This is version 2.2 of MEAD, Macroscopic Electrostatics with Atomic Detail. This README file is written by Donald Bashford.

Please send me an email message at don.bashford@stjude.org if you intend to use MEAD. This helps to get support for MEAD, and also provides a way for you to be notified of new versions of MEAD. If you use MEAD in research for publication, you can cite the following papers:

D. Bashford & K. Gerwert (1992) "Electrostatic Calculations of the pKa Values of Ionizable Groups in Bacteriorhodopsin" J. Mol. Biol. vol. 224 pp. 473-486

Donald Bashford. "An Object-Oriented Programming Suite for Electrostatic Effects in Biological Molecules" In Yutaka Ishikawa, Rodney R. Oldehoeft, John V. W. Reynders, and Marydell Tholburn, editors, "Scientific Computing in Object-Oriented Parallel Environments", volume 1343 of Lecture Notes in Computer Science, pages 233--240, Berlin, 1997. ISCOPE97, Springer.

Bug reports are welcome. Send them to don.bashford@stjude.org

The main documentation for MEAD is this README file, whose main purpose is to explain what's in the distribution and how to run the programs. For a breif outline of the design of MEAD, see the ISCOPE97 paper cited above. (It's on the FTP site too as iscope-paper.ps)

Please look at the file INSTALLATION even if you have installed a previous version of MEAD before. We have switched to the use of a configure script, rather the the multiple-makefile scheme used before. Also, the organization of both the source files and the installed files has changed.

INTRODUCTION

MEAD is a set of software objects for the purpose of modeling the electrostatics of molecules using a semi-macroscopic picture in which the solvent and the molecular interior have different dielectric constants, the boundary between the different dielectric regions is dependent on the detailed atomic structure of the molecule, and the electrostatic potential is determined by the Poisson-Boltzmann equation. This version of MEAD includes modeling of a membrane as a low dielectric slab, possibly with a water-filled channel through a protein in the membrane.

MEAD is written in C++, which is a significant departure from most molecular software, which is more commonly written in Fortran or (recently) C. My purpose in choosing C++ was to explore the object-oriented programming style that C++ facilitates and to make a software system that other people could borrow pieces from and make extensions to in a convenient way.

along with this.

> This program is distributed in the hope that it will be useful,
> but WITHOUT ANY WARRANTY; without even the implied warranty of
> MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
> GNU General Public License for more details.

See the INSTALLATION file for a list of machines MEAD is known to run on and instructions on compiling MEAD. Also see the PROBLEMS file for some discussion of known problems on certain machines.

REPRODUCTION OF PUBLISHED RESULTS

This distribution of MEAD has the programs, data files and parameter files needed to reproduce the results reported in several publications. Files for the pK calculations on the triclinic structure of lysozyme reported in D. Bashford & M. Karplus (1990) Biochemistry vol. 29, pp. 10219-25 are in the subdirectory, examples/lysozyme.
Files for some of the results reported in D. Bashford & K. Gerwert (1992) "Electrostatic Calculations of the pKa Values of Ionizable Groups in Bacteriorhodopsin" J. Mol. Biol. vol. 224 pp. 473-486, are in the subdirectory, examples/br. Data files to reproduce some of the results from Donald Bashford, David A. Case, Claudio Dalvit, Linda Tennant and Peter E. Wright (1993) "Electrostatic Calculations of Side-chain pKa Values in Myoglobin and Comparison with NMR Data for Histidines", Biochemistry vol. 23, pp. 8045--8056, are in the directory examples/myoglobin. Some of the test results for the method of calculated solvent accessibility lattices reported in You and Bashford (1995b), J. Comp. Chem. vol. 16, pp. 743-757 can be reproduced using the data files in the subdirectory examples/solvacc.

The input data to reproduce the results for a conformationally flexible site in lysozyme reported in You and Bashford (1995a), Biophys. J. vol 69, pp. 1721-1733, are in a separate distribution file, lysoflex.tar.Z, on the Scripps FTP server. Similarly the data files for the results reported in Dillet, Dyson and Bashford (1998) "Calculations of Electrostatic Interactions and pKa's in the Active Site of Escherichia coli Thioredoxin" Biochemsitry vol 37, pp. 10298-10306. are in thioredoxin.tar.gz on the FTP server. Both these data sets are quite large since they contain a number of protein coordinate files.

The computer programs needed to reproduce the results mentioned above are mostly under apps, although in some cases, problem-specific shell scripts or perl scripts are in the data-file directories. For some of these studies, a version of Paul Beroza's mcti program was used in a final step. Mcti is not distributed by the Bashford lab, although the Scripps FTP currently server has a version of it under pub/case/beroza/pep/xmcti-1.1.1.tar.Z. There may be slight differences between the version of mcti used in house and the one currently on the FTP server.

ORGANIZATION OF MEAD SOURCES: MEAD AS A LIBRARY:

A long-standing goal of MEAD has been to serve not only as a collection of useful applications (documented below) but also as an

object-oriented library to allow other researchers to create their own applications. Starting with version 2.2.0, this is achieved by reorginizing the source code directories, and by having an install procedure that installs MEAD as a library.

Building MEAD causes a static library file libmead.a to be created and installation puts it in a place like /usr/local/lib. Also the various header (*.h) files that are needed to compile a program using the MEAD library are put in a place like /usr/local/include/MEAD by the installation procedure. Applcation programmers then need not put their programs into the distributed source directory. Rather, they can put them in their own directory and put lines like

```
   #include <MEAD/ElstatPot.h>
```

and so on, in their source code, and compile using whatever flags are needed to tell the compiler where to find include files (usually "-I" on Unix compilers); and link using the appropiate flags to find libmead (on Unix something like "-L/usr/local/lib -lmead").

In the distributed source directory, the files needed to make libmead.a are in the directory libmead (which might be renamed or symlinked to MEAD in some situations). Applications programs are in individual subdirectories under apps. The swig subdirectory pertains to the new Python interface (see below). This business of putting each libarary and application into its own subdirectory seems to be necessary for many implementations of C++ if either the libraries or the applications make much use of templates. It also seems to help with portability to Windows.

Unfortunately, there is not yet documentation for MEAD as a library. Your best bet is to look at the source code of some of the simpler applications like potental or solvate or (getting a bit more complex) solinprot. Don't start with multiflex, it's too hairy.

PYTHON INTERFACE:

Recently, we have begun the process of "wrapping" MEAD to make it available from the Python scripting language. We have tried to make the Python interface as closely analogous as possilbe to the C++ interface. Only a portion of the higher-level parts of the MEAD library are wrapped at present, and documentation is lacking. For more information about the Python wrapping, see the directory, swig, and the README file therein. The python wrapping is currently optional in the building process, see INSTALLATION for more.

We don't yet have documentation for the Python interface, but you can get an idea by looking at some simple examples in that directory, sol.py genpot.py and bug.py. To see a full list of the classes and functions available (but with no documentation strings), see MEAD.py.

To run the simply python examples, type "python genpot.py", and so on. This will only work if "make install" has installed the python interface to MEAD (see INSTALLATION). If you're just sitting in the swig directory, but haven't done the install, you'll need to edit the application files by and change the lines "from PyMead import MEAD" to "import MEAD"

NOTE: bug.py will cause a Python run-time error to be signalled. That's what it's supposed to do! This means that errors dedected within the MEAD C++ code are get translated into python exceptions which scripts can catch and handle, rather than bringing down the whole python system.

The README file in the swig subdirectory has a description of the methods used to wrap MEAD for Python. The Python interface is largely the work of Thi Thien Nguyen.

This distribution includes five programs implemented using the MEAD object library: potential solvate, solinprot, multiflex and sav-driver. It also includes the program redti, which does not use the MEAD objects. The next five sections are fairly brief descriptions of the purposes and particular features of the five MEAD programs. Detailed discussion of most command line options and file formats is deferred to the sections, "THE COMMAND LINE" and "FILE FORMATS", because many of the programs share many of the same options and formats. Finally the program, REDTI is described.

POTENTIAL

Calculates the potential due to the molecule, MolName (a name specified on the command line), whose coordinates radii, and charges are specified in a file MolName.pqr, and writes to standard output its values at points specified by the MolName.fpt file. The units of the output potentials are input charge units divided by input length units. For typical PDB-derived input files, this would be proton-charge/Angstrom. In that case, to get kcal/mol/proton-charge, multiply by 331.842.

The -CoarseFieldOut and -CoarsefieldInit options will cause the coarsest potential lattice to be written to, or initialized from, an AVS (Advanced Visualization System) "field" file. By default, the units are as above for the the output potentials, but if you give the option "-AvsScaleFactor f", where f is a floating point number, the field will be scaled by that factor.

Potential requires the files MolName.pqr and MolName.ogm. MolName.fpt is not strictly required, but if neither the .fpt file nor the -CoarseFieldOut option are used, the program produces no output of the calculated potentials. The -epsin flag is mandatory.

See below for general explanations of files, arguments and flags.

SOLVATE

Solvate calculates the Born solvation energy of a molecule -- that is, the difference in the electrostatic work required to bring its atom charges from zero to their full values in solvent versus vacuum. MolName (a name specified on the command line) is the molecule(s) for which the the calculation is to be done and whose coordinates radii and charges are specified in a file MolName.pqr Solvate requires a MolName.pqr file and a MolName.ogm file (see below) as inputs. THE -epsin FLAG IS MANDATORY. This is a change from older version. I

found that people got confused when the default epsin (previously 4.0) was contrary to people's expectations; so no more default behavior! The Born solvation energy is written to standard output in kcal/mole. Physical conditions and units for I/O can be set by flags on the command line (see below). By default solvate assumes we are going from vacuum (eps=1) to water (eps=80). Note that solvate uses the -epssol and -epsvac flags rather than the -epsext flag to control solvent conditions. You can try it out on a sphere to check agreement with the Born formula. See the born subdirectory.

A NEW FEATURE of solvate is turned on with the -ReactionField flag. In this case, a MolName.fpt file is expected to contain points at which you want the value of the reaction field. It will be written out to the file, MolName.rf.

SOLINPROT

USAGE: solinprot [flags] solute protein

Calculates the "solvation" of the molecule named by solute (for which there must be solute.pqr and solute.ogm files) inside the molecule named by protein (for which the must be a protein.pqr file) which is, in turn, in some solvent (water, by default). The interior of the solute is presumed to have the dielectric constant, epsin1 (given by the -epsin1 flag) and regions interior to the protein but exterior to the solute are presumed to have the dielectric constant, epsin2 (given by the -epsin2 flag) and regions exterior to both protein and solute are presumed to have dielectric constant, epsext (80.0 by default). The protein may contain charges and their interaction with the solute will contribute to "solvation energy." The solvated energy is calculated relative to a vacuum calculation in which the dielectric constant has a value of epsin1 inside the solute and epsvac (1.0 by default) outside.

The calculation works like this: First the potential due to the solute charges (call them rho_solute) in the above described dielectric environment is calculated. Call this potential phi_sol. Next the potential due to rho_solute is calculated in the vacuum dielectric environment described above. Call this potential phi_vac. The reaction field component of the solvation energy is then, (rho_solute*phi_sol - rho_solute*phi_vac)/2, where "*" indicates a suitable sum or integral of charge times potential. So far, this is the same as the solvate program except for the three-dielectric environment on the solvated side. We also need the contribution due to protein charges, rho_protein, interacting with the solute: rho_protein*phi_sol.

The flags are similar to those for the solvate program except that the -epsin1 and -epsin2 flags (see below) are mandatory and the -epsin flag is forbidden. The -epsext flag is used instead of -epssol for specifying the solvent dielectric. (Is this a bug?). The -ProteinField flag, described below, is also available.

MULTIFLEX

Multiflex does the electrostatic part of a titration calculation for a

multi-site titrating molecule. It can do single-conformer calculations based on the methods described in Karplus and Bashford (1990) and Bashford and Gerwert (1992) which assumes a rigid molecule, or it can included limited conformational flexibility by the method of You and Bashford (1995a). In the latter case, the user must supply the coordinates for the conformational variants and the corresponding non-electrostatic energies. This can be done with a program like CHARMM.

For single-conformer calculation, multiflex works like the old multimead program. It takes MolName.pqr, MolName.ogm, MolName.mgm, MolName.sites and MolName.st files as inputs (see below) and as its main outputs, produces a MolName.pkint file, which contains the calculated intrinsic pK's; a MolName.g file, which contains site-site interactions in units of charge squared per length; and a MolName.summ file which summarizes the self and background contributions to the intrinsic pK of each site. The MolName.pkint file and the Molname.g file can be used directly as input to redti, the program for calculating titration curves. Multiflex also produces a file, MolName.sitename.summ for each titrating site which contains some summary information that is mainly interesting for multi-conformer (flexible) calculations; and it produces a large number of *.potat files, which are binary files that are useful if a job is interrupted and restarted (see below). The -epsin flag is mandatory. Other flags can be used to change units and/or physical conditions or include a membrane.

For the "flexible" calculations which involve multiple conformers, multiflex needs the input files described above but it also needs additional files: For each flexible site there must be a file, MolName.sitename.confs, in which "sitename" is constructed from the MolName.sites file by the procedure,

```
  "7  GLU"  --> "GLU-7"
```

These .confs files tell about the possible conformers of that site. They contain lines of the form:

```
  confname  mac_non_elstat_energy  mod_non_elstat_energy
```

where "confname" is the name of the conformer and the next two entries are non-electrostatic energies of the conformer in the macromolecule and in the model compound corresponding to this site. The energies must be given in kcals/mole unless the -econv flag (see below) has been given to change the energy units.

For each conformer named in a .confs file, there must be a .pqr file having the coordinates, charges and radii for the whole protein in that conformer. It must have the file name:

```
   MolName.sitename.confname.pqr
```

You might need a lot of of .pqr files, for example, the You and Bashford calculations on lysozyme had 36 conformers for each of 12 sites and one MolName.pqr file is always needed so 433 .pqr files were needed for each titration calculation.

Flexible and single-conformer sites can co-exist in the same molecule. The way multiflex tells the difference is that flexible sites have confs files and single-conformer files don't.

SAV-DRIVER

This program gives a demonstration of the SolvAccVol facility of MEAD which was described in our paper, You and Bashford (1995b), J. Comp. Chem. vol. 16, pp. 743-757. It is described in more detail in the file, README.SAV

THE COMMAND LINE:

The programs generally have the command line syntax:

program_name [flag value]... [flag]... MolName

where MolName is the base name for various I/O files. For example, "multiflex [flags] foo" will expect to find files named foo.pqr, foo.ogm, foo.mgm, and foo.sites, and will create foo.pkint, foo.g and foo.summ, as well as writing to standard output. The flags are used to set values pertaining to the physical conditions being modeled and the units used in the input files and the other flags. The default assumptions regarding units are that all inputs with dimension of length (coordinates, grid spacing) are in Angstroms, all inputs with dimensions of charge are in protons, concentrations are in moles/liter and temperature is in Kelvins.

The flags and their default values are:

-epsin f Dielectric constant of molecular interior
In old versions of MEAD this had a default value, but this caused confusion, so now you must specify epsin explicitly.

-epsext f 80.0 Exterior dielectric constant (potential only).

-epssol f 80.0 The dielectric constant of one of the external media in the solvate program, presumably the solvent. (multiflex and solvate)

-epsvac f 1.0 The dielectric constant of the other external media in the solvate program, presumably the vacuum. (solvate only)

-solrad f 1.4 Solvent probe radius used in rolling ball procedure to determine contact surface which is taken as the boundary between epsin and epsext. (Angstroms, by default)

-sterln f 2.0 Ion exclusion layer thickness which is added to atomic radii to determine region inaccessible to salt so that kappa term in PB equation is zero. (Angstroms, by default)

-ionicstr f 0.0 Ionic strength (moles/liter, by default).

-T f 300.0 Temperature (Kelvins by default)

-ReactionField (solvate only) Flag to write values of the solvent reaction field potential at space points specified by the user. The point should be in an input file named MolName.fpt in which points are listed in the format, "(x, y, z)" Corresponding reaction field potential values will be written to a file MolName.rf.

-epsave_oldway Revert to the old style of "epsilon averaging" This refers to the problem of deciding what value of the dielectric to use for the region between two potential lattice points in the finite-difference method. The new way involves inverse averaging and is similar to a proposal by McCammon. The old way is a simple mean. The new way is significantly more accurate; the option to do it the old way is only provided for the sake of reproducing old experimental results. It is not recommended otherwise.

-converge_oldway Revert to the old way of testing for convergence of the SOR method of solving the finite-difference representation of the Poisson--Boltzmann equation. The new method gives improved long-range accuracy for large lattices, but at a sometimes substantial computational cost. See the NEWS file for further discussion.

-kBolt f 5.984e-6 (protons squared)/(Angstrom * Kelvin)
The Boltzmann constant in units of charge unit squared per length unit per temperature unit. You must adjust this if you don't use the default units of length, charge and temperature

-econv f 331.842 (kcal/mole)/(protons squared/Angstrom)
Conversion factor for going from energy units of (charge units squared)/(length unit) to to whatever energy units you want for your output. You must adjust this if you don't use the default units of length and charge, or if you want output in different energy units.

-conconv f 6.022214e-4 ((particles/cubic Angstrom)/(moles/liter))
Conversion factor for going from concentration units used for ionic strength to units of particles per cubic length unit. You must adjust this if you don't use the default units of length and concentration.

-site N Do a titration calculation only for the N-th site that is named in the .sites file. (multiflex only).

-CoarseFieldOut nm For the first (coarsest) lattice specified in the .ogm file, write the lattice potential values to a file, nm.fld, which is an AVS "field" format. (potential only)

-CoarseFieldInit nm For the first (coarsest) lattice specified in the .ogm file, read the lattice potential values from the AVS file, nm.fld, and use them to initialize the coarse lattice. (potential only)

-nopotats Prevent multiflex from creating any .potat files. However, multiflex will still make use of .potat files found in the current directory (see discussion of .potat files below)

-membz f1 f2 Set up a membrane parallel to the x-y plane extending from z=f1 to z=f2. (multiflex only). The "membrane" is a low-dielectric slab: all points within the specified z range are assigned the dielectric constant, epsin. (multiflex only)

-membhole f1 [f2 f3] Make a cylindrical "hole" through the membrane with radius f1 and central axis in the z direction x=f2 and y=f3 (or x=y=0 if f2 and f3 are not given). The meaning of the hole is that points that are inside the hole but outside the protein interior are assigned a dielectric constant of epssol. This is useful for calculations on membrane proteins that make channels or have water-filled cavities, such as bacteriorhodopsin. (multiflex only)

-blab1
-blab2
-blab3 Controls how verbose the program will be. -blab3 is most verbose, and no blab flag at all is least verbose. In the latter case only essential info on the run parameters and results are printed to standard output. Writing to things like .pkint files are not effected.

FILE FORMATS

MolName.pqr :

Atomic coordinate file similar to a PDB (Protein Data Bank) file but with atomic charge and radii in the occupancy and B-factor columns, respectively. More specifically, lines beginning with either "ATOM" or "HETATM" (no leading spaces) are interpreted as a set of tokens separated by one or more spaces or TAB characters. The tokens (including the leading ATOM or HETATM are interpreted as follows:

ignored ignored atname resname resnum x y z charge radius chainid

The chainid token is optional, and any tokens beyond that are

ignored. Tokens can be of arbitrary length, but must not contain spaces or tabs. Lines that do not begin with "ATOM" or "HETATM" are ignored. The programs make no distinction between ATOMs and HETATMs (note above that token 1 is ignored). No atname-resnum-chainid combination can occur more than once. (Atom data is stored internally in associative arrays and syntax of things like .st files depend on ability of atname-resnum-chainid combination to uniquely specify an atom.)

Note that the .pqr format does not support some PDB-isms such as a altLoc fields, and a one-character chainID between resName and resSeq. Doing so would break the whitespace separated tokens convention that allows for easy processing with perl scripts, etc. Instead we put "chainID" in a position more or less analogous with the PDB segID. (Maybe we should have called it segID.)

If you have a PDB file and need to generate a PQR file, this implies making some choices about the charges and radii. This is similar to making a choice about what force field to use in an MD simulation. MEAD per se, doesn't make the choice for you. However, in the utilities subdir are some tools that may be useful if you want to use CHARMM or PARSE parameters. The Amber program suite comes with a program, ambpdb, that allows you to generate a PDB or PQR file given Amber-format files. Another option is the program pdb2pqr from pdb2pqr.sourceforge.net.

MolName.ogm, MolName.mgm :

Specification for the grid method. Each line specifies a level of a focussed set of grid calculations, starting with the coarsest and going to the finest. Lines have the format,

centering grid_dimension grid_spacing

where grid_dimension is an odd integer greater than 1 specifying the number of points along the edge of the grid; grid_spacing is a positive floating point number specifying the spacing between grid points; and centering is ON_ORIGIN, ON_GEOM_CENT or ON_CENT_OF_INTR, according to whether the grid is to be centered on the origin of the coordinate system, the geometric center or the "center of interest" which, for multiflex is on one of the titrating charges in the site being calculated (see .st files, below) and for other programs is on the origin (so ON_CENT_OF_INTR not much use with selfenergy). Multimead needs the .ogm file to specify the grid method for the macromolecule and the .mgm file to specify the grid method for the model compound. Other programs need only the .ogm file. For proper cancelation of grid effects, the finest levels for the model compound and the macromolecule must use the same grid spacing and centering style. The grid center can also be specified explicitly by giving a point in the format, "(x, y, z)" as the centering.

MolName.fpt:

Specifies that "field points" for which the potential program should sample and write out the potential. The x, y, z coordinates should be specified as three numbers written in the usual floating-point way, and separated by any amount of white space. Optionally, the three numbers can by surrounded by parenthesis and separated by commas. Points are separated from each other by white space. Line breaks are considered the same as any other white space.

MolName.sites (multiflex only):

This file specifies which sites in a macromolecule are titrating and what kind they are. For each site it contains a line of the form,

resnum site_type chainid

where resnum is the residue number of the site and will be matched to the residue number field of the atoms in the MolName.pqr file, chainid is to be matched to the chainid given in MolName.pqr, and site_type refers to site_type.st files. Usually, site_type will be similar to a residue type name. The chainid field is optional, but you must be consistent in either including or omitting it in MolName.sites and MolName.pqr.

site_type.st: (multimead and multiflex only)

Multiflex expects a file of the name site_type.st to specify details concerning each site_type that appears in the MolName.sites file. The first line contains one floating point number which is the model compound pK for that type of site. All remaining lines are of the form,

resname atomname prot_charge deprot_charge

where resname and atomname, along with the resnums given in the .sites file match an atom in the .pqr file that is part of a titrating site, prot_charge is that charge of this atom in the protonated state and deprot_charge is its charge in the deprotonated state. It is expected that the sum of all prot_charge subtracted from the sum of all deprot_charge will equal one.

I plan to extend this file's syntax to allow a regular expression to be given that will specify how a model compound is to be made from the protein atom coordinates.

Something.potat : (output)

This is a binary output file produced by some programs. It contains the potential at each atom of MolName (or a model compound) generated by some set of charges. Variations of "something" may denote charge states, sites, conformers, solvated or vacuum or uniform dielectric environments, depending on the application. Atomic coordinates and radii

and the generating charges are also included for the sake of consistency checking. These files allow multiflex, etc. to avoid recalculating a lot of things when all you want to do is add or alter some site, but you must be careful about site names, which .potat files you leave sitting around, and so on.
Specify the -blab2 flag for a blow-by-blow account of attempts to read and write .potat files in multiflex.

MolName.pkint, MolName.g MolName.sitename.summ: (multiflex)

Described in the multiflex summary above.

REDTI:

Redti solves the multiple site titration curve problem given a set of intrinsic pK's (MolName.pkint) and a site-site interaction matrix (MolName.g) using the Reduced Site method described by Bashford & Karplus (1991) J. Phys. Chem. vol. 95, pp. 9556-61. Its command line syntax is:

redti [-cutoff val] [-pHrange val val] [-dry] MolName

where the "val" are numbers giving the protonation fraction cutoff for the reduced site method, the bottom of the pH range to be covered and the top of the range, respectively. The defaults are 0.05, -20.0 and 30.1, respectively. (Yes, that's an extreme pH range). The -dry flag causes redti to do a "dry run" in which it prints the number of sites to be included in the reduced site calculation at each pH point. This is useful for checking whether a calculation will require a prohibitive amount of CPU time since CPU time will go exponentially in the reduced site number.

Redti is written in C rather than C++.

AUTHOR:

Donald Bashford
don.bashford@stjude.org

Hartwell Center
Saint Jude Children's Research Hospital
Memphis, TN