ContinuumElectrostatics

A pDynamo module for protonation state calculations

Mikolaj Feliks

Institut de Biologie Structurale, Grenoble

Last updated: June 20, 2014

1 Introduction

Proteins contain residues, cofactors and ligands that bind or release protons depending on the current pH and the interactions with their molecular environment. These titratable residues, cofactors and ligands will be referred to as sites. Each site exists in at least two charge forms, usually the protonated form and the deprotonated form. Different charge forms of a site are called instances. The titration of proteins is difficult to study experimentally because the available methods, such as calorimetry, cannot determine protonation states of individual sites. The knowledge of these individual protonation states is crucial for understanding of many important processes, for example enzyme catalysis.

The ContinuumElectrostatics module extends the pDynamo software library by Martin Field[1] with a Poisson-Boltzmann continuum electrostatic model that allows for calculations of protonation states of individual sites. The module links pDynamo to the external solver of the Poisson-Boltzmann equation, MEAD. MEAD is a program developed by Donald Bashford[2] and later extended by Timm Essigke[3] and Thomas Ullmann. The electrostatic energy terms obtained by MEAD can be used to calculate energies of all possible protonation states of the protein of interest. However, analytic evaluation of protonation state energies is only possible for proteins with usually less than 20 titratable sites. The ContinuumElectrostatics module provides an interface to the GMCT[4] program by Matthias Ullmann and Thomas Ullmann that can be used to sample protonation state energies using a Monte Carlo method. The GMCT interface makes it possible to study the titration of larger proteins.

2 Copying

The module is distributed under the CeCILL Free Software License, which is a French equivalent of the GNU General Public License. For details, see the files `Licence_CeCILL_V2-en.txt` (or `Licence_CeCILL_V2-fr.txt` for the French version).

3 Goals of the ContinuumElectrostatics module

I wrote the ContinuumElectrostatics module primarily as a pretext to learn how the Poisson-Boltzmann model works in detail. I used this model very often during my studies on enzyme catalysis but never had time nor pressure to learn the details of the theory that was behind. I also wanted to better explore the pDynamo library, understand its programming concepts and finally extend it with something useful. Last but not least, I saw some room for improvement in the previously used tools and scripts.

The ContinuumElectrostatics module is similar in behavior to the `multiflex2qmpb.pl` program, which is part of the QMPB package written by Timm Essigke. The approach taken here is most compatible with the original approach by Donald Bashford. The key novelty is the improved treatment of multiprotic sites, such as histidine. Also, the calculations with the ContinuumElectrostatics module are considerably faster than previously because of the parallelization.

4 Installation and configuration

Before the installation of the ContinuumElectrostatics module, it is necessary to have:

- pDynamo 1.8.0

- Python 2.7

- GCC (any version should be fine)

- Extended MEAD 2.3.0

- GMCT 1.2.3

Extended MEAD and GMCT can be found on the website of Thomas Ullmann:

http://www.bisb.uni-bayreuth.de/People/ullmannt/index.php?name=software

Download the two packages and follow their respective installation instructions. The ContinuumElectrostatics module requires for its functioning two programs from the MEAD package, namely `my_2diel_solver` and `my_3diel_solver`, and the GMCT's main program, `gmct`.

In the next step, check out the latest source code of the module. Note that for checking out the source code you should have Subversion installed as well.

```
svn checkout
    http://pdynamo-extensions.googlecode.com/svn/trunk/ContinuumElectrostatics/
```

Some parts of the module implementing the state vector are written in C and therefore have to be compiled before use. In the future, I plan to shift some more parts of the module from Python to C.

Start from going to the directory `extensions/csource/`. Change the uppermost line in the `Makefile`. This line defines the directory where you have installed pDynamo. After editing, close the file and run "make" to compile the C object file.

Go to the directory `extensions/pyrex/` and again edit the `Makefile`. Change the line starting from "INC2" to the location of your pDynamo installation. Close the file and run "make". It should generate a dynamically linked library, `StateVector.so`, in the directory `Continuum-Electrostatics/`.

At this point, the installation is complete.

Before using the module, the environment variable `PDYNAMO_CONTINUUMELECTROSTATICS` should be set to the module's root directory. This directory should be also added to the `PYTHONPATH` variable. This can be done in the following way (in Bash):

```
export
    PDYNAMO_CONTINUUMELECTROSTATICS=/home/mikolaj/devel/ContinuumElectrostatics
export PYTHONPATH=$PYTHONPATH:$PDYNAMO_CONTINUUMELECTROSTATICS
```

5   Usage

After the installation, it may be worth looking at some of the test cases. I will explain the functioning of the module based on the test case "sites2". This test uses a trivial polypeptide with only two titratable sites, histidine and glutamate. The test "histidine" uses only one site. The other tests use real-life, although small proteins.

By default, the ContinuumElectrostatics module will not empty the scratch directory after completing the run. Starting the job for the second time will cause reading of the existing scratch files instead of running the calculations anew. To avoid this behavior, remove the scratch directory or declare another one during the setup of the continuum electrostatic model. **Notice! The energies calculated by the ContinuumElectrostatics module are given in kcal/mol, in contrast to kJ/mol, which are normally used in pDynamo.**

5.1   Setup of the protein model

The electrostatic model used by the ContinuumElectrostatics module requires that the protein of interest is described by the CHARMM energy model[5]. In the first step, prepare CHARMM topology (PSF) and coordinate (CRD) files. The preparation can be done using the programs CHARMM[6] or VMD[7]. During the preparation of the protein model, all titratable residues in the protein should be set to their standard protonation states at pH = 7, i.e. aspartates and glutamates deprotonated, histidines doubly protonated, other residues protonated. The topology, coordinate and parameter files are loaded at the beginning of the script `sites2.py`:

```
par_tab = ["charmm/toppar/par_all27_prot_na.inp", ]
mol  = CHARMMPSFFile_ToSystem ("charmm/testpeptide_xplor.psf", isXPLOR = True,
    parameters = CHARMMParameterFiles_ToParameters (par_tab))
mol.coordinates3 = CHARMMCRDFile_ToCoordinates3 ("charmm/testpeptide.crd")
```

5.2 Setup of the continuum electrostatic model

In the second step, a continuum electrostatic model is created:

```
ce_model = MEADModel (meadPath = "/home/mikolaj/local/bin/", gmctPath =
    "/home/mikolaj/local/bin/", scratch = "scratch", nthreads = 2)
```

The parameter "meadPath" tells the directory where the MEAD programs, my_2diel_solver and my_3diel_solver, are located. The parameter "gmctPath" is the location of the gmct program. If none of these directories are given, /usr/bin is assumed by default. The parameter "scratch" tells the directory where the MEAD job files and output files will be written to. If not present, this directory will be created. The last parameter, "nthreads", defines the number of threads to be used. By default nthreads=1, which means serial run. Note that "nthreads" can be any natural number and that the calculations scale linearly with the number of threads. Parallelization is done at the coarse-grain level. Since the electrostatic energy terms for a particular instance of a titratable site can be calculated independently from energy terms of other instances of other sites, each instance is assigned a separate thread. A similar approach is taken during the calculations of titration curves, where each pH-step of a curve is calculated separately.

In the next step, the continuum electrostatic model is initialized:

```
ce_model.Initialize (mol)
```

The initialization means partitioning of the protein into titratable sites and a non-titratable background. It also means generating model compounds. At this point, however, the input files for MEAD are not written and only the necessary data structures inside the MEADModel object are created. The Initialize method takes at least one argument, which indicates the CHARMM-based protein model.

The next two lines generate a summary of the continuum electrostatic model and write a table of titratable sites:

```
ce_model.Summary ()
ce_model.SummarySites ()
```

After the model has been initialized, the input files necessary for calculations in MEAD can be written to the scratch directory:

```
ce_model.WriteJobFiles (mol)
```

By default, each site is assigned a separate directory, for example `scratch/PRTA/GLU8`. Inside the directory, there are PQR files for each instance of the site in the protein and in a model compound. The OGM and MGM files specify parameters of lattices used for solving the Poisson-Boltzmann equation. The `back.pqr` file defines the non-titratable background. The `protein.pqr` file defines the whole protein and is used to calculate the boundary between the protein and the solvent. The last file, `sites.fpt`, contains atomic coordinates and charges of all instances and is used for calculating interaction energies between different instances of sites in the protein.

5.3   Calculating electrostatic energies

At this point, the electrostatic energy terms can be calculated:

```
ce_model.CalculateElectrostaticEnergies ()
```

For each instance of each site, two electrostatic energy terms are calculated, namely the Born energy (G_{Born}) and the background energy (G_{back}). Born energy is the electrostatic energy of a set of charges interacting with its own reaction field. Background energy is the electrostatic energy of a set of charges interacting with charges from outside of this set. The two energies are calculated for a particular instance of a site both in the model compound and in the protein. The difference $(G_{\mathrm{Born,protein}} + G_{\mathrm{back,protein}}) - (G_{\mathrm{Born,model}} + G_{\mathrm{back,model}})$ is calculated, which is called the homogeneous transfer energy, $G_{\mathrm{homotrans}}$. Transferring of a site means moving it from the model compound to the protein. In the model compound, the site has a model energy G_{model}, which corresponds to the experimentally known $\mathrm{p}K_{\mathrm{a}}$ value of the deprotonation reaction. The site in the protein has an intrinsic energy $G_{\mathrm{intr}} = G_{\mathrm{model}} + G_{\mathrm{homotrans}}$. The

`my_2diel_solver` program calculates $G_{\text{Born,model}}$ and $G_{\text{back,model}}$. The `my_3diel_solver` program calculates $G_{\text{Born,protein}}$ and $G_{\text{back,protein}}$ and, additionally, electrostatic interaction energies of an instance of a site with other instances of other sites in the protein. The Continuum-Electrostatics module collects $G_{\text{Born,protein}}$, $G_{\text{back,protein}}$, $G_{\text{Born,model}}$, $G_{\text{back,model}}$ and interaction energies from MEAD and calculates $G_{\text{homotrans}}$ and G_{intr} for each instance of each site.

Note that the `my_3diel_solver` program can in principle perform calculations in a three-dielectric environment (solvent, protein, vacuum). However, the model implemented in the ContinuumElectrostatics module only deals with two-dielectric environments, i.e. the solvent phase and the protein/model compound phase.

5.4 Calculating microstate energies

After the G_{intr} values and interaction energies have been calculated for all instances of all titratable sites, one can calculate the energy of a particular protonation state of the protein, i.e. the microstate energy, G_{micro}. The polypeptide in the "sites2" example contains only two sites, glutamate and histidine, so there can be $2^1 * 4^1 = 8$ possible protonation states, because glutamate has two instances ("p" and "d") and histidine has four instances ("HSP", "HSD", "HSE", "fully deprotonated"). For real-life proteins, the number of protonation states is very large. The protonation state of the protein is defined by a state vector. Each component of the state vector represents the protonation state of a site in the protein. The value of the component indicates the current instance of the site. The values of 0 and 1 do not necessarily mean "deprotonated" and "protonated". The next few lines of code generate all possible permutations of the state vector and calculate their respective microstate energies:

```
statevector = StateVector (ce_model)
increment   = True
while increment :
  Gmicro = ce_model.CalculateMicrostateEnergy (statevector, pH = 7.0)
  statevector.Print (ce_model, title = "Gmicro = %f" % Gmicro)
  increment = statevector.Increment ()
```

In the lowest energy protonation state, the histidine is protonated at positions δ and ϵ and

the glutamate is deprotonated. Dissociation of the ϵ-hydrogen from the histidine requires only 0.7 kcal/mol.

5.5   Calculating protonation probabilities

One option for calculating protonation probabilities is to use GMCT externally after the calculations by MEAD are complete. This can be done by writing two files, `gintr.dat` and `W.dat`, containing the calculated intrinsic energies and interaction energies, respectively.

```
ce_model.WriteGintr ()
ce_model.WriteW ()
```

In the ContinuumElectrostatics module, protonation probabilities can be calculated either analytically for small proteins or by using GMCT for larger proteins. The following two lines calculate protonation probabilities analytically and report them in a table. By default, the calculations are done at pH = 7. The default behavior can be changed by passing the "pH" argument.

```
ce_model.CalculateProbabilitiesAnalytically ()
ce_model.SummaryProbabilities ()
```

For comparison, the probabilities are calculated by using GMCT. During the calculations in GMCT, a `scratch/gmct/` directory will be created for storing input and output files.

```
ce_model.CalculateProbabilitiesGMCT ()
ce_model.SummaryProbabilities ()
```

The resulting tables show for each site the probability of occurrence of each instance. Instances with the highest probability of occurrence are marked with "*". Optionally, the argument `reportOnlyUnusual=True` will only show sites in their non-standard protonation states, for example protonated glutamates.

8

5.6 Calculating titration curves

Titration curves are obtained by calculating protonation probabilities for a given pH-range, usually from 0 to 14.

```
ce_model.CalculateCurves (isAnalytic = True, forceSerial = True, directory =
    "curves_analytic")
```

The argument `isAnalytic=True` tells that the protonation probabilities should be calculated analytically. Each pH-step will be calculated in a separate thread. However, the analytic evaluation of titration curves seems to be very inefficient in parallel mode. The argument `forceSerial=True` will switch on serial mode regardless of the number of threads declared previously. For each instance of each site, a data file containing the titration curve will be generated. The files will be placed in the directory specified by the argument `directory`. The curves can be plotted in a plotting program, for example Gnuplot.

The same task can be accomplished by using GMCT in parallel mode:

```
ce_model.CalculateCurves (directory = "curves_gmct")
```

6 Parameter files

Parameter files in YAML format reside in the directory `parameters/`. The file `radii.yaml` defines a list of atomic radii required for calculating the boundary between the protein phase and the solvent phase. The files in the `sites/` subdirectory define parameters for titratable sites. The listing below shows the parameter file for aspartate:

```
---
site       : ASP
atoms      : [CB, HB1, HB2, CG, OD1, OD2]
instances :
  - label   : p
    Gmodel  : -5.487135
    protons : 1
    charges : [-0.21,  0.09,  0.09,  0.75, -0.36, -0.36]

  - label   : d
    Gmodel  : 0.000000
    protons : 0
    charges : [-0.28,  0.09,  0.09,  0.62, -0.76, -0.76]
...
```

9

The site's name is defined by the parameter `site`. `atoms` defines a list of names of atoms that constitute the site. The consecutive lines define instances of the site. `label` is the name of the instance, for example "p" for "protonated". `Gmodel` is the energy of the model compound expressed in kcal/mol. G_{model} has been calculated at the temperature of $300\,\mathrm{K}$ from the pK_a value of aspartate, which is 4.0. The values of model energies are recalculated if a different temperature is to be used. The reason for $G_{model} = 0$ for the deprotonated instance is that the energies are calculated relative to a reference state, which is a fully deprotonated state. The parameter `protons` tells the number of protons bound to the site and is used during the calculation of the microstate energy. Finally, `charges` is a list of atomic charges that differ between the instances. The order of charges is the same as the order of atoms in the `atoms` list. The charges usually come from the CHARMM force field, i.e. they are parametrized for the dielectric constant $\epsilon = 4$.

Proteins often contain non-standard titratable sites, for example ligands or cofactors. The parameter files describing these sites are automatically loaded from the current directory as long as they have the extension "*.yaml". If the names of the added sites coincide with the standard names, the parameters for the standard sites will be overwritten.

7 Future developments

- Extend the model by including rotameric instances

- Include quantum chemical sites as in QMPB

- Include reduction/oxidation reactions in addition to protonation/deprotonation reactions

- Enclose the MEAD and GMCT codes inside pDynamo

- Optionally convert kcal/mol (MEAD units) to kJ/mol (pDynamo units)

- Move parts of the `WriteJobFiles` method to the instance class

- Improve the efficiency of writing job files

- Calculate ETA (Estimated Time for Accomplishment) during MEAD calculations

- Rename variables containing filenames so that they start from "file"

- Make use of the `pqr2SolvAccVol` program to speed up calculations

- Use arrays instead of lists for storing interactions

- Move the method for calculating G_{micro} from Python to C

8 References

[1] M. J. Field, "The pDynamo program for molecular simulations using hybrid quantum chemical and molecular mechanical potentials," *J. Chem. Theory Comput.*, vol. 4, no. 7, pp. 1151–1161, 2008.

[2] D. Bashford, *An Object-Oriented Programming Suite for Electrostatic Effects in Biological Molecules.*, pp. 233–240. 1997.

[3] T. Essigke, *A Continuum Electrostatic Approach for Calculating the Binding Energetics of Multiple Ligands.* PhD thesis, University of Bayreuth, 2008.

[4] R. T. Ullmann and G. M. Ullmann, "Gmct : A monte carlo simulation package for macromolecular receptors," *J. Comp. Chem*, vol. 33, pp. 887–900, 2012.

[5] MacKerell, A. D. et al., "All-atom empirical potential for molecular modeling and dynamics studies of proteins," *J. Phys. Chem. B*, vol. 102, pp. 3586–3616, 1998.

[6] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus, "CHARMM: A program for macromolecular energy, minimization, and dynamics calculations," *J. Comput. Chem.*, vol. 4, pp. 187–217, 1983.

[7] W. Humphrey, A. Dalke, and K. Schulten, "VMD: Visual molecular dynamics," *J. Mol. Graph.*, vol. 14, no. 1, pp. 33–38, 1996.

Websites

- MEAD (Donald Bashford)
  http://stjuderesearch.org/site/lab/bashford/

- Extended MEAD (Thomas Ullmann)

  `http://www.bisb.uni-bayreuth.de/People/ullmannt/index.php?name=extended-mead`

- GMCT (Thomas Ullmann)

  `http://www.bisb.uni-bayreuth.de/People/ullmannt/index.php?name=gmct-gcem`