pdb2pqr

Release 3.0.0

Jul 26, 2020

Contents

Release 3.0.0

Date Jul 26, 2020

CHAPTER 1

Overview

The use of continuum solvation methods such as APBS requires accurate and complete structural data as well as force field parameters such as atomic charges and radii. Unfortunately, the limiting step in continuum electrostatics calculations is often the addition of missing atomic coordinates to molecular structures from the Protein Data Bank and the assignment of parameters to these structures. To address this problem, we have developed PDB2PQR. This software automates many of the common tasks of preparing structures for continuum solvation calculations as well as many other types of biomolecular structure modeling, analysis, and simulation. These tasks include:

- Adding a limited number of missing heavy (non-hydrogen) atoms to biomolecular structures.
- Estimating titration states and protonating biomolecules in a manner consistent with favorable hydrogen bonding.
- Assigning charge and radius parameters from a variety of force fields.
- Generating "PQR" output compatible with several popular computational modeling and analysis packages.
- This service is intended to facilitate the setup and execution of electrostatics calculations for both experts and non-experts and thereby broaden the accessibility of biomolecular solvation and electrostatics analyses to the biomedical community.

CHAPTER 2

Contents

2.1 Getting PDB2PQR

Note: *Before you begin!* PDB2PQR funding is dependent on your help for continued development and support. Please register before using the software so we can accurately report the number of users to our funding agencies.

2.1.1 Web servers

Most functionality is available through our online web servers.

The PDB2PQR web server offers a simple way to use both APBS and PDB2PQR without the need to download and install additional programs.

After registering, please visit http://server.poissonboltzmann.org/ to access the web server.

2.1.2 Python Package Installer (PIP)

Most users who want to use the software offline should install it via **pip** with the following command

```
pip install pdb2pqr
```

from within your favorite virtual environment.

The PIP package provides the *pdb2pqr* Python module as well as the **pdb2pqr30** program which can be used from the command line.

2.1.3 Installation from source code

You can also download the source code from GitHub (we recommend using a tagged release) and building the code yourself with:

```
pip install .
```

from the top-level of the source code directory. Note that developers may want to install the code in "editable" mode with

```
pip install -e .
```

Testing

The software can be tested for correct functioning via Coverage.py with

or pytest

from the top level of the source directory.

2.2 Using PDB2PQR

Note: *Before you begin!* PDB2PQR funding is dependent on your help for continued development and support. Please register before using the software so we can accurately report the number of users to our funding agencies.

PDB2PQR is often used together with the APBS software; e.g., ,in the following type of workflow

1. Start with a PDB ID or locally generated PDB file (see *PDB molecular structure format*).
2. Assign titration states and parameters with **pdb2pqr** to convert the protein and ligands to PQR format (see *PQR molecular structure format*).
3. Perform electrostatics calculations with **apbs** (can be done from within the PDB2PQR web server).
4. Visualize results from within PDB2PQR web server or with *Other software*.

2.2.1 Web server use

Most users will use PDB2PQR through the web server (after registering, of course). However, it is also possible to install local versions of PDB2PQR and run these through the command line.

2.2.2 Command line use

```
pdb2pqr30 [options] --ff={forcefield} {path} {output-path}
```

This module takes a PDB file as input and performs optimizations before yielding a new PQR-style file in `{output-path}`. If `{path}` is a PDB ID it will automatically be retrieved from the online PDB archive.

In addition to the required `{path}` and `{output-path}` arguments, **pdb2pqr30** requires one of the following options:

- `--ff=FIELD_NAME` specifying the forcefield to use. Run `pdb2pqr30 --help` to see specific options.
- `--userff=USER_FIELD_FILE` specifying a user-created forcefield file. Requires `--usernames` and overrides `--ff`.
- `--clean` specifying no optimization, atom addition, or parameter assignment, just return the original PDB file in aligned format. Overrides `--ff` and `--userff` options.

Information about additional options can be obtained by running:

```
pdb2pqr30 --help
```

2.2.3 Examples and algorithms

Examples

In order to perform electrostatics calculations on your biomolecular structure of interest, you need to provide atomic charge and radius information to APBS. Charges are used to form the biomolecular charge distribution for the Poisson-Boltzmann (PB) equation while the radii are used to construct the dielectric and ionic accessibility functions.

The PDB2PQR web service and software will convert most PDB files into PQR format with some caveats. Although PDB2PQR can fix some missing heavy atoms in sidechains, it does not currently have the (nontrivial) capability to model in large regions of missing backbone and sidechain coordinates. Be patient and make certain that the job you submitted to the PDB2PQR website has finished and you have downloaded the resulting PQR file correctly. It usually takes less than 10 minutes for the job to finish.

These examples assume that you have registered and have access to the PDB2PQR web server.

Adding hydrogens and assigning parameters with the PDB2PQR web server

This example uses http://server.poissonboltzmann.org.

Pick a structure

Start by choosing a PDB file to process. Either enter the 4-character PDB ID into PDB2PQR or accession number (1FAS is a good starting choice) or upload your own PDB file. Note that, if you choose to enter a 4-character PDB ID, PDB2PQR will process all recognizable chains of PDB file as it was deposited in the PDB (e.g., not the biological unit, any related transformations, etc.).

Pick a forcefield

For most applications, the choice is easy: PARSE. This forcefield has been optimized for implicit solvent calculation and is probably the best choice for visualization of protein electrostatics and many common types of energetic calculations for proteins. However, AMBER and CHARMM may be more appropriate if you are attempting to compare directly to simulations performed with those force fields, require nucleic acid support, are simulating ligands parameterized with those force fields, etc.

It is also possible to upload a user-defined forcefield (e.g., to define radii and charges for ligands or unusual residues). Please see *Extending PDB2PQR* for more information.

Pick a naming scheme

This choice is largely irrelevant to electrostatics calculations but may be important for some visualization programs. When in doubt, choose the "Internal naming scheme" which attempts to conform to IUPAC standards.

Reconstruct missing atoms (hydrogens)

Under options, be sure the "Ensure that new atoms are not rebuilt too close to existing atoms" and "Optimize the hydrogen bonding network" options are selected. You can select other options as well, if interested.

Download and view the results

Download the resulting PQR file and visualize in a molecular graphics program to examine how the hydrogens were added and how hydrogen bonds were optimized.

Parameterizing ligands with the PDB2PQR web server

This section outlines the parameterization of ligands using the PEOE_PB methods (see Czodrowski et al, 2006 for more information).

The PDB structure 1HPX includes HIV-1 protease complexed with an inhibitor at 2.0 Å resolution. HIV-1 protease has two chains; residue D25 is anionic on one chain and neutral on the other – these titration states are important in the role of D25 as an acid in the catalytic mechanism.

Ignoring the ligand

If we don't want to include the ligand, then the process is straightforward:

1. From the PDB2PQR server web page, enter `1HPX` into the PDB ID field.
2. Choose whichever forcefield and naming schemes you prefer.
3. Under options, be sure the "Ensure that new atoms are not rebuilt too close to existing atoms", "Optimize the hydrogen bonding network", and "Use PROPKA to assign protonation states at pH" options are selected. Choose pH 7 for your initial calculations. You can select other options as well, if interested.
4. Hit the "Submit" button.
5. Once the calculations are complete, you should see a web page with a link to the PROPKA output, a new PQR file, and warnings about the ligand KNI (since we didn't choose to parameterize it in this calculation). For comparison, you might download the the original PDB file and compare the PDB2PQR-generated structure with the original to see where hydrogens were placed.

Parameterizing the ligand

This section outlines the parameterization of ligands using the PEOE_PB methods (see DOI:10.1002/prot.21110).

Ligand parameterization currently requires a *MOL2-format* representation of the ligand to provide the necessary bonding information. MOL2-format files can be obtained through the PRODRG web server or some molecular modeling software packages. PRODRG provides documentation as well as several examples on ligand preparation on its web page.

We're now ready to look at the 1HPV crystal structure from above and parameterize its ligand, KNI-272.

1. From the PDB2PQR server web page, enter `1HPX` into the PDB ID field.
2. Choose whichever forcefield and naming schemes you prefer.

3. Under options, be sure the "Ensure that new atoms are not rebuilt too close to existing atoms", "Optimize the hydrogen bonding network", and "Assign charges to the ligand specified in a MOL2 file" options are selected. You can select other options as well, if interested.
4. Hit the "Submit" button.
5. Once the calculations are complete, you should see a web page with a link to the new PQR file with a warning about debumping P81 (but no warnings about ligand parameterization!).

As a second example, we use the PDB structure 1ABF of L-arabinose binding protein in complex with a sugar ligand at 1.90 Å resolution. To parameterize both this protein and its ligand:

1. From the PDB2PQR server web page, enter *1ABF* into the PDB ID field.
2. Choose whichever forcefield and naming schemes you prefer.
3. Under options, be sure the "Ensure that new atoms are not rebuilt too close to existing atoms", "Optimize the hydrogen bonding network", and "Assign charges to the ligand specified in a MOL2 file" options are selected. You can select other options as well, if interested.
4. Hit the "Submit" button.
5. Once the calculations are complete, you should see a web page with a link to the new PQR file with a warning about debumping P66, K295, and K306 (but no warnings about ligand parameterization!).

Algorithms used by PDB2PQR

Debumping

Unless otherwise instructed with `--nodebump`, PDB2PQR will attempt to remove steric clashes (debump) between residues.

To determine if a residue needs to be debumped, PDB2PQR compares its atoms to all nearby atoms. With the exception of donor/acceptor pairs and CYS residue SS bonded pairs, a residue needs to be debumped if any of its atoms are within cutoff distance of any other atoms. The cutoff is 1.0 angstrom for hydrogen/hydrogen collisions, 1.5 angstrom for hydrogen/heavy collisions, and 2.0 angstrom otherwise.

Considering the atoms that are conflicted, PDB2PQR changes selected dihedral angle configurations in increments of 5.0 degrees, looking for positions where the residue does not conflict with other atoms. If modifying a dihedral angle does not result in a debumped configuration then the dihedral angle is reset and the next one is tried. If 10 angles are tried without success the algorithm reports failure.

> **Warning:** It should be noted that this is not an optimal solution. This method is not guaranteed to find a solution if it exists and will accept the first completely debumped state found, not the optimal state.
>
> Additionally, PDB2PQR does not consider water atoms when looking for conflicts.

Hydrogen bond optimization

Unless otherwise indicated with `--noopts`, PDB2PQR will attempt to add hydrogens in a way that optimizes hydrogen bonding.

The hydrogen bonding network optimization seeks, as the name suggests, to optimize the hydrogen bonding network of the protein. Currently this entails manipulating the following residues:

- Flipping the side chains of HIS (including user defined HIS states), ASN, and GLN residues;

- Rotating the sidechain hydrogen on SER, THR, TYR, and CYS (if available);
- Determining the best placement for the sidechain hydrogen on neutral HIS, protonated GLU, and protonated ASP;
- Optimizing all water hydrogens.

Titration state assignment

Versions 2.1 and earlier of PDB2PQR offered the following methods to assign titration states to molecules at a specified pH.

- PROPKA. This method uses the PROPKA software to assign titration states. More information about PROPKA can be found on its website.
- PDB2PKA. Uses a Poisson-Boltzmann method to assign titration states. This approach is loosely related to the method described by Nielsen and Vriend (2001) doi:10.1002/prot.10.

The current version (3.0.0) of PDB2PQR currently only supports PROPKA while we address portability issues in PDB2PKA.

PDB2PQR has the ability to recognize certain protonation states and keep them fixed during optimization. To use this feature manually rename the residue name in the PDB file as follows:

- Neutral ASP: ASH
- Negative CYS: CYM
- Neutral GLU: GLH
- Neutral HIS: HIE or HSE (epsilon-protonated); HID or HSD (delta-protonated)
- Positive HIS: HIP or HSP
- Neutral LYS: LYN
- Negative TYR: TYM

PDB2PQR is unable to assign charges and radii when they are not available in the forcefield - thus this warning message will occur for most ligands unless a `MOL2` file is provided for the ligand with the `--ligand` option. Occasionally this message will occur in error for a standard amino acid residue where an atom or residue may be misnamed. However, some of the protonation states derived from the PROPKA results are not supported in the requested forcefield and thus PDB2PQR is unable to get charges and radii for that state. PDB2PQR currently supports the following states as derived from PROPKA:

Protonation State	AMBER Support	CHARMM Support	PARSE Support
Neutral N-Terminus	No	No	Yes
Neutral C-Terminus	No	No	Yes
Neutral ARG	No	No	No
Neutral ASP	Yes[1]	Yes	Yes
Negative CYS	Yes[1]	No	Yes
Neutral GLU	Yes[1]	Yes	Yes
Neutral HIS	Yes	Yes	Yes
Neutral LYS	Yes[1]	No	Yes
Negative TYR	No	No	Yes

[1] Only if residue is not a terminal residue; if the residue is terminal it will not be set to this state.

Other software

A variety of other software can be used to visualize and process the results of PDB2PQR and APBS calculations.

Visualization software

Examples of visualization software that work with output from PDB2PQR and APBS:

- PyMOL
- VMD
- Chimera
- PMV

Dynamics simulations

As an example of PDB2PQR and APBS integration with molecular mechanics sofware, the iAPBS library was developed to facilitate the integration of APBS with other molecular simulation packages. This library has enabled the integration of APBS with several molecular dynamics packages, including NAMD, AMBER, and CHARMM.

APBS is also used directly by Brownian dynamics software such as SDA and BrownDye.

2.3 Getting help

2.3.1 GitHub issues

Our preferred mechanism for user questions and feedback is via GitHub issues. We monitor these issues daily and usually respond within a few days.

2.3.2 Announcements

Announcements about updates to the APBS-PDB2PQR software and related news are available through our mailing list; please register for updates.

2.3.3 Old mailing lists

We continue to monitor the pdb2pqr-users and apbs-users mailing lists. However, we are in the process of phasing out these mailing lists (due to high spam content) in favor of user support through GitHub issues.

2.3.4 Contacting the authors

If all else fails, feel free to contact nathanandrewbaker@gmail.com.

2.4 Supporting PDB2PQR

2.4.1 Please register as a user!

Please help ensure continued support for APBS-PDB2PQR by registering your use of our software.

2.4.2 Citing our software

If you use PDB2PQR in your research, please cite one or more of the following papers:

- Jurrus E, Engel D, Star K, Monson K, Brandi J, Felberg LE, Brookes DH, Wilson L, Chen J, Liles K, Chun M, Li P, Gohara DW, Dolinsky T, Konecny R, Koes DR, Nielsen JE, Head-Gordon T, Geng W, Krasny R, Wei G-W, Holst MJ, McCammon JA, Baker NA. Improvements to the APBS biomolecular solvation software suite. Protein Sci, 27 (1), 112-128, 2018. https://doi.org/10.1002/pro.3280
- Unni S, Huang Y, Hanson RM, Tobias M, Krishnan S, Li WW, Nielsen JE, Baker NA. Web servers and services for electrostatics calculations with APBS and PDB2PQR. J Comput Chem, 32 (7), 1488-1491, 2011. http://dx.doi.org/10.1002/jcc.21720
- Dolinsky TJ, Czodrowski P, Li H, Nielsen JE, Jensen JH, Klebe G, Baker NA. PDB2PQR: Expanding and upgrading automated preparation of biomolecular structures for molecular simulations. Nucleic Acids Res, 35, W522-5, 2007. http://dx.doi.org/10.1093/nar/gkm276
- Dolinsky TJ, Nielsen JE, McCammon JA, Baker NA. PDB2PQR: an automated pipeline for the setup, execution, and analysis of Poisson-Boltzmann electrostatics calculations. Nucleic Acids Res, 32, W665-7, 2004. http://dx.doi.org/10.1093/nar/gkh381

2.4.3 Supporting organizations

The PDB2PQR authors would like to give special thanks to the supporting organizations behind the APBS and PDB2PQR software:

National Institutes of Health Primary source of funding for APBS via grant GM069702

National Biomedical Computation Resource Deployment and computational resources support from 2002 to 2020

National Partnership for Advanced Computational Infrastructure Funding and computational resources

Washington University in St. Louis Start-up funding

2.5 Extending PDB2PQR

2.5.1 Adding new forcefield parameters

If you are just adding the parameters of a few residues and atoms to an existing forcefield (e.g., AMBER), you can open the forcefield data file distributed with PDB2PQR (`dat/AMBER.DAT`) directly and add your parameters. After the parameter addition, save the force field data file with your changes. You should also update the corresponding .names file (`dat/AMBER.names`) if your added residue or atom naming scheme is different from the PDB2PQR canonical naming scheme.

2.5.2 Adding an entirely new forcefield

The following steps outline how to add a new force field to PDB2PQR.

You will need to generate a forcefield data file (e.g., `myff.DAT`) and, if your atom naming scheme of the forcefield is different from the PDB2PQR canonical naming scheme, you will also need to provide a names files (`myFF.names`). The format of the names file is described in *PDB2PQR NAMES files*. It is recommended to build your own forcefield data and names files based on existing PDB2PQR `.DAT` and `.names` examples provided with PDB2PQR in the `dat` directory. After finishing your forcefield data file and names file, these can be used with either the command line or the web server versions of PDB2PQR.

2.5.3 Adding new functionality

PDB2PQR welcomes new contributions; the software API is documented in *API Reference*. To contribute code, submit a *pull request* against the master branch in the PDB2PQR repository. Please be sure to run PDB2PQR tests, as described in *Testing*, before submitting new code.

2.6 File formats

2.6.1 Molecular structure formats

PQR molecular structure format

This format is a modification of the PDB format which allows users to add charge and radius parameters to existing PDB data while keeping it in a format amenable to visualization with standard molecular graphics programs. The origins of the PQR format are somewhat uncertain, but has been used by several computational biology software programs, including MEAD and AutoDock. UHBD uses a very similar format called QCD.

APBS reads very loosely-formatted PQR files: all fields are whitespace-delimited rather than the strict column formatting mandated by the PDB format. This more liberal formatting allows coordinates which are larger/smaller than $\pm$ 999 Å. APBS reads data on a per-line basis from PQR files using the following format::

```
Field_name Atom_number Atom_name Residue_name Chain_ID Residue_number X Y Z Charge␣
↪Radius
```

where the whitespace is the most important feature of this format. The fields are:

`Field_name` A string which specifies the type of PQR entry and should either be ATOM or HETATM in order to be parsed by APBS.

`Atom_number` An integer which provides the atom index.

`Atom_name` A string which provides the atom name.

`Residue_name` A string which provides the residue name.

`Chain_ID` An optional string which provides the chain ID of the atom. Note that chain ID support is a new feature of APBS 0.5.0 and later versions.

`Residue_number` An integer which provides the residue index.

`X Y Z` 3 floats which provide the atomic coordinates (in Å)

`Charge` A float which provides the atomic charge (in electrons).

`Radius` A float which provides the atomic radius (in Å).

Clearly, this format can deviate wildly from PDB due to the use of whitespaces rather than specific column widths and alignments. This deviation can be particularly significant when large coordinate values are used. However, in order to maintain compatibility with most molecular graphics programs, the PDB2PQR program and the utilities provided with APBS attempt to preserve the PDB format as much as possible.

PDB molecular structure format

The PDB file format is described in detail in the Protein Data Bank documentation.

MOL2 molecular structure format

The MOL2 file format is a popular method for specifying chemical structure, including atom types, positions, and bonding. It is described in detail in the Tripos documentation.

2.6.2 Parameter formats

PDB2PQR NAMES files

Much of the difficulty in adding a new forcefield to PDB2PQR depends on the naming scheme used in that forcefield.

XML file format

To start, either a flat file or XML file containing the desired forcefield's parameters should be made - see `AMBER.DAT` and `AMBER.xml` for examples. If the forcefield's naming scheme matches the canonical naming scheme, that's all that is necessary. If the naming schemes differ, however, conversions must be made. These are made in the `*.names` file (see `CHARMM.names`, for example). In this file you will see sections like:</p>

```
<residue>
  <name>WAT</name>
  <useresname>TP3M</useresname>
  <atom>
    <name>O</name>
    <useatomname>OH2</useatomname>
  </atom>
</residue>
```

This section tells PDB2PQR that for the oxygen atom O in WAT, CHARMM uses the names OH2 and TP3M, respectively. When the XML file is read in, PDB2PQR ensures that the WAT/O pair points to TP3M/OH2 such that the appropriate parameters are returned. But for naming schemes that greatly differ from the PDB2PQR canonical naming scheme, this could get really ugly. As a result, PDB2PQR can use regular expressions to simplify the renaming process, i.e.:

```
<residue>
  <name>[NC]?...$</name>
  <atom>
    <name>H</name>
    <useatomname>HN</useatomname>
  </atom>
</residue>
```

This section of code will ensure that the H atom of all canonical residue names that match the `[NC]?...$` regular expression point to HN instead. This regular expression matches all three-letter residue names, residue names with

an 'N' prepended (N-Termini), and residue names with a 'C' prepended (C-Termini). For twenty amino acids, sixty residue name changes can all be done by a single section. The use of regular expressions is therefore a much more powerful method of handling naming scheme differences than working on a one to one basis.

There are a few other additional notes when using the `.names` file. First, the `$group` variable is used to denote the matching group of a regular expression, for instance:

```
<residue>
  <name>HI([PDE])$</name>
  <useresname>HS$group</useresname>
</residue>
```

This section replaces HIP/HID/HIE with HSP/HSD/HSE by first matching the HI([PDE])$ regular expression and then using the group that is enclosed by parantheses to fill in the name to use.

Second, sections are cumulative - since CHARMM, for instance, has a patch-based naming scheme, one single canonical residue name can map to multiple forcefield-scheme names. Let's look at how to map an SS-bonded Cysteine (canonical name CYX) to the CHARMM naming scheme:

```
<residue>
  <name>CYX</name>
  <useresname>CYS</useresname>
</residue>
<residue>
  <name>CYX</name>
  <useresname>DISU</useresname>
  <atom>
    <name>CB</name>
    <useatomname>1CB</useatomname>
  </atom>
  <atom>
    <name>SG</name>
    <useatomname>1SG</useatomname>
  </atom>
</residue>
```

The CYX residue is first mapped to CHARMM's CYS, and then to CHARMM's DISU object. All atom names that are found in DISU overwrite those found in CYS - in effect, the DISU patch is applied to CYS, yielding the desired CYX. This cumulative can be repeated as necessary.

Caveats about atom naming

In an ideal world each individual residue and atom would have a standard, distinct name. Unfortunately several naming schemes for atoms exist, particularly for hydrogens. As such, in order to detect the presence/absence of atoms in a protein, an internal canonical naming scheme is used. The naming scheme used in PDB2PQR is the one recommended by the PDB itself, and derives from the IUPAC naming recommendations[1]

This canonical naming scheme is used as the default PDB2PQR output. All conversions in PDB2PQR use the internal canonical naming scheme to determine distinct atom names. In previous versions of PDB2PQR, these conversions were stored in long lists of if statements, but for transparency and editing this is a bad thing. Instead, all conversions can now be found in XML as described above.

[1] J. L. Markley, et al., "Recommendations for the Presentation of NMR Structures of Proteins and Nucleic Acids," Pure & Appl. Chem., 70 (1998): 117-142. DOI:10.1046/j.1432-1327.1998.2560001.x

There are a few additions to the canonical naming scheme, mirrored after the AMBER naming scheme (chosen since for the most part it follows the IUPAC recommendations). These changes are made in `PATCHES.xml`, and allow any of the following to be patched as necessary as well as detected on input:

N* N-Terminal Residue (i.e. NALA, NLEU)

NEUTRAL-N* Neutral N-Terminal Residue

C* C-Terminal Residue (i.e. CLYS, CTYR)

NEUTRAL-C* Neutral C-Terminal Residue

***5** 5-Terminus for Nucleic Acids (i.e. DA5)

***3** 3-Terminus for Nucleic Acids (i.e. DA3)

ASH Neutral ASP

CYX SS-bonded CYS

CYM Negative CYS

GLH Neutral GLU

HIP Positive HIS

HID Neutral HIS, proton HD1 present

HIE Neutral HIS, proton HE2 present

LYN Neutral LYS

TYM Negative TYR

2.7 API Reference

The **pdb2pqr30** command provides a command-line interface to PDB2PQR's functionality. It is built on classes and functions in the *pdb2pqr* module. The API of *pdb2pqr* is documented here for developers who might want to directly use the PDB2PQR code.

Note: The API is still changing and there is currently no guarantee that it will remain stable between minor releases.

2.7.1 Forcefield support

definitions	XML handling for biomolecular residues.
forcefield	Force fields are fun.

pdb2pqr.definitions

XML handling for biomolecular residues.

Authors: Jens Erik Nielsen, Todd Dolinsky, Yong Huang

Classes

Definition(aa_file, na_file, patch_file)	The Definition class contains the structured definitions found in the files and several mappings for easy access to the information.
DefinitionAtom([name, x, y, z])	A trimmed down version of the Atom class
DefinitionHandler()	Handle definition XML file content.
DefinitionResidue()	The DefinitionResidue class extends the Residue class to allow for a trimmed down initializing function.
Patch()	Patch the definitionResidue class

pdb2pqr.forcefield

Force fields are fun.

The forcefield structure is modeled off of the structures.py file, where each forcefield is considered a chain of residues of atoms.

Authors: Todd Dolinsky, Yong Huang

Classes

Forcefield(ff_name, definition, userff[, ...])	Forcefield class
ForcefieldAtom(name, charge, radius, resname)	ForcefieldAtom class
ForcefieldHandler(map_, reference)	Process XML-format topology (force field) files.
ForcefieldResidue(name)	ForcefieldResidue class

2.7.2 Molecular structures

aa	Amino Acid Structures for PDB2PQR
hydrogens	Hydrogen optimization for PDB2PQR
hydrogens.optimize	Hydrogen bond optimization routines.
hydrogens.structures	Topology-related classes for hydrogen optimization.
ligand	Ligand support functions
ligand.mol2	Support molecules in Tripos MOL2 format.
ligand.peoe	Implements the PEOE method described in:
ligand.topology	Ligand topology classes.
na	Nucleic Acid Structures for PDB2PQR
protein	Routines for PDB2PQR
residue	Biomolecular residue class.
structures	Simple biomolecular structures
topology	Parser for TOPOLOGY.xml

pdb2pqr.aa

Amino Acid Structures for PDB2PQR

This module contains the base amino acid structures for pdb2pqr.

Author: Todd Dolinsky

Classes

ALA(atoms, ref)	Alanine class
ARG(atoms, ref)	Arginine class
ASN(atoms, ref)	Asparagine class
ASP(atoms, ref)	Aspartic Acid class
Amino(atoms, ref)	Amino class
CYS(atoms, ref)	Cysteine class
GLN(atoms, ref)	Glutamine class
GLU(atoms, ref)	Glutamic Acid class
GLY(atoms, ref)	Glycine class
HIS(atoms, ref)	Histidine class
ILE(atoms, ref)	Isoleucine class
LEU(atoms, ref)	Leucine class
LIG(atoms, ref)	Generic ligand class
LYS(atoms, ref)	Lysine class
MET(atoms, ref)	Methionine class
PHE(atoms, ref)	Phenylalanine class
PRO(atoms, ref)	Proline class
SER(atoms, ref)	Serine class
THR(atoms, ref)	Threonine class
TRP(atoms, ref)	Tryptophan class
TYR(atoms, ref)	Tyrosine class
VAL(atoms, ref)	Valine class
WAT(atoms, ref)	Water class

pdb2pqr.hydrogens

Hydrogen optimization for PDB2PQR

This is an module for hydrogen optimization routines.

TODO - This module is insane... so many lines!

Authors: Todd Dolinsky, Jens Erik Nielsen, Yong Huang

Functions

create_handler([hyd_path])	Create and populate a hydrogen handler.

Classes

HydrogenRoutines(debumper, handler)	The main routines for hydrogen optimization.

pdb2pqr.hydrogens.optimize

Hydrogen bond optimization routines.

Classes

OptimizationHolder()	A holder class for the XML parser.
Optimize()	The holder class for the hydrogen optimization routines.

pdb2pqr.hydrogens.structures

Topology-related classes for hydrogen optimization.

Classes

Alcoholic(residue, optinstance, routines)	The class for alcoholic residues
Carboxylic(residue, optinstance, routines)	The class for carboxylic residues
Flip(residue, optinstance, routines)	The holder for optimization of flippable residues.
Generic(residue, optinstance, routines)	Generic optimization class
HydrogenAmbiguity(residue, hdef, routines)	A class containing information about the ambiguity
HydrogenConformation(hname, boundatom, ...)	HydrogenConformation class
HydrogenDefinition(name, opttype, optangle, map_)	HydrogenDefinition class
HydrogenHandler()	Extends the SAX XML Parser to parse the Hydrogens.xml class
PotentialBond(atom1, atom2, dist)	A small class containing the hbond structure
Water(residue, optinstance, routines)	The class for water residues

pdb2pqr.ligand

Ligand support functions

Jens Erik Nielsen, University College Dublin 2004

pdb2pqr.ligand.mol2

Support molecules in Tripos MOL2 format.

For further information look at (web page exists: 25 August 2005): http://www.tripos.com/index.php?family=modules,SimplePage,,,&page=sup_mol2&s=0

Classes

Mol2Atom()	MOL2 molecule atoms.
Mol2Bond(atom1, atom2, bond_type[, bond_id])	MOL2 molecule bonds.
Mol2Molecule()	Tripos MOL2 molecule

pdb2pqr.ligand.peoe

Implements the PEOE method described in:

Paul Czodrowski Ingo Dramburg Christoph A. Sotriffer Gerhard Klebe. Development, validation, and application of adapted PEOE charges to estimate pKa values of functional groups in protein–ligand complexes. Proteins, 65,

424-437, 2006. https://doi.org/10.1002/prot.21110

Functions

`assign_terms`(atoms, term_dict)	Assign polynomial terms to each atom.
`electronegativity`(charge, poly_terms, atom_type)	Calculate the electronegativity.
`equilibrate`(atoms[, damp, scale, ...])	Equilibrate the atomic charges.

pdb2pqr.ligand.topology

Ligand topology classes.

Classes

`Topology`(molecule)	Ligand topology class.

pdb2pqr.na

Nucleic Acid Structures for PDB2PQR

This module contains the base nucleic acid structures for pdb2pqr.

Author: Todd Dolinsky

Classes

`ADE`(atoms, ref)	Adenosine class
`CYT`(atoms, ref)	Cytidine class
`DT`(atoms, ref)	Deoxyribo-THY
`GUA`(atoms, ref)	Guanosine class
`Nucleic`(atoms, ref)	This class provides standard features of the nucleic acids listed below.
`RA`(atoms, ref)	Ribo-ADE
`RC`(atoms, ref)	Ribo-CYT
`RG`(atoms, ref)	Ribo-GUA
`RU`(atoms, ref)	Ribo-URA
`THY`(atoms, ref)	Thymine class
`URA`(atoms, ref)	Uridine class

pdb2pqr.protein

Routines for PDB2PQR

This module contains the protein object used in PDB2PQR and associated methods

TODO - this module should be broken into separate files.

Authors: Todd Dolinsky, Yong Huang

Classes

`Protein`(pdblist, definition)	Protein class

pdb2pqr.residue

Biomolecular residue class.

Author: Todd Dolinsky

Classes

`Residue`(atoms)	Residue class

pdb2pqr.structures

Simple biomolecular structures

This module contains the simpler structure objects used in PDB2PQR and their associated methods.

Author: Todd Dolinsky

Classes

`Atom`(atom, type_, residue)	Class Atom
`Chain`(chain_id)	Chain class

pdb2pqr.topology

Parser for TOPOLOGY.xml

Authors: Nathan Baker, Yong Huang

Classes

`Topology`(topology_file)	Contains the structured definitions of residue reference coordinates as well as alternate titration, conformer, and tautomer states.
`TopologyAtom`(parent)	A class for atom topology information
`TopologyConformer`(topology_tautomer)	A class for topology conformer information
`TopologyConformerAdd`(topology_conformer)	A class for adding atoms to a conformer
`TopologyConformerRemove`(topology_conformer)	A class for removing atoms to a conformer
`TopologyDihedral`(parent)	A class for dihedral topology information.
`TopologyHandler`()	Handler for XML-based topology files.
`TopologyReference`(topology_residue)	A class for the reference structure of a residue
`TopologyResidue`(topology_)	A class for residue topology information
`TopologyTautomer`(topology_titration_state)	A class for topology tautomer information
`TopologyTitrationState`(topology_residue)	A class for the titration state of a residue

2.7.3 I/O support

cif	CIF parsing methods
io	Functions related to reading and writing data.
inputgen	Create an APBS input file using psize data
pdb	PDB parsing class

pdb2pqr.cif

CIF parsing methods

This methods use the pdbx/cif parser provided by WWPDB (http://mmcif.wwpdb.org/docs/sw-examples/python/html/index.html)

Author: Juan Brandi

Functions

atom_site(block)	Handle ATOM_SITE block.
author(block)	Handle AUTHOR block.
cispep(block)	Handle CISPEP block
compnd(block)	Handle COMPND block
conect(block)	Handle CONECT block.
count_models(block)	Count models in structure file block
cryst1(block)	Handle CRYST1 block
expdata(block)	Handle EXPDTA block
header(block)	Handle HEADER block
keywds(block)	Handle KEYWDS block
origxn(block)	Handle ORIGXn block
read_cif(cif_file)	Parse CIF-format data into array of Atom objects.
scalen(block)	Handle SCALEn block
source(block)	Handle SOURCE block.
ssbond(block)	Handle SSBOND block
title(block)	Handle TITLE block

pdb2pqr.io

Functions related to reading and writing data.

Functions

dump_apbs(output_pqr, output_path)	Generate and dump APBS input files related to output_pqr.
get_definitions([aa_path, na_path, patch_path])	Get topology definition files.
get_molecule(input_path)	Get molecular structure information.
get_old_header(pdblist)	Get old header from list of PDBs.
get_pdb_file(name)	Obtain a PDB file.
print_pqr_header(pdblist, atomlist, reslist, ...)	Print the header for the PQR file
print_pqr_header_cif(atomlist, reslist, ...)	Print the header for the PQR file in cif format.

Continued on next page

Table 20 – continued from previous page

print_protein_atoms(atomlist[, chainflag, ...])	Get text lines for specified atoms
setup_logger(output_pqr[, level])	Setup the logger to output the log file to the same directory as PQR output.
test_dat_file(name)	Test for the existence of the forcefield file with a few name permutations.
test_for_file(name, type_)	Test for the existence of a file with a few name permutations.
test_names_file(name)	Test for the *.names file that contains the XML mapping.
test_xml_file(name)	Test for the existence of the forcefield file with a few name permutations.

Classes

DuplicateFilter()	Filter duplicate messages.

pdb2pqr.inputgen

Create an APBS input file using psize data

Authors: Todd Dolinsky based on original sed script by Nathan Baker

Functions

build_parser()	Build argument parser.
main()	Main driver
split_input(filename)	Split the parallel input file into multiple async file names

Classes

Elec(pqrpath, size, method, asyncflag[, ...])	An object for the ELEC section of an APBS input file
Input(pqrpath, size, method, asyncflag[, ...])	The input class.

pdb2pqr.pdb

PDB parsing class

This module parses PDBs in accordance to PDB Format Description Version 2.2 (1996); it is not very forgiving. Each class in this module corresponds to a record in the PDB Format Description. Much of the documentation for the classes is taken directly from the above PDB Format Description.

Authors: Todd Dolinsky, Yong Huang

Functions

read_atom(line)	If the ATOM/HETATM is not column-formatted, try to get some information by parsing whitespace from the right.
read_pdb(file_)	Parse PDB-format data into array of Atom objects.
register_line_parser(klass)	Register a line parser in the global dictionary.

Classes

ANISOU(line)	ANISOU class
ATOM(line)	ATOM class
AUTHOR(line)	AUTHOR field
BaseRecord(line)	Base class for all records.
CAVEAT(line)	CAVEAT field
CISPEP(line)	CISPEP field
COMPND(line)	COMPND field
CONECT(line)	CONECT class
CRYST1(line)	CRYST1 class
DBREF(line)	DBREF field
END(line)	END class
ENDMDL(line)	ENDMDL class
EXPDTA(line)	EXPDTA field
FORMUL(line)	FORMUL field
HEADER(line)	HEADER field
HELIX(line)	HELIX field
HET(line)	HET field
HETATM(line[, sybyl_type, l_bonds, ...])	HETATM class
HETNAM(line)	HETNAM field
HETSYN(line)	HETSYN field
HYDBND(line)	HYDBND field
JRNL(line)	JRNL field
KEYWDS(line)	KEYWDS field
LINK(line)	LINK field
MASTER(line)	MASTER class
MODEL(line)	MODEL class
MODRES(line)	MODRES field
MTRIX1(line)	MATRIX1 PDB entry
MTRIX2(line)	MATRIX2 PDB entry
MTRIX3(line)	MATRIX3 PDB entry
MTRIXn(line)	MTRIXn baseclass
NUMMDL(line)	NUMMDL class
OBSLTE(line)	OBSLTE field
ORIGX1(line)	ORIGX3 PDB entry
ORIGX2(line)	ORIGX2 PDB entry
ORIGX3(line)	ORIGX3 PDB entry
ORIGXn(line)	ORIGXn class
REMARK(line)	REMARK field
REVDAT(line)	REVDAT field
SCALE1(line)	SCALE2 PDB entry
SCALE2(line)	SCALE2 PDB entry
SCALE3(line)	SCALE3 PDB entry

Continued on next page

Table 25 – continued from previous page

SCALEn(line)	SCALEn baseclass
SEQADV(line)	SEQADV field
SEQRES(line)	SEQRES field
SHEET(line)	SHEET field
SIGATM(line)	SIGATM class
SIGUIJ(line)	SIGUIJ class
SITE(line)	SITE class
SLTBRG(line)	SLTBRG field
SOURCE(line)	SOURCE field
SPRSDE(line)	SPRSDE field
SSBOND(line)	SSBOND field
TER(line)	TER class
TITLE(line)	TITLE field
TURN(line)	TURN field
TVECT(line)	TVECT class

2.7.4 Other modules that need to be better organized

cells	Cell list to facilitate neighbor searching.
debump	Routines for PDB2PQR
main	Perform functions related to _main_ execution of PDB2PQR.
psize	psize
quatfit	Quatfit routines for PDB2PQR
run	Routines for running the code with a given set of options and PDB files.
utilities	Utilities for PDB2PQR Suite

pdb2pqr.cells

Cell list to facilitate neighbor searching.

Classes

Cells(cellsize)	The cells object provides a better way to search for nearby atoms.

pdb2pqr.debump

Routines for PDB2PQR

This module contains debumping routines to optimize the biomolecule.

Authors: Jens Erik Nielsen, Todd Dolinsky, Yong Huang

Classes

Debump(protein[, definition])	Grab bag of random stuff that apparently didn't fit elsewhere.

pdb2pqr.main

Perform functions related to _main_ execution of PDB2PQR.

This module is intended for functions that directly touch arguments provided at the invocation of PDB2PQR. It was created to avoid cluttering the __init__.py file.

Functions

build_parser()	Build an argument parser.
check_files(args)	Check for other necessary files.
check_options(args)	Sanity check options.
drop_water(pdblist)	Drop waters from a list of PDB records.
is_repairable(protein, has_ligand)	Determine if the protein can be (or needs to be) repaired.
main()	Hook for command-line usage.
main_driver(args)	Main driver for running program from the command line.
non_trivial(args, protein, ligand, ...)	Perform a non-trivial PDB2PQR run.
print_pqr(args, pqr_lines, header_lines, ...)	Print output to specified file
print_splash_screen(args)	Print argument overview and citation information.
run_propka(args, protein)	Run a PROPKA calculation.
setup_molecule(pdblist, definition, ligand_path)	Set up the molecular system.
transform_arguments(args)	Transform arguments with logic not provided by argparse.

pdb2pqr.psize

psize

Get dimensions and other information from a PQR file.

Authors: Dave Sept, Nathan Baker, Todd Dolinksy, Yong Huang

Functions

build_parser()	Build argument parser.
main()	Main driver for module.

Classes

Psize([cfac, fadd, space, gmemfac, ...])	Master class for parsing input files and suggesting settings

pdb2pqr.quatfit

Quatfit routines for PDB2PQR

This module is used to find the coordinates of a new atom based on a reference set of coordinates and a definition set of coordinates.

Original Code by David J. Heisterberg, The Ohio Supercomputer Center, 1224 Kinnear Rd., Columbus, OH 43212-1163, (614)292-6036, djh@osc.edu, djh@ohstpy.bitnet, ohstpy::djh

Translated to C from fitest.f program and interfaced with Xmol program by Jan Labanowski, jkl@osc.edu, jkl@ohstpy.bitnet, ohstpy::jkl

Authors: David Heisterberg, Jan Labanowski, Jens Erik Nielsen, Todd Dolinsky

Functions

`center`(numpoints, refcoords)	Center a molecule using equally weighted points
`find_coordinates`(numpoints, refcoords, ...)	Driver for the quaternion file.
`jacobi`(amat, nrot)	Jacobi diagonalizer with sorted output, only good for 4x4 matrices
`q2mat`(quat)	Generate a left rotation matrix from a normalized quaternion
`qchichange`(initcoords, refcoords, angle)	Change the chiangle of the reference coordinate using the initcoords and the given angle
`qfit`(numpoints, refcoords, defcoords)	Method for getting new atom coordinates from sets of reference and definition coordinates.
`qtransform`(numpoints, defcoords, refcenter, ...)	Transform the set of defcoords using the reference center, the fit center, and a rotation matrix.
`qtrfit`(numpoints, defcoords, refcoords, nrot)	Find the quaternion, q, [and left rotation matrix, u] that minimizes
`rotmol`(numpoints, coor, lrot)	Rotate a molecule
`translate`(numpoints, refcoords, center_, mode)	Translate a molecule using equally weighted points

pdb2pqr.run

Routines for running the code with a given set of options and PDB files.

Functions

`run_pdb2pka`(ph, force_field, pdb_list, ...)	Run PDB2PKA
`run_pdb2pqr`(pdblist, my_protein, ...)	Run the PDB2PQR Suite

pdb2pqr.utilities

Utilities for PDB2PQR Suite

I/O-related utilities/functions should go in io.py

Authors: Todd Dolinsky, Yong Huang

Functions

`add`(coords1, coords2)	Add one 3-dimensional point to another

Continued on next page

Table 34 – continued from previous page

analyze_connectivity(map_, key)	Analyze the connectivity of a given map using the key value.
angle(coords1, coords2, coords3)	Get the angle between three coordinates
cross(coords1, coords2)	Find the cross product of two 3-dimensional points
dihedral(coords1, coords2, coords3, coords4)	Calculate the angle using the four atoms
distance(coords1, coords2)	Calculate the distance between two coordinates, as denoted by
dot(coords1, coords2)	Find the dot product of two 3-dimensional points
factorial(num)	Returns the factorial of the given number n
normalize(coords)	Normalize a set of coordinates
shortest_path(graph, start, end[, path])	Uses recursion to find the shortest path from one node to another in an unweighted graph.
sort_dict_by_value(inputdict)	Sort a dictionary by its values
subtract(coords1, coords2)	Subtract one 3-dimensional point from another

2.8 Release history

2.8.1 PDB2PQR 2.1.1 (2016-03)

New features

- Replaced the Monte Carlo method for generating titration curves with Graph Cut. See http://arxiv.org/1507.07021/

Bug fixes

- Added a check before calculating pKa's for large interaction energies

Known bugs

- If more than one extension is run from the command line and one of the extensions modifies the protein data structure it could affect the output of the other extension. The only included extensions that exhibit this problem are resinter and newresinter.
- Running ligands and PDB2PKA at the same time is not currently supported.
- PDB2PKA currently leaks memory slowly. Small jobs will use about twice the normally required RAM (i.e. ~14 titratable residues will use 140MB). Big jobs will use about 5 times the normally required RAM (60 titratable residues will use 480MB). We are working on this.

2.8.2 PDB2PQR 2.1.0 (2015-12)

New features

- Added alternate method to do visualization using 3dmol.
- Replaced the Monte Carlo method for generating titration curves with Graph Cut. See http://arxiv.org/abs/1507.07021. If you prefer the Monte Carlo Method, please use http://nbcr-222.ucsd.edu/pdb2pqr_2.0.0/

Bug fixes

- Added compile options to allow for arbitrary flags to be added. Helps work around some platforms where scons does not detect the needed settings correctly.
- Fixed broken links on APBS submission page.
- Added some missing files to query status page results.

- Fixed some pages to use the proper CSS file.
- Better error message for `--assign-only` and HIS residues.
- Fixed PROPKA crash for unrecognized residue.
- Debumping routines are now more consistent across platforms. This fixes pdb2pka not giving the same results on different platforms.

Known bugs

- If more than one extension is run from the command line and one of the extensions modifies the protein data structure it could affect the output of the other extension. The only included extensions that exhibit this problem are resinter and newresinter.
- Running ligands and PDB2PKA at the same time is not currently supported.
- PDB2PKA currently leaks memory slowly. Small jobs will use about twice the normally required RAM (i.e. ~14 titratable residues will use 140MB). Big jobs will use about 5 times the normally required RAM (60 titratable residues will use 480MB). We are working on this.

Other comments

- Added fabric script used to build and test releases.
- The newtworkx library is now required for pdb2pka.

2.8.3 PDB2PQR 2.0.0 (2014-12)

New features

- Improved look of web interface.
- Option to automatically drop water from pdb file before processing.
- Integration of PDB2PKA into PDB2PQR as an alternative to PROPKA.
- Support for compiling with VS2008 in Windows.
- Option to build with debug headers.
- PDB2PKA now detects and reports non Henderson-Hasselbalch behavior.
- PDB2PKA can be instructed whether or not to start from scratch with `--pdb2pka-resume`.
- Can now specify output directory for PDB2PKA.
- Improved error regarding backbone in some cases.
- Changed time format on query status page.
- Improved error catching on web interface.

Bug fixes

- Fixed executable name when creating binaries for Unix based operating systems.
- Fixed potential crash when using `--clean` with extensions.
- Fixed MAXATOMS display on server home page.
- PDB2PKA now mostly respects the `--verbose` setting.
- Fixed how hydrogens are added by PDB2PKA for state changes in some cases.
- Fixed psize error check.
- Will now build properly without ligand support if numpy is not installed.

- Removed old automake build files from all test ported to scons.
- Fixed broken opal backend.

Known bugs

- If more than one extension is run from the command line and one of the extensions modifies the protein data structure it could affect the output of the other extension. The only included extensions that exhibit this problem are resinter and newresinter.
- Running ligands and PDB2PKA at the same time is not currently supported.
- PDB2PKA currently leaks memory slowly. Small jobs will use about twice the normally required RAM (i.e. ~14 titratable residues will use 140MB). Big jobs will use about 5 times the normally required RAM (60 titratable residues will use 480MB). We are working on this.

Other comments

- Command line interface to PROPKA changed to accommodate PDB2PKA. PROPKA is now used with `--ph-calc-method=propka --with-ph` now defaults to 7.0 and is only required if a different pH value is required.
- `--ph-calc-method` to select optional method to calculate pH values used to protonate titratable residues. Possible options are "propka" and "pdb2pka".
- Dropped support for compilation with mingw. Building on Windows now requires VS 2008 installed in the default location.
- Updated included Scons to 2.3.3
- PDB2PKA can now be run directly (not integrated in PDB2PQR) with pka.py. Arguments are PDBfile and Output directory.
- No longer providing 32-bit binary build. PDB2PKA support is too memory intensive to make this practical in many cases.

2.8.4 PDB2PQR 1.9 (2014-03)

New features

- Binary builds do not require python or numpy be installed to use. Everything needed to run PDB2PQR is included. Just unpack and use.
- OSX binaries require OSX 10.6 or newer. The OSX binary is 64-bit.
- Linux binaries require CentOS 6 or newer and have been tested on Ubuntu 12.04 LTS and Linux Mint 13. If you are running 64-bit Linux use the 64-bit libraries. In some cases the needed 32-bit system libraries will not be installed on a 64-bit system.
- Windows binaries are 32 bit and were built and tested on Windows 7 64-bit but should work on Windows XP, Vista, and 8 both 32 and 64-bit systems.
- PDB2PQR can now be compiled and run on Windows using MinGW32. See http://mingw.org/ for details.
- PDB2PQR now uses Scons for compilations. With this comes improved automated testing.
- A ligand file with duplicate atoms will cause pdb2pqr to stop instead of issue a warning. Trust us, this is a feature, not a bug!
- Improved error reporting.
- Added support for reference command line option for PROPKA.

- Added newresinter plugin to provide alternate methods for calculating interaction energies between residues.
- Mol2 file handling is now case insensitive with atom names.
- PROPKA with a pH of 7 is now specified by default on the web service.
- Compilation is now done with scons.
- Verbose output now includes information on all patches applied during a run.
- Added stderr and stdout to web error page.
- Added warning to water optimization when other water is ignored.
- Command line used to generate a pqr is now duplicated in the comments of the output.
- Added support for NUMMDL in parser.
- Added complete commandline feature test. Use complete-test target.
- Added propka support for phosphorous sp3. - Thanks to Dr. Stefan Henrich
- Added a PyInstaller spec file. Standalone pdb2pqr builds are now possible.

Bug fixes

- Rolled back change that prevented plugins from interfering with each other. Large proteins would cause a stack overflow when trying to do a deep copy
- Updated INSTALL file to reflect no more need for Fortran.
- Fixed apbs input file to match what web interface produces.
- Fixed user specified mobile ion species not being passed to apbs input file.
- Removed ambiguous A, ADE, C, CYT, G, GUA, T, THY, U, URA as possible residue names.
- Removed eval from pdb parsing routines.
- Updated web links where appropriate.
- Fixed hbond extension output to include insertion code in residue name.
- Fixed debumping routines not including water in their checks. Fixes bad debump of ASN B 20 in 1gm9 when run with pH 7.0.
- Fixed debumping failing to use best angle for a specific dihedral angle when no tested angles are without conflict.
- Fixed debumping using asymmetrical cutoffs and too large cutoffs in many checks involving hydrogen.
- Fixed debumping accumulating rounding error while checking angles.
- Fixed inconsistencies in pdb parsing. - Thanks to Dr. Stefan Henrich
- Fixed problems with propka handling of aromatic carbon/nitrogen. - Thanks to Dr. Stefan Henrich
- Fixed case where certain apbs compile options would break web visualization.
- Fixed improper handling of paths with a '.' or filenames with more than one '.' in them.

Known bugs

- If more than one extension is run from the command line and one of the extensions modifies the protein data structure it could affect the output of the other extension. The only included extensions that exhibit this problem are resinter and newresinter.

Other comments

- Removed numpy from contrib. The user is expected to have numpy installed and available to python at configuration.
- Support for numeric dropped.

2.8.5 PDB2PQR 1.8 (2012-01)

New features

- Updated PROPKA to version 3.0
- Added residue interaction energy extension
- Added protein summary extension
- Combined hbond and hbondwhatit into one extension (hbond) with new command line parameters
- Combined rama, phi, psi into one extension (rama) with new command line parameters.
- Extensions may now add their own command line arguments. Extensions with their own command line arguments will be grouped separately.
- Improved interface for extensions
- Added Opal configuration file.

Bug fixes

- Cleaned up white space in several files and some pydev warnings
- Creating print output no longer clears the chain id data from atoms in the data. (Affected resinter plugin)
- Removed possibility of one plug-in affecting the output of another
- Fixed –protonation=new option for propka30
- Improved time reporting for apbs jobs
- Fixed opal runtime reporting
- Fixed misspelled command line options that prevented the use of PEOEPB and TYL06
- Fixed error handling when certain data files are missing
- Fixed LDFLAGS environment variable not being used along with python specific linker flags to link Algorithms.o and _pMC_mult.so
- Fixed possible Attribute error when applying naming scheme.

2.8.6 PDB2PQR 1.7.1a (2011-09-13)

New features

- Added force field example.

Bug fixes

- Fixed ligand command line option.
- Fixed capitalization of force field in PQR header.
- Fixed error handling for opal errors.
- Fixed web logging error when using ligand files, user force fields, and name files.
- Fixed extension template in documentation.

- Fixed 1a1p example README to reflect command line changes.

2.8.7 PDB2PQR 1.7.1 (2011-08)

New features

- Switched Opal service urls from sccne.wustl.edu to NBCR.
- Added more JMol controls for visualization, JMol code and applets provided by Bob Hanson.
- Changed default forcefield to PARSE in web interface.

Bug fixes

- Fixed crash when opal returns an error.
- Fixed specific combinations of command-line arguments causing `pdb2pqr.py` to crash.
- Fixed opal job failing when filenames have spaces or dashs.
- Fixed gap in backbone causing irrationally placed hydrogens.
- Fixed crash when too many fixes are needed when setting termini.
- Corrected web and command line error handling in many cases.
- Fixed `--username` command line option.
- Fixed ambiguous user created forcefield and name handling. Now `--username` is required if `--userff` is used.
- Fixed `querystatus.py` not redirecting to generated error page.

2.8.8 PDB2PQR 1.7 (2010-10)

- For PDB2PQR web interface users: the JMol web interface for APBS calculation visualization has been substantially improved, thanks to help from Bob Hanson. Those performing APBS calculations via the PDB2PQR web interface now have a much wider range of options for visualizing the output online – as well as downloading for offline analysis.
- For PDB2PQR command-line and custom web interface users: the Opal service URLs have changed to new NBCR addresses. Old services hosted at .wustl.edu addresses have been decommissioned. Please upgrade ASAP to use the new web service. Thank you as always to the staff at NBCR for their continuing support of APBS/PDB2PQR web servers and services.

2.8.9 PDB2PQR 1.6 (2010-04)

New features

- Added Swanson force field based on Swanson et al paper (http://dx.doi.org/10.1021/ct600216k).
- Modified printAtoms() method. Now "TER" is printed at the end of every chain.
- Added Google Analytics code to get the statistics on the production server.
- Modified APBS calculation page layout to hide parameters by default and display PDB ID
- Added "make test-webserver", which tests a long list of PDBs (246 PDBs) on the production PDB2PQR web server.

- Removed `nlev` from `inputgen.py` and `inputgen_pKa.py` as nlev keyword is now deprecated in APBS.
- Added PARSE parameters for RNA, data from: Tang C. L., Alexov E, Pyle A. M., Honig B. Calculation of pKas in RNA: On the Structural Origins and Functional Roles of Protonated Nucleotides. Journal of Molecular Biology 366 (5) 1475-1496, 2007.

Bug fixes

- Fixed a minor bug: when starting `pka.py` from pdb2pka directory using command like "python pka.py [options] inputfile", we need to make sure scriptpath does not end with "/".
- Fixed a bug which caused "coercing to Unicode: need string or buffer, instance found" when submitting PDB2PQR jobs with user-defined force fields on Opal based web server.
- Fixed a bug in `main_cgi.py`, now Opal-based PDB2PQR jobs should also be logged in `usage.txt` file.
- Updated `src/utilities.py` with a bug fix provided by Greg Cipriano, which prevents infinite loops in analyzing connected atoms in certain cases.
- Fixed a bug related to neutraln and/or neutralc selections on the web server.
- Fixed a special case with `--ffout` and 1AIK, where the N-terminus is acetylated.
- Fixed a bug in `psize.py` per Michael Lerner's suggestion. The old version of `psize.py` gives wrong cglen and fglen results in special cases (e.g., all y coordinates are negative values).
- Fixed a bug in `main_cgi.py`, eliminated input/output file name confusions whether a PDB ID or a pdb file is provided on the web server.
- Fixed a bug which causes run time error on the web server when user-defined force field and names files are provided.
- Fixed a bug in `apbs_cgi.py`: pdb file names submitted by users are not always 4 characters long.

2.8.10 PDB2PQR 1.5 (2009-10)

New features

- APBS calculations can be executed through the PDB2PQR web interface in the production version of the server
- APBS-calculated potentials can be visualized via the PDB2PQR web interface thanks to Jmol
- Disabled Typemap output by default, added –typemap flag to create typemap output if needed.
- Enabled "Create APBS Input File" by default on the web server, so that APBS calculation and visualization are more obvious to the users.
- Added warnings to stderr and the REMARK field in the output PQR file regarding multiple occupancy entries in PDB file.
- Added more informative messages in REMARK field, explaining why PDB2PQR was unable to assign charges to certain atoms.
- Updated structures.py, now PDB2PQR keeps the insertion codes from PDB files.
- Added "make test-long", which runs PDB2PQR on a long list (246) of PDBs by default, it is also possible to let it run on specified number of PDBs, e.g., export TESTNUM=50; make test-long
- Updated NBCR opal service urls from http://ws.nbcr.net/opal/... to http://ws.nbcr.net/opal2/...

- Compressed APBS OpenDX output files in zip format, so that users can download zip files from the web server.
- Removed "EXPERIMENTAL" from APBS web solver interface and Jmol visualization interface.
- Updated all APBS related urls from http://apbs.sourceforge.net/... to http:/apbs.wustl.edu/...
- Merged PDB2PKA code, PDB2PKA is functional now.
- Added two new options: –neutraln and –neutralc, so that users can manually make the N-termini or C-termini of their proteins neutral.
- Added a local-test, which addresses the issue of Debian-like Linux distros not allowing fetching PDBs from the web.
- Added deprotonated Arginine form for post-PROPKA routines. This only works for PARSE forcefield as other forcefields lack deprotonated ARG parameters.
- Updated inputgen.py with –potdx and –istrng options added, original modification code provided by Miguel Ortiz-Lombardía.
- Changed default Opal service from http://ws.nbcr.net/opal2/services/pdb2pqr_1.4.0 to http://sccne.wustl.edu:8082/opal2/services/pdb2pqr-1.5

Bug fixes

- Verbosity outputs should be stdouts, not stderrs in web server interface. Corrected this in src/routines.py.
- Fixed a bug in psize.py: for a pqr file with no ATOM entries but only HETATM entries in it, inputgen.py should still create an APBS input file with reasonable grid lengths.
- Added special handling for special mol2 formats (unwanted white spaces or blank lines in ATOM or BOND records).
- Added template file to doc directory, which fixed a broken link in programmer guide.

2.8.11 PDB2PQR 1.4.0 (2009-03)

New features

- Updated html/master-index.html, deleted html/index.php.
- Updated pydoc by running genpydoc.sh.
- Added a whitespace option by by putting whitespaces between atom name and residue name, between x and y, and between y and z.
- Added radius for Chlorine in ligff.py.
- Added PEOEPB forcefield, data provided by Paul Czodrowski.
- Updated inputgen.py to write out the electrostatic potential for APBS input file.
- Updated CHARMM.DAT with two sets of phosphoserine parameters.
- Allowed amino acid chains with only one residue, using –assign-only option.
- Updated server.py.in so that the ligand option is also recorded in usage.txt.
- Updated HE21, HE22 coordinates in GLN according to the results from AMBER Leap program.
- Updated Makefile.am with Manuel Prinz's patch (removed distclean2 and appended its contents to distclean-local).
- Updated configure.ac, pdb2pqr-opal.py; added AppService_client.py and AppService_types.py with Samir Unni's changes, which fixed earlier problems in invoking Opal services.

- Applied two patches from Manuel Prinz to pdb2pka/pMC_mult.h and pdb2pka/ligand_topology.py.
- Updated PARSE.DAT with the source of parameters.
- Created a contrib folder with numpy-1.1.0 package. PDB2PQR will install numpy by default unless any of the following conditions is met:
- Working version of NumPy dectected by autoconf.
- User requests no installation with –disable-pdb2pka option.
- User specifies external NumPy installation.
- Merged Samir Unni's branch. Now PDB2PQR Opal and APBS Opal services are available (through –with-opal and/or –with-apbs, –with-apbs-opal options at configure stage).
- Added error handling for residue name longer than 4 characters.
- Updated hbond.py with Mike Bradley's definitions for ANGLE_CUTOFF and DIST_CUTOFF by default.
- Removed PyXML-0.8.4, which is not required for ZSI installation.
- Updated propka error message for make adv-test – propka requires a version of Fortran compiler.
- Updated na.py and PATCHES.xml so that PDB2PQR handles three lettered RNA residue names (ADE, CYT, GUA, THY, and URA) as well.
- Updated NA.xml with HO2' added as an alternative name for H2'', and H5" added as an alternative name for H5''.
- Updated version numbers in html/ and doc/pydoc/ .
- Updated web server. When selecting user-defined forcefield file from the web server, users should also provide .names file.
- Removed http://enzyme.ucd.ie/Services/pdb2pqr/ from web server list.
- Eliminated the need for protein when processing other types (ligands, nucleic acids).
- Updated psize.py with Robert Konecny's patch to fix inconsistent assignment of fine grid numbers in some (very) rare cases.
- Made whitespace option available for both command line and web server versions.
- Updated inputgen_pKa.py with the latest version.

Bug fixes

- Fixed a legacy bug with the web server (web server doesn't like ligand files generated on Windows or old Mac OS platforms).
- Fixed a bug in configure.ac, so that PDB2PQR no longer checks for Numpy.pth at configure stage.
- Updated pdb2pka/substruct/Makefile.am.
- Fixed isBackbone bug in definitions.py.
- Fixed a bug for Carboxylic residues in hydrogens.py.
- Fixed a bug in routines.py, which caused hydrogens added in LEU and ILE in eclipsed conformation rather than staggered.
- Fixed a bug in configure.ac, now it is OK to configure with double slashes in the prefix path, e.g., –prefix=/foo/bar//another/path
- Fixed a bug in nucleic acid naming scheme.
- Fixed a bug involving MET, GLY as NTERM, CTERM with –ffout option.

- Fixed a bug for PRO as C-terminus with PARSE forcefield.
- Fixed a bug for ND1 in HIS as hacceptor.
- Fixed the –clean option bug.
- Fixed a bug in CHARMM naming scheme.
- Fixed a bug in test.cpp of the simple test (which is related to recent modifications of 1AFS in Protein Data Bank).

2.8.12 PDB2PQR 1.3.0 (2008-01)

New features

- Added "make test" and "make adv-test"
- Fixed problems with "make dist"
- Added integration with Opal for launching jobs as well as querying status
- The user may use NUMPY to specify the location of NUMPY.
- Both PDB2PKA and PROPKA are enabled by default. PDB2PKA is enabled by default since ligand parameterization would fail without this option.
- For a regular user, "make install" tells the user the exact command the system administrator will use to make the URL viewable.
- The default value of 7.00 for the pH on the server form is removed due to a problem with browser refershing.
- Updated warning messages for lines beginning with SITE, TURN, SSBOND and LINK.
- Switched license from GPL to BSD.
- Made a new tar ball pdb2pqr-1.3.0-1.tar.gz for Windows users who cannot create pdb2pqr.py through configure process.
- configure now automatically detects SRCPATH, WEBSITE, and the location of pdb2pqr.cgi. In version 1.2.1, LOCALPATH(SRCPATH) and WEBSITE were defined in src/server.py and the location of pdb2pqr.cgi was specified in html/server.html (index.html). Configure now uses variable substitution with new files src/server.py.in and html/server.html.in to create src/server.py and html/server.html (index.html).
- SRCPATH is automatically set to the current working directory. WEBSITE is automatically set to http://fully_qualified_domain_name/pdb2pqr. Path to CGI is automcailly set to http://fully_qualified_domain_name/pdb2pqr/pdb2pqr.cgi.
- In version 1.2.1, there were 3 variables that needed to be changed to set up a server at a location different from agave.wustl.edu. LOCALPATH, WEBSITE, and the location of the CGI file. In this version, LOCALPATH has been used to SRCPATH to avoid confusion, since LOCALPATH could be interpreted as the local path for source files or the localpath for the server.
- Since configure now automatically sets the locations of files/directories based on the machine and configure options, the default agave.wustl.edu locations are not used anymore.
- A copy of pdb2pqr.css is included.
- configure prints out information about parameters such as python flags, srcpath, localpath, website, etc.
- configure now automatically creates tmp/ with r + w + x permissions.
- configure now automatically copies pdb2pqr.py to pdb2pqr.cgi.

- configure now automatically copies html/server.html to index.html after variable substitution. In src/server.py.in (src/server.py), WEBNAME is changed to index.html.
- ${HOME}/pdb2pqr is the default prefix for a regular user
- /var/www/html is the default prefix for root
- http://FQDN/pdb2pqr as default website.
- "make install" runs "make" first, and the copies the approprite files to –prefix.
- If root did not specify –prefix and /var/www/html/pdb2pqr already
- exists, then a warning is issued, and the user may choose to quit or overwrite that directory.
- Similary, if a regular user did not specify –prefix and ${HOME}/pdb2pqr already exists, then a warning is issued, and the user may choose to quit or overwrite that directory.
- If root does not specify –prefix to be a directory to be inside /var/www/html (for example, –prefix=/share/apps/pdb2pqr), then a symbolic link will be made to /var/www/html/pdb2pqr during "make install".
- configure option –with-url can be specified either as something like http://sandstone.ucsd.edu/pdb2pqr-test or sandstone.ucsd.edu/pdb2pqr-test. It also doesn't matter if there's a '/' at the end.
- If user is root, and the last part of URL and prefix are different, for example, –with-url=athena.nbcr.net/test0 –prefix=/var/www/html/pdb2pqr-test, then a warning will be issued saying the server will be viewable from the URL specified, but not the URL based on pdb2pqr-test. In other words, the server will be viewable from athena.abcr.net/test0, but not athena.nbcr.net/pdb2pqr-test. During "make install", a symbolic link is created to enable users to view the server from –with-url.
- When making a symbolic link for root, if then link destination already exists as a directory or a symoblic link, then the user may choose to continue with creating the link and overwrite the original directory or quit.
- If the user changes py_path when running configure for PDB2PQR, then the change also applies to PROPKA.

Bug fixes

- Fixed the line feed bug. Now PDB2PQR handles different input files (.pdb and .mol2) created or saved on different platforms.
- Fixed "hbondwhatif" warning at start up.

Known issues

- The install directory name cannot contain dots.
- For python 2.2, if PDB2PQR cannot find module sets, then sets needs to be copied from . . . /python2.2/site-packages/MYSQLdb/sets.py to . . . /lib/python2.2

2.8.13 PDB2PQR 1.2.1 (2007-04)

New features

- Updated documentation to include instructions for pdb2pka support, references, more pydoc documents.
- Added ligand examples to examples/ directory
- Added native support for the TYL06 forcefield. For more information on this forcefield please see Tan C, Yang L, Luo R. How well does Poisson-Boltzmann implicit solvent agree with explicit solvent? A quantitative analysis. Journal of Physical Chemistry B. 110 (37), 18680-7, 2006.

- Added a new HTML output page which relays the different atom types between the AMBER and CHARMM forcefields for a generated PQR file (thanks to the anonymous reviewers of the latest PDB2PQR paper).

Bug fixes

- Fixed bug where a segmentation fault would occur in PropKa if the N atom was not the first atom listed in the residue
- Fixed error message that occurred when a blank line was found in a parameter file.
- Better error handling in MOL2 file parsing.
- Fixed bug where ligands were not supported on PDB files with multiple MODEL fields.

2.8.14 PDB2PQR 1.2.0 (2007-01)

New features

- Added autoconf support for pdb2pka directory.
- Added new support for passing in a single ligand residue in MOL2 format via the –ligand command. Also available from the web server (with link to PRODRG for unsupported ligands).
- Numerous additions to examples directory (see examples/index.html) and update to User Guide.

Bug fixes

- Fixed charge assignment error when dealing with LYN in AMBER.
- Fixed crash when a chain has a single amino acid residue. The code now reports the offending chain and residue before exiting.
- Fixed hydrogen optimization bug where waters with no nearby atoms at certain orientations caused missing hydrogens.

2.8.15 PDB2PQR 1.1.2 (2006-06)

Bug fixes

- Fixed a bug in the hydrogen bonding routines where PDB2PQR attempted to delete an atom that had already been deleted. (thanks to Rachel Burdge)
- Fixed a bug in chain detection routines where PDB2PQR was unable to detect multiple chains inside a single unnamed chain (thanks to Rachel Burdge)
- Fixed a second bug in chain detection routines where HETATM residues with names ending in "3" were improperly chosen for termini (thanks to Reut Abramovich)
- Fixed a bug where chains were improperly detected when only containing one HETATM residue (thanks to Reut Abramovich)

2.8.16 PDB2PQR 1.1.1 (2006-05)

Bug fixes

- Fixed a bug which prevented PDB2PQR from recognizing atoms from nucleic acids with "*" in their atom names. (thanks to Jaichen Wang)

- Fixed a bug in the hydrogen bonding routines where a misnamed object led to a crash for very specific cases. (thanks to Josh Swamidass)

2.8.17 PDB2PQR 1.1.0 (2006-04)

New features

- Structural data files have been moved to XML format. This should make it easier for users and developers to contribute to the project.
- Added an extensions directory for small scripts. Scripts in this directory will be automatically loaded into PDB2PQR has command line options for post-processing, and can be easily customized.
- Code has been greatly cleaned so as to minimize values hard-coded into functions and to allow greater customizability via external XML files. This includes a more object-oriented hierarchy of structures.
- Improved detection of the termini of chains.
- Assign-only now does just that - only assigns parameters to atoms without additions, debumping, or optimizations.
- Added a –clean command line option which does no additions, optimizations, or forcefield assignment, but simply aligns the PDB columns on output. Useful for using post-processing scripts like those in the extensions directory without modifying the original input file.
- The –userff flag has been replaced by opening up the –ff option to user-defined files.
- Pydoc documentation is now included in html/pydoc.
- A programmer's guide has been included to explain programming decisions and ease future development.
- A –ffout flag has been added to allow users to output a PQR file in the naming scheme of the desired forcefield.
- User guide FAQ updated.
- The efficiency of the hydrogen bonding detection script (–hbond) has been greatly improved.
- Increased the number of options available to users via the PDB2PQR web server.

Bug fixes

- Updated psize.py to use centers and radii when calculating grid sizes (thanks to John Mongan)
- Fixed bug where PDB2PQR could not read PropKa results from chains with more than 1000 residues (thanks to Michael Widmann)

2.8.18 PDB2PQR 1.0.2 (2005-12)

New features

- Added ability for users to add their own forcefield files. This should be particularly useful for HETATMs.
- Added sdens keyword to inputgen.py to make PDB2PQR compatibile with APBS 0.4.0.
- Added a new examples directory with a basic runthrough on how to use the various features in PDB2PQR.

Bug fixes

- Fixed a bug that was unable to handle N-Terminal PRO residues with hydrogens already present.
- Fixed two instances in the PropKa routines where warnings were improperly handled due to a misspelling.
- Fixed instance where chain IDs were unable to be assigned to proteins with more than 26 chains.

2.8.19 PDB2PQR 1.0.1 (2005-10)

New features

- Added citation information to PQR output.

Bug fixes

- Fixed a bug during hydrogen optimization that left out H2 from water if the oxygen in question had already made 3 hydrogen bonds.

2.8.20 PDB2PQR 1.0.0 (2005-08)

This is the initial version of the PDB2PQR conversion utility. There are several changes to the various "non-official" versions previously available:

- SourceForge has been chosen as a centralized location for all things related to PDB2PQR, including downloads, mailing lists, and bug reports.
- Several additions to the code have been made, including pKa support via PropKa, a new hydrogen optimization algorithm which should increase both accuracy and speed, and general bug fixes.

2.9 Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

P