Protein-Nano Object Integrator (ProNOI)

Supported by NIGMS, NIH, grant number: R01 GM093937

Lead scientist: Mr. Nicholas Smith

Benchmarking: Mr. Brandon Campbell

Background: With the progress of nanotechnology, one frequently has to model biological macromolecules simultaneously with nano-objects. However, the atomic structures of the nano objects are typically not available or they are solid state entities. Because of that, the researchers have to investigate such nano systems by generating models of the nano objects in a manner that the existing software be able to carry the simulations. In addition, it should allow generating composite objects with complex shape by combining basic geometrical figures and embedding biological macromolecules within the system.

Results: Here we report the *Protein Nano-Object Integrator (ProNOI)* which allows for generating atomic-style geometrical objects with user desired shape and dimensions. Unlimited number of objects can be created and combined with biological macromolecules in Protein Data Bank (PDB) format file. Once the objects are generated, the users can use sliders to manipulate their shape, dimension and absolute position. In addition, the software offers the option to charge the objects with either specified surface or volumetric charge density and to model them with user-desired dielectric constants. According to the user preference, the biological macromolecule atoms can be assigned charges and radii according to four different force fields: Amber, Charmm, OPLS and PARSE. The biological macromolecules and the atomic-style objects are exported as a position, charge and radius (PQR) file, or if a default dielectric constant distribution is not selected, it is exported as a position, charge, radius and epsilon (PQRE) file. As illustration of the capabilities of the *ProNOI*, we created a composite object in a shape of a robot, aptly named the Clemson Robot, whose parts are charged with various volumetric charge densities and holds the barnase-barstar protein complex in its hand.

Conclusions: The *Protein Nano-Object Integrator (ProNOI)* is a convenient tool for generating atomic-style nano shapes in conjunction with biological macromolecule(s). Charges and radii on the macromolecule atoms and the atoms in the shapes are assigned according to the user's preferences allowing various scenarios of modeling. The default output file is in PQR (PQRE) format which is readable by almost any software available in biophysical field. It can be downloaded from: http://compbio.clemson.edu/downloadDir/ProNO_integrator.tar.gz

Implementation

The main body of the GUI was designed using an interface coded in Java which communicates with a C++ command line program in the background to generate the atomic-style objects. The program uses the Java Swing libraries for the visual interface design and encapsulates the BioJava implementation of the Jmol molecular viewer in order to provide the user with a clear visual representation of their protein(s) and associated nano-objects.

Once the program boots, it allows the user to either insert objects into an entirely empty file or open up their own PDB/PQR file for editing. If the user loads their own file, an intelligent file-parser will chop up their file into the appropriate metafiles consisting of the objects detected in the file via the tagged REMARK 400 headers and the main body of the protein. These files are contained within the user's HOME directory inside an appropriately named hidden folder and are cleaned up upon the program's exit to conserve the space on the system. The list of parsed objects is then used to populate the associated code objects and GUI tables, complete with each of the parameters used to generate the objects. Once this initial preprocessing is done, the user is then able to manipulate each of the objects individually by either changing the size, shape, or positioning of the object in the space or by changing the atomic properties of the object such as the atomic radius, dielectric constant, atomic identifiers, or object names. The user can also add or delete individual objects and track which objects have been modified since the last compilation of the file by the color-coding of each of the object names in the list: blue for modified objects, gray for unmodified objects.

A key feature of the *ProNOI* program is the linking of the GUI controls to the molecular viewer in order to provide the user with immediate feedback. The sliders for each of the objects are linked to dynamically generated Jmol commands which construct a skeleton of the object's expected location for the regeneration. So, even while the user is moving the sliders for the object, the object's new position can be tracked in real-time.

Once the user's adjustments have been made, the user can regenerate the PDB/PQR file and see exactly how the modeling configuration has changed. This operation is completed in the background by a call to the C++ object manipulation tool, which, if the appropriate executable is not found, will offer the user a helpful file navigation dialog to let them specify exactly where the program is located. The Java GUI will then process all of the parameters from each of the objects, sanitizing and validating each parameter in order to avoid harmful scripts executing on the command line, and then calling the C++ program once for each modified object. The output from the C++ program results in a single PDB/PQR file for each object which has been prefixed with a REMARK 400 header and contained in the hidden directory. These files are then combined with the original data from the PDB file and form a new compiled file in the hidden directory and loaded into the molecular viewer. These actions also preserve the user's current perspective in the protein space which can be very useful for monitoring small changes to the objects.

The C++ object manipulation tool has several additional features worth mentioning. Atomic radii and charges can now be appended to each atom if the user selects the PQR file format for the output. The atomic radii of the object are simply entered into the GUI and passed through but the charges per atom are calculated via a density argument. The C++ program allows surface and volume density parameters to be passed into it in units of electron charge per Angstrom squared for surface charge density or per Angstrom cubed for volumetric charge density options. The charge per atom is then calculated by the following formulae:

$$q_V = d(\tfrac{q}{\text{Å}^3}) \cdot V \qquad q_A = d(\tfrac{q}{\text{Å}^2}) \cdot A \tag{1a}$$

$$V_{sphere} = \tfrac{4}{3}\pi\, r^3 \qquad A_{sphere} = 4\pi\, r^2 \tag{1b}$$

$$V_{Cylinder} = \pi\, r^2 \left|\overrightarrow{dir}\right| \qquad A_{Cylinder} = 2\pi\, r^2 + 2\pi\, r \left|\overrightarrow{dir}\right| \tag{1c}$$

$$V_{Cone} = \tfrac{1}{3}\pi\, r^2 h \qquad A_{Cone} = \pi\, r(r + \sqrt{r^2 + \left|\overrightarrow{dir}\right|^2}) \tag{1d}$$

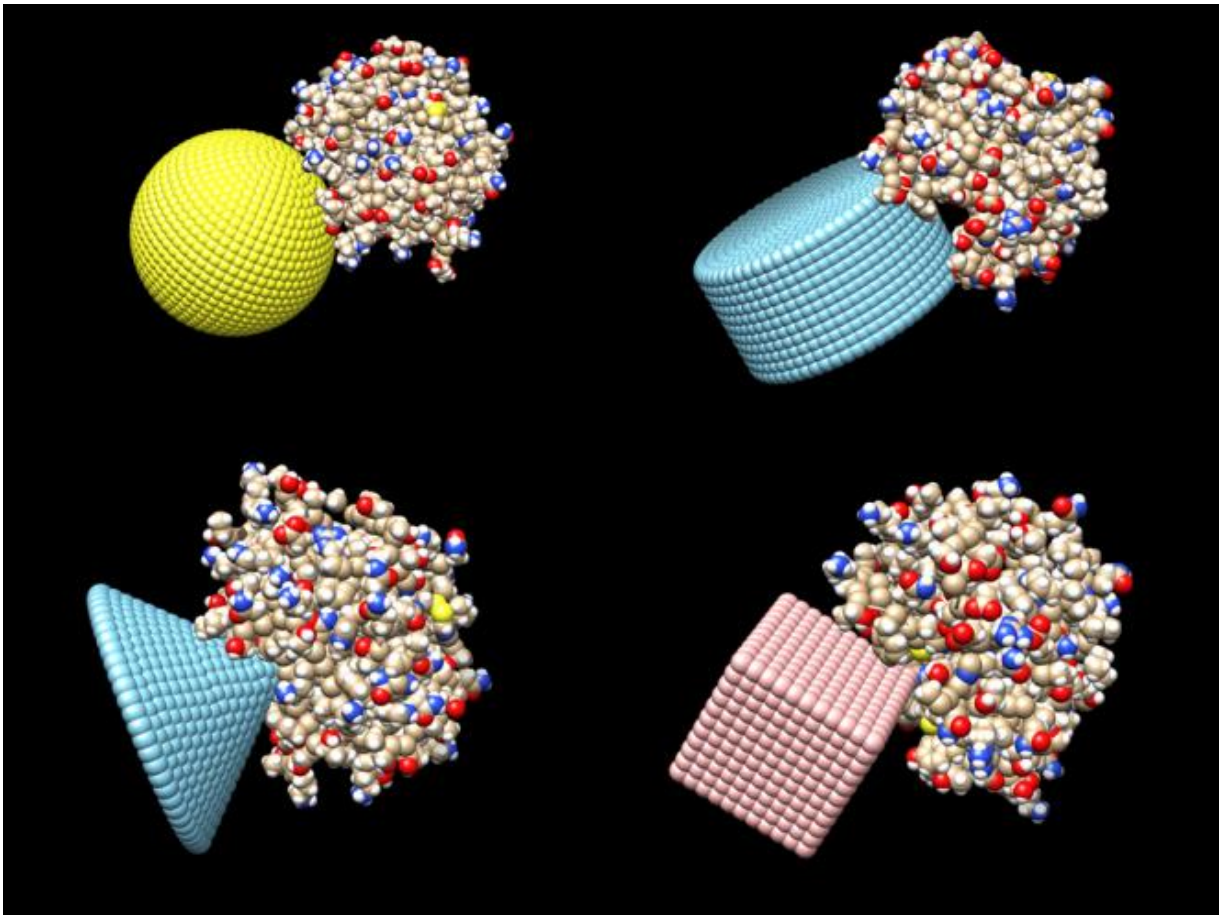
$$V_{Box} = |\vec{a} \times \vec{b} \times \vec{c}| \qquad A_{Box} = 2[\,|\vec{a} \times \vec{b}| + |\overrightarrow{a} \times \vec{c}| + |\overrightarrow{b} \times \vec{c}|] \tag{1e}$$

where q_v and q_A are the charges of each atom, d is the given charge density with its units in parentheses as the charge of the electron per Angstroms squared or cubed, r is the radius of the object from the input, $\overrightarrow{dir}$ is the direction vector also from the input, $\vec{a}$ and $\vec{b}$ and $\vec{c}$ are the input vectors for the box, and V is the volume and A is the area of the specified object. This is then appended to each atom of the object along with the given radius in conformance to the PQR file format.

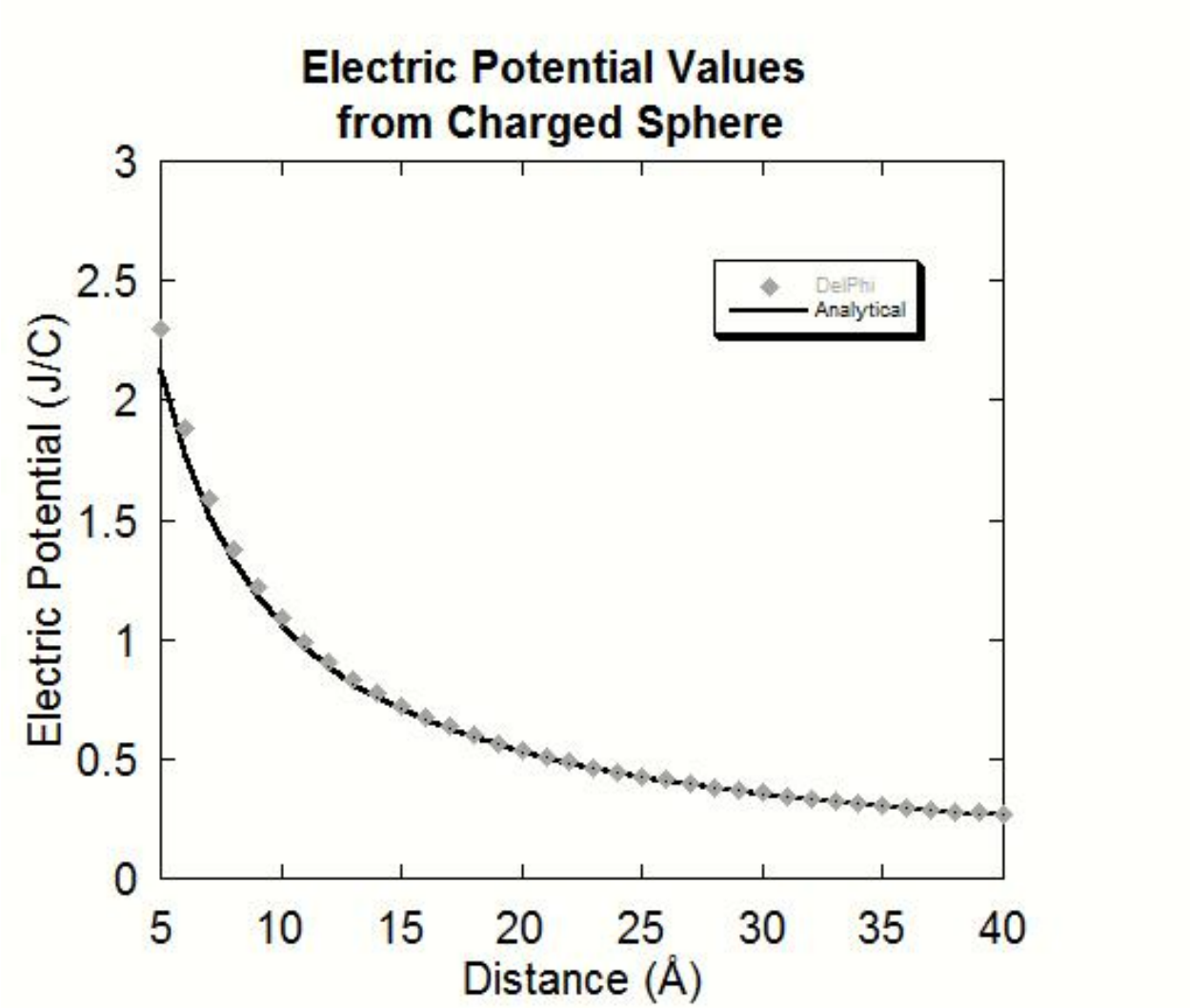
In addition to the object manipulation tools, a force field parameters selector has been added to allow the user to convert their original macromolecule PDB data into PQR format in conformance with a set of parameter files. The current force field parameters used by this program are Amber (v. 98), Charmm (v.22), OPLS and PARSE along with an option in the preferences to upload a properly formatted size (SIZ) and charge (CRG) file-set for a custom force field parameters. The custom force field parameter option is specifically useful for cases involving non-standard compounds, for which the charges and radii must be obtained with other programs, as for example with the antechamber. This selector, upon object generation, scans the macromolecule PDB file for ATOM entries and attempts to find the residue and atom names and then find the corresponding radius and charge for the atom from the data read in from the force field parameter files. If the specific residue name is not found, the program will then try to find the atom name in the global id list and pull the charge and radius from there. If it is still unsuccessful, the program will record the error and set the atom's radius to one and its charge to zero. Once all the entries have been processed and if any errors resulting from missing entries have been recorded, a window will appear displaying each of the missing entries letting the user know which entries were not found and need to be addressed.

Results:

(a) Modeling four standard geometrical nano-objects along with a protein



(b) Modeling electrostatic potential created by a charged sphere



(c) Clemson Robot holding barnase-barstar complex in its hand.