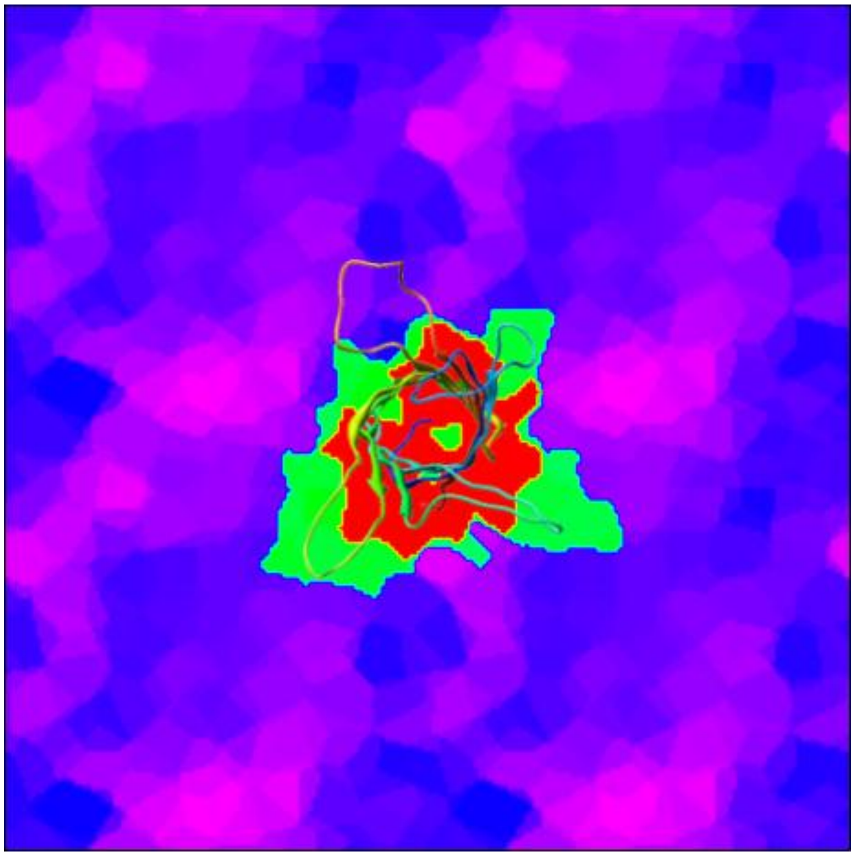
GridMAT-MD: A Grid-based Membrane Analysis Tool for use with Molecular Dynamics

William J. Allen, Justin A. Lemkul, and David R. Bevan

Department of Biochemistry, Virginia Tech



User's Guide

Version 2.0

Table of Contents

1. Preface

GridMAT-MD is an acronym for Grid-based Membrane Analysis Tool for use with Molecular Dynamics. It is a simple Perl script designed to measure two things - the apparent "thickness" of a lipid bilayer, and the area per lipid head group of each lipid that comprises a lipid bilayer. The output can then be visualized using a separate tool – we primarily use the Xmatrix viewer (http://www.matpack.de) or Gnuplot (http://www.gnuplot.info).

We believe GridMAT-MD to be superior to existing tools because (i) it is free to use and modify under the terms of the GNU public license, (ii) no installation is required, provided the Perl interpreter is installed, (iii) it calculates an explicit area for proteins, making no assumptions of "ideal proteins," and (iv) it is very fast to run and easy to operate.

This program can be modified and/or redistributed under the terms of the GNU Public License (see `gpl-3.0.txt`). Should you encounter difficulties, contact one of the developers, and we will do our best to help. Please realize that this is free software, and we do not imply any warranty or guarantee any technical support. If you have ideas, questions, comments, or if you have made useful modifications to the code, please send them to us! We welcome feedback and user contributions.

To learn more about the algorithms behind the program, please refer to our paper:

Allen, W. J., Lemkul, J. A., and Bevan, D. R. (2009) "GridMAT-MD: A Grid-based Membrane Analysis Tool for Use with Molecular Dynamics." *J. Comput. Chem.* **30** (12): 1952-1958.

Or visit our website:

http://www.bevanlab.biochem.vt.edu/GridMAT-MD/index.html

2. Download and Installation

If you are reading these instructions, odds are you have already successfully downloaded and unzipped the appropriate `.tar` file from our website. Within you will find the following:

`GridMAT-MD.pl`	The program
`param_example`	An example parameter file
`GridMAT-MD_ug_v2.0.pdf`	This user's guide
`gpl-3.0.txt`	The GNU Public License, version 3.0

One of the great things about GridMAT-MD is that there is no installation necessary. Most newer machines running Linux, Mac, or Unix should already have Perl installed, which is the only prerequisite to this program. Perl interpreters for Windows are freely available online.

Presently, GridMAT-MD looks for the Perl interpreter at `#!/usr/bin/perl`. If your machine has a different architecture (try the command "`which perl`"), then you will have to change the first line of the file `GridMAT-MD.pl` accordingly.

3. File Preparation

In order to run GridMAT-MD, you need three files. Two should have been downloaded by now (`GridMAT-MD.pl` and `param_example`), and the third is a PDB format `(.pdb)` or a GROMACS format (`.gro`) coordinate file. There are several notes to consider about the file format for compatibility with the program:

- The lipids in the coordinate file should be positioned so that the hydrophobic tails of one leaflet are in contact with the hydrophobic tails of the opposite leaflet (just as in a biological system). We have seen systems in which the hydrophobic tails of each leaflet face towards the outside of the box, allowing periodicity to make the system whole. In this case, measurements made by GridMAT-MD will correspond to distances between periodic images, not the true thickness of the bilayer.

- Coordinate files must be numbered consecutively from 1. This is easily attainable with the GROMACS utility `genconf –renumber`, for example.

- For PDB files, make sure your residue names are only 4-characters long. The ATOM record in PDB files only allows 4 characters; the space for a 5th character is reserved for the chain identifier. If you have longer residue names, simply do a search-and-replace to change them to 4 characters.

- If you are using a multi-frame PDB or GROMACS format file, make sure that each frame in the file contains the exact same number of lines. This includes all header information. This is generally the case when binary trajectories are converted to `.pdb` or `.gro` using the GROMACS utility `trjconv`.

4. Running the Program

The program is run from the command line by typing the following:

```
perl GridMAT-MD.pl param_example
```

Or, if you choose to `chmod +x GridMAT-MD.pl`, then you can type the following:

```
./GridMAT-MD.pl param_example
```

In each case, the program is executed by the Perl interpreter, and the name of the parameter file is the only input read from the command line. Make sure all files are in the same directory and that the read/write access is set appropriately. Output files will be written into the same directory from which the program is executed. Also, be aware that output files will overwrite any pre-existing output files with the identical name.

5. Analysis of Output Files

In any given run, there is the possibility for two types of output files: thickness data and area data. Thickness data will be written to three different files: `xxx.top_thickness.dat`, `xxx.bottom_thickness.dat`, and `xxx.average_thickness.dat`. These files contain the Z-values (or the apparent thickness in the *z*-dimension) for the bilayer from perspective of the top leaflet, the bottom leaflet, or the average of the two, respectively. Please see the publication for more details.

At this time, the user can choose to print the Z-values in three different formats: `column`, `matrix`, or `vector`. For instance, consider this snippet of an output file written in `column` format:

```
4.6585
4.453
4.3605
4.286
4.274
4.274
...
```

The interpretation of this data is as follows. The number of lines in the file will correspond to the product of the grid dimensions used for the calculation. For example, in a file that has the suffix "`20x20.average_pbc.dat`," there would be 400 lines. In the file, all the *x*-dimension values are printed for a corresponding *y*-row in the matrix. For example:

(x_1, y_1)
(x_2, y_1)
(x_3, y_1)
...
(x_{20}, y_1)
(x_1, y_2)
(x_2, y_2)
...
(x_{20}, y_{20})

The other output types, `vector` and `matrix`, follow a similar pattern which can be reasonably identified. These different data formats are useful for reading into an external viewing program. As previously mentioned, we have found success using the Xmatrix viewer and Gnuplot, although other programs are available. (See the Appendix of this document for instructions on how to use Xmatrix and Gnuplot.)

The other type of output file contains area data – more specifically, area per lipid headgroup. The program will print two separate files, one containing top leaflet areas (`top_areas.dat`) and one containing bottom leaflet areas (`bottom_areas.dat`). These are simple text files that list the average area per lipid head group on the top line, followed by the individual residue areas in the subsequent lines. The columns are divided into the lipid residue ID number (from the input `.gro` or `.pdb` file), the Z-value, and the area:

```
Ave APL = 66.0203793103448 sq. Ang

Resid  Z_value  Area (sq. Ang.)
49     3.594    60.40534605
50     3.148    82.80606075
51     3.068    32.26085835
52     4.187    50.83239105
53     3.799    91.3259907
54     3.208    31.0163742
…
```

In addition, if there is a protein in the coordinate file and the parameter `protein` is set to "`yes`" in the parameter file, the lateral area of the protein will be reported as well. Note that the accuracy of the area calculations will always increase as the grid resolution is increased (at the expense of time for calculation). See section 6.4 for more on output files.

In all cases, the resolution of the grid (e.g., 20x20) is included to the name of the output file name. This will be a useful reminder to the user of how to interpret the data within.

6. List of Parameters

The parameter file for this program is a simple hash of keys and arguments, separated by white space. Comment lines must begin with a '#' sign. For each of the parameter descriptions, the name of the parameter and example options [and in some cases, all possible options] are listed on the first line, followed by the usage of that parameter on subsequent lines.

6.1 Input file and input file parameters

```
coord_file              my_bilayer.gro
```
Usage: This is the filename of the `.pdb` or `.gro` format coordinate file. It is expected that there is some sort of lipid within this file. Also acceptable are protein atoms, solvent atoms, ions, and header information.

```
file_type               gro [gro | pdb]
```
Usage: The `file_type` must be specified correctly in order for the program to parse the coordinate file correctly. The only supported file types are `.gro` and `.pdb`.

```
num_frames              1
```
Usage: Single-frame or multiple-frame coordinate files (`.gro` or `.pdb` format) can be processed. The exact number of frames in the coordinate file must be specified in the parameter file by `num_frames`. Each frame in the file must have the exact same number of lines, including all header information, as is typically the case when converting binary trajectories to this format using the GROMACS utility `trjconv`.

```
num_lipid_types         1
```
Usage: The number of unique lipid types (with unique residue names) present in the coordinate file specified as an integer. This number can be arbitrarily large, and it should never be less than 1.

```
resname1                POPC
```
Usage: The residue name of the first lipid type in the coordinate file.

```
atomname1               C12,C13,C32
```
Usage: The name(s) of one or multiple atoms that belong to the lipid with the same residue name (i.e., the lipid in `resname1` corresponds to the atoms in `atomname1`, `resname2` corresponds to `atomname2`, etc.). The atoms specified here are used as a reference point to represent each lipid. Thus, specifying glycerol backbone atoms will result in the backbone-backbone distance, specifying the phosphorous atom will result in the phosphorous-phosphorous distance, etc. If more than one reference atom is specified, the center of mass will be used. Note that multiple atom names must be separated by commas and no spaces.

```
resname2                DPPC
```
Usage: In a mixed bilayer system, multiple lipid residue names will need to be specified. The `resnameN` and `atomnameN` parameters are always used in pairs, and can go up to an arbitrarily high number *N*. The highest value of *N* should equal the integer used for the parameter `num_lipid_types`.

```
atomname2               P8
```
Usage: If `resname2` is defined, `atomname2` must also be defined (this is true for all `resnameN`/`atomnameN` pairs). See the description for `atomname1`.

```
solvent                 SOL
```
Usage: This is the residue name of the solvent molecules in the coordinate file. If the bilayer is a dehydrated system, this can just be left alone or undefined. This is used in some cases to determine the size of the box (see `box_size`) and it is used when determining which atoms are protein atoms.

```
ions                    NA+,CL-
```
Usage: List the names of all ions found in the coordinate file. This is really only necessary to help the program distinguish which atoms are protein atoms and which are not protein atoms. If only one type of ion is present, no commas are needed. Again, notice that the ions are separated by commas and no spaces.

6.2 Grid size and shape

```
box_size                vectors [vectors | solvent]
```
Usage: The program can define the size of the box in three ways: (i) by taking the box vectors listed in the coordinate file (the last of a `.gro` file or in the "`CRYST1`" record of a `.pdb` file), (ii) by taking user-specified vectors from the parameter file, or (iii) it can be

estimated by the min and max coordinates of solvent atoms in the *x*- and *y*-directions. To use options (i) or (ii), set `box_size` to "`vectors`", and to use option (iii), set `box_size` to "`solvent`".

```
override_vectors        5.5,10.0,6.9
```

Usage: When `box_size` is set to "`vectors`", the program will first look for and take vector information from the coordinate file. It will either pull the last line of a `.gro` file or pull the `CRYST1` record information from a `.pdb` file. In the case of multi-frame coordinate files, it will pull new vector information from each frame. The option `override_vectors`, if used, allows the user to ignore vector information in the coordinate file and specify values of their choosing. This is not recommended for multi-frame coordinate files as the box size is expected to change slightly from frame to frame if the trajectory was conducted using an *NPT* ensemble. Also note that the three components of the vector (*x*, *y*, and *z*) are separated by commas and no spaces. The values must be supplied in units of nm, not Å.

```
grid                    100
```

Usage: The grid number is the number of grid points across each axis (*x* and *y*). For example, the number '100' would generate a grid that is 100 x 100 points, or 10,000 points total. A smaller grid number (20 to 25) will generate data that looks like a smooth gradient. This is useful for observing changes in the thickness of the lipid bilayer. A larger grid order (100 to 200) will generate data that looks more like a tessellation. This is useful for determining individual areas per lipid head group. Please see the publication and website for more examples.

```
conserve_ratio          yes [yes | no]
```

Usage: In some cases, it may be desirable to generate a rectangular grid instead of a square grid. If this option is set to "`yes`", the grid number will be the number of grid points along the longer axis, and the number of grid points along the shorter axis will scale to the nearest whole number that approximates the *x*-to-*y* ratio. Note that if a rectangular grid is generated, and the box dimensions change substantially (as may be the case for a long simulation using anisotropic pressure coupling), averaging over multiple frames may be difficult.

6.3 Accounting for protein atoms

```
protein                 yes [yes | no]
```

Usage: If "`yes`", the program will look for any protein atoms that fall within the lipid head groups. If any protein atoms meet the qualification (as defined by the algorithm described in the paper), they will directly compete for grid points with the lipid atoms. The closest atom to each grid point wins that point, and donates its Z-value to the grid. In the case of the lipid atoms, the Z-value is the thickness of the lipid bilayer at that point, and in the case of protein atoms, its Z-value (or `P_value`, see below) is user-defined. Note that if `protein` is set to "`yes`" when conducting a thickness calculation, the `P_value` assigned to the protein may artificially influence thickness results, especially when averaging over multiple frames, and in the case of a mobile (small) protein/peptide. Thus, when conducting thickness calculations, we recommend that `protein` is set to "`no`," even if the system truly contains a protein.

```
precision                1.3
```
Usage: The precision is simply a search radius for the protein atoms. A given protein atom will search for all lipid reference points within a 1.3-nm radius (unless otherwise specified). From that list of lipid reference points, if there is at least one that is higher and one that is lower on the z-axis, that protein atom is defined as falling within the "realm" of the lipid bilayer headgroups. Large values (1.5 – 3.0 nm) here will result in selecting more protein atoms, and small values (0.5 -1.5 nm) here will result in selecting fewer protein atoms. In our tests, we noted no substantial increase in accuracy beyond a value of 1.3, but do feel free to test this conclusion in your own systems.

```
P_value                  5.0
```
Usage: This is the user defined Z-value for protein atoms. The best way to determine what number to use here is to try a couple different numbers and see how they contrast against the Z-values that were determined in the program. If the range of your bilayer thickness is on the order of 3.0 to 5.0 nanometers, try `P_values` of 2.0 or 6.0 nanometers.

6.4 Define desired output files

```
output_prefix            output
```
Usage: This is a user-specified prefix that will be appended to the beginning of all output filenames.

```
output_format            column [column | vector | matrix]
```
Usage: The output thickness data can be written in three different formats: `column`, `vector`, or `matrix` (see Section 5 for details). This option should be chosen based on compatibility with the preferred visualization program.

```
thickness                yes [yes | no]
```
Usage: If "`yes`", the bilayer thickness with respect to the top leaflet, the bottom leaflet, and the average will be written to file.

```
area                     yes [yes | no]
```
Usage: If "`yes`", a list of data including the residue number (from the coordinate file), the Z-value (the relative "thickness" in nanometers of the bilayer at that point), and the area of that particular residue (in square Angstroms) will be written to file. If `protein` is set to "`yes`", and if any protein atoms fall within the reference atoms of the top leaflet, the area of the protein atoms will also be included. Finally, the average area per lipid head group is included.

7. Example Analyses

7.1 Bilayer thickness calculation

Contents of `param_file`:

```
coord_file              DPPC_bilayer.gro
file_type               gro
num_frames              1
num_lipid_types         1
resname1                DPPC
atomname1               P8
solvent                 SOL
ions                    NA+,CL-

box_size                solvent
grid                    20
conserve_ratio          no

protein                 no

output_prefix           output
output_format           column
thickness               yes
area                    no
```

This parameter file is suited for computing the thickness of a bilayer. It reads from the coordinate file named `DPPC_bilayer.gro`, which has only one frame and one type of lipid. It searches for all `P8` atoms from molecules with the `DPPC` residue name. These points are stored as reference points. The solvent residue name is `SOL`, and the `ions` are deleted.

The size of the system depends on the upper and lower *x*- and *y*- boundaries of the solvent box, so that is calculated and the solvent molecules can be deleted from memory. Then a grid is generated of 20 x 20 points that covers the size of the box.

`Protein` is set to “`no`”. This does not necessarily mean there is no protein in the system. This means that, in order to obtain a nice smooth picture of the thickness of the bilayer, there is no need to consider the protein atoms. It may be a useful exercise to run the same protein-lipid system twice with `protein` set to “`yes`”, then “`no`”, comparing the results.

The `output_prefix` on all filenames will be “`output`”. Thickness data will be written to three separate output files, but with a grid resolution this low, it is hard to gather much data about the area per lipid headgroup. The format of the output thickness data will be a single `column`, as described in Section 5.

7.2 Area per lipid headgroup calculation

Contents of `param_file`:

```
coord_file              mixed_bilayer.gro
file_type               gro
num_frames              1
num_lipid_types         2
resname1                DPPC
atomname1               P8
resname2                POPC
atomname2               P1,C1
solvent                 SOL
ions                    NA+,CL-

box_size                vectors
grid                    200
conserve_ratio          yes

protein                 yes
precision               1.3
P_value                 5.0

output_prefix           output
output_format           column
thickness               yes
area                    yes
```

This parameter file is designed to generate the data for area per lipid headgroup and a tessellation-like image. This time we are using a mixed bilayer with two lipid types, named `mixed_bilayer.gro`. There are both `DPPC` and `POPC` lipids present. From the `DPPC`, we will be using the `P8` atoms as reference points, and from the `POPC`, we will be using the center of mass of the `P1` and `C1` atoms as reference points. After the reference points are determined, they are not differentiated in any significant way. The residue name of the solvent is `SOL`, and the `NA+` and `CL-` ions are deleted.

This time we are using the box `vectors` from the coordinate file to determine box size, so the waters are discarded. The `grid` resolution has become significantly larger. The value of `conserve_ratio` is “`yes`”, so the larger side of the box will have `200` grid points, and the shorter side of the box will scale the number of grid points so they are about the same distance apart as the long side.

`Protein` has been set to “`yes`.” Now all of the protein atoms are considered one by one, and the program checks whether each falls within the lipid headgroups. If so, the protein atoms are added to the array of reference points according to which leaflet it falls in (top or bottom). They are given the artificial Z-value, `5.0`.

Now `top_pbc` and `bottom_pbc` will show some important information. When visualized, these files will show a tessellation of oddly-shaped polygons. Each polygon represents a specific lipid headgroup. The average area and area of each specific headgroup is printed to the other output files – `top_areas.dat` and `bottom_areas.dat`. If any protein atoms were found, the lateral area that the protein takes up will be listed in the appropriate area file (depending on whether the protein is in the top, bottom, or both leaflets).

8. Acknowledgment

The authors extend their appreciation to the community of GridMAT-MD users who have helped us to improve this program through their bug reports, inquiries, and useful suggestions, especially including Mohtadin Hashemi.

Appendix A: Interpreting GridMAT-MD Output with Xmatrix and Gnuplot

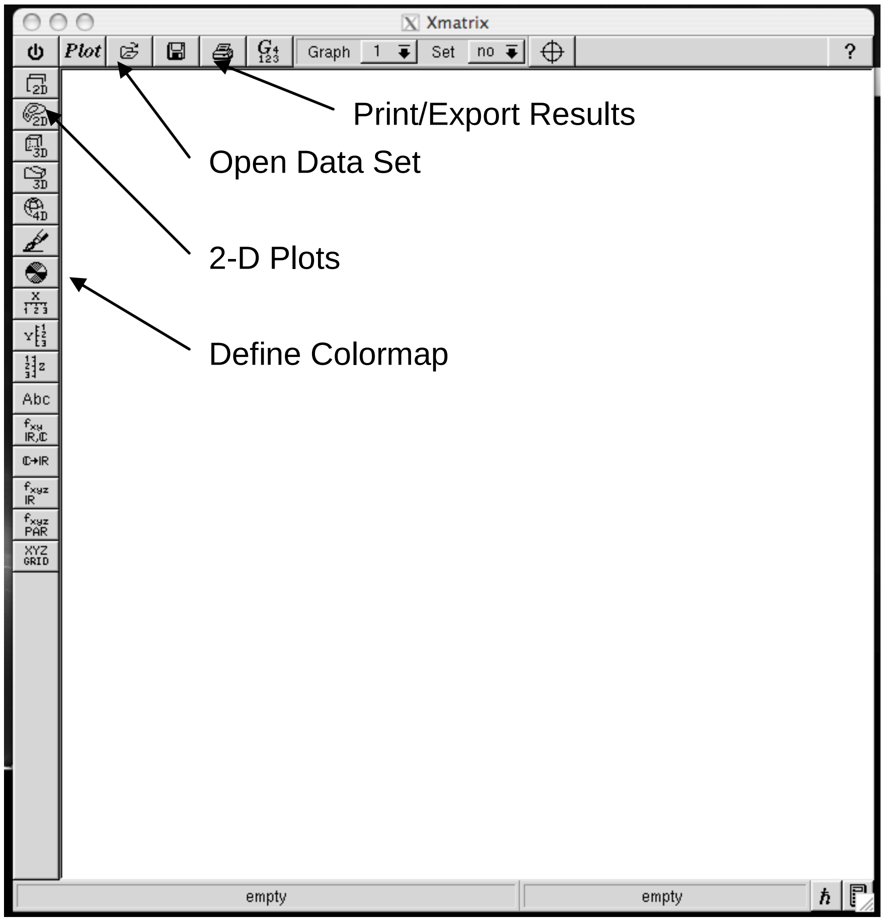
As of version 2.0 of GridMAT-MD, the thickness data can be output in single-column, vector, or matrix format. The following two sections will describe how to visualize single-column data using the Xmatrix viewer, distributed as part of the Matpack C++ Numerics and Graphics Library (http://www.matpack.de), and the matrix data using Gnuplot (http://www.gnuplot.info).

Section 1: Xmatrix

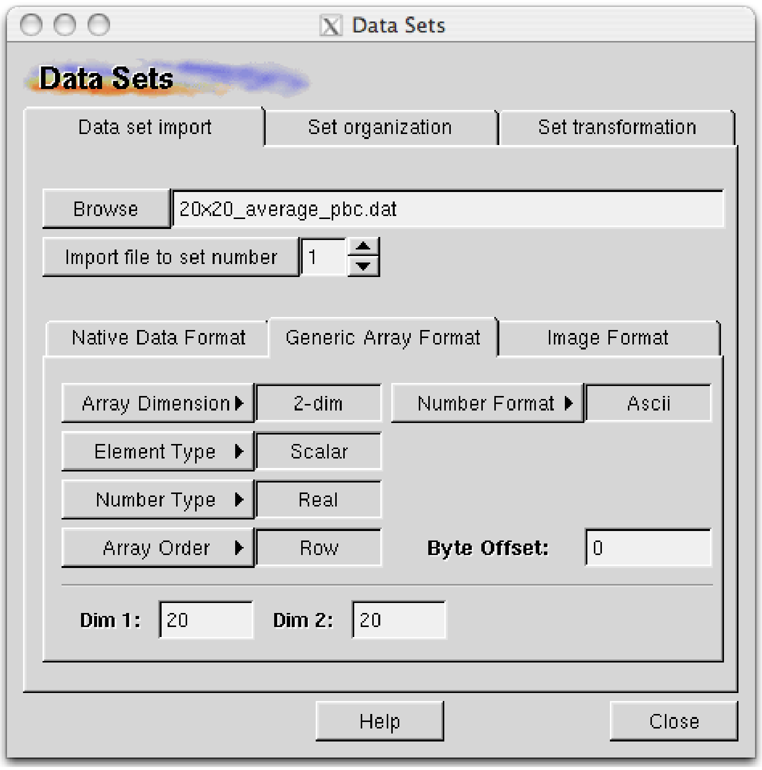
The number of lines in single-column format data will correspond to the product of the grid dimensions specified when running the program. Thus, for a 20 x 20 matrix, there will be 400 lines, corresponding to the 20 *x*-grid points for each of the 20 *y*-grid points. To read the GridMAT-MD output into Xmatrix, launch the program from the command line:

```
$ xmatrix &
```

The main window will launch. The most pertinent options for our purposes here are labeled in the following image.



Click the "Open Data Sets" icon in the top left corner of the main window. This will prompt the Data Sets window:



20x20_average_pbc.dat

Import file to set number: 1

Array Dimension: 2-dim

Number Format: Ascii

Element Type: Scalar

Number Type: Real

Array Order: Row

Byte Offset: 0

Dim 1: 20 Dim 2: 20

This image shows the proper way to load the GridMAT-MD output. Click "Browse" to locate your data file within your local filesystem. Click the "Generic Array Format" tab in the middle of the window. In the "Dim1" field, enter the *x*-dimension of the grid, and in the "Dim2" field, enter the *y*-dimension of the grid.

Click "Import file to set number" to load the array, then close the Data Sets window if it does not do so automatically. The data will likely default to a 3-dimensional contour plot. Click on the "2-D Plots" icon to open the 2-D Plots menu. Simply click "Accept" in the lower left-hand corner of this menu to convert the display format from 3-D contour to 2-D matrix.

At this point you are ready to adjust the output appearance of your plot. Click the "Define Colormap" icon. In this menu, you will be able to adjust the coloring scheme of your plot, as well as the legend. Click "Edit Colormap Axis" to adjust the legend output style. If you need to adjust the data minimum/maximum to accommodate a more regular distribution of numbers, uncheck the "Automatic Min/Max" box and enter your own values. To show the legend on the plot, click the "Show" box, followed by "Accept." At this point, you should be able to move the legend to an appropriate location within the Main Window.

When you have generated the image you wish, click the "Print/Export Results" icon and choose the appropriate output format and settings for your image.

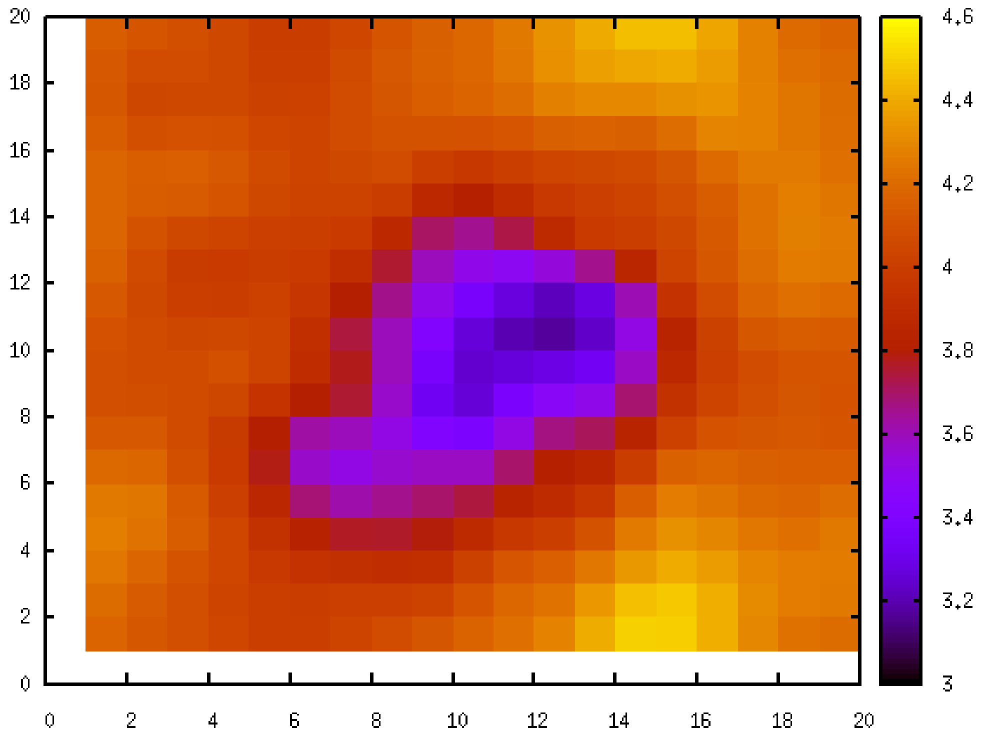
Section 2: Gnuplot

The matrix data format of the GridMAT-MD thickness output is simply a rearrangement of the single-column data into the following format:

```
{z-1-1}   {z-1-2}   {z-1-3}   ...

{z-2-1}   {z-2-2}   {z-2-3}   ...

...
```

Where each cell of the matrix is a record of the Z-value at each *x*-*y* pair of grid points. This format is easily readable into the Gnuplot program. Launch Gnuplot from the command line. Using the following commands will create the plot similar to the one shown below:

```
gnuplot> splot '20x20.average_thickness.dat' matrix using
(1+$1):(1+$2):3

gnuplot> set pm3d map

gnuplot> replot
```



Appendix B: Bug Fixes and Revisions

Version 2.0:

- Bug fix: Cleaned up STDOUT. Fixed by WJA, 04/18/2013.
- Bug fix: Several new and descriptive errors / warnings were added. Fixed by WJA, 04/18/2013.
- Bug fix: The CRYST1 record is now properly pulled from .pdb files. Fixed by WJA, 04/18/2013.
- New feature: Multiple-frame .pdb and .gro files can now be processed. Added by WJA, 04/18/2013.
- New feature: An arbitrarily large number of ligands can be considered in each coordinate file. Added by WJA, 04/18/2013.
- New feature: Three types of output file format can be generated (column, vector, matrix). Added by WJA, 04/18/2013.
- New feature: The output filename prefix is now customizable. Added by WJA, 04/18/2013.
- Updated documentation: New information has been added, especially concerning file preparation and updated options in the parameter file. Added by WJA, 04/18/2013.

Version 1.0.3:

- Bug fix: Incorrect handling of residue names in PDB files. Fixed by WJA, 12/15/2009.

Version 1.0.2:

- New feature: Output is now compatible with Gnuplot, via a post-processing script. Added by JAL, 6/29/2009.
- Updated documentation: New information about how to use Gnuplot. Added by JAL, 6/29/2009.

Version 1.0.1:

- New feature: Compatibility with .pdb files as input. Added by WJA, 1/15/2009.
- Updated documentation: Additional information about output format and how to use Xmatrix. Added by JAL, 2/4/2009.

Version 1.0:

- Bug fix: Incorrect thickness calculations in the absence of solvent. Fixed by WJA, 12/4/2008.