BIOINFORMATICS APPLICATIONS NOTE
Vol. 28 no. 12 2012, pages 1655–1657
doi:10.1093/bioinformatics/bts200

Structural bioinformatics
Advance Access publication April 23, 2012

DelPhi web server v2: incorporating atomic-style geometrical figures into the computational protocol

Nicholas Smith[1,*], Shawn Witham[1], Subhra Sarkar[1,2], Jie Zhang[1,2], Lin Li[1], Chuan Li[1] and Emil Alexov[1,*]
[1]Computational Biophysics and Bioinformatics, Department of Physics and [2]Department of Computer Science, Clemson University, Clemson, SC 29634, USA
Associate Editor: Alfonso Valencia

ABSTRACT
Summary: A new edition of the DelPhi web server, DelPhi web server v2, is released to include atomic presentation of geometrical figures. These geometrical objects can be used to model nano-size objects together with real biological macromolecules. The position and size of the object can be manipulated by the user in real time until desired results are achieved. The server fixes structural defects, adds hydrogen atoms and calculates electrostatic energies and the corresponding electrostatic potential and ionic distributions.
Availability and implementation: The web server follows a client–server architecture built on PHP and HTML and utilizes DelPhi software. The computation is carried out on supercomputer cluster and results are given back to the user via http protocol, including the ability to visualize the structure and corresponding electrostatic potential via Jmol implementation. The DelPhi web server is available from http://compbio.clemson.edu/delphi_webserver.
Contact: nsmith@clemson.edu, ealexov@clemson.edu
Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on January 26, 2012; revised on March 16, 2012; accepted on April 16, 2012

1 INTRODUCTION

Electrostatics play a profound role in molecular biology due to biological macromolecules being made of thousands/millions of atoms, with different size and partial charge situated at short distances. The importance of electrostatic interactions and energies is illustrated by the fact that many biological phenomena are predominantly electrostatic in origin as is salt-dependence of binding (Talley *et al*., 2008) and folding (Tan and Chen, 2011), pH-dependence (Alexov, 2004) and pKa shifts in proteins (Alexov *et al*., 2011) and RNAs (Tang *et al*., 2007).

The main obstacle in modeling electrostatics in biological systems is the presence of water (Baker and McCammon, 2003). Here we utilize a particular resource, DelPhi (Rocchia *et al*., 2002), version 5, which is based on the continuum approach and solves the Poisson–Boltzmann equation (PBE) via the Finite-Difference algorithm. We developed a DelPhi web server which is aimed to provide easy access to performing electrostatic calculations in biological systems without prior knowledge and without having a computational infrastructure in place (Sarkar *et al*., 2012). In addition, with the progress of nano-technology, researchers frequently aim at modeling hybrid systems comprised of nano-objects and biological macromolecules as proteins deposited on a dielectric plate, membranes probed with the tip of an atomic force microscope, interactions of amino acids with spherical nano-particles and many others. However, currently there is no available software and web server capable of modeling such hybrid systems [a web server utilizing APBS to solve PBE in systems comprising of protein and water was recently reported (Unni *et al*., 2011)]. To address this need, we have developed the next generation of DelPhi web server v2, which allows the users to create nano-objects, change their position and shape in real time, and carry out electrostatic calculations in the presence of biological macromolecules.

*To whom correspondence should be addressed.

2 METHODS

2.1 Overall architecture of the server

The design of the server and corresponding modules are described in detail elsewhere (Sarkar *et al*., 2012) and described in Supplementary Material. The main work-flow of the server can be broadly classified into two parts based on its visibility to the end users, namely (i) the client facing server, and (ii) the high performance computational server. Below we describe recent new development to create atomic-style geometrical objects.

2.2 Generation of atomic-style geometrical shapes

Algorithms for generating an atomic-style presentation of four distinctive geometrical shapes (parallelepiped, sphere, cone and cylinder) were designed and are described below. The variables required to construct each shape are circled in black in the appropriate portion of Figure 1.

2.2.1 Parallelepiped The initial dimensions of the figure are built upon user's input which provides the coordinates on the reference corner and three adjacent vertices, as seen in Figure 1a. It is necessary for the figure to be filled with pseudo-atoms such that the size and distance between the pseudo-atoms be a constant scalar parameter, which can be varied to achieve the desired degree of resolution. As this would limit the number of points being constructed on the path of the line, it was found that a seam would form at the ends of the vectors and result in non-uniform objects. To correct this, the maximum number of points k_v allowed on the line $\vec{v}$ is calculated using integer division on the given resolution p. This is translated into a new spacing between the points, represented by p_v, that evenly distributed the points along the line by the following equations:

$$k_v = |\vec{v}|/p \quad p_v = |\vec{v}|/k_v \tag{1}$$

Each vector of the box is then normalized and multiplied by the new spacing to create an incremental vector. This is then used to create an array of points along the path of the original vector, and is also done for the original three vectors of the box. A different incremental vector is added to a copy of the array of points to extend the points over a single side of the box. This is repeated for each side of the box with several if-statements to eliminate duplicate points on the edges and corners. Once the arrays that define the shape are complete, a method for printing the points out in the proper PDB format was designed as can be seen in Figure 1a.

2.2.2 Sphere The user inputs the coordinates of the center of the sphere and its radius as in Figure 1b, which are then used for the pseudo-atomic filling of the resulting sphere. By translating to arc-length, the calculations that involve creating cyclic polygons to fit a maximum number of points inside a circle in the sphere were relaxed. The final process generated a sphere using a completely spherical coordinate system. First, the change in theta, $\Delta\theta$, is calculated by the following formula:

$$k_r = \pi r/p \quad \Delta\theta = \pi r/kr \tag{2}$$

where k_r is the number of points allowed on the circle by the precision p and the radius of the circle r. Then, starting with the top of the sphere and by incrementing θ, a circle is generated for each value of θ between zero and π which then uses the above equation to generate an incremental value for ϕ using 2πr instead of πr. These incremental values are then looped over to generate an array of points that extended over the surface of the sphere. This process results in a uniform distribution of points across the surface of the shape with no singularities.

2.2.3 Cylinder Generally, the cylinder is constructed by generating circles along a vector for the body. A quaternion rotation algorithm handles rotation of desired shapes/sizes. The cylinder is constructed first by calculating the direction vector that pointed from the first origin to the second. This is then translated into spherical coordinates and copied; this clone is incremented by $\pi/2$ in the θ direction and its radius is set to the given radius of the cylinder. By cross multiplying with the original vector, a third normalized vector is created, and its radius set to the given radius. These latter two vectors form the plane normal to the direction vector and thereby normal to the cylinder itself, and the third vector is later used as the axis of rotation for the quaternion. The rotational quaternion q is formed using the following formula:

$$\theta = \hat{z}\cdot\widehat{dir} \quad q = \cos\theta/2 + \vec{v}\sin\pi - \theta/2 \tag{3}$$

where $\hat{z}$ is the unit vector in the z direction, $\widehat{dir}$ is the unit vector pointing toward the direction vector, $\vec{v}$ is the axis of rotation (the third aforementioned vector), and θ is the angle between the z-axis and the direction vector. This quaternion rotates a given vector about the axis of rotation by θ. The rotation equation for rotating a vector $\vec{v}$ is:

$$\vec{v}' = q\vec{v}q^* \tag{4}$$

where q^* is the complex conjugate of the quaternion q and $\overrightarrow{v'}$ is the new rotated vector from the original vector. The cylinder is then built by generating circles of points centered along z-axis with spacing equal to the given resolution of the object between them. The top and bottom of the cylinder are generated by creating circles of varying radii. The radius for each new circle is calculated by transforming a vector that lay in the x–y plane with a magnitude of the given radius into an incremental vector with a length equal to the given precision and then looping over this vector to create an ever-increasing radius up to the edge of the outer cylindrical ring. Once the set of circles are constructed, they are added to by the direction vector to create the top surface of the cylinder. Each of these points are then inserted into the above equation to find the new point in the cylinder's actual direction and written out to the resulting PDB file as rendered in Figure 1c.

2.2.4 Cone The user specifies the coordinates of the two origins and the opening angle in degrees which is exemplified in Figure 1d. The radius r of the cone is calculated using the following formula:

$$r = \left|\overrightarrow{dir}\right| \tan\theta \tag{5}$$

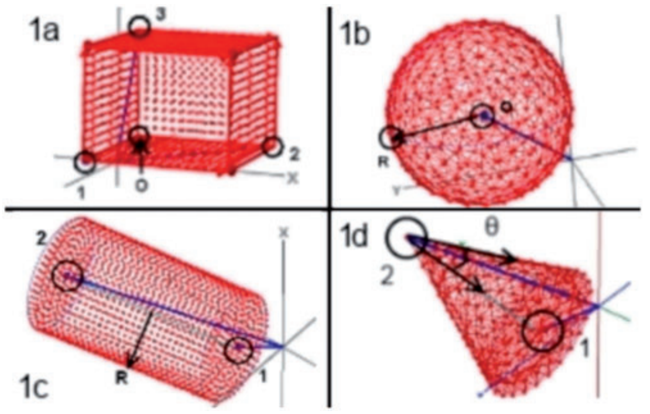

Fig. 1. Objects generated by the program visualized through Jmol (**a**) Parallelepiped, (**b**) Sphere, (**c**) Cylinder and (**d**) Cone

where $\widehat{dir}$ is the direction vector and θ is the opening angle. The direction and right angle vectors are calculated using the same steps as the cylinder as well as the rotational quaternion and its associated parameters. The difference lay in the construction of the outer rings of the cone. The radii of these rings are found by multiplying the sine of the opening angle by a new precision which is then subtracted incrementally from the radius of the bottom level. The new precision is determined by using the outer edge of the cone to redistribute the allowed number of points, similar to previous objects. At each level, the circle of points is formed in the x–y plane using the radius and a z-axis increment of cosine of the opening angle multiplied by the new precision, and rotated using the quaternion methods developed for the cylinder (Fig. 1d).

3 RESULTS

The above algorithms were implemented using C++ code for flexibility and speed and were incorporated into the DelPhi web server (see Supplementary Material for more details). The geometrical object can be generated by selecting 'Yes' for the option 'Do you wish to create objects?' at the 'File upload and input parameter form for DelPhi run' page. Once this option is selected, the user is given the choice to create four different types of geometrical figures: sphere, cylinder, cone and parallelepiped. Upon choosing a particular geometrical figure, a new window appears showing the geometrical shape positioned in a Cartesian reference frame. The same figure provides a set of indicators detailing what the input parameters are for this particular shape. The user should then enter the required parameters that define the position and dimensions of the appropriate geometrical shape. In addition, the user can also enter the desired dielectric constant for the object, which can vary from the protein's dielectric constant. This allows the modeling of various systems including biological macromolecules and nano-objects that have variegated dielectric properties.

The positioning of the geometrical object from the manually entered parameters may not correspond to the desired location of the object due to the offset of the origin in crystallographic structures which is not at (0,0,0). To allow the users to further refine the position, dimensions and shape of the corresponding geometrical object in the DelPhi web server, the object manipulation for the combined protein-object PDB file has been enhanced with real-time manipulation of the object using JavaScript sliders coupled with the Jmol applet. Such an implementation is very useful because it shows a skeleton of the object's new positioning and dimensions

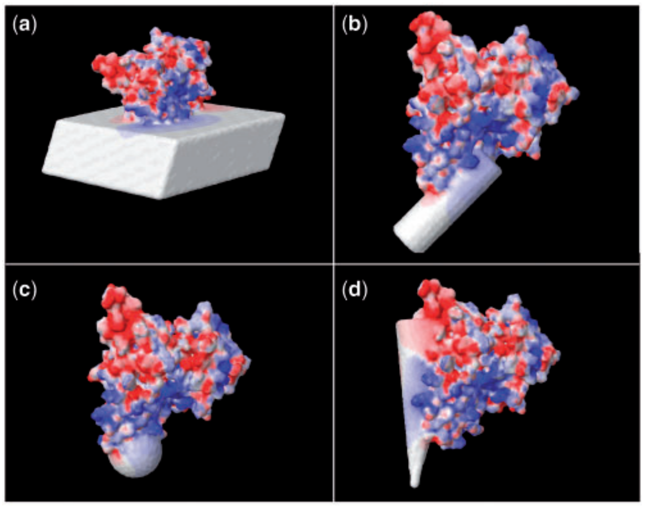

Fig. 2. Output screen of the Delphi web server showing a protein and the corresponding geometrical object with the potential mapped onto the combined surface protein-object

and thus helps users to orient the object exactly. This object can then be generated and re-rendered to show the exact location and size of the object as a final confirmation for the user before the job is processed with DelPhi. The adjustment of the object's position, dimensions and shape can be repeated as many times as desired until the appropriate orientation is found.

The next page of the web server offers various output options including which components of the electrostatic energy need to be calculated and what output files are to be generated as potential or dielectric maps, or site potential files. Then, the input files and selected options are transferred to Clemson's supercomputer cluster, the Palmetto high-performance computing facility (http://www.clemson.edu/ccit/rsch_computing/). Once the jobs are completed, the user receives email notification that the job is finished along with ID number and a link. By clicking on the link, the user is directed to the DelPhi web server download and visualization page. On this page, the user is presented with several ways to download the results, the protonated and fixed PDB files, and the parameter files used in this particular job. In addition, the page utilizes Jmol. If the user had requested a potential map to be calculated, then the potential map in 'CUBE' format is also available for download and for visualization. By clicking on the potential map file, the page then maps the potential onto the molecular surface of the corresponding biological macromolecule. The potential levels and the color scheme can be manipulated by the user for visualization purposes. By using the Jmol applet, the 3D structure mapped with the electrostatic potential can be rotated and zoomed in if necessary.

Figure 2 demonstrates the output capabilities of Delphi web server. It shows a particular protein, the CLIC2 protein (Mi *et al.*, 2008), being deposited on or probed with the four geometrical objects described in this article. It can be seen that the objects' surface is quite smooth indicating the atomic-style representation provides a very good approximation of solid objects. The protein was modeled with a dielectric constant of 4 and the objects with a dielectric constant of 20. It can also be seen that in some cases the object penetrates inside the protein on purpose. Such a scenario can be used to model a membrane protein inserted into membrane slab (Fig. 2a). Figure 2b illustrates the case of a protein touching a cylindrical rod, which can be a fiber wire or another nano-object. Figure 2c shows a protein deposited on a spherical nano-particle. Lastly, Figure 2d demonstrates a cone-protein complex, which can be viewed as the tip of atomic force microscope that has already passed through the protein.

In terms of the calculated energies, the atomic-style representation of the objects was tested against the original implementation of the geometrical shapes in DelPhi (Rocchia *et al.*, 2002), and it was found that the results in terms of the solvation energy are consistent within 0.1%.

Funding: NIGMS award number 1R01GM093937.

Conflict of Interest: none declared.

REFERENCES

Alexov,E. (2004) Numerical calculations of the pH of maximal protein stability. The effect of the sequence composition and three-dimensional structure. *Eur. J. Biochem.*, **271**, 173–185.

Alexov,E. *et al.* (2011) Progress in the prediction of pKa values in proteins. *Proteins*, **79**, 3260–3275.

Baker,N.A. and McCammon,J.A. (2003) Electrostatic interactions. *Meth. Biochem. Anal.*, **44**, 427–440.

Mi,W. *et al.* (2008) The crystal structure of human chloride intracellular channel protein 2: a disulfide bond with functional implications. *Proteins*, **71**, 509–513.

Rocchia,W. *et al.* (2002) Rapid grid-based construction of the molecular surface and the use of induced surface charge to calculate reaction field energies: applications to the molecular systems and geometric objects. *J. Comput. Chem.*, **23**, 128–137.

Sarkar,S. *et al.* (2012) DelPhi web server: a comprehensive online suite for electostatic calculations of biological macromolecules and their complexes. *Commun. Comput. Phys.*, (in press).

Talley,K. *et al.* (2008) On the electrostatic component of protein-protein binding free energy. *PMC Biophys.*, **1**, 2.

Tan,Z.J. and Chen,S.J. (2011) Salt contribution to RNA tertiary structure folding stability. *Biophys. J.*, **101**, 176–187.

Tang,C.L. *et al.* (2007) Calculation of pKas in RNA: on the structural origins and functional roles of protonated nucleotides. *J. Mol. Biol.*, **366**, 1475–1496.

Unni,S. *et al.* (2011) Web servers and services for electrostatics calculations with APBS and PDB2PQR. *J. Comput. Chem.*, **32**, 1488–1491.