numbers, one per line. They are ordered according to the nested loops z(y(x)). A sample header is shown below:

```
GRID_PARAMETER_FILE vac1.nbc.gpf
GRID_DATA_FILE 4phv.nbc_maps.fld
MACROMOLECULE 4phv.new.pdbq
SPACING 0.375
NELEMENTS 50 50 80
CENTER -0.026 4.353 -0.038
125.095596
123.634560
116.724602
108.233879
:
```

As well as the grid maps, **AutoGrid** creates two AVS-readable files, with the extensions '.fld', and '.xyz'. The former is a *field file* summarizing the grid maps, and the latter describes the spatial extent of the grids in Cartesian space. (To read the grid maps into AVS, use a "read field" module.)

The '-o' flag can be used on the **AutoGrid** command line to signify that the '.pdbq' file specified in the grid parameter file is in 'old' PDBQ format (charges are stored in columns 55-61).

14. AutoDock Parameter File Format

AutoDock 2.4 has a completely new and upgraded interface, based on keywords. This new interface is intended to make it easier for the user to set up and control a docking job, and for the programmer to add new commands and functionality. The input file is often referred to as a "docking parameter file" or "DPF" for short. The scripts described in the appendices give these files the extension ".dpf".

Note: All delimiters where needed are white spaces. Default values, where applicable, are given in square brackets [thus]. A comment must be prefixed by the "#" symbol, and can be placed at the end of a parameter line, or on a line of its own. Once again, the type of each keyword argument is described using the C-standard, where:

%s = is an alphanumeric string, and in most cases, a valid filename;
%d = is a decimal integer; and
%f = is a floating point or real number.

Although in theory it should be possible to give these keywords in any order, not every possible combination has been tested, so it may be wise to stick to the following order.

AutoDock Keywords

`seed %ld`
`seed time`

Each job can be seeded with either a user-defined or a time-dependent random-number generator seed. The first form explicitly defines the seed-value for the random number generation. The second uses a keyword, "time", to obtain the number of seconds since the epoch. The epoch is referenced to 00:00:00 CUT (Coordinated Universal Time) 1 Jan 1970.

`types %s`

Atom names for all atom types present in small molecule. Each must be a single character, and only one of: C, N, O, S, H, X, or M. The maximum number of characters allowed in this line is ATOM_MAPS, which is defined in the "autodock.h" include file. Do not use any spaces to delimit the types: they are not needed.

`fld %s`

Grid data field file created by **AutoGrid** and readable by **AVS** (must have the extension ".fld").

`map %s`

Filename for the first **AutoGrid** affinity grid map of the 1st atom type. This keyword plus filename

must be repeated for all atom types in the order specifed by the “types” command. In all map files a 6-line header is required, and energies must be ordered according to the nested loops z(y(x)).

`map %s`

Filename for the electrostatics grid map. 6-line header required, and energies must be ordered according to the nested loops z(y(x)).

`move %s`

Filename for the ligand to be docked. This contains most importantly, atom names, xyz-coordinates, anb partial atomic charges in PDBQ format. (Filename extension should be “.pdbq”).

`about %f %f %f`

Use this keyword to specify the center of the ligand, *about* which rotations will be made. (The coordinate frame of reference is that of the ligand PDBQ file.) Usually the rotation center of the ligand is the mean *x,y,z*-coordinates of the molecule. Inside **AutoDock**, the “about” xyz-coordinates are *subtracted* from each atom’s coordinates in the input PDBQ file. So internally, the ligand’s coordinates become centered at the origin. Units: Å, Å, Å.

`tran0 %f %f %f`
`tran0 random`

Initial coordinates for the center of the ligand, in the same frame of reference as the receptor’s grid maps. The ligand, which has been internally centered using the “about” coordinates, has the xyz-coordinates of the initial translation “tran0 x y z” *added* on. Every run starts the ligand from this location.

Alternatively, the user can just give the keyword “random” and **AutoDock** will pick random initial coordinates instead.

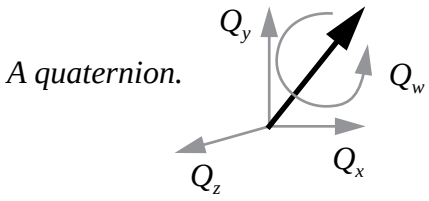
If there are multiple runs defined in this file, using the keyword “runs”, then each new run will begin at this same location.

The user *must* specify the absolute *starting* coordinates for the ligand, used to start each run. The user should ensure that the small molecule, when translated to these coordinates, still fits within the volume of the grid maps. If there are some atoms which lie outside the grid volume, then **AutoDock** will automatically correct this, until the small molecule is pulled completely within the volume of the grids. (This is necessary in order to obtain complete information about the energy of the initial state of the system.) The user will be notified of any such changes to the initial translation by **AutoDock**. (Units: Å, Å, Å.)

`quat0 %f %f %f %f`
`quat0 random`

Respectively: Q_x, Q_y, Q_z, Q_w. Initial quaternion (applied to small molecule) - Q_x, Q_y, Q_z define the unit vector of the direction of rigid body rotation [1, 0, 0], and Q_w defines the angle of rotation

about this unit vector, in ˚ [0˚]. (Units: none,none,none, ˚.)



Alternatively, the user can just give the keyword “random” and **AutoDock** will pick a random unit vector and a random rotation (between 0˚ and 360˚) about this unit vector. Each run will begin at this same random rigid body rotation.

`ndihe %d`

Number of dihedrals or rotatable bonds in the small molecule. This may be specifed only if rotatable bonds have been defined using ROOT, BRANCH, TORS *etc*. keywords in the “`.pdbq`” file named on the “`move`” line. If this keyword is used, then the next keyword, **`dihe0`**, must also be specified. Note that if **`ndihe`** and **`dihe0`** are not specified and there are defined torsions in the ligand PDBQ file, **AutoDock** assumes that the chi_1, chi_2, chi_3, *etc.* are all zero, and does not change the initial ligand torsion angles.

`dihe0 %f %f %f ...`

Initial **relative** dihedral angles; there must be **`ndihe`** floating point numbers specified on this line. Each value specified here will be added to the corresponding torsion angle in the input PDBQ file, at the start of each run. Torsion angles are only specified by two atoms, so the definition of rotations is relative. Units: ˚.

`tstep %f`

Maximum translation step [0.2 Å]. Units: Å.

`qstep %f`

Maximum quaternion rotation step [5.˚]. Units: ˚.

`dstep %f`

Maximum dihedral step [5.˚]. Units: ˚.

`trnrf %f`

Per-cycle reduction factor for translations [1.].

`quarf %f`

Per-cycle reduction factor for quaternions [1.].

`dihrf %f`

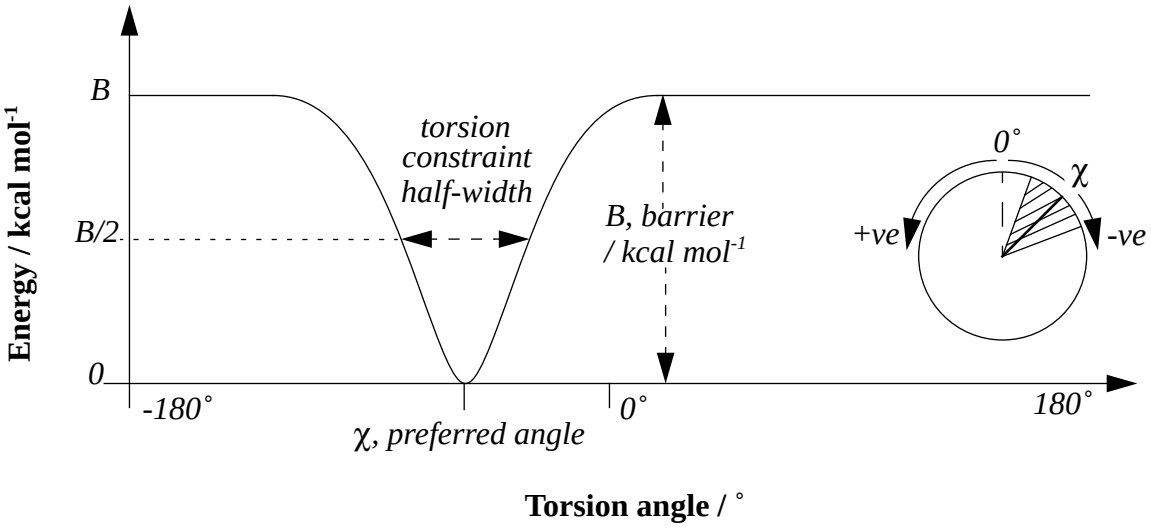
Per-cycle reduction factor for dihedrals [1.].

`barrier %f`

(Optional) This defines the energy-barrier height applied to constrained torsions. When the torsion is at a preferred angle, there is no torsion penalty: this torsion's energy is zero. If the torsion angle falls within a disallowed zone, however, it can contribute up to the full barrier energy. Since the torsion-energy profiles are stored internally as arrays of type 'unsigned short', only positive integers between 0 and 65535 are allowed. [10000].

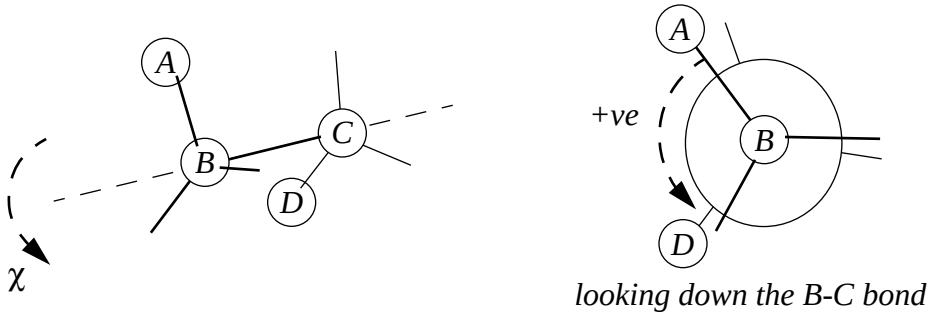
`gausstorcon %d %f %f`

(Optional) Adds a constraint to a torsion. The torsion number is identified by an integer. This identifier comes from the list at the top of the AutoTors-generated input ligand PDBQ file (on the REMARK lines). An energy profile will be calculated for this torsion. An inverted Gaussian bell curve is added for each new constraint. To completely specify each Gaussian, two floating point numbers are needed: the *preferred angle* and the *half-width* respectively (both in degrees). Note that the preferred angle should be specified in the range -180˚ to +180˚; numbers outside this range will be wrapped back into this range. This angle, χ, is ***relative*** to the original torsion angle in the input structure. The *half-width* is the difference between the two angles at which the energy is half the barrier (B/2 in the diagram above). The smaller the half-width, the tighter the constraint.



If you wish to constrain to absolute-valued torsion angles, it will be neccessary to zero the initial torsion angles in the ligand, before input to **AutoTors**. The problem arises from the ambiguous 2-atom definition of the rotatable bond *B-C*. To identify a torsion angle unambiguously, 4 atoms

must be specified: *A-B-C-D*:



looking down the B-C bond

The sign convention for torsion angles which we use is anti-clockwise (counter-clockwise) are positive angles, clockwise negative. In the above diagram, looking down the bond *B-C*, the dihedral angle *A-B-C-D* would be positive.

There is no limit to the number of constraints that can be added to a given torsion. Each new torsion-constraint energy profile is combined with the pre-exisiting one by selecting the minimum energy of either the new or the existing profiles.

`showtorpen`

(Optional) (Use only with "gausstorcon") This switches on the storage and subsequent output of torsion energies. During each energy evaluation, the penalty energy for each constrained torsion, as specified by the "gausstorcon" command, will be stored in an array. At the end of each run, the final docked conformation's state variables are output, but with this command, the penalty energy for each torsion will be printed alongside its torsion angle.

`hardtorcon %d %f %f`

(Optional) This command also adds a torsion constraint to the %d-th torsion, as numbered in the AutoTors-generated REMARKs. The first float defines the *preferred relative angle*, and the second specifies the *full width* of the allowed range of torsion angles (both in degrees). This type of torsion constraint is hard because the torsion is never allowed to take values beyond the range defined. For example, "hardtorcon 3 60. 10." would constrain the third torsion to values between 55˚ and 65˚.

`intnbp_coeffs %f %f %d %d`

Respectively: C_n; C_m; *n; m.* These are the internal pairwise non-bonded energy parameters for flexible ligands, where:

$$V(r) \approx \frac{C_n}{r^n} - \frac{C_m}{r^m}$$

These parameters are needed even if no rotatable bonds were defined in the ligand-PDBQ file. They are only used in the internal energy calculations for the ligand and must be consistent with

those used in calculating the grid maps. (Units: kcal mol^{-1} $Å^{n}$; kcal mol^{-1} $Å^{m}$; none; none, respectively)

`intelec`

(Optional) Internal electrostatic energies will be calculated; the products of the partial charges in each non-bonded atom pair are pre-calculated, and output. Note that this is only relevant for flexible ligands.

`rt0 %f`

Initial "annealing temperature"; this is actually the absolute temperature multiplied by the gas constant *R*. [500. cal mol^{-1}]. $R = 8.314$ J mol^{-1} $K^{-1} = 1.987$ cal mol^{-1} K^{-1}. (Units: cal mol^{-1}.)

`rtrf %f`

Annealing temperature reduction factor, *g* [0.95 $cycle^{-1}$]. See the equation at the bottom of page 5. At the end of each cycle, the annealing temperature is multiplied by this factor, to give that of the next cycle. This must be positive but < 1 in order to cool the system. Gradual cooling is recommended, so as to avoid "*simulated quenching*", which tends to trap systems into local minima.

`linear_schedule`
`schedule_linear`
`linsched`
`schedlin`

These keywords are all synonymous, and instruct **AutoDock** to use a linear or *arithmetic* temperature reduction schedule during simulated annealing. Unless this keyword is given, a *geometric* reduction schedule is used, according to the **`rtrf`** parameter just described. If the linear schedule is requested, then any **`rtrf`** parameters will be ignored. The first simulated annealing cycle is carried out at the annealing temperature **`rt0`**. At the end of each cycle, the temperature is reduced by (**`rt0/cycles`**). The advantage of the linear schedule is that the system samples evenly across the temperature axis, which is vital in entropic calculations. Geometric temperature reduction schedules on the other hand, under-sample high temperatures and over-sample low temperatures.

`runs %d`

Number of automated docking runs [1].

`cycles %d`

Number of temperature reduction cycles [50].

`accs %d`

Maximum number of accepted steps per cycle [100].

`rejs %d`

Maximum number of rejected steps per cycle [100].

`select %d`

State selection flag. This can be either **`m`** for the *minimum* state, or **`l`** for the *last* state found during each cycle, to begin the following cycle [**`m`**].

`outlev %d`

Diagnostic output level. 0 = no output, 1 = minimal output, 2 = full state output at end of each cycle; 3 = detailed output for each step. [1].

`rmstol %f`

RMS deviation tolerance for cluster analysis or 'structure binning' [0.5Å], 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. The lowest energy representative from each cluster is output in PDBQ format to the log file. To keep the original residue number of the input ligand PDBQ file, use the '**`-k`**' flag; otherwise the cluster-representatives are numbered incrementally from 1. (Units: Å).

`rmsnosym`

When more than one run is carried out in a given job, cluster analysis or 'structure binning' will be performed, based on structural rms difference, ranking the resulting families of docked conformations in order of increasing energy. The default method for structure binning allows for atom similarity, as in a tertiary-butyl which can be rotated by +/-120˚, but in other cases it may be desirable to bypass this similar atom type checking and calculate the rms on a one-for-one basis. The symmetry checking algorithm scans all atoms in the reference structure, and selects the nearest atom of identical atom type to be added to the sum of squares of distances. This works well when the two conformations are very similar, but this assumption breaks down when the two conformations are translated significantly. Symmetry checking can be turned off using the **`rmsnosym`** command; omit this command if you still want symmetry checking.

`trjfrq %d`

Output frequency, *n*, for trajectory of small molecule, in steps [0]. If $n = 0$, then no trajectory states will be output; otherwise, every n^{th} state will be output. The *state* consists of 7 floats describing the *x,y,z* translation, the *x,y,z* components of the quaternion unit vector, the angle of rotation about the quaternion axis; and any remaining floats describing the torsions, in the same order as described in the input small molecule PDBQ file).

`trjbeg %d`

Begin sampling states for trajectory output at this cycle. [1]

`trjend %d`

End trajectory output at this cycle. [50].

`trjout %s`

Trajectory filename [<smlmol.pdbq>.trj]. AutoDock will write out state variables to this file every

"trjfrq" steps. Use the "traj" command in AutoDock's command mode to convert this trajectory of state-variables into a series of PDB frames. The "traj" command is described in § "Using the Command Mode in AutoDock"; see also § "Trajectory Files".

`trjsel %s`

Trajectory output flag, can be either '**A**' or '**E**'; the former outputs only *accepted* steps, while the latter outputs *either* accepted or rejected steps.

`watch %s`

Optional) Creates a "watch" file for real-time monitoring of an *in-progress* simulated annealing job. This works only if the "trjfrq" parameter is greater than zero.

The watch file will be in PDB format, so give a ".pdb" extension. This file has an exclusive lock placed on it, while AutoDock is writing to it. Once the file is closed, the file is unlocked. This can signal to a watching visualization program that the file is complete and can now be read in, for updating the displayed coordinates. This file is written at exactly the same time as the trajectory file is updated

`extnrg %f`

External grid energy [1000.] assigned to any atoms that stray outside the volume of the grid during a docking. Units: kcal mol^{-1}.

`rmsref %s`

The RMS deviation of any conformations generated during the docking will be calculated by comparing the coordinates in the file specified by this command. This tends to be useful when the experimentally determined complex structure is known. The order of the atoms in the PDB file specified by this command must match that in the input PDBQ file given by the **`move`** command. These values of RMS will be output in the last column of the final PDBQ records, after the clustering has been performed.

`cluster %s`

(Clustering multi-job output only.) **AutoDock** will go into 'cluster mode'. Use this command only to perform cluster analysis on the combined output, <*PDBQfilename*>, of several jobs. This command can be very useful when many jobs have been distributed to several machines and run in 'parallel'. The docking parameter file will need the following keywords: **`rmstol`** and **`types`**; and optionally write_all_cluster_members and/or **`rmsnosym`**. It is necessary to `grep` the REMARKS along with the ATOM records, since **AutoDock** parses the REMARKS to determine what the energy of that particular conformation was. See the second example **`.dpf`** below.

`write_all_cluster_members`

(Custering multi-job output only.) This command is used only with the **`cluster`** command, to write out all members of each cluster instead of just the lowest energy from each cluster. This affects the cluster analysis PDBQ output at the end of each job.

Example AutoDock Parameter File

An example of a commented **AutoDock** parameter file is given below:

```
seed random
types CNOH                 # atom type names
fld 4phv.nbc_maps.fld# grid data file
map 4phv.nbc_C.map         # C-atomic affinity map
map 4phv.nbc_N.map         # N-atomic affinity map
map 4phv.nbc_O.map         # O-atomic affinity map
map 4phv.nbc_H.map         # H-atomic affinity map
map 4phv.nbc_e.map         # electrostatics map

move xk263pm3.pdbq         # small molecule
about -5.452 -8.626 -0.082 # small molecule center
tran0 -5.452 -8.626 -0.082 # initial coordinates/A
quat0 1. 0. 0.   0.        # initial quaternion:unit-vector(qx,qy,qz);angle/deg(qw)
ndihe 10                   # number of rotatable bonds
dihe0 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. # initial dihedrals/deg
tstep 0.2                  # translation step/A
qstep 5.                   # quaternion step/deg
dstep 5.                   # torsion step/deg
trnrf 1.                   # trans reduction factor/per cycle
quarf 1.                   # quat reduction factor/per cycle
dihrf 1.                   # tors reduction factor/per cycle

intnbp 1272653.000  1127.684  12  6 # C-C internal energy non-bond parameters/Cn,Cm,n,m
intnbp  610155.100   783.345  12  6 # C-N internal energy non-bond parameters/Cn,Cm,n,m
intnbp  588883.800   633.754  12  6 # C-O internal energy non-bond parameters/Cn,Cm,n,m
intnbp   88604.240   226.910  12  6 # C-H internal energy non-bond parameters/Cn,Cm,n,m
intnbp  266862.200   546.765  12  6 # N-N internal energy non-bond parameters/Cn,Cm,n,m
intnbp  249961.400   445.918  12  6 # N-O internal energy non-bond parameters/Cn,Cm,n,m
intnbp   39093.660   155.983  12  6 # N-H internal energy non-bond parameters/Cn,Cm,n,m
intnbp  230584.400   368.677  12  6 # O-O internal energy non-bond parameters/Cn,Cm,n,m
intnbp   38919.640   124.049  12  6 # O-H internal energy non-bond parameters/Cn,Cm,n,m
intnbp    1908.578    46.738  12  6 # H-H internal energy non-bond parameters/Cn,Cm,n,m

rt0  500.                  # initial RT
rtrf 0.95                  # RT reduction factor/per cycle

runs 10                    # number of runs
cycles 50                  # cycles
accs   100                 # steps accepted
rejs   100                 # steps rejected
select m                   # minimum or last

outlev 1                   # diagnostic output level

rmstol 0.5                 # cluster tolerance/A
#rmsnosym                  # no symmetry checking in RMS calc.

trjfrq 7500                # trajectory frequency
trjbeg 45                  # start trj output at cycle
trjend 50                  # end trj output at cycle
trjout xk263pm3.trj        # trajectory file
trjsel E                   # A=acc only;E=either acc or rej
```

```
extnrg 0.0                # external grid energy
```

In this case, the small molecule file '`xk263pm3.pdbq`' has been defined such that it contains 10 rotatable bonds. The docking will be sampled every 7500 steps, from cycle 45 to cycle 50. Either accepted or rejected states will be output. The trajectory file '`xk263pm3.trj`' will hold the state information required to generate the coordinates later on. The external grid energy is set to 0.0, which can allow greater freedom for ligand rotations during docking.

The next example `.dpf` shows how to use the cluster mode in **AutoDock**. The PDBQ files containing the final docked conformations have been extracted from the **`.dlg`** **AutoDock** log files (using the UNIX `grep` command), and stored together in "`vac1.new.nrg.pdb`". The tolerance for the RMS deviation is set to 1.5Å, so only conformations with this RMS or less will be grouped into the same cluster. All cluster members will be written out, instead of just the lowest energy representative of each.

```
types CNOH                # atom_type_names
rmstol 1.5                # cluster_tolerance/A
write_all_cluster_members
cluster vac1.new.nrg.pdb  # structure binning
```

---

15. Running AutoDock

Once the grid maps have been prepared by **AutoGrid**, and the docking parameter file is ready, the user is ready to run an **AutoDock** job. A docking is initiated using the following command:

`autodock [-o][-k][-i][-u][-t] -p molecule.dpf [-l molecule.dlg] &`

Input parameters are specified by '**`-p`** `molecule.dpf`', and the log file containing results of the docking is defined by '**`-l`** `molecule.dlg`'. This is the normal usage of **AutoDock**, and performs a standard docking calculation.

`-o`
can be added to the command line, to signify that the input file specified in the docking parameter file is in *old* PDBQ format, with charges in columns 55-61.

`-k`
keep the original residue number of the input ligand PDBQ file. Normally **AutoDock** re-numbers the starting position to residue-number 0, and any cluster-representatives are numbered incrementally from 1, according to their rank (rank 1 is the lowest energy cluster).

`-i`
used to *ignore* any grid map header errors that may arise due to conflicting filenames. This overrides the header checking that is normally performed to ensure compatible grid maps are being used.

`-u`
returns a message describing the command line *usage* of **AutoDock**.

`-t`
instructs **AutoDock** to parse the PDBQ file to check the *torsion* definitions, and then stop.

The Unix script "**`job`**" can be used to submit an **AutoDock** job, and then perform additional post-processing, such as profiling, extracting job-information and creating a field file for AVS display of the docked results. See the Appendix for more details.

16. Using the Command Mode in AutoDock

AutoDock can be run in "command mode", using the "**`-c`**" flag thus:

```
% autodock -p molecule.dpf -l molecule.clg -c &
```

When **AutoDock** has read in the grid maps specified in "`molecule.dpf`", the program gives the message "COMMAND MODE" and waits for the user to issue a command from the standard input. These commands are described in more detail below. An alternative method of using the command mode is to edit a file containing the commands you wish **AutoDock** to execute (`command.file`) and channel the output to a file (`command.output`), thus:

```
% autodock -p molecule.dpf -l molecule.clg -c < command.file > command.output &
```

AutoDock can be used in a UNIX pipe command. This is valuable when an alternative search procedure is desired. Here, `alt_search_proc` issues commands to the standard output, and reads the results from the standard input. In this case, **AutoDock** is behaving as an energy server for `alt_search_proc`, the alternative search-procedure program.

There are eight recognized commands: **AutoDock**'s command interpreter is not case sensitive.

"eval"	Evaluate this state's total energy.
"epdb"	Evaluate the energy of the named PDBQ file.
"outc"	Output the last state's PDB-formatted Cartesian coordinates.
"oute"	Output (non-bond and electrostatic) energy breakdown, by atom.
"traj''	Convert trajectory file to PDB-formatted Cartesian coordinates.
"stop", "exit", "quit"	Stop the AutoDock command mode interpreter.

Eval

Evaluates the total energy of a state defined by the subsequent state variables. This command utilizes the trilinear interpolation routine in **AutoDock** along with the supplied grid maps defined in the parameter file specified after the '`-p`' flag to return this energy. The internal energy of the small molecule is also taken into account, as dictated by the values of the torsion angles supplied in the *ntor* lines following this `eval` command line; *ntor* is the number of torsion angles defined in the small molecule PDBQ file, as described in the section "Defining Torsions in AutoDock". The usage of this command is:

`eval %f %f %f %f %f %f %f` *} T_x, T_y, T_z, Q_x, Q_y, Q_z, Q_w (in °)*
`%f` *} i^{th} torsion angle, in °.*
:
: *ntor* lines.

where: T_x, T_y, T_z are the coordinates of the center of rotation of the small molecule; Q_x, Q_y, Q_z is the unit vector describing the direction of rigid body rotation, about which a rotation of angle Q_w degrees will be applied. The following *ntor* lines hold the torsion angles in degrees, given in the same order as described in the **AutoDock** log file.

Epdb

Calculates the energy of the molecule provided in the PDBQ file, thus:

`epdb filename.pdbq`

where: "filename.pdbq" is the PDBQ formatted coordinates of a molecule for which the interaction energy with the macromolecule will be returned. The '-o' flag supplied at the **AutoDock** execution line specifies the old format of PDBQ, with charges in columns 55-61; otherwise it is assumed that the charges are in columns 71-76.

This command is useful when the state variables for a given molecule are not known, e.g. the x-ray crystal conformation of the small molecule.

Outc

Returns the coordinates of the small molecule at its current transformed position (in the form of a PDB REMARK). The *x*,*y*,*z* coordinates will be determined by the state variables supplied to the `eval` command.

Oute

Returns the total internal energy of the small molecule and the total energy of the complex, at the current state variables. These two REMARK lines are written in PDB format, to the command output channel and the log file.

Traj

Convert a ".trj" file into PDBQ format. Usage:

`traj filename.trj`

where "filename.trj" is the trajectory file written out by an earlier run of **AutoDock**. This trajectory file contains the state variables for the states sampled during the docking simulation. The torsions are assumed to be in exactly the same order as the input ligand PDBQ file. The torsion angles in the trajectory file are *relative* to the the latter's conformation.

See also the Appendix 20, script "runtrj"; and the next section, "Trajectory Files".

Stop, Exit, Quit

Halts the execution of AutoDock. A value of 0 is returned by the program, and the message "autodock: Successful Completion" is written to the log file and standard error. Timing information is also written. Note: "stop", "exit" and "quit" are synonymous.

17. Trajectory Files

A trajectory (of state variables) can be written out during a normal docking simulation, if the trajectory-frequency (set by the keyword "trjfrq") in the docking parameter file is greater than zero. This value defines the output frequency, in steps, for states sampled during the run. The default trajectory filename extension is ".trj" . These *state variables* are all that is needed to regenerate the coordinates of the small molecule. The trajectory control parameter (either "A" or "E") allows the user to record only *accepted* moves (A); or, moves which are *either* accepted or rejected (E). Just for information, a sample ".trj" trajectory file is shown below; you will not need to create such files (unless you feel like creating an animation!):

```
ntorsions 2
run 1
cycle 1
temp 300.000000
state 1 A -3.745762 -1.432243 -9.518171 23.713793 23.076145 0.713534 -0.023818 0.700216
30.606248
-4.894825
2.661499
:
:
state 6 R -12.679995 -1.452641 -9.259430 21.634645 23.135242 0.653369 -0.440832 0.615448
39.127316
-31.636299
10.261519
state 7 a -8.746072 -1.458231 -9.080998 21.356874 23.325665 0.648312 -0.448577 0.615200
41.075955
-37.935175
```

```
11.918847
:
```

There are several keywords: “run” and “cycle” are self-explanatory; “ntorsions” is the total number of changing torsions in the ligand; “temp” is the annealing temperature for all subsequent entries, unless otherwise stated. Each “state” record has the format:

`state nstep acc_rej_code e_total e_internal x y z qx qy qz qw`

where:

`nstep` = the number of the step, within this cycle;

`acc_rej_code` = ‘`A`’ = an accepted move whose energy was **lower** than its preceeding state;

= ‘`a`’ = an acepted move whose energy was **higher** than its preceeding state, which nevertheless passed the *Monte Carlo* probability test, at this temperature;

= ‘`R`’ = a rejected move.

= ‘`e`’ = an edge-hit, also a treated as a rejected move.

`e_total` = total energy of the system, small molecule + macromolecule;

`e_internal` = internal energy of small molecule only;

`x,y,z` = translation of small molecule center;

`qx,qy,qz,qw` = quaternion, which describes the small molecule’s orientation;

In order to get a coordinate-based trajectory file, for visualization, the command mode of **AutoDock** must be used to regenerate the coordinates from the state variables. Use the “traj” command with the name of the pre-calculated trajectory file. For example, suppose there is a command file called “trj.conv.com” that contains:

```
traj ligand.trj
stop
```

AutoDock would be executed a second time using the following command,

`autodock -p ligand.dpf -l ligand.trj.conv.log -c < trj.conv.com > trj.conv.out`

18. Evaluating the Results of a Docking

At the end of an **AutoDock** execution, in which more than one run was performed, the program outputs a list of clusters and their energies. The clustering or *structure binning* of docked confor-

mations is determined by the tolerance specified in Å by the **`rmstol`** keyword. The best representative from each cluster (that with the lowest energy) is written out in PDBQ format at the end of the log file.

These structures can be read into any appropriate molecular modelling system and the results compared, where possible, with the experimental data. The table of ranked clusters shows the final docked energy for each conformation, and the RMS difference between the lowest energy member of the group and every other member. The RMS for the lowest member of the group is by definition zero. After this table, the structures are output in PDBQ format. Each conformation has a set of REMARK records, one of which describes the RMS difference between itself and the coordinates specified in the original input PDBQ file.This can be useful for comparing how close each docked conformation is to the experimentally determined position.

19. Visualizing Grid Maps and Trajectories Using AVS

Grid maps can be visualised in **AVS** by using a *read field* module. The user must specify the '`.fld`' file that was created by **AutoGrid**, in order to read in the grid maps. An *extract scalar* module selects the grid map of interest, *e.g.* carbon affinity or electrostatics. The resulting grid map data can be analyzed using *arbitrary slicer* and *isosurface* modules, in order to examine cross sections and iso-energy contours respectively. Negative energy contours are most informative for the atomic affinity grid maps, since they reveal favorable regions of binding.

Trajectories can be read into **AVS** also using the *read field* module. The trajectory file is essentially a set of "stacked PDB frames", and must be read in as a two dimensional field (being the number of atoms in the small molecule, and the number of frames in the trajectory file). By paging through this field, using the *orthogonal slicer*, continuous replay of the trajectory can be achieved using an *animate integer* module to control which PDBQ frame is selected by the orthogonal slicer. This animates the sequence of sampled states and allows the user to view in real time the progress of the docking simulation.

APPENDIX 20. Shell Scripts and Auxilliary Tools

cartopdbq

Usage: **`cartopdbq ligand.car > ligand.pdbq`**

Converts from Biosym InsightII .car format to PDBQ format

check-qs

Usage: **`check-qs molecule`**
needs: **`molecule.pdbq`**
creates: **`molecule.err`**

Checks partial atomic charges in PDBQ file; any non-integral charges are reported.

clamp

Usage: **`clamp grid.map > grid.map.NEW`**

Clamps any **AutoGrid** map values that exceed ECLAMP (normally set to 1000.0)

deftors

Usage: **`deftors ligand`**
needs: **`ligand.mol2`**
creates: **`ligand.pdbq, ligand.err, ligand.bnd`** *and* **`ligand.bnd.pdbq`**

Sets up rotatable bonds for **AutoDock**. This script launches **AutoTors**, with the -a, -h and -m flags; checks the charges in the output, with `check-qs`; creates a .`bnd` file with `pdbqtobnd`; and creates an **AVS** .`fld` file using `mkavsheader` for trajectory viewing later on.

dpf-gen

Usage: **`dpf-gen ligand.pdbq > ligand.dpf`**

Generates a default **AutoDock** docking parameter file. You must edit the file before using it. In particular, you will need to edit the filename stem on the `trjout` line.

extjobinfo

Usage: **`extjobinfo file.dlg > file.dlg.inf`**

Extracts information about a particular docking job. Columns contain: cycle, run, annealing temperature, minimum energy, change in energy, number of accepted moves, number of rejected moves, accepted/rejected ratio, total number of moves (accepted + rejected), time taken for this cycle, average time per step.

genpdbq

Usage: **`cat file.pdb file.tor | genpdbq > output.pdbq`**

This is used to extract the order of the atoms and ROOT, BRANCH, TORS records from the `file.tor`, and replace the atom lines with the new coordinates in `file.pdb`. The PDB ATOM records in `file.tor` must be converted into (non-standard) ATMNUM records, keeping just the atom serial number from the original PDBQ file.

get-coords

Usage: **`get-coords file1.vol > file1.txt`**

This is used as part of `prepare`, `prepare-gpf+dpf`, `prepare II` and `prepare III`. It takes the `.vol` file created by `pdb-volume` and creates a line that can be used in the grid parameter file to specify the center of the maps.

gettrjdim

Usage: **`tail -12 file.tlg | gettrjdim > file.tmp`**

Used in `mktrjfld`, to obtain the number of atoms and number of frames in the trajectory log file, `file.tlg`.

gpf-gen

Usage: **`gpf-gen ligand.pdbq > ligand.gpf`**

This script is used to generate a grid parameter file. It takes as its input, a `ligand.pdbq` file, detects all atom types present, and creates the properly formatted parameter file for **AutoGrid**. This is used in `prepare`, `prepare-gpf+dpf`, `prepareII` and `prepare III`.

job, job2

Usage: **`job dpfstem > dpfstem.joblog &`**

Launches a single **AutoDock** job. It assumes that "`dpfstem.dpf`" exists, and executes **AutoDock** using the arguments:

```
$bin/autodock -p dpfstem.dpf -l dpfstem.dlg
```

You must edit this script the first time you use it, so that the environment variables $root, $bin and $sh are correctly set equal to, respectively: the path to the root of **AutoDock** tree, the architecture-dependent binary subdirectory and the Unix scripts subdirectory. The file `dpfstem.joblog` contains the output from the job script.

The variant `job2` takes *two* arguments, the first is as above, while the second is an **AutoDock** flag. For eaxmple:

Usage: **`job2 dpfstem -flag > dpfstem.joblog &`**

This script can be useful for passing one of the arguments `-k` (keep original ligand residue number in output), `-i` (ignore grid map header mismatch errors) or `-o` (old format PDBQ file).

mkbndfld

Usage: **mkbndfld ligfilestem**
Needs: **ligfilestem.bnd**
Creates: **ligfilestem.bnd.fld**

Creates an AVS field file to convey the correct bonding and connectivity to the appropriate modules. This is only important for AVS visualizations.

mkdlgfld

Usage: **mkdlgfld ligand.dlg**
Needs: **ligand.dlg**
Creates: **ligand.dlg.fld**

Extracts the "AVSFLD" records from an AutoDock log file, and puts them in ligand.dlg.fld. These "AVSFLD" descriptors must be removed before the file can be used in AVS.

mkinffld

Usage: **mkinffld job.inf**
***creates*: job.inf.fld**

This is used by AVS users, to create a field file for plotting graphs showing job information extracted by the `extjobinfo` script described above.

mktrjfld

Usage: **mktrjfld filename**
***needs*: filename.tout**
creates: filename.trj.fld

Also used by AVS users, to create a field file for displaying and animating a trajectory generated by AutoDock.

mol2topdbq

Usage: **mol2topdb2 file.mol2 > file.pdbq**

Converts from SYBYL mol2 format into AutoDock PDBQ format.

pdb-center

Usage: **pdb-center [file.pdb | file.pdbq] > file2.pdb**

Calculates the center of mass of each residue; writes these coordinates out using REMARK records.

pdb-center-all

Usage: **pdb-center-all [file.pdb | file.pdbq] > file2.pdb**

Calculates the center of mass of each residue; writes these coordinates out using REMARK records.Also calculates the center of all the residues.

pdb-volume

Usage: **`pdb-volume [ file.pdb | file.pdbq ] > file2.pdb`**

Calculates the center of mass of each residue. Writes out REMARKs showing these coordinates. Draws ASCII diagram showing volume extents of each residue.

pdbq-to-pdb

Usage: **`pdbq-to-pdb file.pdbq > file.pdb`**

Converts from **AutoDock** PDBQ to Brookhaven PDB format.

pdbq55-to-pdbq71

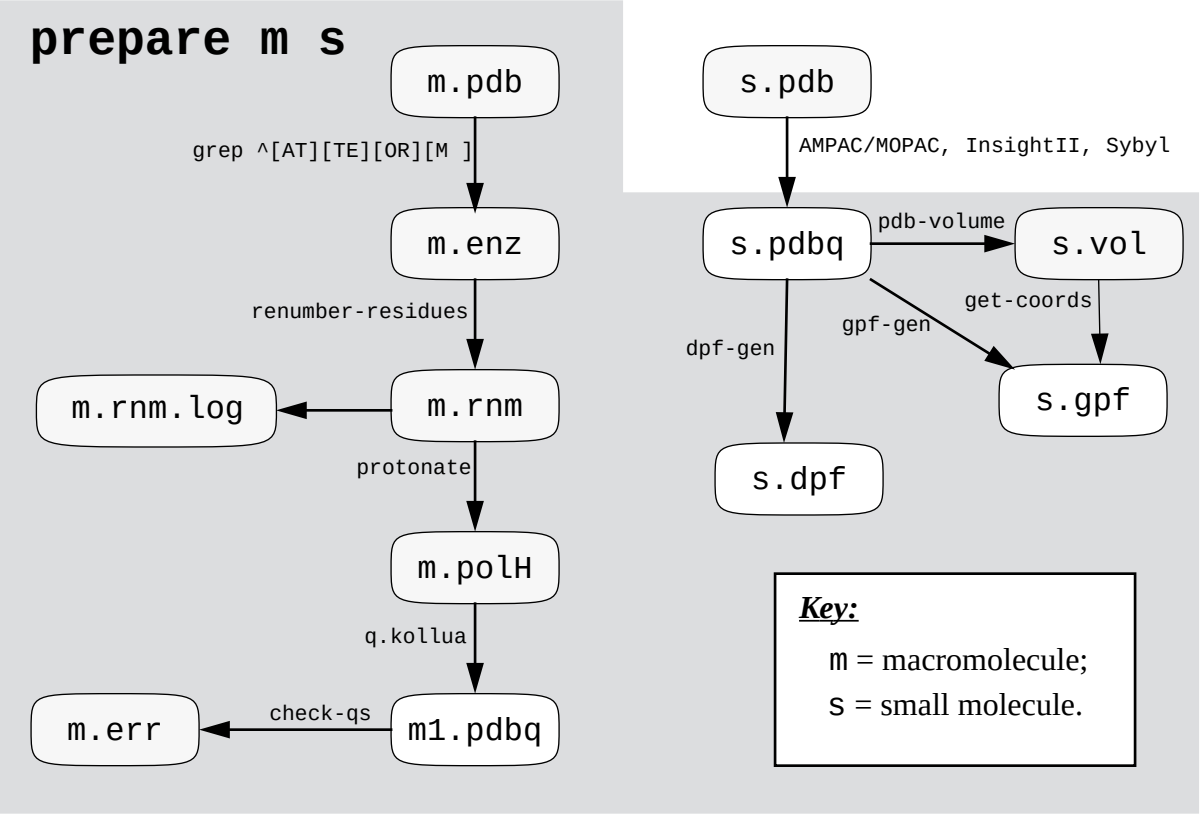
Usage: **`pdbq55-to-pdbq71 old-format.pdbq > new-format.pdbq`**

Converts from old PDBQ format (with charge in columns 55-61) to new PDBQ format (charge in columns 71-76).

prepare

Usage: **`prepare m s`**

where: `m.pdb` and `s.pdbq` contain the receptor and ligand respectively. `Prepare` performs the following eight steps. The macromolecule '`.pdb`' filename stem is represented by "`m`", and the small molecule '`.pdbq`' filename stem by "s":



1. Extracts all ATOM and TER records from `m.pdb` into `m.enz;`
2. Renumbers residues to avoid problems in `protonate`-step;
3. Adds polar hydrogens to `m.enz`, creating `m.polH`;
4. Somewhat crudely assigns partial atomic charges to `m.polH`, creating `m.pdbq`;
5. Checks charges in `m.pdbq`, all errors held in `m.err;`
6. Creates `s.gpf`, a parameter file for **AutoGrid**, based on small molecule file `s.pdbq`;
7. Creates `s.vol`, a volume dimensions file; and finally,
8. Creates `s.dpf`, a parameter file for **AutoDock**, based on small molecule file `s.pdbq`;

Its arguments are the stem of the filename of the macromolecule '`.pdb`' file and that of the small molecule PDBQ file. See the flowchart below for more details. It shows what files are created by '`prepare`', and which scripts or programs are used. Steps 1.-4. are better carried out with a reliable molecular modelling system: these steps can produce some odd results unless carefully checked.

The user must check the `m.err` error file to ensure there are no non-integral charges, either on any residue in the macromolecule, or on the macromolecule as a whole. If there are, then the user must repair the `m.pdbq` file. This problem can arise if there are atoms for which no coordinates were assigned by the crystallographer, *e.g.* due to ambiguous electron density. Assuming there were no problems, `s.gpf` and `s.dpf` should be successfully produced.

prepareII

Usage: **`prepareII macromol smlmol`**

Executes only steps 5. through 8. of `prepare` described above.

prepare-gpf+dpf

Usage: **`prepare-gpf+dpf macromol smlmol`**

Executes only steps 6. through 8.

prepare-dpf

Usage: **`prepare-dpf macromol smlmol`**

Executes only step 8.

renumber-residues

Usage: **`file.pdb > file.rnm`**

Used by `prepare` to renumber residues in the macromolecule contiguously. This step is needed prior to using `protonate`, which may fail if there are gaps in the residue numbers.

runtrj, runtrj-i

Usage: **`runtrj ligand`**
needs: **`ligand.dpf`**, **`ligand.trj`**
creates: **`ligand.tcom, ligand.tlg`** *and* **`ligand.tout`**

This creates an **AutoDock** command file, `ligand.tcom`, which is then used to convert the trajectory written in state variables (`ligand.trj`), into a trajectory written in cartesian coordinates. `Ligand.trj` is created by an earlier run of **AutoDock**, in which `trjfrq` was set to a non-zero value. **`Runtrj-i`** is like `runtrj`, but this passes the "`-i`" flag to **AutoDock** so as to ignore grid map header checking warnings.

APPENDIX 21. Parameters from AutoDock Version 1

Table 4: Lennard-Jones C_6 parameters (AutoDock version 1.0)

	C	***N***	***O***	***S***	***H***
C	1127.684	783.3452	633.7542	1476.364	226.9102
N	783.3452	546.7653	445.9175	1036.932	155.9833
O	633.7542	445.9175	368.6774	854.6872	124.0492
S	1476.364	1036.932	854.6872	1982.756	290.0756
H	226.9102	155.9833	124.0492	290.0756	46.73839

Table 5: Lennard-Jones C_{12} parameters (AutoDock version 1.0)

	C	***N***	***O***	***S***	***H***
C	1272653.	610155.1	588883.8	1569268.	88604.24
N	610155.1	266862.2	249961.4	721128.6	39093.66
O	588883.8	249961.4	230584.4	675844.1	38919.64
S	1569268.	721128.6	675844.1	1813147.	126821.3
H	88604.24	39093.66	38919.64	126821.3	1908.578

Table 6: Hydrogen bonding *12-10* parameters (AutoDock version 1.0)

Atoms i-j	C_{12}	C_{10}
O-H	75570.	23850.
N-H	75570.	23850.
S-H	2657200.	354290.