ContinuumElectrostatics

A pDynamo module for protonation state calculations

Mikolaj Feliks
Institut de Biologie Structurale, Grenoble
Last updated: February 9, 2015

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[1].

The ContinuumElectrostatics module extends the pDynamo software library by Martin Field[2] with a Poisson-Boltzmann continuum electrostatic model that allows for the calculation of protonation states of individual sites[3–5]. The module connects pDynamo to the external solver of the Poisson-Boltzmann equation, MEAD[6]. MEAD was developed by Donald Bashford and later extended by Timm Essigke[7] and Thomas Ullmann[8]. The idea to employ MEAD for the calculation of Poisson-Boltzmann electrostatic energies is similar to how pDynamo interfaces with ORCA to calculate quantum chemical energies and properties.

After the electrostatic energy terms have been obtained from MEAD, there are three ways to calculate the probabilities of protonation states. First, they can be calculated by using the statistical mechanical partition function, which is, however, only possible for small proteins. Second, the probabilities can be estimated by using the module's custom Metropolis Monte Carlo method[9]. Third, they can be estimated by the external Monte Carlo sampling program, GMCT[8;10]. GMCT was written by Matthias Ullmann and extended by Thomas Ullmann. The letter two methods allow 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 and the Monte Carlo sampling work 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.
In principle, the ContinuumElectrostatics module combines the functionalities of two previous software packages, namely the `multiflex2qmpb.pl` tool, which is part of the QMPB package written by Timm Essigke[7], and the Monte Carlo sampling program, GMCT[8;10]. These functionalities can be accessed by the user in a Python-based scripting environment. However, the "engine" of the module that handles actual calculations is located in the underlying layer written in C. The two layers, Python and C, are glued together with Cython.

4 Installation and configuration

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

- pDynamo 1.8.0
- Python 2.7 (including header files; python2.7-dev package in Debian)
- PyYAML 3.10 (python-yaml package in Debian)
- Cython 0.15.1 (optionally; for recompiling Cython sources)
- GCC (any version should be fine)
- Extended MEAD 2.3.0
- GMCT 1.2.3 (optionally)

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`. From code revision 179 onwards, there is no need to use GMCT for Monte Carlo sampling, because the module comes with its own, built-in sampling routines. Nevertheless, to perform sampling with GMCT, it is necessary to only have 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's code are written in C or Cython and therefore have to be compiled before use. These parts include the state vector, the calculation of the microstate energy and the analytic evaluation of protonation state probabilities.

Go to the directory `extensions/cython/`. In the `Makefile`, edit the first line starting from "PDYNAMO_PCORE". It should define the location of the pCore module of the present pDynamo installation. After closing the file, run "make". If you also want to recompile the Cython sources, precede "make" by running "make clean_all". In case the Cython compilation fails, it may be necessary to manually specify the location of the pCore module in the `cython_compile.py` script. Finally, run "make install" to install the compiled library.

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[11]. In the first step, prepare CHARMM topology (PSF) and coordinate (CRD) files. The preparation can be done using the programs CHARMM[12] or VMD[13]. 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:

```
cem = MEADModel (system=mol, pathMEAD="/home/mikolaj/local/bin/",
    pathScratch="scratch", nthreads=2)
```

The first parameter, "system", indicates the CHARMM-based protein model. The parameter "pathMEAD" specifies the directory where the MEAD programs, `my_2diel_solver` and

`my_3diel_solver`, are located. If none of these directories are given, `/usr/bin` is assumed by default. The parameter "pathScratch" 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:

```
cem.Initialize ()
```

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

```
cem.Summary ()
cem.SummarySites ()
```

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

```
cem.WriteJobFiles ()
```

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:

```
cem.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 heterogeneous transfer energy, $G_{\mathrm{heterotrans}}$. 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 pK_{a} value of the deprotonation reaction. The site in the protein has an intrinsic energy $G_{\mathrm{intr}} = G_{\mathrm{model}} + G_{\mathrm{heterotrans}}$. The `my_2diel_solver` program calculates $G_{\mathrm{Born,model}}$ and $G_{\mathrm{back,model}}$. The `my_3diel_solver` program calculates $G_{\mathrm{Born,protein}}$ and $G_{\mathrm{back,protein}}$ and, additionally, electrostatic interaction energies of an instance of a site with other instances of other sites in the protein. The ContinuumElectrostatics module collects $G_{\mathrm{Born,protein}}$, $G_{\mathrm{back,protein}}$, $G_{\mathrm{Born,model}}$, $G_{\mathrm{back,model}}$ and interaction energies from MEAD and calculates $G_{\mathrm{heterotrans}}$ 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 (cem)
increment   = True
while increment:
  Gmicro = cem.CalculateMicrostateEnergy (statevector, pH=7.0)
  statevector.Print (verbose=True, 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 probabilities of protonation states

In the ContinuumElectrostatics module, the probabilities of protonation states can be calculated either analytically for small proteins or by using Monte Carlo sampling 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.

```
cem.CalculateProbabilitiesAnalytically ()
cem.SummaryProbabilities ()
```

For comparison, the probabilities are calculated by using a Monte Carlo method.

```
cem.CalculateProbabilitiesMonteCarlo ()
cem.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.

5.6 Calculating titration curves

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

```
from ContinuumElectrostatics import TitrationCurves
tc = TitrationCurves (cem, method="analytic")
tc.CalculateCurves ()
tc.WriteCurves (directory="curves_analytic")
```

The argument `method="analytic"` tells that the protonation probabilities should be calculated analytically. 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 differently:

```
from ContinuumElectrostatics import TitrationCurves
tc = TitrationCurves (cem, method="MonteCarlo")
tc.CalculateCurves ()
tc.WriteCurves (directory="curves_mc")
```

5.7 Calculating microstate energies of substates

From the calculated protonation probabilities, one can generate a state vector describing the lowest energy protonation state:

```
vector_lowest = StateVector_FromProbabilities (cem)
```

The microstate energy (G_{micro}) in the lowest energy protonation state can be calculated in the following way:

```
Gmicro = cem.CalculateMicrostateEnergy (vector_lowest)
```

By changing the values (=instances) of selected components (=sites) of the state vector, one can calculate energies for a series of substates of the lowest energy state. On the other hand, substate energies can be calculated easier with the `MEADSubstate` class:

```
sites = (("PRTA", "ASP",  2), ("PRTA", "GLU", 14), ("PRTA", "ARG", 15))
substate = MEADSubstate (cem, sites, pH=7.0)
substate.CalculateSubstateEnergies ()
substate.Summary ()
```

The above code was taken from the test case "defensin". Note that `pH = 7.0` is an optional argument and that the given pH should be the same as the pH used during the calculation of protonation probabilities. The code will generate a table of substates with their corresponding relative energies:

```
-------------------------------------------------------------------------
 State   Gmicro   Charge   Protons   PRTA ASP 2    PRTA GLU 14   PRTA ARG 15
-------------------------------------------------------------------------
     1      0.00       -1         1         d             d             p
     2      4.55        0         2         p             d             p
     3      6.69       -2         0         d             d             d
     4     10.05        0         2         d             p             p
     5     11.21       -1         1         p             d             d
     6     14.86        1         3         p             p             p
     7     16.62       -1         1         d             p             d
     8     21.40        0         2         p             p             d
-------------------------------------------------------------------------
```

In the lowest energy state, Asp2 and Glu14 are deprotonated, while Arg15 and Cys31 are protonated. Protonating Asp2 requires about 4.6 kcal/mol.

6 Parameter files

Two parameter formats are recognized, YAML (pDynamo's default format) and EST (QMPB's format). The file `radii.yaml` defines a list of atomic radii required to calculate the boundary between the protein phase and the solvent phase. The files in the `sites/` directory 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]
...
```

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 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 $G_{\mathrm{model}} = 0$ 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 recognized by their extension and can be loaded automatically from the current directory. These additional parameter files can be provided both in EST or YAML formats (the EST format takes precedence). If the names of the new sites coincide with the names of already existing sites, the parameters of the existing 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

- Have a custom Poisson-Boltzmann solver, instead of MEAD
- Optionally convert kcal/mol (MEAD units) to kJ/mol (pDynamo units)
- Make use of the `pqr2SolvAccVol` program to speed up calculations a bit

8 Feedback

The ContinuumElectrostatics module is in a stage of active development. Therefore, bugs are inevitable. If you found a bug, have a question or wish contribute to the code, please write to mikolaj.feliks@gmail.com. Alternatively, you can post comments or report problems on the project's website, `http://code.google.com/p/pdynamo-extensions/`.

9 References

[1] E. Bombarda and G. M. Ullmann, "ph-dependent pk(a) values in proteins-a theoretical analysis of protonation energies with practical consequences for enzymatic reactions," *J. Phys. Chem. B*, vol. 114, no. 5, pp. 1994–2003, 2010.

[2] 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.

[3] D. Bashford and K. Gerwert, "Electrostatic Calculations of the pK_a Values of Ionizable Groups in Bacteriorhodopsin.," *J. Mol. Biol.*, vol. 224, pp. 473–486, 1992.

[4] G. M. Ullmann and E. W. Knapp, "Electrostatic models for computing protonation and redox equilibria in proteins," *Eur. Biophys. J.*, vol. 28, no. 7, pp. 533–551, 1999.

[5] A. Klingen, E. Bombarda, and G. Ullmann, "Theoretical investigation of the behavior of titratable groups in proteins," *Photochem. Photobiol. Sci.*, vol. 5, no. 6, pp. 588–596, 2006.

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

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

[8] T. Ullmann, *Monte Carlo Simulation Methods for Studying the Thermodynamics of Ligand Binding and Transfer Processes in Biomolecules*. PhD thesis, University of Bayreuth, 2012.

[9] P. Beroza, D. R. Fredkin, M. Y. Okamura, and G. Feher, "Protonation of interacting residues in a protein by a Monte Carlo method: Application to lysozyme and the photosynthetic reaction center.," vol. 88, pp. 5804–5808, 1991.

[10] 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.

[11] 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.

[12] 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.

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

Websites

- pDynamo (Martin J. Field)

  `http://pdynamo.org`

- 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`

- Doctoral thesis of Timm Essigke (which was my primary source of knowledge)

  `http://epub.uni-bayreuth.de/655/`

- Doctoral thesis of Thomas Ullmann

  `http://epub.uni-bayreuth.de/199/`

- Doctoral thesis of Astrid Klingen

  `http://epub.uni-bayreuth.de/763/`