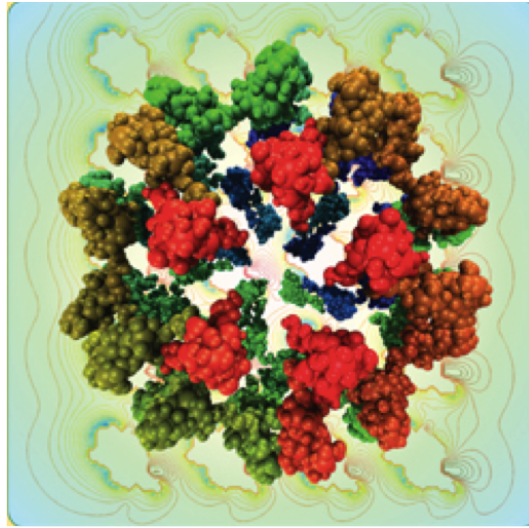


Reference Manual

Poisson Boltzmann Analytical Model (PB-AM)



Enghui-Yap
Albert Einstein College of Medicine

Lisa Felberg
Marielle Soniat
David Brookes
Teresa Head-Gordon
University of California, Berkeley

For more information, please visit <http://thglab.berkeley.edu>

Cover Illustration: An exploded view of a Brome Mosaic Virus capsid composed of $T = 1$ particles (PDB: 1YC6), represented as collections of overlapping spheres, is shown. PB-SAM is a new semi-analytical approach to efficiently solve the linearized Poisson-Boltzmann equation using multipole formalisms for overlapping spheres. The background shows the potential profile for an array of 601YC6 monomers computed using this method.

Recommended Citations:

When citing PB-AM in the literature, the following citation should be used

I. Lotan and T. Head-Gordon (2006). An analytical electrostatic model for salt screened interactions between multiple proteins J. Chem. Theory Comput 2, 541-555.

When citing PB-SAM in the literature, the following citations should be used

1. E.-H. Yap and T. Head-Gordon (2010). New and efficient Poisson-Boltzmann solver for interaction of multiple proteins J. Chem. Theory Comput. (Journal cover) 6, 2214-2224.
2. E.-H. Yap and T. Head-Gordon (2013). Calculating the bimolecular rate of protein-protein association with interacting crowders. J. Chem. Theory Comput. 9(5), 2481-2489.
3. O. N. Demerdash, E.-H. Yap and T. Head-Gordon (2014). Advanced potential energy surfaces for condensed phase simulation. Ann. Rev. Phys. Chem. 65, 149-174.

Acknowledgments: Research support from NIH and DOE is gratefully acknowledged.

Table of Contents

1	Introduction	2
1.1	PB-AM	2
1.2	PB-SAM	2
1.3	Physical Calculations	2
1.4	Brownian Dynamics	3
1.5	Electrostatics	3
2	The Poisson-Boltzmann Analytical Method	4
2.1	PB-AM formulation	4
2.1.1	Physical Calculations	4
2.1.2	Brownian Dynamics	5
2.1.3	Electrostatics	5
2.2	Use	6
2.2.1	Installation	6
2.3	Input Information	6
3	Example files and outputs	12
3.1	Physical calculations	12
3.2	Brownian Dynamics	13
3.3	Electrostatics	14
4	The Poisson-Boltzmann Semi Analytical Method	17
4.1	PB-SAM formulation	17
4.1.1	Physical Calculations	17
4.1.2	Brownian Dynamics	18
4.1.3	Electrostatics	18
4.2	Use	19
4.2.1	Installation	19
4.3	Input Information	19
5	Example files and outputs	25
5.1	Physical calculations	25
5.2	Brownian Dynamics	26
5.3	Electrostatics	27
6	Analysis Tools	30
6.1	Viewing PQR in VMD	30
6.2	Dynamics tools	30
6.2.1	Viewing Dynamics in VMD	30
6.3	Electrostatics tools	30
6.3.1	Viewing Electrostatics in VMD	30
6.3.2	2D ESP plots	32
6.3.3	3D ESP plots	32

7	Common Errors	34
7.1	Errors while reading input file	34
7.2	Segmentation Fault while reading input file	34
7.3	Initial configuration errors	34

Chapter 1

Introduction

The Poisson-Boltzmann Analytical and Semi-Analytical models solve the linearized Poisson-Boltzmann Equation (PBE) for systems hitherto not possible using traditional PBE solvers. This manual describes the methods and their associated suite of programs. We will refer to each method by their acronyms: PB-AM and PB-SAM. There is much shared capability between the two softwares, so when an application or usage can be applied to both, we will refer to both of them as PB-[S]AM, with the S in square brackets. The PBE software suite is licensed as a collection of freely available program under a GPL license.

1.1 PB-AM

The first general analytical solution for computing the screened electrostatic interaction between large numbers of macromolecules of arbitrarily complex charge distributions, assuming they are well described by spherical low dielectric cavities in a higher dielectric medium in the presence of a Debye-Hückel treatment of salt. The method exploits multipole expansion theory for the screened Coulomb potential such that it can describe direct charge-charge interactions and all higher-order cavity polarization effects between low dielectric spherical cavities containing their charges, while treating these higher order terms correctly at all separation distances. The analytical solution is general to arbitrary numbers of macromolecules, is efficient to compute, provides for the first time the ability to provide new benchmarks for other numerical solutions to the linearized Poisson-Boltzmann equation. A number of utilities are described below that use PB-AM results.

1.2 PB-SAM

A semi-analytical extension of the PB-AM method. It too is capable of computing the interactions of multiple molecules with complex charge distributions, but because of its semi-analytical approach, molecules are now represented as a collection of overlapping spheres. This allows for a more complex description of the molecule in comparison to the analytical method. The applications below also apply to the semi-analytical method.

1.3 Physical Calculations

The PB-[S]AM results allow for fast calculation of physically important quantities. Specifically this program can calculate the interaction energy of, and the force and torque applied to each molecule. These results may be written into a file or printed to the terminal.

1.4 Brownian Dynamics

This package also implements dynamics simulations using the Brownian protocol developed by Ermak and McCammon¹. At each time step, the translation and rotation due to the applied force and torque, respectively, are calculated and added to random components of motion. The user can specify spatial and temporal termination conditions for the simulation and write the trajectory to specified files.

1.5 Electrostatics

Another possible output of PB-AM, the user can specify a configuration of an arbitrary number of molecules and get a 2-dimensional or 3 dimensional potential landscape of the system. We have provided example plotting tools, and the 3-dimensional output may be uploaded in VMD for visualization.

¹Ermak, D. L.; McCammon, J. A. *J. Chem. Phys.* 1978, *69*, 1352–1360.

Chapter 2

The Poisson-Boltzmann Analytical Method

2.1 PB-AM formulation

PB-AM is an analytical solution to the linearized Poisson-Boltzmann equations for multiple spherical objects of arbitrary charge distribution in an ionic solution. The linearized Poisson-Boltzmann equation is given as:

$$\nabla[\epsilon(r)\nabla\phi(r)] - \epsilon(r)\kappa^2\phi(r) = 4\pi\rho(r)$$

$$\begin{aligned}\phi_{out}^{(i)} &= \phi_{out}^{(i)} \Big|_{r=a_i} \\ \epsilon_s \frac{\partial \phi_{out}^{(i)}}{\partial r} &= \epsilon_s \frac{\partial \phi_{out}^{(i)}}{\partial r} \Big|_{r=a_i}\end{aligned}$$

Exploiting fast-multipole methods, this boundary value problem can be reduced to the following system of linear equations.

$$A = \Gamma \cdot (\Delta \cdot T \cdot A + E)$$

$A(i)$ represents the effective multipole expansion of the charge distributions of molecule (i). $E(i)$ is the free charge distribution of molecule (i). Γ is a dielectric boundary-crossing operator, Δ a cavity polarization operator, T an operator that transforms the multipole expansion to a local coordinate frame. More details on the method are available in Lotan, Head-Gordon (2006).

2.1.1 Physical Calculations

From the above formulation, computation of the interaction energies ($\Omega^{(i)}$) is given as follows:

$$\Omega^{(i)} = \frac{1}{\epsilon_s} \left\langle \sum_{j \neq i}^N T \cdot A^{(j)}, A^{(i)} \right\rangle$$

Where $\langle M, N \rangle$ denotes the inner product. When energy is computed, forces follow as:

$$\mathbf{F}^{(i)} = \nabla_i \Omega^{(i)} = \frac{1}{\epsilon_s} [\langle \nabla_i T \cdot A^{(i)}, A^{(i)} \rangle + \langle T \cdot A^{(i)}, \nabla_i A^{(i)} \rangle]$$

The method to calculate the torque $\boldsymbol{\tau}^{(i)}$ on molecule is outside the scope of this manual, but is discussed extensively in Lotan, Head-Gordon (2006).

2.1.2 Brownian Dynamics

Brownian dynamics simulations are implemented by treating each molecule as a Brownian particle experiencing a conservative force $\mathbf{F}^{(i)}$ and torque $\boldsymbol{\tau}^{(i)}$, as well as friction and random force due to the solvent. The translation Δr_i and rotation $\Delta \theta_i$ for a time step Δt are then given by

$$\begin{aligned}\Delta r^{(i)} &= \frac{D_{i,trans}\Delta t}{k_B T} \mathbf{F}^{(i)} + \mathbf{S}_i(\Delta t) \\ \Delta \theta^{(i)} &= \frac{D_{i,rot}\Delta t}{k_B T} \boldsymbol{\tau}^{(i)} + \boldsymbol{\Theta}_i(\Delta t)\end{aligned}$$

where $D_{i,trans}$ and $D_{i,rot}$ are the translation and rotational diffusion coefficients for molecule i , respectively and $\mathbf{S}_i(\Delta t)$ and $\boldsymbol{\Theta}_i(\Delta t)$ are the stochastic components of translation and rotation, respectively, which have the following properties:

$$\begin{aligned}\langle \mathbf{S}_i \rangle &= 0, & \langle \mathbf{S}_i^2 \rangle &= 2D_{i,trans}\Delta t \\ \langle \boldsymbol{\Theta}_i \rangle &= 0, & \langle \boldsymbol{\Theta}_i^2 \rangle &= 2D_{i,rot}\Delta t\end{aligned}$$

2.1.3 Electrostatics

In a similar manner to the interaction energy calculations in the Physical Calculations section, the electrostatics computes the potential at any point in space exterior to the molecules, \mathbf{r} , as the inner product of all the solved effective multipole expansions $A^{(j)}$ with a local multipole expansion of a single positive charge at position, \mathbf{r} .

$$\Phi_{out}(\mathbf{r}) = \frac{1}{\epsilon_s} \left\langle \sum_{j=1}^N T \cdot A^{(j)}, A(\mathbf{r}) \right\rangle$$

2.2 Use

2.2.1 Installation

To install this program, first download the source code from the GitHub repository:

```
$ git clone https://github.com/davas301/pb_solvers.git clone_dir
```

Then navigate to the cloned directory `clone_dir`. From here, make a `build` directory, use CMake to create a make file and then make the program. This is done as follows:

```
$ cd clone_dir
$ mkdir build
$ cd build
$ cmake ..
$ make pbam
```

An executable named `pbam` will now be in `clone_dir/build/bin`.

2.3 Input Information

Input Files

The program executable requires an input file as a command line parameter. The input file contains the various arguments and parameters that one may wish to set when running the program. Each line of the input file contains a keyword followed by a variable number of whitespace-delimited parameters, e.g.:

```
keyword1    param1    param2
keyword2    param1    param2    param3
```

Each keyword is described in the table below, along with its associated parameters.

Keyword	Parameters	Description
runname	<name>	<name> is desired internal name of this run.
attypes	<numtypes>	Set the number of different atom types to <numtypes>.
pqr	<idx> <fpath>	The molecule index for this xyz file. Provide input PQR file at <fpath>.
xyz	<idx> <fpath>	The molecule index for this xyz file. Provide input XYZ file at <fpath>.
transrot	<idx> <fpath>	The molecule type index for this translation/rotation file that can be input instead of a xyz file. Provide input file at <fpath>.
randorient		If you want your molecules to be randomly rotated, use this flag
units	<units>	Set the units of output to <units>. The current options are: jmol (Joules/mole), kT (kT/e) and kcalmol (kCal/mole).
salt	<con>	Set salt concentration in the system to <con>.
temp	<T>	Set system temperature to <T>
idiel	<ival>	Set the interior dielectric constant to <ival>.
sdiel	<sval>	Set the solvent dielectric constant to <sval>.
pbx	<boxlength>	Set size of periodic box to <boxlength>.
random	<seed>	Seed the internal random number generator with <seed>.
type	<idx> <ct> <movetype> <dtr> <drot>	Set attributes of an atom type, where <idx> is the integer id of this type, which can be 1 to <numtypes> (above). <ct> is the number of atoms of this type in the system and <movetype> describes the way this type is allowed to move in a dynamics run (move, rot, or stat). If <movetype> is move, then a translational diffusion coefficient <dtr> and a rotational diffusional coefficient <drot> are required. If <movetype> is rot then just <drot> is required.

Keyword	Parameters	Description
runtype electrostatics	<gridpts>	Will run electrostatics calculations. <gridpts> is an optional integer describing the number of evenly spaced points in each dimension to perform calculations on.
dx	<fname>	For electrostatics. Will write the results of electrostatics calculations for every 3D grid point to <fname>, in the same output format as APBS dx file.
3dmap	<fname>	For electrostatics. Will write the results of electrostatics calculations for points on the surface of the molecules in the system.
gridct	<ct>	For electrostatics. <ct> is the number of 2D grids to output.
grid2d	<idx> <fname> <axis> <val>	For electrostatics. Set attributes of a grid output where <idx> is the integer id of this grid, which can be 1 to <ct> (above). Will write output of calculations for a cross section along <axis> (x, y, or z) at <value>.

Keyword	Parameters	Description
runtype dynamics	<outname> <ntraj>	Will perform a brownian dynamics run. A directory where trajectory information will be stored in and the number of trajectories is required.
termct	<ct>	Set number of termination conditions to <ct>.
termcombine	<andor>	Set how termination conditions will be combined. <andor> should be and or or . Default is or .
term	<idx> <type> <val> <mols>	Set attributes of a termination condition where <idx> is the integer id of this condition, which can be 1 to <ct> (above). <type> can be time , x<= , y<= , z<= , or r<= (or the >= equivalents of these), <val> is the value where the simulation will terminate and <mols> is a whitespace-delimited list of molecular indices that this condition applies to (time requires 0, and all else require 1).
term <idx> contact	<confile> <pad>	Set attributes of contact termination condition, where <idx> is the integer id of this condition, <confile> is a path to a file containing the contact information, and <pad> specifies a correction for the case when the contact distance cannot be reached due to the spherical assumption of the model. See below for more info.
xyz	<idx> <trajidx> <fpath>	<idx> is the molecule index for this xyz file. Provide input XYZ file at <fpath>. For the dynamics run, a starting configuration is needed for each trajectory for all the molecule types, so there should be <ntraj> xyz lines for each molecule, the trajectory number denoted by <trajidx>.
runtype energyforce	<outfilename>	Will calculate the interaction energy, the forces and torques for the system input. <outfilename> is a filename that you would like the information printed to. If none is entered, the information will be printed to the command line.

System inputs: PQR, XYZ, Translation/Rotation and Contact files

All the options above require a **PQR** file name. A PQR file can be generated from a PDB file using the PDB2PQR program, available as a web server or for download at:

http://nbc-222.ucsd.edu/pdb2pqr_1.9.0/

<http://www.poissonboltzmann.org/docs/pdb2pqr-installation/>

It may also be formatted manually. The general format of a PQR file is as follows, and is whitespace-delimited:

```
recName serial atName resName chainID resNum X Y Z charge rad
```

Parameter	Description
recName	A string that should either be ATOM or HETATM.
serial	An integer that provides the atom index
atName	A string that provides the atom name.
resName	A string that provides the residue name.
chainID	An optional string that provides the chain ID of the atom.
residueNumber	An integer that provides the residue index.
X Y Z	Three floats that provide the atomic coordinates.
charge	A float that provides the atomic charge (in electrons).
Rad	A float that provides the atomic radius (in Å).

The **XYZ** file simply specifies the desired molecule centers for a given molecule type.

```
mol1X mol1Y mol1Z
mol2X mol2Y mol2Z
mol3X mol3Y mol3Z
```

Translation/Rotation Instead of a XYZ file, one can input a file specifying the translations and rotations that should be applied to each molecule of a particular type. For these files, we follow the PDB standard for rotation matrices and translation vectors, which is as follows:

```
mol1 rot_1_11 rot_1_12 rot_1_13 trans_1_1
mol1 rot_1_21 rot_1_22 rot_1_23 trans_1_2
mol1 rot_1_31 rot_1_32 rot_1_33 trans_1_3
mol2 rot_2_11 rot_2_12 rot_2_13 trans_2_1
mol2 rot_2_21 rot_2_22 rot_2_23 trans_2_2
mol2 rot_2_31 rot_2_32 rot_2_33 trans_2_3
```

where **mol1** and **mol2** are indices of the molecule of the type this file applies to, **rot_i_jk** is the **j,k** index of the rotation matrix for molecule **i** and **trans_i_j** is the **j**th element in the translation vector for molecule **i**.

Contact files describe contacts between two molecular types. Generally this information is used to determine if a dynamics simulation should be terminated (e.g. terminate a simulation after two proteins have docked). The contact file contains lines with the format:

```
moltype1 at1 moltype2 at2 dist
```

where **moltype1** and **moltype2** are indices of the molecular types, **at1** is the index of an atom from the first molecular type, **at2** is the index of an atom from the second molecular type and **dist** is

the maximum distance between the two atoms that defines the contact. Note that sometimes these distances cannot be reached due to the assumption in this model that the molecule is spherical. To correct for this case, one must specify a "pad" distance that is defined as the maximum distance between the radial projections of the atoms onto the surface of their respective spheres that defines a contact.

Chapter 3

Example files and outputs

3.1 Physical calculations

Example physical calculations runfile

name: run.energyforce.inp:

```
1 runtype energyforce
2 runname energyforce.2sp.jmol.out
3
4 units jmol
5 salt 0.01
6 temp 353
7 idiel 4
8 sdiel 78
9
10 attypes 1
11 type 1 2
12 pqr 1 single_charge.pqr
13 xyz 1 positions_2.xyz
```

The files for PQR and XYZ are:

name: single_charge.pqr:

1	ATOM	1	N	NTR	0	1.000	0.000	0.000	-1.0000	3.7300
2	ATOM	1	N	NTR	0	0.000	1.000	0.000	-1.0000	6.3200

name: positions_2.xyz:

1	-10.0	23.4	-8.7
2	0.0	0.0	-2.5

To run:

```
1 $$ ../../bin/pbam run.energyforce.inp
```

And the resulting file:

name: energyforce.2sp.jmol.out:

```
1 My units are Joules/Mol
2 MOLECULE #1
3     POSITION: [-10, 23.4, -8.7]
4     ENERGY: 1328.86
5     FORCE: 1.19858e+08, [-36.6012 1.19858e+08 -1.65408e-05]
6     TORQUE: 1.78426e+06, [1.28425 1.78426e+06 1.9978e-06]
7 MOLECULE #2
8     POSITION: [0, 0, -2.5]
9     ENERGY: 1328.86
```

```

10 | FORCE: 1.19858e+08, [36.6012 -1.19858e+08 1.65408e-05]
11 | TORQUE: 1.78354e+06, [1.28372 1.78354e+06 1.99699e-06]

```

3.2 Brownian Dynamics

Example dynamics runfile

name: run.dynamics.inp:

```

1 runtype dynamics 2
2 runname dyn_cont_barn
3
4 salt 0.01
5 temp 298
6 idiel 4
7 sdiel 78
8
9 termct 1
10 termcombine or
11 term 1 contact 2.5 1 2
12
13 attypes 2
14 type 1 2 move 0.015 0.000045
15 pqr 1 1BRS_chainA.pqr
16 xyz 1 1 pos_1_1.xyz
17 xyz 1 2 pos_1_2.xyz
18
19 type 2 2 move 0.015 0.000045
20 pqr 2 1BRS_chainD.pqr
21 xyz 2 1 pos_2_1.xyz
22 xyz 2 2 pos_2_2.xyz

```

The files for PQR (first 5 lines) and XYZ files for the first trajectories are:

name: 1BRS_chainA.pqr:

1	ATOM	1700	N	ALA B	1	20.757	52.394	30.692	0.1414	1.8240
2	ATOM	1702	CA	ALA B	1	20.602	52.680	29.268	0.0962	1.9080
3	ATOM	1703	C	ALA B	1	19.286	52.138	28.675	0.6163	1.9080
4	ATOM	1704	O	ALA B	1	18.578	51.351	29.318	-0.5722	1.6612
5	ATOM	1705	CB	ALA B	1	21.739	52.033	28.476	-0.0597	1.9080

name: pos_1_1.xyz:

```

1 61.25 61.25 61.25
2 -26.25 61.25 -26.25

```

name: 1BRS_chainD.pqr:

1	ATOM	1	N	LYS D	1	48.330	40.393	9.798	0.0966	1.8240
2	ATOM	2	CA	LYS D	1	47.401	39.287	9.370	-0.0015	1.9080
3	ATOM	3	C	LYS D	1	47.507	38.911	7.890	0.7214	1.9080
4	ATOM	4	O	LYS D	1	47.126	39.582	6.905	-0.6013	1.6612
5	ATOM	5	CB	LYS D	1	45.995	39.632	9.817	0.0212	1.9080

name: pos_2_1.xyz:

```

1 -26.25 61.25 61.25
2 61.25 -26.25 61.25

```

To run:


```
1 $$ ../../bin/pbam run.dynamics.inp
```

And the resulting files:

name: `dyn_cont_barn_[traj#].xyz`: VMD readable XYZ file that shows the trajectory of molecules in the system. The time that is snapshot was printed from is given on the same line as the word Atom. The atoms of your input file are currently labeled N, and the coarse-grain center is labeled "X" in the first column of the XYZ file.

```
1 3135
2 Atoms. Timestep (ps): 0
3 N -7.241 -0.530 18.703
4 N -6.015 -0.503 17.910
5 N -5.784 0.840 17.188
6 N -6.682 1.690 17.128
7 N -6.066 -1.580 16.827
8 N -7.519 -1.481 18.863
9 N -7.084 -0.079 19.584
```

name: `dyn_cont_barn_[traj#].dat`: Statistics from simulation printed out at the same time as each XYZ snapshot. The energy is not computed and should be ignored.

```
1 My units are Internal. Time (ps) 500.4
2 MOLECULE #1
3   POSITION: [0, 0, 0]
4   ENERGY: 0
5   FORCE: 3.39124e-06, [1.69863e-06 2.07547e-06 6.5356e-07]
6   TORQUE: 2.55224e-05, [-2.11728e-05 1.00774e-05 3.08631e-05]
7 MOLECULE #2
8   POSITION: [87.211, 43.861, 21.691]
9   ENERGY: 0
10  FORCE: 3.65373e-06, [-1.87502e-06 -2.21744e-06 -7.27314e-07]
11  TORQUE: 1.91656e-05, [8.14396e-06 -1.22678e-05 1.56284e-05]
```

name: `dyn_nam_barn.stat` : Details about how each simulation has terminated and the time at which this occurred.

```
1 Molecule type 1 has fulfilled condition: r >= 500.00; at time (ps) 1.32367e+06
2 Molecule type 1 has fulfilled condition: r >= 500.00; at time (ps) 1.15712e+06
3 System has fulfilled condition: Type 0 and Type 1 are within 2.50; at time (ps)
  1.90603e+06
4 Molecule type 1 has fulfilled condition: r >= 500.00; at time (ps) 2.18533e+06
5 System has fulfilled condition: Type 0 and Type 1 are within 2.50; at time (ps)
  1.59066e+06
```

3.3 Electrostatics

Example electrostatics runfile

name: `run.electrostatic.inp`:

```
1 runtype electrostatics 140
2 runname electrostatic
3
4 units kT
5 salt 0.01
6 temp 298
7 idiel 4
8 sdiel 78
9
```

```

10 dx out.dx
11
12 3dmap electro_map.out
13
14 gridct 2
15 grid2D 1 out.x.0.dat x 0
16 grid2D 2 out.x.-1.dat x -1
17
18 attypes 2
19 type 1 2
20 pqr 1 single_charge.pqr
21 xyz 1 positions_2.xyz
22
23 type 2 2
24 pqr 2 pos_charge.pqr
25 xyz 2 positions_pos.xyz

```

The files for PQR and XYZ files are:

name: `single_charge.pqr`:

1	ATOM	1	N	NTR	0	0.000	1.000	0.000	4.0000	0.3200
2	ATOM	1	N	NTR	0	0.000	0.000	-1.000	4.0000	0.3200
3	ATOM	1	X	CEN	0	0.000	0.000	0.000	0.0000	2.0000

name: `positions_2.xyz`:

1	0.0	0.0	-5.0
2	0.0	0.0	5.0

name: `pos_charge.pqr`:

1	ATOM	1	N	NTR	0	0.000	1.000	0.000	-4.0000	0.3200
2	ATOM	1	N	NTR	0	0.000	0.000	-1.000	-4.0000	0.3200
3	ATOM	1	X	CEN	0	0.000	0.000	0.000	0.0000	2.0000

name: `positions_pos.xyz`:

1	0.0	5.0	0.0
2	0.0	-5.0	0.0

To run:

```

1 $$ ../../bin/pbam run.electrostatic.inp

```

And the resulting files:

name: `out.dx`:

```

1 # Data from PBAM Electrostat run
2 # My runname is out.dx and units kT/e
3 object 1 class gridpositions counts 140 140 140
4 origin -4 -9 -9
5 delta 0.0571429 0.0e+00 0.0e+00
6 delta 0.0e00 0.128571 0.0e+00
7 delta 0.0e00 0.0e+00 0.128571
8 object 2 class gridconnections counts 140 140 140
9 object 3 class array type double rank 0 items 2744000 data follows
10 2.7203115e-01 3.0271755e-01 3.3459723e-01
11 3.6769040e-01 4.0201595e-01 4.3759129e-01
12 .....
13 -1.3185519e-01 -1.5849252e-01 -1.8359631e-01
14 -2.0722087e-01 -2.2942006e-01 -2.5024714e-01
15 -2.6975467e-01 -2.8799442e-01

```

```

16 attribute "dep" string "positions"
17 object "regular positions regular connections" class field
18 component "positions" value 1
19 component "connections" value 2
20 component "data" value 3

```

name: `electro_map.out`:

```

1 # Data from PBAM Electrostat run
2 # My runname is electro_map.out and units kT/e
3 grid 10 10 10
4 origin -4 -9 -9
5 delta 0.8 1.8 1.8
6   0.00825   0.00006  -2.90002  -5.899956
7   0.00822   0.00071  -2.90002  -5.902602

```

name: `out.x.0.dat`:

```

1 # Data from PBAM Electrostat run
2 # My runname is out.x.0.dat
3 units kT
4 grid 140 140
5 axis x 0
6 origin -9 -9
7 delta 0.128571 0.128571
8 maxmin 39.23 -39.23
9   0.3605004   0.4030045   0.4474874   0.4940082   0.5426260   0.5933995

```

Chapter 4

The Poisson-Boltzmann Semi Analytical Method

4.1 PB-SAM formulation

PB-SAM builds on the PB-AM solution to the linearized Poisson-Boltzmann equation (LPBE). While PB-AM gives a general solution to the LPBE for an arbitrary number of spherical objects, the PB-SAM formulation extends that solution to arbitrary shape. Instead of representing a macromolecule (denoted as I) as a single sphere, each molecule is represented as a collection of overlapping spheres (with each sphere denoted as k). The computation of the interaction energies remains essentially the same self-consistent equation as Eqn. ??

$$\Omega^{(i)} = \frac{1}{\epsilon_s} \sum_{k=1}^{N_S^{(I)}} \langle \underline{T} \cdot \underline{A}^{(i,k)}, \underline{A}^{(i,k)} \rangle \quad (4.1)$$

However, now the interaction energy of a molecule $\Omega^{(i)}$ is the sum of the interaction energies of all its constituent spheres.

Additional complexity is due to overlap between spheres. Whereas PB-AM has a single type of interaction (between separated spheres), PB-SAM has three types of interactions:

- i. between spheres in different molecules
- ii. between overlapping spheres within the same molecule
- iii. between non-overlapping spheres within the same molecule

There is no dielectric discontinuity between spheres within the same macromolecule. More details on the method are available in Yap, Head-Gordon (2010).

4.1.1 Physical Calculations

From the above formulation, computation of the interaction energies ($\Omega^{(i)}$) is given as follows:

$$\Omega^{(i)} = \frac{1}{\epsilon_s} \left\langle \sum_{j \neq i}^N T \cdot A^{(j)}, A^{(i)} \right\rangle$$

Where $\langle M, N \rangle$ denotes the inner product. When energy is computed, forces follow as:

$$\mathbf{F}^{(i)} = \nabla_i \Omega^{(i)} = \frac{1}{\epsilon_s} [\langle \nabla_i T \cdot A^{(i)}, A^{(i)} \rangle + \langle T \cdot A^{(i)}, \nabla_i A^{(i)} \rangle]$$

The method to calculate the torque $\boldsymbol{\tau}^{(i)}$ on molecule is outside the scope of this manual, but is discussed extensively in Lotan, Head-Gordon (2006).

4.1.2 Brownian Dynamics

Brownian dynamics simulations are implemented by treating each molecule as a Brownian particle experiencing a conservative force $\mathbf{F}^{(i)}$ and torque $\boldsymbol{\tau}^{(i)}$, as well as friction and random force due to the solvent. The translation Δr_i and rotation $\Delta \theta_i$ for a time step Δt are then given by

$$\begin{aligned}\Delta r^{(i)} &= \frac{D_{i,trans}\Delta t}{k_B T} \mathbf{F}^{(i)} + \mathbf{S}_i(\Delta t) \\ \Delta \theta^{(i)} &= \frac{D_{i,rot}\Delta t}{k_B T} \boldsymbol{\tau}^{(i)} + \boldsymbol{\Theta}_i(\Delta t)\end{aligned}$$

where $D_{i,trans}$ and $D_{i,rot}$ are the translation and rotational diffusion coefficients for molecule i , respectively and $\mathbf{S}_i(\Delta t)$ and $\boldsymbol{\Theta}_i(\Delta t)$ are the stochastic components of translation and rotation, respectively, which have the following properties:

$$\begin{aligned}\langle \mathbf{S}_i \rangle &= 0, & \langle \mathbf{S}_i^2 \rangle &= 2D_{i,trans}\Delta t \\ \langle \boldsymbol{\Theta}_i \rangle &= 0, & \langle \boldsymbol{\Theta}_i^2 \rangle &= 2D_{i,rot}\Delta t\end{aligned}$$

4.1.3 Electrostatics

In a similar manner to the interaction energy calculations in the Physical Calculations section, the electrostatics computes the potential at any point in space exterior to the molecules, \mathbf{r} , as the inner product of all the solved effective multipole expansions $A^{(j)}$ with a local multipole expansion of a single positive charge at position, \mathbf{r} .

$$\Phi_{out}(\mathbf{r}) = \frac{1}{\epsilon_s} \left\langle \sum_{j=1}^N T \cdot A^{(j)}, A(\mathbf{r}) \right\rangle$$

4.2 Use

4.2.1 Installation

To install this program, first download the source code from the GitHub repository:

```
$ git clone https://github.com/davas301/pb_solvers.git clone_dir
```

Then navigate to the cloned directory `clone_dir`. From here, make a `build` directory, use CMake to create a make file and then make the program. This is done as follows:

```
$ cd clone_dir
$ mkdir build
$ cd build
$ cmake ..
$ make pbsam
```

An executable named `pbsam` will now be in `clone_dir/build/bin`.

4.3 Input Information

Input Files

The program executable requires an input file as a command line parameter. The input file contains the various arguments and parameters that one may wish to set when running the program. Each line of the input file contains a keyword followed by a variable number of whitespace-delimited parameters, e.g.:

```
keyword1    param1    param2
keyword2    param1    param2    param3
```

Each keyword is described in the table below, along with its associated parameters.

Keyword	Parameters	Description
runname	<name>	<name> is desired internal name of this run.
atypes	<numtypes>	Set the number of different atom types to <numtypes>.
pqr	<idx> <fpath>	The molecule index for this xyz file. Provide input PQR file at <fpath>.
xyz	<idx> <fpath>	The molecule index for this xyz file. Provide input XYZ file at <fpath>.
transrot	<idx> <fpath>	The molecule type index for this translation/rotation file that can be input instead of a xyz file. Provide input file at <fpath>.
randorient		If you want your molecules to be randomly rotated, use this flag
units	<units>	Set the units of output to <units>. The current options are: jmol (Joules/mole), kT and kcalmol (kCal/mole).
salt	<con>	Set salt concentration in the system to <con>.
temp	<T>	Set system temperature to <T>
idiel	<ival>	Set the interior dielectric constant to <ival>.
sdiel	<sval>	Set the solvent dielectric constant to <sval>.
pbc	<boxlength>	Set size of periodic box to <boxlength>.
random	<seed>	Seed the internal random number generator with <seed>.
type	<idx> <ct> <movetype> <dtr> <drot>	Set attributes of an atom type, where <idx> is the integer id of this type, which can be 1 to <numtypes> (above). <ct> is the number of atoms of this type in the system and <movetype> describes the way this type is allowed to move in a dynamics run (move, rot, or stat). If <movetype> is move, then a translational diffusion coefficient <dtr> and a rotational diffusional coefficient <drot> are required. If <movetype> is rot then just <drot> is required.

Keyword	Parameters	Description
runtype electrostatics	<gridpts>	Will run electrostatics calculations. <gridpts> is an optional integer describing the number of evenly spaced points in each dimension to perform calculations on.
dx	<fname>	For electrostatics. Will write the results of electrostatics calculations for every 3D grid point to <fname>, in the same output format as APBS dx file.
3dmap	<fname>	For electrostatics. Will write the results of electrostatics calculations for points on the surface of the molecules in the system.
gridct	<ct>	For electrostatics. <ct> is the number of 2D grids to output.
grid2d	<idx> <fname> <axis> <val>	For electrostatics. Set attributes of a grid output where <idx> is the integer id of this grid, which can be 1 to <ct> (above). Will write output of calculations for a cross section along <axis> (x, y, or z) at <value>.

Keyword	Parameters	Description
runtype dynamics	<outname> <ntraj>	Will perform a brownian dynamics run. A directory where trajectory information will be stored in and the number of trajectories is required.
termct	<ct>	Set number of termination conditions to <ct>.
termcombine	<andor>	Set how termination conditions will be combined. <andor> should be and or or . Default is or .
term	<idx> <type> <val> <mols>	Set attributes of a termination condition where <idx> is the integer id of this condition, which can be 1 to <ct> (above). <type> can be time , x<= , y<= , z<= , or r<= (or the >= equivalents of these), <val> is the value where the simulation will terminate and <mols> is a whitespace-delimited list of molecular indices that this condition applies to (time requires 0, and all else require 1).
term <idx> contact	<confile> <pad>	Set attributes of contact termination condition, where <idx> is the integer id of this condition, <confile> is a path to a file containing the contact information, and <pad> specifies a correction for the case when the contact distance cannot be reached due to the spherical assumption of the model. See below for more info.
xyz	<idx> <trajidx> <fpath>	<idx> is the molecule index for this xyz file. Provide input XYZ file at <fpath>. For the dynamics run, a starting configuration is needed for each trajectory for all the molecule types, so there should be <ntraj> xyz lines for each molecule, the trajectory number denoted by <trajidx>.
runtype energyforce	<outfilename>	Will calculate the interaction energy, the forces and torques for the system input. <outfilename> is a filename that you would like the information printed to. If none is entered, the information will be printed to the command line.

System inputs: PQR, XYZ, Translation/Rotation and Contact files

All the options above require a **PQR** file name. A PQR file can be generated from a PDB file using the PDB2PQR program, available as a web server or for download at:

http://nbc-222.ucsd.edu/pdb2pqr_1.9.0/

<http://www.poissonboltzmann.org/docs/pdb2pqr-installation/>

It may also be formatted manually. The general format of a PQR file is as follows, and is whitespace-delimited:

```
recName serial atName resName chainID resNum X Y Z charge rad
```

Parameter	Description
recName	A string that should either be ATOM or HETATM.
serial	An integer that provides the atom index
atName	A string that provides the atom name.
resName	A string that provides the residue name.
chainID	An optional string that provides the chain ID of the atom.
residueNumber	An integer that provides the residue index.
X Y Z	Three floats that provide the atomic coordinates.
charge	A float that provides the atomic charge (in electrons).
Rad	A float that provides the atomic radius (in Å).

The **XYZ** file simply specifies the desired molecule centers for a given molecule type.

```
mol1X mol1Y mol1Z
mol2X mol2Y mol2Z
mol3X mol3Y mol3Z
```

Translation/Rotation Instead of a XYZ file, one can input a file specifying the translations and rotations that should be applied to each molecule of a particular type. For these files, we follow the PDB standard for rotation matrices and translation vectors, which is as follows:

```
mol1 rot_1_11 rot_1_12 rot_1_13 trans_1_1
mol1 rot_1_21 rot_1_22 rot_1_23 trans_1_2
mol1 rot_1_31 rot_1_32 rot_1_33 trans_1_3
mol2 rot_2_11 rot_2_12 rot_2_13 trans_2_1
mol2 rot_2_21 rot_2_22 rot_2_23 trans_2_2
mol2 rot_2_31 rot_2_32 rot_2_33 trans_2_3
```

where **mol1** and **mol2** are indices of the molecule of the type this file applies to, **rot_i_jk** is the **j,k** index of the rotation matrix for molecule **i** and **trans_i_j** is the **j**th element in the translation vector for molecule **i**.

Contact files describe contacts between two molecular types. Generally this information is used to determine if a dynamics simulation should be terminated (e.g. terminate a simulation after two proteins have docked). The contact file contains lines with the format:

```
moltype1 at1 moltype2 at2 dist
```

where **moltype1** and **moltype2** are indices of the molecular types, **at1** is the index of an atom from the first molecular type, **at2** is the index of an atom from the second molecular type and **dist** is

the maximum distance between the two atoms that defines the contact. Note that sometimes these distances cannot be reached due to the assumption in this model that the molecule is spherical. To correct for this case, one must specify a `paddistance` that is defined as the maximum distance between the radial projections of the atoms onto the surface of their respective spheres that defines a contact.

Chapter 5

Example files and outputs

5.1 Physical calculations

Example physical calculations runfile

name: run.energyforce.inp:

```
1 runtype energyforce
2 runname energyforce.2sp.jmol.out
3
4 units jmol
5 salt 0.01
6 temp 353
7 idiel 4
8 sdiel 78
9
10 attypes 1
11 type 1 2
12 pqr 1 single_charge.pqr
13 xyz 1 positions_2.xyz
```

The files for PQR and XYZ are:

name: single_charge.pqr:

1	ATOM	1	N	NTR	0	1.000	0.000	0.000	-1.0000	3.7300
2	ATOM	1	N	NTR	0	0.000	1.000	0.000	-1.0000	6.3200

name: positions_2.xyz:

1	-10.0	23.4	-8.7
2	0.0	0.0	-2.5

To run:

```
1 $$ ../../bin/pbam run.energyforce.inp
```

And the resulting file:

name: energyforce.2sp.jmol.out:

```
1 My units are Joules/Mol
2 MOLECULE #1
3     POSITION: [-10, 23.4, -8.7]
4     ENERGY: 1328.86
5     FORCE: 1.19858e+08, [-36.6012 1.19858e+08 -1.65408e-05]
6     TORQUE: 1.78426e+06, [1.28425 1.78426e+06 1.9978e-06]
7 MOLECULE #2
8     POSITION: [0, 0, -2.5]
9     ENERGY: 1328.86
```

```

10 | FORCE: 1.19858e+08, [36.6012 -1.19858e+08 1.65408e-05]
11 | TORQUE: 1.78354e+06, [1.28372 1.78354e+06 1.99699e-06]

```

5.2 Brownian Dynamics

Example dynamics runfile

name: run.dynamics.inp:

```

1 runtype dynamics 2
2 runname dyn_cont_barn
3
4 salt 0.01
5 temp 298
6 idiel 4
7 sdiel 78
8
9 termct 1
10 termcombine or
11 term 1 contact 2.5 1 2
12
13 attypes 2
14 type 1 2 move 0.015 0.000045
15 pqr 1 1BRS_chainA.pqr
16 xyz 1 1 pos_1_1.xyz
17 xyz 1 2 pos_1_2.xyz
18
19 type 2 2 move 0.015 0.000045
20 pqr 2 1BRS_chainD.pqr
21 xyz 2 1 pos_2_1.xyz
22 xyz 2 2 pos_2_2.xyz

```

The files for PQR (first 5 lines) and XYZ files for the first trajectories are:

name: 1BRS_chainA.pqr:

1	ATOM	1700	N	ALA B	1	20.757	52.394	30.692	0.1414	1.8240
2	ATOM	1702	CA	ALA B	1	20.602	52.680	29.268	0.0962	1.9080
3	ATOM	1703	C	ALA B	1	19.286	52.138	28.675	0.6163	1.9080
4	ATOM	1704	O	ALA B	1	18.578	51.351	29.318	-0.5722	1.6612
5	ATOM	1705	CB	ALA B	1	21.739	52.033	28.476	-0.0597	1.9080

name: pos_1_1.xyz:

```

1 61.25 61.25 61.25
2 -26.25 61.25 -26.25

```

name: 1BRS_chainD.pqr:

1	ATOM	1	N	LYS D	1	48.330	40.393	9.798	0.0966	1.8240
2	ATOM	2	CA	LYS D	1	47.401	39.287	9.370	-0.0015	1.9080
3	ATOM	3	C	LYS D	1	47.507	38.911	7.890	0.7214	1.9080
4	ATOM	4	O	LYS D	1	47.126	39.582	6.905	-0.6013	1.6612
5	ATOM	5	CB	LYS D	1	45.995	39.632	9.817	0.0212	1.9080

name: pos_2_1.xyz:

```

1 -26.25 61.25 61.25
2 61.25 -26.25 61.25

```

To run:

```
1 $$ ../../bin/pbam run.dynamics.inp
```

And the resulting files:

name: `dyn_cont_barn_[traj#].xyz`: VMD readable XYZ file that shows the trajectory of molecules in the system. The time that is snapshot was printed from is given on the same line as the word Atom. The atoms of your input file are currently labeled N, and the coarse-grain center is labeled "X" in the first column of the XYZ file.

```
1 3135
2 Atoms. Timestep (ps): 0
3 N -7.241 -0.530 18.703
4 N -6.015 -0.503 17.910
5 N -5.784 0.840 17.188
6 N -6.682 1.690 17.128
7 N -6.066 -1.580 16.827
8 N -7.519 -1.481 18.863
9 N -7.084 -0.079 19.584
```

name: `dyn_cont_barn_[traj#].dat`: Statistics from simulation printed out at the same time as each XYZ snapshot. The energy is not computed and should be ignored.

```
1 My units are Internal. Time (ps) 500.4
2 MOLECULE #1
3   POSITION: [0, 0, 0]
4   ENERGY: 0
5   FORCE: 3.39124e-06, [1.69863e-06 2.07547e-06 6.5356e-07]
6   TORQUE: 2.55224e-05, [-2.11728e-05 1.00774e-05 3.08631e-05]
7 MOLECULE #2
8   POSITION: [87.211, 43.861, 21.691]
9   ENERGY: 0
10  FORCE: 3.65373e-06, [-1.87502e-06 -2.21744e-06 -7.27314e-07]
11  TORQUE: 1.91656e-05, [8.14396e-06 -1.22678e-05 1.56284e-05]
```

name: `dyn_nam_barn.stat` : Details about how each simulation has terminated and the time at which this occurred.

```
1 Molecule type 1 has fulfilled condition: r >= 500.00; at time (ps) 1.32367e+06
2 Molecule type 1 has fulfilled condition: r >= 500.00; at time (ps) 1.15712e+06
3 System has fulfilled condition: Type 0 and Type 1 are within 2.50; at time (ps)
  1.90603e+06
4 Molecule type 1 has fulfilled condition: r >= 500.00; at time (ps) 2.18533e+06
5 System has fulfilled condition: Type 0 and Type 1 are within 2.50; at time (ps)
  1.59066e+06
```

5.3 Electrostatics

Example electrostatics runfile

name: `run.electrostatic.inp`:

```
1 runtype electrostatics 140
2 runname electrostatic
3
4 units kT
5 salt 0.01
6 temp 298
7 idiel 4
8 sdiel 78
9
```

```

10 dx out.dx
11
12 3dmap electro_map.out
13
14 gridct 2
15 grid2D 1 out.x.0.dat x 0
16 grid2D 2 out.x.-1.dat x -1
17
18 attypes 2
19 type 1 2
20 pqr 1 single_charge.pqr
21 xyz 1 positions_2.xyz
22
23 type 2 2
24 pqr 2 pos_charge.pqr
25 xyz 2 positions_pos.xyz

```

The files for PQR and XYZ files are:

name: `single_charge.pqr`:

1	ATOM	1	N	NTR	0	0.000	1.000	0.000	4.0000	0.3200
2	ATOM	1	N	NTR	0	0.000	0.000	-1.000	4.0000	0.3200
3	ATOM	1	X	CEN	0	0.000	0.000	0.000	0.0000	2.0000

name: `positions_2.xyz`:

1	0.0	0.0	-5.0
2	0.0	0.0	5.0

name: `pos_charge.pqr`:

1	ATOM	1	N	NTR	0	0.000	1.000	0.000	-4.0000	0.3200
2	ATOM	1	N	NTR	0	0.000	0.000	-1.000	-4.0000	0.3200
3	ATOM	1	X	CEN	0	0.000	0.000	0.000	0.0000	2.0000

name: `positions_pos.xyz`:

1	0.0	5.0	0.0
2	0.0	-5.0	0.0

To run:

```

1 $$ ../../bin/pbam run.electrostatic.inp

```

And the resulting files:

name: `out.dx`:

```

1 # Data from PBAM Electrostat run
2 # My runname is out.dx and units kT
3 object 1 class gridpositions counts 140 140 140
4 origin -4 -9 -9
5 delta 0.0571429 0.0e+00 0.0e+00
6 delta 0.0e00 0.128571 0.0e+00
7 delta 0.0e00 0.0e+00 0.128571
8 object 2 class gridconnections counts 140 140 140
9 object 3 class array type double rank 0 items 2744000 data follows
10 2.7203115e-01 3.0271755e-01 3.3459723e-01
11 3.6769040e-01 4.0201595e-01 4.3759129e-01
12 .....
13 -1.3185519e-01 -1.5849252e-01 -1.8359631e-01
14 -2.0722087e-01 -2.2942006e-01 -2.5024714e-01
15 -2.6975467e-01 -2.8799442e-01

```

```

16 attribute "dep" string "positions"
17 object "regular positions regular connections" class field
18 component "positions" value 1
19 component "connections" value 2
20 component "data" value 3

```

name: `electro_map.out`:

```

1 # Data from PBAM Electrostat run
2 # My runname is electro_map.out and units kT
3 grid 10 10 10
4 origin -4 -9 -9
5 delta 0.8 1.8 1.8
6   0.00825   0.00006  -2.90002  -5.899956
7   0.00822   0.00071  -2.90002  -5.902602

```

name: `out.x.0.dat`:

```

1 # Data from PBAM Electrostat run
2 # My runname is out.x.0.dat
3 units kT
4 grid 140 140
5 axis x 0
6 origin -9 -9
7 delta 0.128571 0.128571
8 maxmin 39.23 -39.23
9   0.3605004   0.4030045   0.4474874   0.4940082   0.5426260   0.5933995

```


Chapter 6

Analysis Tools

6.1 Viewing PQR in VMD

Load pqr file

```
1 >> set selall [atomselect top "all"]
```

To center the pqr

```
1 >> $selall moveby {-x -y -z}
```

where x, y, and z are half the box length

Graphical representations: To view the charges inside the CG center, from the toolbar, select Graphics > Representations. In the selected atoms, type

```
1 not name X
```

Change the coloring method to Charge, and the Drawing Method to VDW. Then select the Create Rep button, and in the selected atoms, type

```
1 not name X
```

Change the Drawing Method to VDW and the Material to Transparent. The Graphical Representations and final images are in figure below.

6.2 Dynamics tools

6.2.1 Viewing Dynamics in VMD

First load the PQR file, and then any XYZ files onto the same molecule.

6.3 Electrostatics tools

6.3.1 Viewing Electrostatics in VMD

To load electrostatic results: File > New Molecule. In the window that appears, toggle the *Load files for:* to select the currently loaded PQR file. Then select *Browse* to find the location of the dx file. Once found, hit the *Load* button and let the dx file load.

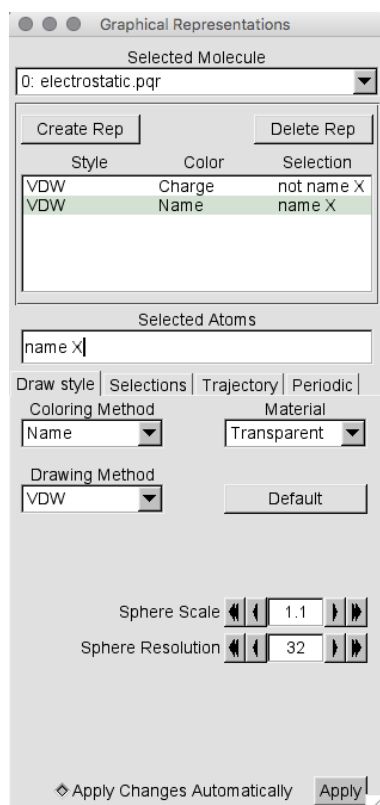


Figure 6.1: Graphics

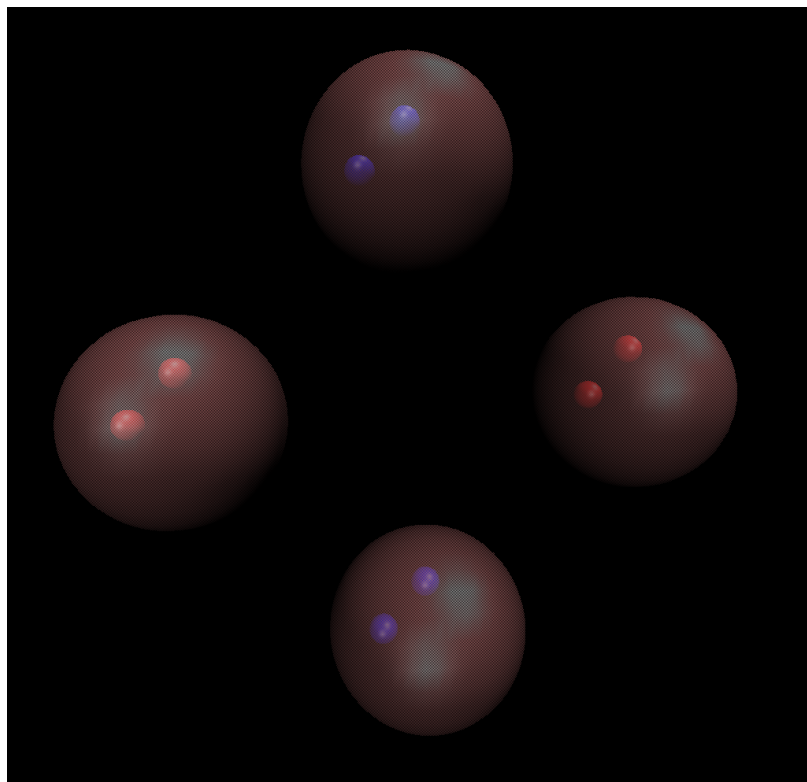


Figure 6.2: Final view

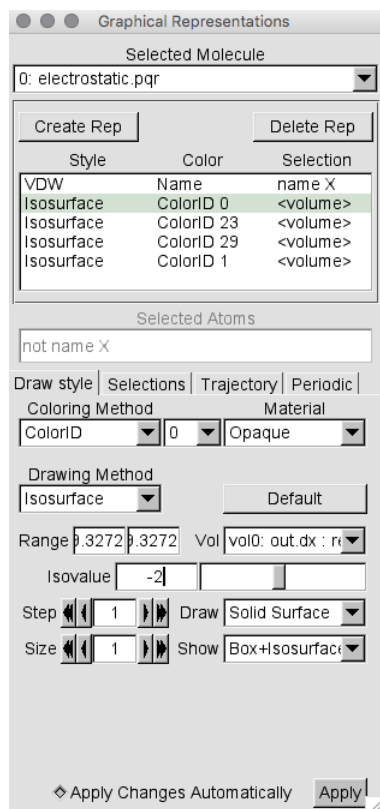


Figure 6.3: DX Graphics

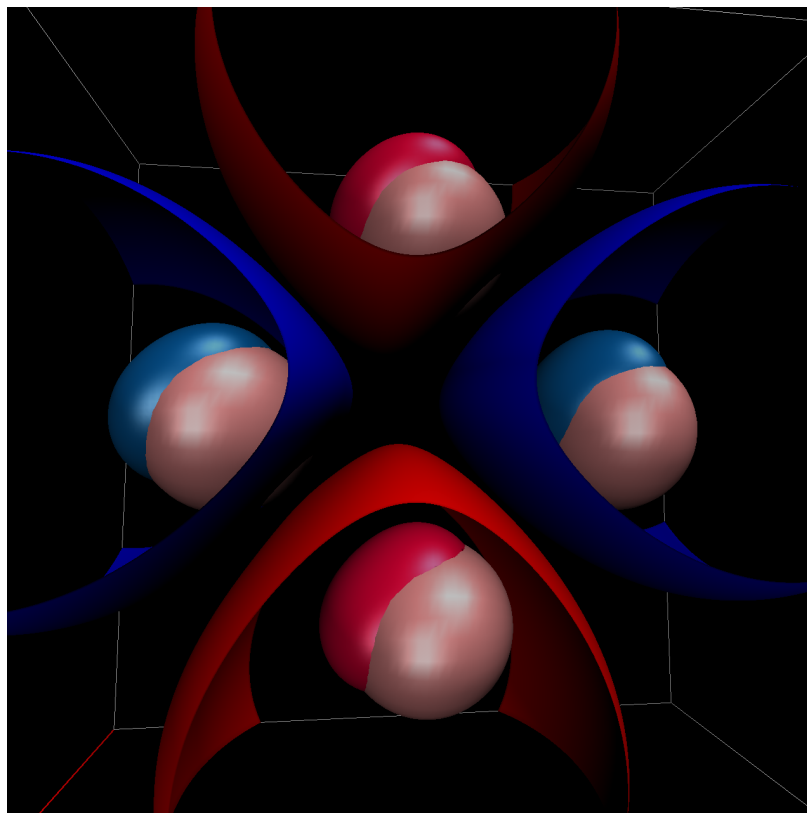


Figure 6.4: DX representation

6.3.2 2D ESP plots

From the electrostatic option in PB-AM, a file is created with the following format:

```

1 # Data from PBAM Electrostat run
2 # My runname is barnase.x.0.dat
3 units kT
4 grid 200 200
5 axis x 0
6 origin -51.2204 -51.2204
7 delta 0.512204 0.512204
8 maxmin 1.35396 0
9      0.2352107      0.2360552      0.2368904      0.2377159

```

In our `tools` directory we have provided a python script for plotting this potential. Change filename within `.py` file, and the desired output name.

```

1 >>> python potential_plot

```

creates a JPG file of the cross sectioned potential, like the one given below:

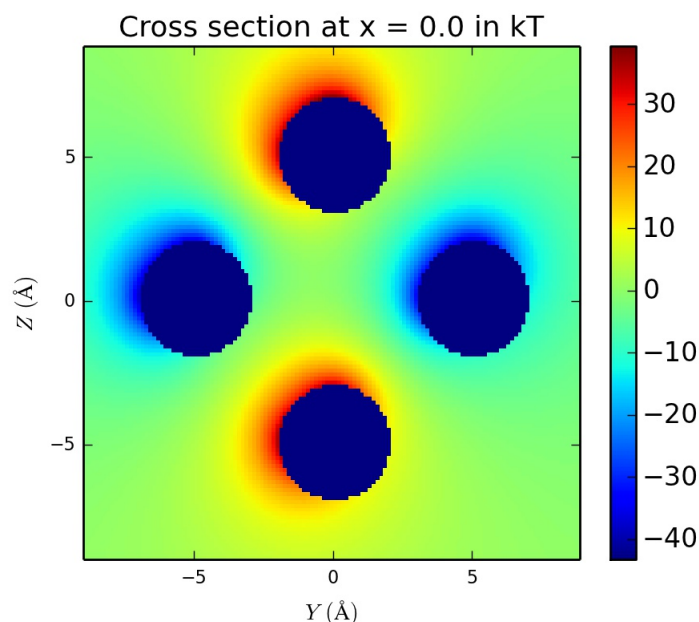


Figure 6.5: Potential plot

6.3.3 3D ESP plots

From the electrostatic option in PB-AM, a file is created with the following format:

```

1 # Data from PBAM Electrostat run
2 # My runname is electro-map.out and units kT
3 grid 10 10 10
4 origin -4 -9 -9
5 delta 0.8 1.8 1.8
6      0.00000      0.00000      -2.90000      -5.899581

```

In our `tools` directory we have provided a python script for plotting this potential. Change filename within `.py` file, and the desired output name.

```
1 >> python plot_3d_surf
```

creates many JPG files of the cross sectioned potential, like the one given below:

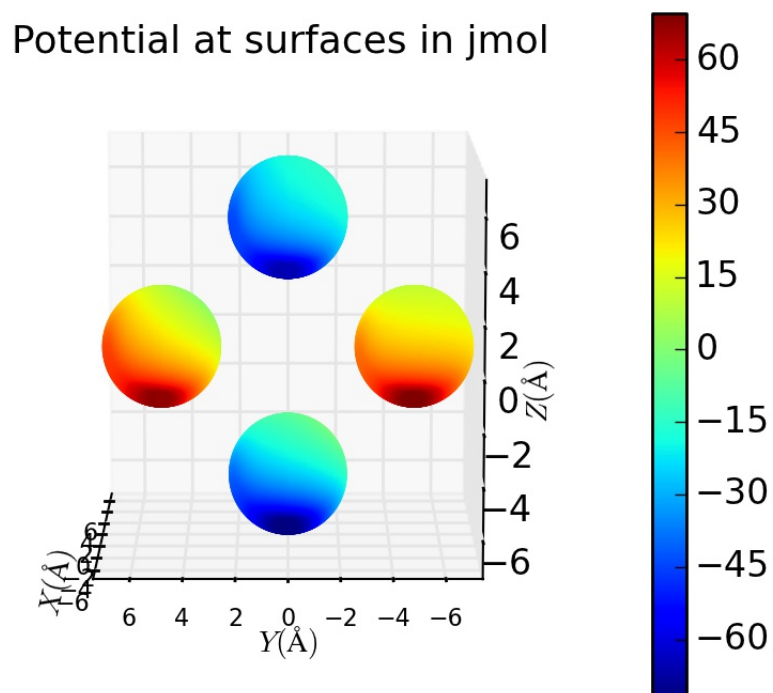


Figure 6.6: Molecule surface potential plot

Chapter 7

Common Errors

7.1 Errors while reading input file

Our file reader catches some errors, like files not existing or being unable to open them. Look at the verbose stream of the program to make sure all your flags are being caught, and check the stream for errors as well.

7.2 Segmentation Fault while reading input file

Our file reader is not very robust. Make sure there is only a single space between keywords and options. Make sure that all options are specified for a given keyword.

7.3 Initial configuration errors

Many issues may arise if your molecules are overlapping, which is not allowed in the PB solvers' models. The center of the molecule will be placed at the positions given by each XYZ file. Check the printed out PQR file for the initial configuration, which you can load into VMD. Try changing the xyz file(s). This error may also appear if the box length specified with the pbc keyword is too small. Try increasing the box length.