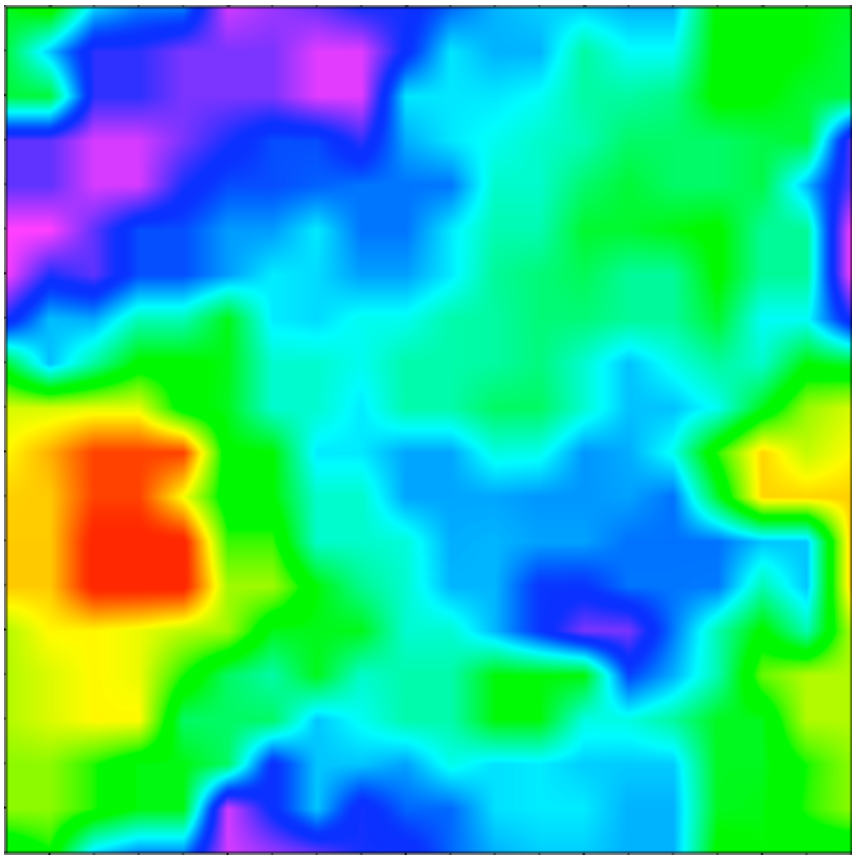
GridMAT-MD: A Grid-based Membrane Analysis Tool for use with Molecular Dynamics

William J. Allen, Justin A. Lemkul, and David R. Bevan
Department of Biochemistry, Virginia Tech



User's Guide

Version 1.0.2

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), but it is also possible to import the data in Gnuplot (http://www.gnuplot.info).

We believe this tool 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

Odds are, if you are reading this, you successfully downloaded and unzipped the appropriate `.tar` file from our website. Within you will find the following:

> `GridMAT-MD.pl`: The actual program.
> `param_example`: An example parameter file to help get you started
> `GridMAT-MD_ug.pdf`: The users guide
> `gpl-3.0.txt`: The GNU Public License, version 3.0

One of the wonderful 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.

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. The lipids in the coordinate file should be positioned so that their hydrophobic tails of one leaflet touch 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.

4. Running the Program

The program is run from the command line by typing either one of two commands:

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

Or, if you choose to `chmod +x GridMAT-MD.pl`:

```
./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.

5. Analysis of Output Files

There are five output files you could print with any one run, but they are rarely all needed at the same time. Three of them contain a single (different) column of numbers, the Z-values of the calculations performed by GridMAT-MD:

(`top_pbc.dat, bottom_pbc.dat, average_pbc.dat`)

```
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 would be called "`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})

This data is read as a matrix 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 raw format of the output data makes translation to other formats possible through scripting. An accessory script (`convert_to_gnuplot.pl`) is included in the GridMAT-MD distribution for converting the raw data output to a format suitable for use with Gnuplot.

The other type of data concerns area per lipid headgroup. If selected, the program will print out two separate files, one for the top and one for the bottom leaflet. It is a simple file that lists the average area per lipid head group on the top line, followed by the individual areas in the subsequent lines. The columns are divided into the lipid residue ID number (from the `.gro` or `.pdb` file), the Z-value, and the area:

```
(top_areas.dat, bottom_areas.dat)

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
…
```

Also, if you have a protein in the coordinate file and the `protein` parameter set to "`yes`", the lateral area of the protein will sneak its way into that file too. See section 6.4 for more on output files.

In all cases, the size of the grid (20 x 20, e.g.) is appended to the beginning of the output file name. This allows you to perform multiple runs with different grid sizes without the files writing over each other. And yes, they will write over each other.

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 an example option is listed on the first line, followed by the usage of that parameter on the following lines.

6.1 Input file and input file parameters

`bilayer` `my_bilayer.gro`
Usage: This is the filename of the `.pdb` or `.gro` format coordinate file. The program parses through the data in the file based on the extension, so if you have a PDB formatted file, make sure the extension is `.pdb`, etc. It is expected that there is some sort of lipid within this file. Also acceptable are protein atoms, solvent atoms, and ions.

`resname` `POPC`
Usage: The residue name of the lipid in your membrane, just as it appears in the `.gro` file. If you have a mixed bilayer, see `resname2`. This program does not currently work with bilayers that have more than two types of lipid, although the code could easily be extended to accommodate more than two types of lipids.

`atomname` `C12,C13,C32`
Usage: The `atomname` of the reference atom you wish to select. If you select more than one reference atom, the program computes the center of mass of those points. If you want to just pick one reference atom (phosphorus, for example), no commas are needed. Notice that these atoms are separated by commas and NO SPACES.

`resname2` `DPPC`
Usage: If you are analyzing a mixed bilayer, you will need to identify the residue name of the second lipid.

`atomname2` `P1`
Usage: If `resname2` is defined, `atomname2` must also be defined. Yes, even if they are the same as `atomname`.

`solvent` `SOL`
Usage: This is the name of the solvent used in your system. 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 your system here. This is really only necessary to help the program distinguish which atoms are protein atoms and which are not protein atoms. If you only have one ion, no commas are needed. Again, notice carefully that the ions are separated by commas and NO SPACES.

6.2 Grid size and shape

`box_size vectors`

Usage: The program can define the size of the box in two ways; either by the box `vectors` at the end of the coordinate file, or by the size of the `solvent` box. Therefore, the two options for `box_size` are "`vectors`" and "`solvent`" (it must be one or the other).

`vectors 5.5,10.0,6.9`

Usage: When `box_size` is set to "`vectors`", the program must look for vectors somewhere. It can pull the last line of the `.gro` file, but if the user provides a `.pdb` file, the vectors are not always listed. This option allows the user to set the appropriate vectors. Note that it only applies when you are using a `.pdb` file. Also note that the three vectors (x, y, and z) are separated by commas and NO SPACES.

`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.

`conserve_ratio yes`

Usage: In some cases, you may want 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`

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 you set `protein` to "`no`," even if your 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 Angstroms) here will result in selecting more protein atoms, and small values (0.5 -1.5 Angstroms) here will result in selecting fewer protein atoms. In out 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

`top_pbc yes`

Usage: If "`yes`", will print the bilayer thickness with respect to the top leaflet. Setting to "`no`" or anything other than "`yes`" will result in this data not being printed.

`bottom_pbc yes`

Usage: If "`yes`", will print the bilayer thickness with respect to the bottom leaflet. Setting to "`no`" or anything other than "`yes`" will result in this data not being printed.

`average_pbc yes`

Usage: If "`yes`", will average the matrices printed with top_pbc and bottom_pbc. This may be the best evaluation of the thickness of the lipid bilayer. Setting to "`no`" or anything other than "`yes`" will result in this data not being printed.

`top_areas yes`

Usage: If "`yes`", will print 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). If protein is defined, 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. Setting to "`no`" or anything other than "`yes`" will result in this data not being printed.

`bottom_areas yes`

Usage: `See top_areas`. Same, except for the bottom leaflet.

7. Example Analyses

7.1 Bilayer thickness calculation

```
param_file:

bilayer                 DPPC_bilayer.gro
resname                 DPPC
atomname                P8
solvent                 SOL
ions                    NA+,CL-

box_size                solvent
grid                    20
conserve_ratio          no

protein                 no

top_pbc                 yes
bottom_pbc              yes
average_pbc             yes
top_area                no
bottom_area             no
```

This parameter file is suited for making an image of the thickness of the bilayer. It reads from the coordinate file named `DPPC_bilayer.gro,` and 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` was set to "`no`". This does not necessarily mean there is no protein in the system. All this means is that, in order to obtain a nice smooth picture of the thickness of the bilayer, there is no need to consider the protein atoms. To see the difference, run this again later with `protein` set to "`yes`."

For the output, `top_area` and `bottom_area` are set to "`no`". With a grid resolution this low, it is hard to gather much data about the area per lipid headgroup, so this data is irrelevant. However, the other three outputs are set to "`yes`". The program will print three matrices of data – a representation of the thickness from the perspective of the top leaflet, a representation of thickness from the perspective of the bottom leaflet, and an average of those two matrices. If this is your first time running this program, print and visualize all three. Set them side-by-side and look at what is going on. Soon after you may realize that you really only need one matrix – `average_pbc.`

7.2 Area per lipid headgroup calculation

```
param_file:

bilayer                 mixed_bilayer.gro
resname                 DPPC
atomname                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

top_pbc                 yes
bottom_pbc              yes
average_pbc             no
top_area                yes
bottom_area             yes
```

This parameter file is designed to generate the data for area per lipid headgroup and a tessellation image. This time we are using a mixed bilayer, conveniently 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` at the bottom of 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_area` and `bottom_area`. If any protein atoms were found, the lateral area that the protein takes up will be listed in the appopriate area file

(depending on whether the protein is in the top, bottom, or both leaflets). You will also be able to see it in the corresponding pbc file. If it does not stand out in contrast enough, try increasing the `P_value`.

Appendix: Interfacing the Output of GridMAT-MD with Xmatrix and Gnuplot

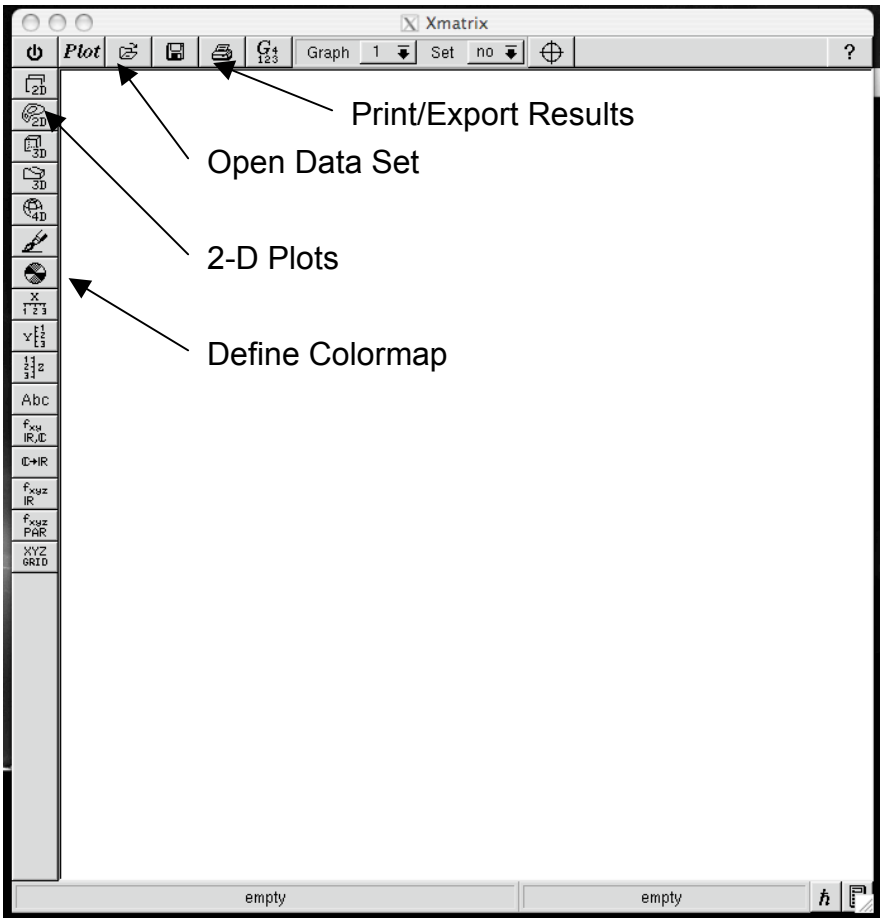
As of version 1.0.1 of GridMAT-MD, the only output format is a single-column data array. The number of lines in this file will correspond to the product of the grid dimensions specified when running the program. Thus, for a 20 x 20 matrix, there will be 400 lines, corresponding to the 20 *x*-grid points for each of the 20 *y*-grid points. The simplest use that we have found so far for this output format is to read the .dat file into the Xmatrix viewer, distributed as part of the Matpack C++ Numerics and Graphics Library (http://www.matpack.de). The data can also be formatted for use with Gnuplot (http://www.gnuplot.info).

Section 1. Xmatrix

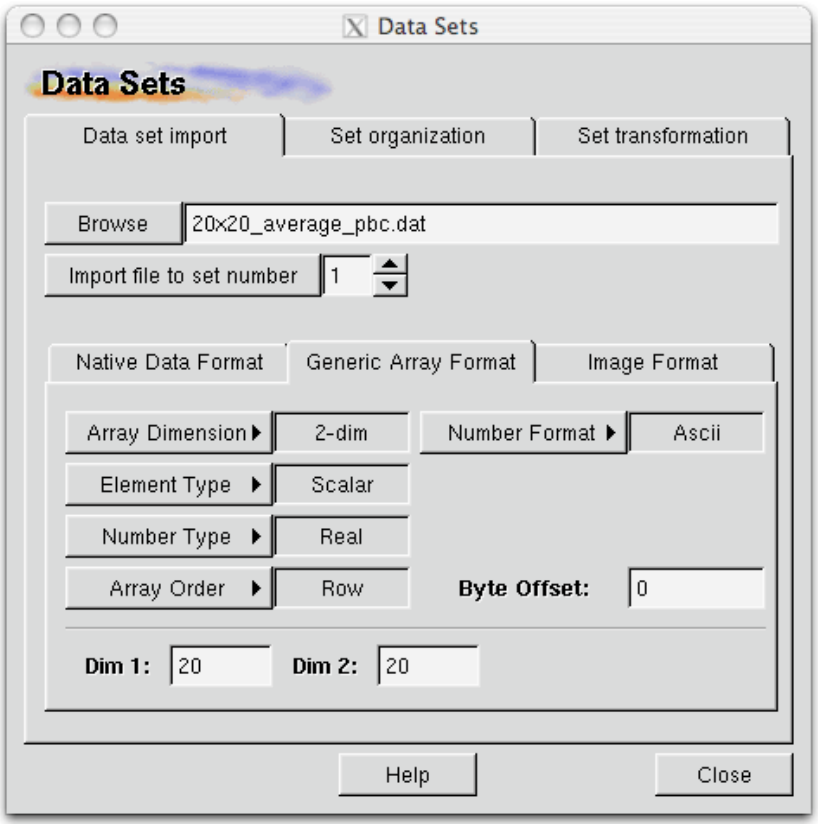
To read the GridMAT-MD output into Xmatrix, launch the program from the command line:

```
$ 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:



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 raw data format (single-column matrix) of the standard GridMAT-MD output makes it possible for scripts to alter the formatting very easily. We provide one such script (`convert_to_gnuplot.pl`) for just this purpose. The output of this script will be properly formatted for use as a Gnuplot surface. To execute the program, supply your data filename, as well as the *x*- and *y*-grid dimensions, i.e.:

```
perl convert_to_gnuplot.pl 20x20_average_pbc.dat 20 20
```

A new file will be created, called "20x20_average_pbc_gnuplot.dat." Launch Gnuplot from the command line. Using the following commands will create the plot similar to the one shown below:

```
gnuplot> splot '20x20_average_pbc_gnuplot.dat' matrix using
(1+$1):(1+$2):3

gnuplot> set pm3d map

gnuplot> replot
```