Version 4.0

Edited by Todd Ewing

Contents

Introduction

On-line Document Instructions

This manual has been distributed in several formats. The postscript (*.ps) format is suitable for printing. The portable document format (*.pdf) is suitable for on-line viewing and printing, but requires the Adobe Acrobat Reader program (version 3.0). This program is available for most computing platforms and may be downloaded free over the web from Adobe. Please see http://www.adobe.com. The hypertext format (*.html) is suitable for on-line viewing using any web browser.

By viewing this manual on-line, you may take advantage of the hypertext links throughout the document. Hypertext links are identified by color (for example, see Table 1. New Features). Activating a link with the mouse cursor will cause information in the related section to be brought into view.

Installation

DOCK is currently distributed via email, ftp, or tape cartridge (8mm exabyte or QIC-150). Create a directory for the DOCK hierarchy in the desired installation location. Uncompress the `tar` archive you received (`uncompress dock*.tar.Z`) and extract it into this directory (`tar xvf dock*.tar`). The file `0README` outlines the directory hierarchy and provides installation instructions. DOCK is written in Fortran and C. It has been tested on Silicon Graphics workstations, but it should compile with some modifications on other Unix platforms.

What's New in DOCK 4.0

Table 1. New Features

Orientation Search	The binding mode orientation search of DOCK has been rewritten, so that matching is now completely exhaustive within the geometric bounds specified by the user. The orientation search can be performed in several ways, as automated matching, as manual matching, or as a random search.
Conformation Search	All rotatable bonds are searched exhaustively using the simultaneous search. Alternatively, a rigid core is docked and the flexible parts reattached incrementally using the anchor-first search. Conformations are locally optimized.
Chemical Score	Chemical complementarity of docked molecules may be enhanced with a user-tailored scoring function. The "chemical score" may be partitioned into any number of chemical interactions whose definitions are under complete user control.
Chemical Screen	Molecules may be rapidly filtered with a similarity screen based on an active molecule or with a pharmacophore screen.
DOCK Functionality	Docking functions have been separated into independent components. Each may be combined in a flexible fashion to perform many different computational tasks.

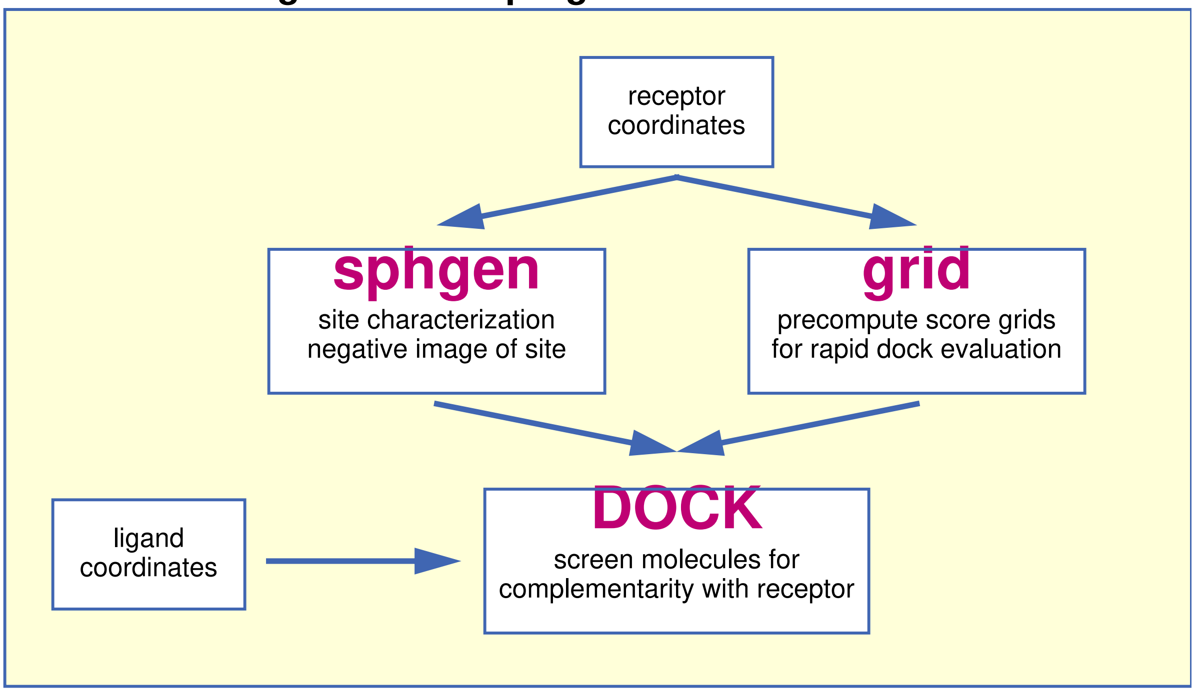
Overview of the DOCK program suite

C.M. Oshiro

Programs

The relationship between the main programs in the dock suite is depicted in Figure 1. These routines will be described below; more details can be found in various papers. We list a small subset of papers. Review articles on the method can be found in Kuntz [1] and Kuntz, Meng and Shoichet [2].

Figure 1. Main programs in DOCK suite



Program sphgen identifies the active site, and other sites of interest, and generates the sphere centers which fill the site. It has been described in the original paper: Kuntz, *et al.* [3]. Program grid generates the scoring grids; details can be found in Shoichet, Bodian and Kuntz [4] and Meng, Shoichet and Kuntz [5]. Within the DOCK suite of programs, the program DOCK matches spheres (generated by sphgen) with ligand atoms and uses scoring grids (from grid) to evaluate ligand orientations; descriptions can be found in Kuntz, *et al.* [3] and Shoichet, Bodian and Kuntz [4]. Program DOCK also minimizes energy based scores; description of minimization can be found in Meng, Gschwend, Blaney and Kuntz [6].

Several stand-alone docking-related programs exist. Program cluster generates alternative clusters of sphere centers within the active site. It uses as input, files from program sphgen. For energy scoring, ligand atom van der Waal attractive and dispersive factors and partial charges are also required.

General Concepts

This document is intended to give an overview of the ideas which form the basis of the DOCK suite of programs. It is not intended to be a reference manual or a user's guide for the programs, nor a substitute for all the papers written on DOCK. Rather, it gives a synopsis of the structure of the programs and concepts underlying the programs.

The DOCK suite of programs is designed to find favorable orientations of a ligand in a “receptor.” It can be subdivided into (i) those programs related directly to docking of ligands and (ii) accessory programs. We limit the discussion in this section to only those programs and methods related to docking a ligand in a receptor. A typical receptor might be an enzyme with a well-defined active site, though any macromolecule may be used (*e.g.* a structural protein, a nucleic acid strand, a “true” receptor). We’ll use an enzyme as an example in the rest of this discussion.

The starting point of all docking calculations is generally the crystal structure of an enzyme from an enzyme-ligand complex. The ligand structure may be taken from the crystal structure of the enzyme-ligand complex or from a database of compounds, such as the Cambridge Crystallographic Database [7] or the Concord-generated [8] set of coordinates from the Available Chemicals Directory, or ACD, (from Molecular Design, Ltd., San Leandro, CA). The primary consideration in the design of our docking programs has been to develop methods which are both rapid and reasonably accurate. These programs can be separated functionally into roughly two parts, each somewhat independent of the other:

- Routines which determine the orientation of a ligand relative to the receptor.
- Routines which evaluate (score) a ligand orientation.

There is a lot of flexibility. You can generate orientations outside of DOCK and score them with the DOCK evaluation functions. Alternatively, you can develop your own scoring routines to replace the functions supplied with DOCK.

The ligand orientation in a receptor site is broken down into a series of steps, in different programs. First, a potential site of interest on the receptor is identified. (Often, the active site is the site of interest and is known *a priori*.) Within this site, points are identified where ligand atoms may be located. A routine from the DOCK suite of programs identifies these points, called sphere centers, by generating a set of overlapping spheres which fill the site. Rather than using DOCK to generate these sphere centers, important positions within the active site may be identified by some other mechanism and used by DOCK as sphere centers. For example, the positions of atoms from the bound ligand may be used as these sphere centers. Or, a grid may be generated within the site and each grid point may be considered as a sphere center. Our sphere centers, however, attempt to capture shape characteristics of the active site (or site of interest) with a minimum number of points and without the bias of previously known ligand binding modes.

To orient a ligand within the active site, some of the sphere centers are “matched” with ligand atoms. That is, a sphere center is “paired” with an ligand atom. Many sets of these atom-sphere pairs are generated, each set containing only a small number of sphere-atom pairs. In order to limit the number of possible sets of atom-sphere pairs, a longest distance heuristic is used; (long) inter-sphere distances are roughly equal to the corresponding (long) inter-atomic ligand distances. A set of atom-sphere pairs is used to calculate an orientation of the ligand within the site of interest. The set of sphere-atom pairs which are used to generate an orientation is often referred to as a *match*. The translation vector and rotation matrix which minimizes the rmsd of (transformed) ligand atoms and matching sphere centers of the sphere-atom set are calculated and used to orient the entire ligand within the active site.

The orientation of the ligand is evaluated with a shape scoring function and/or a function approximating the ligand-enzyme binding energy. All evaluations are done on (scoring) grids in order to minimize the overall computational time. At each grid point, the enzyme contributions to the score are stored. That is, receptor contributions to the score, potentially repetitive and time consuming, are calculated only once; the appropriate terms are then simply fetched from memory.

The shape scoring function is an empirical function resembling the van der Waal attractive energy. To generate the shape score, the receptor terms from the grid point nearest to each non-hydrogen ligand atom are summed together. That is, the shape score is determined simply by the position of each ligand atom on the shape scoring grid.

The ligand-enzyme binding energy is taken to be approximately the sum of the van der Waal attractive, van der Waal dispersive, and Coulombic electrostatic energies. Approximations are made to the usual molecular mechanics attractive and dispersive terms for use on a grid. To generate the energy score, the ligand atom terms are combined with the receptor terms from the nearest grid point, or combined with receptor terms from a “virtual” grid point with interpolated receptor values. The score is the sum of over all ligand atoms for these combined terms. In this case, the energy score is determined by both ligand atom types and ligand atom positions on the energy grids.

As a final step, in the energy scoring scheme, the orientation of the ligand may be varied slightly to minimize the energy score. That is, after the initial orientation and evaluation (scoring) of the ligand, a grid-based rigid body simplex minimization is used to locate the nearest local energy minimum. The sphere centers themselves are simply approximations to possible atom locations; the orientations generated by the sphere-atom pairing, although reasonable, may not be minimal in energy.

Specific Concepts: mechanisms to limit CPU time

Sphere Centers

From an unknown source: "...what's good about DOCK is that it uses spheres; what's bad about DOCK is that it uses spheres..."

Spheres are generated to fill the target site. The sphere centers are putative ligand atom positions. Their use is an attempt to limit the enormous number of possible orientations within the active site. Like ligand atoms, these spheres touch the surface of the molecule and do not intersect the molecule. The spheres are allowed to intersect other spheres; *i.e.* they have volumes which overlap. Each sphere is represented by the coordinates of its center and its radius. Only the coordinates of the sphere centers are used to orient ligands within the active site (see above). Sphere radii are used in clustering.

The number of orientations of the ligand in free space is vast. The number of orientations possible from all sets of sphere-atom pairings is smaller but still large and cannot be generated and evaluated (scored) in a reasonable length of time. Consequently, various filters are used to eliminate from consideration, before evaluation, sets of sphere-atoms pairs, which will generate poorly scoring orientations. That is, only a small subset of the number of possible ligand orientations are actually generated and scored. The distance tolerance is one filter. Sphere "coloring" and identification of "critical" spheres are other filters.

Sphere-sphere distances are compared to atom-atom distances. Sets of sphere-atom pairs are generated in the following manner: sphere i is paired with atom I if and only if for every sphere j in the set and for every atom J in the set,

$$\left| d_{ij} - d_{IJ} \right| < \varepsilon \qquad \text{Equation 1}$$

where d_{ij} is the distance between sphere i and sphere j, d_{IJ} is the distance between atom I and atom J, and ε is a somewhat small user-defined value.

> *Note: since DOCK matches spheres with ligand atoms by comparing distances between sphere pairs and ligand atom pairs, the mirror image of the ligand atoms (used in the match) may be a better fit, in the rmsd sense, to the spheres, than the atoms of the real, non-mirror-reflected ligand. Consider, for example, the distances between the four atoms bonded to a chiral carbon center and the distances between the four atoms bonded to the mirror-image of that chiral carbon center: the distances are the same, but the two sets of four atoms cannot be superimposed upon one another, unless the chirality of one is reversed. In a similar manner, the chirality of the ligand atoms used in the match may be opposite to that of the matching spheres.

Chemical Matching

DOCK spheres are generated without regard to the chemical properties of the nearby receptor atoms. Sphere "chemical matching" or "coloring" associates a chemical property to spheres and a sphere of one "color" can only be matched with a ligand atom of complementary color. These chemical properties may be things such as "hydrogen-bond donor," "hydrogen-bond acceptor," "hydrophobe," "electro-positive," "electro-negative," "neutral," *etc.* Neither the colors themselves, nor the complementarity of the colors, are determined by the DOCK suite of programs; DOCK simply uses these labels. With the inclusion of coloring, only ligand atoms with the appropriate chemical properties are matched to the complementary colored spheres. It is probably more likely, then, that the orientation generated will produce a favorable score. Conversely, by excluding colored spheres from pairing with certain ligand atoms, the number of (probably) unfavorable orientations which are generated and evaluated can be reduced. Note that requiring complementarity in matching does not mean that all ligand atoms will lie in chemically complementary regions of the enzyme. Rather, only those ligand atoms, when paired with a colored sphere which is part of the sphere-atom match, will be guaranteed to be in the chemically complementary region of the enzyme (provided chirality of the spheres is the same as that of the matching ligand atoms).

Critical Points

The "critical point" filter requires that certain spheres be part of the set of sphere-atom pairs used to orient the ligand [9]. Designating spheres as critical points forces the ligand to have at least one atom in that area of the enzyme, where that sphere is located. This filter may be useful, for example, when it is known that a ligand must occupy a particular area of an active site. This filter removes from consideration any orientation that does not guarantee at least one ligand atom in critical areas of the enzyme (provided chirality of the spheres is the same as that of the matching ligand atom).

Scoring Filters

After a ligand is oriented within the active site, the orientation is evaluated. In an attempt to reduce the total computational time, after the ligand is oriented in the site, ligand atoms are first checked to determine whether or not they occupy space already occupied by the receptor. This is often referred to as “bump checking.” If too many of such “bumps” are found, then the ligand is likely to intersect the receptor even after minimization; consequently, the ligand orientation is discarded before evaluation.

Caveats

In the attempt to balance computational processing time and accuracy, approximations and simplifications were made to the scoring functions. The interaction energy function, for example, lacks explicit hydrogen-bonding terms, solvation/desolvation terms, or hydrophobicity terms. More accurate methods do exist for evaluating ligand docking, but at the expense of additional computational time. DOCK will do no better than the accuracy of its scoring function. That is, its ability to predict a novel ligand binding orientation and reproduce a crystal orientation is limited by the accuracy of its scoring function.

References

1. Kuntz, I.D. Structure-based strategies for drug design and discovery. *Science* **257**: 1078-1082, 1992.

2. Kuntz, I.D., Meng, E.C. and Shoichet, B.K. Structure-based molecular design. *Acc. Chem. Res.* **27**(5): 117-123, 1994.

3. Kuntz, I.D., Blaney, J.M., Oatley, S.J., Langridge, R. and Ferrin, T.E. A geometric approach to macromolecule-ligand interactions. *J. Mol. Biol.* **161**: 269-288, 1982.

4. Shoichet, B.K., Bodian, D.L. and Kuntz, I.D. Molecular docking using shape descriptors. *J. Comp. Chem.* **13**(3): 380-397, 1992.

5. Meng, E.C., Shoichet, B.K. and Kuntz, I.D. Automated docking with grid-based energy evaluation. *J. Comp. Chem.* **13**: 505-524, 1992.

6. Meng, E.C., Gschwend, D.A., Blaney, J.M. and Kuntz, I.D. Orientational sampling and rigid-body minimization in molecular docking. *Proteins*. **17**(3): 266-278, 1993.

7. Allen, F.H., Bellard, S., Brice, M.D., Cartwright, B.A., Doubleday, A., Higgs, H., Hummelink, T., Hummelink-Peters, B.G., Kennard, O., Motherwell, W.D.S., Rodgers, J.R. and Watson, D.G. The Cambridge Crystallographic Data Centre: computer-based search, retrieval, analysis and display of information. *Acta Cryst.* **B35**: 2331-2339, 1979.

8. Rusinko, A., Sheridan, R.P., Nilakatan, R., Haraki, K.S., Bauman, N. and Venkataraghavan, R. Using CONCORD to construct a large database of 3-dimensional coordinates from connection tables. *J. Chem. Info. Comput. Sci.* **29**: 251-255, 1989.

9. DesJarlais, R.L. and Dixon, J.S. A Shape- and chemistry-based docking method and its use in the design of HIV-1 protease inhibitors. *J. Comput-Aided Molec. Design*. **8**: 231-242, 1994.

Users Guide

Scope of This Guide

This section is intended as a supplement to the DOCK Reference Manual. It contains two main sections. The Getting Started section is geared towards the new user who needs some direct guidance through the docking process. The Advanced Techniques section is geared towards the experienced user and introduces new features and concepts in version 4.0.

What DOCK Can Do for You

The new version of DOCK can be used in many ways to aid in computational tasks. The following table, Table 1 lists how the new features can be combined together to accomplish different tasks. Only the basic tasks will be discussed in the Getting Started section. Please refer to the Advanced Techniques section (or follow the links contained with the table) for a discussion of each feature and refer to the Reference Manual for a listing of all associated parameters.

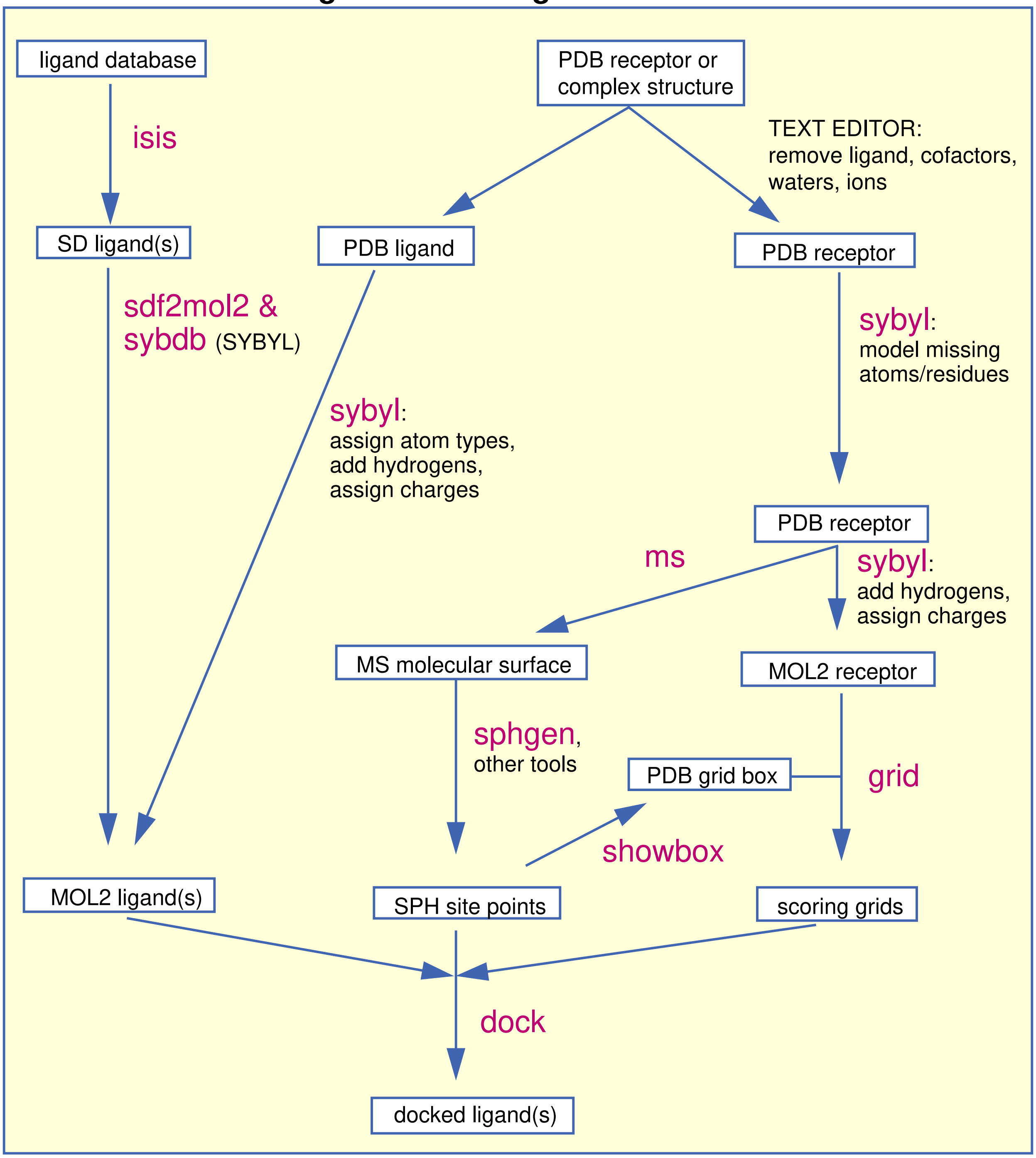
Table 1. DOCK Functionality

	flexible_ligand	orient_ligand	score_ligand	multiple_ligands	chemical_screen	parallel_jobs
Perform a conformation search of a molecule.	X					
Generate configurations of a molecule in a site.	X	X				
Dock a flexible molecule (with anchor-first search optional).	X	X	X			
Dock a database of flexible molecules.	X	X	X	X		
Generate conformations and compute internal scores.	X		X			
Generate conformations for a database of molecules.	X			X		
Key a database for a chemical screen (similarity or pharmacophore).	X			X	X	
Parse a database, processing each conformer by parallel jobs.	X			X		X
Perform an orientation search of a molecule in a site.		X				
Perform rigid docking of the input conformation of a molecule.		X	X			
Perform rigid docking of a database.		X	X	X		
Perform scoring or score optimization of input position of a molecule.			X			
Score/minimize database of molecules or molecule configurations.			X	X		
Read/write a database of molecules (file format conversion).				X		
Filter a database by chemical screen (similarity or pharmacophore).				X	X	
Parse a database, processing each molecule by parallel jobs.				X		X

Getting Started

written by Cindy Corwin and Todd Ewing

Figure 1. Docking Task Flowchart



Overview

This section of the users guide is intended to help a new user of dock get started with their first docking project. It describes the steps a user would typically take to apply the programs to a macromolecular target and potential ligands of interest. While the reference manual describes in detail the various input and output files, this guide is meant to convey the process in informal terms. Some of the difficulties we have encountered as well as approaches we have found useful are discussed.

DOCK is a program for locating feasible binding orientations, given the structures of a "ligand" molecule and a "receptor" molecule. What is considered feasible depends on how the orientations are evaluated. Current options are a contact (shape-fitting) score,a force field interaction energy and a new user-defined chemical scoring scheme. DOCK generates many orientations of one ligand. The best-scoring orientation of each molecule is saved, and the best-scoring molecules are written out. Some of the molecules in the list of best-scoring compounds, perhaps with modifications, may be interesting as potential new ligands for the receptor.

The basic requirement for docking is a structure of the macromolecule of interest. The docking procedure can be divided into four general stages: ligand preparation, site characterization, scoring grid calculation, and docking itself. Please refer to Figure 1 while reading this guide.

Site characterization is the process of deciding what areas of the receptor site to study. This is done by constructing site points to map out the negative image of the active site. These site points are used by DOCK to construct orientations of the ligand.

Scoring grid calculations are necessary so that DOCK can evaluate ligand orientations rapidly.

The final stage of the process is running DOCK and viewing the results. DOCK uses the site points to generate ligand orientations, then uses the precomputed grids to evaluate the orientations. The best-scoring molecules or orientations may be viewed using a molecular graphics program.

There are multiple tasks involved in the docking process, and each task can require many decisions over input parameters. We hope this beginner’s guide will make it easier to navigate through the tasks and to select sensible parameters.

Organizing Your Workspace

It is a good idea to make a make a new UNIX directory for each docking project using the UNIX `mkdir` command. Within this project directory make a sub-directory for each of the main tasks. Make a `struc/` sub-directory to hold the ligand and receptor coordinates and molecular surfaces. Make a `site/` sub-directory to hold the site point files. Make a `grid/` sub-directory to hold the scoring grid files. Make a `dock/` sub-directory to hold the dock files.

A Caution Concerning Disk Space

The output from some of the programs associated with DOCK, particularly MS, SPHGEN, and DOCK itself, may require substantial amounts of disk storage. Check before starting your job to make sure there is space available. It is a good idea to be cautious at first: use restrictive parameters choices with only a handful of ligands, to make sure that you are getting the results you desire. While DOCK jobs are running, check to be sure they are not creating overly large files.

Ligand Preparation

Single Ligand

Before you can dock a ligand, you will need atom types and charges for every atom in the ligand. It is recommended that you use SYBYL MOL2 format for the ligand file since it contains fields for atom types and charges. For a single ligand (or several ligands), you can use sybyl to prepare a MOL2 file for the ligand.

Ligand Database

Check if a database of ligands has already been prepared at your site. Again, we recommend that this database be in SYBYL MOL2 format. If the MOL2 database does not exist, then you will need to construct it. Typically, the Available Chemicals Directory (ACD) is used as a ligand database. This database is distributed by Molecular Design Ltd. (San Leandro, CA) for use with the isis database package. The ACD can be output from ISIS in an SD-format file. Use sdf2mol2 & sybdb to generate a MOL2 file from the SD file. This conversion requires sybyl (from Tripos) to assign atom types and charges.

Site Characterization

Working With Macromolecular Models and Generating the Molecular Surface

Removing Ligands and Crystallographic Waters

The macromolecular structure you are working with may include a ligand, and crystal structures usually contain water molecules and sometimes ions which were found on the surface of the protein. These molecules are usually not included as input to ms. To prepare for molecular surface generation, make a copy of the protein coordinate file. If there is a ligand present, remove it by deleting all of its records (they often start with HETATM in Brookhaven Protein Data Bank format files) from your copy of the file. (Note - sometimes, as in the case of a cofactor or catalytic metal ion, it may make chemical sense to keep a ligand in the PDB file.) Whether or not crystallographic waters and ions should be preserved when generating surfaces for use by sphgen is a matter of some debate. In structures of complexes, water molecules and ions are often found in the protein binding pocket along with the ligand(s). However, ligands can displace waters and ions, and the volume of a receptor site will be explored more completely if the waters and ions are removed, so if you don't have particular reasons for preserving any of the water molecules or ions in the crystal, it is probably best to remove all of them. Waters are usually located near the end of the PDB file and are often HETATM records with HOH or WAT residue types. Ions are often near the waters in the PDB file.

Please note that the PDB file used for generating the molecular surface should not include hydrogen atoms. NMR structures will include hydrogens; delete the hydrogens from a copy of each structure and use that copy in ms.

Creating the Molecular Surface

The dot surface which will be used to produce spheres is generated by the program ms, available from Quantum Chemistry Program Exchange (QCPE). When setting up for docking, it is acceptable just to generate surface for the site of interest and adjacent regions (see documentation for **get_near_res** and automs); this will also reduce the computer time used by SPHGEN. *Note: SPHGEN requires that the surface points must have associated normals.*

If you use the QCPE version of MS, you must run reformatms to convert the surface to the format used by SPHGEN (both formats are described in the reference manual section on REFORMATMS). REFORMATMS is interactive and requires the surface and the PDB file used to generate the surface.

Users of the UCSF MidasPlus package may use the output from the DMS program directly as input for SPHGEN.

Representing the Site With Spheres

We typically use SPHGEN to construct shape-based site points, but you may use any other program to construct site points. With the use of other programs you may include considerations of chemical complementarity in your site points. A common alternative to SPHGEN is the Goodford's grid program (Peter Goodford).

SPHGEN

sphgen uses the points of the molecular surface and their associated normals to determine spheres to fill the site. It then reduces the number of spheres to one per atom and groups them into clusters. You can inspect these clusters and regroup the spheres if necessary.

Creating `INSPH`

The parameters which tell SPHGEN exactly how to create the surface are placed in a file called `INSPH`, which must be present when SPHGEN is run. The contents of this file are described in the reference manual. To create it, make a file with each variable on a separate line. Most of the parameter values given in the reference manual should work fine. You will need to replace *msfil* with the name of your surface file and *outfil* with the desired name of your output file.

Running SPHGEN

SPHGEN must use the directory containing `INSPH` as its working directory; this means that it should be started while you are in that directory. The SPHGEN output file contains clusters of spheres which have been selected and grouped by SPHGEN; the clusters are listed in order of decreasing size. The last cluster, numbered 0, contains all the spheres produced. It may be used with the program cluster to make new sphere clusters if the original clustered output doesn't describe the site well.

Looking at the Output

Once you've generated spheres, you should look at the sphere clusters using a molecule display program. showsphere may be used to generate a PDB-like file of sphere centers for display. It can also generate a surface for the sphere cluster (in the MS format used by SPHGEN). SHOWSPHERE is interactive. You will be prompted for the name of the cluster file (that is, the SPHGEN output), the number of the cluster, and names for the desired output file. In the PDB-like file of sphere center coordinates, each sphere is a separate residue and the spheres are separated by TER cards.

Getting a Good Sphere Cluster

Displaying the protein and sphere centers together should tell you how each sphere cluster is related to the site you are trying to represent. Examine sphere clusters until you find one that occupies the region into which you want to dock ligands. Clusters of 50 or fewer spheres are best; larger numbers of spheres will cause DOCK to use more computer time. It is generally unwise to try docking with more than 100 spheres, although you may be able to use more if your database is small or you are using chemical matching. Initial sphere clusters are sometimes spider-like structures which include the area of interest but also branch into other regions. If your cluster has too many spheres, branches out, or is unsatisfactory for some other reason, you can correct the problem.

The easiest way to fix a sphere cluster is to use graphics to identify spheres that you don't really need, then remove them. When you've found the unnecessary ones, go back to the *original* sphere cluster file (i.e. the one from SPHGEN) and delete the corresponding lines - the residue number in the PDB-like file of centers is the first number in the line in the sphere file. Remember to change the number of spheres listed on the line with the cluster number to reflect the deletions.

If your cluster is large — more than about 100 spheres — and deleting spheres by hand looks too tedious, you can use cluster to break it into smaller clusters. CLUSTER is described in the reference manual; read the documentation completely before you try it. Start with the parameters given and experiment with the values; small changes can make a big difference in the result. Be aware that if the best cluster found is the same as the original input cluster, the program will appear not to have done anything.

The two methods just described may be combined if the best CLUSTER output is not quite right. More spheres can be deleted from the new cluster, or, if the new cluster is too small, additional spheres may be added graphically. A cluster containing all the desired spheres may then be created by editing the SPHGEN output.

If nothing else works, it is possible to run CLUSTER on all possible spheres rather than a preselected group. Use the analytical clustering algorithm in `cluster` on cluster 0, and experiment until you get what you want. Flagging spheres in important regions of the site may help.

Creating the Scoring Grids

GRID

grid saves information about the steric and electrostatic environment at each point on a grid, so that ligand orientations can be scored rapidly during a DOCK run.

Positioning the Grid

You determine the location and dimensions of the region to be gridded by using the program showbox to create a box which contains the desired region. For GRID, the box should enclose the volume that the ligand orientations are likely to occupy. An easy way to accomplish this is to generate a box which encloses the spheres to be used for docking along with an extra margin. The box generated should be viewed along with the receptor and possibly regenerated until it looks good.

Preparing the Receptor File

First read the receptor PDB file into a text editor. Remove all waters and complexed ligands. Special attention should be given to the names of atoms at the termini and the residue names for histidine and cysteine. You will need to rename each histidine residue depending on the protonation state you want to assign it: HIP for positively charged (hydrogens on both nitrogens), HID for neutral with the delta nitrogen protonated, and HIE for neutral with the epsilon nitrogen protonated. CYS refers to a cysteine with a free sulfhydryl group; CYX refers to a cysteine involved in a disulfide bond (a half-cystine). Note that some structures in the PDB use CYS in disulfides; these should be edited to CYX.

Second the user must construct a SYBYL MOL2 format of the receptor which includes sybyl atom type and partial charge assignments. We routinely use sybyl for this task, but other modeling packages can be used provided you have a way to convert the resulting receptor file into MOL2 format. The following instructions will apply to the use of SYBYL for this task.

In SYBYL, activate the BIOPOLYMER menu from the OPTIONS menu. From the BIOPOLYMER menu, select BROOKHAVEN READ to read in the receptor PDB file. A dialogue box will ask if you want to center the molecule. If you need to retain the reference frame of the receptor (e.g. for consistency with other collaborators) then don't center the coordinates. Instead, you will need to manually find the receptor since it will probably not appear on the screen. Hit the lower button on the left side of you SYBYL window which looks like a molecule surrounded by arrows. In the small window that appears, hit the button called "reset extents." Now you should see the receptor from a distance. Use the far right mouse button to rotate the receptor to its highest point on the screen. Use the middle mouse button to translate it to the center of the screen. Then use the combination of mouse buttons to zoom in on the receptor. You may need to rotate it up and translate back to the middle a few times to keep it from escaping the window. You will need to repeat this exercise every time you read in any molecule that is in the unperturbed frame of reference of the receptor (i.e. bound ligands).

Check if all of the atoms were identified properly by SYBYL. You can label problem atoms by selecting LABEL ATOMS. In the atom selection window, press the SET button. If you see a set type called "Unknown Atoms" then select it. Any atoms that were not recognized by SYBYL will now be labelled. This most often occurs with oxygen atoms at the C-terminus, with unusual amino acids, or if cofactors or ligands were not removed. If the terminal carboxyl oxygens are the problem, then rename them with the text editor and reread the receptor. For other problem atoms you will need to consult the SYBYL manual.

Next, you should model in any incomplete residues. Check the original PDB file for a list of residues whose density was to weak to model completely. If no list exists, then check the total charge on each residue as reported by GRID when you run it later; if some residues have non-integer charges, then you may need to come back to SYBYL and model them in. To do this, identify the residue to fix. From the BIOPOLYMER menu, select MODIFY, then MUTATE RESIDUE. Click on the residue you want to modify, then select which type of residue you want to mutate it to (use the same type of residue).

Add hydrogens by using the BIOPOLYMER menu option. It is important to add ALL atoms, not just POLAR atoms, since GRID needs them to identify VDW atom types. Load charges from the BIOPOLYMER menu. You may use ALL atom or UNITED atom KOLLMAN charges. Write out the receptor to a MOL2 file by using the write option in the FILE menu.

Running GRID

Input to GRID is interactive. Just type `grid -i grid.in` to launch it. All input parameters will be saved in the file called grid.in. After all parameters have been input, hit CTRL-C to kill the job. Relaunch it in background mode by typing `grid -i grid.in -o grid.out&`.

All recommended parameter values will be suggested by GRID when run interactively, but here are a few suggestions. `grid_spacing` values between 0.2 and 0.5 are recommended; fine grids are preferred if there is sufficient memory in the computer. Any combination of grid point spacing and box size can be used, but it is recommended that about a million total grid points be used. Of course this value depends on memory resources.

A dielectric function of 4.0*r* or 4.5*r* and a cutoff of 10.0 Angstroms or more are appropriate in most cases. (This dielectric corresponds to specifying `distance_dielectric yes`, `dielectric_factor 4` or `4.5`, and `energy_cutoff_distance 10`.) If a constant dielectric is selected, an “infinite” cutoff (one large enough to include the whole receptor) should be used.

It is important to check the residue charges that are output by GRID. If any non-integer charges are reported, then some residues may have improper charges assigned to them, or they are not completely modeled in the input file. If no charged residues are reported, then check to make sure that charges were properly loaded in the input file.

Four output files, named `grid.bmp`, `grid.cnt`, `grid.chm`, and `grid.nrg`, will be produced which hold the bump grid, contact grid, chemical grid and force-field grid, respectively.

Running DOCK

Starting a DOCK Run

You are now ready to run DOCK. Since DOCK can use a substantial amount of CPU time, it is a good idea to check whether there are other jobs running on the same machine. Consider any other users sharing your computers when deciding whether to start more than one run at a time. Be aware of any policies your site has regarding submitting background jobs.

The easiest way to select dock parameters is to run DOCK interactively. Do this by typing `dock -i dock.in`. You will be prompted for a value for each parameter. Any value you enter will be stored in the file `dock.in`. This file does not need to exist beforehand. If it does exist, then DOCK will extract all the relevant parameters it can find from the file. For each parameter, DOCK will supply a default value. If you want to use the default value, just hit return. The following tables list recommended values for running DOCK in two different ways: first to dock a single ligand, and second to dock a database of ligands. If you are viewing this manual on-line, then click on any of the keywords to view the reference entry for it.

Table 2. Recommended DOCK parameters for a new user.

2A: General

Keyword	Suggestions
`flexible_ligand`	`no`; try later
`orient_ligand`	`yes`; searches ligand orientations
`score_ligand`	`yes`; scores each ligand orientation

2A: General

Keyword	Suggestions
minimize_ligand	no; try later
multiple_ligands	no
random_seed	0 is fine

2B: Orientation Search

Keyword	Suggestions
match_receptor_sites	yes; matches ligand to site points
random_search	no
ligand_centers	no
automated_matching	yes; otherwise need to enter geometric match parameters
maximum_orientations	500; number of orientations to try
write_orientations	yes; to write out multiple orientations for single molecule
rank_orientations	yes; to save the top scoring orientations
rank_orientation_total	10; to save the top 10 orientations

2C: Scoring

Keyword	Suggestions
intermolecular_score	yes
gridded_score	yes
grid_version	4
bump_filter	yes
bump_maximum	3
contact_score	no
chemical_score	no
energy_score	yes

2C: Scoring (Continued)

Keyword	Suggestions
atom_model	u; for united atom model
vdw_scale	1
electrostatic_scale	1

2D: Input

Keyword	Suggestions
ligand_atom_file	Enter the ligand MOL2 file name here (including the directory path if this file is not in the current directory).
receptor_site_file	Enter the SPHGEN site point file name here.
score_grid_prefix	Enter the GRID file name prefix here.
vdw_definition_file	~dock/parameter/vdw.defn

2E: Output

Keyword	Suggestions
ligand_energy_file	dock_nrg.mol2

Table 3. Recommended beginner's DOCK parameters for a database search run

3A: Parameters to Modify from Database Run

Keyword	Previous	Suggestions
multiple_ligands	no	yes; to consider multiple molecules
maximum_orientations	500	50; just so that the run doesn't take too long
write_orientations	yes	no; to write only the best orientation for each molecule

3B: Multiple Ligands

Keyword	Suggestions
`parallel_jobs`	`no`
`ligands_maximum`	first try 10 to make sure everything is working, then set it to a number larger than the number of molecules in input file
`initial_skip`	`0`; **n**>0 will skip the first **n** molecules in input file
`interval_skip`	`0`; **n**>0 will skip **n** molecules for each molecule processed
`heavy_atoms_minimum`	`4`; to discard small molecules
`heavy_atoms_maximum`	`50`; to discard large molecules
`rank_ligands`	`yes`; to save a top score list
`rank_ligand_total`	`50`; to save the top 50 molecules
`restart_interval`	`100`; to save restart info every 100th molecule processed

3C: Additional Input and Output

Keyword	Suggestions
`quit_file`	`dock.quit`
`dump_file`	`dock.dump`
`info_file`	`dock.info`
`restart_file`	`dock.rst`

If you happen to enter the wrong value for any parameter and wish to change it, then you may edit the `dock.in` file directly and modify the parameter value. Once all parameters have been entered, DOCK should begin the calculation and the `dock.in` file is complete. You may kill the process with a CTRL-C and relaunch the process in background by typing "`dock -i dock.in -o dock.out&`". If you would like to run DOCK multiple times from the same directory, then you may use a different name for `dock.in` and `dock.out`. Just be sure to change the names of the output files inside the new `dock.in` file so that two processes don't end up overwriting each other's output files.

Check a few minutes after you start the run to be sure that it is still going; if it has stopped, look for mistakes in the input. Beginners should check disk usage occasionally while the job is running, just in case the program is creating incredibly large files which might overflow the available space.

During a database search run (which can take anywhere from hours to days to weeks to finish), you can follow DOCK's progress through the database by inspecting the `*.info` file.

Restarting a Search Run

In database search mode of DOCK (when `multiple_ligands`, `orient_ligand`, and `rank_ligands` are selected), DOCK periodically saves information necessary to restart the search from its current location in the database in a `*.rst` file. If there is a power failure or the system crashes, you can set up a new run to start where the last one was stopped. First, make a copy of `dock.out` so that status of the previous run are saved. Then relaunch the job using the `-r` flag at the command line. (Do not change the remaining files, since DOCK needs them to restart successfully.) When the restarted run finishes, the sorted list of ligands in the output file will include the top scorers from the entire database.

Looking at the Results

DOCK puts the resulting molecule orientations in a file for each type of scoring function used. The scores are given in the comment records at the beginning of each molecule record. If you have selected MOL2 format for your output files and your graphical viewer does not read this format, then convert the file to PDB by typing `dock -i dock_nrg.mol2 -o dock_nrg.pdb`.

Other Post-Docking Tasks

Depending on your particular project, you might be interesting in any one or several of the following post-docking techniques:

- Rescoring of hits with alternative scoring function;
- Redocking of hits with increased orientation sampling and/or conformational sampling;
- Similarity searching based on hits; or
- Further molecular modeling/molecular dynamics/FEP of hits;

Advanced Techniques

written by Todd Ewing

Introduction

This section of the manual provides a discussion of many of the advanced features available in DOCK. It is intended for users who already have some familiarity with using DOCK.

Orientation Search

DOCK version 4.0 has a new orientation search algorithm, or matching algorithm, which is more robust than before (see Ewing and Kuntz [6]). An orientation search is requested with the `orient_ligand` parameter. The published search technique has been further extended so that the amount of orientation sampling can be controlled in two ways:

- Automated Matching —Specify the number of orientations, and dock will generate matches until enough orientations passing the bump filter have been formed. Matches are formed best first, with respect to the difference in the ligand and site point internal distances.
- Manual Matching —Specify the distance and node parameters, and DOCK will generate all the matches which satisfy them. The number of orientations scored is equal to the total matches minus the orientations discarded by the bump filter.

There are a number of sophisticated options available to tailor the orientation search. These options include:

- Random Search
- Degeneracy Checking
- Ligand Mirroring
- Chemical Matching
- Critical Points

Multiple orientations may be written out for each molecule using the `write_orientations` parameter, otherwise only the best orientation is recorded. A ranked list of the orientations may be written using the `rank_orientations` parameter. Otherwise, all orientations passing a score cutoff are written out. The score cutoff is specified with the `contact_maximum` parameter and so on for each type of scoring. If `write_orientations` is requested without scoring, then all orientations are written.

Automated Matching

With `automated_matching`, DOCK performs the same amount of orientation searching on each molecule. If the `match_receptor_sites` parameter is set, then manual matching is used as a black box engine for the orientation search (otherwise a random search is performed). The only sampling parameter needed is the `maximum_orientations` parameter, which is the number of desired orientations which survive the bump filter. Matches are formed in order of the smallest distance error first, so that the highest quality orientations are guaranteed to come sooner rather than later. This method of control is incredibly easy. It is most appropriate when docking a single molecule. It should not be used for database docking, since manual matching performs better because it biases the amount of sampling depending on the size and shape of the ligand. In addition, if the user wishes to use advanced matching features, like chemical matching and critical points, then manual matching must be used.

Manual Matching

If the match_receptor_sites parameter is set but not the automated_matching parameter, then manual matching is performed. It is controlled by the match parameters listed in Table 4. The matching parameters provide an intuitive way to control sampling. When multiple molecules are docked, matching will bias sampling towards molecules with more internal distance similarity with the receptor site points. The additional chemical and critical matching constraints provide a way to prune matching and further bias sampling towards more interesting molecules.

Table 4. Description of Matching Parameters

distance_tolerance	The distance tolerance can be viewed as the uncertainty in the distance comparisons or site point positions. The more generous the uncertainty in the distance comparisons, the more sampling will be performed. This parameter should be the first parameter to adjust if you need to change the amount of sampling.
distance_minimum	The distance minimum allows matching to focus on the longer distances which convey more information about molecule or site shape. This value can be conveniently set large enough to discard atoms directly bonded to each other. When docking large molecules, this value can be set higher.
nodes_minimum	The minimum number of nodes must be at least three to specify a unique rigid transformation. A value of four or more will allow every match to include information about chirality. Match chirality can be used to explore the mirror image of a molecule for docking. The higher this parameter, the better the ligand atoms in the match represent the entire molecule.
nodes_maximum	This value may be set arbitrarily high to prevent it from influencing matching. It may be set equal to the nodes minimum when performing pharmacophore-style matching if only a few specific site interactions are of interest.

Random Search

The random_search option is intended for advanced users. If match_receptor_sites is also set then random matching is performed, in which ligand centers and receptor sites are randomly matched regardless of internal distances. Otherwise, a random transformation search is performed, in which ligands are randomly rotated and translated within the rectangular box enclosing all the site points. Both methods could be employed when the user is concerned about the quality of the site point positions, or would simply like to try a richer set of generated orientations.

Site Point Construction

The random_search option is useful for exploring issues relating to site point construction. As discussed in Ewing and Kuntz [6], both random matching and random transformation were useful control algorithms to test the effectiveness of distance-based matching. The relative performance of random matching with respect to random transformation indicates how well the site points map out the relevant volume of the active site. The relative performance of distance-based matching with respect to random matching indicates how well individual positions of each site point correspond to good ligand atom positions. By using both of these search methods, an advanced user may quantify the quality of site points constructed by alternative methods to sphgen.

The random transformation search may in fact be used to construct site points to supplement those from sphgen. Using this search, the user may probe a site with different molecular probes much like the atomic probes used in Goodford's grid program. The best-scoring positions may then be used to position site points.

Degeneracy Checking

Degeneracy checking is a method implemented during matching to increase the diversity of the resulting orientations. It is selected with the `check_degeneracy` parameter. It is not an available feature if `automated_matching` has been selected. The method of Gschwend and Kuntz [11] implemented in dock version 3.5 has been updated to be easier to use and more robust. Degenerate matches are now defined as matches which are a subset of a larger match. In the nomenclature of graph theory, the surviving matches are maximally connected and are true cliques.

For degeneracy checking to work, `nodes_maximum` must be greater than `nodes_minimum` so that subsets can occur. In general, just set `nodes_maximum` arbitrarily high (15 or so). At most a two-fold reduction in matches is achieved using this feature.

Ligand Mirroring

When a match contains four or more nodes, the chirality of the ligand and receptor points involved in the match is checked. Half of the time, the ligand and receptor points have opposite chirality. See Ewing and Kuntz [6] for more discussion. Normally these improper matches are discarded, but they can be rescued with the `reflect_ligand` option, which allows the chirality of the ligand to be reversed by using its mirror image. This is useful for molecules which are either achiral or are available as a racemate.

Chemical Matching

The `chemical_match` feature is used to incorporate information about the chemical complementarity of a ligand orientation into the matching process. As in Kuhl et. al [15], chemical labels are assigned to site points and ligand atoms. The site point labels are based on the local receptor environment. The ligand atom labels are based on user-adjustable chemical functionality rules. These labeling rules are identified with the `chemical_definition_file` parameter and reside in an editable file (see chem.defn on page 106). A node in a match will produce an unfavorable interaction if the atom and site point components have labels which violate a chemical match rule. The chemical matching rules are identified with the `chemical_match_file` parameter and reside in an editable file (see chem_match.tbl on page 107). If a match will produce unfavorable interactions, then the match is discarded. The speed-up from this technique depends how extensively site points have been labeled and the stringency of the match rules, but an improvement of two-fold or more can be expected.

The process of labeling site points must currently be done by hand. The user should load the site points and the receptor coordinates into a graphic program and study the local environment of each point. Developing an automated method to perform this task is still an active area of research. Labeled site points may be input as either a SPH format or SYBYL MOL2 format coordinate file. Check sphgen on page 84 for file format specifications. An example is shown in Table 5. To store labeled site points in a MOL2 file, select an atom type for each label of interest. Then edit the chem.defn file to include the selected atom types. Site point definitions can be distinguished from ligand atom definitions by explicitly requiring that no bonded atoms can be attached (ie. followed by `[*]`). The example chem.defn on page 106 includes a site point definition as the last definition for each label. Using the convention in that example file, site points should be labeled as follows: hydrophobic, "`C.3`"; donor, "`N.4`"; acceptor, "`O.2`"; polar, "`F`".

Table 5. Example of chemical labels in SPH format

```
DOCK 3.5 receptor_spheres
color hydrophobic        1
color acceptor           2
color donor              3
cluster     1   number of spheres in cluster    49
    7   2.34500  36.49000  16.93500   1.500    0 0  1
    8  -0.05200  42.29900  14.18800   1.500    0 0  1
    9  -0.67000  41.20600  11.59800   1.500    0 0  1
   17  -6.00000  34.00000  17.00000   1.500    0 0  3
   18  -5.00000  29.00000  22.00000   1.500    0 1  3
                           ...
```

Caveats on Chemical Matching

It can take a significant amount of effort to chemically label a large site and to verfiy that the docking results are what were expected. If you use this chemical matching, plan to spend some time in preparation and validation BEFORE running an entire database of molecules.

In concert with degeneracy checking, chemical matching is able to discard matches that not only contain bad interactions but that can be expanded to include other bad interactions. Although this helps reduce the bad interactions in an orientation, it can only do so within the constraints of the `distance_tolerance`, which can be rather tight. In addition, the number of interactions monitored in a match is usually small (3-5) compared to the total number of ligand atoms, so the preponderance of atoms may be in less than favorable environments. Therefore, chemical matching does not guarantee that all resulting orientations are chemically complementary, but instead that the resulting orientations are *enriched* in complementarity.

It must be pointed out that the ultimate arbiter of which orientations of a ligand are saved is actually the scoring function. If the scoring function is unable to discriminate what the user feels are bad chemical interactions, then any improvement with chemical matching will probably be obscured. In addition, if score optimization is used, then the orientation will be perturbed from the original chemically-matched position to a new score-preferred positions.

Critical Points

The `critical_points` feature is used to focus the orientation search into a subsite of the receptor active site [4, 23]. For example, identifying molecules that interact with the catalytic residues might be of chief interest. Any number of points may be identified as critical, and any number of groupings of these points may be identified. Consequently, several receptor subsites may be targeted simultaneously. If a particular cluster of critical points is big enough to interact with more than one ligand atom, then use the `multiple_points` parameter. An alternative to using critical points is to discard all site points that are some distance away from the subsite of interest, while retaining enough site points to define unique ligand orientations.

This feature can be highly effective at reducing matching by five-fold or more. It is particularly useful to also assign chemical labels to the critical points to further focus sampling.

Conformation Search

The conformation of a flexible molecule may be searched or relaxed using the `flexible_ligand` option. Only the torsion angles are modified, not the bond lengths or angles. Therefore, the input geometry of the molecule needs to be of good quality. A structure generated by CONCORD is sufficient.

The user may request a conformation search using the `torsion_drive` parameter and/or torsion minimization using the `torsion_minimize` parameter. The torsion angle positions reside in an editable file (see flex_drive.tbl on page 111) which is identified with the `flex_drive_file` parameter. Internal clashes are detected during the torsion drive search based on the `clash_overlap` parameter, which is independent of scoring function.

If `multiple_ligands` are being processed, then the `flexible_bond_maximum` cutoff is used to discard overly flexible molecules.

When scoring is requested, the user has the option of computing intramolecular terms using the `intramolecular_score` parameter. For the sake of speed, only the interactions between segments is considered. If a segment has not moved, then the contribution of its interaction with the receptor to the intermolecular score is not recalculated. If any two segments have not moved, then the contribution of their interaction to the intramolecular score is not recalculated.

The treatment of flexible molecules will be elaborated further. The first stage of processing is the identification of rigid segments. The second stage of processing is the conformation search, at which point the user has the choice of two methods. An anchor-first search may be selected, in which a molecule conformation is constructed and minimized one segment at a time, starting from an anchor segment. Alternatively, a simultaneous search will be performed, in which the entire molecule conformation is constructed and minimized in one step.

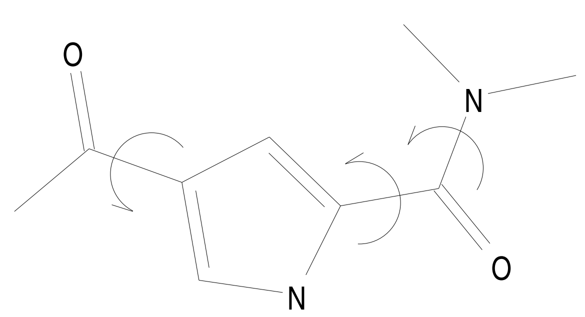
Identification of Rigid Segments

A flexible molecule is treated as a collection of rigid segments. Each segment contains the largest set of adjacent atoms separated by non-rotatable bonds. Segments are separated by rotatable bonds.

The first step in segmentation is ring identification. All bonds within molecular rings are treated as rigid. This classification scheme is a first-order approximation of molecular flexibilty, since some amount of flexibility can exist in non-aromatic rings. To treat such phenomenon as sugar puckering and chair-boat hexane conformations, the user will need to supply each ring conformation as a separate input molecule.

Additional bonds may be specified as rigid by the user. Please refer to the subsequent section, Manual specification of non-rotatable bonds.

Figure 2. Flexible Bond Identification



The second step is flexible bond identification, and is illustrated in Figure 2 for a sample molecule. Each flexible bond is associated with a label defined in an editable file (see flex.defn on page 110). The parameter file is identified with the `flex_definition_file` parameter. Each label in the file contains a definition based on the atom types (and chemical environment) of the bonded atoms. Each label is also flagged as minimizable. Typically, bonds with some degree of double bond character are excluded from minimization so that planarity is preserved. Each label is also associated with a set of preferred torsion positions. The location of each flexible bond is used to partition the molecule into rigid segments. A segment is the largest local set of atoms that contains only non-flexible bonds.

Manual specification of non-rotatable bonds

The user can specify additional bonds to be non-rotatable, to supplement the ring bonds automatically identified by DOCK. Such a technique would be used to preserve the conformation of part of the molecule and isolate it from the conformation search. Non-rotatable bonds are identified in the SYBYL MOL2 format file containing the molecule. The bonds are designated as members of a STATIC BOND SET named RIGID. Please see SYBYL MOL2 format on page 99 for an example of such an identification.

Creation of the RIGID set can be done within SYBYL. With the molecule of interest loaded into SYBYL, select the Build/Edit->Define->Static Set->Bond command. Then select each bond by picking the adjacent atoms. When the "Set Name" dialog comes up, supply the name "RIGID" in capital letters. When the "Comment String" dialog comes up, enter any text you wish. Write out the molecule to file.

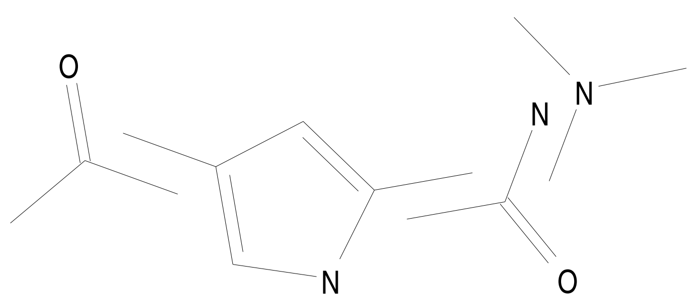
Alternatively, the RIGID set can be entered into the MOL2 file by hand. To do this, go to the end of the MOL2 file. If no sets currently exist, then add a SET identifier on a new line. It should contain the text "`@<TRIPOS>SET`". On a new line add the text "`RIGID STATIC BONDS <user> **** Comment`". On the next line enter the number of bonds that will be included in the set, followed by the numerical identifier of each bond in the set.

Anchor-First Search

The anchor-first search is an efficient divide-and-conquer algorithm based on the method of Leach and Kuntz [19] and the greedy algorithm. It is specified using the `anchor_search` parameter.

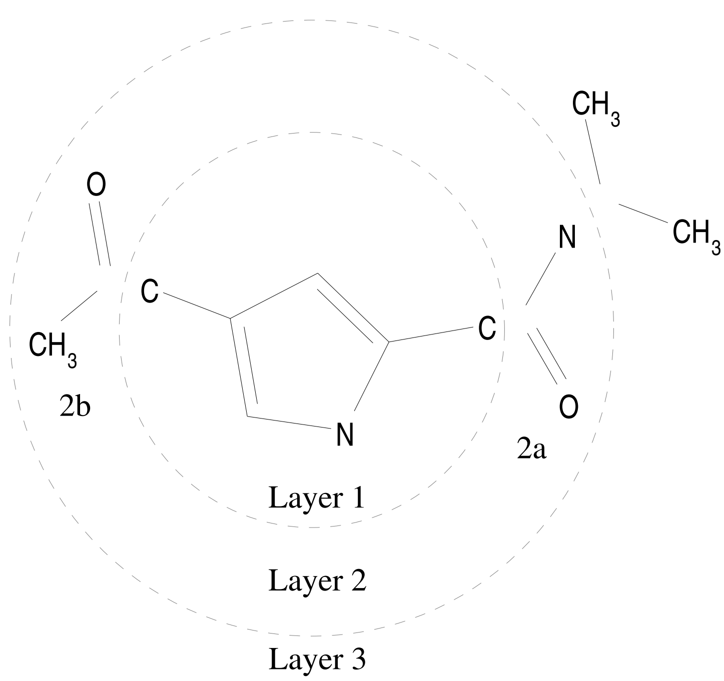
An anchor segment is selected from the rigid segments in an automatic fashion (see Manual specification of anchor segment to override this behavior). As illustrated in Figure 3 for a sample molecule, the molecule is divided into segments that overlap at each rotatable bond. The segment with the largest number of heavy atoms is selected as the anchor. If the `multiple_anchors` parameter is set, then all segments which pass the `anchor_size` cutoff are tried separately as anchors.

Figure 3. Overlapping Segments



When an anchor has been selected, then the molecule is redivided into non-overlapping segments, which are then arranged concentrically about the anchor segment. This process is illustrated in Figure 4 for a sample molecule. Segments are reattached to the anchor according to the innermost layer first -- and within a layer -- the largest segment first.

Figure 4. Layered Non-overlapping Segments



The anchor is processed separately (either oriented, scored, and/or minimized). The anchor position can be optimized prior to the conformation search with the `minimize_anchor` parameter.

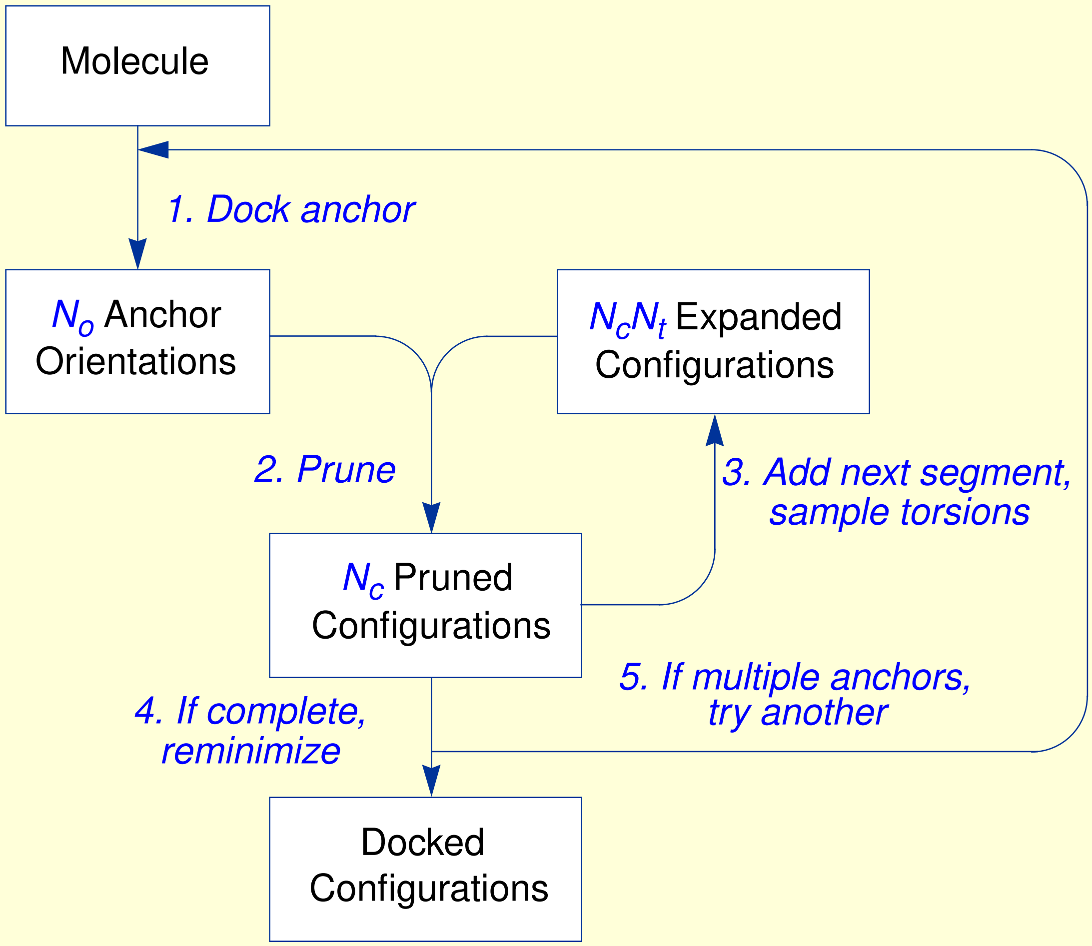
The remaining segments are subsequently re-attached during the conformation search. See Figure 5 for a diagram of the anchor-first docking process. The conformation search corresponds to steps 2 and 3 which form a complete cycle. An extensive analysis of the docking can be performed by setting the `write_partial_structures` parameter which causes all partially-built structures to be written out during the conformation search. Two files will be generated for each cycle of the conformation search. For the first cycle, one file will contain the anchor orientations from step 1 in Figure 5 and the other file will contain the pruned orientations from step 2. For all subsequent cycles, one file will contain the conformationally-expanded configurations from step 3, and the other file will contain the pruned configurations from step 2. The names of the files will be based on the name given for the ligand output file, but will have a sequence of numbers appended to it as shown in Table 6.

Table 6. Filename Construction when Writing Partial Structures

Files are **`name-A-LL-S-E.ext`**, with the components defined as follows:	
`name`	The base file name of the ligand output file.
`A`	Anchor number (single digit).
`LL`	Layer number (two digits).
`S`	Segment number within layer (single digit).
`E`	Ensemble number within cycle (1=conformation expanded; 2=pruned).
`ext`	File extension (`*.mol2`, `*.pdb`, etc.).

If `torsion_drive` has been selected, then the torsion positions of the intervening bond are searched when each segment is reattached. If `torsion_minimize` has been selected, then the intervening torsion may be relaxed. Minimization of the bond is performed in isolation, or in concert with inner torsions

Figure 5. Anchor-First Docking Algorithm



N_c is approximately `configurations_per_cycle`.

N_o is `maximum_orientations` if `automated_matching` selected, or is determined by manual matching parameters.

N_t is average number of positions per torsion in flex_drive.tbl file.

if the `reminimize_layer_number` parameter is set to a non-zero value. Relaxing multiple layers helps prevent the search from getting stuck in dead-ends. Although computationally expensive, the position of the anchor may be simultaneously optimized during the conformation search with the `reminimize_anchor` parameter. When all segments have been added, the entire molecule may be relaxed if the `reminimize_ligand` parameter is set.

Docking with the Anchor-First procedure

The process of docking a molecule using the anchor-first strategy is shown in Figure 5. The amount of searching is under full user control. The process begins with docking the anchor. This step is controlled with the `orient_ligand` parameters (please refer to Orientation Search on page 28) and results in N_o anchor positions. The conformation search begins by pruning these orientations according to rank and position (see Pruning the conformation search tree below) to produce N_c positions. Each subsequent cycle of the conformation search involves expanding the ensemble of partially built binding positions by adding a new segment and performing a torsion search on the newly formed bond and then contracting the ensemble with pruning. The torsion search on each newly formed bond results in an expansion of the set of N_c partial configurations to N_cN_t configurations. The size of N_t is based on the number of increments used for the current bond and can be modified by altering the entries in the flex_drive.tbl file. The expanded set of binding positions is then pruned back to N_c configurations. The conformation search continues expanding and pruning the set of partial binding position until each binding position represents a complete molecule.

This search technique is particularly useful for docking, but it also may be used for conformation analysis and stand-alone minimization.

Pruning the conformation search tree

During each cycle of the conformation search, the expanded set of partial configurations is pruned based on the setting of `configurations_per_cycle`. The pruning attempts to retain the best, most diverse configurations using a top-first pruning method which proceeds as follows. The configurations are ranked according to score. The top-ranked configuration is set aside and used as a reference configuration for the first round of pruning. All remaining configurations are considered candidates for removal. A weighted root-mean-squared distance (wRMSD) between each candidate and the reference configuration is computed according to Equation 1.

$$\mathrm{wRMSD} = \left(\frac{\sum_{i=1}^{Natoms} L_i[(x_i - x_{ri})^2 + (y_i - y_{ri})^2 + (z_i - z_{ri})^2]}{\sum_{i=1}^{Natoms} L_i} \right)^{\frac{1}{2}} \qquad \text{Equation 1}$$

where L_i is the layer to which atom i is assigned. The RMSD is weighted in this fashion to make it more sensitive to the position of the outer segments. The outer segments are more important because they have a greater influence over the position of subsequently added segments.

Each candidate is then evaluated for removal based on its rank and wRMSD using the inequality shown in Equation 2. If the factor is greater than `configurations_per_cycle`, the candidate is removed. Based on this factor, a configuration with rank 2 and 0.2 Angstroms wRMSD is comparable to a configuration with rank 20 and 2.0 Angstroms wRMSD. The next best scoring configuration which survives the first pass of removal is then set aside and used as a reference configuration for the second round of pruning, and so on.

$$\left(\frac{\text{Rank of candidate}}{\text{wRMSD}} \right) > \texttt{configurations\_per\_cycle} \qquad \text{Equation 2}$$

This pruning method attempts to balance the twin goals of recovering the best scoring and the most different binding configurations without introducing additional user parameters. The pruning method replaces the hierarchical clustering method used in the initial release of DOCK version 4.0, because it is faster (N logN versus N^2) and biases its search time towards molecules which sample a more diverse set of binding modes. As the value of `configurations_per_cycle` is increased, the anchor-first method approaches an exhaustive search.

Time requirements

The time demand grows linearly with `configurations_per_cycle`, the number of flexible bonds and the number of torsion positions per bond, as well as the number of anchor segments explored for a given molecule. Using the notation in Figure 5, the time demand can be expressed as

$$\text{Time} \propto N_a(N_o + N_c N_t N_b) \qquad \text{Equation 3}$$

where the additional terms are:

N_a	is the number of anchor segments tried per molecule.
N_b	is the number of rotatable bonds per molecule.

Manual specification of anchor segment

The user can override the automatic anchor selection performed by DOCK by specifying a STATIC ATOM SET named ANCHOR in the molecule input file. For an example, please see SYBYL MOL2 format on page 99. See the previous discussion of Manual specification of non-rotatable bonds for related instructions. It must be pointed out that the user can include as many atoms as desired in the ANCHOR set, but only the first atom will be used. The anchor segment which includes the ANCHOR atom will then be used as the segment anchor. In order to make a larger anchor than would be produced using the automatic segmentation based on the location of rotatable bonds, the user will need to manually specify the necessary bonds as non-rotatable (with a RIGID BOND set).

Simultaneous Search

If an anchor-first search is not selected, then a simultaneous search is performed by default. All torsions are searched and/or minimized in concert. The conformation search is performed prior to the orientation search, so each conformation is docked independently. The simultaneous search technique is useful for conformation analysis and for constructing a chemical screen database (see Chemical Screen on page 44).

Constraining the search

During a simultaneous search, DOCK performs an exhaustive or a random search, depending on the flexibility of the molecule and the value of `conformation_cutoff_factor`. The cutoff on the number of conformations generated for a molecule is calculated by Equation 4.

$$N_{cut} = (\texttt{conformation\_cutoff\_factor})(\text{\# of rotatable bonds}) \qquad \text{Equation 4}$$

If the total conformations for a molecule is below N_{cut} an exhaustive search of all conformations is performed. Otherwise, a random search of N_{cut} conformations is performed, in which torsion positions are selected randomly from the allowed positions. This definition of N_{cut} was used so that the time demand grows linearly with the number of rotatable bonds, which is a fair compromise between the exponential growth of the total possible conformations and a fixed cutoff applied uniformly to all molecules.

Caveat on backtracking procedure

A backtracking procedure is used during the generation of each conformation. If an internal clash is detected for the torsion position of a rotatable bond within a partially-built structure, then another torsion value is attempted. If no torsion values will work, then the procedure backtracks to the preceding rotatable bond and assigns a new torsion for it. This procedure works fine when internal clashes can be resolved easily, but in the worst case, for a molecule with a huge number of rotatable bonds and a clash that cannot be resolved at the last rotatable bond, the procedure will be confounded and tend to consume cpu time. For this reason, when processing a database of molecules, be sure to use a reasonable value of `flexible_bond_maximum` to discard overly flexible molecules.

Torsion Minimization

The torsion angles of rotatable bonds may be included in the score optimization. This provides for a much more efficient conformation search since fewer torsion positions need to be sampled. Each torsion flagged for movement is assigned a simplex vertex along with the six rigid body degrees of freedom. Only non-bonded interatomic terms are included in the scoring evaluation; no explicit torsion terms are included. Therefore, only torsions flagged as minimizable in the flex.defn file are included (e.g. double bonds are excluded by default). When an anchor-first search is performed with segments from multiple layers being minimized, then inner torsions are assigned smaller initial torsion step values since perturbations in these torsions have a greater impact on conformation.

Scoring

DOCK uses several types of scoring functions to discriminate among orientations and molecules. Scoring is requested using the score_ligand parameter. Each scoring function is treated independently during the calculation and results are written to separate output files. In order to combine the results of two or more scoring functions, apply the additional functions in separate post-docking scoring calculations.

DOCK will score the interactions of a ligand with the receptor if the intermolecular_score parameter is set. If flexible_ligand is set, then DOCK will score the interactions between rigid segments within a molecule if the intramolecular_score parameter is set. The total score is the sum of the intramolecular score and the intermolecular score, EXCEPT when different molecules are being compared for a rank_ligands list. After a flexible molecule has been docked, and it is being considered for the ranked ligand list, then the total score is set equal to only the intermolecular score plus whatever size penalty the user has specified with the contact_size_penalty parameter and so on for each type of scoring.

To enable rapid score evaluation during docking, the score potentials are precalculated on a three-dimensial grid using grid. However, continuum scoring may be performed by turning off the gridded_score flag. Continuum scoring may be used to evaluate a ligand:receptor complex without the investment of a grid calculation, or to perform a more detailed calculation without the numerical approximation of the grid. Continuum scoring is also triggered when an intramolecular_score is requested, but is only used for intramolecular score terms. When continuum scoring is requested, then score parameters normally supplied to grid, must also be supplied to DOCK. It is left to the user to make sure consistent values are supplied to both programs. Older grids calculated by CHEMGRID may also be read by specifying a value of 3.5 for the grid_version parameter.

The score is used to identify interesting orientations and molecules. If a top-scoring list is requested, like rank_orientations or rank_ligands, then DOCK will maintain a sorted list of a user-defined length for output. But if sorted lists are not requested, then DOCK will need to know what score cutoff to use to write out orientations or molecules. This cutoff is supplied by the user with the contact_maximum parameter and so on for each scoring function. This score cutoff may be overridden for orientations near the input orientation using the rmsd_override parameter.

Bump Filter

Orientations may be filtered prior to scoring to discard those in which the molecule significantly overlaps receptor atoms. This feature is enabled with the bump_filter flag, but is only available if the gridded_score flag is also set. At the time of construction of the bump filter, the amount of atom VDW overlap is defined with the bump_overlap parameter. At the time of bump evaluation the number of allowed bumps is defined with the bump_maximum parameter. If score optimization is being performed, then a few number of bumps should be allowed, since the minimizer can recover from such clashes. In addition, a few bumps often indicate an orientation that interacts intimately with the site and often leads to a strongly favorable orientation after minimization.

Contact Score

The contact score function is enabled with the contact_score flag. The contact score is a simple summation of the number of heavy atom contacts between the ligand and receptor. At the time of construction of the contact scoring grid, the distance threshold defining a contact is set with the contact_cutoff_distance. Atom VDW overlaps are penalized by checking the bump filter grid, or with the contact_clash_overlap parameter for the intramolecular score. The amount of penalty is specified with the contact_clash_penalty parameter.

The contact score provides a simple assessment of shape complementarity. It can be useful for evaluating primarily non-polar interactions.

Energy Score

The energy score is activated with the energy_score parameter. It is based on the non-bonded terms of the molecular mechanic force field (please refer to Equation 1 on page 74 for more background). During grid construction (and continuum scoring) every term in the function may be tailored by the user. The distance dependence of the Coulombic function is set with the distance_dielectric parameter. The dielectric constant is adjusted with the dielectric_factor parameter. The distance dependence of the Lennard-Jones function is set with the attractive_exponent and repulsive_exponent parameters. Typically a 6-12 potential is used, but it can be softened up by using a 6-8 or 6-9 potential. Regardless of the exponent values selected, the same radii and well-depths are used. The VDW well-depths and radii are stored in an editable file (see vdw.defn on page 105) which is identified with the vdw_definition_file parameter. In addition, the model for non-polar hydrogens may be selected with the atom_model parameter. With a united-atom model, the non-polar hydrogens are given zero VDW potentials and any partial charge residing on them is transferred to the adjacent carbon. The united atom model provides a smoother intermolecular potential which requires fewer steps of minimization. However, the all-atom model is more accurate, and perhaps captures some aromatic interaction coulombic terms otherwise missing.

Chemical Score

Chemical scoring allows the energy scoring function to be further tailored to enhance recognition of chemical complementarity. The attractive portion of the VDW term can often dominate the energy for uncharged molecules. With chemical scoring this term is scaled depending on the chemical labels assigned to the interacting atoms. It is activated by the chemical_score parameter. The chemical labels and definitions are the same as those for chemical matching (see chem.defn on page 106). The chemical interaction table resides in an editable file (see chem_score.tbl on page 108) and is identified with the chemical_score_file parameter.

Chemical scoring can be used to incorporate qualitative aspects of solvation. For instance hydrophobic-polar interactions can be made non-attractive or even repulsive. Further, it can be used to screen for molecules that contain a particular functional group (in concert with chemical matching) for presentation to a receptor active site. The interaction table could also be derived using statistical techniques from binding and structure data to improve the modeling of a particular site or class of sites.

Chemical scoring is used for intermolecular scoring only. If intramolecular score is requested, then the regular energy score is computed for internal energy.

This type of scoring should be considered experimental. Parameterization is left to the user. It should be used at your own risk.

RMSD Score

The RMSD score evaluates of the difference in conformation and orientation of two identical molecules. It is available when gridded_score is not requested and is activated by the rmsd_score parameter. The reference molecule is supplied with the receptor_atom_file, and the molecule to check is supplied with the ligand_atom_file, which may contain multiple molecules.

Since the RMSD is treated as a score, it may in fact be minimized. This procedure is useful for evaluating the difference of a CONCORD conformation from crystal conformation.

Score Optimization

Score optimization allows the conformation and orientation of a molecule to be adjusted to improve the score. Although the calculation is expensive, it makes the conformation and orientation search more efficient because less sampling becomes necessary. Optimization is activated with the minimize_ligand parameter. The optimizer currently uses the simplex algorithm which does not require evaluation of derivatives. It does however depend on a random number generator which makes it not only sensitive to the initial seed provided with random_seed parameter, but also to the order of evaluation. So results will vary if molecules are supplied in a different order. The amount of variance should be small, though. For detailed calculations, it is recommended that the optimization be repeated with different random number seeds to check convergence.

The initial step size of the minimizer is specified with the initial_translation, initial_rotation, and initial_torsion parameters. The length of minimization may be controlled with the maximum_iterations parameter.

Even if several scoring functions have been requested, not all need to be minimized. Specification of which functions are to be minimized is done with contact_minimize and so forth for each scoring function. The termination criteria of minimization is specified with contact_convergence and so forth.

Since minimization may converge prematurely, each call to the minimizer is actually composed of multiple cycles. The number of cycles is controlled with the maximum_cycles parameter. Additional cycles of minimization are spawned if the previous simplex has made a significant difference in the conformation or orientation AND if the score has passed below a threshold set by the user. The difference in configuration is measured by the vector magnitude of the final simplex vertex array. The difference must exceed the cycle_convergence parameter to be significant. A value of 1.0 for this parameter would correspond to at least one of the simplex vertices moving a distance equal to the initial step size. The score threshold prevents repeated cycles of minimization of a configuration that is really of no interest. The score threshold is set with contact_termination an so forth for each scoring function.

The performance of the minimizer can be monitored using the `-p` flag (see Command-line Arguments on page 53 and Performance on page 73).

Database Processing

The most common application of DOCK is to process a database of molecules to find potential inhibitors or ligands of a target macromolecule. However, with the new separation of components in version 4.0, the database processing tools can be combined with other tasks, like stand-alone scoring, score optimization, or chemical screen.

Database processing is signalled with the `multiple_ligands` parameter. A subset of the database may be processed using the `ligands_maximum`, `initial_skip`, and `interval_skip` reading parameters and the `heavy_atoms_minimum` and `heavy_atoms_maximum` size-selection parameters. If scoring has been selected, then molecules can be output as a ranked list using the `rank_ligands` parameter. When comparing molecules, the score of large molecules may be penalized using the `contact_size_penalty` parameter and so on for each scoring function. If ligands are not ranked, then all orientations recorded for each molecule are written (see Orientation Search on page 28 for how multiple orientations are recorded). When no orientation search is performed (i.e. stand-alone scoring), then molecules are written if they pass a score cutoff set by the `contact_maximum` parameter and so on for each scoring function.

Database jobs produce two output files in addition to the molecule output files. The `restart_file` parameter specifies the file which stores the current `rank_ligands` list. If the job is terminated prematurely, then it may be restarted with the `-r` flag (see Command-line Arguments on page 53) and the run is initialized with information in the existing restart file. The frequency at which the restart file is updated is specified with the `restart_interval` parameter. In addition, the `info_file` parameter specifies the file which stores information about the current progress of the run.

Database jobs may also be interacted with during execution via the presence of two input files. The `dump_file` parameter specifies a file whose presence will trigger the job to write out the current results to the info file, the restart file and the molecule output files. The user may create this file at any time to inspect the current results. The dock job will automatically delete the dump file after a dump has taken place. In addition, the `quit_file` parameter specifies a file whose presence will trigger the job to write out results (like a dump request) and also terminate execution. This parameter is useful for gently terminating a job without loss of information for restart at a later time. The presence of either of these files is only checked in between processing molecules, so it may take up to a minute for such a file to be noticed.

Preliminary Docking

Docking an entire database can take a considerable amount of time. The length of time depends primarily on the sampling parameters for the orientation and conformation search and the minimization parameters. Even when docking is distributed over multiple workstations, the calculation can take several days or weeks. Since the optimal search parameters are site dependent, it is important to do some preliminary docking calculations with subset of the database to identify good parameters.

As sampling parameters are increased, the results will initially improve but will eventually converge. The optimal parameters correspond to where the results have just converged. Multiple short docking jobs can be submitted using UNIX shell scripts. The results that should be monitored are presented in Ewing and Kuntz [6]. The most important is the weighted rank correlation, which reports how well the rankings of the top-scoring molecules have been predicted.

The following is a discussion of different ways to construct a subset of the molecule database.

Extracting specific molecules

Since the PTR format database file contains the molecule name and description, entries can be retrieved based on these fields. Use UNIX `fgrep` to select the molecule, or molecules.

```
fgrep " BENZENE " database.ptr > benzene.ptr
fgrep -f subset.list database.ptr > subset.ptr
```

The file `molecules.list` would contain a list of the names you would like to extract. The subset molecules can be readily converted to a format for viewing with the following command.

```
dock -i subset.ptr -o subset.pdb
```

Extracting a random subset

A random subset of a molecular database can be used to help identify appropriate docking parameters, before docking an entire database. Selecting a random subset is easy using a PTR format database file. Use the following UNIX `nawk` command to select an average of one out every 1000 entries in the database.

```
nawk '{if (rand < 0.001) print}' database.ptr > subset.ptr
```

Extracting an interval subset

Alternatively, if you literally want one molecule for every 1000 molecules without any randomness, then use a different call to UNIX `nawk`.

```
nawk '{if ((++n % 1000) == 0) print}' database.ptr > subset.ptr
```

This can also be achieved by using the `interval_skip` multiple ligand parameter in dock. This latter method is much slower, however, because the coordinates of all the skipped molecules must be read.

Parallel Jobs

Since a database docking calculation is ideal for parallelization over multiple computers, the parallel jobs feature was added to ease the organizational burden of this task. This feature is activated with the `parallel_jobs` parameter. With this feature, a dock job can be launched on every workstation or cpu at a user's disposal. These jobs process a single database, each at its own pace. To prevent each job from duplicating each other's work unnecessarily, a server job is used to parse the database and hand out molecules, one at a time, to each client job. Each client job and the server job requires its own input file and output files. When all processing is complete, the user must coalesce the results from each client job.

Client or server behavior is designated using the `parallel_server` parameter. Setting it to "`yes`" causes server behavior; "`no`" causes client behavior. The server name is defined with the `server_name` parameter. Any number of client jobs can be delegated to the server using the `client_total` parameter. Each client name must be supplied with the `client_name_1` and subsequent parameters. It is recommended that the server job be executed on the computer which stores the molecule database. Then in the event of any network difficulties, the server job is never disconnected from the database.

The client jobs need to the have the `parallel_jobs` parameter activated, but not the `parallel_server` parameter. The `server_name` parameter must be consistent with that supplied to the server job. The name of the client job is specified with the `client_name` parameter, which must be one of the client names supplied to the server. The client jobs need to be launched from the same directory as the server job since they communicate via local temporary files. This requires that client jobs can only be launched on machines that cross-mount the working directory. Clients may be taken off-line (via the `quit_file`) and restarted without disrupting the server or other client jobs. If the server job is given a quit signal, then it automatically signals all client jobs to quit as well.

Client jobs may be instructed to either store a ranked list or to write out all results to file. Since the PTR format files take up so little disk space, an entire database can be written out without taken up more space than the top few thousand molecules written in SYBYL MOL2 format (about 25 megabytes of disk space). If the clients store ranked lists, then make sure that the list length for each client is equal to the total length of interest (a few hundred at least). This rule helps avoid artifacts from the parallelization when the results are coalesced.

When all jobs are complete, then the results must be combined. This process can be done seamlessly by using the UNIX `cat` command on the output molecule files. If PTR format is used, then molecules can be reranked using the UNIX `sort` command on the score field. If SYBYL MOL2 format is used, then perform stand-alone scoring on the catenated file and output the molecules in a ranked list.

Chemical Screen

The chemical screen option enables rapid screening of molecules either prior to or following docking. It is requested with the chemical_screen parameter. It can be used to perform a pharmacophore screen or a similarity screen. It is based on the same chemical labels and definitions as chemical matching and the chemical score (see chem.defn on page 106). These tools should be viewed as rudimentary in functionality compared to what is available in commercial small-molecule software packages. However, considering their ease of use and compatibility with DOCK, they should be of interest to the DOCK user.

The chemical search key encodes information about the number of distances between chemical groups in a molecule and the magnitudes of the distances. The number of distances between two different chemical groups, n_{ij}, records information about the composition of the molecule. The distances themselves are stored in a binary fingerprint, f_{ij}, which records information about the spatial distribution of properties in the molecule. Since the binary fingerprint may lose information about the frequency of occurrence of certain distances, particularly for flexible molecules, the number of distances record is necessary.

Keying a database is performed with the construct_screen parameter. The fingerprint architecture is specified with the distance_begin, distance_end, and distance_interval parameters. When keying the database, a simultaneous search of conformations should be performed to generate an ensemble of conformers. For each conformer, the distances between chemical groups is computed and stored in the binary fingerprint. The keyed database is actually a PTR format database file with the additional fields for chemical keys. Once the database has been keyed, it may then be searched using the screen_ligands parameter.

Pharmacophore Screen

A pharmacophore search is a useful way to prescreen a database to identify molecules that might interact favorably with a small number of receptor atoms with well-defined geometry. It is requested with the pharmacophore_screen parameter. The pharmacophore is actually a set of points with associated chemical labels serving as a three dimensional hypothesis for a binding model. The pharmacophore could be constructed from a set of active ligands, or extracted from a receptor structure. This feature is useful when only a few interactions with the receptor are of interest and can be used to construct a pharmacophore. The chemical screen would then be able to discard all molecules that would never be able to make the desired interactions.

Given a molecule ***a*** and a pharmacophore ***b***, the rules for determining whether the molecule might satisfy the pharmacophore are for all *i* and *j* not equal to *i*:

$$n_{ij}^{b} \le n_{ij}^{a} \qquad \text{Equation 5}$$

$$f_{ij}^{b} = f_{ij}^{a} \cap f_{ij}^{b} \qquad \text{Equation 6}$$

where n_{ij} and f_{ij} are defined in the Chemical Screen section, above.

In order to account for uncertainty in the distances, a distance_tolerance is used to blur the distance key of the molecule.

After the chemical screen run has discarded molecules that could never satisfy the pharmacophore, then a regular dock run can be performed to find the molecules that do satisfy it. In the follow-up run, use a simultaneous search of conformations along with chemical matching to the pharmacophore site points.

Similarity Screen

A similarity search is useful to identify all molecules in a database which might be similar in chemical property distribution to a particular molecule of interest. Such a search is useful after docking, once a set of active molecules has been identified. It is activated with the `similarity_screen` parameter. The cutoff for writing out hits is specified with the `dissimilarity_maximum` parameter. Dissimilarity is used to be consistent with the other scoring functions in which smaller values represent favorable values.

Since the chemical key contains both a binary fingerprint and an integer count of distances, a modified Tanamoto index is used as shown in Equation 7. This similarity metric maintains the connection between the fingerprint and the count of chemical distances.

$$Similarity = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} min(n_{ij}^{a}, n_{ij}^{b})\ count(f_{ij}^{a} \cap f_{ij}^{b})}{\sum_{i=1}^{N}\sum_{j=1}^{N} max(n_{ij}^{a}, n_{ij}^{b})\ count(f_{ij}^{a} \cup f_{ij}^{b})} \qquad \text{Equation 7}$$

For flexible molecules, the distance fingerprints can become saturated. The similarity can still be discerned by the distance count element of the key.

Some chemical groups may be treated as equivalent to other groups (e.g. hydroxyls and hydrogen bond donors). Such chemical equivalency can be supplied in an editable file (see chem_screen.tbl on page 109) and identified with the `chemical_screen_file` parameter.

Macromolecular Docking

Though DOCK is typically used to process small molecules, it can be used to study the interactions of macromolecular ligands. The chief difference in protocol is that to use the `match_receptor_sites` procedure for the orientation search, then special ligand centers must be used to represent the ligand. This is signalled by setting the `ligand_centers` parameter. The ligand centers must reside in a file identified with the `ligand_center_file` parameter.

The ligand centers may be constructed with sphgen, using spheres to describe the positive image of the macromolecule. See Shoichet and Kuntz [26], for examples and discussion of macromolecular docking.

If multiple orientations are written to PDB formatted file, then the residue numbers are not disturbed. Normally, dock gives each orientation in the output PDB file a sequential residue number. However, if multiple substructures (residues) are present in the molecule input file, then this procedure is precluded.

References

1. Blaney, J.M. Ph.D. dissertation, University of California, San Francisco, 1982.
2. Connolly, M.L. Analytical molecular surface calculation. *J. Appl. Cryst*. **16**: 548-558, 1983.
3. Connolly, M.L. Solvent-accessible surfaces of proteins and nucleic acids. *Science*. **221**: 709-713, 1983.
4. DesJarlais, R.L., and Dixon, J.S. A shape and chemistry-based docking method and its use in the design of HIV-1 protease inhibitors. *J. Comput.-Aided Molec. Design* **8**(3): 231-242, 1994.
5. DesJarlais, R.L., Sheridan, R.P., Seibel, G.L., Dixon, J.S., Kuntz, I.D. and Venkataraghavan, R. Using shape complementarity as an initial screen in designing ligands for a receptor binding site of known three-dimensional structure. *J. Med. Chem*. **31**(4): 722-729, 1988.
6. Ewing, T.J.A, and Kuntz, I.D. Critical evaluation of search algorithms used in automated molecular docking. *J. Comput. Chem.* **18**(9): 1175-1189, 1997.
7. Ferro, D.R. and Hermans, J. A different best rigid-body molecular fit routine. *Acta Cryst*. **A33**: 345-347, 1977.
8. Fletcher, R. "Practical Methods of Optimization." New York: Interscience, 1960.
9. Gilson, M.K., Sharp, K.A. and Honig, B.H. *J. Comp. Chem*. **9**: 327, 1987.
10. Goodford, P.J. A computational procedure for determining energetically favorable binding sites on biologically important macromolecules. *J. Med. Chem*. **28**: 849-857, 1985.
11. Gschwend, D.A, and Kuntz, I.D. Orientational sampling and rigid-body minimization in molecular docking revisited — On-the-fly optimization and degeneracy removal. *J. Comput.-Aided Molec. Design*, **10**:123-132, 1996.
12. Kabsch, W. A solution for the best rotation to relate two sets of vectors. Acta Cryst. **A32**: 922-923, 1976.
13. Kabsch, W. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Cryst.* **A34**: 827-828, 1978.
14. Klapper, I., Hagstrom, R., Fine, R., Sharp, K. and Honig, B. *Proteins*. **1**: 47-59, 1986.
15. Kuhl, F.S., Crippen, G.M., and Friesen, D.K. A Combinatorial Algorithm for Calculating Ligand Binding. *J. Comput. Chem.* **5**:24-34, 1984.
16. Kuntz, I.D., Blaney, J.M., Oatley, S.J., Langridge, R. and Ferrin, T.E. A geometric approach to macromolecule-ligand interactions. *J. Mol. Biol*. **161**: 269-288, 1982.
17. Kuntz, I.D. Structure-based strategies for drug design and discovery. *Science*. **257**: 1078-1082, 1992.
18. Kuntz, I.D., Meng, E.C. and Shoichet, B.K. Structure-based molecular design. *Acc. Chem. Res*. **27**(5): 117-123, 1994.
19. Leach, A.R., and Kuntz, I.D. Conformational analysis of flexible ligands in macromolecular receptor sites. *J. Comput. Chem.* **13**(6): 730-748, 1992.
20. Meng, E.C., Shoichet, B.K. and Kuntz, I.D. Automated docking with grid-based energy evaluation. *J. Comp. Chem*. **13**: 505-524, 1992.

21. Meng, E.C., Gschwend, D.A., Blaney, J.M. and Kuntz, I.D. Orientational sampling and rigid-body minimization in molecular docking. *Proteins*. **17**(3): 266-278, 1993.

22. Meng, E.C., Kuntz, I.D., Abraham, D.J. and Kellogg, G.E. Evaluating docked complexes with the HINT exponential function and empirical atomic hydrophobicities. *J. Comp-Aided Mol. Design*. **8**: 299-306, 1994.

23. Miller, M.D., Kearsley, S.K., Underwood, D.J. and Sheridan, R.P. FLOG - A system to select quasi-flexible ligands complementary to a receptor of known three-dimensional structure. J. Comput.-Aided Mol. Design

24. Nelder, J.A. and Mead, R. *Computer Journal* **7**: 308, (1965).

25. Richards, F.M. *Ann. Rev. Biophys. Bioeng*. **6**: 151-176, 1977.

26. Shoichet, B.K. and Kuntz, I.D. Protein docking and complementarity. *J. Mol. Biol*. **221**: 327-346, 1991.

27. Shoichet, B.K., Bodian, D.L. and Kuntz, I.D. Molecular docking using shape descriptors. *J. Comp. Chem*. **13**(3): 380-397, 1992.

28. Shoichet, B.K., Stroud, R.M., Santi, D.V., Kuntz, I.D. and Perry, K.M. Structure-based discovery of inhibitors of thymidylate synthase. *Science*. **259**: 1445-1450, 1993.

29. Shoichet, B.K. and Kuntz, I.D. Matching chemistry and shape in molecular docking. *Protein Eng*. **6**(7): 723-732, 1993.

30. Weiner, S.J., Kollman, P.A., Case, D.A., Singh, U.C., Ghio, C., Alagona, G., Profeta, S., Jr. and Weiner, P. A new force field for molecular mechanical simulation of nucleic acids and proteins. *J. Am. Chem. Soc*. **106**: 765-784, 1984.

31. Weiner, S.J., Kollman, P.A., Nguyen, D.T. and Case, D.A. An all atom force field for simulations of proteins and nucleic acids. *J. Comp. Chem*. **7**: 230-252, 1986.

Reference Manual

This page intentionally blank.

DOCK

Overview

Version 1.0/1.1

Robert Sheridan, Renee DesJarlais, Irwin Kuntz

The program DOCK is an automatic procedure for docking a molecule into a receptor site. The receptor site is characterized by centers, which may come from SPHGEN or any other source. The molecule being docked is characterized ligand centers, which may be its non-hydrogen atoms or volume-filling spheres calculated in SPHGEN. The ligand centers and receptor centers are matched based on comparison of ligand-center/ligand-center and receptor-center/receptor-center distances. Sets of ligand centers match sets of receptor centers if all the internal distances match, within a value of `distance_tolerance`. Ligand-receptor pairs are added to the set until at least `nodes_minimum` pairs have been found. At least three pairs must be found to uniquely determine a rotation/translation matrix that will orient the ligand in the receptor site. A least-squares fitting procedure is used (Ferro and Hermans, 1977). Once an orientation has been found, it is evaluated by any of several scoring functions. DOCK may be used to explore the binding modes of an individual molecule, or be used to screen a database of molecules to identify potential ligands.

Version 2.0

Brian Shoichet, Dale Bodian, Irwin Kuntz

DOCK version 2.0 was written to give the user greater control over the thoroughness of the matching procedure, and thus over the number of orientations found and the CPU time required (Shoichet, Bodian, and Kuntz, 1992). In addition, certain algorithmic shortcomings of earlier versions were overcome. Versions 2.0 and higher are particularly useful for macromolecular docking (Shoichet and Kuntz, 1991) and applications which demand detailed exploration of ligand binding modes. In these cases, users are encouraged to run CLUSTER in conjunction with SPHGEN and DOCK.

To allow for greater control over searches of orientation space, the ligand and receptor centers are pre-organized according to their internal distances. Starting with any given center, all the other centers are presorted into “bins” based on their distance to the first center. All centers are tried in turn as “first” positions, and all the points in a bin which has been chosen for matching are tried sequentially. Ligand and receptor bins are chosen for matching when they have the same distance limits from their respective “first” points. The number of centers in each bin determines how many sets of points in the receptor and the ligand will ultimately be compared. In general, the wider the bins, the greater the number of orientations generated. Thus, the thoroughness of the search is under user control.

Version 3.0

Elaine Meng, Brian Shoichet, Irwin Kuntz

Version 3.0 retained the matching features of version 2.0, and introduced options for scoring (Meng, Shoichet, and Kuntz, 1992). Besides the simple contact scores mentioned above, one can also obtain molecular mechanics interaction energies using grid files calculated by CHEMGRID (which is now superseded by GRID in version 4.0). More information about the ligand and receptor molecules is required to perform these higher-level kinds of scoring. Point charges on the receptor and ligand atoms are needed for electrostatic scoring, and atom-type information is needed for the van der Waals portion of the force field score. Input formats (some of them new in version 3.5) are discussed in various parts of the documentation; one example of a “complete format” (including point charges and atom type information) is SYBYL ASCII (MOL2) format (Tripos Associates, Inc., St. Louis, MO 63117). Parameterization of the receptor is discussed in the documentation for CHEMGRID. In DOCK, ligand parameters are read in along with the coordinates; input formats are described below. Currently, the options are: contact scoring only, contact scoring plus Delphi electrostatic scoring, and contact scoring plus force field scoring. Atom-type information and point charges are not required for contact scoring only.

Version 3.5

Mike Connolly, Daniel Gschwend, Andy Good, Connie Oshiro, Irwin Kuntz

Version 3.5 added several features: score optimization, degeneracy checking, chemical matching and critical clustering.

Version 4.0

Todd Ewing, Irwin Kuntz

Version 4.0 was a major rewrite and update of DOCK. A new matching engine was developed which is more robust, efficient, and easier to use. Orientational sampling can now be controlled directly by specifying the number of desired orientations. Additional features include chemical scoring, chemical screening, and ligand flexibility.

Command-line Arguments

Table 1. DOCK command-line arguments

flag	optional argument	behavior
`-i`	`input_file`	INPUT FILE — Parameters are extracted from `input_file`, or `dock.in` if not specified.
`-o`	`output_file`	OUTPUT FILE — Results are written to `output_file`, or `dock.out` if not specified. If a `-o` flag is present then DOCK runs in batch mode, otherwise it runs interactively (see below).
`-s`		STANDARD INPUT — Parameters are entered interactively without the construction of an input file. This option is generally not recommended.
`-r`		RESTART — Run is initialized with information from a restart file. This option is used to restart a `rank_ligands` docking run that was terminated prematurely.
`-p`		PROFILE — Time spent in docking routines and minimizer statistics are profiled. This option helps the user to identify the bottleneck of a particular calculation and to choose optimal minimizer parameters. See Performance on page 73.
`-t`		TERSE — Reduced output level. This option helps reduce the size of the output file when docking a large number of molecules.
`-v`		VERBOSE — Increased output level. This option allows additional data to be included in the output. Recommended for single molecule runs.

DOCK may be executed in either interactive or batch mode, depending on whether output is written to a file. In interactive mode, the user is requested only for parameters relevant to the particular run and default values are provided. This mode is recommended for the initial construction of the input file and for short calculations. In batch mode, input parameters are read in from the input file and all output is written to the output file. This mode is recommended for long calculations once an input file has been generated interactively.

Interactive mode

- `dock -i dock.in`

  When launched this way, DOCK will extract all relevant parameters from `dock.in` (or any file supplied by the user). If additional parameters are needed (or if the `dock.in` file is non-existent or empty), DOCK will request them one at a time from the user. Reasonable default values are presented. Any parameters supplied by the user will be automatically appended to the `dock.in` file. If the user would like to change any previously entered values, the user can edit in the `dock.in` file using a text editor.

- `dock -i`

  DOCK will behave as above, but will assume the input file to be `dock.in`.

- `dock -s`

DOCK will run interactively, but will not check any input file for parameters and will not append any entered parameters to a file.

Batch mode

- `dock -i dock.in -o dock.out`

  DOCK will run in batch mode, extracting all relevant parameters from `dock.in` (or any file supplied by the user) and will write out all output to `dock.out` (or any file supplied by the user). If any parameters are missing or incorrect, then execution will halt and an appropriate error message will be reported in `dock.out`.
- `dock -i -o`

  DOCK will behave as above, but will assume the i/o files to be `dock.in` and `dock.out`.
- `dock`

  If a file called `INDOCK` is present in the current working directory, then DOCK will use it as an input file. Output will be written to a file called `OUTDOCK`. This mode is present for reverse compatibility with previous versions of DOCK.

Molecule File Conversion

- `dock -i old.mol2 -o new.pdb`

  If the input and output file have recognized coordinate file extensions (`*.mol2, *.pdb, *.xpdb, *.sph, *.ptr`), then DOCK will automatically construct its own parameter input file to perform a file conversion operation.

Molecule File Input/Output

Although molecule files can be input and output at the command line to perform file conversion, they are more commonly specified within the input file to specify such things as the ligand input coordinates, the receptor site point and atom coordinates, and the ligand output coordinates for each scoring function. When each file name is read in from the input file, it is checked to make sure it has one of the recognized file extensions (`*.mol2, *.pdb, *.xpdb, *.sph, *.ptr`), so that DOCK can identify the proper the file format.

Passing coordinates through a UNIX pipe

DOCK can be run as part of a UNIX pipe of commands, in which molecule coordinates are passed between a series of commands. To configure a dock run to operate within a UNIX pipe, dock must be instructed to read molecule coordinates from the standard input stream and/or write coordinates to the standard output stream. To do this, the streams must not be in use for interactive processing (ie. dock output must be directed to an output file with the `-o` command line flag). Within the input file, DOCK can be instructed to use the standard streams for coordinates by using `stdin.ext` when specifying the input file name and `stdout.ext` for the output file name. The `*.ext` file extension must be one of the coordinate file extensions recognized by DOCK so that the proper file format can be used.

Input Parameters

File Format

Input parameters may be supplied in a text file. There are few simple format rules for this file.

Table 2. Format of Input File

Rule	Consequence
Parameter names must be the first word on a line to be recognized.	Any character or word preceding a parameter effectively comments it out.
The word following a recognized parameter name is interpreted as the parameter value.	Comments can appear on the same line, after a parameter and value.
Blank lines are allowed.	Comments can appear on separate lines.
Empty files are allowed.	Missing parameters will be flagged (batch mode) or requested (interactive mode).
The order of parameters is irrelevant.	

Parameters

DOCK has a large number of input parameters because of its wide array of functionality. Parameters are read in hierarchically, with the most general parameters first. They are also ordered according to broad classes of functionality in order to make the process of selection more intuitive. For instance all scoring-related parameters are requested together. Since only the relevant parameters are requested based on preceding selections, it is recommended that initial parameter selection be done interactively to generate a sensible input file as painlessly as possible.

The following list of parameters are grouped according to the order they are requested at run time. Again, many parameters are listed here that will not be requested during your particular run.

The default values listed here may be different from those provided during a particular run, because DOCK tries to recommend sensible parameter values based on preceding selections.

Table 3. DOCK input Parameters

3A: General

Parameter	Type	Default	Description
Parameters in this category determine the general behavior of the dock run. Each option can be used in isolation and many in combination to allow DOCK to perform a diverse repertoire of tasks. If none of these flags are selected, then DOCK simply reads in a single molecule and writes it out, which is useful for file format conversion.			
`flexible_ligand`	boolean	no	Flag to allow ligand flexibility.
`orient_ligand`	boolean	no	Flag to perform an orientation search.

3A: General (Continued)

Parameter	Type	Default	Description
score_ligand	boolean	no	Flag to perform scoring of orientations or conformations.
minimize_ligand	boolean	no	Flag to perform local score minimization of orientations or conformations.
multiple_ligands	boolean	no	Flag to process multiple ligands.
chemical_screen	boolean	no	Flag to perform chemical screening (multiple_ligands must be selected, but not orient_ligand or score_ligand).
parallel_jobs	boolean	no	Flag to perform docking as parallel jobs orchestrated by a server dock job (multiple_ligands must be selected).
random_seed	integer	0	Integer to seed the random number generator.

3B: Ligand Flexibility

Parameter	Type	Default	Description
Ligand flexibility parameters can be used in various combinations to relax an input conformation, to perform a conformational search of a molecule, or to perform a completely flexible docking of a molecule. See Conformation Search on page 32 for more discussion.			
anchor_search	boolean	no	Flag to organize rigid segments into concentric layers around an anchor. Only the anchor is included in initial scoring/orienting. The outer segments are reattached in a subsequent conformation search. • yes = Anchor-First Search is performed. • no = Simultaneous Search of all torsions is performed prior to other processing.
multiple_anchors	boolean	no	Flag to select several anchors for a molecule. • yes = All rigid segments meeting anchor_size criteria will be tried independently as anchors. • no = The largest segment is used.

3B: Ligand Flexibility (Continued)

Parameter	Type	Default	Description
anchor_size	integer	10	Minimum number of heavy atoms for an anchor if multiple_anchors requested. If no segment satisfies this value, then the largest segment is used.
write_partial_structures	boolean	no	Flag to write out all partially-built structures during the anchor-first search. This option is useful to monitor the intermediate stages of a flexible ligand dock run. The structures from each cycle of growth are written to a separate file, with each file name constructed as shown in Table 6. on page 34.
torsion_drive	boolean	no	Flag to perform a conformational search by driving each torsion through a set of low-energy dihedral values read from the flex_drive_file.
clash_overlap	real	0.5	Amount of atom VDW overlap allowed during a torsion_drive. If two ligand atoms approach closer than this fraction of the sum of the VDW radii, then the conformation is discarded. Similar to bump_overlap in grid. • 0 = Complete overlap allowed. • 1 = No overlap allowed.
configurations_per_cycle	integer	25	Sets the amount of sampling during an anchor_search with a torsion_drive. This value sets the approximate number of configurations retained during each cycle of the search. See Pruning the conformation search tree on page 36 for a full description.

3B: Ligand Flexibility (Continued)

Parameter	Type	Default	Description
conformation_cutoff_factor	integer	5	Sets the amount of sampling during a simultaneous search with a torsion_drive. This value, multiplied by the number of rotatable bonds in the molecule, sets the cutoff on the number of conformations generated, Ncut. • If the number of molecule conformations, N, is less than Ncut, an exhaustive systematic search of the N conformations is performed. • Otherwise, a random search of Ncut conformations is performed, with torsion values selected randomly from the flex_drive_file.
torsion_minimize	boolean	no	Flag to perform torsion relaxation based on intramolecular_score and/or intermolecular_score.
reminimize_layer_number	integer	2	Number of previous layers to minimize while minimizing the current segment. Only neighboring segments in inner layers (and their neighbors in outer layers) are minimized. This feature helps rescue conformations from dead ends during the search.
minimize_anchor	boolean	yes	This flag specifies whether the anchor is minimized during the initial anchor docking. In general this flag should be turned on.
reminimize_anchor	boolean	yes	If a partially-built molecule is minimized during conformation search, but no minimizable bonds are active, then a rigid-body minimization is performed. This flag will force such a rigid-body reminimization throughout the search. This method is a more expensive (but effective) way to rescue conformations.
reminimize_ligand	boolean	yes	Flag to reminimize the molecule after the conformation search, letting all torsions and the anchor position relax simultaneously. This process resolves any accumulated strain built up during the search.

3B: Ligand Flexibility (Continued)

Parameter	Type	Default	Description
flexible_bond_maximum	integer	10	The maximum number of flexible bonds allowed in a molecule, when multiple_ligands are processed.
write_conformations	boolean	no	When a conformation search is performed (torsion_drive selected) without an orientation search, then multiple conformations may be stored with this flag. Otherwise, only the best scoring conformation is written.
write_conformation_total	integer	100	This parameter specifies the number of conformations to store per molecule.

3C: Orientation Search

Parameter	Type	Default	Description
Orienting the ligand is fundamental to the docking process. Orienting is traditionally done by matching. Alternative orienting procedures are also available, particularly the random search, which can optionally be run without any site points. For typical uses, the traditional matching process is still recommended. Please refer to Orientation Search on page 28 for more discussion.			
match_receptor_sites	boolean	no	Flag to perform traditional site point-directed matching.
random_search	boolean	no	Flag to randomly search ligand orientations. • With match_receptor_sites, all matches are constructed randomly rather than based on distance comparisons. • Otherwise, orientations are randomly constructed inside the smallest rectangular volume that encloses all points read in as site points. See Random Search on page 29 for more discussion.
ligand_centers	boolean	no	Flag to use ligand centers read in from a separate file for matching instead of ligand heavy atoms. Please refer to Macromolecular Docking on page 46 for more discussion.

3C: Orientation Search (Continued)

Parameter	Type	Default	Description
automated_matching	boolean	yes (single ligand), no (multiple ligands)	Flag to let matching proceed in an automated fashion until a desired number of orientations have been formed. It is recommended for single ligand dock runs. For database searching, manual matching is recommended because dock then spends more time on ligands which are complementary to the site.
maximum_orientations	integer	5000, (500 if automated matching)	With automated_matching or random_search, this sets number of orientations to generate for each molecule. Otherwise, it sets the upper limit on the number of orientations made by matching.
write_orientations	boolean	no	When performing an orientation search with rigid ligands, this flag allows multiple orientations to be saved. Otherwise, only the best-scoring orientation of each ligand is saved.
rank_orientations	boolean	no	If multiple orientations are saved, then this flag causes orientations to be ranked by score. Otherwise, all orientations that pass a cutoff (see contact_maximum, etc) are written out in the order they are encountered.
rank_orientation_total	integer	100	Number of ranked orientations to store.
write_configurations	boolean	no	When performing an orientation search in combination with a conformation search (torsion_drive selected), this flag allows multiple configurations (conformations + orientations) to be saved. Otherwise, only the best-scoring configuration of each ligand is saved.
write_configuration_total	integer	100	Number of configurations to store for each molecule (ranked by score).

3D: Matching

Parameter	Type	Default	Description
Matching is the traditional procedure driving the orientation search in DOCK. If automated_matching is selected, then the amount of sampling is controlled by maximum_orientations only (and no other matching parameters can be set by the user). If it is not selected (manual matching), then the amount of sampling is controlled by the node and distance parameters. Other constraints on manual matching are available based on chemical labeling or critical clusters. See Manual Matching on page 29 for further discussion.			
nodes_minimum	integer	3	Smallest number of atom-site point interactions needed to construct an orientation.
nodes_maximum	integer	10	Largest number of atom-site point interactions considered to construct an orientation
distance_tolerance	real	0.25	Maximum difference between all intra-ligand and intra-receptor distances in a match. This is the chief sampling parameter for matching.
distance_minimum	real	2.0	Minimum intra-ligand or intra-receptor distance allowed in a match. This parameter biases matching toward longer distances which convey more information about ligand or site shape.
check_degeneracy	boolean	no	Flag to discard matches that are subsets of larger matches.
reflect_ligand	boolean	no	Flag to dock the mirror image of a ligand to rescue an improper match. Half of all matches with 4 or more nodes require reflection, otherwise they are discarded. Use of this parameter is not generally recommended.
critical_points	boolean	no	Flag to force matching to include members from a particular group (or groups) of site points in every match. This parameter is useful to focus docking around a few key residues in an active site.
multiple_points	boolean	no	Flag used in combination with critical_points to allow multiple points from the same critical cluster to appear in a match.
chemical_match	boolean	no	Flag to use chemical labeling to identify bad interactions within a match so that the orientation can be discarded before it is even generated.

3E: Scoring

Parameter	Type	Default	Description
Each orientation is scored according to the options selected in this section. Several scoring functions exist and are treated independently of each other. To filter molecules based on more than one function simultaneously, you will need to rescore in a subsequent step. Most scoring is speeded up by precalculating a potential on a 3D grid, but continuum scoring is available.			
`intramolecular_score`	`boolean`	`no`	Flag to compute score between rigid segments. This feature is available if `flexible_ligand` is selected.
`intermolecular_score`	`boolean`	`no`	Flag to compute score between ligand and receptor.
`gridded_score`	`boolean`	`yes`	Flag to use precomputed grids to evaluate the score, otherwise a continuous evaluation is made.
`grid_version`	`real`	`4`	Option to select grids computed by current version of grid or by version 3.5 CHEMGRID.
`grid_points`	`integer`	`1000000`	If a version preceding 4 is selected, then this parameter specifies how many grid points are contained in the grids.
`bump_filter`	`boolean`	`no`	Flag to screen each orientation for clashes with receptor prior to scoring and minimizing.
`bump_maximum`	`integer`	`0`	Maximum number of allowed bumps.
`contact_score`	`boolean`	`no`	Flag to perform contact scoring.
`contact_cutoff_distance`	`real`	`4.5`	Interaction distance for contact scoring. Please refer to `contact_cutoff_distance` in grid.
`contact_clash_overlap`	`real`	`0.75`	Amount of VDW overlap allowed. If two atoms approach closer than this fraction of the sum of their VDW radii, then the contact score is penalized. • `0` = Complete overlap allowed. • `1` = No overlap allowed.
`contact_clash_penalty`	`real`	`50`	Amount that contact score is penalized for each clash.

3E: Scoring (Continued)

Parameter	Type	Default	Description
chemical_score	boolean	no	Flag to perform chemical scoring. This feature is included for experimental purposes only. Parameterization is left to the user. Use at your own risk.
energy_score	boolean	no	Flag to perform energy scoring.
energy_cutoff_distance	real	10	Maximum distance between two atoms for their contribution to the energy score to be computed.
distance_dielectric	boolean	yes	Flag to make the dielectric depend linearly on the distance.
dielectric_factor	real	4	Coefficient of the dielectric. See Equation 1 on page 74 for context.
attractive_exponent	integer	6	Exponent of attractive Lennard-Jones term for VDW potential.
repulsive_exponent	integer	12	Exponent of repulsive Lennard-Jones term for VDW potential.
atom_model	string	u	Flag for how to model non-polar hydrogens. • u = United atom model. Hydrogens attached to carbons are assigned a zero VDW well-depth and the partial charge is transferred to the carbon. • a = All atom model. Hydrogens attached to carbons have regular VDW well-depth and partial charge is not modified.
vdw_scale	real	1	Scaling factor of vdw component of energy score.
electrostatic_scale	real	1	Scaling factor of electrostatic component of energy score.
rmsd_score	boolean	no	Flag to perform rmsd scoring, which is the rmsd of the molecules in the ligand_atom_file with respect to the molecule in the receptor_atom_file. Both molecules must have identical atoms.

3E: Scoring (Continued)

Parameter	Type	Default	Description
contact_maximum chemical_maximum energy_maximum rmsd_maximum	real	0	If orientations or ligands to be written, but not ranked, then they must pass this score cutoff to be written to file.
contact_size_penalty chemical_size_penalty energy_size_penalty	real	0	If ligands to be ranked, then they may be penalized by this value for each heavy atom. This helps correct for the uncomplexed score and reduce the size bias.
rmsd_override	real	0	If orientations to be written, but not ranked, then orientations with an RMSD (with respect to the input orientation) less than this value are written to file regardless of score.

3F: Minimization

Parameter	Type	Default	Description
Minimization allows on-the-fly adjustment of a molecule's orientation and/or conformation to improve its score. Though this calculation is CPU intensive, it improves the efficiency of the orientation or conformation search. The simplex algorithm uses random displacements to seed the search. Consequently, minimization results vary depending on the random seed selected by the user and the order of input molecules and site points. If high-quality results are required, then repeat the run several times with different random seeds.			
contact_minimize chemical_minimize energy_minimize rmsd_minimize	boolean	n	Flags to perform minimization with respect to each scoring function.
initial_translation	real	1	The maximum initial step size (in Angstroms) of the simplex in each cartesian dimension. The actual step size is a random value between zero and this value.
initial_rotation	real	0.1	The maximum initial step size (unitless) for each axis of rotation. The quaternion representation is used. A value of one corresponds to a 180 degree rotation.

3F: Minimization (Continued)

Parameter	Type	Default	Description
initial_torsion	real	10	The maximum initial step size (in degrees) for each torsion when flexible_ligand set. When several layers are minimized together, the outermost layer gets a full step and the step for each inner layer is divided by 2, 3, etc.
maximum_iterations	integer	100	Maximum number of simplex iterations for each cycle of minimization.
contact_convergence chemical_convergence energy_convergence rmsd_convergence	real	0.5	Convergence criteria with respect to each scoring function. At any iteration, if all vertices of the simplex have a score within this value of the best score, then minimization terminates.
maximum_cycles	integer	1	Maximum number of minimization cycles. After the first cycle, the initial step sizes are divided by 2, 3, etc.
cycle_convergence	real	1	The distance a minimization cycle must travel to trigger another cycle. The vector distance of the final simplex vertices is used, which is normalized with respect to the initial step size.
contact_termination chemical_termination energy_termination rmsd_termination	real	1	If the score is greater than this value after a cycle of minimization, then no more cycles are attempted. This parameter is useful to avoid prolonged minimization of unrecoverable orientations or conformations.

3G: Chemical Screening

Parameter	Type	Default	Description
Chemical screening allows for rapid filtering of a database based on chemical and 3D distance descriptors. Before any screening can be done, the database must be keyed using the `construct_screen` parameter at which time ligand flexibility should be considered. Then `screen_ligands` can be activated to do pharmacophore screening or similarity screening. Make sure to use the same distance bin dimensions. See Chemical Screen on page 44 for more discussion.			
`construct_screen`	`boolean`	`no`	Flag to construct distance chemical keys for each input molecule.
`screen_ligands`	`boolean`	`no`	Flag to screen input molecules based on distance chemical keys.
`pharmacophore_screen`	`boolean`	`no`	Flag to screen based on whether molecule keys include pharmacophore pattern.
`similarity_screen`	`boolean`	`no`	Flag to screen based on similarity of molecule keys to target keys.
`dissimilarity_maximum`	`float`	`0.25`	Maximum dissimilarity with target to write out molecule.
`distance_begin`	`float`	`2`	Smallest distance of interest in fingerprint.
`distance_end`	`float`	`17`	Largest distance of interest in fingerprint.
`distance_interval`	`float`	`0.5`	Distance resolution of fingerprint. The total number of distance keys cannot exceed 30, since a 32-bit key is currently used with the terminal bits being reserved for out-of-range values.

3H: Parallel Jobs

Parameter	Type	Default	Description
The parallel job parameters provide a convenient way to process a large database of molecules and distribute the workload over many computers. One dock job must be configured as a server job. Any number of other jobs running on separate machines are configured as client jobs. The server job reads the entire database and parses molecules out to the client jobs for processing. It is recommended that the server job be executed on the computer which stores the database. The server job and each client job requires its own input file. See Database Processing on page 42 for more discussion.			
`parallel_server`	`boolean`	`no`	Flag to identify this job as the server job.
`server_name`	`string`	`server`	The name used by the server job to communicate with client jobs.
`client_total`	`integer`	`5`	The number of client jobs at the disposal of this server job.
`client_name_1` `client_name_2` `client_name_3 ...`	`string`	`clientN`	If `parallel_server` set, then this list of names of client jobs is requested.
`client_name`	`string`	`client`	If `parallel_server` not set, then the name of this particular client job is requested. To be recognized by the server, this name must be included in the `client_name_`N list supplied to the server job.

3I: Multiple Ligands

Parameter	Type	Default	Description
The parameters in this category control the processing of a database of ligands. See Database Processing on page 42 for more discussion.			
`ligands_maximum`	`integer`	`1000`	The maximum number of ligands to read in from the input file. This INCLUDES skipped ligands.
`initial_skip`	`integer`	`0`	The initial number of ligands to skip. This is useful to position the reading stream to a particular point in the input file.
`interval_skip`	`integer`	`0`	The number of ligands to skip for every ligand processed. This is useful to perform a preliminary scan a database for timing purposes, or to coordinate the processing of a database over multiple machines.

3I: Multiple Ligands (Continued)

Parameter	Type	Default	Description
heavy_atoms_minimum	integer	0	Minimum number of heavy atoms. In general, set this to at least three (or nodes_minimum) when an orientation search is performed.
heavy_atoms_maximum	integer	100	Maximum number of heavy atoms.
rank_ligands	boolean	no	Flag to rank best molecules. • yes = Only top scorers written. • no =For each molecule, the best orientation (or set of orientations with write_orientations) is written.
rank_ligand_total	integer	100	Number of ranked ligands.
restart_interval	integer	100	Number of ligands to process between each time restart information is saved.

3J: Input

Parameter	Default	Description
The user must supply several kinds of input files. • **Coordinate** Contains molecules/site points (*.mol2, *.pdb, *.xpdb, *.ptr, *.sph). • **Grid** Contains precalculated score potentials (*.bmp, *.cnt, *.chm, *.nrg). • **Parameter** Contains VDW, chemical and flexibility parameters (*.defn, *.tbl). • **Control** Empty. Used to interact with a running job (*.quit, *.dump).		
ligand_atom_file	ligand.mol2	File containing ligand atom coordinates.
ligand_center_file	ligand_center.sph	File containing ligand site points (see Macromolecular Docking on page 46).
receptor_site_file	receptor_site.sph	File containing receptor site points.
score_grid_prefix	score_grid	Prefix for files containing precalculated score grids.
receptor_atom_file	receptor.mol2	File containing receptor atom coordinates. Used for continuum (no grids) scoring when gridded_score not set.
vdw_definition_file	$PATH/vdw.defn	File containing VDW labels and parameters. See vdw.defn on page 105.

3J: Input (Continued)

Parameter	Default	Description
chemical_definition_file	$PATH/chem.defn	File containing chemical labels and definitions. See chem.defn on page 106.
chemical_match_file	$PATH/chem_match.tbl	File containing chemical interaction table for use when chemical_match set. See chem_match.tbl on page 107.
chemical_score_file	$PATH/chem_score.tbl	File containing chemical interaction table for use when chemical_score set. See chem_score.tbl on page 108.
chemical_screen_file	$PATH/chem_screen.tbl	File containing chemical interaction table for use when similarity_screen set. See chem_screen.tbl on page 109.
flex_definition_file	$PATH/flex.defn	File containing flexible bond labels and definitions for use when flexible_ligand set. See flex.defn on page 110.
flex_drive_file	$PATH/flex_drive.tbl	File containing torsion parameters for a torsion_drive search. See flex_drive.tbl on page 111.
quit_file	*.quit	When multiple_ligands is set, this file may be created by the user at any time to signal the dock job to terminate execution. If rank_ligands is set, the best molecules are written to file and an up-to-date restart file is generated.
dump_file	*.dump	If rank_ligands is set, this file may be created by the user at any time to signal the dock job to write the best molecules to file and resume execution.

3K: Output

Parameter	Default	Description
DOCK writes up to three kinds of output files. • **Coordinate** Contains docked molecules (*.mol2, *.pdb, *.xpdb, *.ptr, *.sph). • **Info** Contains current ligand rankings. • **Restart** Contains restart information.		

`ligand_out_file`
`ligand_contact_file`
`ligand_chemical_file`
`ligand_energy_file`
`ligand_rmsd_file`
	`ligand_out.mol2`
	`ligand_cnt.mol2`
	`ligand_chm.mol2`
	`ligand_nrg.mol2`
	`ligand_rms.mol2`
		Files containing docked, minimized, rescored, or reformatted molecules for each type of scoring.

`info_file`	`*.info`	File containing summary information about the current `rank_ligands` list.
`restart_file`	`*.rst`	File containing detailed information about current `rank_ligands` list which is sufficient for restarting if the current job is prematurely terminated. See Command-line Arguments on page 53 for restart instructions.

Output

DOCK reports several kinds of information as output which is either written to screen in interactive mode or written to file in batch mode. Each section of the output is discussed below.

Heading

```
      UUUUUUUUUU     CCCCCCC     SSSSSSS    FF/   FFF/
      UU/    UU/  CC/     CC/  SS/     SS/  FF/ FFF/
     UU/    UU/  CC/     CC/  SS/          FFFFF/
    UU/    UU/  CC/     CC/  SS/          FF/ FF\
   UU/    UU/  CC/     CC/  SS/     SS/  FF/   FF\
  UUUUUUUUUU/    CCCCCCC/    SSSSSSS/   FF/     FF\


University of California at San Francisco, DOCK 4.0

__________________Job_Information_________________
launch_time                        Wed Mar 19 12:05:01 1997
host_name                          mycomputer
memory_limit                       126062592
working_directory                  /usr/people/user
user_name                          user
```

The heading lists general information about the current job. The `launch_time`, `host_name` and `working_directory` fields are useful to help the user archive the run conditions. The `memory_limit` field refers to the amount of RAM in bytes available to the job. This is can at most be equal to the physical memory of the computer, but may be less if limits have been imposed on the users shell. If the job requires more RAM than is available, then execution is stopped and an error message is reported.

Parameters

```
_________________General_Parameters_________________
flexible_ligand                 no
orient_ligand                   no
multiple_ligands                yes
score_ligand                    no
chemical_screen                 no
parallel_jobs                   no

_____________Multiple_Ligand_Parameters_____________
ligands_maximum                 <infinity>
initial_skip                    0
interval_skip                   0

_____________________File_Input_____________________
ligand_atom_file                old.mol2

_____________________File_Output____________________
ligand_none_file                new.pdb
```

All input parameters are echoed into the output. In fact, the output file may be used as an input file for another run. This is a useful technique to clean up an old, cluttered input file and make a tidy, well-organized input file.

This particular parameter list is nearly the shortest possible since most general parameters have been turned off. It is constructed automatically when DOCK is used to convert a molecule file format (see Molecule File Conversion on page 54).

If a parameter is not needed it is not reported in the output. The user can override this feature and force all parameters to be written out by using the `-v` flag (see Command-line Arguments on page 53).

Results

```
__________________________Docking_Results__________________________
Name           : ligand1
Description    : ****
Orientations tried                                    :        616
Orientations scored                                   :        500

Best energy score                                     :     -59.02
  Intramolecular energy score                         :      12.18
  Intermolecular energy score                         :     -71.20
RMSD of best energy scorer (A)                        :       2.39

Elapsed cpu time (sec)                                :     492.87
```

The docking results of each molecule are output as above. In terse mode (`-t` flag) all results are printed on a single line.

This particular output is from the flexible docking of a molecule.

Performance

```
______________________Docking_Performance______________________

Procedure timings                                 time (s)  percent
Read                                                  0.11        0
Screen                                                0.00        0
Orientation Search                                    0.48        0
Orientation Score                                    49.90       10
Conformation Anchor                                   0.05        0
Conformation Peripheral                             442.38       90
Other                                                 0.02        0
Total                                               492.94      100

Minimizer usage                          minimum   average  maximum
Calls per molecule                          4200      4200     4200
Score improvement per call                     0   2.7e+10  8.7e+13
Vertices per call                              1         3        6
Cycles per call                                1         2        5
Iterations per cycle                           1        15      100
```

This section of the output is generated if the user used the `-p` command-line flag (see Command-line Arguments on page 53). It is intended to help the user to understand the performance bottlenecks of a particular run and to make informed parameter choices. Since minimization can consume significant portions of cpu time, it is profiled in more detail.

This particular output is from a flexible docking run in which a majority of the cpu time is spent during the peripheral conformation search.

GRID

Author: Todd Ewing
Based on work by Elaine Meng and Brian Shoichet

Overview

GRID creates the grid files necessary for rapid score evaluation in DOCK. Three types of scoring are available: contact, chemical and energy scoring. The scoring grids are stored in files ending in *.cnt, *.chm, and *.nrg respectively. When docking, each scoring function is applied independent of the others and the results are written to separate output files.

GRID also computes a bump grid which identifies whether a ligand atom is in severe steric overlap with a receptor atom. The bump grid is identified with a *.bmp file extension. The file containing the bump grid also stores the size, position and grid spacing of all the grids.

The grid calculation must be performed prior to docking. The calculation can take up to 45 minutes, but needs to be done only once for each receptor site. Since DOCK can perform continuum scoring without a grid, the grid calculation is not always required. However, for most docking tasks, such as when multiple binding modes for a molecule or multiple molecules are considered, it will become more time efficient to precompute the scoring grids.

Bump Checking

Prior to scoring, each orientation can be processed with the bump filter to reject ones that penetrate deep into the receptor. Orientations that pass the bump filter are then scored and/or minimized with any of the available scoring functions.

The way a bump is detected has been changed from previous versions of DOCK. Instead of a strict distance cutoff, a bump is based on the sum of the van der Waals radii of the two interacting atoms. The user specifies what fraction of the sum is considered a bump. For example, the default definition of a bump is if any two atoms approach closer than 0.75 of the sum of their radii.

The way bump information is stored on a grid has also changed. Rather than storing a True/False value at every grid point, GRID stores an atomic radius which corresponds to smallest radius of ligand atom at the grid position which would still trigger a bump. During docking, for a given orientation, the position of each atom is checked with the bump grid. If the radius of the atom is greater than or equal to the radius stored in the bump grid, then the atom triggers a bump. To conserve disk space, the atom radius is multiplied by 10 and converted to a short unsigned integer, which takes up as much memory (1 byte) as the True/False value used in the previous implementation.

Energy scoring

The energy scoring component of DOCK is based on the implementation of force field scoring (See 20 in the References section of the Users Guide) in earlier versions of DOCK. Force field scores are approximate molecular mechanics interaction energies, consisting of van der Waals and electrostatic components:

$$E = \sum_{i=1}^{lig} \sum_{j=1}^{rec} \left(\frac{A_{ij}}{r_{ij}^{a}} - \frac{B_{ij}}{r_{ij}^{b}} + 332 \frac{q_i q_j}{D r_{ij}} \right)$$

Equation 1

where each term is a double sum over ligand atoms *i* and receptor atoms *j*, which include the quantities listed below.

E	Intermolecular interaction energy
r_{ij}	Distance between atoms *i* and *j*
A_{ij} and B_{ij}	van der Waals repulsion and attraction parameters
a and b	van der Waals repulsion and attraction exponents
q_i and q_j	Point charges on atoms *i* and *j*
D	Dielectric function
332	Factor to convert electrostatic energy to kcal/mol.

Generalization of the VDW component

The van der Waals component of the scoring function has been generalized to handle any combination of repulsive and attractive exponents (providing that $a > b$). The user may choose to "soften" the potential by using a 6-9 Lennard-Jones function. The general form of the van der Waals interaction between two identical atoms is presented in Equation 2,

$$E_{vdw} = C\varepsilon\left(\frac{2R}{r}\right)^a - D\varepsilon\left(\frac{2R}{r}\right)^b \qquad \text{Equation 2}$$

which includes the following additional quantities.

C and D	Coefficients to be determined
ε	Well depth of interaction energy
R	van der Waals radius of atoms

The coefficients *C* and *D* can be determined given the two following boundary conditions.

$$\frac{dE_{vdw}}{dr} = 0, \text{ at } r = 2R \qquad \text{Equation 3}$$

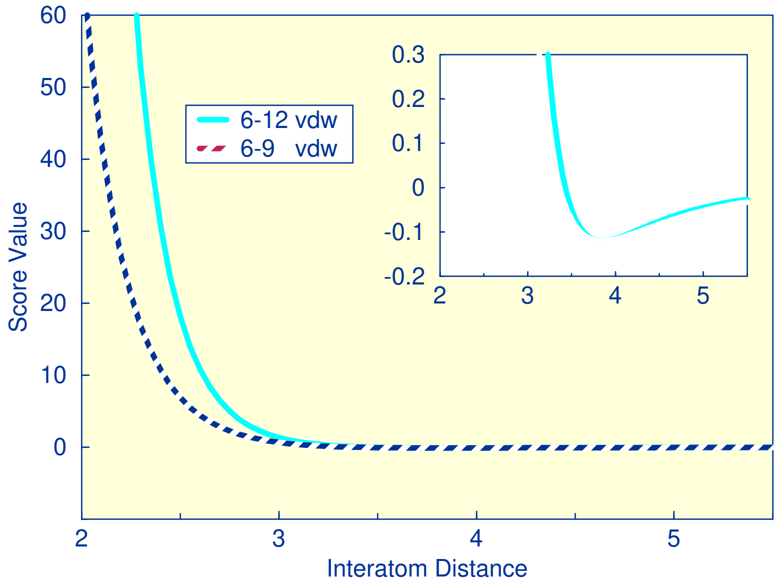
$$E_{vdw} = -\varepsilon, \text{ at } r = 2R \qquad \text{Equation 4}$$

Application of these boundary conditions to Equation 2 yields an expression of the van der Waals interaction with a generalized Lennard-Jones potential.

$$E_{vdw} = \varepsilon\left(\frac{b}{a-b}\right)\left(\frac{2R}{r}\right)^a - \varepsilon\left(\frac{a}{a-b}\right)\left(\frac{2R}{r}\right)^b \qquad \text{Equation 5}$$

The consequence of using a different exponent for the repulsive term are illustrated in Figure 1. Notice that the well position and depth are unchanged, but that the repulsive barrier has shrunk by about a quarter Angstrom.

Figure 1. Distance dependence of Lennard-Jones Function



The radius and well depth used to generate this figure (1.925 A and 0.12 kcal/mol) represent a united-atom model of CH_2.

Precomputing potentials on a grid

By inspection of Equation 1 and Equation 5, the repulsion and attraction parameters (A_{ij} and B_{ij}) for the interactions of identical atoms can be derived from the van der Waals radius, R, and the well depth, ε.

$$A_{ii} = \varepsilon\left(\frac{b}{a-b}\right)(2R)^a \quad \text{and} \quad B_{ii} = \varepsilon\left(\frac{a}{a-b}\right)(2R)^b \qquad \text{Equation 6}$$

In order to evaluate the interaction energy quickly, the van der Waals and electrostatic potentials are precomputed for the receptor and stored on a grid of points containing the docking site. Precomputing the van der Waals potential requires the use of a geometric mean approximation for the A and B terms, as shown in Equation 7.

$$A_{ij} = \sqrt{A_{ii}}\sqrt{A_{jj}} \quad \text{and} \quad B_{ij} = \sqrt{B_{ii}}\sqrt{B_{jj}} \qquad \text{Equation 7}$$

Using this approximation, Equation 1 can be rewritten:

$$E = \sum_{i=1}^{lig}\left(\sqrt{A_{ii}}\sum_{j=1}^{rec}\frac{\sqrt{A_{jj}}}{r_{ij}^a} - \sqrt{B_{ii}}\sum_{j=1}^{rec}\frac{\sqrt{B_{jj}}}{r_{ij}^b} + 332q_i\sum_{j=1}^{rec}\frac{q_j}{Dr_{ij}}\right) \qquad \text{Equation 8}$$

Three values are stored for every grid point k, each a sum over receptor atoms that are within a user-defined cutoff distance of the point:

$$A_{rec} = \sum_{j=1}^{rec}\frac{\sqrt{A_{jj}}}{r_{ij}^a}, \quad B_{rec} = \sum_{j=1}^{rec}\frac{\sqrt{B_{jj}}}{r_{ij}^b}, \text{ and } \quad Q_{rec} = 332\sum_{j=1}^{rec}\frac{q_j}{Dr_{ij}} \qquad \text{Equation 9}$$

These values, with trilinear interpolation, are multiplied by the appropriate ligand values to give the interaction energy. GRID calculates the grid values and stores them in files. The values are read in during a DOCK run and used for force field scoring. Substituting Equation 9 into Equation 8, the interaction energy is:

$$E = \sum_{i=1}^{lig}\left(\sqrt{A_{ii}}A_{rec} - \sqrt{B_{ii}}B_{rec} + q_iQ_{rec}\right) \qquad \text{Equation 10}$$

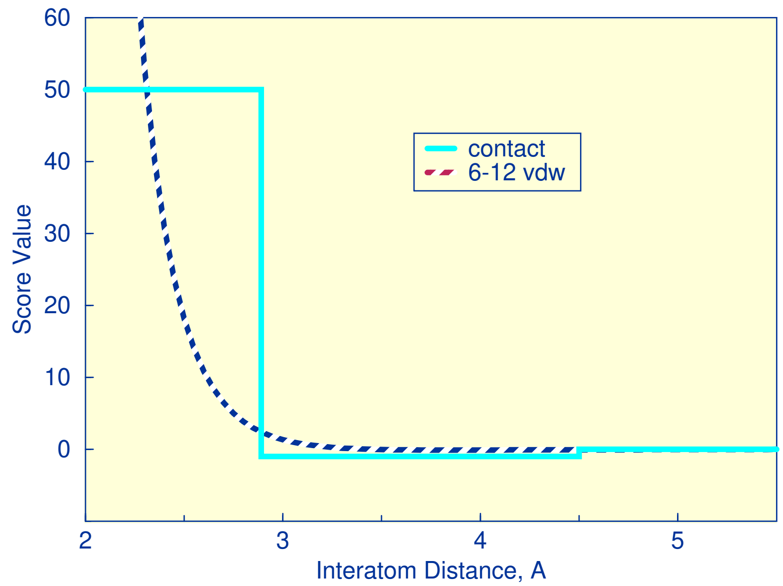
Atoms that fall outside the grid, if any, are given interaction energies of zero.

The user determines the location and dimensions of the grid box using the program SHOWBOX. It is not necessary for the whole receptor to be enclosed; only the regions where ligand atoms may be placed need to be included. The box merely delimits the space where grid points are located, and does not cause receptor atoms to be excluded from the calculation. Besides a direct specification of coordinates, there is an option to center the grid at a sphere cluster center of mass. Any combination of spacing and *x, y,* and *z* extents may be used.

Contact Scoring

Contact scoring in GRID incorporates the scoring performed with the DISTMAP program developed by Shoichet and Bodian. The score is a summation of the heavy atom contacts (every atom except hydrogen) between the ligand and receptor. A contact is defined as an approach of two atoms within some cutoff distance (usually 4.5 Angstroms). If the two atoms approach close enough to bump (as identified with the bump grid) then the interaction can be penalized by an amount specified by the user. The distance dependence of the contact score is represented in Figure 2.

Figure 2. Distance dependence of contact score function



Comparison with 6-12 LJ potential from Figure 1.

Contact distance = 4.5 A
Bump overlap = 0.75
Clash penalty = 50

Unlike the convention in DISTMAP, an attractive score in GRID is negative and a repulsive score is positive. This switch of sign was necessary to allow the same minimization protocol to be used for contact scoring as implemented for energy scoring.

Chemical Scoring

Chemical scoring is an optional scoring function which is based on energy scoring, but incorporates empirical concepts of molecular interaction. It includes the van der Waals and electrostatic terms from energy scoring, but the attractive portion of the van der Waals term has been modified. The attractive portion is scaled depending on the chemical nature of the interacting atoms.

The philosophy of chemical scoring is to give some control over scoring to the user without requiring any programming effort. Please see the section titled Chemical Score on page 40 for more discussion.

Chemical labeling of atoms

Atoms are identified with particular chemical labels depending on their atom type and the types of adjacent atoms. See chem.defn on page 106 for an example of chemical labels and definition rules. The file containing these rules is a text file and can be modified by the user. Consequently, the user can create any set of labels and definitions desired.

Scaling of interactions

The van der Waals interaction between two atoms is scaled based on the rules contained in a user-supplied file (see chem_score.tbl on page 108). The scaling factors provided with the distributed version of dock are merely suggestions. They are not derived from first principles, nor are they fitted to experimental data. They should be considered preliminary and can be adjusted depending on the interests of the user and the considerations of a particular test system.

Command-line Arguments

Table 4. GRID command-line argument summary

flag	optional argument	behavior
`-i`	`input_file`	INPUT FILE — Input parameters extracted from `input_file`, or `grid.in` if not specified
`-o`	`output_file`	OUTPUT FILE — Output written to `output_file`, or `grid.out` if not specified
`-s`		STANDARD INPUT — Input parameters entered interactively
`-t`		TERSE — Reduced output level
`-v`		VERBOSE — Increased output level

GRID may be executed in either interactive or batch mode, depending on whether output is written to a file. In interactive mode, the user is requested only for parameters relevant to the particular run and default values are provided. This mode is recommended for the initial construction of the input file. In batch mode, input parameters are read in from the input file and all output is written to the output file. This mode is recommended once an input file has been generated interactively.

Interactive mode

- `grid -i grid.in`

  When launched this way, GRID will extract all relevant parameters from `grid.in` (or any file supplied by the user). If additional parameters are needed (or if the `grid.in` file is non-existent or empty), GRID will request them one at a time from the user. Reasonable default values are presented. Any parameters supplied by the user will be automatically appended to the `grid.in` file. If the user would like to change any previously entered values, the user can edit in the `grid.in` file using a text editor.
- `grid -i`

  GRID will behave as above, but will assume the input file to be `grid.in`.
- `grid -s`

  GRID will run interactively, but will not check any input file for parameters and will not append any entered parameters to a file.

Batch mode

- `grid -i grid.in -o grid.out`

  GRID will run in batch mode, extracting all relevant parameters from `grid.in` (or any file supplied by the user) and will write out all output to `grid.out` (or any file supplied by the user). If any parameters are missing or incorrect, then execution will halt and an appropriate error message will be reported in `grid.out`.
- `grid -i -o`

  GRID will behave as above, but will assume the i/o files to be `grid.in` and `grid.out`.

Input Parameters

File Format

See Table 2. on page 55 for file format specifications.

Parameters

Like DOCK, GRID should be executed in interactive mode to construct an input file since not all parameters need to be specified for most runs. After all parameters have been entered, use CTRL-C to kill the process and resubmit it in batch mode using the same input file.

Table 5. GRID input parameters

5A: General Parameters

Parameter	Type	Default	Description
compute_grids	boolean	no	Flag to compute scoring grids.
grid_spacing	real	0.3	The distance between grid points along each axis.
output_molecule	boolean	no	Flag to write out the coordinates of the receptor into a new, cleaned-up file. Atoms are resorted to put all residue atoms together. Terminal SYBYL capping groups are merged with the terminal residues. This option can be useful to fix ligands also, so that hydrogens added by SYBYL are put with the proper residue.

5B: Scoring Parameters

Parameter	Type	Default	Description
contact_score	boolean	no	Flag to construct contact grid.
contact_cutoff_distance	real	4.5	Maximum distance between heavy atoms for the interaction to be counted as a contact.
chemical_score	boolean	no	Flag to construct chemical grid.
energy_score	boolean	no	Flag to perform energy scoring.

5B: Scoring Parameters (Continued)

Parameter	Type	Default	Description
energy_cutoff_distance	real	10	Maximum distance between two atoms for their contribution to the energy score to be computed.
atom_model	string	u	Flag for how to model of non-polar hydrogens. • u = United atom model. Hydrogens attached to carbons are assigned a zero VDW well-depth and the partial charge is transferred to the carbon. • a = All atom model. Hydrogens attached to carbons have regular VDW well-depth and partial charge is not modified.
attractive_exponent	integer	6	Exponent of attractive Lennard-Jones term for VDW potential. See Equation 1 on page 74 for context.
repulsive_exponent	integer	12	Exponent of repulsive Lennard-Jones term for VDW potential.
distance_dielectric	boolean	yes	Flag to make the dielectric depend linearly on the distance.
dielectric_factor	real	4	Coefficient of the dielectric.
bump_filter	boolean	no	Flag to screen each orientation for clashes with receptor prior to scoring and minimizing.
bump_overlap	real	0.75	Amount of VDW overlap allowed. If the probe atom and the receptor heavy atom approach closer than this fraction of the sum of their VDW radii, then the position is flagged as a bump. • 0 = Complete overlap allowed. • 1 = No overlap allowed.

5C: File Input

Parameter	Default	Description
receptor_file	receptor.mol2	Receptor coordinate file. Partial charges and atom types need to be present.
box_file	site_box.pdb	File containing showbox output file which specifies boundaries of grid.
vdw_definition_file	$PATH/vdw.defn	VDW parameter file.
chemical_definition_file	$PATH/chem.defn	Chemical label definition file for use when chemical_score set.

5D: File Output

Parameter	Default	Description
score_grid_prefix	grid	Core file name of grids (file extension will be appended automatically).
receptor_out_file	receptor_out.mol2	File for cleaned-up receptor when output_molecule set.

Output

The output of GRID contains several types of information. Like DOCK, it outputs general information about the current job and echoes the parameters selected from the input file. In addition, it reports information about the receptor and the grids.

Receptor information

```
Reading in coordinates of receptor.
Merging AMN 163 cap residue with THR1 1 residue.
Merging CXL 164 cap residue with ALA162 162 residue.
CHARGED RESIDUE THR1                    :    1.000
CHARGED RESIDUE ARG9                    :    1.000
CHARGED RESIDUE LYS51                   :    1.000
CHARGED RESIDUE ARG52                   :    1.000
CHARGED RESIDUE GLU56                   :   -1.000
CHARGED RESIDUE ARG57                   :    1.000
CHARGED RESIDUE HIP77                   :    1.000
CHARGED RESIDUE ASP78                   :   -1.000
CHARGED RESIDUE LYS127                  :    1.000
CHARGED RESIDUE ASP146                  :   -1.000
CHARGED RESIDUE HIP153                  :    1.000
CHARGED RESIDUE GLU156                  :   -1.000
CHARGED RESIDUE LYS161                  :    1.000
CHARGED RESIDUE ALA162                  :   -1.000

Total charge on UNNAMED                 :    4.000
```

This portion of the output lists any merged cap residues. The cap residues are introduced by SYBYL. Charged residues are also listed. If any residues have a non-integer charge, then either the charges were not properly loaded into the receptor input file, or some atoms are missing from the input file. These problems should be resolved before continuing with the grid calculation.

To display more information about parameters that GRID assigns to the receptor, use the `-v` option (see Command-line Arguments on page 79).

Grid Information

```
Reading in grid box information.
Box center of mass                       :   -1.315   36.145   21.153
Box dimensions                           :   27.525   26.519   26.686
Number of grid points per side [x y z]   :       93       90       90
Total number of grid points              :   753300

Generating scoring grids.
Percent of protein atoms processed       :        0
Percent of protein atoms processed       :       10
Percent of protein atoms processed       :       20
...etc...
```

Information about the grid geometry is reported. The total number of grid points can be adjusted by changing the input box dimensions or changing the grid spacing. Generally a value under 1 million is appropriate, but this depends on system resources.

The progress of the calculation is reported as the percent of protein atoms processed. This calculation usually takes up to 30 minutes depending on the grid geometry and the receptor size.

SPHGEN

Author: Irwin D. Kuntz
Modified by: Renee DesJarlais, Brian Shoichet

Overview

SPHGEN generates sets of overlapping spheres to describe the shape of a molecule or molecular surface (Kuntz *et al.,* 1982; DesJarlais *et al.,*1988). For receptors, a negative image of the surface invaginations is created; for a ligand, the program creates a positive image of the entire molecule. Spheres are constructed using the molecular surface described by Richards (1977) calculated with the program ms (Connolly, 1983a, 1983b). Each sphere touches the molecular surface at two points and has its radius along the surface normal of one of the points. For the receptor, each sphere center is "outside" the surface, and lies in the direction of a surface normal vector. For a ligand, each sphere center is "inside" the surface, and lies in the direction of a reversed surface normal vector. Spheres are calculated over the entire surface, producing approximately one sphere per surface point. This very dense representation is then filtered to keep only the largest sphere associated with each receptor surface atom. The filtered set is then clustered on the basis of radial overlap between the spheres using a single linkage algorithm. This creates a negative image of the receptor surface, where each invagination is characterized by a set of overlapping spheres. These sets, or "clusters," are sorted according to numbers of constituent spheres, and written out in order of descending size. The largest cluster is typically the ligand binding site of the receptor molecule. The program showsphere writes out sphere center coordinates in PDB format and may be helpful for visualizing the clusters.

Input

The input file names and parameters are read from a file called `INSPH`, which should not contain any blank lines:

parameter	format	example
msfil	`A80`	`2ptc.ms`
srftp	`A1`	`R`
dentag	`A1`	`X`
dotlim	`F`	`0.0`
radmax	`F`	`4.0`
radmin	`F`	`1.4`
outfl	`A80`	`2ptc.clus`

msfil is the name of the file containing the molecular surface calculated using the program MS and must include surface normals. SPHGEN expects the Fortran format

(A3, I5, X, A4, X, 2F8.3, F9.3, X, A3, 7X, 3F7.3).

This format is quite different from the QCPE molecular surface file format. For more details, see the documentation for reformatms and autoMS.

surftp indicates whether the spheres should lie "outside" the surface, as for a receptor (`R` or `r`), or "inside" the surface, as for a ligand (`L` or `l`).

dentag allows the user to specify that a subset of the surface points are to be used in calculating the spheres. This "density tag" may be set to the values `1, 4, 9`, or `0`, indicating that only points having 1, 4, 9, or 0, respectively, in column 42 of the molecular surface file will be used; alternatively, the value `X` or `x` indicates that all points will be used. It is recommended that `X` be used unless the system is particularly large (>75,000 surface points). It is most efficient to use a partial molecular surface (if it is known in advance which region is of interest) as the calculation time scales approximately with the square of the number of points.

dotlim is used to prevent the generation of large spheres whose points of surface contact are quite close together. Each pair of points *i* and *j* are examined as potential sphere-defining points. *dotlim* is the lower limit on the dot product of the vector from *i* to *j* and the vector from the sphere center to *j* for points defining a sphere. *dotlim* is typically set to 0.0, although possible values range from –1.0 to 1.0; negative values, however, may be useful for flat sites such as the major groove of B-form DNA.

radmax is the maximum sphere radius in Angstroms. Spheres with radii larger than *radmax* are discarded. This is important for the clustering done within SPHGEN, where clusters are defined as sets of overlapping spheres. Decreasing *radmax* decreases the cluster sizes by eliminating large "connector" spheres. In general, values from 4.0 to 5.0 Angstroms are used; values of 0.0 or less default to 5.0 Angstroms.

radmin is the minimum sphere radius in Angstroms. Spheres with radii smaller than *radmin* are discarded. This should be unnecessary because the molecular surface should not produce spheres of radius less than the probe radius. However, some versions of MS occasionally place surface points very close together. This can result in SPHGEN generating very small spheres which are not useful in characterizing the shape of the active site. It is generally advisable to keep spheres with radii equal to the probe radius (typically 1.4 or 1.5 Angstroms). Note that *radmin* can be set to 0.0 to allow the use of the "extra radius" surface developed by David Barry (unpublished results).

outfil is the name of the file to which the clustered spheres will be written.

Output

Some informative messages are written to a file called `OUTSPH`. This includes the parameters and files used in the calculation. The spheres themselves are written to *outfil*. They are arranged in clusters with the cluster having the largest number of spheres appearing first. The sphere cluster file consists of a header followed by a series of sphere clusters. The header is the line

DOCK 3.5 receptor_spheres

followed by a color table. The color table contains color names (format `A30`) each on a separate line. As SPHGEN produces no colors, the color table is simply absent. The sphere clusters themselves follow, each of which starts with the line

cluster   n   number of spheres in cluster   i

where `n` is the cluster number for that sphere cluster, and `i` is the number of spheres in that cluster. Next, all spheres in that cluster are listed in the format

(I5, 3F10.5, F8.3, I5, I2, I3)

where the values correspond to, respectively,

- The number of the atom with which surface point *i* (used to generate the sphere) is associated.
- The *x, y,* and *z* coordinates of the sphere center.
- The sphere radius.
- The number of the atom with which surface point *j* (second point used to generate the sphere) is associated.
- The critical cluster to which this sphere belongs.
- The sphere color. The color is simply an index into the color table that was specified in the header. So 1 corresponds to the first color in the header, 2 for the second, *etc.* 0 corresponds to unlabeled.

The clusters are listed in numerical order from largest cluster found to the smallest. At the end of the clusters is cluster number 0. This is not an actual sphere cluster, but a list of *all* of the spheres generated whose radii were larger than the minimum radius, *before* the filtering heuristics (*i.e.* allowing only one sphere per atom and using a maximum radius cutoff) and clustering were performed. Cluster 0 may be useful as a starting point for users who want to explore a wider range of possible clusters than is provided by the standard SPHGEN clustering routine. The program cluster takes the full sphere description as input, and allows the user to explore different sphere descriptions of the site. This is particularly useful for macromolecular docking; it is often inefficient to use spheres that fill the entire volume of the "ligand" macromolecule. In addition, only a portion of a cavity in the "receptor" macromolecule may be of interest for docking purposes. If the standard clustered output from SPHGEN provides a satisfactory description of the ligand molecule or receptor site, running cluster is not necessary.

The program creates three temporary files: `temp1.ms`, `temp2.sph`, and `temp3.atc`. These are used internally by SPHGEN, and are deleted upon completion of the computation.

Accessories

ADDPRH

Author: Andrew Leach

ADDPRH is used to add hydrogens atoms to proteins. Either PDB or AMBER atom names can be specified. Hydrogens are added in "favorable" geometries, but this does not take into account intramolecular hydrogen bonding. Hydrogens are also added to waters; to add some variety into the orientation of these hydrogens, a random orientation about the three euler angles is performed before adding the hydrogens.

AUTOMS

Author: Andy Good, Daniel Gschwend

AUTOMS is an extremely useful tool for setting up for sphgen. This script converts PDB files to QCPE MS input format, runs a QCPE MS surface calculation, converts the resulting surface to UCSF MS format with reformatms, creates a SYBYL dot file of the surface (if ms2dot is available), and prepares an `INSPH` file for running SPHGEN. The requirements are only two files: a receptor PDB file, and an `exclude.pdb` file. The `exclude.pdb` file is a subset of the receptor and contains the residues which should not be surfaced in the MS calculation. This file can be created in several ways. The easiest, if SYBYL is available, is to select all residues in the receptor within some radius of a known ligand, invert the selection, and write out the file in Brookhaven format as `exclude.pdb`. One alternative that does not require SYBYL is to use get_near_res to locate all receptor atoms or residues within a specified radius of a known ligand and invert the resulting pdb file using invertPDB.

Usage: `autoMS` *`receptor_PDB_file [surface_density] [probe_radius]`*

where *`surface_density`* and *`probe_radius`* are optional, defaulting to 3.0 dots/Å2 and 1.4Å, respectively. **Note**: before using the first time, the directory specification for the DOCK hierarchy must be updated inside the AUTOMS script.

CHARGE

Author: Daniel Gschwend

CHARGE is a simple NAWK script which provides residue composition for a protein, including number of charged residues and total charge. grid's output should agree with the value generated from this program.

Usage: `charge` *`pdbfile`*

CHEMPROP

Author: Renee DesJarlais

Overview

The interactive program CHEMPROP is designed to aid the user in examining the properties of a receptor in the vicinity of a small molecule that has been oriented using DOCK. The program takes as input a receptor coordinate file, a file containing the coordinates of one or more ligands, and several parameters, described below, depending on the option chosen. There are two options: electrostatics and hydrogen bonding.

In the electrostatic option, the electrostatic potential from the receptor is calculated at each ligand molecule atom center. The electrostatic potential at the position of ligand atom *j*, e_j, is calculated using equation 1, where q_i is the partial charge on the receptor atom *i*, *D* is the dielectric constant, and r_{ij} is the distance between atoms *i* and *j*:

$$e_j = \sum_{i=1}^{lig} \frac{q_i}{D r_{ij}} \qquad \text{Equation 11}$$

Only the receptor atoms contribute to the value of the electrostatic potential. The partial atomic charges used are those from the AMBER united-atom force field (Weiner, *et al.,* 1984). Only standard amino acid residues can be accommodated by this program. The receptor file must include the hydrogen atoms attached to nitrogens, hydroxyl oxygens, and sulfurs, and the lone pairs on the sulfurs. A new PDB-format coordinate file is written out for the ligand molecules, where the electrostatic potential is printed in the temperature factor column. The molecules can then be displayed using a molecular graphics package and colored according to electrostatic potential.

The hydrogen bond option helps the user identify places on the ligands where it might be appropriate to design in a group capable of hydrogen bonding to the receptor. Potential hydrogen bond positions are identified as any ligand atom within a user-specified distance of a receptor nitrogen or oxygen atom. This option was intended mainly for use with ligand heavy atom (nonhydrogen) coordinates only. Two files are output: a coordinate file, and a file listing the potential hydrogen bonds for each ligand in the input file. In the coordinate file, each potential hydrogen bond is written in PDB format as a residue with two atoms. One atom is located at the receptor nitrogen or oxygen and the other is located at the ligand atom. The residues are separated by TER cards. The residue and atom names depend on the protein atom. The residue is named `ACC` if the protein atom is a carbonyl oxygen, `DNR` if the protein atom is an amide or amine nitrogen, and `doa` if the protein atom is a hydroxyl oxygen or a histidine side chain nitrogen. The receptor and ligand atoms in an `ACC` residue are named `A` and `D`, respectively; in a `DNR` residue they are named `D` and `A`, respectively; and in a `DOA` residue they are named `E1` and `E2`, respectively. Viewing these possible hydrogen bonds with the receptor and ligand is useful for design purposes and assessing whether the angles are consistent with strong hydrogen bonding.

Usage

The input file names and parameters can be entered interactively. Interactive use is reasonable when fewer than about 30 ligands are to be examined. The user is prompted for the names of the receptor and ligand files and the type of calculation (electrostatic potential or hydrogen bond). If an electrostatic potential calculation is being performed, the user must select a constant or distance-dependent dielectric, a cutoff distance, and a name for the output file. If a hydrogen bond calculation is being performed, the user must enter a cutoff distance for hydrogen bonds and names for the output files. Finally, after performing one calculation, the user is given the option of performing another without exiting from the program.

CLUSTER

Authors: Brian Shoichet, Irwin D. Kuntz

Overview

CLUSTER allows greater flexibility in creating a sphere description of a site or molecule. It is particularly useful in macromolecular docking, and in general, when the original cluster file from sphgen does not adequately describe the site or molecule of interest.

The CLUSTER program is a more elaborate version of the cluster subroutine in SPHGEN (Kuntz *et al.*, 1982). A single-linkage clustering algorithm is applied, based on the radial overlap between spheres. Unlike SPHGEN, CLUSTER does not heuristically remove spheres; it can operate on the total set of possible spheres rather than just the largest sphere per surface atom. This complete description is contained in the cluster 0 from SPHGEN. User-defined criteria control the clustering process; clusters can be tailored to a certain size (number of spheres), a certain range of sphere radii, or a certain region of space. The program allows one to try different clustering parameters without rerunning SPHGEN.

Input

The program can be run either interactively or from a command file. The following is an explanation of a sample command file. These command files should not contain any blank lines.

parameter	**format**	**example**
clufil	`A80`	`2ptc.all`
nclus	`I`	`1`
maxrad	`F`	`5.0`
m2xrad	`F`	`5.0`
povlap	`F`	`10.0`
clusiz	`I`	`60`
minsiz	`I`	`20`
minflg	`I`	`1`
outfil	`A80`	`2ptc.all.rcl`
yn	`A1`	`y`
(if *yn* is `y` or `Y`):		
minrad	`F`	`1.3`
rincr	`F`	`0.2`
nearnb	`F`	`0.5`
nearad	`F`	`0.25`

clufil is the file containing the input spheres from SPHGEN.

nclus is the number of the cluster to recluster (there is generally more than one in the original SPHGEN cluster file). When the full set of spheres is being used, there is only one “cluster” and *nclus* should be set to 0.

maxrad is the primary maximum sphere radius for clustering, in Angstroms. Only spheres with radii less than or equal to *maxrad* can be used as linkers between groups of spheres, making them into a single larger cluster. Values from 2.5 to 5.0 Angstroms are generally most useful. *maxrad* also defines the end point for analytical clustering (see below); it is the final value of *rcut*.

m2xrad is the secondary maximum sphere radius for clustering, in Angstroms. This variable allows spheres with radii larger than *maxrad* to be included in clusters, but does not allow them to act as linkers. *m2xrad* must be equal to or greater than *maxrad*; smaller values default to *maxrad*. All spheres exceeding the *m2xrad* criterion will be discarded. *m2xrad* is typically set to *maxrad* for analytical clustering and 5.0 Angstroms otherwise.

povlap is the percent radial overlap between two spheres necessary to define a pair. If this variable is set to 0.0, spheres will be defined as overlapping when they intersect to any degree. The larger the value of *povlap*, the greater the overlap necessary to define the spheres as a pair for cluster purposes. Typical values range from 0.0 to 20.0.

clusiz is the maximum number of spheres allowed to be in a cluster. Growth of a cluster is frozen when this limit is reached; spheres that would otherwise be added are discarded. Limiting the cluster size leads to decreased coalescence and therefore greater numbers of clusters. Values of 50 to 75 are suggested.

minsiz is the minimum number of spheres a cluster must have to be included in the output. *minsiz* must be less than *clusiz*; values of 20 to 30 are suggested.

minflg is the minimum number of flagged spheres a cluster must have to be included in the output. Flagging is done by placing any non-blank characters following the information for the sphere(s) of interest in *clufil*. ***The flagging feature is no longer supported.***

yn indicates whether analytical clustering will be done. Analytical clustering refers to iteratively increasing the value of the primary maximum sphere radius. It is especially useful when the input sphere set is large (>1000, as when the full sphere description is being used). If *yn* equals `N` or `n`, analytical clustering will not be done, and no further input is read. Analytical clustering replaces *maxrad* with the variable *rcut*, which increases from *minrad* to *maxrad* in step sizes of *rincr*. Each value of *rcut* corresponds to a cycle of clustering. In this way, the user can quickly determine which parameters will yield a cluster of the desired size. For a set of 1000 spheres, a typical analytical run with averaging takes about 20 seconds on a Silicon Graphics Iris 4D/25 workstation. Most of the CPU time is spent in averaging the spheres; for this reason, run time scales approximately with the square of the number of spheres.

minrad is the starting value for *rcut* in analytical clustering.

rincr is the incremental increase in *rcut* per iteration in analytical clustering.

nearnb is the maximum distance between the centers of spheres that may be averaged into a composite sphere, for analytical clustering. This variable is used to simplify very large sets of spheres. When clusters are written out, only the sphere closest to the composite will be included. A value of 0.5 Angstroms is reasonable for sets of approximately 1000 spheres.

nearad is the maximum difference in magnitude between the radii of spheres that may be averaged into a composite sphere, for analytical clustering. A value of 0.25 Angstroms is reasonable for sets of approximately 1000 spheres.

Output

For analytical clustering, the output sphere cluster file consists of several sphere cluster files concatenated together. Each group begins with its own header (`dock 3.5 receptor_spheres`, *etc*.). The user must hand-edit this file to select the best group of clusters.

COLSPH

Author: Mike Connolly

COLSPH reads and writes a sphere cluster file, adding a color (label) table and sphere labels. The color is determined by evaluating either a Delphi map or a force-field grid at the sphere center, and comparing that value to the ranges specified in a user-defined *range file*, which defines ranges for a series of colors. The program prompts for the information it needs:

- scoring map type (2 = Delphi, 3 = force-field)
- the Delphi map file name or the force-field grid prefix
- range file name
- input sphere cluster file name
- output sphere cluster file name

The range file is the only one that needs explanation. Its format is as follows:

```
color_name range_min range_max
```

The spacing between the values does not matter. The `color_name` can be up to 30 characters in length. The `range_min` and `range_max` must be *real* numbers, and specify the lower and upper ends of the range respectively. There may be up to 100 colors and 200 range statements.

CONDENSE

Author: Daniel Gschwend

CONDENSE takes as input a residue list generated by get_near_res and compacts it into a format suitable for an MS `-i` file (for the UCSF version of MS - for the QCPE MS version, see autoMS). This functionality comes in handy when attempting to generate an MS surface for a portion of a receptor. The program get_near_res can be used to obtain a listing of all receptor atoms within a specified distance of a small molecule, *e.g.* a crystallographically observed ligand. CONDENSE will then reformat and compact the listing to less than 100 entries for compatibility with UCSF MS. This program supports command-line operation: type `condense -h` for details.

CONECT

Author: Elaine Meng

CONECT is an interactive program that appends CONECT records to a PDB file containing the coordinates of a single molecule. The user is prompted for the names of the input and output files. The presence of a bond is determined as follows: two atoms are bonded if the distance between them is less than or equal to the sum of their covalent bond radii plus a tolerance of 0.4 Angstroms.

CONVSYB

Author: Elaine Meng

The interactive program CONVSYB converts SYBYL MOL2 format files into a number of other formats useful for DOCK and some of the accessory programs. Output options include “extended PDB format,” DOCK 3.0 database format, and standard PDB format. CONVSYB assumes that each @<TRIPOS>MOLECULE record type indicator corresponds to a single covalently bonded structure, and uses the first 9 characters of the line following this record as the refcode for the structure. Multi-MOL2 files are handled correctly.

FDAT2PDB

Author: Daniel Gschwend

Converts a Cambridge Crystallographic Database FDAT file (as from a search with QUEST) to PDB format. Multiple structures in the same FDAT file are handled correctly.

Usage: `fdat2pdb` *`cambridge_prefix`* (an input file called *`cambridge_prefix`*`.fdat` is expected, and output file called *`cambridge_prefix`*`.pdb` will be created).

GET_NEAR_RES

Author: Daniel Gschwend

Given a ligand PDB file and a receptor PDB file, performs either of two functions:

- Writes a PDB file containing all atoms of all residues in the receptor which have their **C**α atoms within a specified distance of the ligand. A list file is also written which gives the closest Cα-to-ligand distance for each residue written.

- Writes a PDB file containing all atoms of all residues in the receptor which have **any** atom within a specified distance of the ligand. A list file is also written which gives the closest receptor-ligand distance for each residue written.

The list file output may be converted to an UCSF MS `-i` file with the condense program. This utility can also be used with the autoMS program using QCPE MS surfaces: to generate an `exclude.pdb` file from GET_NEAR_RES output, see invertPDB.

HBDATA

Author: Andy Good

This program automatically runs Goodford's grid program (version 11.01, distributed independently; Goodford, 1985) for a variety of probes, creating SYBYL contour files and files containing centers of favorable interaction energy for sphgen file creation. It also runs GRIN and will stop if GRIN encounters a problem with the PDB file. The program requires an input file called `HBIN` which should follow this example:

`3dfr`	name of parent protein (PDB file name - used for file naming)
`/bert/grid11.01`	root directory for GRID
`-12.0`	minimum *x* coordinate
`26.0`	minimum *y* coordinate
`4.0`	minimum *z* coordinate
`9.0`	maximum *x* coordinate
`48.0`	maximum *y* coordinate
`33.0`	maximum *z* coordinate
`1.0`	grid density - # of planes per Angstrom of GRID map
`-7.0`	highest permitted interaction energy included in site point creation
`1.0`	exclusion sphere to remove local low energy pts. near each local minimum
`2`	# of probes in probe list that will be used - first two are `N3+`, `O::`

Notes for use in conjunction with mol2sph: The GRID energy of each point is assigned to the charge field for reference purposes and atom types are set to dummy. Both of these fields must be corrected, the SPHGEN file converted into PDB format using showsphere, the resulting atom types modified and combined within SYBYL before they can be written to MOL2 format in preparation for mol2sph conversion.

IDTOSYB

Author: Elaine Meng

This program is designed to be used in combination with SYBYL to convert PDB files into SYBYL MOL2 format files. Do not use it by itself. The generated MOL2 files are incomplete. First run IDTOSYB on the PDB file, then read the generated MOL2 file into SYBYL to verify the atom types, add hydrogens, and assign partial charges. SYBYL does a less than satisfactory job of assigning atom types and bonds when reading in small molecule PDB files. IDTOSYB uses bond length data from the Cambridge Structural Database to assign atom types in a far superior fashion, although still not perfect. Hydrogens need not be present in the input, and multiple ligands in the same file are processed if separated by TER cards.

Usage: see pdb2syb, a front-end to IDTOSYB.

INVERTPDB

Author: Daniel Gschwend

A shell script to extract all atoms in a larger PDB file which are not in a smaller PDB file, where the latter is a subset of the former. For example, after acquiring a PDB file of all protein atoms or residues within a specified radius of a known ligand with get_near_res, the inverse atoms (*i.e.* those far away from the ligand) may be selected to generate an `exclude.pdb` file for use with pdb2ms or autoMS.

Usage: `invertPDB` *`larger_PDB_file smaller_PDB_file > inverted_PDB_file`*

MOL2SPH

Author: Andy Good

Takes SYBYL MOL2 format files and converts them into sphgen format files, allowing automatic coloring of the receptor. Currently, `N.3` atoms are classed as H-bond donors, `O.3` as H-bond acceptors, and `S.3` as hydrophobes, but these color setting can be changed to user preferences by simply hacking the file. The charge value can also be used to assign critical clusters, with the critical cluster number equal to the truncated value of the charge.

MS2DOT

Author: Andy Good

MS2DOT converts UCSF MS surface files into SYBYL dot files. This allows visualization in SYBYL of the surface created for use with sphgen. *Note: this program is not installed by default, as it requires* SYBYL *programming libraries for compilation.* To install this program you must explicitly compile it with `make ms2dot` in the `accessories` source directory, provided you have SYBYL libraries installed. Please read the `Makefile` in the `accessories` directory for further details.

OLDSCORE

Author: Elaine Meng

OLDSCORE is an interactive program that allows scoring of DOCK output according to the "old," exponential algorithm (as in version 1.1; DesJarlais *et al.,* 1988). The user is interactively asked for the names of the receptor file, the ligand file, and a file for output, plus the values of *concut*, *dmin*, and *discut*. There may be many ligands in the ligand file, as long as each ends with a TER card. Bad contacts are counted. There are slight differences in the scores directly from DOCK 1.1 and the scores reported by this program, because the coordinates of ligand orientations are rounded to three decimal places when written out.

PDB2MS

Author: Andy Good

This program is used to create input files for QCPE MS from standard PDB files. The receptor file must be called `fort.1`; the residues of the receptor which should be included in the surface calculation but not have a surface drawn should be placed in a file called `fort.2`; output is written to `fort.3`. This program is part of the automated process of autoMS.

PDB2SYB

Author: Daniel Gschwend

This program is a front-end to idtosyb. Please read the description of IDTOSYB before using this program.

Usage: `pdb2syb` *`PDB_file`*

Note: before using the first time, update the pathnames inside the script for location of IDTOSYB.

PDBRENUM

Author: Daniel Gschwend

PDBRENUM is a versatile residue and atom renumbering utility. It provides many features with respect to where to start numbering, how to treat separate molecules, and also supports individual treatment of water molecules, including their separation by TER cards for easier visualization. PDBRENUM also attempts to update CONECT records when atoms are renumbered. This program supports command-line operation: type `pdbrenum -h` for details.

PDBTOSPH

Author: Elaine Meng

PDBTOSPH makes a sphere cluster file out of the non-hydrogen ATOM records (not HETATM) in a PDB file.

PTRENTRY

Author: Todd Ewing

PTRENTRY is a self-running nawk script which extracts selected molecule entries from a PTR format file and displays the data for each molecule in a readable format. A single molecule, or a range of molecules, may be selected.

- To extract the first molecule, type: `ptrentry file.ptr 1`
- To extract the first 10 molecules, type: `ptrentry file.ptr 1 10`

PTRFIELD

Author: Todd Ewing

PTRFIELD is a self-running nawk script which extracts selected fields from a PTR format file. It is useful to shuffle the field order in a file to make sorting sorting easy. This would be accomplished as follows:

- `ptrfield file.ptr NRG FILE FPOS END | sort +1n > file_sort.ptr`

QCPE_MS

Author: Mike Connolly

sphgen requires that a molecular surface be calculated and that it be converted to the UCSF MS format using reformatms. QCPE_MS is distributed only as a tool for the autoMS script. You should contact QCPE directly for obtaining the complete software package for MS (#429) - see Sources on page 112 for contact information.

REFORMATMS

Author: Renee DesJarlais

REFORMATMS converts an MS file of QCPE format to an MS file of the format read by sphgen. REFORMATMS requires a Brookhaven Protein Data Bank coordinate file as well as the QCPE MS file as input. The program is interactive.

The QCPE MS file must be in the long format: The Fortran format for this file and the information contained in it are listed below:

```
(3I5, I2, 3F9.3, 4F7.3, I2)
```

n1, n2, n3, shape, x, y, z, area, xn, yn, zn, buried

n1	atom number of atom that surface point is on or closest to
n2	other atom that probe touches (0 for convex)
n3	third atom that probe touches, *n3*>*n2* (0 for convex and saddle)
shape	1: convex, 2: saddle, 3: concave
x, y, z	coordinates of surface point
area	solvent-accessible area
xn, yn, zn	components of the unit vector normal
buried	0: exposed, 1: buried, blank: not determined

The lines must be sorted by *n1*.

The MS format that SPHGEN reads and the information that it contains are listed below:

(A3, I5, 2X, A3, 3(F8.3, X), X, A3, 4F7.3)

resnm, nres, atnm, x, y, z, srftag, area, xn, yn, zn

resnm	residue name of the closest atom, or the atom itself (if *srftag* = `A`)
nres	residue number of the closest atom, or the atom itself (if *srftag* = `A`)
atnm	name of the closest atom, or the atom itself (if *srftag* = `A`)
x, y, z	coordinates of the point or atom
srftag	`A`: atom, `SR0`: reentrant point, `SS0`: saddle point, `SC0`: convex point
area	solvent-accessible area (blank if *srftag* = `A`)
xn, yn, zn	components of the unit vector normal (blank if *srftag* = `A`)

RMSD

Author: Daniel Gschwend

Calculates root mean squared deviation in Angstroms per atom for two PDB files. Treatment of hydrogens is optional. Also, one may compare two files side by side, or use the first molecule in the first file as a reference for all molecules in the second file. All atoms must be in identical order and have the same atom names for both input files. This program supports command-line operation: type `rmsd -h` for details.

SDF2MOL2 & SYBDB

Author: Daniel Gschwend

As an interface to the commonly used databases from Molecular Design Limited, we are providing two conversion schemes to convert SD files (such as from ISIS) to SYBYL MOL2 format files.

The conversion scheme described here was developed to be easy to use and accurate in its atom-typing and partial charge computation. Although still an area of active development, it has to date been tested visually on several thousand compounds.

This scheme consists of two programs run sequentially, SDF2MOL2 is written in Fortran and SYBDB in SYBYL's programming language, SPL. Taken together, one can convert an MDL SDF-format database into a SYBYL MOL2 format database which has appropriate SYBYL atom types assigned, hydrogens added, and partial charges computed. To find out more about the conversion process, including how to use it, please consult the `00README` file in the directory `./source/database/sdf2mol2` under the DOCK root.

Note: The second phase conversion requires SYBYL for hydrogen addition and charge computation. The former program, SDF2MOL2, may still be of some use to users who do not have SYBYL but have molecular modeling packages than can read MOL2 format (e.g. INSIGHT). Hydrogen addition, substructure removal, and charge computation must then be performed within the context of your own modeling package.

SHOWBOX

Author: Elaine Meng

SHOWBOX is an interactive program that allows visualization of the location and size of the grids that will be calculated by the program grid, using any graphics program that can display PDB format. The user is asked whether the box should be automatically constructed to enclose all of the spheres in a cluster. If so, the user must also enter a value for how closely the box faces may approach a sphere center (how large a "cushion" of space is desired) and the sphere cluster filename and number. If not, the user is asked whether the box will be centered on manually entered coordinates or a sphere cluster center of mass. Depending on the response, the coordinates of the center or the sphere cluster filename and number are requested. Finally, the user must enter the desired box dimensions (if not automatic) and a name for the output PDB-format box file.

SHOWSPHERE

Authors: Stuart Oatley, Elaine Meng, Daniel Gschwend

SHOWSPHERE is an interactive program; it produces a PDB-format file of sphere centers and an MS-like file of sphere surfaces, given the sphere cluster file and cluster number. The surface file is not in QCPE MS format (see the documentation on reformatms on page 95 for further details); generation is optional. The user may specify one cluster or "all," and multiple output files will be generated, with the cluster number appended to the end of the name of each file. The input cluster file is created using sphgen. SHOWSPHERE requests the name of the sphere cluster file, the number of the cluster of interest, and names for the output files. Information is sent to the screen while the spheres are being read in, and while the surface points are being calculated.

SPLITMOL

Author: Daniel Gschwend

A full-featured structure file splitting utility which accepts either PDB or SYBYL MOL2 format files. A range of molecules can be extracted from the input, and a user-specifiable number of molecules is put into each file created. This program supports command-line operation: type `splitmol -h` for details.

Molecule File Formats

Dock programs read and write in several coordinate formats.

- SYBYL MOL2 format
- PDB format
- PTR format
- SPH format

SYBYL MOL2 format

This format is used for general molecule input and output of DOCK. Although previous versions of DOCK supported an extended PDB format to store molecule information, the current version now uses MOL2 as the primary molecule format. This format has the advantage of storing all the necessary information for atom features, position, and connectivity. It is also a standardized format that other modeling programs can read.

Specification

Please refer to sybyl documentation for format specifications.

Of the many record types in a MOL2 file, DOCK recognizes the following: MOLECULE, ATOM, BOND, SUBSTRUCTURE and SET. In the MOLECULE record, DOCK utilizes information about the molecule name and number of atoms, bonds, substructures and sets. In the ATOM record DOCK utilizes information about the atom names, types, coordinates, and partial charges. In the BOND record, DOCK utilizes the atom identifiers for the bond. In the SUBSTRUCTURE record, DOCK records the fields, but does not utilize them. The SET records are entirely optional. They are used only in special circumstances, like when ligand flexibility is considered..

Example

This example file illustrates all the elements of the MOL2 file read and written by dock. It includes optional SET records which are used by the ligand flexibility routines.

```
@<TRIPOS>MOLECULE
Histidine
   20    20     1     0     2
SMALL
GASTEIGER
****
Histidine with Main Chain as Rigid Anchor
```

```
@<TRIPOS>ATOM
     1 N1        -1.0947    0.5371    1.7186 N.4       1 <1>        0.2252
     2 C2        -0.9885    0.9170    0.2765 C.3       1 <1>        0.0213
     3 C3        -0.2043   -0.1565   -0.4766 C.3       1 <1>        0.0354
     4 C4        -2.3725    1.0376   -0.3154 C.2       1 <1>        0.0897
     5 O5        -2.7546    2.1336   -0.8057 O.co2     1 <1>       -0.5442
     6 C6         1.1797   -0.2771    0.1153 C.2       1 <1>        0.0328
     7 N7         2.2791    0.4215   -0.2757 N.pl3     1 <1>       -0.3074
     8 C8         1.5387   -1.0911    1.1173 C.2       1 <1>        0.0462
     9 C9         3.3256    0.0285    0.4990 C.2       1 <1>        0.0853
    10 N10        2.9039   -0.8872    1.3511 N.2       1 <1>       -0.2465
    11 H11       -1.6259    1.2643    2.2287 H         1 <1>        0.2001
    12 O12       -3.1452    0.0423   -0.3188 O.co2     1 <1>       -0.5442
    13 H13       -0.1461    0.4545    2.1242 H         1 <1>        0.2001
    14 H14       -0.4726    1.8710    0.1898 H         1 <1>        0.0918
    15 H15       -0.7202   -1.1105   -0.3899 H         1 <1>        0.0385
    16 H16       -0.1270    0.1200   -1.5261 H         1 <1>        0.0385
    17 H17        2.3126    1.1114   -1.0125 H         1 <1>        0.1528
    18 H18        0.8943   -1.7774    1.6466 H         1 <1>        0.0845
    19 H19        4.3357    0.4040    0.4286 H         1 <1>        0.1000
    20 H20       -1.5855   -0.3703    1.8010 H         1 <1>        0.2001

@<TRIPOS>BOND
     1    1    2 1
     2    2    3 1
     3    2    4 1
     4    3    6 1
     5    4    5 ar
     6    6    7 1
     7    6    8 2
     8    7    9 1
     9    8   10 1
    10    9   10 2
    11    1   11 1
    12    4   12 ar
    13    1   13 1
    14    2   14 1
    15    3   15 1
    16    3   16 1
    17    7   17 1
    18    8   18 1
    19    9   19 1
    20    1   20 1

@<TRIPOS>SUBSTRUCTURE
     1 ****        1 TEMP              0 ****  ****    0 ROOT

@<TRIPOS>SET
ANCHOR          STATIC     ATOMS     <user>    **** Anchor Atom Set
1 2
RIGID           STATIC     BONDS     <user>    **** Rigid Bond Set
2 1 3
```

PDB format

This format should be used only for display purposes. Since it does not contain fields for atom types or partial charges, vital information will be lost if dock output is routinely stored in this format. It is recommended that all dock output be stored in either SYBYL MOL2 format or PTR format. If you site does not have access to a method to view MOL2 format, just convert files to PDB when vewing is necessary. See Molecule File Conversion on page 54 for instructions.

PTR format

Specification

PTR (or pointer) format is a compact representation for molecules which does not actually contain coordinates. Instead, it contains the transformations to the coordinates and "points" back to a source file where the untransformed coordinates reside.

All information about a molecule is packed into one line. Each item of data is identified by a field name. Some fields are required, like the molecule source information. Some are optional, like the coordinate transformation information. All others are considered comments and are ignored, like the molecule name and score information.

Table 6. PTR Format Data Fields

6A: Required PTR Fields

These fields must be present in every PTR record.

Field	Type	Description
`<FILE>`	`String`	Directory path and name of file which contains molecule.
`<FPOS>`	`Integer`	Byte position in file of molecule.
`<END>`		End of record flag.

6B: Optional PTR Fields1 of 2

If these fields are present, they will be read and used to update the molecule. The transformation fields will modify the position or conformation of the molecule.

Field	Type	Description
`<TRANS>`	`Real[3]`	XYZ translation vector in Angstroms.
`<ROT>`	`Real[3]`	Quaternion rotation vector (unitless, range 0-1).
`<TORS>`	`Integer`	Number of rotatable torsions.
`<REFL>`	`Integer`	Flag for chiral reflection (`0` or `1`).

6B: Optional PTR Fields2 of 2

If these fields are present, they will be read and used to update the molecule. The transformation fields will modify the position or conformation of the molecule.

Field	Type	Description
`<T1>,<T2>, ...`	`Integer, Real`	Bond identifier and torsion angle.
`<KEY>`[c]	`Integer`	Number of chemical keys.
`<KFOLD>`[c]	`Integer`	Flag for folded chemical keys.
`<KI1>,<KI2>, ...`[c]	`Integer`	Chemical key label and distance fingerprint

c. This field is only read during a chemical screening run.

6C: Comment PTR Fields

These fields are generated for output only. They are completely ignored during input.

Field	Type	Description
`<ID>`	`Integer`	Line position in ptr file.
`<SRC_ID>`	`Integer`	Line position in input ptr file.
`<NAME>`	`String`	Molecule name.
`<DESCR>`	`String`	Molecule description.
`<BMP>`	`Integer`	Number of bumps.
`<CNT>`	`Real`	Contact score.
`<CHM>`	`Real`	Chemical score.
`<NRG>`	`Real`	Total energy score.
`<INTRA>`	`Real`	Intramolecular component of score.
`<INTER>`	`Real`	Intermolecular component of score.
`<VDW>`	`Real`	VDW component of score.
`<ELE>`	`Real`	Electrostatic component of score.
`<RMSD>`	`Real`	RMS deviation of current orientation from input.

Example

Database Entry

```
<ID> 10 <NAME> CAMYLOFIN_C00000105 <DESCR> **** <FILE> /marco/db/
mol2.95.1/cmc/cmc.2.mol2 <FPOS> 46204 <END>
```

Flexible Docking Output

```
<ID> 5 <SRCID> 5 <NAME> DANAZOL_C00002993 <DESCR> **** <FILE> db/
db3.mol2 <FPOS> 14144 <TRANS> 5.00991 29.9234 16.2096 <ROT> 0.270641
1.05337 0.0158567 <TORS> 2 <TANCHOR> 1 <T1> 20 153.624 <T2> 21 -28.0673
<REFL> 0 <BMP> 2 <NRG> -32.43 <INTRA> -0.48 <INTER> -31.95 <VDW> -32.53
<ELECTRO> 0.09 <RMSD> 34.95 <END>
```

Chemical Screen Database Entry

```
<ID> 1 <NAME> DAZOQUINAST_C00005118 <DESCR> **** <FILE> /marco/db/
mol2.95.1/cmc/cmc.1.mol2 <FPOS> 9689688 <KEY> 5 <KFOLD> 1 <KI1> 0 <KJ1>
10 2d7 <KJ2> 35 3fff <KJ3> 0 0 <KJ4> 20 f65 <KJ5> 0 0 <KI2> 1 <KJ2> 21
fd <KJ3> 0 0 <KJ4> 28 ab7e <KJ5> 0 0 <KI3> 2 <KJ3> 0 0 <KJ4> 0 0 <KJ5>
0 0 <KI4> 3 <KJ4> 6 c1a <KJ5> 0 0 <KI5> 4 <KJ5> 0 0 <END>
```

Usage

Check out the ptrentry and ptrfield utilities to help manipulate PTR files.

The <FILE> field can contain the relative path or absolute path of a filename. If you use the absolute path, then the ptr file can be used in any other directory. Otherwise, the ptr file is only useful in the idrectory in which it was originally made.

SPH format

Please refer to sphgen on page 84 for a description of the file format.

Parameter Files

Introduction

The parameter files contain atom and bond data needed during DOCK calculations. The definition (*.defn) files contain atom and bond labeling data. The table (*.tbl) files contain additional data for chemical interactions and flexible bond torsion positions. They may be edited by the user.

Atom definition rules

The definition files use a consistent atom labeling convention for which an atom in virtually any chemical environment can be identified. The specification of adjacent atoms is nested using the elements listed in Table 7. Some sample definitions are provided in Table 8.

- Each element must be separated by a space.
- If more than one adjacent atom is specified, then ALL must be present (i.e. a boolean AND for rules within a line).
- If a label can have multiple definition lines, then any ONE of them must be satisfied for inclusion (i.e. a boolean OR for rules on different lines).

Table 7. Atom definition elements

element	function
atom type	Specifies partial or complete atom type. A partial specification is more general (i.e. "C" versus "C.3"). An asterisk (*) specifies ANY atom type.
`( )`	Specifies atoms that must be bonded to parent atom.
`[ ]`	Specifies atoms that must NOT be bonded to parent atom.
integer	Specifies the number of an atom that must be bonded.

Table 8. Example definitions

example	explanation
`C.2 ( 2 O.co2 )`	A carboxylate carbon.
`.3 [ 3 H ]`	Any sp^3 hybridized atom that is not attached to three hydrogens.
`C. [ O. ] [ N. [ 2 O.2 ] [ 2 C. ] ]`	Any carbon not attached to an oxygen or a nitrogen (unless the nitrogen is a nitro or tertiary nitrogen).

vdw.defn

This file contains atom labels and definitions for van der Waals atom typing.

- The following data types are associated with each atom: VDW radius, VDW well-depth, flag for heavy atom, number of attached atoms.
- Some labels are used only for the united-atom model, some for only the all-atom model, and some for either.
- A label may have multiple definitions.

For an explanation of the `definition` field, see Atom definition rules on page 104.

Table 9. Sample entries from vdw.defn

```
name            Carbon_sp/sp2
atom_model      either
radius          1.850
well_depth      0.120
heavy_flag      1
valence         4

definition      C
_____________________________________
name            Carbon_All_sp3
atom_model      all
radius          1.800
well_depth      0.060
heavy_flag      1
valence         4

definition      C.3
_____________________________________
name            Carbon_United_CH3
atom_model      united
radius          2.000
well_depth      0.150
heavy_flag      1
valence         4

definition      C. ( 3 H )
```

chem.defn

This file contains labels and definitions for chemical labeling.

- Nothing in addition to a label is assigned to an atom.
- A label may have multiple definition lines.

For an explanation of the `definition` field, see Atom definition rules on page 104.

Table 10. Entries from chem.defn

```
name            hydrophobic

definition      C. [ O. ] [ N. [ 2 O.2 ] [ 2 C. ] ] ( * )
definition      N.pl3 ( 3 C. )
definition      Cl ( C. )
definition      Br ( C. )
definition      I ( C. )
definition      C.3 [ * ]
---------------------------------------------------------------
name            donor

definition      N. ( H )
definition      N.4 [ * ]
---------------------------------------------------------------
name            acceptor

definition      O. [ H ] [ N. ] ( * )
definition      O.3 ( 1 * ) [ N. ]
definition      O.co2 ( C.2 ( O.co2 ) )
definition      N. [ H ] [ N. ] [ O. ] [ 3 . ] ( * )
definition      O.2 [ * ]
---------------------------------------------------------------
name            polar

definition      O.3 ( H )
definition      F [ * ]
```

chem_match.tbl

This file contains the interaction matrix for which chemical labels can form an interaction in matching.

- The labels must be identical to labels in chem.defn.
- The `table` flag indicates the beginning of the interaction table.
- Compatible labels are identified with a one, otherwise a zero.

For an explanation of the `definition` field, see Atom definition rules on page 104.

Table 11. Example of chem_match.tbl

```
label   null
label   hydrophobic
label   donor
label   acceptor
label   polar

table
1
1       1
1       0       1
1       0       0       1
1       0       1       1       1
```

chem_score.tbl

This file contains the interaction matrix for how an chemical interaction score between chemical labels is scaled.

- The labels must be identical to labels in chem.defn.
- The `table` flag indicates the beginning of the interaction table.
- Each element is a scaling factor.

For an explanation of the `definition` field, see Atom definition rules on page 104.

Table 12. Example of chem_score.tbl

```
label   null
label   hydrophobic
label   donor
label   acceptor
label   polar

table
0
0       1
0       -1      -1
0       -1      1       -1
0       -1      1       1       1
```

chem_screen.tbl

This file contains the interaction matrix for how to scale the contribution of each chemical key when computing the overall similarity.

- The labels must be identical to labels in chem.defn.
- The `table` flag indicates the beginning of the interaction table.
- Each element is a scaling factor.

For an explanation of the `definition` field, see Atom definition rules on page 104.

Table 13. Example of chem_screen.tbl

```
label   null
label   hydrophobic
label   donor
label   acceptor
label   polar

table
0
0       1
0       1       1
0       1       1       1
0       1       1       1       1
```

flex.defn

This file contains labels and definitions for flexible bond identification.

- The `drive_id` field corresponds to a torsion type in the flex_drive.tbl file.
- The `minimize` field is a flag for whether the bond may be minimized.
- Two definition lines must be present. Each definition corresponds to an atom at either end of the bond.

For an explanation of the `definition` field, see Atom definition rules on page 104.

Table 14. Selected entries from flex.defn

```
name            sp3-sp3
drive_id        3
minimize        1

definition      .3 [ 3 H ] [ 3 O.co2 ]
definition      .3 [ 3 H ] [ 3 O.co2 ]
----------------------------------------
name            sp3-sp2
drive_id        4
minimize        1

definition      .3 [ 3 H ] [ 3 O.co2 ]
definition      .2 [ 2 H ] [ 2 O.co2 ]
----------------------------------------
name            sp2-sp2
drive_id        2
minimize        0

definition      .2 [ 2 H ] [ 2 O.co2 ]
definition      .2 [ 2 H ] [ 2 O.co2 ]
```

flex_drive.tbl

This file contains torsion positions assigned to each rotatable bond when the `torsion_drive` parameter is used in DOCK.

- The `drive_id` field corresponds to each torsion type.
- The `positions` field specifies the number of torsion angles to sample.
- The `torsions` field specifies the angles that are sampled.

Table 15. Selected entries from flex_drive.defn

```
drive_id        2
positions       2
torsions        0 180
-----------------------------------------
drive_id        3
positions       3
torsions        -60 60 180
-----------------------------------------
drive_id        4
positions       4
torsions        -90 0 90 180
-----------------------------------------
drive_id        6
positions       6
torsions        -150 -90 -30 30 90 150
```

Sources

compiled by C.Corwin

Available Chemicals Directory

MDL Information Systems, Inc.
14600 Catalina Street
San Leandro, CA 94577
phone (510) 895-1313
fax (510) 352-2870

Brookhaven Protein Data Bank

Protein Data Bank
Chemistry Department, Building 555
Brookhaven National Laboratory
Upton, NY 11973
phone (516) 282-3629
fax (516) 282-5751
e-mail pdb@bnl.gov
gopher://pdb.pdb.bnl.gov/11

Cambridge Crystallographic Database

For industrial users:
The Cambridge Crystallographic Data Centre
12 Union Road
Cambridge
CB2 1EZ
U.K.
phone +44 223 336408
fax +44 223 336033

For US academic users:
Dr. William L. Duax
Medical Foundation of Buffalo
Research Laboratories
73 High Street
Buffalo, NY 14203-1196
phone (716) 856-9600

fax (716) 852-4846
e-mail WPPDS%mfb@edu.buffalo.cc.ubvms

Cambridge information

http://csdvx2.ccdc.cam.ac.uk

DelPhi

Biosym Technologies
10065 Barnes Canyon Road
San Diego, CA 92121
phone (619) 458-9990

Goodford's GRID

Henrietta Elton
Molecular Discovery Limited
West Way House, Elms Parade
Oxford OX2 9LL
ENGLAND
Phone +44-993-830385
Fax +44-993-830966

ISIS

MDL Information Systems, Inc.
14600 Catalina Street
San Leandro, CA 94577
phone (510) 895-1313
fax (510) 352-2870

MS

see Quantum Chemistry Program Exchange (QCPE)

Quantum Chemistry Program Exchange (QCPE)

Creative Arts Bldg. 181
Indiana University
Bloomington, IN 47405
phone (812) 855-5539
fax (812) 855-4784
e-mail qcpe@ucs.indiana.edu
gopher://gopher.gdb.org/1ftp%3aqcpe6.chem.indiana.edu%40/

SYBYL

Tripos Associates
1699 S. Hanley Road, Suite 303
St. Louis, MO 63144-2913
phone (800) 323-2960

The UCSF MIDASPLUS molecular display program

MIDAS Software Distribution
Computer Graphics Laboratory
School of Pharmacy
University of California
San Francisco, CA 94143-0446

More resources

The NIH Molecular Modeling Home Page
http://www.nih.gov/molecular_modeling/mmhome.html

Index

This page intentionally blank.