DOT 2.0 User Guide

Victoria A. Roberts
Michael E. Pique
Susan Lindsey
Martin S. Perez
Elaine E. Thompson
Lynn F. Ten Eyck

San Diego Supercomputer Center
University of California, San Diego
9500 Gilman Drive
La Jolla, CA 92093
http://www.sdsc.edu/CCMS
Email: dot-help@sdsc.edu

October 1, 2008

Contents

Chapter 1

Introduction

```
DOT 2 User Manual CVS $Id: du-intro.tex,v 1.8 2008/10/02 01:57:02 mp Exp $
```

A What DOT does

DOT is a macromolecular docking program that carries out a complete, systematic rigid-body search for two molecules. Intermolecular interaction energies are calculated as the sum of electrostatic and van der Waals terms, which are efficiently evaluated as correlation functions. One molecule (S) is kept stationary and a second molecule (M) is moved about S. The electrostatic and shape properties of both molecules are mapped onto equal-sized grids. M is translated by being centered at each grid point of S, interaction energies are calculated, M is rotated, and the very efficient translational search is repeated. The calculation is dependent on the size of the grid and the number of orientations applied to M. The output is a ranked list of placements of M about S, from which coordinate files in PDB format can be generated. The resulting configurations can be analyzed visually with computer graphics, filtered by biochemical or spectroscopic data, analyzed to find clustering, subjected to methods that introduce flexibility,

An important consideration is the size of the grid. The grid representing S is repeated in all directions (periodic boundary conditions). Therefore, the grid must be large enough that M does not see adjacent copies of S or their properties.

The significantly enhanced new version of the DOT software package provides the following:

- Automated setup of DOT input files starting with protein coordinate files from the PDB.
- Improvements in molecular potentials that have been described in literature are now part of the automated setup.
- Error checking during setup of input files to detect potential problems before the docking calculation is run.
- Faster - DOT now runs 33% faster.
- Portability - will run on Linux, Mac OS X, and Solaris.
- Reevaluation of top-ranked DOT protein-protein complexes with ACE (pairwise atomic contact energy), which takes into account desolvation energy.

B Key DOT Chapters

- Chapter 8 - DOT Installation.
- Chapter 2 - DOT Tutorial, try out first.
- Chapter 3 - Docking your own system, the basics.
- For help - Send email to the DOT help line (dot-help@sdsc.edu) or post to the DOT Users Forum (dot-users@sdsc.edu). You must be a subscriber to the mailing list to post. Please see: http://lists.sdsc.edu/mailman/listinfo/dot-users to subscribe to the list.

C Acknowledgements

We thank the Department of Energy (DOE), the National Science Foundation (NSF), and the National Institutes of Health (NIH) for funding.

DOT setup depends upon the programs APBS, Reduce, and MSMS. We appreciate the work of the authors of these programs and strongly encourage you to acknowledge their efforts should you publish work aided by DOT. The appropriate acknowledgements are:

- APBS: Baker NA, Sept D, Joseph S, Holst MJ, McCammon JA. Electrostatics of nanosystems: application to microtubules and the ribosome. Proc. Natl. Acad. Sci. USA 98, 10037-10041 (2001).
- Reduce: J. Michael Word , Simon C. Lovell, Jane S. Richardson, David C. Richardson. Asparagine and Glutamine: Using hydrogen atom contacts in the choice of side-chain amide orientation. J. Mol. Biol 285, 1735-1747 (1999).
- MSMS: Michel Sanner, Arthur J. Olson, Jean Claude Spehner. Reduced Surface: an efficient way to compute molecular surfaces. Biopolymers 38, 305-320 (1996).

Chapter 2

Try DOT: a Tutorial Example

```
DOT 2 User Manual CVS $Id: du-tutorial.tex,v 1.5 2007/12/08 03:59:54 mp Exp $
```

We assume DOT and its utilities have been installed, see Chapter 8 for instructions. For this tutorial, you will also need the programs APBS, Reduce, and MSMS installed on your computer. In this tutorial, you will set up your environment for running DOT, run DOT with prepared input files to test that DOT is properly located and running, and prepare DOT input files to test that the DOT utilities and the programs APBS, Reduce, and MSMS are properly located and running.

A Set up user environment

To find DOT, you must set the DOT_ROOT environment variable to where DOT is installed, For example, if DOT is installed in /usr/local/dot2, to set your program-search path to include DOT programs, do:
For csh/tcsh:

```
setenv DOT_ROOT /usr/local/dot2
source $DOT_ROOT/bin/share/dot2.setup.csh
```

If you put these two lines into your home directory .cshrc file you will not have to type them again.
For sh/bash:

```
export DOT_ROOT=/usr/local/dot2
source $DOT_ROOT/bin/share/dot2.setup.bash
```

If you put these two lines into your home directory .login or .bashrc file you will not have to type them again.

To check that DOT can now be located, type:

```
rundot --help
```

You should see a single line giving a "Usage:" message. If not, check that your PATH matches the location where the DOT distribution is installed by typing

```
ls $DOT_ROOT
```

which gives a listing that should include the directories bin, data, src, and test, among others.

B Copy the tutorial files to your directory

Make a new directory, for example "testdot", and copy the DOT tutorial files to it:

```
mkdir testdot
cd testdot
dot_tutorial
```

This puts the tutorial files in two directories named “test-rundot” and “test-prepscript”.

C Run DOT with prepared input files

You can now do a short DOT run to check that all is well. Type the two lines:

```
cd test-rundot
rundot udgugi.deg72.nb0.parm
```

This will start a DOT run on your computer, docking udg (uracil-DNA glycosylase) with ugi (an inhibitor protein), using a coarse (thus fast) 72-degree rotation increment.

You will see a page or two of log file output, followed by an opportunity for you to record an entry in a lab notebook file named “dotruns”:

```
Result for dotruns log file (Type control-D to end) >
```

You can then type remarks, such as

```
first example run, took 2 minutes
```

and then type ENTER (or RETURN), then control-D on a line by itself to complete your entry.

The DOT output for all runs started in a given directory goes into a subdirectory named “runs”. Rundot creates subdirectory in “runs” starting with a numerical time-stamp and ending with “udgugi.deg72.nb0”, taken from the name of the .parm file, For example, by doing the DOT run on October 1, 2007 at 7:32 PM, we got the directory name “20071001.1932.udgugi.deg72.nb0”. If you go there, you will find the directory “top30pdb”, which contains PDB files of the ugi inhibitor:

```
cd runs/20071001.1932.udgugi.deg72.nb0/top30pdb
```

These PDB files of ugi are relative to the udg coordinates in the file ‘udg.cen.noh.pdb’ one directory up (in 20071001.1932.udgugi.deg72.nb0 in our case). Examine the file ‘udg.cen.noh.pdb’ and the ugi files in “top30pdb” with your favorite molecular visualization program to see the 30 top-ranked complexes found by DOT.

D Prepare DOT input files using supplied PDB files

Next, you will prepare the DOT input files, starting with two PDB files we supply in the distrbution.

D.1 Make sure necessary programs are installed

You need three programs, MSMS, APBS, and Reduce, to prepare your molecules for DOT, although DOT itself runs without them. First, we will check to see if these three programs are already available. Type

```
which msms
which apbs
which reduce
```

We supply copies of APBS and Reduce in the distribution, in $DOT_ROOT/bin/$ARCHOSV, so if your computer platform is one of the ones supported in the current DOT distribution, the path to these two programs should print out after the “which” command.

We do not supply a copy of MSMS, so you will need to download it if you do not already have it. See Chapter 8 of this manual for instructions. Once you have installed MSMS (and APBS and Reduce if you do not use the ones in the DOT distribution), you must add its location to your shell PATH. For example, if MSMS is installed in /usr/local/bin, type: For csh/tcsh:

```
set path=(/usr/local/bin $path)
```

For sh/bash:

```
export PATH="/usr/local/bin:$PATH"
```

D.2 Run “prepscript”

Go to the supplied prepscript test directory:

```
cd testdot/test-prepscript
```

Then go to the subdirectory coords:

```
cd coords
ls
```

In the coords directory, there are two PDB files, udg.pdb and ugi.pdb, and the ready-to-run script, ‘prepscript’.

Now, run prepscript, keeping a log file: For csh:

```
./prepscript |& tee prepscript.log
```

For bash:

```
./prepscript 2>&1 |tee prepscript.log
```

This will take several minutes and generate a few pages of log file output, including a few warning messages you can disregard. The last few lines of the output should resemble:

```
Stationary potential clamp high will be 1.8275
Stationary potential clamp low will be -1.7861
Stationary molecule files are complete
/test-prepscript/coords
Generating sample DOT parm file udgugi.zero-rotation.nb0.parm
Generating sample DOT parm file udgugi.deg72.nb0.parm
Generating sample DOT parm file udgugi.deg06.nb0.parm
Generating sample DOT parm file udgugi.deg06.nb10.parm
To remove all files normally created by prepscript, type
rm *.parm *.cen.* *.center.* *.minmax *.log *.com
```

DOT input files for the stationary molecule will be in the directory coords/udg, files for the moving molecule will be in coords/ugi. DOT parameter files (.parm) will be created in the parent ‘test-prepscript’ directory. If you like, you can go up to that directory now and run DOT using the DOT input files you just prepared:

```
cd ..
rundot udgugi.deg72.nb0.parm
```

Chapter 3

DOT Quick Start Guide

```
DOT 2 User Manual CVS $Id: du-qs.tex,v 1.14 2008/05/09 04:57:58 mp Exp $
```

DOT, its utilities, and the programs MSMS, APBS, and Reduce must already be installed, see Chapter 8 for instructions.

A Set up user environment

Goal: The goal of this section is to make the DOT program, DOT utilities, and the auxilary programs MSMS, APBS, and Reduce available to the user.

A.1 $DOT_ROOT

To find DOT and its utilities, the DOT_ROOT environment variable must be set to where DOT is installed. For example, if DOT is installed in "/usr/local/dot2", type: For csh/tcsh:

```
setenv DOT_ROOT /usr/local/dot2
```

For sh/bash:

```
export DOT_ROOT=/usr/local/dot2
```

Your path must be set to access DOT utilities, scripts, and data:

```
For csh/tcsh:  source $DOT_ROOT/bin/share/dot2.setup.csh
For sh/bash:   source $DOT_ROOT/bin/share/dot2.setup.sh
```

A.2 MSMS, APBS, Reduce

You need the three programs MSMS, APBS, and Reduce to prepare your molecules for DOT, although DOT itself runs without them. If you have successfully run the tutorial (Chapter 2), these programs are already in your shell PATH. To verify, type:

```
which msms
which apbs
which reduce
```

Copies of APBS and Reduce are supplied in the DOT distribution in $DOT_ROOT/bin/$ARCHOSV, so if your computer platform is one of the ones supported in the current DOT distribution, the path to these two programs should print out after the "which" command. If MSMS is not installed, see Chapter 8 of this manual for instructions. After installing

MSMS, its location must be added to your shell PATH. For example, if MSMS is installed in /usr/local/bin, type:

```
For csh/tcsh:  set path=(/usr/local/bin $path)
For sh/bash:   export PATH="/usr/local/bin:$PATH"
```

B Set up your molecular system

Goal: In this section, you create a directory structure for your project, and generate and run the scripts necessary to create the input files for DOT.

B.1 Create Working Directories

To create the directory structure for your project, first create a main directory, and under this a subdirectory "coords". For example, if the project directory is named "projdir", type:

```
mkdir projdir
cd projdir
mkdir coords
```

B.2 Select the stationary and moving molecules.

Copy the coordinates (in PDB format) of the two molecules to be docked to the "coords" subdirectory of your project directory. You must decide which molecule will be stationary and which molecule will be moved about the stationary molecule. Generally, the molecule with the largest dimension should be the stationary molecule and the smaller molecule should be the moving molecule. This is computationally most efficient (see Section K.1), resulting in both a smaller grid size and fewer orientations of the moving molecule to get reasonable rotational sampling. Throughout this manual stat.pdb will be the stationary molecule stat.pdb and mov.pdb will be the moving molecule. Typically we use names from the PDB, like 1cco.pdb. If you are only using one molecule from a PDB file that contains multiple copies of the molecule, edit the file in "coords" so that it contains only the single molecule.

So now we have

```
projdir/coords/stat.pdb
projdir/coords/mov.pdb
```

B.3 Create prepscript for your molecular system.

In the coords subdirectory, create a copy of prepscript customized for your molecular system by

```
gen_dot_prepscript -m movingpdb -s stationarypdb [-d griddim] [-l residue_library]
[-o prepscript]
```

where "movingpdb" and "stationarypdb" are the PDB files for your two molecules (required), in our example mov.pdb and stat.pdb. By default, gen_dot_prepscript will generate a prepscript that figures out a reasonable size for the grid, uses the default residue library, and is named "prepscript". The optional flags can be used to override these defaults.

Example 1: I want to make prepscript for molecules stat.pdb and mov.pdb, let prepscript determine the grid size, and use the default residue library.

```
gen_dot_prepscript -m mov.pdb -s stat.pdb
```

Example 2: I want to make prepscript for molecules stat.pdb and mov.pdb, use a 128^3 grid with 1 Å spacing, and use my library /home/me/misc/myreslib.rlb.

```
gen_dot_prepscript -m mov.pdb -s stat.pdb -d 128 -l /home/me/misc/myreslib.rlb
```

Both examples create a new file “prepscript” in the coords directory, in our case

```
projdir/coords/prepscript
```

that is customized for the input PDB files mov.pdb and stat.pdb.

B.4 Run prepscript

To run prepscript, keeping a log file, do:

```
For bash:  ./prepscript 2>&1 | tee prepscript.log
For csh:   ./prepscript |& tee prepscript.log
```

For an outline of the steps performed by prepscript and a detailed description of each step, see Chapter 4, page 13.

B.5 Files created by prepscript

Prepscript creates two subdirectories in “coords”, with names based on your molecules. Each directory contains the needed DOT input files for each molecule. In addition, parameter files for running DOT will be put in the main project directory.

In our example with stat.pdb and mov.pdb, where prepscript determined we should use a grid 128 Å on each side (1 Å grid spacing) and the ionic strength for the electrostatic potential calculation is 150mM (default), the following DOT input files are created:

1. Stationary molecule files, in projdir/coords/stat
   (a) stat.128.150m.dx – the electrostatic potential of the stationary molecule, where the general name of the file is molname.[grid_dim].[ionic_strength].dx.
   (b) stat.cen.noh.xyzcrv – the shape potential of the stationary molecule, based on heavy atoms only.
   (c) stat.cen.noh.pdb – the PDB file of the stationary molecule, heavy atoms only, for evaluation.
2. Moving molecule files, in projdir/coords/mov
   (a) mov.cen.polh.xyzq – partial atomic charges for the moving molecule, including polar hydrogen atoms.
   (b) mov.cen.noh.xyz – shape of moving molecule, represented by the atomic centers of heavy atoms only.
   (c) mov.cen.noh.pdb – the PDB file of the moving molecule, heavy atoms only, for evaluation.
3. Parameter files, in projdir
   (a) statmov.zero-rotation.nb0.parm – Single orientation for mov.pdb, no penetrations of mov into stat allowed.
   (b) statmov.deg72.nb0.parm – 72° orientational search (60 orientations) for mov.pdb, no penetrations of mov into stat allowed.
   (c) statmov.deg06.nb0.parm – 6° orientational search (54,000) for mov.pdb, no penetrations of mov into stat allowed.
   (d) statmov.deg06.nb10.parm – 6° orientational search (54,000) for mov.pdb, up to 10 penetrations of mov into stat allowed.

These are the files that DOT uses. Additional files are made in the molecule subdirectories, generated as prepscript proceeds, see Section G.

B.6 Most likely problems

The prepscript script should run with no intervention needed IF

1. The two molecules are proteins with standard residues. The programs in prepscript can handle the various protonation states of His and Cys (free and disulfide).
2. Both molecules begin with residue ID 1 (any chain ID is okay), and you want each N-terminus to be a positively charged amino group.
3. The checks done in prepscript are appropriate for your system, in particular the total charge is not huge.

> **Why?** These problems ALL revolve around the need to assign atomic charges and molecular radii for the stationary molecule to do the electrostatic potential calculations and to assign atomic charges for the moving molecule. The program Reduce, which adds hydrogen atoms depending on the local environment, handles standard amino acid residues. The default DOT residue library contains atomic charges and molecular radii for standard amino acids, including the typical protonation states of His, Cys free or in a disulfide, and charged N- and C-termini. Prepscript assigns a unique residue name for each uniquely charged and protonated state of each standard amino acid. If a residue does not appear in the DOT residue library, prepscript will stop. Prepscript will also stop if the total charge on a molecule is not an integer. This is most likely to happen when a known residue type is not properly protonated. This check is a KEY check that you are building your system properly.

Residues other than standard amino acids. Appropriate charges are also available for HEM and DNA residues; the libraries can be customized to handle new residues (see section 8a., page 29. The program Reduce can also handle DNA residues, and may also add hydrogen atoms properly to unusual functional groups or modified amino acid residues (see section 4e., page 27).

N-terminal residues of a protein. If the N-terminal residue does not have a residue ID of 1, see section 4a., page 24. You must decide if you want it neutral (beginning with a standard residue, which has a single hydrogen atom on the N atom) or positively charged. The most likely problem is a missing hydrogen atom at the beginning of a peptide chain when you intended the terminus to be neutral. If the total charge is off by 0.248, suspect a missing main chain amide hydrogen atom!

Prepscript checks that may not apply to your particular system. One example where checks done in prepscript may not be appropriate for your system is if the total charge on one molecule is very large, which is quite likely for DNA (see section M, page 34).

C Run DOT

Goal: To run DOT and examine DOT output.

C.1 Running DOT

To run the program DOT, first go to the main directory of your project. For our example “projdir”:

```
cd projdir
```

The sample DOT parameter files (.parm) made by prepscript are in this directory.

The command to run DOT is:

```
rundot parameterfile [-h hostfile] [optional comments]
```

The parameter file is required. If no hostfile is specified, DOT will run on one processor on the computer that you are logged into. If your optional comments contain special characters, you should surround them with single quotes. Examples are given below.

For each new molecular system, we STRONGLY suggest first running a test case that uses one processor and does only a single rotation for the moving molecule. This will check that DOT found the input files, that $DOT_ROOT is set properly, and that DOT is running okay. For our project, starting with stat.pdb and mov.pdb, the command to run a single rotation of the moving molecule is:

```
rundot statmov.zerorot.nb0.parm
```

For your system, the .parm files will have your molecule names, as made by prepscript. This runs the DOT on one processor, the one you are logged into.

At the end of the run, you are prompted enter any additional comments, ending with control-D.

C.2 Running DOT on multiple computer/processors

For your project, you will want to do a full rotational search. You could do this on a single processor, but the run can be done much quicker on multiple computers or processors. DOT is designed to run very efficiently on multiple computers in parallel using MPI (Message Passing Interface). All of the computers that you use must have access to the appropriately compiled DOT program. Each computer must also have access to your project directory. Furthermore, you must be able to either “rsh” or “ssh” to each computer. Getting ssh or rsh may require help from a systems administrator. See Chapter 5, Section D.2.

ALERT! If this is the first time you have used DOT with multiple computers or processors, see Section C.5 below before running a time-consuming production run!

To run on multiple computers create a text file in your project directory that is a list of the computers, one name per line. For example, for 3 workstations I make a file called ‘myhosts’

```
dopey
sleepy
grumpy
```

If I want to use multiple processors on a single computer, put one line for each processor. For example, if I have an 8-processor computer dopey and I want to use 3 processors, my “myhosts” file would contain

```
dopey
dopey
dopey
```

For our big production run, we want to sample the rotational search for the moving molecule with a fineness equivalent to that of the translational search. The number of orientations needed is dependent on the size of the moving molecule. We have found that the 6 degree search is appropriate for globular proteins with up to 120 residues (diameter of up to 40 Å). Prepscript creates two .parm files that do a 6 degree rotational search: in one, no atoms of the moving molecule may penetrate the interior of the stationary molecule, in the other, up to 10 moving molecule atoms are allowed to penetrate the stationary molecule. For our system starting with stat.pdb and mov.pdb, to allow no penetrations (or bumps) by the moving molecule, type:

```
rundot statmov.deg06.nb0.parm -h myhosts ’my 0 bump production run’
```

This DOT run will do 54,000 evenly spaced orientations of the moving molecule, which should take a few hours on 5 to 10 processors. For a finer rotational search, examine your .parm file.

To do the same rotational search, but allow up to 10 atoms of the moving molecule to penetrate the stationary molecule, do:

```
rundot statmov.deg06.nb10.parm -h myhosts ’my 10 bump production run’
```

Allowing penetrations (bumps) can be useful for unbound molecules when some structural rearrangement occurs in the complex.

C.3 Where DOT output goes

Each invocation of "rundot" in projdir creates a subdirectory in projdir/runs in this format:

```
projdir/runs/DATE.TIME.prefix_from_parm_file
```

DOT output is put in this subdirectory. For example for 'my 10 bump production run' above started on August 2, 2007 at 10:20 AM, output will be in

```
projdir/runs/20070802.1020.statmov.deg06.nb10
```

The directory name is constructed from the date and time the run was started, the names of the molecules used, the number of orientations applied to the moving molecule, and the number of bumps allowed. The last three fields are taken directly from the name of the .parm file, in this case statmov.deg06.nb10.parm.

In your project directory, the file

```
dotruns
```

is created, which is a cumulative log of all DOT runs started in projdir. This file logs the directory names and your comments at the beginning and end of each run. Very handy!

C.4 DOT output and evaluation

The basic output of DOT put into the runs/DATE.TIME.prefix_from_parm_file output directory is an .e6d file. For our example starting with stat.pdb and mov.pdb:

statmov.top2000.e6d

This file contains the information for generating the 2000 (default) top-ranked placements of the moving molecule. These placements are relative to the centered coordinates of the stationary molecule, in our case, stat.cen.noh.pdb, which is copied into the output directory from projdir/coords/stat.

"Rundot" also performs a preliminary evaluation of the DOT run by running the script "evaluate_dot_run", which processes the information in the .e6d file. The evaluation includes creating PDB files of the 30 top-ranked placements of the moving molecule in the subdirectory "top30pdb". See Chapter 6 for a detailed description of the evaluation performed by "evaluate_dot_run" and further evaluations that you can do.

C.5 Testing DOT on multiple computers/processors

If this is the first time you have run DOT on multiple machines, we suggest that you first test parallel processing on a small DOT run. Prepscript makes a parameter file for doing just 60 rotations of the moving molecule. For your test run, do:

```
rundot statmov.deg72.nb0.parm -h hosts 'test of multiple processors'
```

This will test that all of the machines can access an appropriately compiled DOT program and its utilities, that all machines can access your project directory, and that you get either "rsh" or "ssh" to each computer.

> Bug Warning: currently the number of computers must be no more than the number of orientations of the moving molecule. Since you probably have only a dozen or so computers available and usually have hundreds or thousands of orientations this is not likely a problem except for the single-rotation test case.

Chapter 4

Prepscript - creating input files for DOT

`DOT 2 User Manual CVS $Id: du-prepscript.tex,v 1.11 2008/04/02 03:52:04 vickie Exp $`

Goal: This chapter gives detailed descriptions of the steps needed to set up your molecular system for DOT docking runs. The scripts described in this section are

- gen_dot_prepscript – generates a prepscript customized for your system
- prepscript – creates molecular input files and parameter files for DOT

and the utilities and scripts called by these.

`DOT 2 User Manual CVS $Id: du-prep-outline.tex,v 1.10 2008/04/02 03:52:04 vickie Exp`

A Outline of setup steps

1. Accessing DOT scripts and utilities and auxilliary programs
2. Examine the molecule structures for completeness
3. Select stationary and moving molecules
4. Set up directory structure
5. Set up customized library for new functional groups
6. Create prepscript: gen_dot_prepscript
7. prepscript general features
   (a) tools that prepscript uses
   (b) Assigns systematic naming to needed files: cen=centered, noh=no hydrogen atoms, polh= with polar hydrogen atoms, allh=with all hydrogen atoms
   (c) Assigns file suffixes to indicate file type
   (d) Pause command for user intervention
8. Prepscript: steps common to both molecules – dot2-prep-mol-common
   (a) Define an acceptable range for the total charge on each molecule
   (b) Remove waters, H atoms, alternate locations from PDB files
   (c) Center both molecules with no H atoms (cen.noh) [pdb_make_centered]
   (d) Add hydrogen atoms to PDB coords using Reduce (cen.allh)

  (e) Create files with heavy atoms and polar H atoms with RES names to match library (cen.polh) [pdb_rename_res_by_hydrogens, striph]
  (f) Moving molecule: Create shape file (.xyz) based on heavy atoms
  (g) Moving molecule: Create partial atomic charge file (.xyzq)

9. Prepscript: Calculate diameters of molecules to determine grid size
10. Prepscript: Stationary molecule
    (a) Stationary molecule: Create electrostatic potential (.dx)
    (b) Create APBS command and parameter files
    (c) Run APBS
    (d) Stationary molecule: Create shape potential (.xyzcrv)
    (e) Run MSMS (heavy atoms only) to define excluded and favorable volumes.
    (f) Determine electrostatic clamping values
11. Prepscript: Create parameter files for DOT run (.parm)

B Accessing DOT scripts and utilities and auxilliary programs

It is assumed that you have set up your user environment to access

- DOT scripts and utilities – see Chapter 2, section A
- the programs APBS, Reduce, and MSMS – see Chapter 2, section D.1. Note the DOT distribution does not include MSMS.

If you successfully ran the tutorials (Chapter 2), your user environment includes the environment variable $DOT_ROOT and the following directories are in your path. Platform independent bash and csh scripts are in:

$DOT_ROOT/bin/share

Compiled C/C++ utilities are in:

$DOT_ROOT/bin/$ARCHOSV

where $ARCHOSV is replaced by your platform/operating system name that is determined by the script $DOT_ROOT/bin/share/archosv. If you are using the APBS and Reduce programs distributed with DOT, they are in

$DOT_ROOT/bin/$ARCHOSV

. If you followed the installation instructions for MSMS (Chapter 8, Section **??**.**??**), it will also be in

$DOT_ROOT/bin/$ARCHOSV

The directory

$DOT_ROOT/data

includes residue atomic charge libraries (.rlb), rotation sets that are applied to the moving molecule (.eul), auxilliary files for MSMS (calculation of molecular surfaces) and ACE (for evaluation), and sample files.

C Examine and complete starting PDB files

Prepscript, the script that creates the DOT input files, takes as input two starting PDB files, each with a single copy of a macromolecule. If multiple copies of the macromolecule are present in the asymmetric unit of a crystallographic file, the user must decide which set of coordinates is to be used for docking and create an input PDB file that contains just that molecule. The molecule can include multiple chains. Prepscript will remove water molecules, hydrogen atoms, and multiple positions of a residue in the PDB files unless the user explicitly does not want this to happen and edits prepscript appropriately. Each residue in the PDB file must include all of the heavy (nonhydrogen) atoms that correspond to that residue in the residue library. It is okay if the macromolecule is missing an entire residue or a region consisting of many residues, such as missing residues at the N- or C-termini or disordered loop regions that do not appear in the PDB file.

C.1 Missing atoms in side chains.

If a PDB file lacks the full set of heavy atoms that define a residue, the PDB curators usually indicate this in REMARK 470, MISSING ATOMS. The heavy atoms must match those associated with the residue name, hence the user has two choices: (1) build the full side chain or (2) rename the residue to match the existing atoms. For example, given a lysine residue with atoms only to Cγ, the user could either build the full Lys side chain by adding atoms Cδ, Cϵ, and Nζ, or could remove the Cγ atom, leaving only the Cβ atom of the side chain, and rename the residue “ALA”. In this case, the advantage of building the full side chain is that the correct total charge is retained. The disadvantage is that the side chain was probably disordered, and may have multiple, similar-energy conformations. Generally, we build the full side chain. Building side chains is outside the scope of the DOT program suite, but proprietary programs, such as Insight II (Accelrys), and open source tools, such as SWISS PROT**, can be used to do this task. Since residues with disordered atoms are usually on the surface of the protein, they can affect the shape and electrostatic properties of interaction surfaces and hence should be built carefully. For example, added atoms should be checked for steric clashes with other atoms in the molecule.

If the missing heavy atoms are not added, prepscript will find that the protein’s total charge is not an integer, report the error, and stop.

C.2 Missing loops and chain ends

Missing residues are often indicated in the PDB file as REMARK 465, MISSING RESIDUES. In general, we do not build in missing residues for rigid-body docking. The user may decide that missing regions need to be built to make a complete model, but these regions usually are missing because they are disordered and probably have multiple conformations. If the missing loop or N- or C-termini regions are not built, there are two considerations. First, the resulting termini should be uncharged so that spurious charged groups, which can significantly affect the overall charge distribution, are not introduced. One way the user can approach this is to leave the termini as standard amino acids. This results in incomplete covalent bonding for the terminal main-chain atoms, N at the N-terminus and C at the C-terminus, which is not a problem for the computational docking. Alternatively, the user could use a molecular modeling program to build acetyl (ACE) groups onto N-termini or N-methyl groups (NME) onto C-termini, also leaving both ends neutral. Second, missing regions, especially those at the N- and C-termini, can indicate surface regions where an incoming molecule is unlikely to bind. Docked configurations found by DOT that contact these incomplete termini may be eliminated as likely complexes.

```
DOT 2 User Manual CVS $Id: du-prep-statmov.tex,v 1.4 2007/10/30 06:06:19 vickie Exp $
```

D Assigning the stationary and moving molecules

In the DOT docking calculation, one molecule is stationary and the other molecule is moved about it. There are several criteria for selecting which molecule will be stationary and which will be moved. (1) The computational time required for a translational and orientational search of a given fineness. The computational time is dependent on the size of the grid and the number of orientations applied to the moving molecule. (2) Suitability or convenience of the potentials used to describe the molecules, since the stationary and moving molecule potentials are quite different from each other. (3) Other factors specific to the molecular system.

D.1 Computation time.

Computational time for a given resolution of the search is shortest when the larger molecule is stationary (S) and the smaller molecule is assigned as moving (M). The computational time is dependent on two factors: the number of points in the grid and the number of orientations applied to the moving molecule.

The grid representing S is repeated in all directions (periodic boundary conditions). Given the default grid spacing used by DOT of 1 Å, grids that are 32, 64, 96, 128, 160, 192, and 224 Å on each side are the most efficient for the DOT algorithm. The goal is to select the minimum grid size where M does not see adjacent copies of S or their properties. A good rule of thumb is that when M contacts S, M lies entirely inside the grid. A generous measure of this is that the grid dimension should be larger than $S + 2M$, where S and M are the largest diameters for the two molecules. $S + 2M$ will be smaller when the larger molecule is assigned as S.

The properties of the molecules may be a factor in choosing the size of the grid. For example, if S creates a strong electrostatic potential, there may be significant values at the edges of the grid. This is most likely when S is highly charged or has a large region with a high concentration of residues with similar charge. When M is centered on a grid point distant from S, M may see electrostatic potential from copies of S that distorts the electrostatic energy. In these cases, a grid size larger than $S + 2M$ may be worth considering.

The computational time is linearly dependent on the number of orientations applied to the moving molecule. For a given set of orientations, the rotational space of the moving molecule will be more finely sampled for a smaller molecule. For example, for two molecules, one with twice the diameter of the other, the larger molecule would require 8 times the orientations to give the same orientational sampling of the smaller molecule.

D.2 Suitability of potentials.

The potentials of S are calculated once, whereas the potentials of M must be recalculated for each orientation of M. Since the potentials of S are calculated only once, they can be quite detailed and computationally intensive. The shape potential of S requires calculation of molecular surfaces and determination of the volumes of the grid that represent the excluded and favorable regions. The electrostatic potential of S is calculated by Poisson-Boltzmann methods, which take into account ionic strength effects and the dielectric boundaries between solute and solvent. Both the shape and electrostatic potential calculation for the stationary molecule take several minutes. For computational feasibility, the shape and electrostatic properties of the moving molecule must be rapidly calculated. In DOT, the shape is represented by the atomic coordinates of all non-hydrogen atoms and the charge distribution by point charges at the atomic coordinates, including polar hydrogen atoms. Both are rapidly mapped onto the grid for each orientation. If the user wants one molecule be defined in more detail than the other, that molecule should be assigned as S.

We have found the reverse case when a fragment of double-stranded DNA is used to represent part of a long DNA strand []. The DNA fragment works best as M, where its charge distribution is represented by atomic point charges. When the electrostatic potential for a DNA fragment is calculated by Poisson-Boltzmann methods, the greater solvent accessibility of the ends creates a nonuniform electrostatic potential along the DNA strand. Near the ends of the fragment, the electrostatic potential is neutral; only in the center of the fragment is the potential due do the phosphate atoms constant. Thus the electrostatic potential of the DNA fragment is highly dependent on its length. When the DNA fragment is represented as point charges, the charge distribution is consistent over the full fragment.

D.3 Other factors.

One protein environment may be more suitable represented as either S or M. For example, if one molecule is embedded in a membrane environment, this could be included as part of the description of the stationary molecule.

```
DOT 2 User Manual CVS $Id: du-prep-directory.tex,v 1.4 2007/10/30 06:06:19 vickie Exp
```

E Set up directory structure with starting coordinates

First, create a main directory for your project, then create a subdirectory “coords”. For example, if we decide the main project directory will be named “projdir”, we move to the directory where “projdir” is to be created and type:

```
mkdir projdir
cd projdir
mkdir coords
```

This makes the directory structure:

```
projdir/coords/
```

Put the prepared PDB files of the moving and stationary molecules “projdir/coords”. For example, if our starting PDB files are stat.pdb (stationary molecule) and mov.pdb (moving molecule), we have

```
projdir/coords/stat.pdb
projdir/coords/mov.pdb
```

DOT 2 User Manual CVS $Id: du-prep-library.tex,v 1.1 2008/05/09 05:29:49 mp Exp $

F Set up customized library for new functional groups

F.1 Standard residue library for proteins

The default residue library is

```
$DOT_ROOT/data/uhbd.amber84.prot.rlb
```

This residue library contains partial atom charges and radii for

- the 20 amino acid residues,
- the 20 amino acid residues as N-terminal residues of a peptide chain,
- the 20 amino acid residues as C-terminal residues of a peptide chain,
- HIS protonated on NE2 alone and ND1 alone (neutral) and protonated on both,
- CYS, both free and as part of a disulfide
- ACE, acetyl group that sometimes begins a peptide chain,
- NME, methyl amino group that sometimes ends a peptide chain

For a list and description of the residue names, see

```
$DOT_ROOT/data/uhbd.amber84.prot.README
```

The library includes polar hydrogen atoms, but not nonpolar hydrogen atoms. Partial atomic charges match those from AMBER for amino acid residues with polar hydrogen atoms only (Weiner et al., (1984), J. Amer. Chem. Soc. 106, 765-784). The library format is that used by the UHBD (University of Houston, Brownian Dynamics) program (Gilson et al., (1993) J. Phys. Chem. 97, 3591). The library has two sections. The first “EQUIVALENCE” allows the user to equivalence an atom name in their PDB files to the atom name used in the library. For example, the main-chain amide ‘H’ used to sometimes be called ‘HN’, so for ALA, the line in the “EQUIVALENCE” section is

```
ALA HN ALA H
```

which maps HN in ALA in the input PDB file to H in ALA in the residue library. These lines are less needed now, with the remediation and standardization of atom names in the PDB.

The second section, "AMBER" contains the residue name (resi), atom name (atom), charge (chrg), and radius (radi). The library currently includes 2 additional fields for UHBD Brownian dynamics, epsi and sigm, but these are not needed for DOT and should be set to 0.0 when new residues are added to the library.

F.2 Additional residue libraries

The directory

```
$DOT_ROOT/data/cofactors
```

contains residue libraries for DNA (including 5′ and 3′ termini), RNA (with 5′ termini beginning with either the sugar or including a preceding phosphate group and 3′ termini), GTP, ATP, and heme (Fe(II)) that are consistent with the polar atom atomic charges of AMBER. Currently, only a single library file can be used, so these libraries need to be appended to the standard protein library. For example, for a DNA/protein system in our directory "projdir", I make a subdirectory data, copy the standard protein library and the DNA library into it, and append the DNA library to make a new library file uhbd.amber84.prot.dna.rlb.

```
cd projdir
mkdir data
cd data
cp $DOT_ROOT/data/uhbd.amber84.prot.rlb .
cp $DOT_ROOT/data/cofactors/dna.amber.rlb .
cat uhbd.amber84.prot.rlb dna.amber.rlb > uhbd.amber84.prot.dna.rlb
```

To use your new library in prepscript, you must add the -l flag and the **FULL** pathname of your new library in your gen_dot_prepscript command (Section G).

DOT 2 User Manual CVS $Id: du-prep-gen.tex,v 1.7 2008/04/02 03:52:04 vickie Exp $

G Create Prepscript

"Prepscript" is an executable script that produces all the needed DOT input files, including molecular descriptions for the stationary and moving molecules and parameter files for running DOT (see Section ** for a list of files). You will use **$DOT_ROOT/bin/gen_dot_prepscript** to generate "prepscript" customized for your system. For gen_dot_prepscript and prepscript to work, you need to

- Have your path set up to access DOT scripts and utilities (Chapter 2, section A).
- Have your path set up to access the programs REDUCE, MSMS, and APBS (Chapter 2, section **??**).
- Set up the directory structure with starting coordinates (Section E)
- If needed, make a customized residue library for non-standard protein residues and other ffunctional groups, such as cofactors (Section F)

For our example, with "projdir" as our project directory and the starting PDB files stat.pdb (stationary molecule) and mov.pdb moving molecule), both PDB files are copied into **projdir/coords**, as set up in E.

The script "gen_dot_prepscript" will create prepscript for your system. Syntax is:

```
gen_dot_prepscript -m movingpdb -s stationarypdb [-d grimdim] [-l residue_library]
```

An appropriate grid dimension is calculated by prepscript if it is not specified, see section J. The default library assigns atomic charges and radii and is in:

```
$DOT_ROOT/data/uhbd.amber84.prot.rlb
```

Example 1. Working in the project directory “projdir” with the PDB files stat.pdb and mov.pdb “projdir/coords”, using the default library, and letting prepscript calculate the grid size.

```
cd projdir/coords
gen_dot_prepscript -m mov1.pdb -s stat.pdb
```

Example 2: Same coordinates, but we want to use a 128^3 grid with 1 Å spacing and the customized residue library with atomic charges “myreslib.rlb” in projdir/lib:

```
gen_dot_prepscript -m mov.pdb -s stat.pdb -d 128 -l projdir/lib/myreslib.rlb
```

Note the the full pathname should be specified for the library. A customized library can be built by adding your new functional groups to a copy of the default library. For example, our researcher copied the default protein residue library

```
$DOT_ROOT/data/uhbd.amber84.prot.rlb
```

into the directory “projdir/lib” and named this residue library file myreslib.rlb. The library assigns partial atomic charges and atomic radii to each atom of a residue. To add a new residue to the library, the user should assign radii consistent with those in the default library and obtain reasonable partial atomic charges consistent with those of the AMBER polar hydrogen force field [], following the format of the library.

DOT 2 User Manual CVS $Id: du-prep-general.tex,v 1.5 2008/04/02 03:52:04 vickie Exp $

H Prepscript general

H.1 General Approach

Prepscript will work without intervention if the two molecules are proteins with standard residues, contain no cofactors not in the current charge library, and both start with residue number 1, which should be treated as a positively charged N-terminus. If intervention is needed, the PAUSE command can be inserted with a comment to remind the user that a file needs to be adjusted before proceeding.

It is important to check the log files for errors! They will be located in the “stat” and “mov” directories (where, again, ’stat’ and ’mov’ are example names used in this manual: normally you will have called your stationary and moving molecules names like ’1abc.pdb’ and ’5gfd.chainB.pdb’).

H.2 Running prepscript

For prepscript to work, you need to

- Have your path set up to access DOT scripts and utilities (Chapter 2, section A).
- Have your path set up to access the programs REDUCE, MSMS, and APBS (Chapter 2, section **??**).
- Set up the directory structure with starting coordinates (Section E)

“Prepscript” is run in the “coords” subdirectory of your project. We highly recommend you make a log file when running prepscript. To run prepscript, type:

For csh:

```
./prepscript |& tee prepscript.log
```

For bash:

```
./prepscript 2>&1 |tee prepscript.log
```

Example. Working in the project directory “projdir” using csh:

```
cd projdir/coords
./prepscript |& tee prepscript.log
```

H.3 Assignment of file names

Prepscript automatically adds to the file name of the original input PDB files to indicate what is in the file and the file type. Systematic names include

- cen = centered
- noh = heavy atoms with no hydrogen atoms
- polh = heavy atoms with only polar hydrogen atoms
- allh = heavy atoms with all hydrogen atoms

Suffixes that indicate file types include

- xyz = Contains X, Y, and Z coordinates only
- xyzq = Contains X, Y, and Z coordinates and partial atomic charge
- xyzqr = Contains X, Y, and Z coordinates, partial atomic charge, and radius
- xyzqr.xml = as xyzqr but in format for APBS input
- uhbdgrd = electrostatic potential grid created in UHBD format
- dx = electrostatic potential grid created in APBS format
- xyzcrv = shape file

If the preparation script succeeds then the following files will be generated. $projdir/coords/mov/mov.pdb
$projdir/coords/mov/mov.center.xyz
$projdir/coords/mov/mov.cen.polh.xyzq
$projdir/coords/mov/mov.cen.polh.pdb_to_xyzq
$projdir/coords/mov/mov.cen.polh.pdb
$projdir/coords/mov/mov.cen.polh.xyz

$projdir/coords/stat/stat.pdb
$projdir/coords/stat/stat.center.xyz
$projdir/coords/stat/stat.cen.uhbd.pdb
$projdir/coords/stat/stat.cen.polh.pdb

$projdir/coords/stat/stat.cen.pdb
$projdir/coords/stat/stat.cen.noh.pdb
$projdir/coords/stat/stat.cen.polh.xyzqr.xml
$projdir/coords/stat/stat.cen.polh.apbs.log
$projdir/coords/stat/stat.cen.128.150.0m.dx (or similar)
$projdir/coords/stat/stat.cen.noh.xyzcrv
$projdir/coords/stat/stat.cen.noh.r+1.4.p=1.4.xyzcrv
$projdir/coords/stat/stat.cen.noh.clamp.minmax

H.4 The “pause” command, pausing during creation of DOT input files

If you insert

pause ’*reason*’

into prepscript or the scripts called by prepscript, such as “dot2-prep-mol-common”, the script will pause at this statement, which the ’reason’ reminding you of what you need to do. NOTE: Use the single quote “ ’ ”to delineate the comment. Type enter (or return) will make your script resume.

Example 1: I want to check the DOT input files for the moving molecule before I move on to processing the stationary molecule. I put a “pause” statement in prepscript after the moving molecule is processed:

```
dot2-prep-mol-common -p mov.pdb -l $reslib -w moving
exit_if_error $?  dot2-prep-mol-common reported an error, prepscript quitting
popd
echo Moving molecule files are complete

pause ’check moving molecule files mov.cen.noh.xyz and mov.cen.polh.xyzq’
```

Example 2: I have a CYS residue that is bound to a Cu atom in plastocyanin, my moving molecule. Since there is no hydrogen atom on the S atom, it will be automatically assigned as CYX, a neutral residue that is part of a disulfide. However, I have created a residue CYM, that has the appropriate charge of -1 needed for CYS as a metal ligand. To make sure the residue gets assigned properly, I have to intervene after all hydrogen atoms have been added by REDUCE, but before atomic charges are assigned. This requires making a customized version of ’dot2-prep-mol-common’ for my molecule mov.pdb. After all hydrogen atoms are added with REDUCE, but before non-polar hydrogens are removed, I put:

```
pause ’edit mov.cen.allh.pdb to replace Cu-bound CYS 84 with CYM’
```

I edit the file and then type enter (or return) to continue the processing of DOT input files.

```
 DOT 2 User Manual CVS $Id: du-prep-steps.tex,v 1.8 2008/04/02 03:52:04 vickie Exp $
```

I Prepscript: Steps common to both molecules – dot2-prep-mol-common

The script “dot2-prep-mol-common” is run in prepscript for both the moving and the stationary molecules. The script “dot2-prep-mol-common” does the following steps:

- removes hydrogen atoms and water molecules from the starting PDB files
- centers the molecules
- calls the program Reduce to build hydrogen atoms

- creates the centered PDB files with heavy atoms and polar hydrogen atoms, with each protonation/charge state having a unique residue name
- creates the moving molecule shape file (.xyz)
- creates the atomic charge file (.xyzq) and checks total molecular charge

These steps create all the required DOT input files for the moving molecule and create the files needed to calculate the shape and electrostatic potentials of the stationary molecule.

I.1 Check on range for total molecular charge

Before running "dot2-prep-mol-common", prepscript sets the parameters "qtotmin" and "qtotmax" to define an acceptable range for the total charge on each molecule. The default in prepscript is

```
qtotmin=-20
qtotmax=20
```

If the total charge of either molecule is not in this range, Prepscript will quit with an error message. If you know the total charge of your molecule is an integer outside this range, adjust the limits accordingly. For example, a fragment of double-stranded DNA with 12 base pairs has a total molecular charge of -22, so changing "qtotmin" to equal -30 will allow prepscript to run without an error. Within prepscript, the total molecular charge is checked for each molecule by the script "dot2-prep-mol-common" and checked in the log file made by either APBS or UHBD when the electrostatic potential for the stationary molecule is calculated.

If the the total charge for both the stationary and the moving molecule is in the range defined by "qtotmin" and "qtotmax", but is not an integer, prepscript will quit with an error message. This turns out to be an excellent check of whether atomic charges got assigned correctly. Residues or associated residues have integral charges in the residue library. A non-integral charge usually indicates missing atoms, either because atoms are missing in the original PDB files or because hydrogen atoms were not added correctly.

```
DOT 2 User Manual CVS $Id: du-prep-noh.tex,v 1.4 2008/04/02 03:52:04 vickie Exp $
```

I.2 Creating no-hydrogen files (noh files)

The script "dot2-prep-mol-common", called by prepscript, removes water molecules, hydrogen atoms, and alternate positions for the same atom from the input PDB file. For a starting molecule named mol.pdb, the file created is:

- mol.noh.pdb

in PDB format.

> **Why?** Input PDB files can include small molecules that would not be considered to be part of molecular shape presented to docking partners. These molecules include water molecules, ions from buffer, and some metal ions. Prepscript removes water molecules, but the user must decide if other molecules are an intrinsic part of the molecular structure and remove those that are not. For example, a metal ion may be an important cofactor (keep) or present between molecules in the crystal as a heavy atom derivative (remove). Some crystallographic structures and most NMR structures include some or all hydrogen atoms. These atoms may have nonstandard names or geometries, depending on the method of refinement. This step of prepscript will remove them, and they will be replaced in later steps. If the user wants to retain hydrogen atoms from the PDB, skip this step. Some PDB have multiple locations for some atoms, say a mobile side chain. Prepscript selects only the first listed alternate location for an atom.

If multiple molecules are present in the asymmetric unit of a crystallographic file, the user must decide which molecule is to be used for docking, and edit the input file accordingly to contain just one molecule.

```
DOT 2 User Manual CVS $Id: du-prep-center.tex,v 1.5 2008/04/02 03:52:04 vickie Exp $
```

I.3 Centering molecules

The script “dot2-prep-mol-common”, called by prepscript, finds the geometric center of each molecule and translates the coordinates so that the center lies at (0,0,0). Files of centered coordinates have ”cen” added to their name. In prepscript, the centering is applied to the molecule with no hydrogen atoms, so for a starting molecule named mol.pdb, files created are:

- mol.cen.noh.pdb - the centered PDB file with no hydrogen atoms
- mol.centered.xyz - the geometric center of the original PDB file (**Check this)

> **Why?** It is especially important that the moving molecule is centered. This ensures an even sampling by the set of orientations applied to the moving molecule, since the orientations are rotated about (0,0,0). If the moving molecule is significantly off center, large regions of rotational space may be missed.
> For the stationary molecule, centering positions the molecule in the center of the grid, providing the largest distance between the molecule and the edges of the grid for the moving molecule to fit into. It may be convenient to move the stationary molecule from a centered position. For example, in dockings to a fragment of cytochrome c oxidase, the face of the fragment that is connected to the membrane-bound portion was translated to the bottom of the grid, leaving available more space in the grid for the molecule being docked.

3a. Scripts, utilities used

```
$DOT_ROOT/bin/share/pdb_make_centered  mol.pdb > mol.cen.pdb
```

This finds the geometric center of the molecule and writes a new PDB file with the same atoms but moved so the molecule’s center is at the point (0,0,0).

```
 DOT 2 User Manual CVS $Id: du-prep-allh.tex,v 1.14 2008/04/02 03:52:04 vickie Exp $
```

I.4 Protonation with REDUCE (allh files)

The script “dot2-prep-mol-common”, called by prepscript, uses the program REDUCE to add hydrogen atoms, both polar and nonpolar, to the already centered coordinates. Within prepscript, given a starting PDB file, for example mol.pdb, the PDB file created by Reduce will be named:

`mol.cen.allh.pdb - the centered PDB file with all hydrogen atoms`

When Reduce protonates the macromolecule, it determines a reasonable protonation state for His and Cys side chains from the local environment.

For protonation, 3 problems need to be considered:

1. The appropriate state of the N-terminal residue of a peptide chain.
2. Protonation state of residues like His and Cys.
3. Protonation state of unusual functional groups.

Reduce automatically assigns N-termini as positively charged IF the residue has residue number 1. Other cases require special consideration, see below. INCORRECT PROTONATION OF N-TERMINI is the most likely reason for a nonintegral charge on a protein when the atomic point charges are assigned (see Section 4a.).

For a true N-terminal residue, the N-terminus should be an amino group (-NH_3), consisting of atoms N, H1, H2, and H3, that contributes an overall charge of +1 to the residue. For a true C–terminal residue, the C-terminus should be a carboxylate group (-CO_2), consisting of atoms C, O, and OXT, that contributes an overall charge of -1 to the residue. The termini of the coordinates often do not represent the true ends of the protein; there can be disordered regions at the ends of one or both termini that are not in the PDB coordinates. These missing residues should be indicated by a REMARK

465, MISSING RESIDUES in the PDB file. If the termini are not the true termini, they should be uncharged. The easiest way to do this is to leave the termini as standard amino acids. Creating a neutral N-terminus requires manipulation of the PDB files created by Reduce, see Section 4a.). In some protein coordinates, the last residue of a peptide chain includes an OXT atom, even though it is not the true terminus. If you know the C-terminus should be uncharged, remove the OXT atom.

Reduce also checks for the proper orientation of Gln and Asn side chains, determines the protonation state of His residues from the local environment, and looks for and corrects steric clashes within the macromolecule.

4a. Protonation of N-terminal amino acids in proteins

In proteins, we call N-terminal residues all those that do not have a covalent bond between the main-chain N atom and a preceding, C=O group, defined in Reduce by a C–N distance between ** and ** Å. N-terminal residues include

1. the true N-terminus of a peptide chain
2. the starting residue of a peptide chain where preceding residues are missing
3. a residue in the middle of a peptide chain when coordinates for the preceding, covalently bonded residue are not in the PDB file, as in a disordered loop.

For true N-termini, we want a -NH_3 group (positively charged); for other N-termini we want a -NH group (neutral), forming a complete residue but with an incomplete covalent shell for the N atom.

Case 1: Residue number 1 is the true N-terminus of the peptide chain. This case requires no user intervention. REDUCE will, by default, add 3 hydrogen atoms to residue 1 of a peptide chain, creating a positively charged -NH_3 group. Three hydrogen atoms will ONLY be added if REDUCE determines that the N atom is not attached to a preceding C=O group (distance criterion). For example, if the protein chain is numbered 1A, 1B, 1, 2,..., REDUCE puts the -NH_3 group on the first residue, 1A. In the next step of “dot2-prep-mol-common”, the presence of the N-terminal hydrogen atoms will be used to determine that this is a positively charged N-terminal residue, and the corresponding partial atomic charges will be assigned.

Case 2: The peptide chain begins with a residue having a residue number other than 1 and the user wants this residue to have a positively charged -NH_3 group at the N-terminus. This case requires editting the user’s customized prepscript so that Reduce will add hydrogen atoms as desired. Edit the call to dot2-prep-mol-common for the appropriate molecule by adding the “Nterm” flag, as indicated in the comment line above the call. The “Nterm” flag is feed into the Reduce command line.
Example: My stationary molecule stat.pdb begins with residue number 82 and this residue should have an -NH_3 group. Within “prepscript”, I edit the line that calls “dot2-prep-mol-common” to process the stationary molecule:

```
dot2-prep-mol-common -r Nterm82 -p stat.pdb -l $reslib -w stationary
```

With this command, all residues with residue numbers less than or equal to 82 will have 3 hydrogen atoms added to their N atom, but only if there is no preceding, covalently bonded C=O in the coordinate file. Warning: If there are multiple chains in the stationary molecule, Reduce will add 3 hydrogen atoms to all the N-terminal residues with residue numbers 82 or less. Reduce does not consider chain IDs.

Case 3: The N-terminal residue is not the true start of the chain, so it should be neutral to avoid introducing an inappropriate positive charge when atomic charges are assigned. The goal is to put a single hydrogen atom on the N atom of this residue. This case requires

- making a customized version of “dot2-prep-mol-common” for the molecule
- editting “prepscript”
- editting the PDB file created by Reduce.

Currently, REDUCE adds no protons to a backbone N atom when there is no preceding C=O AND the residue number is not 1 AND the -Nterm option is not specified. To achieve our goal, we must make Reduce add 3 hydrogen atoms to the backbone N atom, then edit the PDB file made by REDUCE with all hydrogen atoms.

Example. Our starting stationary molecule coordinates, stat.pdb, begin with residue 82 which is preceeded by a disordered N-terminal peptide that is missing in the PDB coordinate file. Therefore, we want residue 82 to have a neutral charge N-terminus. First, copy dot2-prep-mol-common from $DOT_ROOT/bin/share to the coords directory of our project and give it a unique name:

```
cd projdir/coords
cp $DOT_ROOT/bin/share/dot2-prep-mol-common dot2-prep-mol-stat
```

Second, edit “dot2-prep-mol-stat” by adding a “pause” command to allow editting of

```
stat.cen.allh.pdb
```

the PDB file created by REDUCE. The “pause” command should be inserted after Reduce is run (creating the *.allh* file) but before non-polar hydrogens are removed. For our case, the pause command would be:

```
pause ’correct protonation state of residue 82’
```

And this would be inserted in our customized “dot2-prep-mol-stat” to read as follows:

```
grep ERROR $reduce_log > /dev/null
exit_if_matches_found $?  reduce reported an error, $pgm quitting, see $reduce_log
exit_if_file_missing_or_empty $cen_allh_pdb

pause ’correct protonation state of residue 82’
echo Remove non-polar hydrogens
```

Third, edit prepscript to call our customized “dot2-prep-mol-stat”.

```
../dot2-prep-mol-stat -r Nterm82 -p stat.pdb -l $reslib -w stationary
```

Now run prepscript. Prepscript will pause after running Reduce, with the message

```
correct protonation state of residue 82
```

Reduce will have added 3 hydrogen atoms named H1, H2, and H3 to the N atom of residue 82. To make the neutral N-terminus, atoms H1, H2, and H3 of residue 82 need to be replaced by a single H. Edit stat.cen.allh.pdb to remove H2 and H3, and change the name of H1 to H, the standard backbone amide proton name. Make sure the ’H’ and the rest of the fields on the line are in the correct columns. Then, resume prepscript by typing enter (or return).

Case 4: There is a disordered loop in the middle of the protein chain, such that a peptide segment is missing. This creates a residue with no preceding, covalently bonded C=O group in the middle of the peptide chain that should be built as a neutral residue to avoid introducing inappropriate positive charge in the protein. The goal is to have a single hydrogen atom on the N atom for this residue. As for Case 3, this case requires

- Making a customized version of “dot2-prep-mol-common” for the molecule in which a “pause” command is inserted after the “allh” PDB file is made by Reduce.
- Editting “prepscript” to add the appropriate flag for Reduce and call the customized version of “dot2-prep-mol-common”.
- Editting the PDB file (*allh*pdb) created by Reduce when the “pause” command pauses prepscript.

If prepscript is run without these changes, Reduce will not add any hydrogen atoms to the N atom lacking the preceding, covalently bonded C=O group and the total charge for the residue will not an integer (a warning in prepscript) AND the total charge for the peptide chain will not an integer (an error that will cause “prepscript” to halt).

Example 1: Coordinates of my stationary stat.pdb are missing the N-terminal peptide, with the chain beginning at residue 82, and missing the loop segment, residues 100-120. So I want residues 82 and 121 each to have a neutral N-terminus with the N atom having a single proton. As in Case 3, I make a customized copy of "dot2-prep-mol-common" that includes a "pause" statement after the stat.cen.allh.pdb:

```
cd projdir/coords
cp $DOT_ROOT/bin/share/dot2-prep-mol-common dot2-prep-mol-stat
```

inserting the "pause" command

```
pause 'correct protonation state of residues 82 and 121'
```

Then, I edit prepscript to call the customized "dot2-prep-mol-stat".

```
../dot2-prep-mol-stat -r Nterm121 -p stat.pdb -l $reslib -w stationary
```

Reduce will then add 3 hydrogen atoms to the N-terminus of any amino acid with a residue number up to 121 that is not preceded by a C=O group. When prepscript is called, it will pause after the stat.cen.allh.pdb file is created and the user edits both residues 82 and 121, replacing the atom name "H1" with "H " and deleting "H2" and "H3".

Example 2: Coordinates of my stationary stat.pdb begin at residue 82, which is the true N-terminus, and are also missing the loop segment, residues 100-120. So I want residue 82 to have a positively charged N-terminus and residue 121 to have a neutral N-terminus with a single hydrogen atom. I proceed as in Example 1, but when "prepscript" pauses, I only need to edit residue 121, replacing the atom name "H1" with "H " and deleting "H2" and "H3".

4b. Protonation states of His and Cys

Reduce handles protonation of standard states of His and Cys without intervention. Depending on pH and environment, the imidazole ring of His can be protonated on either ND1 or NE2 (neutral) or on both ND1 and NE2 (positively charged). In addition, the two possible orientations of the imidazole ring should be examined (180 degree ring flip), since, at the typical resolution of crystallographic protein structures, the two orientations cannot be distinguished from the electron density alone. REDUCE considers both orientations of the imidazole ring when it determines the most likely protonation state []. For DOT, we suggest making His residues positively charged only when there is a compelling reason for doing so, such as a His residue in which both ND1 and NE2 form hydrogen bonds with carboxylate groups, such as those of Asp and Glu. Reduce takes this conservative approach, usually adding only one H atom to the two N atoms of the imidazole ring. Reduce also recognizes when His is ligated to a metal ion, and adds hydrogen atoms appropriately. The user can use other methods to determine the His protonation state, including algorithms that attempt to determine the pKa. For an isolated, neutral His residue at pH 7, protonation on NE2 is slightly more favorable than protonation on ND1. Some molecular modeling programs, such as Insight (Accelrys) use this as the default protonation state at pH 7. So far, Reduce's method of basing the protonation state on the local environment has given the best docking results with DOT.

Cys can be both free (protonated on SG) or form a disulfide (no hydrogen atom). REDUCE determines the protonation state based on local environment. Reduce also recognizes when Cys is ligated to a metal ion, and adds hydrogen atoms appropriately.

4c. Reduce geometry check: HIS, ASN, and GLN side chains

At the resolution of crystallographic protein structures, the two possible orientations of the His imidazole ring and the side chain amides of Asn and Gln cannot be distinguished. Reduce checks the hydrogen bonding environment of the crystallographic orientation and the 'flipped' position (180 degree rotation), and scores which one is best. Reduce then creates the flipped coordinates, if needed, and adds hydrogen atoms. In the His imidazole ring, the positions of CD2 and ND1 are exchanged and the positions of CE1 and NE2 are exchanged in the flipped conformation. For Asn and Gln, the N and O atoms of the side chain amide are exchanged. In the Reduce mode used by prepscript, the side chains are flipped only when the environment indicates a clear preference for the 'flipped' orientation. We have found that the defaults used by Reduce work well. The user may be interested in using Reduce interactively through the Molprobity web site (**http://molprobity.biochem.duke.edu/**), where the user can view the two orientations in the protein environment and select which should be used.

4d. Reduce geometry check: Steric problems

Reduce uses the hydrogen atoms that it adds to check for steric problems and applies small motions to the side chains to relieve them.

4e. Residues other than standard amino acids

These groups include functionalized amino acids, cofactors, DNA residues, metal ions, etc. REDUCE is likely to do reasonable job, especially the CONECT records from the PDB file are retained. We have tested Reduce on HEM and DNA residues, although with the recent changes in the PDB, the names of residues in DNA and RNA is a mess. Reduce has a utility to add new functional groups (see Reduce documentation). The user may want to try adding H atoms first to new functional groups and then make a customized version of

```
dot2-prep-mol-common
```

in which the call to pdb_dehydrogen), which removes hydrogen atoms in the starting PDB files, is commented out.

```
DOT 2 User Manual CVS $Id: du-prep-polh.tex,v 1.7 2008/04/02 03:52:04 vickie Exp $
```

I.5 Create files with heavy and polar H atoms (polh) and RES names for charge library

The script "dot2-prep-mol-common", called by prepscript, takes the centered PDB files with all hydrogen atoms created by REDUCE (allh files), and selects all the heavy atoms plus the polar hydrogen atoms (polh files), thereby removing all of the nonpolar hydrogen atoms. Polar hydrogen atoms are those attached to N, O, and S. For common variants of standard amino acids, unique residue names are assigned based on the protonation pattern. Currently, prepscript can figure out common variants of His and Cys residues and whether an amino acid is at the N- or C-terminus of a peptide chain. For a starting molecule named mol.pdb, the input file is the centered PDB file with all hydrogen atoms:

```
mol.cen.allh.pdb
```

and the file created is the centered PDB file with polar hydrogen atoms and adjusted residue names:

```
mol.cen.polh.pdb
```

Why? The current DOT energy terms and charge libraries are balanced for using atomic charge sets based on heavy atoms plus polar hydrogen atoms. Therefore, nonpolar hydrogen atoms must be stripped from the coordinate files. Since each variant of a residue has its own charge distribution, each must have a unique residue name that corresponds to its entry in the charge library, so that partial atomic charges can be correctly assigned.

5a. His, Cys, and N-terminal amino acid residues

Prepscript automatically assigns unique residue names to common variants of His and Cys and to charged amino acid residues at the N- or C-terminus of a peptide chain. For His, the variants are recognized by the protonation patterns of the imidazole ring.

HIS or HIE	Protonated on NE2
HID	Protonated on ND1
HIP	Protonated both on ND1 and NE2

For Cys, variants are recognized by the protonation of SG:

CYS	Free cysteine, protonated on SG
CYX	Cysteine that is part of a disulfide, no hydrogen atom added to SG

Charged N- and C-termini of peptide chains are identified by

N-terminus, example ALAN	Presence of atoms 1H, 2H, and 3H
C-terminus, example ALAC	Presence of atom OXT

For example, if an amino acid residue, say Ala, includes the atom types 1H, 2H, and 3H, prepscript will assign it as being a positively charged N-terminal residue with the residue name ALAN.

5b. Charges - should add up to correct formal charge

```
DOT 2 User Manual CVS $Id: du-prep-xyzq.tex,v 1.4 2008/04/02 03:52:04 vickie Exp $
```

I.6 Moving molecule shape (.xyz)

The script “dot2-prep-mol-common”, called by prepscript, creates the “.xyz” file, which contains the centered coordinates of the heavy atoms of the molecule. This DOT input file defines the molecular shape of the moving molecule. Each line of the “.xyz” file has fields X, Y, Z (in Angstroms). Example .xyz file fragment:

```
-24.004 10.540 13.354
14.847 -1.450 26.429
23.264 -49.230 39.562
```

For example, for our starting moving molecule coordinates, mov.pdb, the centered .xyz file is

```
mov.cen.noh.xyz
```

which has the same number of atoms as mov.cen.noh.pdb.

This file is created in dot2-prep-mol-common.

```
pdb_to_xyz mov.cen.noh.pdb > mov.cen.noh.xyz
```

DOT will read the .xyz file and then map each coordinate of the moving molecule (heavy atoms only) onto the nearest grid point.

I.7 Moving molecule partial atomic charges (.xyzq)

The script “dot2-prep-mol-common”, called by prepscript, creates the “.xyzq” file, which contains all heavy atoms and polar hydrogen atoms and their partial atomic charges. This DOT input file defines the charge distribution of the moving molecule. In the “.xyzq” file, each line has fields X, Y, Z (in Angstroms), and partial atomic charge. Example .xyzq file fragment:

```
-24.004 10.540 13.354 -0.9
14.847 -1.450 26.429 0.8
23.264 -49.230 39.562 1.2
```

For example, for our starting moving molecule coordinates, mov.pdb, the centered .xyzq file is

```
mov.cen.polh.xyzq
```

which has the same number of atoms as mov.cen.polh.pdb. This file is created in dot2-prep-mol-common.

```
pdb_to_xyzq mov.cen.polh.pdb > mov.cen.polh.xyzq
```

DOT will read the .xyzq file and then interpolate the partial atomic charges of the moving molecule (heavy and polar hydrogen atoms) over the eight nearest grid points.

I.8 Creating new functional groups

For new functional groups the user will have to edit the output polh file to assign new residue names that identify the functional group uniquely. To do this, the user must

1. Make a customized atomic charge library that contains the new functional group
2. Make a customized version of dot2-prep-mol-common that includes a "pause" command
3. Edit prepscript to call the customized dot2-prep-mol-common

add the pause command (section **) to the prepscript file, so that prepscript pauses while the user edits the polh file. For example, a PDB file contains a Cys that is a Cu ligand. The CYS residue name is in the original PDB file and Reduce has CYS in its library and correctly protonates the residue. Reduce correctly adds no proton to the CYS SG because Reduce identifies the SG-Cu bond. Prepscript automatically identifies this CYS as CYX due to the protonation pattern. However, the user knows that CYS as a Cu ligand should have an overall charge of -1, rather than being neutral as it is in a disulfide. The user creates a new residue entry in the charge library, CYM, with appropriate partial atomic charges. The user then edits the output polh file to change the CYX residue to CYM, so that the correct atomic charges are assigned.

8a. Other functional groups

If there are other functional groups or molecules, such as cofactors, present in the PDB file, the user should

- Check to see if this group is in the charge library.
- Check that the atom and residue names in the library and the PDB file match.
- If needed, create new entry in the charge library with a unique RES name and partial charges and atomic radii.
- Check the total charge on the residue is as expected.
- Check that Reduce adds hydrogen atoms appropriately.
- Check that the correct polar hydrogen atoms appear in the polh file.

Assignment of partial atomic charges can be straightforward from examination of the current charge library. For example, if I had the partial atomic charges assigned for CYM, a Cys ligated to Cu, but I need CYM with a charged C-terminus, I would create CYMC, which added typical charges for the -CO_2 group.

One the other hand, the atomic charges for CYM itself are more difficult and beyond the scope of the DOT software package. For a cofactor, a quantum mechanics program such as Gaussian (**) could be used. Some cofactors, such as HEM, have published sets of point charges, because parametrization of these functional groups has been done for molecular dynamics simulations. Complex functional groups, such as metal clusters, require complex calculations

The DOT team has charge libraries for DNA bases and HEM groups we are happy to share with you, just email us.

```
DOT 2 User Manual CVS $Id: du-prep-gridsize.tex,v 1.7 2008/04/02 03:52:04 vickie Exp S
```

J Determining grid size

Optimally, the moving molecule in any orientation should fit in the grid surrounding the stationary molecule when the two molecules are close. This ensures that the moving molecule, when close to the stationary molecule, will not be influenced by the shape or electrostatic properties of the stationary molecules in adjacent cells. The default grid spacing is currently 1 Å. We have found that a grid spacing of 2 Å is too coarse to give good results with macromolecules and also gives a poor approximation of the electrostatic potential of the stationary molecule. A grid spacing smaller than 1 Å is not computationally efficient for macromolecules. If the user is looking at small molecule docking over a smaller region of space, a finer grid spacing may be needed. The rotation sets provided will give a rotational search for a small molecule comparable to the translational search with small grid spacing. The default is a cubic grid.

J.1 Default method based on molecular diameters

To ensure that the moving molecule sits inside the grid surrounding the stationary molecule, prepscripts calculates

S + $2M$

where S is the largest length for the stationary molecule in X, Y, or Z and M is the largest diameter for the moving molecule. Prepscript then selects the grid size that is closest to, but larger than, this sum.

J.2 Sizes efficient for the calculation.

Grid sizes of 64, 128, 160, 192, 224, and 256 are most efficient for the FFT calculation. The calculation time is proportional to

$NlogN$

where N is the number of grid points. A grid size of 128 takes about 9 times longer than a grid of 64 and a grid size of 160 takes about 2 times longer than a grid of 128.

Larger grids also take longer for the electrostatic potential calculation (using APBS or UHBD), and grids larger than about 200-cubed need to run the electrostatic calculation on computers with more than 2 gigabytes of memory (not just swap space) to finish in a reasonable time (an hour). For a given grid size, APBS takes several-fold less time than UHBD, but has much larger memory requirements. For example, running APBS using a 256-cubed grid needs 3.6 gigabytes whereas a 192-cubed grid needs only 1.6 gigabytes. Finally, when you use the options that prepscript supplies, grid dimensions that are multiples of 32 work best for APBS.

J.3 User selection of grid dimension

The user can override the grid size calculated by prepscript by using the -d parameter in the gen_dot_prepscript command line. For example

```
gen_dot_prepscript -m moving.pdb -s stat.pdb -d 128
```

will create a prepscript with a grid size of 128. For a couple of test cases, we have found that the grid size nearest, but smaller than, the molecule diameter sum above gives almost identical results at a decreased computational cost.

J.4 Scripts, utilities used

```
DOT 2 User Manual CVS $Id: du-prep-stat.tex,v 1.2 2007/11/13 07:07:45 vickie Exp $
```

K Prepscript: Stationary molecule

The shape and electrostatic potentials describing the stationary molecule are more detailed and more time-consuming to calculate than those of the moving molecule. NOTE: It is essential that both potentials are calculated in the same coordinate space! Prepscript tries to ensure this by controlling the input file coordinates that are used. For example, for the starting stationary molecule, stat.pdb, the coordinate file

```
stat.cen.polh.pdb
```

is used to create the electrostatic potential grid, and the coordinate file

```
stat.cen.noh.pdb
```

is used to create the shape potential. Besides necessary DOT utilities, creating the potential files requires the programs MSMS, for creating molecular surfaces, and APBS (or UHBD), for calculating the electrostatic potential.

```
DOT 2 User Manual CVS $Id: du-prep-grid.tex,v 1.9 2008/05/09 04:57:58 mp Exp $
```

K.1 Calculate Electrostatic Potential for Stationary Molecule (dx)

The electrostatic potential grid is currently calculated by APBS. Prepscript creates the parameter (or 'command') file and runs APBS. (Prepscript can use UHBD instead; run

```
prepscript --uhbd
```

) The following parameters for calculating the electrostatic potential are set up in prepscript and can be edited by the user:

pot_ionstr=150	Ionic strength, millimolar
pot_maxits=500	For UHBD only

The APBS calculation can take many minutes to run. The following line in prepscript runs the script "dot2-prep-potgrid-apbs", which lists the default APBS parameters, sets up the APBS parameter file (.in file), sets up the coordinate file read by APBS which contains the atomic coordinates, partial charges, and atomic radii for each atom (heavy and polar hydrogen atoms) in the stationary molecule, and runs APBS, creating a log file and the electrostatic potential grid (.dx file).

Example: For our starting stationary molecule stat.pdb, the coordinates used for the electrostatic calculation are the centered coordinates with polar hydrogen atoms

```
stat.cen.polh.pdb
```

For a grid 128 Å on a side (1 Å spacing) with the calculation done at 150 mM ionic strength, the APBS input files created by "dot2-prep-potgrid-apbs" are the parameter file

```
stat.cen.128.150m.apbs.in
```

and the atomic position/charge/radius file

```
stat.cen.polh.xyzqr
```

Output files are the electrostatic grid

```
stat.cen.128.150m.dx
```

and the log file

```
stat.cen.128.150m.apbs.log
```

Prepscript checks that the total charge calculated by APBS is an integer by looking at the APBS log file. If the total charge in the APBS log file is not an integer, prepscript will stop. Even if the total charge is an integer, it is a good idea to check that the charge is correct, look for "Net charge" in the APBS log file. This charge should be the same as prepscript calculated in the last line of the .xyzq file. Note: As a run-alone program, APBS calculates the electrostatic potential for any set of coordinates, taking X, Y, Z, charge, and radius as input. APBS does no internal checking of whether residues are complete. The wise user will check the expected charge on a molecule given the number of charged side chains (Lys, Arg, Asp, Glu), the charge state of the termini, the protonation state of His (as determined by Reduce), and the charge due to cofactors, and compare that charge with those calculated by APBS and in the .xyzq file.

Prepscript controls the size of the grid used by APBS so that it is compatible with the DOT calculation. The grid dimensions in APBS are the DOT dimensions plus one. For example, for our DOT calculation done on a grid 128 Å on a side with 1 Å grid spacing for stat.pdb, the APBS input file made by prepscript

```
stat.cen.128.150m.apbs.in
```

has parameters

```
grid 1.0 Grid spacing
dime 129 129 129 Grid dimensions in the x, y, and z directions
```

Other parameters for APBS are set to be similar to the UHBD parameters that were used to balance the electrostatic and shape terms in DOT. The user can decide to alter these parameters, but we have found that different parameters influence the magnitude of the electrostatic potential over parts of the grid, particularly near the molecular surface. This could effect the balance of the electrostatic and van der Waals energy terms calculated by DOT.

```
DOT 2 User Manual CVS $Id: du-prep-xyzcrv.tex,v 1.6 2007/11/21 22:19:41 mp Exp $
```

K.2 Stationary Shape Description (xyzcrv)

The shape of the stationary molecule is described by an excluded volume surrounded by a 3 Å favorable layer (the .xyzcrv file). The program MSMS is used to create the surfaces that are used to define these volumes. MSMS rolls a probe sphere (default radius of 1.4 Å) over the van der Waals radii to generate a continuous smooth surface representation of the molecule.

Prepscript invokes DOT utilities to create the needed MSMS surfaces, determine the grid points inside and between the volumes, assign values, and create the stationary molecule shape potential volume file read by DOT.

- Spheres with center, radius, value that will be mapped onto grid by DOT The volume is specified by a list of spheres that DOT reads and fills in in the order they appear in the file.
- Excluded volume - all grid pts inside molecular surface The shape potential is based on volumes inside and between molecular surfaces made with the MSMS program. Two surfaces are made: 1) the solvent-excluded (‘molecular’ or Connolly) surface, and 2) a molecular surface formed using atoms with their van der Waals radii expanded by 3 Angstrom. As above, only the nonhydrogen atoms of the coordinates are used. All grid points within these two surfaces are determined; those between the two surfaces are assigned a favorable value, those inside the solvent-excluded surface are assigned an unfavorable value. The resulting list of grid points and values is the xyzcrv file read by DOT to build the shape potential grid. These files typically have 50,000 to 100,000 lines.
- Favorable volume - all grid pts between molecular surface and surface made with 3A extended atomic radii.
- Additional atom-related properties can be added.

The molecular surface (solvent-excluded or Connolly surface) is used to define the excluded volume of the molecule, with all grid points inside this volume defined as “forbidden” (F). An expanded molecular surface is made by adding 3 Å to the radius of each atom. Grid points inside this surface, but outside the solvent-excluded surface, are defined as “attractive” (A). NOTE: All surfaces are made using the heavy atoms only, no hydrogen atom positions are included in the calculation.

Example: For our starting stationary molecule stat.pdb, the input file for the molecular surfaces is:

```
stat.cen.noh.pdb
```

The call to generate the .xyzcrv file in prepscript is:

```
gen_xyzcrvs ccp.cen.noh.pdb $xyzdim $xyzdim $xyzdim $xyzstep \
2>&1 tee gen_xyzcrvs.log
```

where $xyzdim are the x, y, and z dimensions of the grid and $xyzstep is the grid spacing (prepscript assumes a default of 1 Å).

The resulting .xyzcrv file, in our example

```
stat.cen.noh.xyzcrv
```

is a free-format text file, each line of which has fields X, Y, Z, Action Code, radius, and optional value to be filled into the sphere with that center and radius.

1. X, Y, and Z are the coordinates for the sphere center, in Angstroms, in the centered stationary molecule's coordinate system.
2. The Action code is one letter:
   - F for 'forbidden' (the excluded interior volume, replaces any previous values in the sphere's volume)
   - A for 'attractive' (favorable van der Waals, does not replace forbidden region),
   - R for 'replace with attractive' (favorable van der Waals, regardless of previous value)S for 'sum' (replace with sum of previous value and specified value, does not replace forbidden region)
3. The radius of the sphere in Angstroms. If smaller than the grid spacing, only the single nearest grid point will be filled in.
4. An optional value can be used. Normally this is omitted because it is determined by the Action Code; see dotio.c for details and dotio.h for the actual numeric values. Currently, these are 1 for attractive (A) and 1000 for forbidden (F).

Example .xyzcrv file fragment:

```
-24 1 13 A 0.1
-24 2 -4 A 0.1
-24 2 -3 A 0.1
-24 2 -2 A 0.1
-24 2 5 A 0.1
17 2 -2 F 0.1
17 2 -1 F 0.1
17 2 0 F 0.1
17 2 1 F 0.1
17 3 -6 F 0.1
```

In this typical case, the spheres all have a radius of 0.1 Å, therefore, for the default grid spacing of 1 Å, include only a single grid point. These are the forbidden grid points that were determined to lie inside the solvent-excluded surface and the attractive grid points that were determined to lie between the solvent-excluded and expanded surfaces.

The .xyzcrv file typically has 50,000 to 100,000 lines, which are all the grid points assigned as forbidden or attractive. The grid points that are not assigned an Action Code or value in the .xyzcrv file have the value 0.

The construction of the .xyzcrv file makes it easy to add additional atom-related properties.

Example, if it is known that a region of the stationary molecule is not available for the incoming molecule, the automatically assigned attractive layer in this region can be overwritten by:

- Selecting the atoms or secondary structure elements that represent this surface.
- Creating spheres of forbidden area centered at these atoms.
- Appending these spheres to the .xyzcrv file created by prepscript.

K.3 Electrostatic Clamping Values

The electrostatic clamping values are calculated by prepscript and inserted into the parameter file it generates.

Prepscript instructs DOT to perform a technique known as electrostatic clamping; the interested reader may consult Roberts et al. [**?**] for a thorough explanation of the rationale. We use the van der Waals volume file for the steric part of the docking problem. However, there are well–documented problems with using the electrostatic potential at the van der Waals surface. We also define a solvent accessible surface which is an expansion of the first based on a solvent radius of 1.4 Angstroms defined in genxyzcrvs. We will calculate the electrostatic potential at the solvent excluded surface. The maximum range at this surface are then used as the clamp limits fo the potential at the van der Waals surface. This has the effect of smearing out unrealistically concentrated charge at the surface.

```
DOT 2 User Manual CVS $Id: du-prep-parmfile.tex,v 1.4 2008/05/09 04:57:58 mp Exp $
```

L DOT Parameter File (.parm)

Once all the molecule input files have been generated successfully as indicated by the presence of the files above and any messages displayed, prepscript then creates sample DOT '.parm' files that specify the DOT calculation to be performed.

The DOT distribution includes a template for the DOT 2.0 parameter file in **$DOT_ROOT/data/dot_parm_template**. The default values are used unless they are set explicitly in prepscript. The grid dimensions, grid spacings, the location of the four structure files, and clamping range are set by prepscript. When proposed docked structures penetrate this surface the molecules may be thought to bump into each other. In fact, to allow for flexibility not captured by our rigid model we may allow a specified number of "bumps" in our results. This number will be specified in the dot.parm file.

```
DOT 2 User Manual CVS $Id: du-prep-files.tex,v 1.7 2007/11/21 22:19:41 mp Exp $
```

(empty du-prep-files.tex file)

```
DOT 2 User Manual CVS $Id: du-prep-checks.tex,v 1.3 2007/08/16 06:30:11 mp Exp $
```

M Prepscript checks throughout processing

Although, at this point the user has most assuredly provided appropriate input files, a little error checking couldn't hurt. Prepscript will check for the existence of wate and a particular, but common, non-polar hydrogen. Also, since the structure should contain the polar hydrogens, we also check that the file contains at least some hydrogens. These tests will only find the most common errors in the pdb files.

M.1 Not able to find REDUCE, APBS, MSMS

M.2 Error reported by REDUCE

M.3 Problems removing nonpolar H atoms (striph)

M.4 Molecular description files not made correctly (most common problem is file is empty)

M.5 Non-integral molecular charge

M.6 Large positive or negative ±20) molecular charge

M.7 Grid dimensions could not be calculated or too big.

Since prepscript centers the coordinates with added polar H atoms and places the center of mass at the nearest grid point, the midpoint of these centered files should be off by no more than 1A from 0,0,0. The files without polar H atoms will vary slightly from 0,0,0. Note that if you end up recentering (say discover later that you were missing some atoms, add them, and recenter), ALL of the DOT input files for those recentered coordinates must be remade.

```
 It is very important that the moving molecule is quite close to centered.
How to check the midpoints:
```

```
pdb\_to\_xyz < centered\_pdbfile > | minmax > pdbfile.minmax
```

Compare the first few lines of the centered PDB files, with and without polar H atoms. The x,y,z coordinates of the heavy atoms should be exactly the same.

M.8 APBS not running properly

8a. PDB to XYZQ

Typically, programs that are used to calculate the electrostatic potential around a molecule utilize a PDB file for the molecule and and appropriate residue charge library. Here we assume the use of the included charge library. The residues and atoms are looked up in this library and written to an XYZQ file which contains the coordinates and charges of the moving molecule.

Verfication: For the .xyzq file (moving molecule) check that the .xyzq file (last line) agrees with your calculation. Check there were no error messages in the .xyzq file. Check the beginning of the file, are the coordinates exactly the same as the corresponding PDB file? For this test of prepscript, I would also remove all the comment lines from the .xyzq file (make a temporary file), check the center (just minmax ¡tmpfile¿), which should be exactly the same as the centered PDB file, and check that the 4th column (charge) adds up properly.

For example:

```
awk ’{t+=$4}END{print t}’ tmpfile.xyzq
```

will sum the 4th column of the file tmpfile.xyzq. To remove all the extra (comment lines) in the .xyzq file using vi, try,

```
:%g/^#/d
```

This removes all lines that start with ’#’

M.9 Problem with determining clamping values - uhbdgrd_lookup_xyz

Chapter 5

DOT

```
DOT 2 User Manual CVS $Id: du-dot.tex,v 1.12 2008/05/09 04:57:57 mp Exp $
```

A Sample input file

The easiest way to start DOT is by invoking rundot, a script we provide. You always need a DOT parameter file and the four input files discussed above. If you are running in ‘parallel mode’, on more than one computer, you need a ‘hosts’ file, see later in this section. **rundot** will examine the parameter file and use it to create a new directory (folder) for each run, named according to the date, time, and parameters used. These folders will be placed in the project’s ‘runs’ folder, which will be created if it does not itself already exist.

*** rundot options ****

B Most likely to change parameters - no. of orientations, number of allowed bumps

DOT’s functions are controlled from a parameter file, which DOT reads at the beginning of a run. You will likely use several parameter files during a research project, with different numbers of allowed bumps (atom-atom clashes), fewer or more rotations to try, and saving different numbers of results. In general, start out with no bumps, just a few hundred rotations (or just one rotation), and save only the best 2000 results. After you have examined the output of the initial shake-down run, increase the number of rotations (smaller degree spacing implies more rotations) and save perhaps the best 20000 or more results.

Small moving molecules need fewer rotations to sample adequately their orientation, large molecules need more. Table *** shows the rotation files provided in the DOT distribution. As a rule of thumb, choose a rotation step that is about the same as the 3-D diagonal of your grid, which is $\sqrt{3} \times gridspacing$. For example, if your moving molecule has a radius of 15 angstroms, 1 angstrom grid spacing (1.7 angstrom 3-D diagonal) would subtend 1.7 * 60 degrees (per radian) / 15 = 6 degrees, so the ”deg06.eul” file would be appropriate.

Table ** Rotation files included in DOT distribution

Spacing	Number of	File name	
(degrees)	Orientations	(in $DOT_ROOT/data)	
4	232020	deg04.eul	
4	12869	deg04.near180.eul	Rotations in range 175 to 180 degrees
6	54000	deg06.eul	
8	27000	deg08.eul	
10	14400	deg10.eul	
12	9000	deg12.eul	
20	1800	deg20.eul	
72	60	deg72.eul	
90	24	deg90.eul	
360	1	zero-rotation.eul	

It is best to make your parameter files with names meaningful to you, we find a pattern like “udgugi.deg06.nb10.parm” to work well; indicating that the stationary molecule is udg, the moving molecule is ugi, the rotation spacing is 6 degrees, and the number of allowed bumps is 10. Note that DOT ignores the file name and just looks at the content, however.

We find it easiest to make changes by copying to a new name and editing the copy. For example, prepscript’s parm files save the 2000 best-ranked placements. To save 20000 of them, copy the sample parameter file (that prepscript will have made for you as statmov.deg06.nb0.parm) to a name such as statmov.deg06.nb0.best20000.parm . Edit that copy and change the line
output_how_many_best_values 2000
to
output_how_many_best_values 20000

If your moving molecule is small, you might not need as fine a rotation spacing as the 6 degrees that prepscript’s parm files use. Suppose you decide 10 degrees is sufficient; copy statmov.deg06.nb0.parm to statmov.deg10.nb0.parm and change deg06 to deg10 in the line
rot_file $DOT_ROOT/data/deg06.eul .

To allow ten bumps, change the 0 to 10 in the line
mov_atoms_in_stat_interior_limit 0

Section *** gives details on the many other parameters for DOT but these three are (by far) the ones most commonly changed.

C Molecular description files needed by DOT

The parameter file also names the four necessary input files for DOT:

mov.cen.polh.xyzq electrostatic charges of atoms of moving molecule

stat.cen.polh.*.apbsgrd electrostatic potential of stationary molecule

mov.cen.noh.xyz van der Waals spheres of moving molecule

stat.cen.nolh.xyzcrv shape potential of stationary molecule

D Running DOT

D.1 Single processor mode

You need only a DOT parameter file and the four input files discussed above.
rundot ***parmfile optional comments for log***
These optional comments will be appended to **$projdir/dotruns.txt**

D.2 Multiple processors, such as a workstation farm

You need a DOT parameter file, the four input files discussed above, and a “hosts” file that lists the names of the computers to run DOT on, one per line.
rundot ***parmfile*** **–hosts** ***dot.hosts optional comments for log***

If you have a ‘dual processor’ (or ‘quad’, or more) computer, like a recent Macintosh “Intel Core Duo” or a new Sun, you can use all the processing cores by making a hosts file that has your computer name (type ”hostname” to see what it is) in it multiple times: for example, if your computer is named ”dopey”, make a hosts file with two lines like this (but use dopey.local on a Macintosh):

```
dopey
dopey
```

Try four copies on a quad processor, and so on. [The MPICH manual tells other ways to do this, but this is simple and works quite well for DOT's loose parallelism.]

You can mix different computer platforms in your hosts file as long as DOT has been compiled and installed for that computer type, the DOT 2.0 release includes Intel Linux, Power-PC and Intel Mac OS X (Darwin), and Sun SPARC Solaris. For example, if your local network has an Intel Linux named "dopey", a Power-PC Macintosh named "happy", and a Sun Solaris named "bashful", just make a 3-line text file named dot.hosts in your DOT project directory containing:

```
dopey
happy
bashful
```

We have had best results putting first the name of the computer you're starting DOT on, but please let us know how this works for you. (Once you get going and you have many machines listed, you can comment out ones you don't want for a particular run by putting a "#" in front of those lines.)

Be sure the $DOT_ROOT directory as well as your DOT project directory are mounted on all the hosts. We use NFS (Sun's network file system) for this but any local area shared file system should work (let us know of problems or successes please).

Finally, you must be able to run programs on all of these hosts from the start-up host without having to type your password. Be sure of this by typing (for example):
ssh happy date ; ssh bashful date

If you cannot do this using ssh, try again using rsh: rsh happy date ; rsh bashful date
If rsh works but ssh does not, set the environment variable P4_RSHCOMMAND to rsh (sh/bash: export P4_RSHCOMMAND=rsh; csh/tcsh:setenv P4_RSHCOMMAND rsh) before trying to run DOT in parallel.

You can learn about how to set up your ssh and rsh environments so they do not require passwords, see the MPICH web documentation, especially http://www-unix.mcs.anl.gov/mpi/mpich1/docs/mpichman-chp4/node127.htm#Node127 and http://www-unix.mcs.anl.gov/mpi/mpich1/docs/mpichman-chp4/node128.htm#Node128 See also the discussion in the "Installation" chapter of this manual.

If you are unable to get either "ssh" or "rsh" to work without passwords, see the MPICH "chp4" manual to set up a secure server. The MPICH manual also describes several fancier and more flexible ways to run DOT in parallel; in particular, you can set up (once) a "machines" file then run DOT using "mpirun -np ...", see http://www-unix.mcs.anl.gov/mpi/mpich1/docs/mpichman-chp4/mpichman-chp4.htm You can also run DOT on a Beowulf-style computer cluster or on a massively parallel supercomputer such as an IBM BlueGene; please email the DOT help line (dot-help@sdsc.edu) for help.

D.3 Multiple processors, single supercomputer

The procedure varies among the supercomputer centers. CCMS has experience with BlueGene and we are eager to help you but cannot offer written instructions yet.

E DOT actions on input files

E.1 Stationary molecule shape potential (.xyzcrv)

1. Map onto grid
2. Forbidden - default value 1000
3. Why another Forbidden value might be needed
4. Forbidden - needs to be larger than max number of M atoms in favorable layer
5. Attractive - default value 1

E.2 Moving molecule shape (.xyz)

1. Coordinates of moving mol. heavy atoms are mapped onto nearest grid point by DOT
2. Why not interpolated? Problems with excluded stat. mol volume

E.3 Stationary molecule electrostatic potential (.apbsgrd)

1. Electrostatic clamping - what and why
2. Interior zeroing - what and why

E.4 Moving molecule partial atomic charges (.xyzq)

1. Atomic charges interpolated onto nearest 8 grid points by DOT
2. Distributes M atomic charge better over grid - slightly better answers

F What DOT computes

F.1 van der Waals energy

- Count of number of M heavy atoms in favorable layer surrounding S
- Count times -0.1 kcal/mol to give interaction energy
- S favorable layer 3 A thick.

F.2 Electrostatic energy

- Stationary molecule: elec. pot. grid made using Poisson-Boltzmann methods
- Moving molecule: partial atomic charges
- Intermolecular elec. energy calculated as mov. mol partial charges in elec. pot. of stationary mol.
- Elec. pot. grid is created by APBS
- Charge library used for both S and M has charges for heavy atoms plus polar H model
- Library currently has amino acids, plus charged N and C termini for all.
- Library currently has DNA nucleic acids, plus 5 and 3 termini for all
- APBS command file generated by prepscript

G DOT parameters

dot2.parm DOT parameter file DOT's functions are controlled from a parameter file, which DOT reads at the beginning of a run.

G.1 Sample Parameter files

G.2 Parameter Reference (Table)

Dot 2 parameters - $Id : dot2 - parameters.tex, v1.42007/12/0701 : 17 : 03mpExp$

Commonly-used and required parameters are in **bold**.

parameter	type	default	notes
dot_version	string	none	Parameter file format identifier (ignored)
fftw_plan	string	”patient”	Experts only, also ”estimate”, ”measure”
do_logNmerge	boolean	false	Use for massively parallel runs (BlueGene)
fussy	boolean	true	Halt if likely mistakes are detected
dot_grid_size	int	128	X,Y,Z size of the grid in grid points
sizex	int	128	X size of the grid in grid points
sizey	int	128	Y size of the grid in grid points
sizez	int	128	Z size of the grid in grid points
dot_grid_step	float	1.	(Angstroms)
stat_pot_file	filename	””	APBS, UHBD, or DelPhi grid file
stat_vdw_file	filename	””	xyzcrv shape file
stat_pdb_file	filename	””	pdb file - passed to evaluate_dot_run
mov_charge_file	filename	””	xyzq file
mov_vdw_file	filename	””	xyz file, default is mov_charge_file data
mov_pdb_file	filename	””	pdb file - passed to evaluate_dot_run
rot_file	filename	””	Euler angle file
out_base	string	””	base name for all output files
stat_pot_clamp_high	float	+6.	Max electrostatic potential value used
stat_pot_clamp_low	float	-6.	Min electrostatic potential value used
stat_pot_interior_scale	float	0.	Experts only, interior electrostatic scaling
stat_pot_interior_zero	boolean	true	Synonym for stat_pot_interior_scale 0.0
stat_vdw_interior	float	1000.	Experts only, sets ”forbidden” value
vdw_weight	float	-0.1	van Der Waals energy term weighting
electrostatic_weight	float	1.0	electrostatic energy term weighting
mov_atoms_in_stat_interior_limit	integer	0	how many bumps to allow
do_partition_sum	boolean	false	compute partition sum (”free energy”)
partition_sum_temp	float	300.0	Kelvin
do_energy	boolean	false	Retain grid of best energy per grid cell
do_bgrids	boolean	false	Automatically implies do_energy
do_histograms	boolean	false	
output_log_detail	integer	1	Values 4 to 8 are resonable
output_how_many_best_values	integer	200	how many globally-best energies to retain
output_how_many_per_gridcell_best_values	integer	0	Also called ”saved_best_values”
output_how_many_partition_sum_best_values	integer	unlimited	Effective only if do_partition_sum is true
output_all_Ethreshold	float	-1000.	Experts only, report every energy $<$ this

Note: Dot accepts ”true”, ”yes”, ”on”, or ”enabled” to set booleans true; Dot interprets anything else as false.

Note: Dot source code generally uses variable names different from these parms.

G.3 Parameter Guide

Parameter ‘mov_atoms_in_stat_interior_limit’. Sets the maximum number of moving molecule atoms allowed to penetrate the stationary molecule’s excluded volume. Informally called the number of bumps.

Parameter ‘vdw_weight’. Sets the weighting in the composite energy of the van der Waals count (number of moving molecule atoms inside the stationary molecule’s favorable layer). The default is -0.1, so that each moving molecule atom in the favorable layer contributes -0.1 kcal/mol to the van der Waals energy. We have found that this default provides a good balance with the electrostatic energy term.

Parameter ‘output_how_many_best_values’. Saves the top N solutions, including multiple solutions centered at the same grid point with different orientations. This better reveals clusters that may indicate correct solutions. Not available if DOT was compiled without HEAP option.

Parameter ‘output_all_Ethreshold’. Saves all solutions, including multiple solutions centered at the same grid point with different orientations, with energies more favorable than the specified threshold. Needs to be used carefully to avoid outputting a very large number of solutions.

Parameter ‘output_how_many_per_gridcell_best_values’. Saves the top N placements taken from the list of the best ranked solution at each grid point. In other words, if two solutions centered at the same grid point but with different orientations were highly ranked, only the one with the best energy was included in the list. Not available if DOT was compiled without OLDSORT option.

Chapter 6

Evaluation

```
DOT 2 User Manual CVS $Id: du-evalrun.tex,v 1.7 2008/02/14 04:35:38 vickie Exp $
```

A After DOT run:

$projdir>cd runs
$projdir/runs> ls
$projdir/runs>(date.time).statmov.nb=(# of bumps).(rotations)
There will be a list of DOT results directory for each simulation run each will contain the following files. **runmpidot**
hosts
statmov.dot2.parm

date.run
kill_remote_processes.sh
statmov.*.log
statmov.topN.e6d
statmov.top200.ace.d=6.0.eval.e6d ***
statmov.all.e6d
log

B evaluate_dot_run

If DOT finishes without reporting errors, the rundot script automatically runs a second script, evaluate_dot_run, to create PDB files for an initial evaluation. You can also run evaluate_dot_run later, as many times as you want, by giving it the name of the directory to do the evaluation in (such as runs/20070927.2359.1jcg2rsa.deg06.nb0)

The evaluate_dot_run script provided in $DOT_ROOT/bin/share will create new directories in the runs/... directory named top30pdb, top30ace6-pdb, and top30ace9-pdb. The top 30 raw DOT moving molecules will be in top30pdb named mov.0001.pdb, mov.0002.pdb, etc., where 'mov' is the name you gave to your moving molecule. These are the top 30 as reported by DOT's built-in electrostatic + quick van der Waals term. The evaluate_dot_run will also score the top 2000 DOT placements using the sum of the DOT electrostatic energy and an empirical "ACE" term based on atom types [**?**], with both a 6 Angstrom and 9 Angstrom cutoff. The ACE-rescored top 30 moving molecules will be in top30aceDDD-pdb named mov.NNNN.pdb, where NNNN is their original DOT ranking and DDD is the cutoff distance (6 or 9 here). The evaluate_dot_run script also makes, in these two directories, a combo file containing the C-alpha backbone atoms only, separated by TER records, named top30.ca.pdb and top30ace.ca.pdb. (The 6 Angstrom cutoff might be better for evaluating dockings of bound structures or those with little conformational change, and the 9 Angstrom cutoff better for systems showing more change upon binding [**?**])

You must examine and compare these against the *centered* stationary molecule, not the original one you supplied to prepscript. You will find this file in your coords/stat/stat.cen.noh.pdb (where 'stat' is the name you gave to your stationary molecule). The rundot script also makes a copy of it, by that name, in the runs/... directory at the conclusion of each successful DOT run.

You eventually will want to customize evaluate_dot_run. To do this, copy it from $DOT_ROOT/bin/share into your project directory (cp $DOT_ROOT/bin/share/evaluate_dot_run .), make it executable (chmod +x evaluate_dot_run), and modify it as you wish. When rundot finds an evaluate_dot_run there, rundot will run it in preference to the default version in $DOT_ROOT/bin/share (or, in fact, in preference to any other found in your $path).

C Future: files to evaluate DOT run

D log files

E e6d output files

name.topNNN.e6d

F Quick comparison of center and orientation

Distance of the center and orientation from a reference value - useful for comparisons with the correct solution or a preferred solution. Needs only the DOT output file of solutions (translations and orientations) not the full coordinates.

G Creating PDB files

pdbgen mov.cen.noh.pdb shortname [file.e6d]

H RMSD values between PDB files

RMSD differences among a set of possible solutions. Useful for identifying similar solutions among the top few solutions. Requires making the PDB files, with at least C$alpha$ positions.

I Bump-checking PDB files

Requires making the PDB files.

J Residue-residue interactions

Useful for comparison with a known solution or a preferred solution. The method used to evaluate results in the CAPRI competition. Requires making the PDB files.

K Re-ranking by ACE scores

Example: re-rank the top 20,000 DOT results by sum of ACE potential (using default options) and DOT electrostatics, then generate PDB files of the top 20:

```
    e6d_first 20000 statmov.top20000.e6d \
| *ace* ../../coords/stat/stat.cen.pdb ../../coords/mov/mov.cen.pdb \
| e6d_sortby E_elec+ACE6 \
| e6d_first 20 \
| pdbgen mov.cen.pdb mov
```

L Distance filtering

A distance filter is useful in applying known biochemical data, such as that a specific residue or set of residues lie in the interface.

The basic tool is the DOT utility dotxyzfilter, whose arguments are a distance, a count, a stationary xyz file, a moving molecule xyz file, and an input e6d file. It passes through (ie, filters) any e6d placements for which at least count stationary atoms are within the specified distance of one or more moving molecule atoms.

Here is an example script that finds any placements where moving residue 228’s CZ atom is within 6 Angstroms of stationary residue 96’s CD atom. (Note all these examples assume they are run in the directory that holds your results, such as runs/20080704.statmov.nb0)

```
#! /bin/bash
statmol=stat.cen.noh.pdb
statres=96
statatom=CD
movmol=mov.cen.noh.pdb
movres=228
movatom=CZ

# create .xyz files
# working file name, eg stat.228.CZ.xyz
statxyz=${statmol%.*}.$statres.$statatom.xyz
pdb_resnum $statres $statmol | pdb_atomtype $statatom | pdb_to_xyz > $statxyz

movxyz=${movmol%.*}.$movres.$movatom.xyz # working name
pdb_resnum $movres $movmol | pdb_atomtype $movatom | pdb_to_xyz > $movxyz

# run filter, input from statmov.top2000.e6d, output to statmov.filter.e6d
# Filter will pass any e6d placements that have
# one or more of the specified moving molecule atoms
# within 6 Angstroms of any of the specified stationary molecule atoms
count=1
distmax=6
dotxyzfilter $distmax $count  $statxyz $movxyz < statmov.top2000.e6d > statmov.filter.e

# generate PDB files of top 30 that pass filter
e6d_first 30 statmov.filter.e6d | pdbgen $movmol filter
```

Here is a fancier example script that finds any placements where 5 or more C-alpha atoms in stationary residues 25 to 33 are within 6 Angstroms of any atom in moving residues 228, 240, or 293. Without the -r flag, it would report the “reverse”: any placement where 5 or more atoms in moving residues 228 etc were within 6 Angstroms of any of the C-alpha atoms in stationary residues 25 to 33.

```
#! /bin/bash
statmol=stat.cen.noh.pdb
movmol=mov.cen.noh.pdb

# create xyz files
statxyz=${statmol%.*}.filter.xyz # working name
pdb_resrange 25 33 $statmol | pdb_ca | pdb_to_xyz > $statxyz

movxyz=${movmol%.*}.filter.xyz # working name
pdb_resnum ’228 240 293’ $movmol | pdb_to_xyz > $movxyz

# run filter, input from mov.top2000.e6d, output to mov.filter.e6d
# filter will pass any e6d placements with
# any five or more stationary atoms within 6 Angstoms of any moving atom
count=5
distmax=6
dotxyzfilter -r $distmax $count  $statxyz $movxyz < statmov.top2000.e6d > statmov.filte

# generate PDB files of top 30 that pass filter
e6d_first 30 statmov.filter.e6d | pdbgen $movmol filter
```

The output of one filter can be piped into another since each writes to standard output and reads from standard input. You can also concatenate the output from filters, however, the incoming e6d files must have the same fields (a limitation of the e6d file format).

The dotxyzfilter utility has options to count the moving instead of the stationary atoms, or to report only boolean results; run

```
dotxyxfilter --help
```

M Creating AVS etc. visualization input files

Chapter 7

DOT Utility Programs

```
DOT 2 User Manual CVS $Id: du-utils.tex,v 1.2 2008/05/09 04:57:58 mp Exp $
```

A Introduction

The DOT distribution includes a set of programs intended to help set up and evaluate DOT runs. About half of these are scripts (written in a blend of csh, bash, and awk), and about half are compiled programs (written in C or C++) The scripts are installed in $DOT_ROOT/bin/share, and the programs in $DOT_ROOT/bin/$ARCHOSV, where ARCHOSV is one of the supported platforms for DOT such as sun4SunOS5. As long as your $PATH includes both of these, you will not have to worry about which is which, just type the name (or write it into your own scripts) and it will work.

Most of these programs were written by the CCMS team, but we draw your attention to ‘reduce’, which is from the laboratory of David and Jane Richardson at Duke University, written by JṀichael Word. We encourage you to cite their work in any publications that result from your use of DOT, along with the work of the APBS, MSMS, FFTW, and MPICH teams (see ‘Acknowledgements’ section of this manual).

The following is an admittedly telegraphic rundown of these programs, for most of them you can type their name followed by “–help”. The scripts often have more information in them, which you can look at with any editor, and the source for the compiled programs will be found in the DOT installation directory $DOT_ROOT/src/util.

ace-desolvation-oo (compiled program) computes desolvation values

acevalues-oo (compiled program)

dotxyzfilter (compiled program) evaluation distance filter

bgrid_info (compiled program) inspects (seldom used) DOT binary grid

bgrid_minmax (compiled program) inspects (seldom used) DOT binary grid

bgrid_sort (compiled program) sorts (seldom used) DOT binary grid

convert-uhbd-grid_to_bgrid (compiled program) creates (seldom used) DOT binary grid

bgrid_bestenergy (compiled program) inspects (seldom used) DOT binary grid

analyze-triangles (compiled program) checks for degenerate triangles, used by prepscript

compare-edges (compiled program) checks triangle list edges, for data debugging

compare-faces (compiled program) checks triangle list faces, for data debugging

compare-verts (compiled program) checks triangle list vertices, for data debugging

create-triangles (compiled program) creates triangle list from MSMS output

expand-triangles (compiled program) enlarges triangles by normals **?

matrix_to_eul (compiled program) converts rotation matrices to DOT euler angles

fill-double-hull (compiled program) fills region between polyhedra

fill-hull (compiled program) fill region within polyhedron, used by prepscript

print-half-edges (compiled program) **?***

6dtoxfm (compiled program) converts xyz-euler to 4x3 matrix

e6d_closeness (compiled program) reports angle and translation between xyz-euler values and specified target. See also e6d_closest and e6d_select_by_dist_angle.

e6d_closest (compiled program) finds xyz-euler values within specified tolerances of a specified target. See also e6d_closeness and e6d_select_by_dist_angle.

e6dexpand (compiled program) fills cubical grid from e6d file, for visualization

orient_survey (compiled program) checks euler files for completeness and non-redundancy

ACE-script-oodot2 (script) runs ACE evaluation with specified options

Analyze-MSMS-script (script) **??**

Expand-tri-script (script) **??**

Fill-double-hull-script (script) runs fill-double-hull with specified options

Fill-hull-script (script) runs fill-hull with specified options

MSMS-exp-script (script) **??**

acenames (script) converts PDB atom names to internal ACE codes

apbsgrd_lookup_xyz (script) finds values in an apbs .dx grid, used by prepscript

archosv (script) reports what computer platform it is run on, used by prepscript

bgrid_to_avsfield (script) converts (seldom used) DOT binary grid for AAVS visualiazation program

create_host_file (script) makes an MPICH p4pg file from simplified input, used by prepscript

create_p4pg_entry (script) makes an MPICH p4pg entry for a specified host , used by prepscript

dot (script) currently out-of-date script to run DOT on supercomputers

dot0matrix (script) converts obsolete DOT0 euler angles to rotation matrix

dotmatrix (script) converts current DOT 2 euler angles to rotation matrix

dot_pdb_e6d_eval_rmsd (script) computes RMSD in angstroms between a target molecule and a moving molecule as positioned by xyz-euler values.

dotpause (script) puts a DOT run to sleep on the user’s workstation

dotresume (script) resumes a dotpause-d DOT run

dot2-prep-gridsize (script) computes size of grid for a run, used by prepscript

dot2-prep-mol-common (script) computes files needed for both moving and stationary molecules, used by prepscript

dot2-prep-potgrid-apbs (script) computes electrostatic potential grid using APBS, used by prepscript

dot2-prep-potgrid-uhbd (script) computes electrostatic potential grid using UHBD, used by prepscript

dot2.setup.bash (script) sets user's execution path to include DOT programs (bash shell version)

dot2.setup.csh (script) sets user's execution path to include DOT programs (C-shell version)

e6d_append_expression (script) adds metadata field to e6d header block

e6d_first (script) selects first N entries from an e6d file

e6d_nonsimilar (script) quickly eliminates similar placements from an e6d file

e6d_select_by (script) selects entries that match a criterion from an e6d file

e6d_select_by_dist_angle (script) quickly selects placements from an e6d file that are within a distance and angular tolerance from a specified target. See also e6d_closeness and e6d_closest.

e6d_sort_by (script) sorts an e6d file by a specified field. Intended to be used in a pipeline followed by e6d_first.

evaluate_dot_run (script) basic post-DOT-run evaluation, creates PDB files of moving molecule as placed by DOT

expand_environment_variables (script) expands $VAR if VAR is an environment variable, used by rundot

eul_to_matrix (script) converts DOT euler angle to rotation matrix

gen_apbs_com (script) generates APBS input command file from template, used by prepscript

gen_uhbd_com (script) generates UHBD input command file from template, formerly used by prepscript

gen_dot_parm (script) generates a DOT parameter file from a template in $DOT_ROOT/data

gen_dot_prepscript (script) generates file 'prepscript' customized to user's moving and stationary molecule names

gen_xyzcrvs (script) makes stationary molecule shape description file, used by prepscript

gentestuhbdgrd (script) creates dummy UHBD grids for software testing

hostlist_to_mpichhosts (script) converts list of computers into form needed by MPICH, used by prepscript.

minmax (script) reports minimum and maximum values in a file, field by field

minmaxmean (script) reports minimum, maximum, and mean values in a file, field by field

pdb_atom (script) PDB file filter that passes only atoms, not headers

pdb_atomhetatm (script) PDB file filter that passes only atoms and hetatomss, not headers

pdb_ca (script) PDB file filter that passes only C-alpha atoms, used by evaluate_dot_run

pdb_cat_with_ter (script) Concatenates specified PDB files, inserting a TER (chain termination) record in between each, used by evaluate_dot_run

pdb_dealtloc (script) PDB file filter that passes only the first of any 'alternate locations' for an atom, used by prepscript.

pdb_dehydrogen (script) PDB file filter that removes all hydrogen atoms, used by prepscript.

pdb_dewater (script) PDB file filter that removes all water molecules, used by prepscript.

pdb_make_centered (script) centers a PDB file by geometric bounding box, used by prepscript.

pdb_rename_res_by_hydrogens (script) renames residues in a PDB file accoring to their polar hydrogen pattern, used by prepscript.

pdb_replace_selenium (script) PDB file filter that renames selenium atoms to sulfurs.

pdb_rmsd_matrix (script) prints square matrix of RMSD values between all pairs in a specified set of PDB files, which must have the same number of atoms in the same order. Evaluation tool.

pdb_rot (script) rotates a PDB file by specified rotation matrix about the origin.

pdb_rottrans (script) does pdb_rot followed by pdb_trans, q.v.

pdb_to_boxcenter (script) computes bounding box of PDB file

pdb_to_acenames (script) converts PDB atom names to ACE internal codes

pdb_to_dot (script) **??**

pdb_to_uhbd (script) renames PDB (pre-version-3) hydrogens to form acceptable to UHBD (essentially AMBER names), also clears chain-id column.

pdb_to_vol (script) **??**

pdb_to_xyz (script) prints the x,y,z coordinates of each atom in a PDB file, used by prepscript for the moving molecule

pdb_to_xyzatomres (script) prints the x,y,z coordinates, atom types, and residue types of each atom in a PDB file.

pdb_to_xyzcrv (script) computes the DOT stationary molecule shape description, including forbidden interior and attractive vdw layer, used by prepscript.

pdb_to_xyzq (script) prints the x,y,z coordinate, charge [and optionally, radius] of each atom in a PDB file. The values come from the AMBER-style 'rlb' file in $DOT_ROOT/data/uhbd.amber84.prot.rlb unless a different library is specified.

pdb_to_xyzqr (script) prints the x,y,z coordinate, charge, and radius of each atom in a PDB file. The charges come from the AMBER-style 'rlb' file in $DOT_ROOT/data/uhbd.amber84.prot.rlb unless a different library is specified. The radii come from the MSMS-style file in $DOT_ROOT/data/atmtypenumbers. Not currently used in DOT.

pdb_to_xyzr (script) prints the x,y,z coordinate and radius of each atom in a PDB file. The radii come from the MSMS-style file in $DOT_ROOT/data/atmtypenumbers. Used by prepscript for the MSMS calculations of the stationary molecule volume.

pdb_trans (script) translates a PDB file by specified translation vector, used by prepscript. See also pdb_trans and pdb_rottrans.

pdbchecksubunit (script) **??**

pdbdiameter (script) reports largest radial dimension of a PDB file, used by prepscript.

pdbfromdot (script) **??** old version of pdbgen?

pdbgen (script) makes a moving-molecule PDB file for each placement in an e6d file, assigning each a name derived from its original ranking in the DOT run that made the e6d file.

rundot (script) user-level script to run DOT in an automatically created new subdirectory, with logging of the run including user commentary.

striph (script) PDB filter that removes non-polar hydrogens, retaining polar hydrogens (as defined in $DOT_ROOT/data/uhbd.amber84.prot.rlb unless a different library is specified.)

uhbdgrd_limit (script) replaces out-of-bounds values in a UHBD electrostatic potential grid.

uhbdgrd_lookup_xyz (script) finds values in a uhbd .grd grid, formerly used by prepscript

uhbdlog_to_xyzq (script) reports atom-by-atom charges after UHBD program has looked them up in its library. Useful for debugging AMBER-style libraries.

vol_to_xyzcrv (script) **??**

xyzq_check_ok (script) verifies that total charge is integral and within reasonable limits, used by prepscript.

xyzq_xyzr_to_xyzqrxml (script) merges separately-computed charge and radius files into input suitable for apbs program. Not currently used by DOT.

xyzqr_to_xyzqrxml (script) converts x, y, z, charge, radius file into input suitable for apbs program.

Chapter 8

DOT Installation

DOT 2 User Manual CVS $Id: du-installdot.tex,v 1.23 2008/05/09 05:27:44 mp Exp $

A Introduction

Welcome to DOT! The following instructions provide a basic overview of a simple DOT installation on your system.

The DOT distribution consists of a single tarred and zipped file containing pre-compiled binaries and two external libraries for your particular platform, as well as shell scripts, two of three needed external programs, user documentation, and the DOT2 license.

We currently offer binaries for the Red Hat Intel Linux platform as well as command line binaries for Mac OS X (both Intel and PowerPC processors), and for SPARC Solaris 8 (Sun OS 5). For all other systems you will need to compile and install from source. Please feel free to contact the Dot help line, dot-help@sdsc.edu, regarding any issues or requests.

B DOT Installation Quick Start Guide

In the instructions that follow we are assuming you will be using the bash shell.

You may install DOT in your home directory or, if you have root access to your machine, you may install DOT in /usr/local/bin or an equally appropriate location.

We recommend you unpack the distribution in a directory that is mounted on all computer platforms on which you expect your users to run DOT, such as a globally exported directory on an NSF file server. If you do this, your users will be able to run DOT on the platforms included as binaries in our distribution with very little more installation.

After unpacking the distribution, you will need to set the DOT_ROOT environment variable to the directory of the unpacked distribution. In the following example, we install DOT into a directory off our home directory and rename it dot2.0:

```
tar xfz DOT2.0_beta0.tar.Z
mv DOT2.0_beta0 dot2.0
export DOT_ROOT=¨$HOMEdot2.0¨
cd $DOT_ROOT
```

Please note that if you are going to compile from source, see the configure/make instructions at the end of this chapter.

In any case, you will still need to download the MSMS program and (if you want to run DOT in parallel on multiple CPUs) to configure and make MPICH: see below for instructions.

C What is in the DOT distribution

The distribution contains the DOT executables for both single and multi-processor runs, shell scripts necessary to generate DOT input and analyze DOT output, two pre-compiled external libraries:

1. the Message Passing Interface library (MPICH, version 1.2.7p1 http://www.mpich.org) needed for multi-processor (parallel) runs.

2. the Fast Fourier Transform library (FFTW, version 3.1.2 http://www.fftw.org) needed for all DOT runs.

and two of the three external programs required to generate DOT input:

1. APBS, 1.0.0, 21 April 2008, from http://sourceforge.net/projects/apbs, supplied with the DOT distribution.

2. Reduce, version 3.13.080428, from http://kinemage.biochem.duke.edu/software, supplied with the DOT distribution.

3. MSMS, download from http://mgltools.scripps.edu/downloads, NOT in the DOT distribution, see below.

DOT and DOT utilities are in $DOT_ROOT/bin. The bin subdirectory contains a “share” subdirectory, which contains platform-independent shell scripts. The bin subdirectory also contains platform-specific subdirectories, which each contain compiled DOT, DOT utilities, APBS, and Reduce. Executables are available for the following four platforms, with the indicated subdirectory names:

1. Intel Mac OS X, subdirectory i86Darwin8

2. PowerPC Mac OS X, subdirectory ppcDarwin8

3. Intel Linux, subdirectory i86Linux2

4. Sun SPARC Solaris 8, subdirectory sun4SunOS5

The directory names are generated by the script ’$DOT_ROOT/bin/share/archosv’. This script is run when you run the script to create DOT input or the script to run DOT. It queries the computer you are on and returns a string $ARCHOSV accordingly, so that the correct executable is found for your machine architecture and operating system. If you want to check that executables are available for your computer, type:

```
$DOT_ROOT/bin/share/archosv
```

The resulting string should match one of the directory names in $DOT_ROOT/bin listed above.

Note that the Mac OS X executables have been compiled on Mac OS X 10.4 Tiger (Darwin 8). However, they work also on 10.5 Leopard (Darwin 9), therefore we have set ‘archosv’ to report ‘Darwin8’ on Darwin 9 (Leopard).

Also included in the distribution are the $DOT_ROOT subdirectories, “src” and “data”.

D External programs needed to create DOT input files

D.1 General utilities: shell, awk/gawk

DOT utilities are either compiled programs (which are installed in $DOT_ROOT/bin/$ARCHOSV), or are written as shell scripts (which are installed in $DOT_ROOT/bin/share). Many of these shell scripts invoke standard Unix/Linux utilities, including awk, bash, csh, head, tail, sort, m4, sed, grep, rm, and date. None of these should be a problem, but if you find incompatibilies, let us know. The ‘m4’ program (used by gen_dot_prepscript) comes with Mac OS X, but only if you install the Developer Tools, so we supply ‘m4’ in the distribution, in the appropriate $DOT_ROOT/bin/...Darwin.. subdirectory. Some of the scripts need a version of ‘awk’ that has certain capabilities and is called ‘awk’ on some platforms and ‘nawk’ on others; the scripts first try to use ‘nawk’ and if ‘nawk’ is not found, they use ‘awk’. In all cases, Gnu AWK ‘gawk’ can be used. A few of DOT’s newest analysis tools use the “Ruby” programming language. If you need to run dot_pdb_e6d_eval_rmsd or pdb_rmsd_matrix and do not have Ruby installed, you may download it for free from http://ruby-lang.org/en/downloads; any version 1.8.2 or later is fine.

Creating the required input for DOT needs 3 specialized external programs MSMS, Reduce, and APBS. MSMS can be downloaded, as described below. The DOT distribution includes pre-compiled binaries for Reduce and APBS. The DOT utilities invoke the programs by the names

```
msms
reduce
apbs
```

These programs are needed only on the platform on which your users will be preparing input for DOT. If your users will be preparing input only on a particular platform, such as Mac OS X on Intel, you need install them for that platform only.

Detailed explanation of each program and library follows.

D.2 MSMS

The MSMS program does molecular surface calculations needed by prepscript. MSMS is from the laboratory of Michel Sanner. Binary executables of MSMS for all of the platforms supported in the current DOT distribution are available from http://mgltools.scripps.edu/downloads. Be aware that you will need to scroll down this page to find the MSMS package. Download the appropriate "tar.gz" file, run 'gunzip ....tar.gz', then make a new directory (say, i86Darwin8 if that is your platform), cd into that directory, then run 'tar xvf ../...tar.gz'. This will make about a dozen files, including documentation and release notes; the executable file is named 'msms.platform....versionnumber'. Copy that file to $DOT_ROOT/bin/$ARCHOSV/msms. For example, if for our Mac, we did

```
tar xvf msms_MacOSX_2.6.1.tar
```

the resulting files include the file msms.MacOSX.2.6.1 and we put a copy in the appropriate $DOT_ROOT/bin directory:

```
cp msms.MacOSX.2.6.1 $DOT_ROOT/bin/i86Darwin8/msms
```

(Note that the MSMS team is currently distributing a PowerPC executable (ppcDarwin8) but not an Intel-native one; we have tested the PowerPC msms on an Intel Macintosh (i86Darwin8) and it runs fine because Mac OS X recognizes it and invisibly runs it inside "Rosetta", which simulates a Power PC on an Intel. So copying the same binary to both ppcDarwin8 and i86Darwin8 is OK.)

D.3 REDUCE

The Reduce program adds hydrogens to molecular models in an intelligent and flexible way needed by prepscript. Reduce is from the laboratory of David and Jane Richardson at Duke University, written by J.Michael Word. Reduce is free, open-source software available from http://kinemage.biochem.duke.edu/software
The DOT project is currently using reduce.3.13.080428 .

Reduce needs to know where its heteratom group dictionary is. Through the courtesy of the authors of Reduce, we distribute a copy of this dictionary in the $DOT_ROOT/data directory, file reduce_wwPDB_het_dict.txt .

D.4 APBS

The APBS (Adaptive Poisson-Boltzmann Solver) performs electrostatic potential calculations needed by prepscript. APBS is from the laboratory of Nathan Baker (baker@biochem.wustl.edu) of the Department of Biochemistry and Molecular Biophysics, Center for Computational Biology, Washington University in St. Louis. APBS is free, open-source software available from http://sourceforge.net/projects/apbs . After you have downloaded the executable binary for your platforms of interest, you may install apbs anywhere you like as long as it is in your $path when you run prepscript. We found that the Mac OS X (Darwin8) version, a "universal binary" for both PPC and i86 Macintoshes, needs administrator privileges to install; this appears to be a mistake in the installation program. The DOT project is currently using APBS version 1.0.0, 21 April 2008.

E External libraries needed to run DOT

The DOT distribution includes two external libraries, FFTW and MPICH. MPICH is needed for all platforms on which your users will be running DOT in parallel, and you will need to configure and make it for them.

E.1 MPICH

The MPICH software allows DOT calculations to be done in parallel on a local-area network of computers. MPICH is from the Argonne National Laboratory of the United States Department of Energy and is free, open-source software.

You only need MPICH if you wish to run a DOT job on more than one computer. In principle, you can run DOT using MPICH without recompiling the MPI libraries but we do not know a simple way to install only the user-level MPI scripts so here is how to install all necessary MPI routines for running on a network of local workstations such as Intel Linux, PPC or Intel Macintosh, or SPARC Solaris.

To run MPI programs on your local network, your users must be able to use either ‘ssh’ or else ‘rsh’ without typing passwords, this can be thorny if security is a severe concern, see http://www-unix.mcs.anl.gov/mpi/mpich1/docs/mpichman-chp4/node127.htm#Node127 and http://www-unix.mcs.anl.gov/mpi/mpich1/docs/mpichman-chp4/node128.htm#Node128

We found at TSRI that the MPICH installation ”configure” step also appears to check for the abilty of you, the installer, to run ssh without a password, and configurations done by people who could not do this were not usable by people who could, so our advice is that installers should make sure they can ”ssh (localhost)” without a password.

The DOT project currently uses MPICH 1 version 1.2.7p1. (DOT does not use MPICH 2 because currently MPICH 2 does not let users mix different kinds of computers in a single parallel run, important in a typical heterogeneous workstation network).

MPICH is available from http://www-unix.mcs.ans.gov/mpi/mpich1/download.html (about 16 MB) Download into $DOT_ROOT/mpich, unpack as $DOT_ROOT/mpich/mpich-1.2.7p1

Configure, build, and install for each platform ($ARCHOSV), on a host that supports that platform (eg, Intel Macintosh for i86Darwin8):

```
source $DOT_ROOT/bin/share/dot2.setup.bash (or .csh)
cd $DOT_ROOT/mpich/$ARCHOSV
```

(note: we found that on Mac OS X (Darwin8) we had to set the environment variable RSHCOMMAND at this point. On csh, type ”setenv RSHCOMMAND /usr/bin/ssh”. On bash, type ”export RSHCOMMAND=/usr/bin/ssh”)

```
../configure_for_dot2
```

This runs ../mpich-1.2.7p1/configure –prefix=$DOT_ROOT/mpich-1.2.7p1/$ARCHOSV –with-device-name=ch_p4 –disable-f77 –disable-cxx –disable-f90modules –disable-short-longs

```
make
```

Note: a make install is not necessary if you use the prefix above.

See http://www-unix.mcs.anl.gov/mpi/mpich1/docs/mpichman-chp4/node44.htm#Node44 for more configuration advice.

We found we can ignore messages like
cd .. && /bin/sh /usr/local/dot2.0/src/config_aux_dir/missing –run autoconf
autom4te: cannot lock autom4te.cache/requests with mode 2 (perhaps you are running make -j on a lame NFS client?): Operation not supported
make: *** [../configure] Error 1

These seem to be a clock-skew problem between the different hosts but we’re not sure. If you get these, it should be OK to use

```
make -i; make -i install
```

so “make” will ignore errors.

F What else you need to be able to recompile DOT programs

F.1 GNU Autoconf/Automake

We strongly recommend you download the most current versions of

- The GNU Autoconf tools from http://www.gnu.org/software/autoconf/ (the DOT project is using version 2.61).
- The GNU Automake tools from http://sources.redhat.com/automake/ (the DOT project is using version 1.10).

F.2 FFTW

The FFTW fourier transform library is needed to compile DOT whether you want to run parallel or single-processor. FFTW is free software, available from http://www.fftw.org, see http://www.fftw.org/fftw3_doc/Installation-on-Unix.html#Installation-on-Unix The DOT project is currently using version 3.1.2. Download into $DOT_ROOT/fftw3, unpack as $DOT_ROOT/fftw3/fftw-3.1.2. For each platform ($ARCHOSV):

```
cd $DOT_ROOT/fftw3/$ARCHOSV
../configure_for_dot2
```

This runs ../fftw-3.1.2/configure –enable-portable-binary –enable-float –prefix=$DOT_ROOT/fftw3/$ARCHOSV

```
make
make check # (optional but recommended)
make install
```

F.3 Recommended directory layout

Under $DOT_ROOT/: bin, src, data, mpich, fftw3
Under $DOT_ROOT/bin: share and individual platform directories
In $DOT_ROOT/bin/share: non-compiled executable scripts DOT users need
In $DOT_ROOT/bin/$ARCHOSV: compiled executable programs DOT users need
Under $DOT_ROOT/src: dot, util, and share
In $DOT_ROOT/src/dot: dot C source files
Under $DOT_ROOT/src/dot/$ARCHOSV: individual platform build directories
In $DOT_ROOT/src/util: dot utility C and C++ source files
Under $DOT_ROOT/src/util/$ARCHOSV: individual platform build directories
In $DOT_ROOT/src/share: distribution copy of executable scripts
In $DOT_ROOT/data: non-compiled, non-executed resources DOT utilities need
Under $DOT_ROOT/mpich/$ARCHOSV: distribution and individual platform directories
Under $DOT_ROOT/fftw3/$ARCHOSV: distribution and individual platform directories

Note that in four cases (src/util, src/dot, mpich, and fftw3), you do not run configure/make in those directories but in a subdirectory that is named after a particular platform ($ARCHOSV). For example, on i86Linux2, you would first make fftw3 and mpich, then the utilities, then DOT itself:

```
cd $DOT_ROOT/mpich/i86Linux2 ; ../configure_for_dot2 ; make
cd $DOT_ROOT/fftw3/i86Linux2 ; ../configure_for_dot2 ; make ; make install
cd $DOT_ROOT/src/util/i86Linux2 ; ../configure ; make ; make install
cd $DOT_ROOT/src/dot/i86Linux2 ; ../configure ; make ; make install
```