The VOLUMES program

or 'How not to write a program'

by Robert Esnouf, Oxford & Leuven

(This version 17th July 1997)

1) Introduction

The program VOLUMES was originally a tiny program put together one evening to work out the volume of a small molecule by creating a simple, orthogonal three−dimensional grid covering the molecule. Grid points were set to '1' if they were within the van der Waals radius of an atom centre and '0' otherwise then adding up the number of '1's gives the molecular volume. The program was extended to look for pockets in protein structures and calculate their volume in a similar way. The internal layout of the program was now set and further extensions have been limited by these humble beginnings. Until recently all grids were integer arrays even when used to store real numbers such as distances! For most current uses the program is obscure and unnecessarily complex − these notes are merely for those masochists among you who like to suffer for their art.

Despite the drawbacks there are certain functions of VOLUMES where the results are better than those obtainable from better−written rivals and when it comes to producing smooth surfaces for BobScript there is no other way of doing it. So, for those of you who are already wearing concrete reinforced boxer shorts and are not worried by a program that is the FORTRAN equivalent of a mind−expanding drug, here goes...

2) Commands

All input to VOLUMES uses my parser so keywords are case−insensitive, abbreviations can be used and spaces and line−breaks can be mixed freely. In this version of the program there are 27 main commands, listed here in alphabetical order. As with great Shakespearean actors, merely learning the words isn't enough − it's just as important to get them in the right order! So, some examples of VOLUMES input files are given in the next section.

`ACCESSVOL` *[gridname.c]* **`VALUE`** *centre.i newvalue.i probe−radius.r*

Sets grid points in the current (or named) grid which are accessible to a probe (of radius defined by *probe−radius.r*). The function assumes that the grid has already been set to contain the grid points accessible to the centre of the probe, these points having the value *centre.i*. The value put at each grid point is determined by the old value, the value specified by *newvalue.i* and the current `CALCMODE`. Grid points having the value *centre.i* are not affected. This function is intended for use in concert with the `PROBEVOL` function and it is normal (although not necessary) for the same *probe−radius.r* to be used in each case. A typical scheme for defining a pocket volume is to define the volume accessible to the centre of a probe (`PROBEVOL`), to select the connected volume of interest (`SELECT`) and finally to use `ACCESSVOL` to define the full pocket.

CALCMODE < SET | AND | OR | XOR | ADD | SUB >

`CALCMODE` affects how values in an integer grid are modified by most other commands. Each grid point can be set to a value between 0 and 1023 (for historical reasons). Most commands work in two stages: first they find out which points to alter and flag them, then they alter the grid points according to the value currently at each grid point and the new value specified in each command. The operation used to combine these values is controlled by the `CALCMODE` command. '`SET`' means that the grid point is set to the new value, '`AND`' means that the two values are logically ANDed together, *etc. etc.* Proper use of this command can be crucial for getting the right results.

CHANGE *[gridname.c]* **FROM** *oldvalue.i* **TO** *newvalue.i*

Changes values of grid points in the current (or named) grid which have the value specified by *oldvalue.i*. The value put at each grid point is determined by the old value, the value specified by *newvalue.i* and the current `CALCMODE`. This command is frequently used to clear away unwanted bits of volumes after they have been identified by a unique value.

CLEARGRID *[gridname.c]* **VALUE** *clear.i*

Clears the current (or named) integer grid to the specified value. Except when using it is normally safe to omit the grid name.

COMBINE *selection.c*

`COMBINE` is used after using the `VOLUME` function. It simply sums up the number of grid points having the selected values, printing out the total and the volume to which this corresponds. The values are selected by the string *selection.c* and some examples for this string are: '`1`', '`1,2,3`', '`0,4`', '`3-7`' and '`4-7,12-15`'.

COPYGRID *gridfrom.c* **TO** *gridto.c* **OK**

Copies both the real and integer grids from *gridfrom.c* to *gridto.c*. For the copy between the integer grids the current `CALCMODE` is obeyed. Thus, to copy a grid destructively you must use `CALCMODE SET`. The real grid is simply copied from the first to the second grid.

DEFINEGRID < *grid−input* > **END**

The program always works on grids, which are three−dimensional orthogonal arrays of real and integer values. Within the program all grids have the same dimensions, size and grid spacing. These parameters are defined by `DEFINEGRID`. When this command is used all existing grids are deleted, but no new ones are made until the `MAKEGRID` command is used. Grid limits are altered if necessary so that the origin would coincide with a grid point. The <*grid−input*> commands are:

BOX *lx.r ly.r lz.r ux.r uy.r uz.r*	Defines a box for the grid to cover by defining six real numbers corresponding to the lower−(*x*,*y*,*z*) coordinate and the upper−(*x*,*y*,*z*) coordinate. Alternatively a box may be defined in terms of atom positions in specified residues (`RESIDUES`). The extra padding provided by the `EXPAND` option will be added to this grid size.
EXPAND *extra.r*	Defines the amount of extra padding space to be added around the defined grid (default value is 0). It is normally set to be at least the grid spacing plus the largest atom radius. If this value is not big enough then the envelopes not fit within the grid.
RESIDUES *residue−sel.c*	Defines a box for the grid to cover by defining a residue range. The PDB file must already have been read in (`READPDB`) and

then a grid size is defined which exactly covers the atoms in the residue range. Some example residue range strings are: '`-`' (all residues), '`1-998`' and '`101-104,204,229`'. A box can also be defined using the `BOX` option. The extra padding provided by the `EXPAND` option will be added to this grid size.

SPACING *spacing.r* — Defines the fineness of the grid. The grid spacing is applied in all three dimensions. For quick functions a small grid spacing is usually better, but for complex surfaces large spacings may be required (say 4Å). In this latter case one has to be careful to avoid letting atoms fall between grid points.

DELETEGRID *[gridname.c]* **OK**

Deletes the definition of the current grid (or the named grid). In the current program it is difficult to see a situation where one would need to do this.

DISTANCE *[gridname.c]* **VALUE** *centre.i distance−cutoff.r*

This function was included in the VOLUMES program to make the visual representation of envelopes smoother (also '`SMOOTH`', '`DISTSMOOTH`', '`PARTMOL`' and their '`R`' equivalents). The integer functions are approximate, storing distances in 1/10th Å units. `DISTANCE` sets grid points in the current (or named) grid whose value is not *centre.i* to be the distance from the nearest grid point having the value *centre.i* (ignoring the current `CALCMODE`). For speed, the search for the nearest grid point having the value *centre.i* is terminated if the distance is greater than *distance−cutoff.r* and this distance cutoff value is storeed at that grid point.

DISTSMOOTH *smooth−fraction.r cycles.i*

Smooths an envelope in the current grid (due to an oversight one can't specify a grid name!) whose boundaries are defined in terms of distances from centre grid points (presumably by the function `DISTANCE`). To get useful results, the value for grid points at the heart of the envelope must be 0 and other values indicate distances from the nearest zero point. For each grid point the average of the values in its 26 neighbouring grid points is calculated. This value and the value at the grid point are then averaged using the equation

$$smooth{-}fraction.r \times neighbour{-}average + (1 - smooth{-}fraction.r) \times grid{-}point{-}value$$

and the result is stored at that grid point. The number of cycles of smoothing is specified by *cycles.i*. Typically, ten cycles with a *smooth−fraction.r* of 0.2 gives respectable results.

FREEVOL *[gridname.c]* **VALUE** *newvalue.i residue−sel.c*

`FREEVOL` sets grid points in the current (or named) grid which are not within the volume of the molecule (or part of a molecule) specified by the *residue−sel.c* string. See the `DEFINEGRID` for a definition of the residue selection string. The value put at each grid point is determined by the old value, the value specified by *newvalue.i* and the current `CALCMODE`. Radii for individual atoms (scaled by the parameter defined by the `INFLATE` command) are defined when the PDB file is read in. (See also `MOLVOL` and `PROBEVOL`).

HYDROGENS < **OFF** | **ON** >

This function is used at the very start of a run of VOLUMES. It flags whether hydrogen atoms are used or not by the program. Setting '`HYDROGENS ON`' means that hydrogen atoms will be treated as separate atoms in their own right, given appropriate radii and used in functions such as `MOLVOL`. Setting '`HYDROGENS OFF`' means that hydrogen atoms will be ignored by the `READPDB` function and inflated atom radii are assigned to non−hydrogen atoms to compensate for their absence. Very similar answers are obtained either way.

INFLATE *factor.r*

This function should be used near the start of a run of VOLUMES, certainly before using READPDB. It defines the factor (default value 1) by which atom radii are artificially inflated. It is normally used with very coarse grids in order to stop atoms from 'falling in the gaps' between grid points. It can also be used for creating smooth surfaces to represent proteins.

MAKEGRID *[* **VALUE** *initint.i] [* **RVALUE** *initreal.r] gridname.c*

Having defined grid parameters using DEFINEGRID, this command will make integer and real grids and associate them with the specified grid name. All points in the integer grid will be set to 0 or the value specified by the VALUE option, points in the real grid will be set to 0.0 or the value specified by the RVALUE option. Some grid names are not allowed to avoid confusion with the syntax of other commands, but I usually call them 'GRID'.

MOLVOL *[gridname.c]* **VALUE** *newvalue.i residue−sel.c*

Sets grid points in the current (or named) grid which are within the volume of the molecule (or part of a molecule) specified by the *residue−sel.c* string. See DEFINEGRID for a definition of the residue selection string. The value put at each grid point is determined by the old value, the value specified by *newvalue.i* and the current CALCMODE. Radii for individual atoms (scaled by the parameter defined by the INFLATE command) are defined when the PDB file is read in. (See also FREEVOL and PROBEVOL).

PARTMOL *[gridname.c]* **VALUE** *near.i not−near.i threshhold.i residue−sel.c*

This function works on an integer grid to define parts of envelopes (possibly smoothed 'distance' envelopes) which are due to the part of a molecule specified by the *residue−sel.c* string rather than the rest of the molecule. See DEFINEGRID for a definition of the residue selection string. The function uses a dummy grid to work out which grid points are closer to atoms in the specified residue selection than they are to other atoms in the molecule. (Due to the dodgy algorithm, this calculation can take several minutes.) Having worked this out the function then examines the main grid:

i) Grid points near to the selected part of the molecule are altered such that its new value is determined by the old value, the value specified by *near.i* and the current CALCMODE.

ii) Grid points which are not near to the selected part of the molecule and whose value is less than *threshhold.i* are altered such that its new value is determined by the old value, the value specified by *not−near.i* and the current CALCMODE.

iii) Grid points which are not near to the selected part of the molecule but whose value is greater than (or equal to) *threshhold.i* are set to have the value (*threshhold.i*–1), ignoring the current CALCMODE.

Why so complex? The idea of the function is to make it possible to colour part of an envelope (say, corresponding to a single domain) differently from other parts of the molecular surface. This requires a separate envelope for that domain which has the same shape as the envelope for the whole molecule in places where the partial envelope defines the molecular surface. Smoothing functions work differently for whole molecules and partial molecules and thus give different surfaces. With PARTMOL a smoothed envelope is created for the whole molecule and then the molecular envelope due to the selected part is selected as the final step. The *threshhold.i* parameter relates to the contouring level used to draw the envelope and ensures that there are no clefts between domains in the final picture. For good results, the contouring level should be *threshhold.i*–0.5. Setting *threshhold.i* to −1 disables the effect. With smoothed distance envelopes I recommend using 'CALCMODE AND' with *near.i* as 1023 and *not−near.i* as 0. (See the examples section).

PROBEVOL *[gridname.c]* **VALUE** *newvalue.i residue−sel.c probe−radius.r*

Sets grid points in the current (or named) grid which are accessible to the centre of a probe of radius defined by *probe−radius.r*. Grid points are considered accessible if they are not within the volume of the molecule (or partial molecule) specified by the *residue−sel.c* string or within the probe radius of that volume. See DEFINEGRID for a definition of the residue selection string. The value put at each grid point is determined by the old value, the value specified by *newvalue.i* and the current CALCMODE. Radii for individual atoms (scaled by the parameter defined by the INFLATE command) are defined when the PDB file is read in. (See also FREEVOL and MOLVOL). This command is used in conjunction with ACCESSVOL to define internal pockets within a protein.

RDISTANCE *[gridname.c]* **FROM** *residue−sel.c*

This function works on real grids and it replaces the DISTANCE function for integer grids. It sets grid points in the current (or named) grid to be the distance from the nearest atom in the molecule (or partial molecule) specified by the *residue−sel.c* string. See DEFINEGRID for a definition of the residue selection string. The code for this function is not optimised, so it can take several minutes to calculate.

READPDB *filename.c*

Deletes any current atom data in memory and then reads in the named PDB−format file. This command also assigns radii to each atom so if the INFLATE command is to be used it has to be used before reading in the PDB file.

RPARTMOL *[gridname.c]* **VALUE** *near.i threshhold.r residue−sel.c*

This function works on real grids and it replaces the PARTMOL function for integer grids. It affects both integer and real grid points in the current (or named) grid depending on whether or not the nearest atom to the grid point is in the molecule (or partial molecule) specified by the *residue−sel.c* string. See DEFINEGRID for a definition of the residue selection string. Like PARTMOL, it uses a dummy grid to calculate an envelope of grid points which are near to the residue selection (using the same, rather time−consuming, algorithm). It then sets integer grid points within this envelope to a value determined by the old value, the value specified by *near.i* and the current CALCMODE. For the real grid the only effect is on grid points outside the envelope and which have a value less than *threshhold.r*. These grid points are set to have the value *threshhold.r*. The value of *threshhold.r* should be slightly greater than the contouring level used for rendering the surface (see the examples section).

RSELECT *[gridname.c]* **VALUE** *threshhold.r newreal.r newint.i* < **XYZ** *x.r y.r z.r* | **ATOM** *residue.c atom.c* >

This function works on real grids and it replaces the SELECT function for integer grids. It selects grid points in the current (or named) grid which are connected to the specified starting position by real grid points with a value greater than (or equal to) the *threshhold.r* value. See SELECT for how to define the starting position. RSELECT affects the integer grid, setting grid points which are connected to the starting position to a value determined by the old value, the value specified by *newint.i* and the current CALCMODE. For the real grid the only effect is on grid points not connected to the starting position and which have a value greater than *newreal.r*. These grid points are set to have the value *newreal.r*. Typically *newreal.r* is set to be slightly less than *threshhold.r* (see the examples section).

RSMOOTH *smooth−fraction.r cycles.i*

This function works on real grids and it replaces the DISTSMOOTH function for integer grids. It smooths an envelope of real values in the current grid. (Due to an oversight one can't specify a grid name!) Usually these real values will represent distances from an atom selection (presumably as the result of using RDISTANCE). For each grid point the average of its value and the values in its 26 neighbouring grid points is calculated in the same manner as with DISTSMOOTH. The number of cycles of smoothing is specified by *cycles.i*. Typically, ten cycles with a *smooth–fraction.r* of 0.2 gives respectable results.

SELECT *[gridname.c]* **VALUE** *newvalue.i* < **XYZ** *x.r y.r z.r* | **ATOM** *residue.c atom.c* >

Sets grid points in the current (or named) grid which are connected by grid points of the same value to a specified grid point. The starting grid point is specified either by an absolute coordinate (*x.r*, *y.r*, *z.r*) or by the position of the single atom specified by *residue.c* and *atom.c*. The program stores the value at the starting point as does the three–dimensional equivalent of a 'flood fill' in drawing programs to find points which are connected to this point by grid points of the same value. (Each neighbour to the starting point which has the same value is marked and then becomes the starting point for a new search *etc. etc.*) The value put at each connected grid point is determined by the old value, the value specified by *newvalue.i* and the current CALCMODE. This routine is useful for selecting single pockets within proteins (when used in conjunction with PROBEVOL and ACCESSVOL).

SMOOTH *[gridname.c]* **VALUE** *inside.i outside.i* < **SINGLE** *savein.i* | **DOUBLE** *savein.i saveout.i* > *cycles.i*

This function smooths an envelope in the current (or named) grid whose grid points have the value *inside.i* (points being considered outside the envelope if they have any other value). Grid points which become part of the envelope during smoothing are given the value *inside.i* and those which are trimmed from the envelope get the value *outside.i* (ignoring the current CALCMODE). SMOOTH adds up how many neighbours of each grid point are inside and how many are outside the envelope (total number of neighbours is 26) and from that decides whether the grid point should be reclassified as inside or outside. Two types of smoothing are possible: SINGLE smoothing only affects grid points within the envelope, requiring them to have at least *savein.i* neighbours also within the envelope to remain part of the the envelope; DOUBLE smoothing checks both 'inside' and 'outside' grid points, requiring 'inside' points to have at least *savein.i* 'inside' neighbours to remain inside the envelope and 'outside' grid points to have at least *saveout.i* 'outside' neighbours to remain outside the envelope. The second smoothing scheme better preserves the volume of the envelope, whereas the first scheme always shrinks an envelope. The final parameter, *cycles.i*, controls how many times the smoothing function should be performed. Typically, *savein.i* and *saveout.i* both have the value 13.

USEGRID *[gridname.c]* **OK**

Makes the named (or current) grid the new current grid. This command is rarely used since when you specify a grid name with other commands it becomes the current grid anyway.

VOLUME *[gridname.c]* **END**

Prints out the number of integer grid points in the current (or named) grid having each value. It does not affect the grid in any way, but gives useful check of intermediate results.

WRITEGRID *[gridname.c]* **AS** *file.c* *[* **REAL** *]* *title.c order.c cellx.r celly.r cellz.r*

Writes out the current (or named) grid as a CCP4 ASCII map file. The map name is given by *file.c* and the title, axis order and cell dimensions are used to construct a map header. These arguments are not important but they have to be supplied and will be adjusted if necessary. If the REAL option is specified then the real grid is written, otherwise the integer grid is written. Valid strings for *order.c* are 'XYZ', 'XZY', 'YXZ', 'YZX', 'ZXY' and 'ZYX'.

3) A few example files

This first example simply calculates the volume of a small molecule. It shows the original intention of the program:

```
! Read the protein model

  HYDROGENS ON
  READPDB apa-r90385.pdbh

! Make a grid covering a box surrounding the NNI

  DEFINEGRID SPACING 0.1 RESIDUE 999 EXPAND 2 END
  MAKEGRID GRID

! Set a 2 for all points accessible to the centre of a water probe

  CALCMODE SET
  MOLVOL VALUE 2 999

! Current grid:
!           0: nothing
!           2: inside the the NNI

! Show the volume of the inhibitor as the volume of the grid points set to 2

  VOLUME END
  STOP
```

The second example is an input file to show part of the smoothed molecular surface for RT centred on the NNI–binding pocket. The use of variables (words beginning with '$'s) is part of the parser and not the VOLUMES program, it helps to keep most of the input file nearly identical for different inhibitors:

```
! Variables for the parser to keep the rest NNI independent

  $NNI=MKC
  $RES=1-998
  $ATOM=C7
  $GX=140 $GY=110 $GZ=75

! Read the protein model

  HYDROGENS ON
  READPDB $NNI_e.pdbh

! Make big enough box to surround the protein

  DEFINEGRID SPACING 0.5 RESIDUE 999 EXPAND 14 END
  MAKEGRID GRID

! Current grid:
!           0: nothing

! Set a 2 for all points accessible to the centre of a water probe

  CALCMODE SET
  PROBEVOL VALUE 2 $RES$ 1.4

! Current grid:
!           0: nothing
!           2: accessible to centre of water probe

! Select just the volume connected to the NNI site
```

```
  CHANGE FROM 2 TO 4
  SELECT VALUE 2 ATOM 999 $ATOM$
  CHANGE FROM 4 TO 0

! Converted the probe centre envelope to distances from the probe centre to
! make contouring smoother

  DISTANCE VALUE 2 3.0
  WRITE GRID AS $NNI$.asc "RT molecular surface for $NNI$" YXZ $GX $GY $GZ
  STOP
```

The third example calculates and smooths a surface for a molecule using a coarse grid and a large inflation factor for atoms. The big advantage of a coarse grid is that surfaces can then be made up from fewer triangles when they are drawn using BobScript. The inflated radii also reduce the problem of internal pockets in the structure:

```
! Read in the protein

  HYDROGENS OFF
  INFLATE 4
  READPDB apo_e.pdb

! Make a grid covering the protein: fairly crude will do fine

  DEFINEGRID SPACING 2 RESIDUE - EXPAND 8 END
  MAKEGRID GRID

! Make the smoothed envelope

  CALCMODE SET
  MOLVOL VALUE 2 -

! Select points connected to the bulk solvent (the XYZ value is at the edge of
! the grid

  SELECT VALUE 1 XYZ -22 -67 -33
  CHANGE FROM 0 TO 2
  CHANGE FROM 1 TO 0
  DISTANCE VALUE 0 10.0
  DISTSMOOTH 0.3 10

  WRITE GRID AS apo.asc " " YXZ 100 100 100
  STOP
```

The fourth example calculates and smooths a surface for a molecule using a coarse grid and a large inflation factor for atoms in the same way as the previous example. Then it uses the PARTMOL function to select part of the molecule (RT) relating to one domain (p66 thumb). The threshhold value for the selection is 41, and in the BobScript input file the envelope is contoured at 40.5. If you understand this example then you are a true VOLUMES–expert and can have a medical certificate saying that you have a serious mental imbalance:

```
! Read in the protein

  HYDROGENS OFF
  INFLATE 4
  READPDB apo_e.pdb

! Make a grid covering the protein: fairly crude will do fine

  DEFINEGRID SPACING 2 RESIDUE - EXPAND 8 END
  MAKEGRID GRID

! Make the smoothed envelope

  CALCMODE SET
  MOLVOL VALUE 2 -
```

```
! Select points connected to the bulk solvent (the XYZ value is at the edge of
! the grid)

  SELECT VALUE 1 XYZ -22 -67 -33
  CHANGE FROM 0 TO 2
  CHANGE FROM 1 TO 0
  DISTANCE VALUE 0 10.0
  DISTSMOOTH 0.3 10

! Select just the part of the envelope that corresponds to the p66 thumb

  CALCMODE AND
  PARTMOL VALUE 1023 0 41 243-311

  WRITE GRID AS p66t.asc " " YXZ 100 100 100
  STOP
```

The final example is similar to the previous one except that it uses functions which manipulate real grids not the integer ones. Since it is dealing with distances all the time there is no need to use an inflation factor for atom radii. Notice that two grids are used, the one called COPY is simply used to store a copy of the envelope defining the whole smoothed protein surface to speed up the calculation of envelopes for a set of domains. The envelope for the whole protein is calculated using RDISTANCE, any internal cavities are removed using RSELECT and finally the surface is gently smoothed using RSMOOTH. For each domain in turn, the whole envelope is restored with COPYGRID and then RPARTMOL is used to select just the part of the envelope due to the required domain. Using the parameters in this input file I find that rendering the maps in BobScript with contour levels of 2.8 for the whole envelope and 2.81 for the domain envelope gives nice results:

```
! Input file to calculate a smoothed envelope for a protein and a partial
! for two different domains of that protein.

! Read in the protein

  HYDROGENS OFF
  READPDB a.pdb

! Make a grid covering the protein: fairly crude will do fine

  DEFINEGRID SPACING 2 RESIDUE - EXPAND 4 END
  MAKEGRID GRID
  MAKEGRID COPY
  USEGRID GRID OK

! Make the smoothed envelope for the whole protein (the XYZ value is at the
! edge of the grid)

  CALCMODE SET
  RDISTANCE FROM 1-996
  RSELECT VALUE 2.5 1.9 1 XYZ -20 -64 -22
  RSMOOTH 0.25 8
  WRITE GRID AS whole.asc REAL " " YXZ 100 100 100
  COPYGRID GRID TO COPY OK

! Make the fingers domain envelope

  COPYGRID COPY TO GRID OK
  RPARTMOL VALUE 2 4.0 1-88,121-146
  WRITE GRID AS fingers.asc REAL " " YXZ 100 100 100

! Make the thumb domain envelope

  COPYGRID COPY TO GRID OK
  RPARTMOL VALUE 2 4.0 243-311
  WRITE GRID AS thumb.asc REAL " " YXZ 100 100 100

  STOP
```

4) An apology

Finally, just before signing off this piece of documentation I would like to apologize once again for the state of this program. It has become very much a thing that bits get added to in a disorganized way. If I had time I would re–write the whole program in a more logical manner and try and get a paper out on it, since several of the functions it provides give better results than its competitors. I hope that by producing some documentation I haven't made it less likely that I give the code this new lease of life the ideas in it deserve. If you have any more ideas or requests then you can get in touch with me and I'll make the program even more confusing!

Cheers,

Robert, Leuven, 17/7/97
(`robert@s151h16.rega.kuleuven.ac.be` or `robert@biop.ox.ac.uk`)