Constructing APBS Grids for Molecules

Gary A. Huber

December 23, 2020

1 Introduction

One of the main inputs to a Browndye simulation is the electrostatic grid, usually from APBS, or a nested collection of such grids, around a molecule. These grids can require large amounts of memory, and when more accuracy is desired, the grid must have a finer resolution, and thus require more memory. It is also infeasible for the grid to be very large, so some sort of outer boundary must be determined, and a more simple model for the electric field should be used beyond the outermost grid. Until now, the specification of grids for a Browndye simulation has not been an exact science, although Browndye's simulation programs will warn if the outermost grid is too small. Use of nested grids can give a large advantage, because at larger distances from the molecular surface, a more coarse grid can still give acceptably accurate values of the electric field. However, there has been no clear method for specifying nested grids that give acceptable accuracy while minimizing amount of memory needed. Below we present some theory and the latest addition to the Browndye software to achieve that end.

2 Theory

Computation of the force in Browndye at a point charge is performed by finding the grid parallelpiped containing the point, estimating the electric field at of the eight grid points by finite differences, and using trilinear interpolation to estimate the field at the point charge. In the following analysis, we use the electric potential value rather than the field, but the results should imply acceptable values for the field as well as for the potential.

We construct a point charge with a Debye-Hückel screened potential,

$$V(r) = \frac{\exp(-\frac{r}{L})}{r} \tag{1}$$

where L is the Debye length, and construct cubes with various sizes and at various distances and orientations with respect to the charge. Inside each cube, we construct several points and compute the actual value of the potential at each point, compare it to a trilinear interpolation from the eight corners, and record the greatest difference. Given a threshold of error, we can find the largest allowed cube size as a function of distance of cube center from the point charge. We can use such a function to make sure that any coarser nesting grids are made up of cubes that are far enough from the molecular surface.

The actual computer experiment was performed by picking a distance from the point charge, constructing an icosohedron at that radius, refining the triangles by two levels, and construcing a cube of a given width centered at each radial point, with edges parallel to the coordinate axes. Within the cube, an $11 \times 11 \times 11$ array of points was constructed, including the eight corner points. The difference between actual and interpolated potential was measured and the maximum recorded. For each radius and cube width, the largest deviation from among all cubes was recorded, thus giving a function of error as a function of cube distance and width.

We then determined, at a given radius, which cube width would give a maximum relative error of 0.01. The cube width was varied using the bisection algorithm while evaluating the function described above. The procedure gave a function of minimal acceptable cube width as a function of distance from the charge.

It makes sense that the error should be closely correlated with the second derivative of the radial potential, because most of the error arising from a linear interpolation would come from the second-order term. Thus, the relative error ϵ would be described by

$$\epsilon \propto \frac{V"(r)}{V(r)} w^2 \tag{2}$$

where w is the box width, r the distance. From this, the following proportionality would hold:

$$w \propto \sqrt{\frac{\epsilon V"(r)}{V(r)}} \tag{3}$$

It turns out that the following, using the Debye-Hückel potential in Eq. 2, fits the results very well:

$$w/L = 2.84\sqrt{\frac{\epsilon}{2\frac{L}{r} + 2\left(\frac{L}{r}\right)^2 + 1}} \tag{4}$$

For large Debye length, this reduces to

$$w = 2.84r\sqrt{\frac{\epsilon}{2}} \tag{5}$$

3 Use for Nested Grids

Although the above formulas are derived for a relative error, we decided to use an absolute error threshold of $0.01k_BT$, since the Boltzmann factor is the key energy scale for Brownian dynamics. For now, the described software generates APBS file specifications for grids with cubic spacing. We also assume that, outside of all grids, the electric potential is radially symmetric and follows Eq. 1 for a point charge at the hydrodynamic mobility center of the molecule. Finally, the spacing for the innermost grid, which contains the whole molecule, is specified; this is usually on the order of $1\mathring{A}$.

The grid shapes are determined from the smallest box, aligned with the coordinate axes, that contains the atoms. All grids are then constructed by adding a constant padding in each direction to the faces of that smallest box. The charged atoms are read in from the PQR XML file, and all potential calculations are generated by using the multipole method (same code used to generate the Born desolvation grids) and assuming that each atom is a Debye-Huckel point source. (Of course, this neglects the boundary polarization in the actual field, but this approximation should be sufficient for purposes of determining the grids.)

The outer bounds are determined by finding the smallest grid for which the largest deviation of the single-charge approximation from the full potential calculation on the grid surface is $0.01k_BT$. A square grid of points, with the specified minimum grid spacing, is generated on each of the six faces, and the potential deviation is computed at each point. The padding is found by the bisection algorithm. This is not the exact shape of the outer grid, because APBS has certain constraints on numbers of grid points in each direction. However, it is an *inner bound* on the outermost grid.

The inner grid is determined by starting from the smallest possible box and adjusting the padding until the allowed grid width at the faces is equal to the specified inner grid spacing. On each face, a square grid of points is generated as described for the outer bounds, and the potential is computed on each point. The ratio of $0.01 k_B T$ to the maximum absolute potential value is used as the relative error ϵ in Eq. 4. The padding, which is used as r in Eq. 4, is adjusted using the bisection algorithm.

However, this does not finally determine the shape of the inner grid. APBS works best with the number of grid points in each direction having the form

$$n_g = c2^n + 1 \tag{6}$$

where n and c are positive integers. For this application, c ranges from 1 to 6 and n ranges from 1 to 11. So, given the inner grid bounds and grid spacing, constraints are imposed by Eq. 6 and the requirement that the grid must contain the initial bounds. The possible combinations of grid points in each direction are searched in order to find the smallest grid that satisfies the constraints. This same procedure is used to find the dimensions of the other grids as well.

The bounds of the next grid are computed by using the same procedure, varying the padding until the allowed grid spacing is equal to twice the spacing of the previous grid. Then, the grid dimensions are computed as above. Grids are generated, each with grid spacing twice the previous grid, until a grid is generated that completely contains the outermost bounds computed at the outset.

Occasionally, it will happen that the next outermost grid is not much larger than its predecessor, so little advantage is gained by having two grids rather than one grid of finer resolution. The memory penalty (assumed proportional to the number of grid points) by having one grids is compared to threshold 1.5

$$\frac{M_{f1}}{M_{c1} + M_{f0}} < 1.5 \tag{7}$$

where M_{f1}, M_{c1}, and M_{f0} represent the memory taken up by the outer grid with the finer spacing, the outer grid with the coarser spacing, and the inner grid with the finer spacing, respectivly. If the penalty is less than the threshold, the inner grid is simply expanded to the size of the outer grid but with the finer spacing. The next grid after that will then have spacing that is fourfold greater than its predecessor.

4 Documentation for the Code

4.1 make_apbs_inputs

The program *make_apbs_inputs*, available in Browndye, takes as input the usual input file to *bd_top*, and sends to standard output a new input file with references to new APBS files to be generated. This output (new input file) can then be used as input to *bd_top*. At the same time, it also creates one or more APBS input files, which are named according to the the names of their respective molecular cores. The input to *make_apbs_inputs* might have additional elements that are understood by *make_apbs_inputs* but not *bd_top*; these are used to generate the APBS input files.

First, the *solvent* tag might have additional information on the ions, and on the radius used to define the solvent-accessible surface:

```
<solvent>
  <ions>
    <ion>
      <radius> real </radius>
      <charge> real </charge>
      <conc> real </conc>
    </ion>
    </ions>
    <solvent_radius> real </solvent_radius>
  </solvent>
```

The ion information mirrors the information in an APBS input file, and is used to generate the Debye length, so when the ions are explicitly included, the *debye_length* tag should be left out. The temperature information in the kT tag and the solvent dielectric are also used. The solvent radius gets passed to APBS, and also affects the size of the inner-most grid.

The *core* tag may have additional information:

```
<core>
  <grid_spacing> real </grid_spacing>
  <no_field> true/false </no_field> # optional
</core>
```

If the *electric_field* tag is already present, the core is skipped and no APBS files are generated. If no grids are to be generated, and there is no *elec-*

tric_field tag, than the *no_field* tag is included with *false* as its value. Otherwise, the innermost grid spacing is given in Ångstroms.

When this file is read in, the information used as above to generate the APBS input files, and the corresponding APBS output files are included in the output of *make_apbs_inputs*. The grid files are generated by appending indices, starting with zero at the innnermost grid, onto the core name.

4.2 run_apbs_inputs

In order to run APBS on the generated input files, the program *run_apbs_inputs* can be run, using the output of *make_apbs_inputs* as its input. It runs APBS in the proper order, since the inner grids depend on the outer grids for their boundary conditions. The outermost grid for each core uses the single Debye-Hückel (*sdh*) boundary condition.

5 Conclusion

The grids generated by these programs have been tested informally on a few examples by comparing to full-sized grids, but a more systematic test is needed to make this publishable.