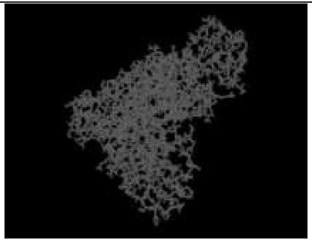
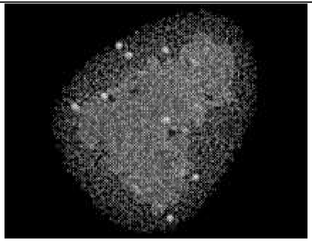


SOLVATE 1.0



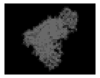
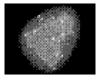
Theoretical Biophysics Group
Ludwig-Maximilians-Universität München, Germany
http://www.imo.physik.uni-muenchen.de/~grubi
email: Helmut.Grubmueller@Physik.uni-muenchen.de

Overview

`SOLVATE` is a program to construct an atomic solvent environment model for a given atomic macromolecule model (solute) for use in molecular dynamics simulations.

`SOLVATE`

- generates irregularly-shaped solvent volumes, adapted to a given solute's structure,
- allows efficient computation of boundary forces as required in molecular dynamics simulations,
- guarantees a minimal solute-boundary-distance,
- allows to specify a minimum solvent surface curvature in order to avoid 'flat' surface regions,
- generates 'disordered' (= fluid) water, not a grid of water molecules (= ice),
- locally minimizes the positions of all water molecules,
- optionally places salt ions (Na^+/Cl^-) obeying a Debye-Hückel distribution,
- optionally places (and marks) buried water molecules,
- and is X-PLOR/CHARMm-compatible, i.e., input and output is in pdb-/psf-format.

 **SOLVATE 1.0** 

Getting and Installing SOLVATE

`Solvate 1.0` can be obtained as a compressed tar-file from anonymous ftp (ftp.imo.physik.uni-muenchen.de/pub/solvate_1.0/solvate_1.0.tar.Z).

To install it, move the tar-file to your favourite directory and enter:

```
zcat solvate_1.0.tar.Z | tar -xvf -
```

The source `solvate.c` is just one ANSI-C-file without any system-specific stuff (except for `malloc()` --- RS6000 and NeXT users: pay attention!), so it should compile right away with something like

```
cc -ansi -O -o solvate solvate.c -lm
```

It is tested under IRIX System V.4 (SGI), HP-UX, and Digital UNIX V3.2C; appropriate `Makefile`s are included.

The distribution contains a postscript version of this manual (`docu.ps`) as well as an html-version (`html/docu.html` with references to all other files in the `html`-directory).

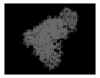
The distribution also contains two sample-solutes, `bpti.pdb`/ `bpti.psf` and `globh.pdb`/ `globh.psf`. To check your installation, try:

```
solvate -t 5.0 -n 2 -ion bpti solbpti
```

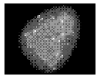
After a while (approx. 10 minutes on a DEC ALPHA/2100) you should see the following output:

```
  .:::::..    SOLVATE 1.0   (C) 1996 Helmut Grubmueller
 ::oOOOoo::        Purpose: make a water shell around solutes
 ::OOO::::        http://www.imo.physik.uni-muenchen.de/~grubi
  :::::         Helmut.Grubmueller@Physik.uni-muenchen.de

Reading solute from file bpti.pdb ... 568 atoms read.
Reading atomic charges from file bpti.psf ... total charge is +6.00 e.

Calculating bounding sphere ... done.
Creating convex volume (max. radius is 100000.000000) ... done.
Gauss-sampling convex volume ... done.
Approximating volume with 2 gaussians: Phase 1/4...2/4...3/4...4/4...done.
Saving parameter for 2 gaussians to file gaussians.lis ... done.

Adjusting boundary distance by varying gaussian widths ...   5.046 A
Gaussian widths are being scaled by a factor of 1.579210
Saving boundary parameters to file boundary.lis ... done.
Creating volume within boundary ... done.
Sorting grid points according to distance from solute ... done.
Performing distance approximation statistics with 58283 grid points ... 100%
Checking approximation for distance from boundary ... 100%

Estimated number of water molecules: 2297
Placing water molecules ... found 1692
Grouping water molecules ... 100%

The water molecules form 2 groups.
Group 1 contains 1691 water molecules
Group 2 contains 1 water molecule
So we have one single, 0 twins, and 0 triplets

Placing ions ... placed 2 sodium ions and 8 chloride ions.
Equilibrating ions (2000000 Monte Carlo moves) ... 100%

Saving solute, 1682 water molecules, and 10 ions to file solbpti.pdb ... done.

Solvate ready. No errors.
```

Helmut Grubmueller
Wed Jun 19 19:00:00 MET DST 1996



SOLVATE 1.0



Purpose

Molecular dynamics (MD) studies of biological macromolecules (such as proteins or DNA) require a large amount of computations, mainly due to the large number of atoms (typically several thousands) such molecules consist of. Even when using supercomputers, this computational complexity limits such simulations to very short time scales, typically a few nanoseconds.

Furthermore, just considering the macromolecule of interest is not enough, for the physiological environment of the molecule (e.g., for globular proteins, water solvent and salt ions) strongly interacts with the dissolved macromolecule. Therefore, also part of that environment has to be included within the simulation system, which then usually contains several tens of thousands of atoms [1,2]. Obviously, one wishes to include only as many solvent molecules within the simulation system as *really* necessary, while keeping the influence of the (artificial) system boundary (the surface of the solvent) onto the dissolved macromolecule as small as possible. SOLVATE serves this purpose.

SOLVATE is a program to construct an atomic model of a solvent environment for a given atomic macromolecule model (solute). Its main feature is that it can create *irregularly*-shaped solvent volumes specifically adapted to the (mostly also irregular) shape of the solute. As a result, the number of atoms that have to be considered in the MD-simulation is significantly reduced in comparison to the usual box- or sphere-shaped volumes, so that the simulations run considerably faster.

SOLVATE accepts pdb-files (Brookhaven Data Bank/X-PLOR format) of the solute as input; the output is a pdb-file of the solute plus a number of water molecules and, optionally, sodium and chloride ions placed in accordance with a Debye-Hückel-distribution. To generate ions, an X-PLOR protein structure (psf-)file of the solute (which contains the partial charges of the solute atoms) is required in addition to the pdb-file. An X-PLOR-script to create a protein structure file of the solute/water/ion-system can be generated. Note that the output structure is *not* equilibrated, since this can be done with the usual programs, e.g., CHARMm [3], X-PLOR [4], or EGO [5].

Clearly, just creating solute-adapted solvent volumes is not an art --- it can easily be done, e.g., with the `around`-function of X-PLOR. However, just to have such a solvent/solute-system is not enough for an MD-simulation. Rather, water molecules near the surface of the solvent volume must be subjected to ‘boundary-forces’ in order to prevent these water molecules from evaporating into the vacuum. Ideally, these forces should substitute all those forces (e.g., electrostatic interactions and van der Waals contacts), which would originate from solvent molecules *outside* the simulation system. The socalled ‘stochastic boundary’-approach [6,7] provides such forces in a simple approximation by use of a suitable boundary potential.

Such boundary potentials are generally functions of the distance between the particle, onto which the force acts, and the boundary of the simulation system. Hence, for an efficient computation of the forces derived from this potential, it should be of simple analytic form. This is the reason why, typically, box- or sphere-shaped solvent-volumes are used.

SOLVATE provides an (almost) arbitrarily-shaped solvent volume *and* a simple analytic description of its boundary using few multivariate gaussian functions. That choice of description enables efficient computation of distances between water molecules and the boundary during subsequent MD-simulations. The necessary algorithms will soon be implemented into the MD-program EGO, but they can also easily be implemented within CHARMm or X-PLOR.

Helmut Grubmueller
Wed Jun 19 19:00:00 MET DST 1996

SOLVATE 1.0

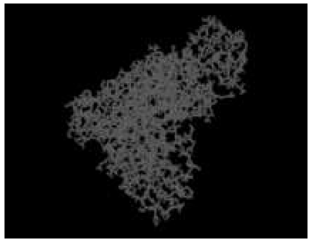
How SOLVATE works

SOLVATE creates the solute/solvent/ion simulation system in a number of steps. For optimal use of SOLVATE, knowledge of these steps is advantageous. As an example, the accompanying pictures show the generation of a solvent shell around a protein complex (immunoglobulin/lysozyme), 6Å thick, including ions. Starting from the coordinate- and structure-file of the complex (`glob.pdb` and `glob.psf`), the shell was created with the command

```
solvate -t 6.0 -n 8 -ion glob globsol
```

yielding the file `globsol.pdb`.

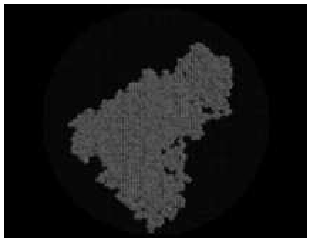
STEP 1: Read in solute



SOLVATE reads the atom coordinates $\mathbf{a}_i = (a_{x,i}, a_{y,i}, a_{z,i})^T$ and atom names of the solute (Figure) from the pdb-file specified on the command line. From the atom names SOLVATE derives approximate van der Waals parameters (radii σ_i and interaction strengths ϵ_i). If ions are to be placed (as in our case), SOLVATE must know about the electrostatics of the solute, which it derives from the atomic partial charges read from the psf-file.

If no pdb-file is given, or if the pdb-file contains no atoms, one ‘dummy-atom’ (with zero radius) is created at the cartesian origin as the ‘solute’. By that means a ‘pure’ spherical water droplet can be created.

STEP 2: Create minimal convex volume



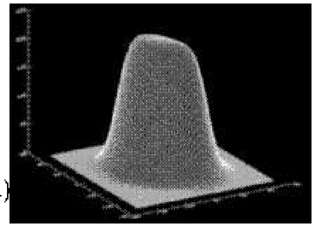
(a) Bounding sphere

On the basis of the atomic positions of the solute the smallest convex volume containing the solute is computed and represented by a regular set of ‘grid points’ with 1Å spacing. To that end, in a first step center and radius of the solute’s ‘bounding sphere’, i.e., the smallest sphere containing the solute, are computed. This bounding sphere is then filled with grid points (see Figure; the solute seems not to completely fill the bounding sphere; however it does, since the sphere and the solute are three-dimensional objects).

(b) Slicing

From that spherical volume the minimal convex volume is subsequently constructed by ‘slicing away’ parts of the spherical volume in many different directions by a ‘knife’ (solid lines) that just touches the solute. To vary the flatness of the surface of the minimal convex volume, a ‘bended knife’ (dashed lines) can be used by specifying a maximum boundary curvature radius. The N_g grid points $\mathbf{g}_i$ which survive the slicing procedure span the desired minimal convex volume.

An ideal solute-adapted boundary would be given by a surface enclosing the minimal convex volume at a given constant distance **d**.



To a good approximation, such an ideal surface can be defined as an iso-surface of a density function $\tilde{f}(\mathbf{x})$,

$$\tilde{f}(\mathbf{x}) = \sum_{i=1}^{N_g} \exp\left[-\frac{(\mathbf{x}-\mathbf{g}_i)^2}{2d^2}\right] \qquad (1)$$

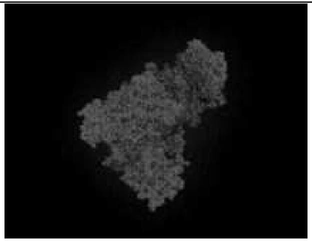
by requiring $\tilde{f}(\mathbf{x}) = f_0$ with suitably chosen f_0. The figure shows a cut through this density function, $\tilde{f}(x, y, z = 0)$, for our example solute.

This boundary certainly fulfills the geometric requirements, but its computational treatment is highly inefficient, since the number N_g of exponentials to be computed in Eq. 1 (the number of grid points spanning the convex volume) is typically of the order $10^5 \ldots 10^6$.

STEP 3: Compute an approximate density function

Note, however, that the density function $\tilde{f}$ defined above is quite smooth, since the distance between grid points (1Å) is much smaller than the width of the (univariate) gaussians used (typically 10Å). Therefore, $\tilde{f}$ can be approximated to sufficient accuracy by a sum of much fewer (N_c) *multivariate* gaussians,

$$\tilde{f}(\mathbf{x}) \approx f(\mathbf{x}) = \sum_{j=1}^{N_G} h_j \exp[-(\mathbf{x}-\mathbf{y}_j)^T s^2 \mathbf{A}_j (\mathbf{x}-\mathbf{y}_j)] \quad , \qquad (2)$$



where h_j are the heights and $\mathbf{y}_j$ are the centers of the N_c gaussians. The matrices $\mathbf{A}_j$ specify the shapes of the gaussians; their overall extension in space can be varied by a scale-factor **s**. Experience shows that usually very few (less than 10) gaussians are sufficient, so that the computational cost for the necessary distance computations in MD simulations becomes negligible. To find optimal heights, centers and shape matrices, `SOLVATE` uses a recently proposed maximum likelihood density estimation method [8]. After having computed these parameters, they are written to the file `gaussians.lis`.

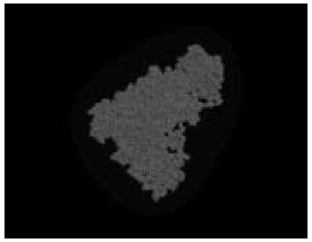
The Figure shows a set of dots, the density of which obeys the density function **f**.

STEP 4: Adjust boundary distance from solute

`SOLVATE` uses a fixed isosurface level $f_0 = 0.25/h$ (where $h = \sum_j h_j/N_c$ is the average height of the gaussians). Obviously, the distance of the solvent surface from the solute, as defined by f_0, is not known at this point. To ensure that the smallest distance equals a given distance **d**, `SOLVATE` iteratively varies the scale-factor **s** (i.e., the widths of the gaussians) until the minimum distance between solute and solvent surface approaches the desired value.

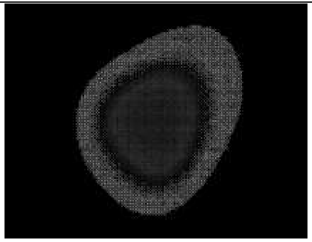
All parameters necessary to define $f(\mathbf{x})$ (h_j, $\mathbf{y}_j$, and $s \cdot \mathbf{A}_j$) are now written to the file `boundary.lis` for later use by an MD-program, e.g., by EGO.

STEP 5: Create solvent volume



In a similar way as in STEP 2, the boundary surface $f(\mathbf{x}) = f_0$ is filled with a number of grid points $\mathbf{v}_i$ (see figure). For every grid point the minimum distance to the solute, $d_i = \min_j |\mathbf{v}_i - \mathbf{a}_j|$, is determined and stored; the grid points are then sorted by increasing minimum distance, which will be useful to efficiently place the solute (water) molecules. Those grid points which are located very close to the boundary can be used to visualize that boundary and are therefore written to the file `surface_stat.lis` if desired.

STEP 6: Perform distance approximation statistics



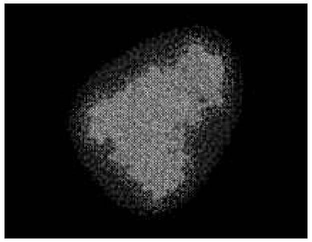
As will be described in Sec. 6, the distance of a given point $\mathbf{x_0}$ of the solvent volume to the boundary can be efficiently estimated from $f(\mathbf{x}_0)$ (shown in the figure, colour-coded) to a sufficient accuracy (≈ 0.1Å). To check the accuracy of the distance computation, for each grid point $\mathbf{v}_i$ the efficiently estimated distance is compared with the accurate distance, and, if the `-s` option is set, the resulting error statistics is appended to the file `surface_stat.lis`.

In our example, this statistics reads

```
[MINIMUM INVALID DENSITY]
0.935194
[DISTANCE ERROR STATISTICS (ABS. ERR / DENSITY / DISTANCE)]
0.01 0.401408 3.245818
0.02 0.451980 4.120950
0.05 0.538697 5.512379
0.10 0.631438 6.651468
0.20 0.753683 8.271388
0.50 0.985918 10.718131
1.00 10000000000000000000.000000 62.000000
```

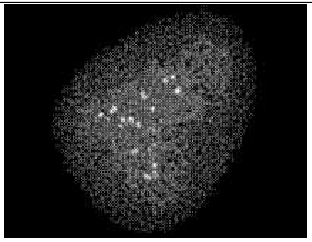
which means that all distances smaller than 3.245818 Å (for these, **f**<0.401408), are computed within an error of 0.01 Å; all distances smaller than 4.120950 Å (**f**<0.451980) within an error of 0.02 Å and so on. No error of 1.0 Å or larger occurred. Distance computations are valid for all locations within the boundary where **f** is below the minimum invalid density (0.935194).

STEP 7: Place water molecules



Using the sorted grid points $\mathbf{v}_i$, and starting at points closest to the solute, the solvent volume is filled with water molecules, one molecule after the other. In this process, for each grid point $\mathbf{v}_i$ `SOLVATE` checks whether the distances of $\mathbf{v}_i$ to all solute atoms as well as to all water molecules already placed is larger or equal to the appropriate van der Waals distance. If not, the respective grid point is discarded; if yes, a water molecule is placed at location $\mathbf{v}_i$ and, by steepest descent, subsequently moved to a nearby energetically favorable position (only van der Waals energies are considered here). By this procedure, water molecules close to the solute are placed first (drawn in blue in the figure), followed by water molecules further apart (the ones placed last are drawn in red).

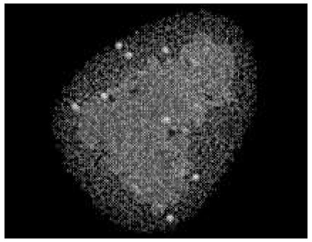
STEP 8: Group water molecules



Water molecules closest to the solute are likely ones placed in ‘caves’ *inside* the solute (such caves exist, e.g., inside proteins). To distinguish such ‘buried’ water molecules (drawn as balls in the figure) from bulk water (small angles), all water molecules are grouped according to their connectivity. Typically a few dozen ‘groups’ consisting of just one isolated molecule, of a pair or a triplet of molecules (depending on the size of the ‘cave’) will be identified as well as the bulk group containing all the other water molecules. The groups are consecutively numbered, starting with #1 for the bulk group.

Note that `solvate` places buried water molecules only according to steric criteria, **not** according to energetic criteria. If buried water molecules found by `solvate` are to be included within a subsequent MD-simulations, it has to be checked whether their free energy is low enough so that they are likely to really be there. A good estimate is provided by the program `Dowser` (http://femto.med.unc.edu/Research/dowser.html or similar software.

STEP 9: Place ions (optional)



Sodium (light blue) and chloride ions (green) are placed in the solvent volume at isotonic (physiological) concentration (0.154mol/l) obeying the Debye-Hückel distribution, which depends on the locations of charged atoms of the solute (red, blue): on average, each charged atom at the surface of the solute is surrounded by a ‘cloud’ of socalled counter-ions. The size of this cloud is given by the Debye-Hückel length κ, $1/\kappa^2 = 2e^2/(\epsilon_0 \epsilon k_B T)$, where **e** is the elementary charge, $\epsilon_0\epsilon$ is the dielectric constant, k_B is the Boltzmann constant, and **T=300K** the temperature. The density $n_i(r)$ (**i**=Na,Cl) of an ion cloud caused by a solute atom with partial charge $z_a \cdot e$ is a function of the distance **r** from the charged atom, approaches the isotonic concentration $n_i^{(0)}$ for large **r**, and is computed in linear approximation,

$$n_i(r) = n_i^{(0)} \left(1 - \frac{z_i z_a e^2}{4\pi\epsilon_0 \epsilon k_B T} \cdot \frac{e^{-\kappa r}}{r}\right) \quad , \qquad (3)$$

with $z_{\mathrm{Na}} = +1$ and $z_{\mathrm{Cl}} = -1$. Due to the linear approximation, $n_i(r)$ may become negative, in which case it is set to zero. The total ionic density is the determined from the linear superposition of all the ion clouds around the charged solute atoms. In a first step, all ions (the number of which is determined from the charge density integral over the solvent volume) are placed at random according to that Debye-Hückel total ionic density.

Since the Debye-Hückel approximation is a mean field description, at this point ion-ion correlations are not yet described. To find ionic positions which obey also these higher order correlations, and to avoid artifacts due to the linear approximation, the ions are subsequently subjected to a large number (2,000,000) of Monte-Carlo moves, where the Coulomb field of the solute, and now also the inter-ionic Coulomb field is considered.

The current version of `SOLVATE` does not yet allow to use a salt concentration different from the isotonic concentration, neither is it possible to include ions other tan sodium and chloride or to set a temperature different from 300K.

STEP 10: Place Hydrogens

Since up to now, the water molecules have been treated as simple van der Waals spheres (one per molecule) representing only the oxygen positions, two hydrogen atoms per water molecule have to be added. These are oriented at random; a realistic short range order of these dipoles can be created within a short MD run (10 picoseconds is long enough).

STEP 11: Write pdb-file

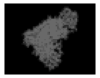
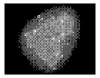
Finally, a pdb-file is written, containing the solute, the positions of the water molecules, and, if present, the positions of the ions.

In the pdb-file each group of water molecules is assigned a unique segment-identifier, starting with `W100` for the innermost group, `W101` for the second and so on. Bulk water is stored last.

If present, ions are appended at the end of the pdb-file; sodium ions first (atom name `NA`, ‘molecule’ `INA`, segment-identifier `NA`), then the chloride ions (atom name `CL`, ‘molecule’ `ICL`, segment-identifier `CL`)

Optionally, `SOLVATE` generates an X-PLOR-script to create a (psf-) structure file for the final solute/solvent-system. Appropriate topology-files and parameter-files to describe the water molecules from the X-PLOR distribution (e.g., `toph19.sol` and `param19.sol`) are required to generate a psf-file.

Helmut Grubmueller
Wed Jun 19 19:00:00 MET DST 1996

 SOLVATE 1.0 

Command line options and parameters

`SOLVATE` receives all input parameters from the command line, so it can easily be used within shell-scripts.

Available options are:

```
solvate [-t thick] [-r radius] [-n ngauss] [-ug] [-ub]
        [-s] [-v] [-bulk] [-w] [-ion] [-psf] [infile] outfile
```

`thick`
: specifies the minimum water shell thickness in Angstrom (default: 10.0). Nowhere will the solute be closer to the surface of the solvent than this value. Attention: Values smaller than 3.0 may confuse `SOLVATE`.

`radius`
: specifies the maximum boundary curvature radius of the solvent surface in Angstrom (default: 100000.0). A very large value (e.g., the default value) means that the surface can (but not necessarily does) have rather flat parts. Smaller values generate 'rounder' solvent volumes, and, as a consequence, more solvent molecules. The value of `radius` should not be considerably smaller than the size of the solute.

`ngauss`
: specifies the number of gaussians to be used to define the solvent surface (default: 1). The more detailed irregular features of the surface are required, the more gaussians should be used. Consequently, thin water shells typically require more gaussians than thicker ones. Note, however, that a large number may slow down subsequent molecular dynamics simulations. Typical values for `ngauss` are five to ten.

`-s`
: If the `-s` option is set, the file `surface_stat` is written, which contains (i) a set of grid points specifying the solvent surface, (ii) the error statistics for the distance estimate described above, and (iii) information about how many water molecules belong to which group of molecules.

`-v`
: If the `-v` option is set, the file `volume_stat` is written (attention: this file may become quite large!), which lists for every grid point within the solvent volume its **x**-, **y**-, and **z**-coordinate, the value $f(x, y, z)$ of the density function at this point, its accurate distance and the approximate distance (which is the efficient estimate used in subsequent MD-simulations) from the solute surface, as well as an approximate value of the curvature of the surface at the point next to (x, y, z) (which also is an efficient estimate that can be used in MD-simulations).

`-ug`
: If the `-ug` (use gaussians) option is set, all steps required for the computation of the approximate density function **f** (STEPs 2 and 3) are skipped, and the required parameters for defining **f** are read from the file `gaussians.lis` instead, which is always written when **f** is computed. The reason for this option is that STEP 3 is quite time consuming.

`-ub`
: If the `-ub` (use boundary) option is set, all steps required for the computation of the boundary description through **f** (STEPs 2, 3, and 4) are skipped, and the required parameters for defining **f** and the scale factor **s** are read from the file `boundary.lis` instead, which is always written after the boundary distance from the solute has been adjusted (STEP 4). The reason for this option is that STEP 4 is quite time consuming.

`-bulk`
: The `-bulk` option suppresses output of buried water molecules, i.e., only bulk water is written to the output pdb-file.

`-w`
: The `-w` option suppresses output of the solute, i.e., only water molecules (usually with a hole in the middle, where the solute is located) and ions are written to the output pdb-file.

`-ion`
: If the `-ion` option is set, STEP 9 is carried out to place sodium and chloride ions into the solvent according to a isotonic Debye-Hückel density. If the option is not given, STEP 9 is skipped, and no ions are output.

`-psf`
: If the `-psf` option is set, `SOLVATE` writes an X-PLOR-script `mkpsf.inp"` which can be used to generate a structure file for the solute/solvent-system, as required for subsequent MD-simulations (the command `xplor < mkpsf.inp` will do the job).

`infile`
: specifies the name of the input pdb-/psf-file of the solute. If the `-ion` option is set, both pdb- and psf-file are required; otherwise only the pdb-file is needed. No extension must be given!
The input file name may be omitted, in which case `SOLVATE` creates a pure spherical water droplet centered at the origin.

`outfile`
: specifies the name of the output file for the solute/solvent-system. No extension should be given.

The command line options can be given in arbitrary sequence.

Helmut Grubmueller
Wed Jun 19 19:00:00 MET DST 1996

References

1 Helmut Heller, Michael Schaefer, and Klaus Schulten. Molecular dynamics simulation of a bilayer of 200 lipids in the gel and in the liquid crystal phase. *J. Phys. Chem.*, 97(31):8343--8360, 1993.

2 Helmut Grubmüller, Berthold Heymann, and Paul Tavan. Ligand binding: Molecular mechanics calculation of the streptavidin-biotin rupture force. *Science*, 271(5251):997--999, 1996.

3 Bernhard R. Brooks, Robert E. Bruccoleri, Barry D. Olafson, David J. States, S. Swaminathan, and Martin Karplus. CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *J. Comp. Chem.*, 4(2):187--217, 1983.

4 A. T. Brünger. *X-PLOR*. The Howard Hughes Medical Institute and Department of Molecular Biophysics and Biochemistry, Yale University, 260 Whitney Avenue, P.O. Box 6666, New Haven, CT 06511 (1988), May 1992.

5 Markus Eichinger, Helmut Grubmüller, and Helmut Heller. *User Manual for EGO_VIII, Release 2.0*. Theoretische Biophysik, Institut für Medizinische Optik, Universität München, Theresienstr. 37, D-80333 München, Germany (1995); electronic access: http://www.imo.physik.uni-muenchen.de/ego.html.

6 C. L. Brooks III and M. Karplus. Deformable stochastic boundaries in molecular dynamics. *J. Chem. Phys.*, 79:6312--6325, 1983.

7 Axel Brünger, Charles Brooks, and Martin Karplus. Active site dynamics of ribonuclease. *Proc. Natl. Acad. Sci. USA*, 82:8458--8462, Dec. 1985.

8 M. Kloppenburg and P. Tavan. In preparation.

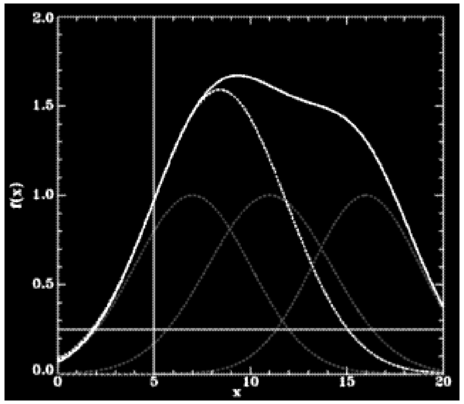
Helmut Grubmueller
Wed Jun 19 19:00:00 MET DST 1996

SOLVATE 1.0

Distance Computation in MD-Simulations

Once a density function $f(\mathbf{x})$ is computed by `SOLVATE` and the parameters defining **f** are stored in the file `boundary.lis`, any MD-program can access that file and use **f** to compute boundary forces acting on water molecules near the surface of the solute/solvent system. This force is always a function of the distance to that surface, which therefore must be computed efficiently. In this section we describe how this can be done to an accuracy sufficient for most purposes.

For the computation of the distance **r** between the location of a water molecule, $\mathbf{x}_0$, and the surface point closest to $\mathbf{x}_0$ we first have to find that surface point. We assume that, from the construction of **f** as a sum of few gaussians, this surface point is sufficiently close to the intersection of a line through $\mathbf{x}_0$, the direction of which is given by the gradient $\nabla f(\mathbf{x}_0)$, with the surface $f(\mathbf{x}) = f_0$. So we have to compute that intersection point.



Since this cannot be done analytically, and to avoid a possible (but slow) iterative method, we approximate **f** for points on the line defined above beween $\mathbf{x}_0$ and the intersection point. In that region, **f** (solid white line) is by construction (a sum of gaussians, red, dashed lines) dominated by the gaussian next to $\mathbf{x}_0$ (thin, vertical line) and thus can be approximated by a simple (one-dimensional) gaussian function **g** (white dashed line),

$$g(x) = k \exp\left[-\frac{(x-a)^2}{2\sigma^2}\right] \quad .$$

The parameters **k**, **a**, and σ, which define **g**, are obtained from the value $y_0 := f(\mathbf{x}_0)$, the gradient $y_1 := |\nabla f(\mathbf{x}_0)|$, as well as the second derivative along the gradient, $y_2 := \nabla f(\mathbf{x}_0) \cdot J \cdot \nabla f(\mathbf{x}_0) / y_1^2$ (where **J** is the Jacobian matrix of **f** at $\mathbf{x}_0$),

$$
\begin{aligned}
\sigma^2 &= \frac{y_0^2}{y_1^2 - y_0 y_2} \quad , && (5)\\
k &= y_0 \exp\left[\frac{y_1^2}{2(y_1^2 - y_0 y_2)}\right] \quad , && (6)\\
u &= \frac{y_0 y_1}{y_1^2 - y_0 y_2} \quad . && (7)
\end{aligned}
$$

The intersection point x_1, approximated by requiring $g(x_1) = f_0$, can now be calculated analytically, and the distance **r** we are interested in reads

$$
r = \left| u - \sqrt{2\sigma^2 \ln(k/f_0)} \right| \quad . \qquad (8)
$$

As a rule of thumb, all distances required in an MD-simulation are computed with an error smaller than 0.2 Å, which becomes even smaller (0.02 Å) for water molecules close to the surface.

Through a similar approximation, also the average curvature of the solvent at the intersection point, which may enter the boundary forces as a second parameter, can be obtained.

Helmut Grubmueller
Wed Jun 19 19:00:00 MET DST 1996

SOLVATE 1.0

General License Agreement and Lack of Warranty

This software is distributed in the hope that it will be useful but *without any warranty*. The author does not accept responsibility to anyone for the consequences of using it or for whether it serves any particular purpose or works at all. No warranty is made about the software or its performance.

Use and copying of this software and the preparation of derivative works based on this software are permitted, so long as the following conditions are met:

- The copyright notice and this entire notice are included intact and prominently carried on all copies and supporting documentation.
- No fees or compensation are charged for use, copies, or access to this software. You may charge a nominal distribution fee for the physical act of transferring a copy, but you may not charge for the program itself.
- If you modify this software, you must cause the modified file(s) to carry prominent notices describing the changes, who made the changes, and the date of those changes. Please do inform the author about major modifications.
- Any work distributed or published that in whole or in part contains or is a derivative of this software or any part thereof is subject to the terms of this agreement. The aggregation of another unrelated program with this software or its derivative on a volume of storage or distribution medium does not bring the other program under the scope of these terms.

This software is made available *as is*, and is distributed without warranty of any kind, either expressed or implied. In no event will the author or his institutions be liable to you for damages, including lost profits, lost monies, or other special, incidental or consequential damages arising out of or in connection with the use or inability to use (including but not limited to loss of data or data being rendered inaccurate or losses sustained by third parties or a failure of the program to operate as documented) the program, even if you have been advised of the possibility of such damages, or for any claim by any other party, whether in an action of contract, negligence, or other tortuous action.

SOLVATE is written by Helmut Grubmüller, Theoretical Biophysics Group, Institut für Medizinische Optik, Ludwig-Maximilians-Universität München, München, Germany. Copyright © 1996. All rights reserved.

Helmut Grubmueller
Wed Jun 19 19:00:00 MET DST 1996