Molecular Surface Guide

Michael L. Connolly

1259 El Camino Real, #184
Menlo Park, CA 94025
U.S.A.

Version 3.9.2
February 7, 2002

Table of Contents

Chapter 1: Installation 5
Downloading the Software 6
Suggested Directory Structure 6
Compiling the Programs 6
Chapter 2: Concepts 7
Protein Surface 8
Solvent Accessibility 8
Piecewise-Quartic Molecular Surface 10
Volumes 10
Cavities 10
Density 11
Curvature 11
Chapter 3: Molecules 13
Atomic Coordinates 14
Chemical Bonds 14
Atom Types and Radii 14
Overriding the Defaults 14
Chapter 4: Computing Volumes and Areas 15
Computing Volumes 16
Using the Defaults. 16
Setting the Probe Radius. 16
Setting Atomic Radii 16
Computing Areas 16
Using the Defaults. 16
Areas and Volumes in the Same Run 17
Sorting the Atomic Areas Output File. 17
Chapter 5: Coloring 19
Pre-Defined Colors 20
Outer and Inner Colors 20
Uniform Coloring 20
Component Coloring 21
Atom Coloring 21
Shape Coloring 21
Chapter 6: Rendering Molecules 23
Crambin Ball-and-Stick Model. 24
Surface Colored by Atom 25
Displaying Translucent Surfaces 25
Clipping Surfaces 27
Solid Surfaces 27
Chapter 7: Dot Surfaces 29
Points, Areas and Normal Vectors 30
Surface Point Output File 30
Atomic Areas Output File 30
Displaying Dot Surfaces 30
Chapter 8: Polyhedra 31
Computing Polyhedra 32
Using the Defaults. 32
Tessellation Finess Parameter 32
Tessellation Fineness File. 32
Displaying Polyhedra 32
Plotting Surfaces 33
Chapter 9: Identifying Internal Cavities 35
Displaying Cavities. 36
Cavities Output File 37
Chapter 10: Solvent-Excluded Densities 39
Computing Densities. 40
Computing a Polyhedron from a Density 40
Displaying Densities 40
Chapter 11: Measuring Curvature 41
Solid Angle 42
Solid Angle Computations. 42
Single Polyhdral Surface 42
Three Concentric Spheres per Vertex. 42
Contouring Vertex Properties 42

Contouring in Stereo 43
Rotating Colored Polyhedra 43
Measuring Density Curvature 44
Curvature and Local Symmetry *44*
Sector Densities *44*
Chapter 12: Interfaces of Protein Complexes 45
Visualizing Protein Complexes. 46
A Pair of Polyhedra 47
Surface Buried in an Interface 48
Computing a Surface Between Two Densities 48
Polyhedra Evaluated at an Interfacial Surface 49
Densities Evaluated at an Interfacial Surface 49
Chapter 13: Command-Line Arguments 51
msroll 52
Grid Spacing Parameter *52*
Molecule Input File *52*
Name *52*
Probe Radius *52*
Piecewise Quartic Output File *52*
Atomic Radii Input File *52*
Polyhedron Output File *52*
Cusp Intersections *53*
msform 53
msdraw 53
Purpose *53*
Output File Names and Formats *53*
Image Size Parameter *54*
Tilt of Clipping Plane *54*
Border Parameter *54*
Alignment Parameter *54*
Title String *54*
Header Type *54*
Z Clipping *54*
Stereo Angle *55*
Shading Type *55*
Fineness Parameter *55*
mstran 55
Chapter 14: Setting Atom Fields 57
Atom Fields. 58
Sets of Atoms 59
Atom Field Files 59
Selecting Sets of Atoms 59
Setting Atom Fields 60
Chapter 15: Scene Script 61
Scene Filename Argument 62
Command Syntax 62
Global Commands 62
Rotation *62*
Shading Model *62*
Light Source Direction *62*
Overlap Hue *63*
Molecule Commands 63
Molecule *63*
Read PQMS Surface *63*
Read Polyhedral Molecular Surface *63*
Normal Vector Length *63*
Ball-and-Stick Model *63*
Define Color *64*
Input Coloring *64*
Atom Coloring *64*
Uniform Coloring *64*
Component Coloring *64*
Shape Coloring *65*
Atom Opacity *65*
Uniform Opacity *65*
Component Opacity *65*

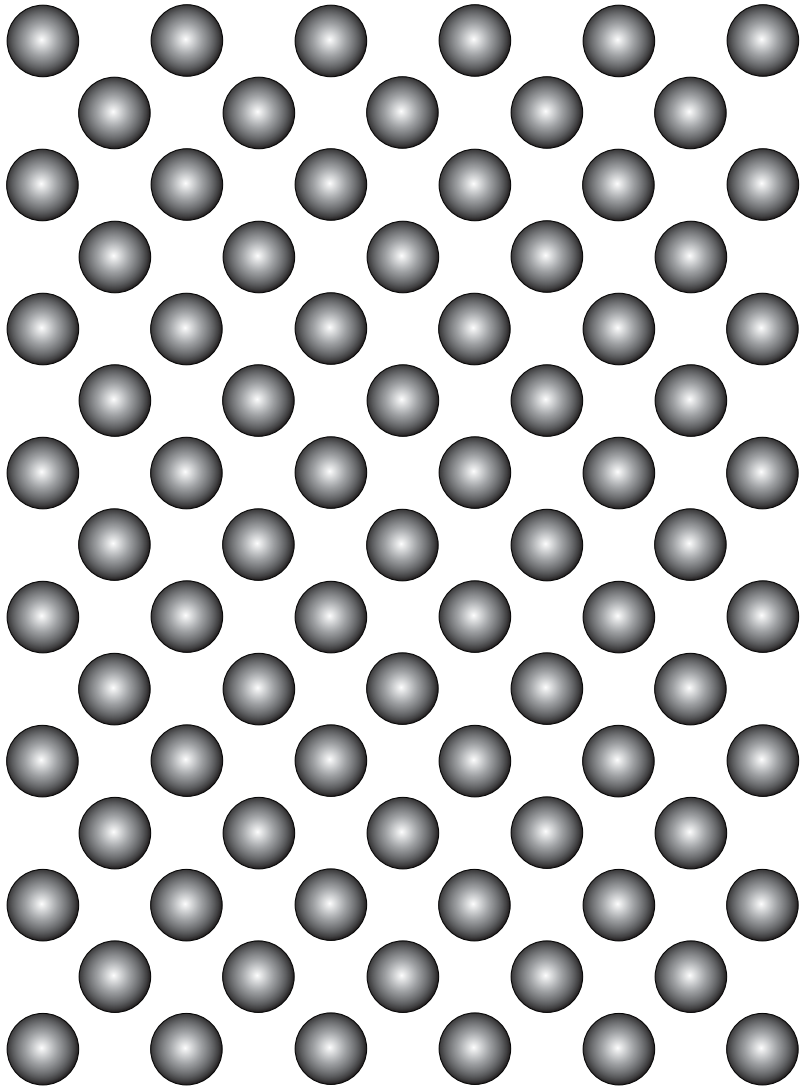
Shape Opacity *65*
Solid Shade *65*
Plot Line Width *66*
Elbow Parameter *66*
Ball Radius Parameter *66*
Bond Radius *66*
Blanking Polyhedron Edges *66*
No Clipping *67*
Contouring Commands 67
Define Ramp *67*
Contour *67*
Chapter 16: Disk File Formats 69
Atomic Radii 70
Atom Name Patterns 70
Atomic Areas 71
Volumes and Cavities 71
Surface Points 71
Polyhedron 72
Density 72
AVS Field Format *72*
PQMS Disk Format 73
Header Record *73*
Record Types *73*
Surface Elements *74*
Image File Formats 75
Indexed and RGB Color *75*
Sun Raster *75*
SGI Image *75*
Vector File Formats 75
SGI Inventor *75*
Plotting: HPGL and PostScript *76*

C h a p t e r ***1***

Installation

Chapter Overview

- Downloading the Software
- Suggested Directory Structure
- Compiling the Programs



A. Downloading the Software

Go to www.biohedron.com and follow the link that says source code. You will then be challenged for your userid and password.

You will need to obtain these two items by emailing the author at connolly@biohedron.com and stating you would like to try out the software in order to decide whether to buy it. If you succeed in convincing me to trust you and you get past this password challenge, you will find yourself in a directory with several files.

Download two of them: msp.tar.Z and doc.tar (the other files in this directory are older versions, plus object code for people who do not have compilers). After downloading msp.tar.Z, run the UNIX uncompress command. This will give you msp.tar. The msp.tar and doc.tar files can then have their contents extracted by the UNIX tar command. See the man pages on uncompress and tar for details.

- On a Macintosh you can use StuffIt Deluxe.
- On Windows, you can use WinZip.

B. Suggested Directory Structure

Parallel to the directory into which you unpacked the source files, you should create a bin directory. The makefile will try to put the executable files in ../bin. Of course, you can modify the makefile to adapt to whatever directory structure that you have.

C. Compiling the Programs

The source code is ANSI C. In a UNIX shell, you can make all of the executable files by typing in:

```
% make all
```

There are also targets for each individual program:

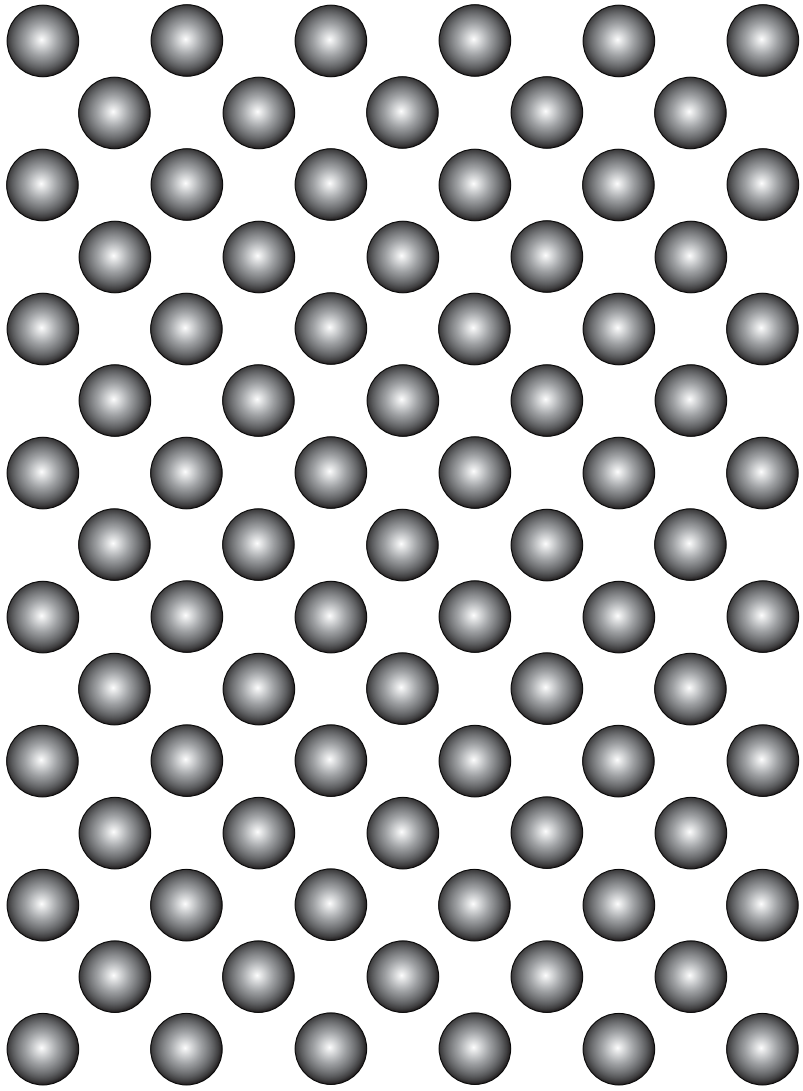
```
% make msroll
% make msform
% make msdraw
% make mstran
```

C h a p t e r **2**

Concepts

Chapter Overview

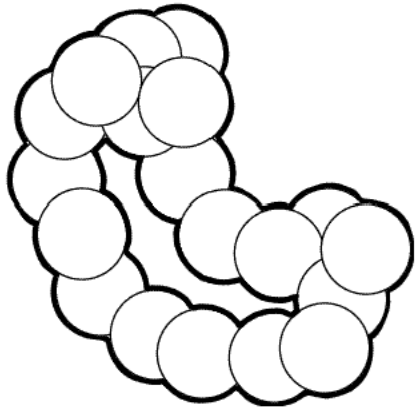
- Protein Surface
- Solvent-Accessibility
- Piecewise-Quartic Molecular Surface
- Volumes
- Cavities
- Density
- Curvature



A. Protein Surface

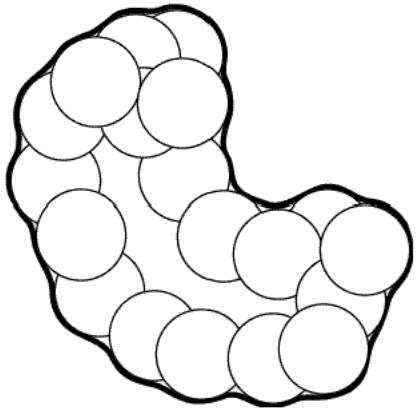
The topography of protein surfaces is quite irregular. The analysis of the topography of protein surfaces has had application in structure-based drug design. In particular, drugs bind in crevices on protein surfaces.

Figure 1 Atomic Surface



An important first step in analyzing protein topography is defining a surface for a protein molecule. The simplest space-filling representation of a molecule is the hard-sphere model. Each atomic sphere is given the van der Waals radius of the atom. The van der Waals surface of the molecule is made up of the parts of the van der Waals surface of each atom that do not lie inside any other atom. Much of the van der Waals surface of a protein is on the inside of the protein (Figure 1), so this is not a suitable surface for studying the binding of other molecules onto protein surfaces.

Figure 2 Molecular Surface



B. Solvent Accessibility

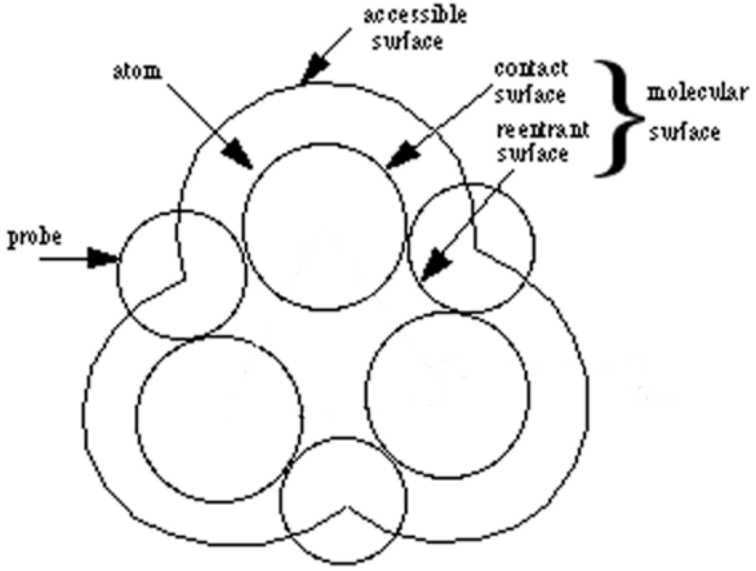
The outer surface of a protein molecule can be identified by using ideas of B.K. Lee & F.M. Richards (Journal of Molecular Biology, 1971) on rolling a water-sized probe sphere over a molecule (Figure 2). The surface of a macromolecule can be defined to be the part of the molecule that is accessible to solvent. The solvent molecule (water) is represented by a sphere of radius 1.4-1.7 angstrom, called the probe sphere.

The solvent-excluded volume is that volume of space that the probe is excluded from by collisions with the

atoms of the molecule. This volume is bounded by the molecular surface.

All surface and volume methods have a dependence on factors such as atomic radii, the probe radius, the quality of atomic coordinates, and whether explicit hydrogen atoms have been included. For protein and nucleic acid molecules, the hydrogen atom positions are generally not known, and heavy atoms with hydrogens are approximated by a single sphere, whose van der Waals radius is augmented by one to three tenths of an angstrom. Of course, all molecules have thermal motions, which are not modeled by these static, geometrical methods.

Figure 3 Accessible Surface



The accessible surface is the trace of the probe sphere center as it rolls over the molecule of interest. The contact surface is that part of the van der Waals surface of the atom that can be touched by a probe sphere. The reentrant surface is the inward facing part of the probe sphere as it is touching more than one atom. Together, the contact surface and the reentrant surface form a continuous sheet called the molecular surface (Figure 3).

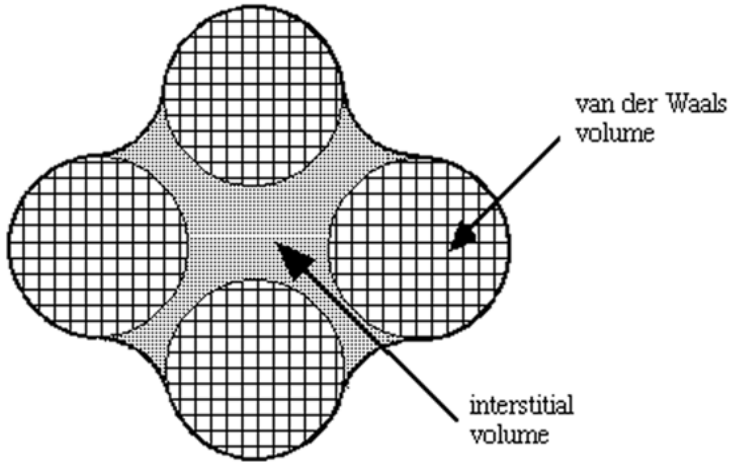
C. Piecewise-Quartic Molecular Surface

In three dimensions, the contact surface is made up of convex spherical regions, while the reentrant surface is made up of saddle-shaped rectangles and concave spherical triangles. Because the saddle-shaped rectangles are part of the surface of a torus, which is defined by a fourth-degree polynomial, the surface must be called piecewise quartic. The piecewise-quartic molecular surface is sometimes approximated by polyhedral surfaces.

D. Volumes

The solvent-excluded volume is the region of space that the probe sphere is excluded from by collisions with the atoms of the molecule. The volume enclosed by the union of the atom spheres is called the van der Waals volume. The solvent excluded volume minus the van der Waals volume is called the interstitial volume (Figure 4).

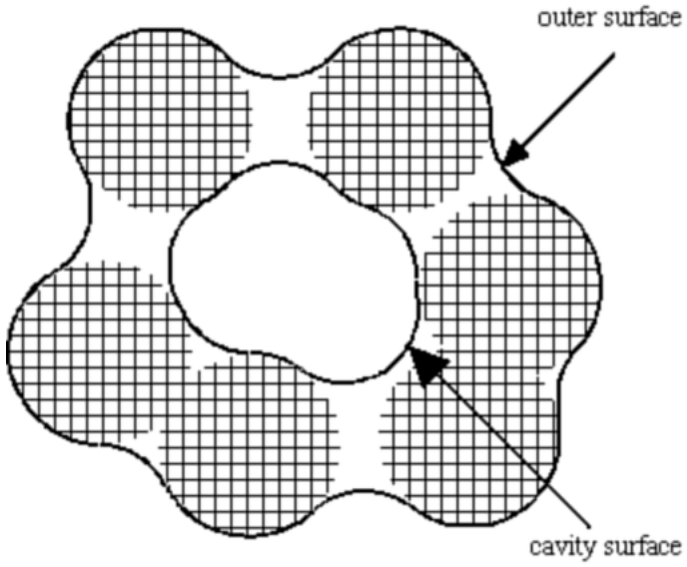
Figure 4 Solvent-Excluded Volume



E. Cavities

In macromolecules, such as proteins, there can be not just small interstices, but also larger gaps between atoms in the interior, gaps that are large enough to hold a water-size probe sphere. These internal voids or cavities will each be enclosed by a component of the molecular surface (Figure 5). The definition of internal cavities depends upon the probe radius chosen. Also, an internal cavity may be disconnected from the outer solvent in the static, X-ray structure, but connected when protein breathing motions are considered.

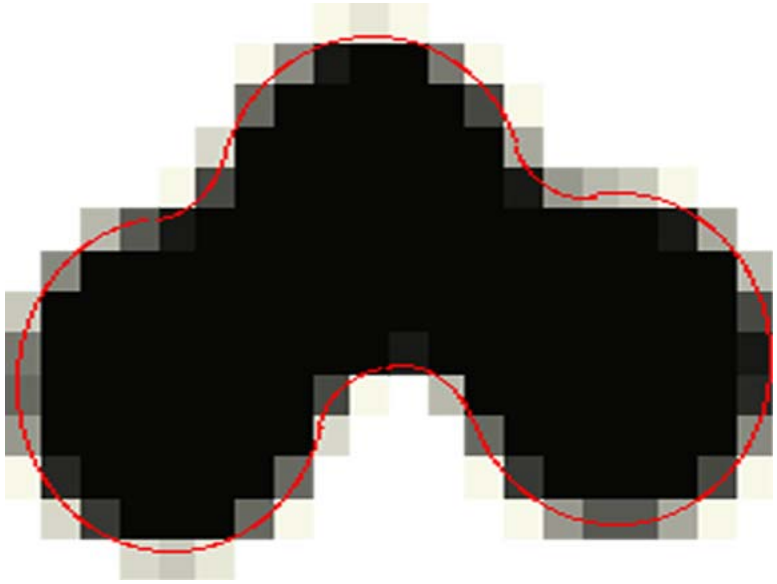
Figure 5 Cavity Surface



F. Density

Densities are a lattice of numerical values. All density values will lie between 0.0 and 1.0. From a crystallographic point of view, this value range more resembles occupancies, than electron densities.

Figure 6 Density/Occupancy



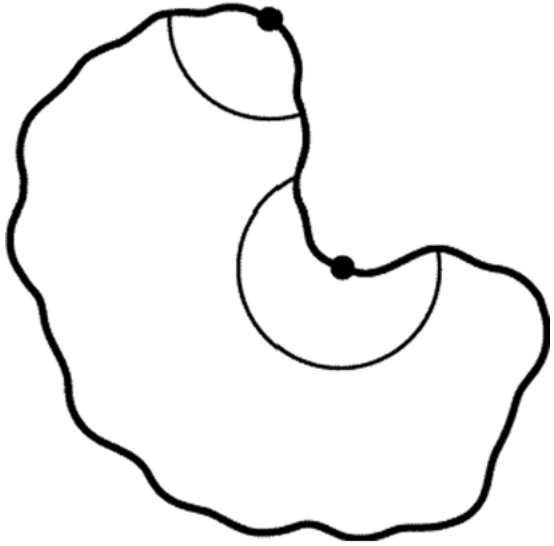
A convenient representation is to use shades of grey (Figure 6). A smaller cube width gives a more detailed representation, but it also increases the computation time and memory requirements. Also, the fineness should not exceed the quality of the data.

G. Curvature

An idea related to accessibility is to center at each point on the protein surface a sphere of roughly amino-acid-residue size, and then compute the part of the sphere that is inside the surface (Connolly, 1986). This gives an area measured in square angstroms, which

may be divided by the square of the sphere radius to give a solid-angle measure in steradians between 0 and 4 π. This method is analogous to accessibility, but identifies features at a larger scale. The value is large for concave regions and small for convex regions (Figure 7).

Figure 7 Curvature Measured by Angle

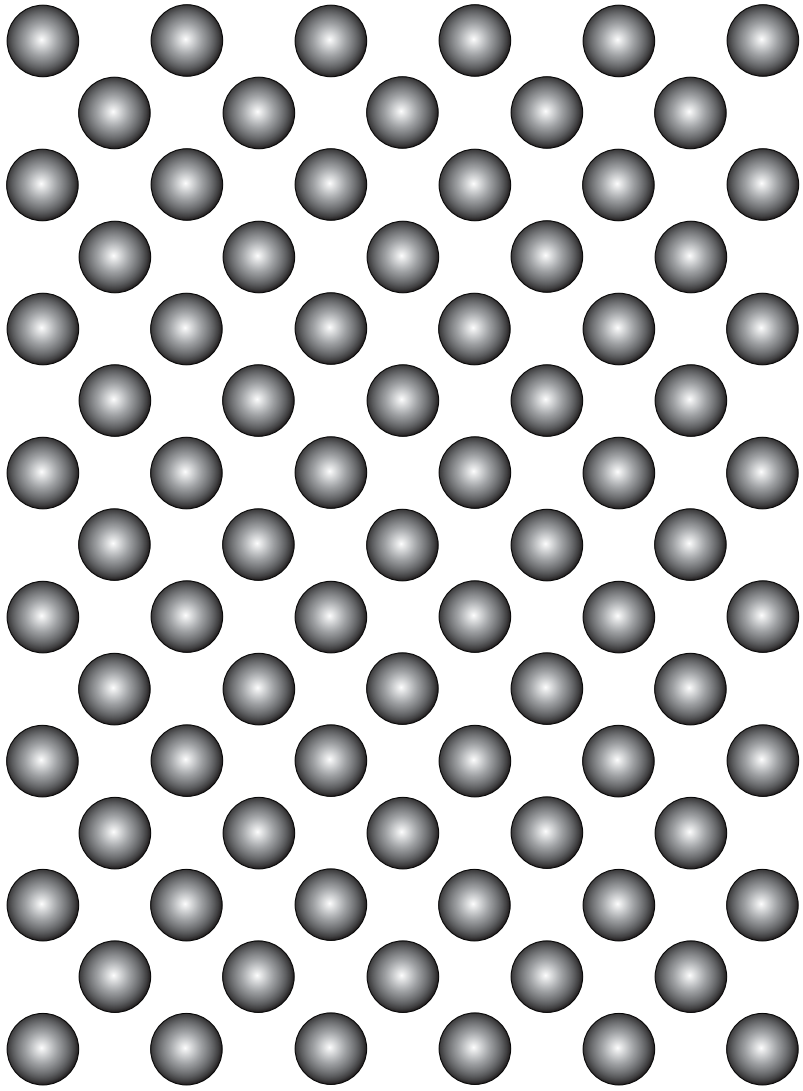


C h a p t e r ***3***

Molecules

Chapter Overview

- Atomic Coordinates
- Chemical Bonds



A. Atomic Coordinates

The three-dimensional coordinates of a molecule are the original input to the programs. These coordinates must be in protein data bank format. The program selects only those PDB input file records that begin with ATOM or HETATM. The program also replaces embedded blanks in some atom names by an underscore. These coordinates are used in a primary way by pqms, to compute surfaces and volumes, and in a secondary way by msdraw, to draw a ball-and-stick model.

B. Chemical Bonds

The connectivity is determined based upon interatomic distances and covalent radii. This is the only use made of covalent radii; they are not used in the surface calculation. Two atoms are connected if their separation is less than the sum of their van der Waals radii, plus a user-specifiable tolerance. Only atoms belonging to the same molecule will be connected.

C. Atom Types and Radii

Van der Waals radii are assigned by a two-step procedure. First the residue name and the atom name in the PDB file are used to determine that atom type, which is an integer, and then the atom type is used to assign a van der Waals radius. Similarly for the covalent radius used to determine connectivity. The defaults in the ms.c file can be overridden in the following way.

D. Overriding the Defaults

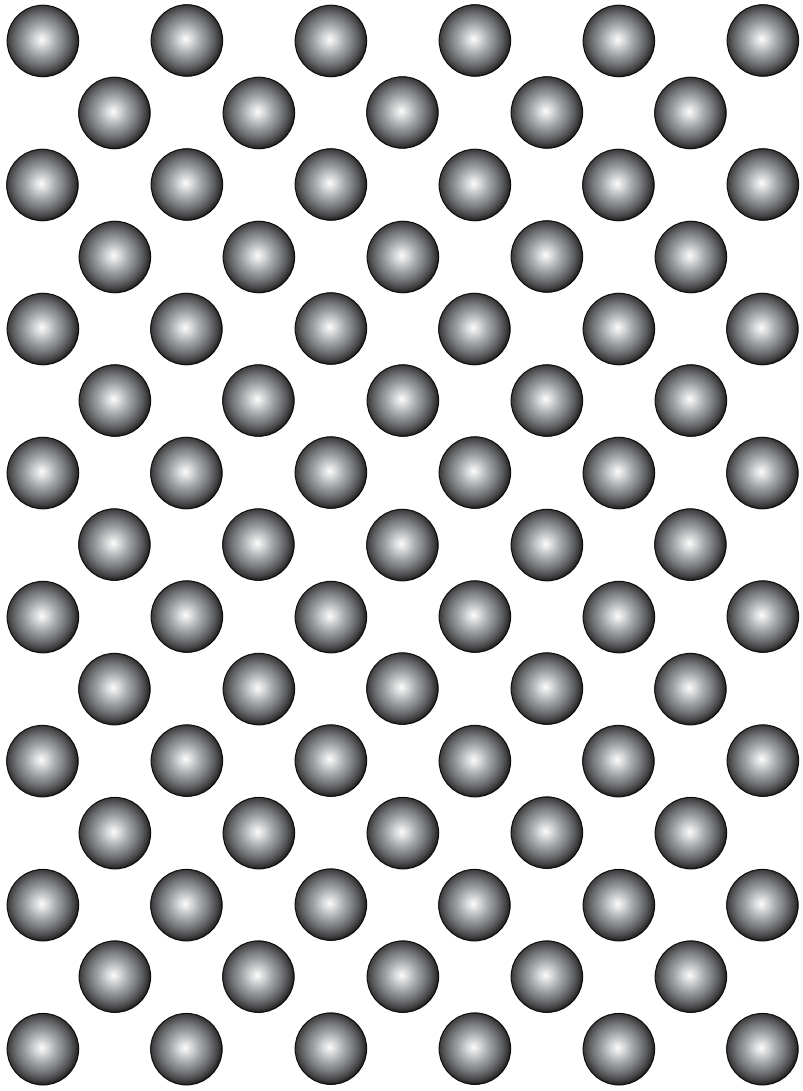
The program has a default set of atomic radii, specified in the ms.c file. To use your own radii, prepare a radius file, and read it with the -r flag. To also override the classification of atoms into atom types, prepare two files, a pattern file and a radius file, and then read them with the -r and -y flags. See “Atomic Radii” on page 70.

C h a p t e r ***4***

Computing Volumes and Areas

Chapter Overview

- Computing Volumes
- Computing Areas



A. Computing Volumes

Using the Defaults

The msroll program is used to compute volumes and areas of proteins. The simplest way to compute the volume of a protein is to accept the default values for atomic radii and the probe radius. There are then only two arguments to the program:

- the protein coordinate input file
- the volume output file

The order of arguments does not matter. Either command below will work fine. However, type it all on one line; the command runs to a second line here in the manual only because of the column width limitation.

```
% msroll -m protein.pdb -v
  protein.vol
% msroll -v protein.vol -m
  protein.pdb
```

The output file information includes the total area and volume of the molecule, plus a table of cavity areas, volumes and centers. Cavities are discussed in more detail in the chapter on cavities. Both piecewise-quartic and polyhedral volumes and areas are given. Do not expect the polyhedral and piecewise-quartic numbers to match too closely, unless the triangulation is very fine. See “Computing Polyhedra” on page 32.

Setting the Probe Radius

The simplest parameter to specify is the probe radius. The default value is 1.5 ångstrom, which is the size of a water molecule.

```
% msroll -m protein.pdb -p 1.8 -v
  protein.vol
```

Setting Atomic Radii

The user can choose her own atomic radii by specifying a disk file. See “Atomic Radii” on page 70. This chapter also gives the default atomic radii built into the program.

```
% msroll -m protein.pdb -r
  atoms.radii -v protein.vol
```

B. Computing Areas

Using the Defaults

The msroll usages discussed above give the area of the protein as a whole, and the areas of individual cavity surfaces, but not the areas of individual atoms. For that, you must use the -a argument, which writes the

atomic areas to a disk file. The format of this file is given in the chapter on disk file formats.

```
% msroll -m protein.pdb -a
  protein.area
```

The probe radius and atomic radii can be set as described above for volumes.

Areas and Volumes in the Same Run

One can also write both areas and volumes with a single command:

```
% msroll -m protein.pdb -p 1.8 -r
  atoms.radii -a protein.area -v
  protein.vol
```

Sorting the Atomic Areas Output File

`-a` *`file`* `[ bac | bca ]`

In the simplest situation, where there is no second argument for the area command, the output file contains one line for each atom. This line gives the contact, reentrant, molecular (contact+reentrant) and accessible areas of the atom. The bac argument causes the record order of the file to be mainly by atom and secondarily by component. For example:

```
% msroll -m protein.pdb -a
  protein.bac bac
```

The bac argument causes the record order of the file to be mainly by component and secondarily by atom. For example:

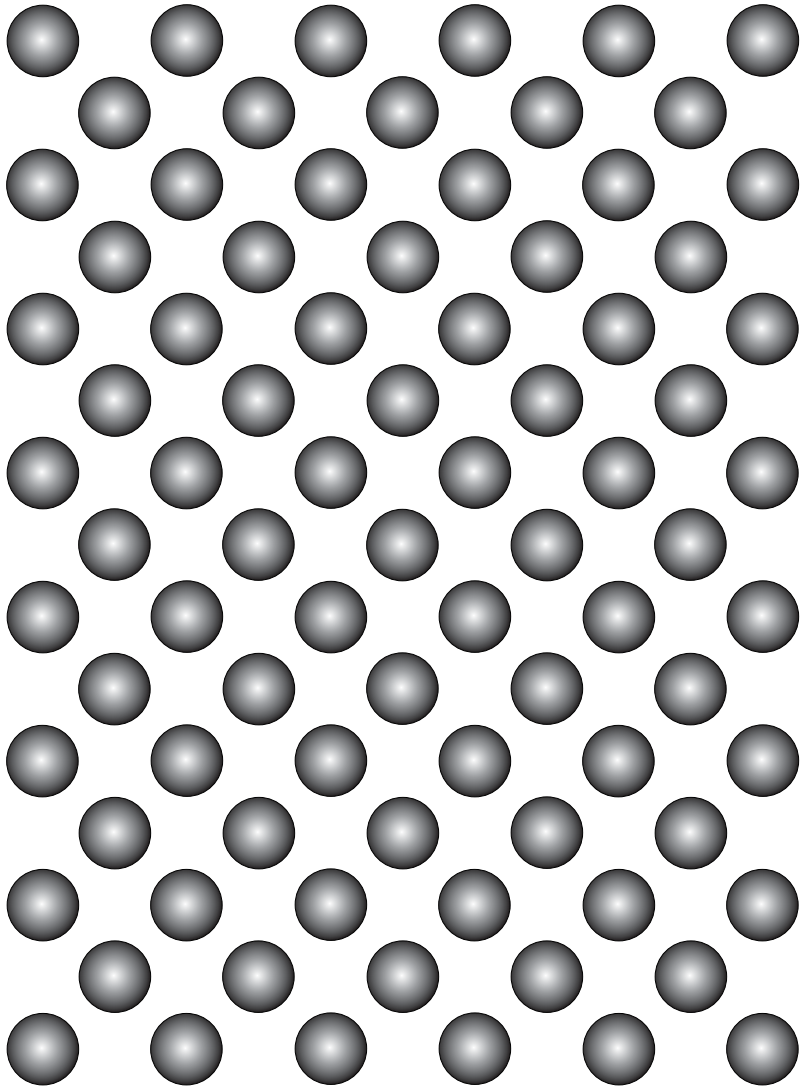
```
% msroll -m protein.pdb -a
  protein.bca bca
```

C h a p t e r **5**

Coloring

Chapter Overview

- Pre-Defined Colors
- Outer and Inner Colors
- Uniform Coloring
- Component Coloring
- Atom Coloring
- Shape Coloring



A. Pre-Defined Colors

The colors below are predefined. The integers are used in polyhedron files, while the names are used in the msdraw script files.

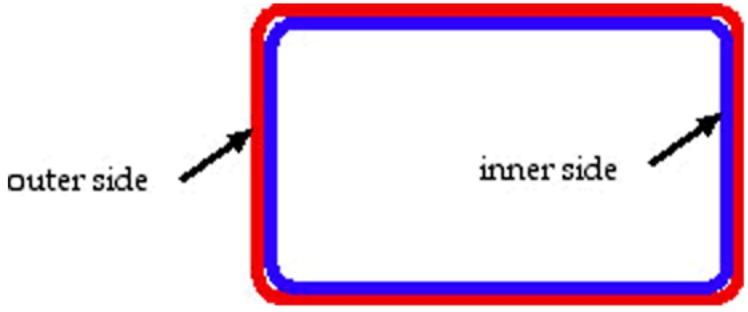
Table 1: Pre-defined Colors

#	red	green	blue	name
0	0.25	0.25	0.25	black
1	1.00	1.00	1.00	white
2	1.00	0.00	0.00	red
3	0.00	1.00	0.00	green
4	0.00	0.00	1.00	blue
5	0.00	1.00	1.00	cyan
6	1.00	0.00	1.00	magenta
7	1.00	1.00	0.00	yellow
8	0.50	0.50	0.50	grey
9	0.75	0.75	0.75	gray
10	0.40	0.20	0.10	brown
11	1.00	0.40	0.00	orange
12	0.50	0.00	0.30	purple
13	0.00	0.00	0.30	navy
14	0.40	0.40	1.00	sky
15	1.00	0.50	0.50	pink
16	0.80	1.00	0.00	yellow_green
17	0.80	0.40	0.10	tan

B. Outer and Inner Colors

There are several different types of coloring. Some of them distinguish between the outer side and the inner side. By outer and inner, I mean like the outer and inner surfaces of a bottle. No distinction is made for internal cavities. The outer side is the side facing the probe, including when the probe is in an internal cavity. The inner side is visible only if their is clipping.

Figure 8 Inner Coloring



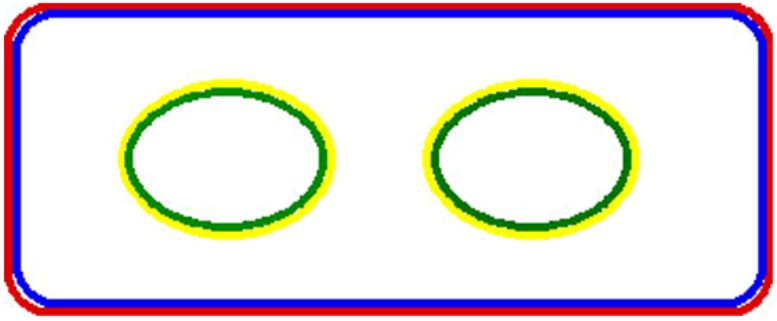
C. Uniform Coloring

In uniform coloring, only two hue numbers are given for the entire surface: one for the outer side and one for the inner side (Figure 8).

D. Component Coloring

In component coloring, there are two hue numbers for each surface component, one for the outer side and one for the inner side. In the diagram (Figure 9), we have given both internal cavities the same color.

Figure 9 Cavity Coloring.



The colors are given in Table 2:.

Table 2: Component Colors

Component	*Outer side*	*Inner side*
1: Outer surface	red	blue
2: First cavity	green	yellow
3: Second cavity	green	yellow

E. Atom Coloring

Atom coloring is based upon the color field of each atom.

F. Shape Coloring

Shape coloring is based upon a table,which has six hue numbers.

Table 3: Shape Colors

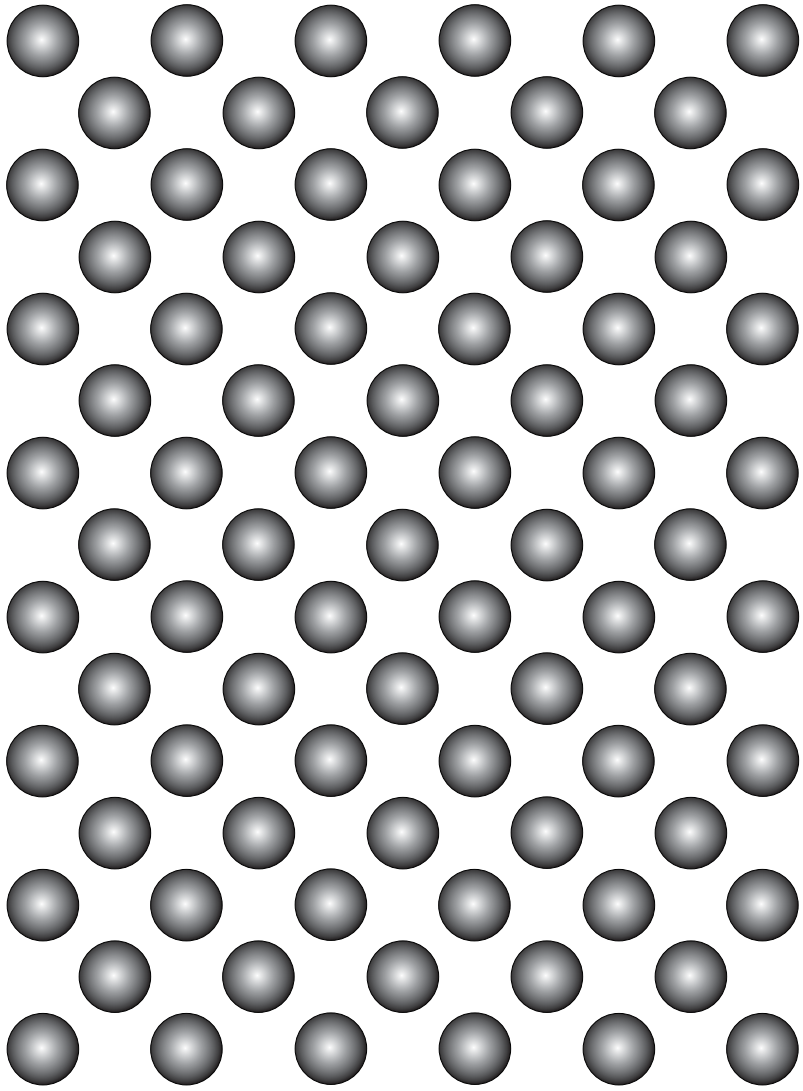
Shape	*Outer Side*	*Inner Side*
convex	hue	hue
saddle	hue	hue
concave	hue	hue

C h a p t e r ***6***

Rendering Molecules

Chapter Overview

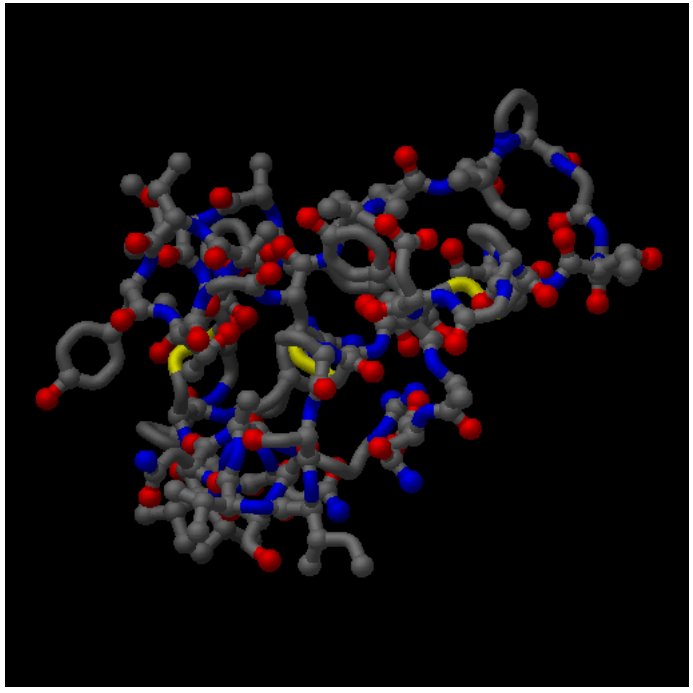
- Crambin Ball-and-Stick Model
- Surface Colored by Atom
- Displaying Translucent Surfaces
- Clipping Surfaces



A. Crambin Ball-and-Stick Model

A rendering of a ball and stick model of a protein is shown in Figure 10.

Figure 10 Chemical Model of Crambin



The coloring for the rendered model is taken from the color fields of the atoms. You can create this image with the shell commands and scene script below:

Shell commands:

```
% msdraw -i bas.scr -r bas.sun sun
% sdtimage bas.sun
```

The sun argument above causes the disk file to be written in 8-bit sunraster format. The msdraw program reads the following script.

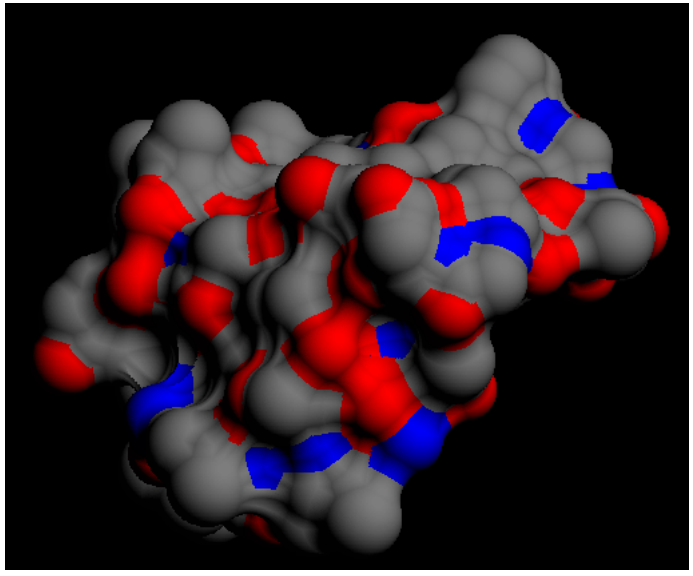
The bas.scr file:

```
light_source -0.5 1.0 1.0
shading_model 0.2 0.2 0.4 0.2 2.0
molecule crn 1crn.pdb
tolerance 0.25
elbow 2.5
connect crn # creates sticks
crn color = grey
oxygens = atom matches O
oxygens color = red
nitrogens = atom matches N
nitrogens color = blue
sulfurs = atom matches S
sulfurs color = yellow
ball_and_stick
atom_coloring
```

B. Surface Colored by Atom

A molecular surface of crambin, colored by atom type, is shown in Figure 11. The image was created with the shell commands and scene script below.

Figure 11 Crambin with Atom Coloring



Shell commands:

```
% msroll -m 1crn.pdb -q crn.pqms
% msdraw -i ms.script -r ms.sun sun
% sdtimage ms.sun
```

The ms.script file read by msdraw consists mostly of coloring commands. The pdb file specified for msdraw must be the same as the one for msroll.

ms.script

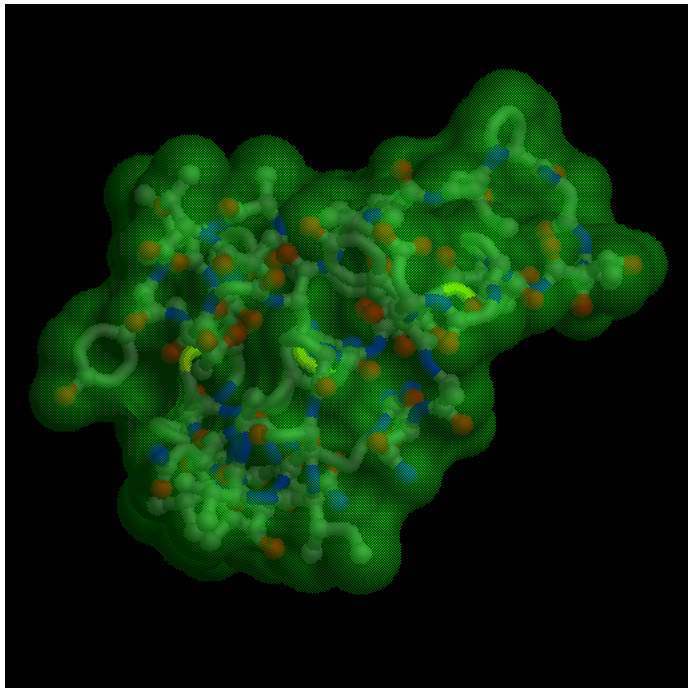
```
molecule crn 1crn.pdb
crn color = grey
oxygens = atom matches O
oxygens color = red
nitrogens = atom matches N
nitrogens color = blue
sulfurs = atom matches S
sulfurs color = yellow
read_surface crn.pqms
atom_coloring
```

C. Displaying Translucent Surfaces

The small plant protein crambin is shown in Figure 12 It has a green, translucent surface and a multi-colored ball-and-stick model. If you examine the green surface closely, by zooming in on the PDF file, you will see

that the translucency effect has been generated by omitting every other green surface pixel in a checkerboard pattern.

Figure 12 Translucent Surface of Crambin.



The image was created with the shell commands and scene script given below. There is one msroll UNIX shell command to create the surface, one msdraw command to render the surface using msdraw, and one command to look at the image file, using the SGI imgview utility. Both msroll and msdraw read atomic coordinates in pdb format. The msroll program reads them so that it can roll a sphere over the atoms to generate a surface, while the msdraw program reads them so that it can connect the atoms to form a ball-and-stick chemical model. The msdraw program takes its instructions from the script file specified by its -i argument. The output file raster file will be written in SGI image format.

Shell commands for rendering and viewing a translucent surface:

```
% msroll -m 1crn.pdb -q crn.pqms
% msdraw -i lucent.script -r lucent.rgb
% imgview lucent.rgb
```

The scene script file contains directions for only one molecule, crambin. It names the molecule “crn”, since within the program molecule names must begin with a letter, not a number. The connect command creates the bonds. The next command means that all atoms of crn are to have their color attribute set to grey. This might be written as crn.color = grey in many scripting languages. The scripting language operates on sets of atoms, rather than individual atoms. This obviates the need for loops to set the values of arrays of atoms. While the crn atom set is predefiined, the set of oxygen atoms must be defined. This is done by chosing all atoms whose name begins with the letter O. Once the

oxygens set is defined, its atoms have their color set to red. Similar commands take care of the nitrogens and sulfurs. These kinds of commands for dealing with sets of atoms are discussed more fully in the chapter on atom fields. The ball-and-stick chemical model of spheres and cylinders is next created, and colored according to the atom colors. A bond cylinder will have two colors if it connects atoms of different colors. The piecewise-quartic molecular surface is read in. It is made semi-transparent by giving it an opacity of 0.5, and it is colored green.

Script file translucent.script

```
molecule crn 1crn.pdb
connect crn # creates bonds
crn color = grey
oxygens = atom matches O
oxygens color = red
nitrogens = atom matches N
nitrogens color = blue
sulfurs = atom matches S
sulfurs color = yellow
ball_and_stick
atom_coloring read_surface crn.pqms
uniform_opacity 0.5 0.5
uniform_coloring green
```

D. Clipping Surfaces

`-z zclip`

This is an optional argument to the msdraw program. The clipping plane is assumed to be parallel to the xy plane, and the argument must be in the range -1.0 to 1.0. The number is not in angstroms, but represents a fraction of the z range of the molecules. That is, a value of 1.0 clips nothing, a value of 0.0 clips through the middle, and a value of -1.0 clips everything. Triangles that cross the clipping plane will be truncated. Bonds crossing the clipping plane will also be cut.

`-x tiltx`

`-y tilty`

These arguments are used to specify the tilt of the clipping plane. They can be in the range 0.0 to 1.0. Let x and y denote the values specified by the -x and -y arguments. Then (x, y, 1.0) gives a vector perpendicular to the clipping plane. For example, an x value of 1.0 gives a tilt of 45 degrees. It is not possible to specify negative values for these two flags.

E. Solid Surfaces

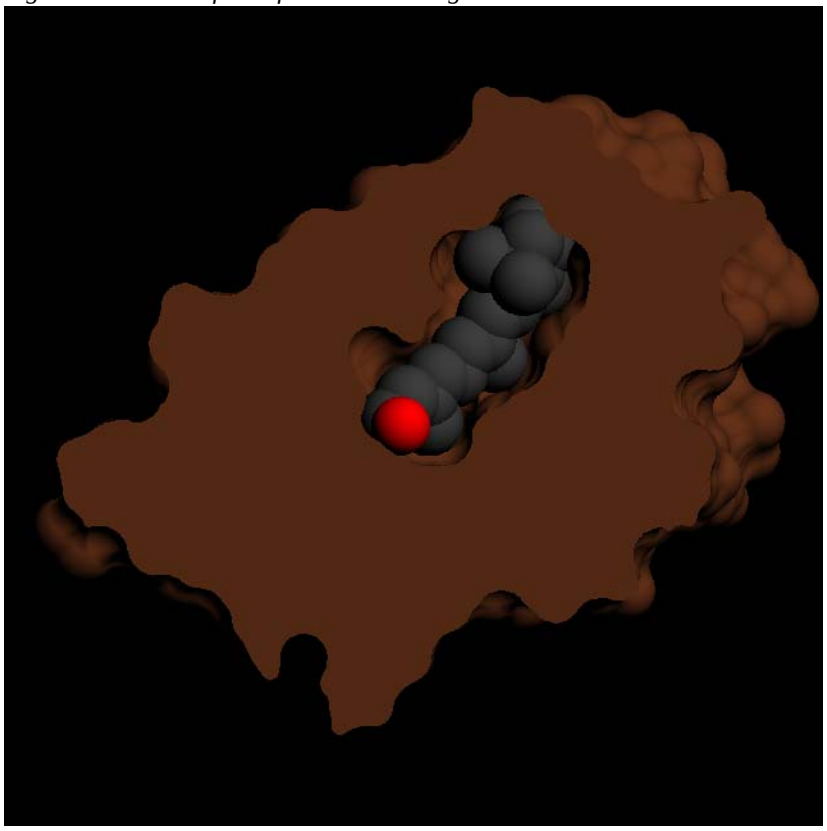
The solvent-excluded volume of the protein molecule can be made to appear solid by use of the solid_shade

script command. The command takes one argument, the shade (from 0 to 255) of the solid shade. The surface being made to appear solid should be given a uniform interior color. The solid_shade command should appear after the coloring command. If there is more than one solid surface in the scene, you should also put an overlap_hue command at the top of the file, before the first molecule command. This command takes one argument, the color of the overlap for interpenetrating solvent-excluded volumes. An example is given below for retinol binding protein.

```
rotation 90.0 0.0 1.0 0.0
overlap_hue green
molecule crb
surface crb.pqms
uniform_coloring brown brown
solid_shade 200
molecule retinol 1crb.pdb
retinol -= residue != RTL
oxygens = atom matches o
oxygens color = red
carbons = atom matches c
carbons color = black
surface retinol.pqms
atom_coloring
no_clipping
```

There are two surfaces in this example: for the protein and for the ligand. The protein is cut and solid. The ligand is given a van der Waals surface (retinol.pqms) and is not cut (note the no_clipping command).

Figure 13 Solid Surface of Retinol-Binding Protein

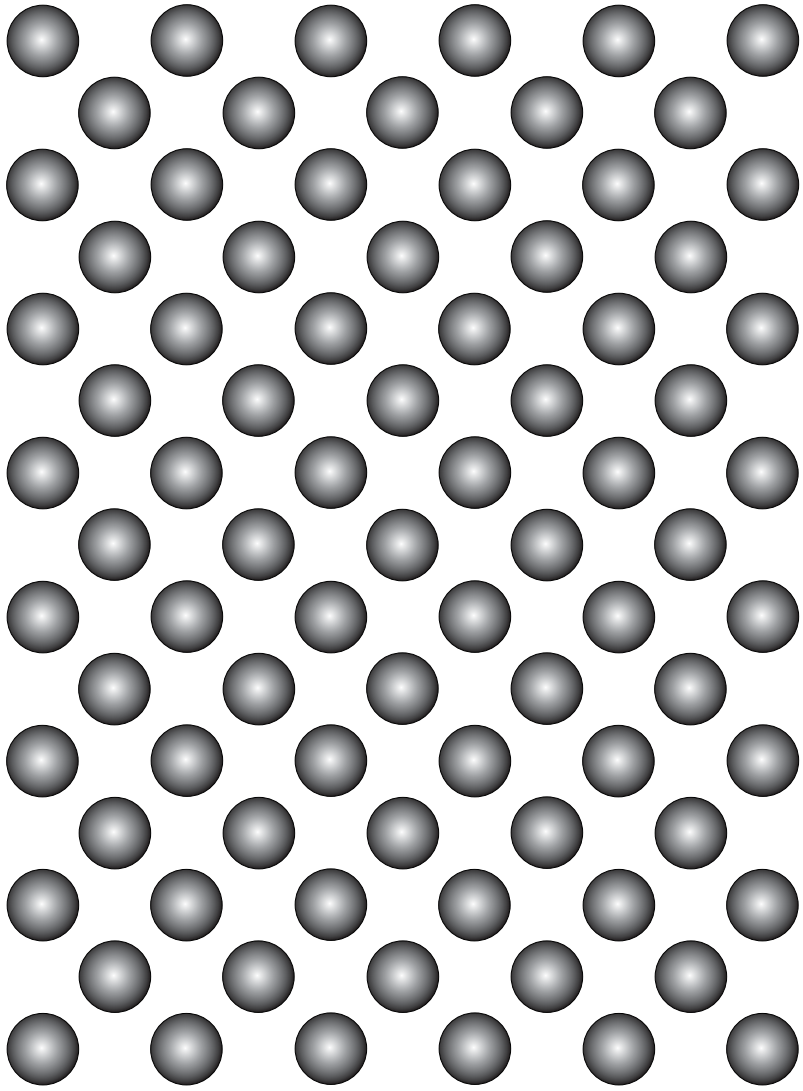


C h a p t e r **7**

Dot Surfaces

Chapter Overview

- Points, Areas and Normal Vectors
- Surface Point Output File
- Atomic Areas Output File
- Displaying Dot Surfaces



A. Points, Areas and Normal Vectors

A dot surface is a set of points in space. It is like a polyhedron, but with no edges and no faces. Its minimal nature gives it an elegant simplicity.

Each dot is specified by its x, y and z coordinates. It has an area, which gives the area of the part of the surface assigned to it. It also has a vector, of unit length, pointing directly away from the surface, toward the center of the probe sphere.

B. Surface Point Output File

`-j` *`file`*

All the surface points will be written to the disk file, along with their normal vectors, if calculated. The records will be in ascending atom number order. See “Surface Points” on page 71.

C. Atomic Areas Output File

`-k` *`file`*

This file gives the areas from the dot surface algorithm. The output file consists of one record per atom, and each record has three fields: contact area, reentrant area, and molecular area.

D. Displaying Dot Surfaces

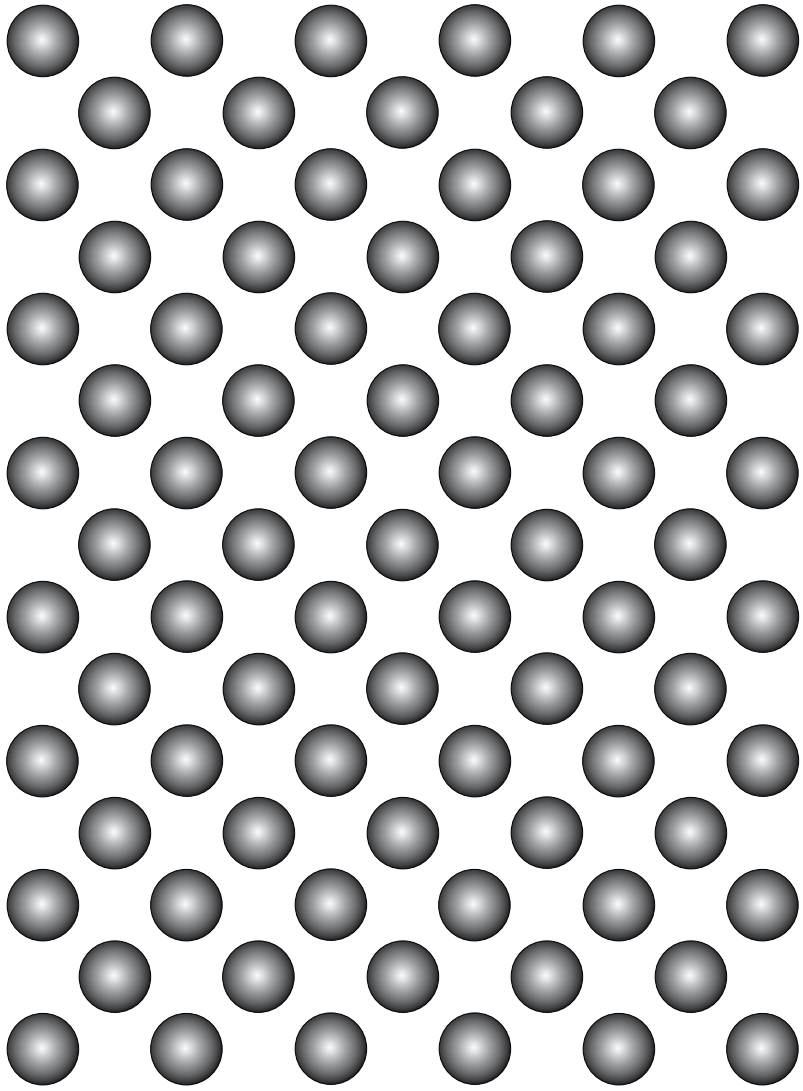
The dot surfaces produced by msroll are in the same format as those produced by the MS Fortran program, available from the Quantum Chemistry Program Exchange. There exist many programs for displaying these dot surfaces.

C h a p t e r **8**

Polyhedra

Chapter Overview

- Computing Polyhedra
- Displaying Polyhedra
- Plotting Polyhedra



A. Computing Polyhedra

Using the Defaults

The simplest command for computing a polyhedral molecular surface specifies only the input and output disk files:

```
% msroll -m protein.pdb -t
  surface.vet
```

Tessellation Finess Parameter

The polyhedral surface is made up of triangles. They are computed by subdividing the faces of the piece-wise-quartic molecular surface, not by connecting the points of the dot surface. The fineness parameter should be in radians. The maximum allowed value for this angle field is 1.5 radians. For a value of 1.5 radians, the surface will be crude. The default value of 1.0 gives a reasonably smooth surface. A small number generates a large number of small triangles. A large number generates a small number of large triangles. The execution time of the program increases inversely as the square of this parameter

```
% msroll -m protein.pdb -t
  surface.vet -f 0.8
```

Tessellation Fineness File

```
% msroll -m protein.pdb -t
  surface.vet -f fine.script
```

The script file (not shown) sets the angle fields of the atoms to the desired values. The atom's angle field tells the msroll program how finely to tessellate (triangulate) the surface belonging to this atom.

The fineness file can also restrict which atoms of the pdb file end up in molecule's atom set. One includes commands such as:

```
protein -= residue == HOH
```

This removes that waters from the input molecule crambin.

B. Displaying Polyhedra

The mstran program will convert a molecular surface polyhedron file to Silicon Graphics Inventor file, which can be displayed with the SGI ivview utility program.

```
% mstran -t polyhedron -o
  inventor_file
% ivview inventor_file
```

C. Plotting Surfaces

Plots are also generated by the msdraw program. The polyhedral surface is the input to the program. The triangles are subjected to hidden-line elimination. The output is written to disk in either HPGL or PostScript.

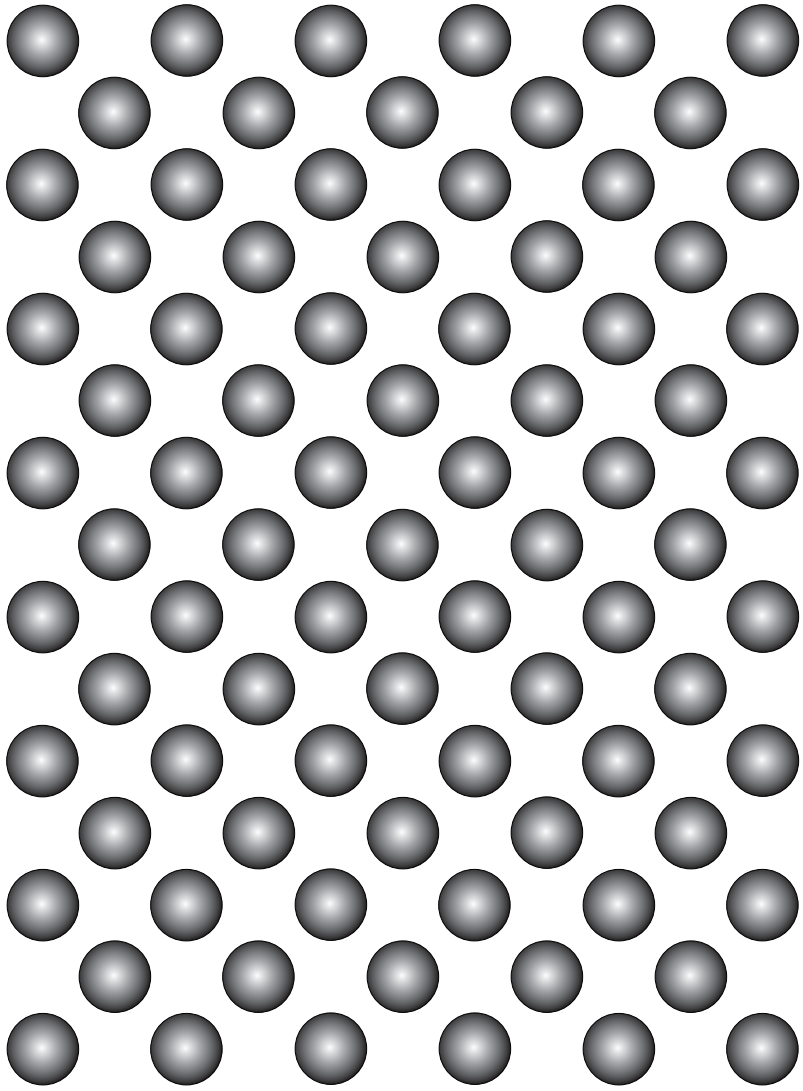
```
% msdraw -i script -p plot.ps ps
% msdraw -i script -p plot.hp hpgl
```

C h a p t e r **9**

Identifying Internal Cavities

Chapter Overview

- Displaying Cavities
- Cavities Output File



A. Displaying Cavities

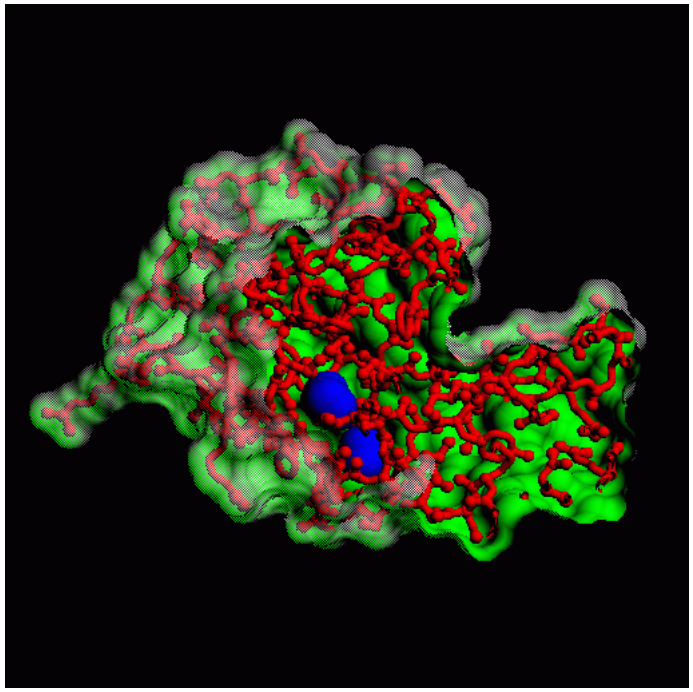
A rendering of a red model of lysozyme, with a translucent front surface and blue internal cavities, is seen in Figure 14. The front surface is white and translucent, while the rear surface is opaque and green. The cavity surfaces are blue. All the atoms and bonds are red. The front part of the surface has been clipped away.

The cavities are regions in the interior of the protein large enough to accomodate the probe. It has been found that if they are bordered by polar atoms, they generally contain water molecules. The msroll and msdraw programs are instructed to ignore the water molecules in the 2lyz.pdb file by using the following command.

```
lyz -= residue matches HOH
```

The –= assignment operator means subtract the set on the right from the set on the left. The set on the right is all the atoms whose residue names begin with HOH. HOH is the name that the protein data bank gives to water molecules.

Figure 14 Lysozyme with Internal Cavities in Blue



This image was produced with the commands and scene script below. The instuction for ignoring the water molecules shows up in the script files given to both msroll and msdraw.

```
Shell commands for rendering and
viewing a clipped surface with
```

internal cavities:

```
% msroll -m lyz.pdb -q lyz.pqms -f lyz.script
% msdraw -i cavities.script -r cavities.rgb -z 0.4
% imgview cavities.rgb &
```

The water-exclusion command is the sole line in the lyz.script file.

```
lyz -= residue matches HOH
```

In the scene script file, the protein is rotated 45 degrees around the y-axis. The last three numbers of the rotation command give the rotation axis as an xyz vector. This command is given first, because it applies to all the molecules in the scene. In our case, there is only one molecule, lysozyme. The molecule is read in not from the sister pdb directory as before, but from the current working directory. The waters are eliminated from the lyz atom set by the matching command. The coloring and chemical model commands are similar to the previous example.

Script file translucent.script

```
rotation 45.0 0.0 1.0 0.0
molecule lyz lyz.pdb
lyz -= residue matches HOH
connect lyz
lyz color = red
ball_and_stick
atom_coloring
read_surface lyz.pqms
component_coloring white green blue
component_opacity 0.5 1.0 1.0
```

B. Cavities Output File

`-c` *`file`*

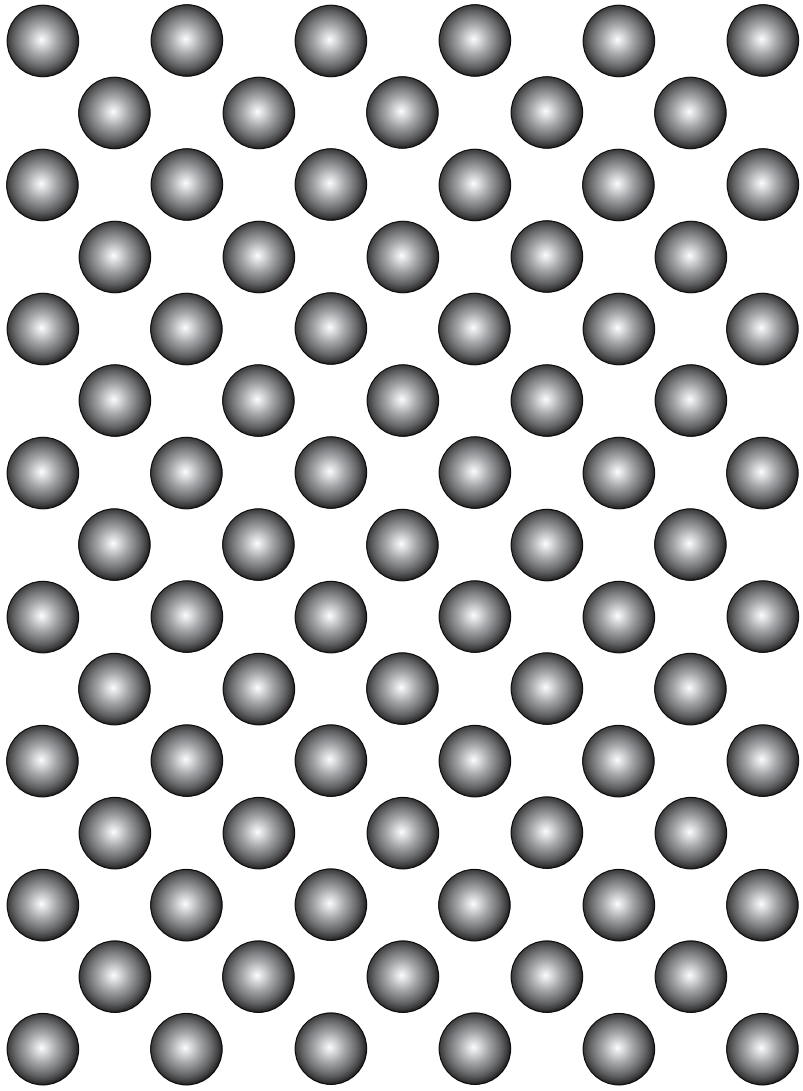
This is an argument to msroll. The cavity surface is written to a special file. This is useful if you want to look at only the cavity surface and not the outer surface. It can be used in conjunction with the -t flag, in which case the outer surface is written to the file specified with that flag.

Chapter ***10***

Solvent-Excluded Densities

Chapter Overview

- Computing Densities
- Computing a Polyhedron from a Density
- Displaying Densities



A. Computing Densities

The msform program can convert a polyhedral molecular surface into a three-dimensional array of cubes, each with a number between 0.0 and 1.0. The cubes entirely within the polyhedron receive a value of 1.0. The cubes entirely outside the polyhedron receive a value of 0.0. Cubes entirely inside the small surfaces bounding internal cavities also have zero densities. Cubes intersecting the polyhedral surface have densities somewhere inbetween 0 and 1. The density runs between 0.0 (no occupancy) and 1.0 (full occupancy). It is an occupancy density, not an electron density.

A polyhedron is converted into a density by: (a) superposing upon the polyhedron a cubical grid, (b) identifying those cubes that lie entirely within the polyhedron, (c) identifying those cubes that intersect the surface, and (d) for each intersecting cube, computing the volume of the part of the cube that lies within the polyhedron. The cubes entirely within the polyhedron are given a density of 1.0, and those lying partly within the polyhedron are given a density equal to the volume of the truncated cube divided by the volume of a whole cube.

The polyhedral surface serves as the input. The cube width is specified as a parameter.

```
msform -t polyhedron -w width -o
  density [-v inventor_file]
```

The density file format is AVS. See “Density” on page 72. Example:

```
% msform -t protein.vet -w 2.0 -o
  protein.avs
```

B. Computing a Polyhedron from a Density

```
msform -d density [-l level] -o
  polyhedron
```

The program computes a polyhedral molecular surface from a density. The level parameter specifies what level to contour at. The default is 0.5.

C. Displaying Densities

The mstran program will convert a molecular volume density file to a Silicon Graphics Inventor file, which can be displayed with the SGI ivview utility program.

```
mstran -d density -o inventor_file
  -c level -h hue
```

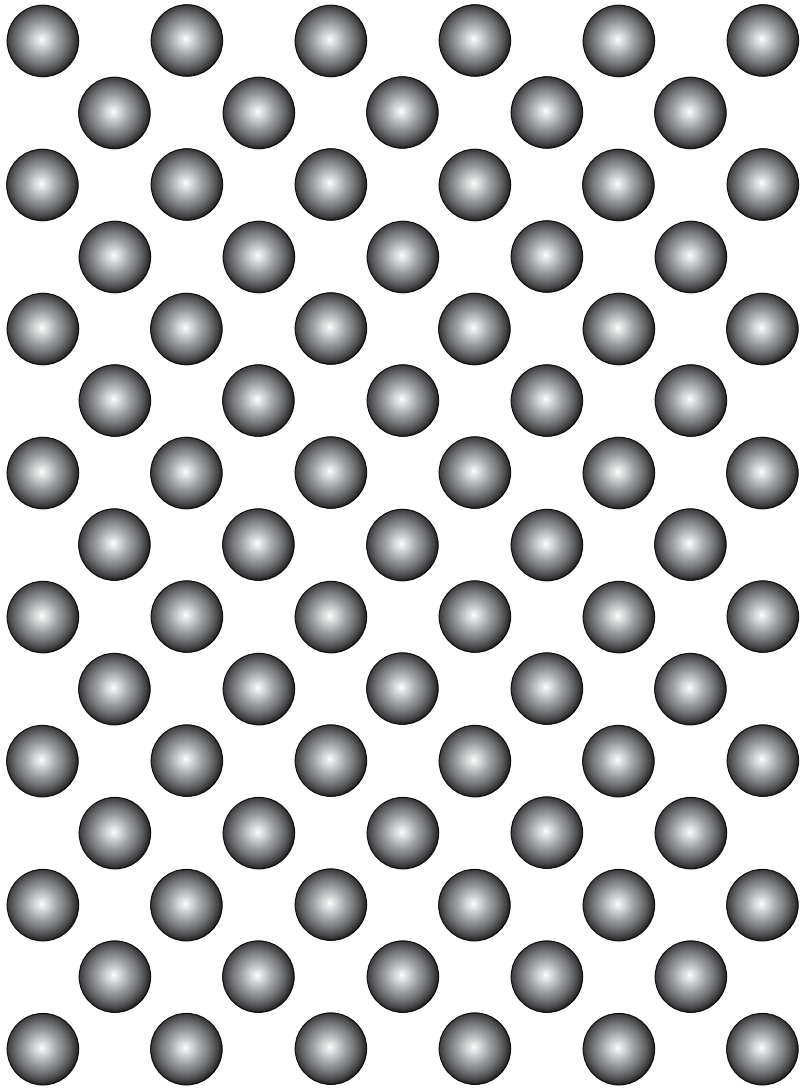
In the syntax the level parameter is a number between 0.0 and 1.0. A good value is 0.5. The hue is one of the predefined colors (red, green, blue, ...).

Chapter ***11***

Measuring Curvature

Chapter Overview

- Solid Angle
- Contouring Vertex Properties
- Contouring in Stereo
- Rotating Colored Polyhedra
- Measuring Density Curvature



A. Solid Angle

Solid Angle Computations

It can compute the curvature of the polyhedral surface by intersecting the polyhedron with spheres where the sphere centers are defined by the polyhedron vertices. By measuring the fraction of the region of each sphere that lies inside the solvent-excluded volume, one can quantitate surface region convexity and concavity.

Single Polyhdral Surface

```
% msform -t polyhedron -r radius -o
  polyhedron
```

This computes one solid angle for each polyhedron vertex. The first vertex value field (u) is replaced by the solid angle of the evaluation sphere centered at that point. See “Polyhedron” on page 72.

Three Concentric Spheres per Vertex

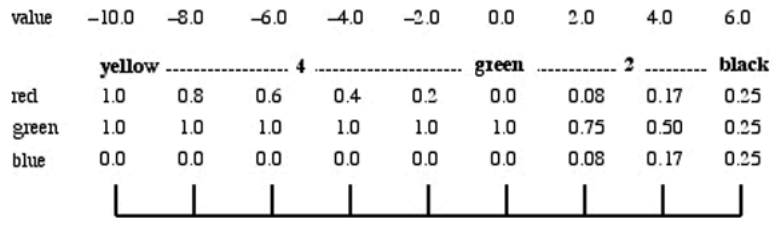
```
% msform -t polyhedron -r radius -x
  increment -o polyhedron
```

If the -x parameter is specified, the radius, radius + increment, radius + 2 * increment spheres will generate three values for the vertex (u, v, and w). If the analytical sphere-polyhedron algorithm fails, a numerical algorithm is used to compute omega. The program prints out the number of omegas computed numerically.

B. Contouring Vertex Properties

Vertex properties are represented by signed real numbers. There is no limit to the range. They are associated with vertices. Each vertex can have up to three properties, which are labeled u, v and w. The properties are converted to hues using a ramp, which is specified in the define_ramp command. Property values lying outside the range of the ramp are given the bottom or top hue number, as appropriate. The command "define_ramp yellow 4 green 2 black" gives this ramp (Figure 15).

Figure 15 Color Ramp



For rendered image, there is one band corresponding to each tick mark above. For plotted contours, there is one contour for each tick mark above.

C. Contouring in Stereo

Below are the of shell commands and script for endering and plotting a shape-colored model of a protein in stereo.

Shell commands for rendering and plotting surface shape

```
% msroll -m protein.pdb -t protein.vet
% msform -t protein.vet -r 5.0 -o protein.veto
% msdraw -i omega.script -r left.rgb -p left.ps -a 6.0
% msdraw -i omega.script -r right.rgb -p right.ps
% imgview left.rgb & ; imgview right.rgb &
% xpsview left.ps & ; xpsview right.ps &
```

The following script file is read by msdraw.

Script file omega.script

```
molecule protein
read_polyhedron protein.veto
# pre-defined color ramp yellow-green-blue
contour u 3.0 12.0 ygb
```

D. Rotating Colored Polyhedra

The mstran program will convert a vet format molecular surface polyhedron file to Silicon Graphics Inventor file, which can be displayed and rotated with the SGI ivview utility program.

```
mstran -t polyhedron -f u|v|w  -l
  lower -u upper -h color_ramp -o
  inventor_file
```

The syntax will color the polyhedron according to the real numbers in the vertex value fields. There are three vertex value fields (u, v and w). For example, msform

stores the solid angle in the u field. Unlike in msdraw, you cannot define your own color ramp, but instead must use one of the predefined color ramps: green, red, blue, tan, orange, magenta, and pink. The first three ramps (green, red, blue) give three distinct bands, the later ramps have more intermediate colors.

E. Measuring Density Curvature

Curvature and Local Symmetry

```
msform -t polyhedron -d density -r
  radius -o polyhedron
```

The program measures the shape of the density in the vicinity of each vertex of the polyhedron. A sphere is constructed centered at each vertex and the cubes partially or entirely within the sphere are summed. The polyhedron is modified, the local volume is written in the first (u) value field, the amount of two-foldness in the second (v), and the amount of three-foldness in the third (w) value field.

Sector Densities

```
msform -t polyhedron -d density -o
  polyhedron -z orange
```

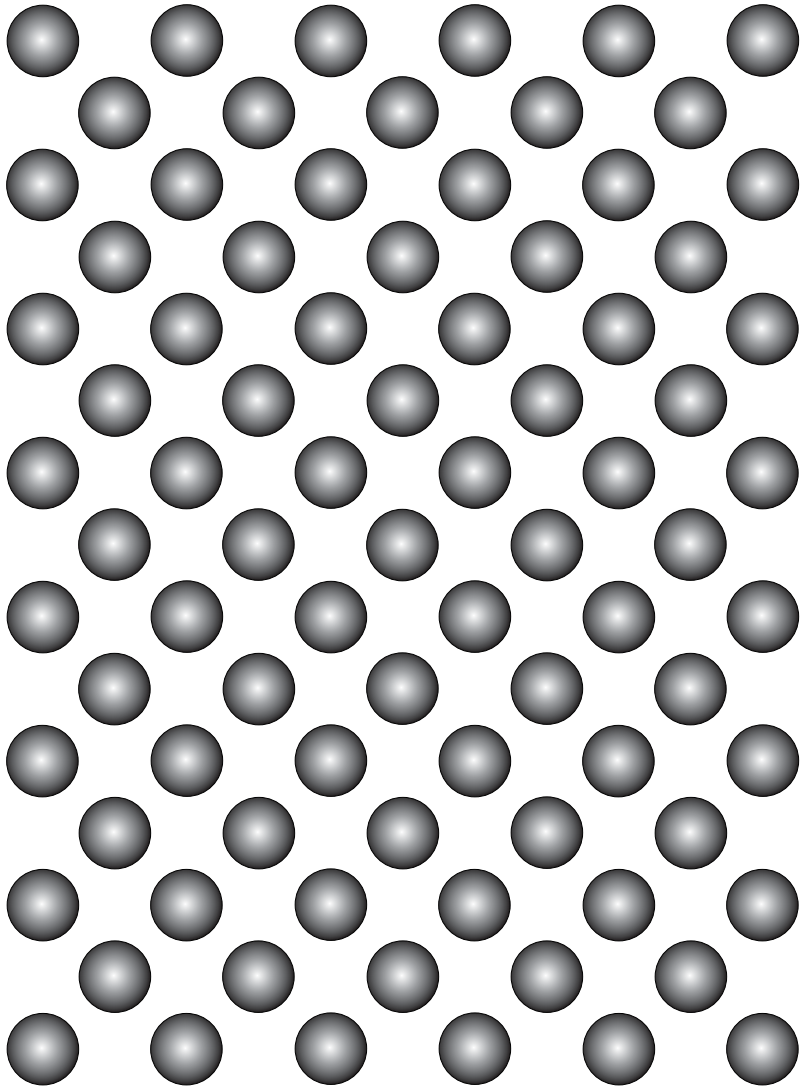
In this usage, the polyhedron vertices are given values based upon the -z orange file. An orange is an intersection sphere that is subdivided into sectors (wedges, slices). The orange file has one line, which has five fields: x, y, z, radius and number of sectors. The program computes the amount of each sector that lies inside the proteins solvent-excluded density. This gives a number between 0.0 and 1.0. There is one number for each sector. The axis of the orange is the normal vector at the specified center. The normal vector is computed, as usual, as pointing from the centroid of the sphere-protein intersection volume to the specified intersection sphere center. But these sector values are not written to the output disk file; instead, something more subtle is done. Every vertex on the polyhedral surface either lies outside the orange, or inside one of the slices of the orange. All vertices outside the orange sphere are given the value 500. All vertices falling into a slice are given the value (0.0 to 1.0) of the slice. The polyhedron, with these values in the u field, is written to the specified disk file. The polyhedron is rotated before written. It is rotated so that the normal vector points along the positive z axis.

Chapter ***12***

Interfaces of Protein Complexes

Chapter Overview

- Visualizing Protein Complexes
- A Pair of Polyhedra
- Surface Buried in an Interface
- Computing a Surface Between Two Densities
- Polyhedra Evaluated at an Interfacial Surface
- Densities Evaluated at an Interfacial Surface



A. Visualizing Protein Complexes

Interfaces can also be studied with msdraw. Figure 16 shows the rendering of a clipped complex. The trypsin enzyme is green, and the bovine pancreatic trypsin inhibitor protein is red. It was created with the shell commands below.

Shell commands for pancreatic trypsin complex

```
% msroll -f ptci.script -m ../pdb/2ptc.pdb -q ptci.pqms -n ptci
% msroll -f ptce.script -m ../pdb/2ptc.pdb -q ptce.pqms -n ptce
% msdraw -i complex.script -b 18.0 -r complex.sun sun -z 0.05 -s 600
% sdtimage complex.sun
```

Note that the msroll commands both rename the molecule and use script files. Each script file refers to the protein by its new name. The purpose of each script file is to reduce the set of atoms to just one of the two molecules: just the enzyme or just the inhibitor. They also exclude non-protein atoms (HETATM).

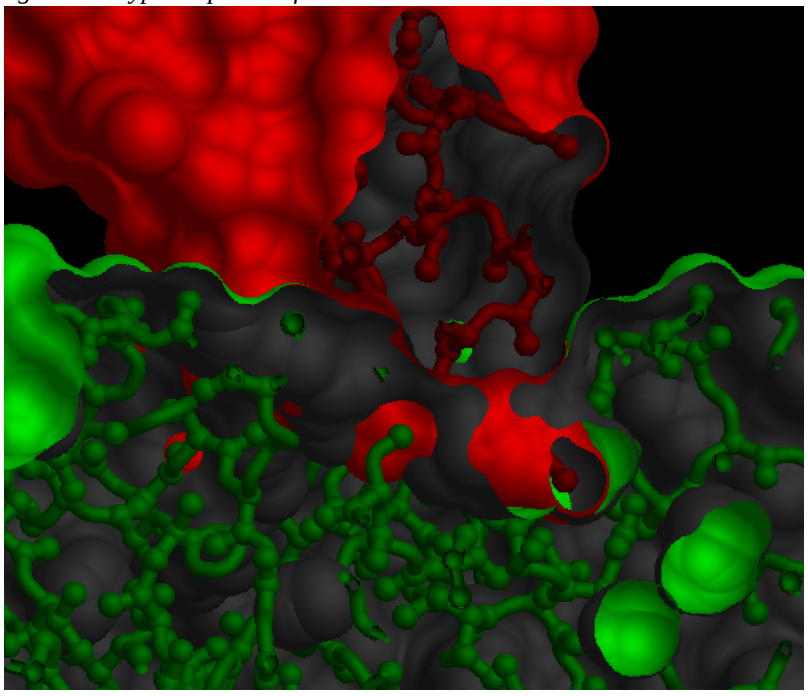
ptce.script

```
ptce -= pdb matches HETATM
ptce -= subunit matches I
```

ptci.script

```
ptci -= pdb matches HETATM
ptci -= subunit matches E
```

Figure 16 Trypsin-bpti interface



The msdraw script file for creating this illustration is rather long and is divided into two parts.

complex.script, part 1

```
rotation 45.0 0.0 1.0 0.0
molecule lyz lyz.pdb
lyz -= residue matches HOH
connect lyz
lyz color = red
ball_and_stick
atom_coloring
read_surface lyz.pqms
component_coloring white green blue
component_opacity 0.5 1.0 1.0
# global commands
shading_model 0.2 0.2 0.4 0.2 2.0
rotation 40.0 0.0 1.0 0.0
define_color dark_red 0.5 0.0 0.0
define_color dark_green 0.0 0.5 0.0
define_color dark_grey 0.25 0.25
```

complex.script, part 2

```
# enzyme
molecule ptce ../pdb/2ptc.pdb
source ptce.script
tolerance 0.25
elbow 2.5
connect ptce # creates sticks
ball_and_stick
uniform_coloring dark_green
read_surface ptce.pqms
uniform_coloring green dark_grey
# inhibitor
molecule ptci ../pdb/2ptc.pdb
source ptci.script
tolerance 0.25
elbow 2.5
connect ptci # creates sticks
ball_and_stick
uniform_coloring dark_red
read_surface ptci.pqms
uniform_coloring red dark_grey
```

B. A Pair of Polyhedra

```
msform -t polyhedron -n polyhedron
  -r radius -o polyhedron
```

This computes solid angles for two polyhedra at once. This command uses two polyhedra (-t and -n). The first (-t) polyhedron is the one that will have values assigned to its vertices, and will be written as output (-o). The three vertex value fields will contain three solid angles. The first will be the solid angle subtended by the primary polyhedron as measured by a sphere centered at a vertex of the primary polyhedron. The second will be the solid angle subtended by the opposing polyhedron as measured by a sphere centered at this same vertex of the primary polyhedron. The third number will be the sum of the first two. It will be close to four pi steradians for a close fit.

C. Surface Buried in an Interface

```
msform -t polyhedron -n polyhedron
  -r radius -x extension -o
  polyhedron
```

This feature is used to identify surface buried in a protein-protein interface. The first (-t) polyhedron is the one that will have values assigned to its vertices. The number specified as the extension is the distance the intersection sphere is moved out along the normal vector. If one specifies for both the radius and extension parameters the same number as the probe radius used to compute the two protein surfaces, then the intersection sphere will have the same center and the same radius as the probe tangent to that vertex. The first (u) and third (w) fields in the primary polyhedron vertices will be given the value zero, and the second field (v) will be the solid angle of the intersection of the sphere with the second (-n) polyhedron. If that value is positive, that indicates that the probe collides with the opposing protein, and so that vertex is buried. The values are written into the value fields of the output (-o) polyhedron.

D. Computing a Surface Between Two Densities

```
msform -d density -y density [-r
  radius] [-s 0|1] -o polyhedron
```

The program computes a polyhedral interfacial surface between the two given densities. The two densities need to have the same cube width. A third density is temporarily created. The third density is the first density minus the second density. The contour level is always 0.0. A sphere is constructed centered at each cube center and the cubes partially or entirely within the sphere are used. The radius parameter specifies what radius to use for computing the local volume and normal vector. The smooth=1 parameter means use smoothed density, where the average value of the den-

sity inside the sphere of the specified radius is used. For smooth=0, the value of the density at the sphere center is used. The cubes near the interface will have values near 0.0 and vectors that point from the first density to the second. If the interface is not a close fit, or if the cubes are small enough to fit in the gaps between the densities, there will be holes in the interfacial surface, unless smoothing is used.

Computing an interfacial surface between two protein densities is done by running msform several times. The output of the final run of msform is run through mstran to convert it to SGI Inventor format.

Shell commands for interfacial surface

```
% msform -t protein.vet -w 2.0 -o protein.avs
% msform -t ligand.vet -w 2.0 -o ligand.avs
% msform -d protein.avs -y ligand.avs -r 3.0 -o interface.vet
% mstran -t interface.vet -o interface.iv
```

E. Polyhedra Evaluated at an Interfacial Surface

```
msform -t polyhedron -n polyhedron
  -h polyhedron -r radius -o
  polyhedron
```

The fifth usage style computes three solid angles, one for one protein, one for another protein, and their sum, as did an earlier style, but differs in that there are now three polyhedra. The first (-t) polyhedron is used solely to define vertices to center intersection spheres at. The second (-n) and third (-h) polyhedra are the ones that are intersected with the sphere. The output polyhedron is the first polyhedron, but with three vertex values set. The purpose of this feature is to use as the first polyhedron not a protein surface polyhedron, but an interfacial surface computed by a previous run of msform (see above).

F. Densities Evaluated at an Interfacial Surface

```
msform -d density -y density -t
  polyhedron -r radius -o
  polyhedron
```

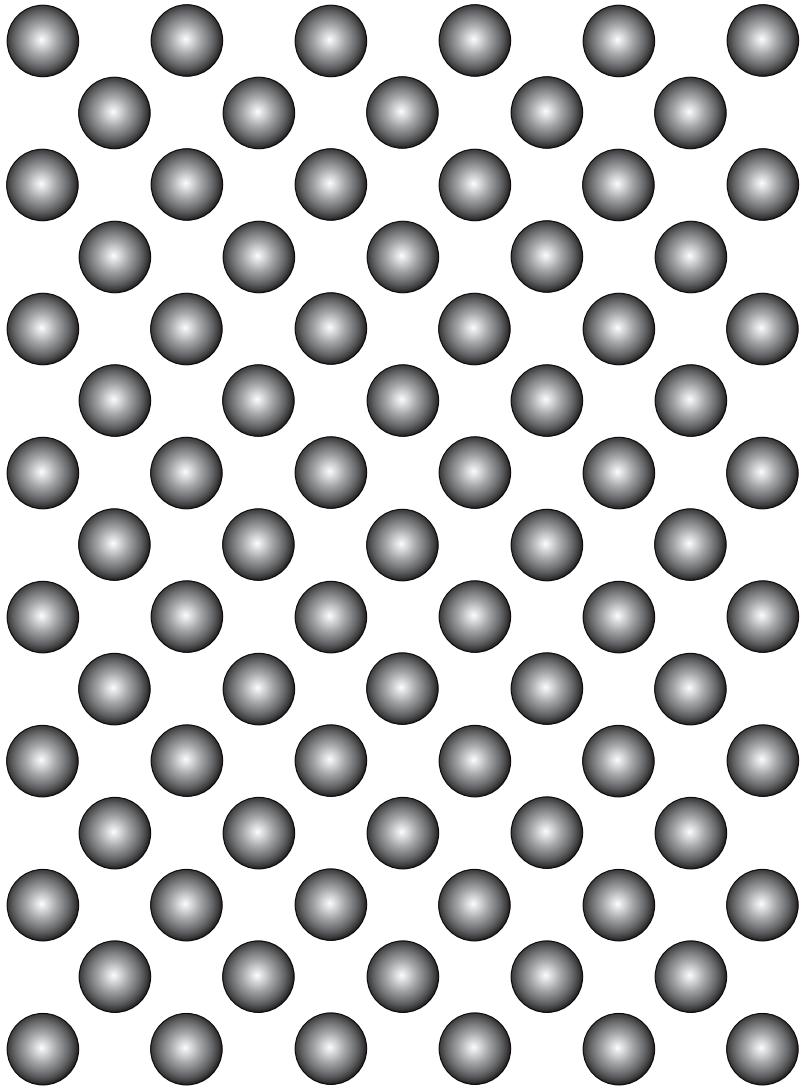
The program measures the shape of the two given densities (-d, -y) in the vicinity of each vertex of the given (-t) interface-bisecting polyhedral surface. The values are written in the value fields of the output (-o) polyhedron. The two densities need to have the same cube width. A sphere is constructed centered at each vertex and the cubes partially or entirely within the sphere are used. The interface polyhedron is modified, so that the local volume normal vector stored into the normal vector fields, and the shape in the three value fields. The first value (u) is the amount of two-foldness for the first density, the second value (v) is the amount of two-foldness in the second density, and the third value (w) is the local volume of the sum of the two densities. It will be close to 1.0 for regions where the fit is tight. The two-fold amounts should be similar for a close fit.

C h a p t e r ***1 3***

Command-Line Arguments

Chapter Overview

- msroll
- msform
- msdraw
- mstran



A. msroll

The execution time increases linearly with the number of atoms, and as the cube of the probe radius. The arguments are are given below.

Error Message File

`-e` *`file`*

The error message file (also contains informative messages).

Grid Spacing Parameter

`-g` *`spacing`*

This command specifies the fineness of the probe-collision grid; the neighbor grid is twice this. The purpose of these grids is to speed up neighbor-finding (cube grid) and probe collision checks (bit grid). It is not possible to use one grid, but not the other. The grids are optional. If not given, no grid will be used. There is a certain amount of overhead involved with the grids, so it is not advisable to use them unless the molecule has at least one thousand atoms.

Molecule Input File

`-m` *`file`*

The atomic coordinates, in pdb format.

Name

`-n` *`name`*

The name of the molecule, overriding the one inferred from the name of the pdb file.

Probe Radius

`-p` *`radius`*

The minimum allowed value is 0.0. This gives a van der Waals surface. Large values cause the program to be very slow. Default=1.5.

Piecewise Quartic Output File

`-q` *`file`*

The binary surface file is written to disk. See “PQMS Disk Format” on page 73.

Atomic Radii Input File

`-r` *`file`*

The van der Waals radii. See “Atomic Radii” on page 70.

Polyhedron Output File

`-t` *`file`*

The polyhedral surface file is computed and written to disk. This file is read by msdraw. See “Polyhedron” on page 72.

Cusp Intersections

`-x` *`integer`*

There is user control over cusp trimming. The maximum order of cusp edge intersections trimmed can be specified by an integer, where the integer is 0, 1, 2 or 3. Zero is recommended for rerunning large proteins that generate cusp trimming error messages on the first run.

B. msform

There are many things this program can do, and what it does is determined by which command line flags are present. Typing just the program name will print out all the different usages, but without the explanatory paragraphs below. The flags are:

`-t` *`file`* *(in vet format)*
`-n` *`file`* *(second polyhedral input file)*
`-h` *`file`* *(third polyhedral input file)*
`-d` *`file`* *(density input file)*
`-y` *`file`* *(second density input file)*
`-o` *`file`* *(polyhedron or density)*
`-l` *`level`* *(default level = 0.5 for one density)*
`-x` *`radius`* *(extension or expansion)*
`-w` *`width`* *(of density cubes)*
`-r` *`radius`* *(of intersection sphere)*

C. msdraw

Purpose

The program reads the surface files computed by msroll and creates three kinds of output files: (a) an image file of pixels that shows a smooth surface with hidden-surface elimination, (b) plots with hidden-line elimination, and (c) vector files in SGI Open Inventor format.

Output File Names and Formats

```
msdraw -i scriptfile -r rasterfile
  [format] -p plotfile [format] -v
  vectorfile
```

Images can be written in 3 formats: sgi, sun and avsimage. The default format is "sgi" image, which is a 24-bit color format. For sun or avsimage formats, add the word sun or avsimage after the file name. The "sun" format produces the sun rasterfile format, which can be displayed under X Windows using the xview or xloadimage utilities. This is an 8-bit color format using a color lookup table. For AVS image format, use

"avsimage" (for AVS field format, use "avsfield"). Plot output file format is either PostScript (the default) or "hpgl", which is HP-GL (Hewlett-Packard Graphics Language). MSDraw reads a script file in order to define the scene. The script file name is given after the -i flag. See “Scene Filename Argument” on page 62.

Image Size Parameter

`-s image_size`

The image is always square, and the integer specifies its width (default 512).

Tilt of Clipping Plane

`-x tiltx`

`-y tilty`

These flags are used to specify the tilt of the clipping plane. They can be in the range 0.0 to 1.0. Let x and y denote the values specified by the -x and -y flags. Then (x, y, 1.0) gives a vector perpendicular to the clipping plane. For example, an x value of 1.0 gives a tilt of 45 degrees. It is not possible to specify negative values for these two flags.

Border Parameter

`-b border`

The argument is a floating-point number in angstroms. The window is shrunk (new in 3.9) in all four directions (left, right, bottom, top) by the amount given in the border command. Default border value: 0.0.

Alignment Parameter

`-n alignment`

Causes the window bounds to be rounded outwards to a multiple of the given parameter. Default=1.0.

Title String

`-t title`

Affects plotting only. This title will be assigned as a comment in the PostScript output file.

Header Type

`-h header`

The header can be either inventor or vrml. The inventor option is the default and causes the SGI Open Inventor header to be written, while vrml causes 3D output files to be written with a VRML header.

Z Clipping

`-z zclip`

The clipping plane is assumed to be parallel to the xy plane, and the argument must be in the range -1.0 to 1.0. The number is not in angstroms, but represents a fraction of the z range of the molecules. That is, a value of 1.0 clips nothing, a value of 0.0 clips through the middle, and a value of -1.0 clips everything. For plotting and Inventor output, triangles that cross the clipping plane will be truncated, and the new line that is introduced will be given the vet file's triangle color, or the polyhedron color, if specified. Bonds crossing the clipping plane will also be cut. See “No Clipping” on page 67.

Stereo Angle

`-a stereo_angle`

This command is used for producing stereo pairs. The molecules, clipping plane (if present) and the light source will be rotated by the specified angle. It may be necessary to use a border and perhaps also an alignment parameter to make sure that the two images are drawn to the same scale.

Shading Type

`-g shading`

The argument is flat for flat or faceted triangles, and smooth for phong shading (default).

Fineness Parameter

`-f fineness`

This parameter rarely needs to be specified. The argument is a floating point number in the range 1.0 to 2.0, controlling the fineness of the sampling in the rendering algorithm. If the fineness is too low, a few pixels will be vacant; if it is too high, the computation will take longer. Default fineness: 1.35.

D. mstran

This program will convert vet format molecular surface polyhedra and molecular volume density files to Silicon Graphics Inventor files, which can be displayed with the SGI ivview utility program.

```
% mstran -t polyhedron -f u|v|w  -l
  lower -u upper -h color_ramp -o
  inventor_file
```

This syntax will color the polyhedron according to the real numbers in the vertex value fields. There are three vertex value fields (u, v and w). For example, msform stores the solid angle in the u field. Unlike in msdraw, you cannot define your own color ramp, but instead must use one of the predefined color ramps: green, red, blue, tan, orange, magenta, and pink. The first three

ramps (green, red, blue) give three distinct bands, the later ramps have more intermediate colors.

```
% mstran -d density -c level -h hue
   -o inventor_file
```

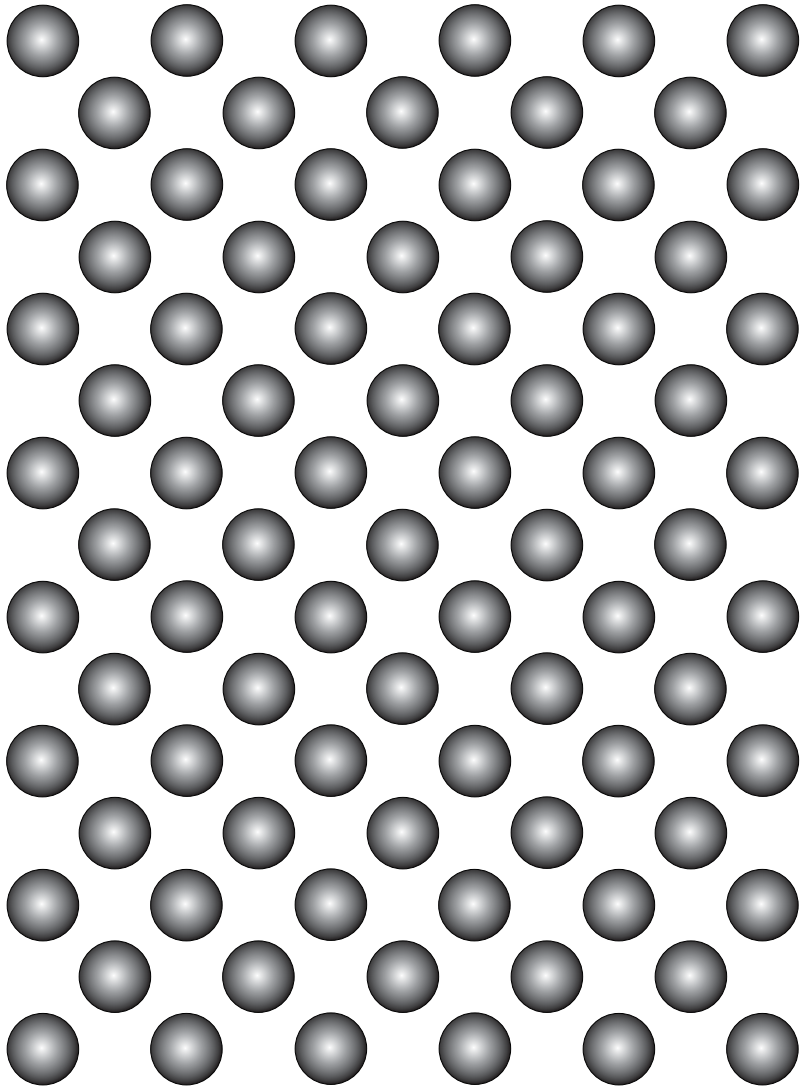
In this syntax the level parameter is a number between 0.0 and 1.0. A higher number will create more cubes. A good value is 0.5. The hue is one of the predefined colors (see msdraw) such as red, green, blue, cyan, magenta, yellow, white, black, grey, etc. All the cubes will be this color.

Chapter ***14***

Setting Atom Fields

Chapter Overview

- Atom Fields
- Sets of Atoms
- Atom Field Files
- Selecting Sets of Atoms
- Setting Atom Fields



A. Atom Fields

Some of these fields require explanation. The character string fields may be at most 6 characters long. The "atom" field means atom name in the PDB file; the "kind" field means a user-defined atom type, such as C2 for methylene carbon; the type field means the integer used to choose a van der Waals radius from the radii file; the "element" field is the atomic number. The "anumber" field is not the atomic number, but is the atom number in the PDB file. The "radius" field is the van der Waals radius; the "covalent" field is the covalent radius and is used in determining connectivity. The "ball" field is the radius for ball-and-stick figures in msdraw. Simply reading an input PDB file sets the values of many of these fields. The values of the other fields are set by various algorithms that the program uses. They can be overridden by the user. Some fields such as kind are null if not set by the user. The ball radii are set to 0.5.

Table 4: Atom Fields

Field Name	*Type*	*Examples*
angle	real	1.1
anumber	integer	45
atom	character string	CD2
ball	real	0.3
center	3 reals	3.5 4.2 -8.8

Table 4: Atom Fields

Field Name	*Type*	*Examples*
color	integer	2
covalent	real	0.77
density	real	4.0
element	integer	7
kind	character string	C3
occupancy	real	1.0
opacity	real	0.5
pdb	character string	HETATM
radius	real	1.75
residue	character string	ASN
rnumber	integer	343
sequence	character string	343B
subunit	character	A
suffix	character	B
tfactor	real	3.18
type	integer	9

B. Sets of Atoms

The atoms are numbered starting at 1. The numbers are based upon the order the atoms are read in the input file. If there is more than one molecule in the input file, the numbering is NOT restarted at one for the second molecule. So each atom has a unique number.

Each set is described by a linked list of ranges, and each range is described by begin and ending atom numbers. For example: 23-45, 58-101, 189-212. There are operations for unions and intersections of sets. The purpose of these atom sets is to set properties of atoms, such as colors, and to excluded some atoms from the calculations.

C. Atom Field Files

A file of atom-field commands is read by the "-f filename" argument to msroll. These commands are also read after the "molecule" command of msdraw and are applied to that molecule.

D. Selecting Sets of Atoms

Define sets of atoms based upon input field values from the PDB file:

setname = field operator constant

where operator is one of:

Operators

```
matches
  initial_substring
inside (for spheres)
outside (for spheres)
above (for planes)
below (for planes)
== numeric or string
!= numeric or string
< numeric
<= numeric
> numeric
```

The "matches" operator is used for determining atom type from atom name: "atom matches C" is true for any atom name beginning with the letter C. Case is ignored. The "inside" and "outside" operators select atoms inside or outside a sphere. The "above" and "below" operators select atoms on one side or the other of a plane. The "above" operator selects atoms on the side of the plane in the direction the normal vector. Note that only the "=" and "!=" can be used for character strings; the greater than and less than operators are used only for numbers.

E. Setting Atom Fields

Define output field values for atoms in these sets by field assignment commands:

```
setname field = constant
```

Further commands related to atom sets and atom properties are shown in the list below. The +=, -= and *= assignment commands are similar to the simple = assignment command. The clear command causes a set to become empty. The connect command creates bonds for atoms in the specified set. Two atoms are connected if their separation is less than the sum of their van der Waals radii, plus the tolerance. The tolerance should be specified before the connect command. The disconnect command removes all bonds from an atom in the first set to an atom of the second set. The bond radius is set by the bond_radius command. The general ball radius can be modified by the ball_radius command.

Atom Set Commands

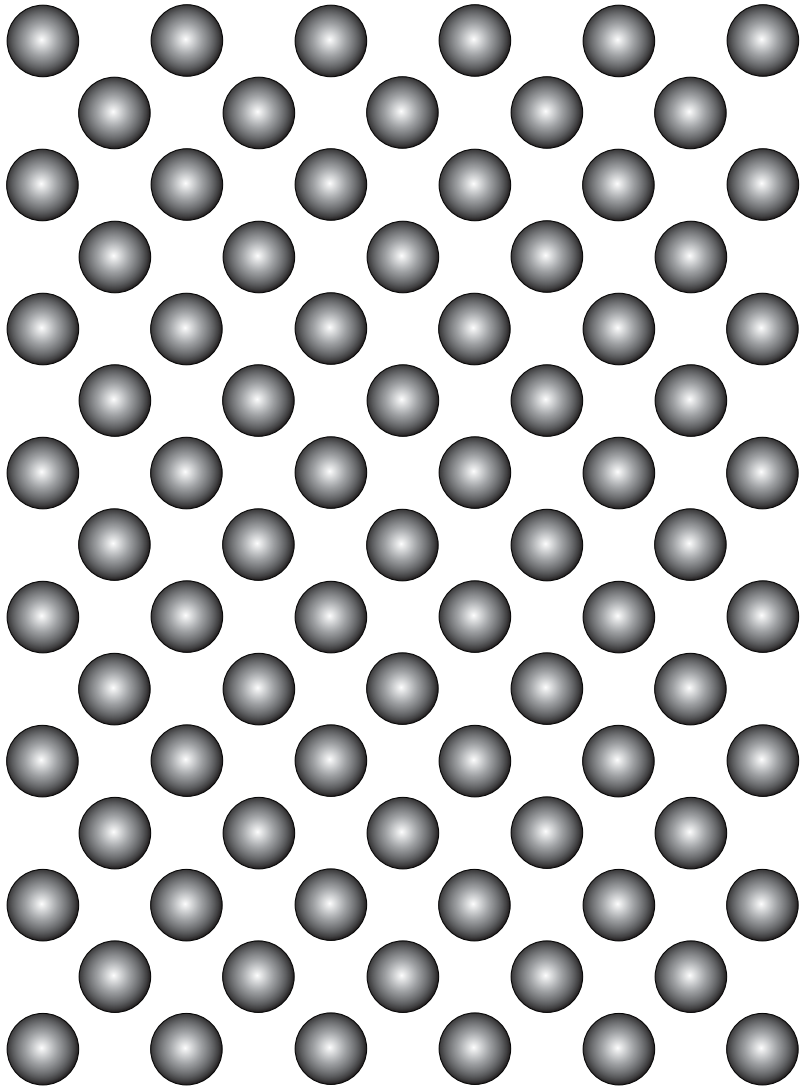
```
setname = field operator
  constant
setname += field operator
  constant
setname -= field operator
  constant
setname *= field operator
  constant
set2 = set1
set3 = set1 * set2
set3 = set1 + set2
set3 = set1 - set2
sphere spherename x y z radius
plane planename x y z xn yn zn
clear setname
connect setname [tolerance]
```

Chapter ***15***

Scene Script

Chapter Overview

- Script Filename Argument
- Command Syntax
- Global Script Commands
- Commands for Reading Molecular and Surface Data
- Script Commands for Contouring



A. Scene Filename Argument

`-i` *`script_file`*

The script file contains information for drawing one or more molecules.

B. Command Syntax

A command cannot run to more than one line. There cannot be two commands on one line. Each command begins with a keyword defining the type of command. The format is free format. Any characters after a # will be ignored and can be used as comments. Blank lines are ignored. Keywords must be in lower case and cannot be abbreviated. In the command descriptions below, optional command arguments are enclosed in square brackets. When either of two alternatives are acceptable, they are separated by a pipe (|) symbol.

C. Global Commands

These commands affect the whole picture. There are three commands that, if present, should appear only at the top of the script file: rotation, shading_model, and light_source. They are optional.

Rotation

`rotation` *`angle xn yn zn`*

This specifies an angle (in degrees) and an axis. The vector need not be of unit length.

Shading Model

`shading_model` *`depth ambient diffuse specular exponent`*

The arguments are all floating-point numbers. The first four numbers must add up to 1.00. They give the fraction of the intensity (gray) levels to assign to the relevant function. The last number is the exponent of the cosine for the specular reflection (highlight). The depth parameter gives the amount of depth cueing. The ambient parameter gives the brightness of the object at the regions farthest away from the direction of the light source. Default:

```
shading_model 0.0 0.0 0.8 0.2 2.0
```

Light Source Direction

`light_source` *`xn yn zn`*

The direction vector of a light source at infinity. The vector does not need to be of unit length. Default: 0.5 0.5 1.0

After these commands, if present, come the descriptions of each molecule. Each molecule is introduced with a molecule command, followed by commands to

read surface output files from msroll and to control how the objects are displayed.

Overlap Hue

`overlap_hue` *`hue`*

The hue for interpenetrating solvent-excluded volumes for two (or more) surfaces, each with a solid_shade command. Default: white.

D. Molecule Commands

Molecule

`molecule` *`name pdb_file`* [*`radii_file`*]

The name should be less than 40 characters. It becomes the name of the set of atoms read in from the pdb file, and is used in the connect command (see below). The radius file is optional; if missing, radii will be taken from the ms.c file.

Read PQMS Surface

`read_surface` *`filename`*

This command will render the molecular surface in the specified binary surface file.

Read Polyhedral Molecular Surface

`read_polyhedron` *`filename`*

This command will render the polyhedral molecular surface, which should be an output file from the msroll program.

Normal Vector Length

`normals` *`length`*

Vertex normals of the specified length will be drawn. There will be one vector for each vertex of the polyhedron. Each vertex normal is the average of the triangle normals of the triangles that meet at the vertex. Short vectors (around 0.1) will give the look of a dot surface. As always, hidden-line elimination is performed. The command should occur sometime after the read_polyhedron command. It affects the plot only, not the rendered image.

Ball-and-Stick Model

`ball_and_stick`

This command will render a ball and stick model for image output and it will draw a stick figure for plot output. The atomic coordinates are taken from the current molecule file, the connectivity from that created by the prior connect commands for this molecule. The radii of the balls are determined by the ball field of the

atoms of the current molecule, which in turn is set by the atom-property commands pertaining to the current molecule. This command will also render a tube model, in some circumstances. If there are exactly two bonds to an atom, they have equal radii, and the value of the elbow parameter (default = 2.0) is greater than 1.0, then the atom will be drawn as an elbow (part of a torus), instead of as a sphere. This will also occur if there are more than two bonds to the atom, provided that the two fattest bonds have radii that are equal and larger than the radius of the third fattest bond to the atom. To disable tubes, so that you always get only balls for atoms, set elbow to 0.0 (see the elbow command, below). The color argument affects the plotted output, only. If the color is not specified, the plotted model will be black. The coloring for the rendered model is taken from the color fields of the atoms as specified in the atom property file commands pertaining to the current molecule.

Define Color

```
define_color dark_green 0.0 0.5 0.1
```

This command defines new colors. The color name can be up to 15 letters. There can be up to 256 colors.

Input Coloring

```
input_coloring
```

This command applies only to polyhedral input files. The coloring is taken from the hue field in the edge and triangle records of the vet file. This command should occur after the read_surface or read_polyhedron command.

Atom Coloring

```
atom_coloring
```

This command has no parameters. The hues are taken from the color field of the atoms, as set by the atom property commands. This command should occur after the read_surface or read_polyhedron command.

Uniform Coloring

```
uniform_coloring hue1 [hue2]
```

The first hue is for the outer side and the second for the inner side, which is visible only if there is clipping or transluscency. This command should occur after the read_surface or read_polyhedron command.

Component Coloring

```
component_coloring hue1 hue2 hue3
```

The first two arguments refer to the outer and inner sides of the outer surface; the third argument refers to the inner (cavity) surfaces. This command should occur after the read_surface or read_polyhedron command.

Shape Coloring

```
shape_coloring contact reentrant
shape_coloring convex saddle
  concave
```

This command should occur after the read_surface or read_polyhedron command.

Atom Opacity

```
atom_opacity
```

The atom opacities are set by atom property commands. This command should occur after the read_surface or read_polyhedron command.

Uniform Opacity

```
uniform_opacity opacity1 [opacity2]
```

The uniform opacity argument is in the range 0.0 to 1.0. A value of 0.0 will make the surface invisible, a value of 1.0 will make it opaque. The values 0.25, 0.50, and 0.75 cause every fourth, every other, and three out of four pixels to be drawn, respectively. Other values are rounded to the nearest of these five values. The words transparent (0.0), translucent (0.5) and opaque (1.0) can also be used instead of numbers. This command should occur after the read_surface or read_polyhedron command.

Component Opacity

```
component_opacity outer inner
  cavity
```

The component opacities are in the range 0.0 to 1.0. This command should occur after the read_surface or read_polyhedron command.

Shape Opacity

```
shape_opacity contact reentrant
```

The opacity arguments are reals in the range 0.0 to 1.0. This command should occur after the read_surface or read_polyhedron command.

Solid Shade

```
solid_shade shade
```

The shade argument should be in the range 0 to 255. This command should occur after the read_surface or read_polyhedron command.

Plot Line Width

```
outer_line_width real
inner_line_width real
cavity_line_width real
bond_line_width real
```

These four commands affect plotting only. This line width is a factor that multiplies the basic line width. The basic line width depends upon the output format. It is 0.35 mm for HP-GL, and 1 point for PostScript. The outer and inner parameters refer to the outer and inner sides of the polyhedral surfaces. The cavity width applies to internal cavities, and the bond width to the stick figure. The defaults are 1.0 (outer), 0.2 (inner), 1.5 (cavity) and 1.5 (bond). These commands should follow the molecule command, but occur before the read_polyhedron and ball_and_stick commands.

Elbow Parameter

```
elbow real
```

The argument is a floating point number in the range 0.0 to 10.0, controlling the ratio of the elbow torus radius to the bond radius. If the value is below 1.0, no elbows will be drawn. The purpose of the command is to control the rendering of the tube model. Default: 2.0. This command should occur after the molecule command, but before the ball_and_stick command.

Ball Radius Parameter

```
ball_radius real
```

The argument is a floating point number that specifies the ball radius for the ball and stick model. It is possible to override this value with atom property commands assigning values to the atom's ball field (see below). Default: 0.5. This command should occur after the molecule command, but before the ball_and_stick command.

Bond Radius

```
bond_radius real
```

The argument is a floating point number that specifies the cylinder radius for the ball and stick model. Default: 0.3. This command should occur after the molecule command, but before the ball_and_stick command.

Blanking Polyhedron Edges

```
blank
```

This command causes the polyhedron edges not to be drawn, but it still hides lines behind it. It should occur after the read_polyhedron command.

No Clipping

```
no_clipping
```

This command refers to just the current object. It should occur after the molecule command and some of the commands. It will cause the object not to be clipped by the clipping plane, if specified. It is generally used to have the chemical model protrude from a clipped surface. The command should occur after the command defining the object (ball_and_stick, polyhedron or surface) that is not to be clipped.

E. Contouring Commands

The program computes contours for the shape function (u, v or w). It will also contour the x, y or z coordinates, to produce a stack of planar contours. The contouring can be output in one of three ways: (a) as bands of color in a rendered image, (b) as contour lines in a plot, or (c) as three-dimensional contours in an SGI Open Inventor file.

Define Ramp

In the define_ramp command, there must be at least two colors mentioned, up to as many as will fit on a line. The total number of color levels must be less than 256. Examples:

```
define_ramp  tramp  orange 10 green
define_ramp  onramp  yellow 3 green
  2 cyan 2 blue 2 navy
```

Some ramps are predefined:

```
define_ramp red yellow 0 red 0 blue
define_ramp green yellow 0 green 0
  blue
define_ramp blue sky 0 blue 0 navy
define_ramp tan yellow 2 tan 2
  brown
define_ramp orange yellow 2 orange
  2 red
define_ramp magenta white 2 magenta
  2 purple
define_ramp pink white 2 pink 2 red
```

Contour

```
contour function from to ramp
```

The from and to are real numbers (have a decimal point) and the ramp must have been defined by a define ramp command (see above). The only functions currently allowed are: u, v, w, x, y, z. The u, v and w are the first, second and third vertex value fields. The x, y and z are simply the vertex cartesian coordinates.

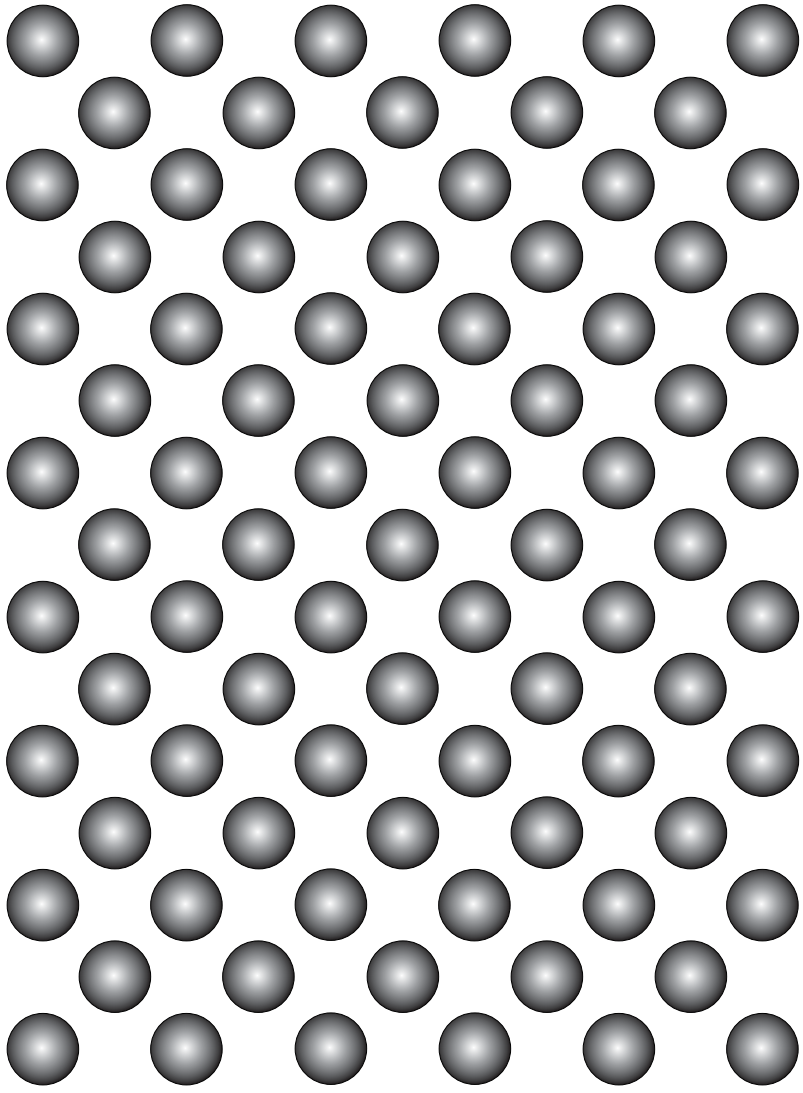
For plotting and vector output, there will be a contour polyline at each ramp level. The ramp specifies the number of values and the color of each, but the contour command specifies the range of values (from -- to). For rendering the polyhedron, there will be a bands of color. The maximum number of contour levels is 255.

Chapter 16

Disk File Formats

Chapter Overview

- Atomic Radii
- Atom Name Patterns
- Atomic Areas
- Volumes and Cavities
- Surface Points
- Polyhedron
- Density
- PQMS Disk Format
- Image File Formats
- Vector File Formats



A. Atomic Radii

Table 5: Atomic Radii

Type	*van der Waals*	*covalent*	*Name*
1	1.60	0.57	O=C
2	1.70	0.66	OH
3	1.60	0.57	OCO
4	1.65	0.70	NH
5	1.70	0.70	NH2
6	1.75	0.70	NH3
7	1.85	0.77	CH
8	1.90	0.77	CH2
9	1.95	0.77	CH3
10	1.80	0.67	CAr
11	1.90	0.70	CHAr
12	1.90	1.04	S
21	1.50	1.50	Metal
31	2.00	0.77	CAlNu
32	1.77	0.67	CArNu
33	1.40	0.66	OSuNu
34	1.64	0.57	O=CNu
35	1.64	0.57	OPONu
36	1.55	0.65	NArNu
37	1.86	0.70	NAlNu
38	1.80	0.95	P
99	1.00	0.50	H

The table above gives the default atomic radii that the msroll and msdraw programs use. The format is space-delimited. The atom type in the first column must correspond to the predefined atom types in the program, or to the atom types in a user-specified pattern file (see below). The first real number is the van der Waals radius and the second is the covalent radius. The first field in the radius file and the third field in the pattern field are the atom type, which provides a bridge between the two files.

B. Atom Name Patterns

Table 6: Atom Name Patterns

Residue	*Atom*	*Type*	*Name*
*	P?	38	P
*	P??	38	P
*	N	4	NH
*	CA	7	CH
*	FE	21	Metal
ASP	OD2	3	OCO
CYS	SG	12	S
GLU	CG	8	CH2

The default pattern table contains 418 rows, so it will not be reproduced in this manual. It can be found in the ms.c file. The format is blank-delimited. The table above gives only a representative sample of rows. A question mark matches any one character. A string consisting of only an asterisk matches any string. There is no feature for, C*, for example, to match any string beginning with C. You would have to use as series of lines: C, C?, C??, C???, C????. The residue

and atom names can be at most 5 characters. The fourth field, the atom kind, is currently not used. The last match found is used, so less-specific matches should be listed first.

C. Atomic Areas

This file gives the solvent-accessible areas of individual atoms. There is one line per atom. There are four floating point numbers per line. The first floating point number is the contact area, the second is the reentrant area, the third is their sum (molecular area), and the fourth is the accessible area. The accessible area describes the trace or trajectory of the center of the probe sphere, and has often been related to physical chemistry. Consult Fred Richards' publications for a discussion of molecular areas and volumes.

D. Volumes and Cavities

There is one line for each surface component. Cavities have negative volumes. The total solvent-excluded volume of the molecule is listed last. This file gives the total contact, total reentrant, total molecular (contact+reentrant), and total accessible areas of the molecule, and the solvent-excluded volume of the molecule. This volume does not include the volumes of any internal cavities (packing defects large enough to accommodate the probe). These summary areas and volumes are followed by a table of centroids, volumes, and areas for each surface component. A surface component is a connected piece of surface. For small molecules there will be only one component. For large molecules, such as proteins, the first component is the outer surface, and the following components are surfaces bounding cavities. The volume for the outer surface includes the volumes of all the cavities. The cavity volumes are always negative, which is how they may be distinguished from a surface bounding a separate, small molecule located a few angstroms from the protein surface. Thus simply summing the volume column of this table will give the solvent-excluded volume given at the beginning of the file. That is, the cavity volume is not considered to be part of the solvent-excluded volume of the protein. The component areas are molecular areas (contact area plus reentrant area) followed by accessible areas.

E. Surface Points

The surface output file has one line per surface point. The surface points are in ascending atom number order, with the contact points of each atom preceding the reentrant points. The output format is similar to that of MS (QCPE program #429), except that: (a) the

second and third atom numbers of reentrant surface points are not correct, (b) shape number is always 2, even when the probe touches three atoms, (c) contact and reentrant points are written to a single file, and (d) in atom number order, so there is no need for sorting or merging.

F. Polyhedron

The first record has three integers ("%d %d %d"). The first integer is the number of vertices, the second integer is the number of edges, and the third integer is the number of triangles. The vertex coordinates then follow. Each line has the C format: "%12.6f %12.6f %12.6f %7.4f %7.4f %7.4f %10.6f %10.6f %10.6f %3d %5d %3d". The first three fields are the vertex coordinates and the next three fields are a unit normal vector for the vertex.The next three fields are the function values (default=0.0) for the vertex. They are there for assigning properties to vertices, for example shape or electrostatic potential (by some external program). The next three fields are integers: component number, atom number and color (hue). Next come the edge records. Each one has C format: "%6d %6d %3d %5d %3d". The first two numbers specify the vertex numbers, which start at one. The vertex numbers correspond to the order of the vertex records earlier in this file. The next three numbers are the component number, the atom number and the color (hue) number. For edges on the boundary between different atoms or different colors, one of the two possibilities will be chosen randomly. Finally come the triangles. Each line has three edge numbers (the edge numbers start at one and are implicitly given by the order of the edge records), followed by three vertex numbers, followed by a surface-component number, an atom number and a color number. The edge numbers are signed; that is, a negative number means traverse the edge in the reverse direction. The edges and vertices are in counter-clockwise order when viewed from outside the molecule. The C format of a triangle line is: "%7d %7d %7d %6d %6d %6d %3d %5d %3d". The atom and color numbers are determined by which atom the center of the triangle is closest to. The triangles are sorted by atom number.

G. Density

AVS Field Format

The msroll program writes density in AVS field format, which consists of an ASCII header, followed by two form-feed characters, and then the binary data. This format can be found at: http://www.aber.ac.uk/~ccuwww/docs/g/g5.html.

H. PQMS Disk Format

Header Record

The file begins with a header record of 512 bytes. Most of this record is reserved for future versions of this file. The version number gives the digit(s) to the left of the decimal point, and the subversion number gives the digit(s) to the right of the decimal point. For example, a version number of 2 and a subversion number of 9 gives 2.9. The version and subversion numbers of all programs are equal for each release of the Molecular Surface Package, and this number is the number in the PQMS file. All PQMS files begin with the four letters: PQMS. These files are sometimes called pqms-files to distinguish them from the dot surface files produced by the MS program. The format of this header is:

Table 7: Header Record

Bytes	*Type*	*Contents*
1-8	char	"PQMS" & 4 trailing null bytes
9-12	long	version number
13-16	long	subversion number
17-24	double	probe radius
25-28	long	number of varieties
29-32	long	number of vertices
33-36	long	number of circles
37-40	long	number of arcs

Table 7: Header Record

Bytes	*Type*	*Contents*
41-44	long	number of faces
45-48	long	number of cycles
49-52	long	number of edges
53-56	long	number of components
57-60	long	number of atoms
61-64	long	reserved
65-128	char	name of molecule
129-512	char	reserved

Record Types

After the header record are the records that describe each geometrical element of the surface. These records are divided into thirteen (13) types. The mathematical term "variety" means a surface such as spheres, tori and cylinders. The msroll program uses only spheres and tori to define the piecewise-polynomial molecular surface. The msdraw program also uses the cylinder. The record types are:

Table 8: Record Types

Type	*Name*
1	atom variety
2	torus variety
3	probe variety
4	vertex
5	circle

Table 8: Record Types

Type	*Name*
6	convex arc
7	concave arc
8	convex face
9	saddle face
10	concave face
11	cycle
12	edge
13	component

Surface Elements

The type of record is given by the first two bytes of the record. Sometimes records of different types have the same format. In particular, the atom variety, torus variety and probe variety all have the same format. The convex arc and concave arc have the same format. Finally, the convex face, saddle face and concave face have the same format. The counters in the header record are used by msdraw to know how many of each format of record to read. The formats are given below:

Table 9: Surface Elements

bytes	*type*	*value*
Variety		
1-4	long	record type (1, 2 or 3)
5-16	3 longs	atoms numbers
8-28	3 double	center coordinates

Table 9: Surface Elements

bytes	*type*	*value*
29-32	double	radius
33-44	3 doubles	unit vector along axis
Vertex		
1-4	long	record type (4)
5-16	3 doubles	center coordinates
Circle		
1-4	long	record type (5)
5-8	long	subtype
9-20	3 doubles	center coordinates
21-24	double	radius
25-36	3 doubles	unit vector along axis
Arc		
1-4	long	record type (6 or 7)
5-8	long	subtype
9-12	long	circle number
13-20	2 longs	vertex numbers
21-24	long	error flag
Face		
1-4	long	record type (8, 9 or 10)
5-8	long	variety number
9-12	long	number of first cycle
13-16	long	component number
17-20	long	> 0: hue; < 0: error flag
Cycle		
1-4	long	record type (11)
5-8	long	number of next cycle of face

Table 9: Surface Elements

bytes	*type*	*value*
9-12	long	number of first edge
13-16	long	number of edges
Edge		
1-4	long	record type (12)
5-8	long	error flag
9-12	long	arc number (signed)
Component		
1-4	long	record type (13)
5-8	long	subtype
9-12	double	volume
13-16	double	molecular area
17-28	3 doubles	accessible area
29-40	3 doubles	center of component

I. Image File Formats

Indexed and RGB Color

The color table is written for SUN Raster format only. All the other formats use true color. For SUN Raster format, the pixels are indices into the color table. For all other formats, they are red, green and blue values. For the three AVS formats, the pixels are written so that all the information for a single pixel is grouped together. For AVS Image and AVS Field formats, a 32-bit pixel is written. For AVS Density format, a 16-bit float is written, so it is only good for gray scale. For SGI RGB format, the red, green and blue components of each pixel are split apart. That is, the file has first all the red components, then all the green components, and finally all the blue components.

Sun Raster

Sunraster 8-bits per pixel format. For the format of this file, consult the "rasterfile.h" #include file of SUN Microsystems. After the header there is a color lookup table. The color lookup table has 256 entries. The number of shades per color depends upon the number of colors actually used in the image.

SGI Image

The new SGI image format has replaced the old format. Use imgview to view it.

J. Vector File Formats

SGI Inventor

This will produce an ascii file in SGI's Inventor format, which can be viewed using ivview. The only tricky aspect is how to handle the coloring. The Open Inventor software provides for two kinds of coloring: (a) vertex-based and (b) triangle based. The programs

uses the hue numbers at the end of the triangle records of the polyhedron files and the hue numbers in the atom hues file (-h), unless there is an -f flag. The "f" stands for function. The u, v, and w arguments select which one of the three columns of vertex values is used. These real numbers are subdivided into eleven bins, each with a color from yellow through green to blue. The range being subdivided is from 0.0 to 1.0 by default, but may be changed by -l and -u flags. The vertex-based coloring of Open Inventor is used for this situation. An awkward construct, "-f h", is used to specify vertex coloring based upon the vertex hue integer.

Plotting: HPGL and PostScript

First the header is written. Then the line segments are written. Interspersed with the line segments are commands to change the line width or the pen color.