



OpenEye
Scientific Software

SZYBKI

Release 1.7.0

OpenEye Scientific Software, Inc.

January 26, 2012

CONTENTS

1	Front Matter	1
2	Installation and Platform Notes	3
2.1	Licenses	3
2.2	Installation	3
2.3	Uninstallation	5
2.4	PVM	6
2.5	Open MPI	7
3	SZYBKI Theory	11
3.1	SZYBKI Theory	11
4	Application	15
4.1	Usage	15
4.2	Examples of using SZYBKI	22
5	Release Notes	27
5.1	SZYBKI 1.7.0	27
5.2	SZYBKI 1.5.1	28
5.3	SZYBKI 1.5.0	29
5.4	SZYBKI 1.3.4	29
5.5	SZYBKI 1.3.3	30
5.6	SZYBKI 1.3.2	30
5.7	SZYBKI 1.3.1	30
5.8	SZYBKI 1.3.0	31
5.9	SZYBKI 1.2.2	31
	Bibliography	33
	Index	35

FRONT MATTER

Copyright 1997-2012 OpenEye Scientific Software, Santa Fe, New Mexico. All rights reserved.

All rights reserved. This material contains proprietary information of OpenEye Scientific Software. Use of copyright notice is precautionary only and does not imply publication or disclosure.

The information supplied in this document is believed to be true but no liability is assumed for its use or the infringement of the rights of others resulting from its use. Information in this document is subject to change without notice and does not represent a commitment on the part of OpenEye Scientific Software.

This package is sold/licensed/distributed subject to the condition that it shall not, by way of trade or otherwise, be lent, re-sold, hired out or otherwise circulated without OpenEye Scientific Software's prior consent, in any form of packaging or cover other than that in which it was produced. No part of this manual or accompanying documentation, may be reproduced, stored in a retrieval system on optical or magnetic disk, tape, CD, DVD or other medium, or transmitted in any form or by any means, electronic, mechanical, photocopying recording or otherwise for any purpose other than for the purchaser's personal use without a legal agreement or other written permission granted by OpenEye.

This product should not be used in the planning, construction, maintenance, operation or use of any nuclear facility nor the flight, navigation or communication of aircraft or ground support equipment. OpenEye Scientific Software, shall not be liable, in whole or in part, for any claims arising from such use, including death, bankruptcy or outbreak of war.

Windows is a registered trademark of Microsoft Corporation. Apple, OS X, and Macintosh are registered trademarks of Apple Computer, Inc. AIX and IBM are registered trademarks of International Business Machines Corporation. UNIX is a registered trademark of the Open Group. RedHat is a registered trademark of RedHat, Inc. Linux is a registered trademark of Linus Torvalds. SPARC is a registered trademark of SPARC International Inc.

SYBYL is a registered trademark of TRIPOS, Inc. MDL is a registered trademark and ISIS is a trademark of Accelrys, Inc. SMILES, SMARTS, and SMIRKS may be trademarks of Daylight Chemical Information Systems. Macromodel is a trademark of Schrodinger, Inc.

Python is a trademark of the Python Software Foundation. Java is a trademark or registered trademark of Sun Microsystems, Inc. in the U.S. and other countries.

Other products and software packages referenced in this document are trademarks and registered trademarks of their respective vendors or manufacturers.

INSTALLATION AND PLATFORM NOTES

2.1 Licenses

To run SZYBKI you will need to obtain a license file for SZYBKI from OpenEye Scientific Software (business@eyesopen.com). The license file should be pointed to by the **OE_LICENSE** environment variable

If you intend on running multi-processor SZYBKI via PVM or Open MPI, only the master machine needs access to the license file.

On Windows, the environment variables can be set under the system Control Panel.

2.2 Installation

2.2.1 General Installation

By default, all OpenEye applications are installed into a single distribution directory tree on the specified machine. The default location for this tree is platform specific and will be detailed below.

The root of the tree (i.e. the `openeye` directory) contains the following subdirectories:

- admin** This directory is intended to contain any administrative scripts and tools associated with the installed applications. Currently, this directory is simply a placeholder on all platforms except for Microsoft Windows, where it contains the uninstaller executables.
- arch** This directory contains the collection of platform specific subdirectories. Each subdirectory contains the actual installed executables and support libraries for the associated platform. In the platform specific subdirectory there will be a subdirectory for each application. Within that will be another subdirectory for each version of that application.
- bin** This directory contains a startup script for each application that has been installed. This script determines, at run-time, what the current platform is and then calls the appropriate executable in the `arch`. This script enables the easy co-existence of multiple platforms and versions of any OpenEye application in the same distribution tree.
- data** This directory contains all of the associated data for the installed applications. There will be a subdirectory for each installed application and within that subdirectory there will be another subdirectory for each specific version of that application.

docs This directory contains all of the documentation associated with the installed applications. There will be a subdirectory for each installed application and within that subdirectory there will be another subdirectory for each specific version of that application.

examples This directory contains all of the examples associated with the installed applications. There will be a subdirectory for each installed application and within that subdirectory there will be another subdirectory for each specific version of that application.

The startup script discussed in the section on the `bin` directory above will have the same name as the installed executable with which it is associated. When the script is called, it will attempt to determine the current platform and run the appropriate executable if installed. If an appropriate executable cannot be found, the script will report that information, as well as a list of the currently installed platforms. The auto-detection can be overridden by setting one of two environment variables:

- **OE_ARCH** can be used to specify a colon separated list of compatible distributions for the current platform such as:

```
redhat-RHEL5-x64:redhat-RHEL4-x64
```

Specification of this environment variable overrides the auto-detection process, if it is present. If none of the compatible distributions listed are found, the script will fall back to the auto-detection process.

- **APPNAME_OE_ARCH** can be used to specify a colon separated list of compatible distributions for a specific application (as specified by changing the **APPNAME** text in the environment variable name) just like **OE_ARCH** as detailed above.

Specification of this environment variable overrides the **OE_ARCH** environment variable as well as the auto-detection process. If none of the compatible distributions listed are found, the script will fall back to the **OE_ARCH** list first and then to the auto-detection process.

Specifying this variable provides a simple way to customize the behavior for individual applications on non-standard platforms.

The startup script also supports a few commandline arguments including:

- | | |
|---------------------|--|
| -path | Specifying this argument will output the full path of the executable to be run. The executable will not be started if this argument is present. |
| -print_arch | Specifying this argument will output the details of the current platform as detected by the script as well as which platform-version of the executable is being run. The executable will be started if this argument is present. |
| -use_version | Specifying this argument followed by a specific version number allows the user to control which released version of the executable to run. |

2.2.2 Linux/Unix

Linux/Unix distributions are provided as a gzipped tarball of the distribution tree described above. Installation is performed by untarring the file in the desired location. Multiple distributions can be installed in the same location without any challenge.

To ensure that the installed applications can be called from the command line, be sure to add the full path of the `openeye/bin` subdirectory to the **PATH** environment variable. For instance, if the distribution was installed into `/usr/local/openeye`, the **PATH** environment variable should contain: `/usr/local/openeye/bin`.

2.2.3 Windows

Windows distributions are provided as a standard EXE installer. By default, all OpenEye applications will install into the `C:\OpenEye` directory.

An OpenEye group with an application specific subgroup will be added to the *Start* menu. The application specific subgroup will contain links to the documentation, the uninstaller, as well as to a Windows command shell which has the appropriate **PATH** settings already defined to allow the user to simply type the executable name at the prompt without concern for where the executable is actually installed.

For GUI applications, a link to the application will be created on the desktop as well as in the application specific subgroup of the *Start* menu.

2.2.4 Mac OS X

Mac OS X distributions are provided as a standard *pkg* installer delivered as a *dmg* disk image. By default, all OpenEye applications will install into the `/Applications/OpenEye` directory.

To ensure that the installed applications can be called from the command line in the *Terminal*, be sure to add `/Applications/OpenEye/bin` to the **PATH** environment variable.

For GUI applications, an application bundle which can be clicked on to start, will be present in the `/Applications/OpenEye` directory. This bundle cannot be moved independent of the `OpenEye` directory. For instance, the entire `OpenEye` directory can be moved as one piece, but moving the application bundle or the contents of any of the subdirectories in the `OpenEye` directory may cause the application to not start. However, the bundle can still be dragged into the Dock and run from there without any problem.

2.3 Uninstallation

2.3.1 Linux/Unix

To uninstall a single distribution of a product the relevant subdirectories for that product and version simply need to be deleted from within the following directories:

- arch** In the `openeye/arch` directory is a platform specific subdirectory. Within this are directories for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled. For example, to delete or uninstall v1.0.0 of a product, delete the folder “<product_name>/1.0.0”.
- data** In the `openeye/data` directory is a subdirectory for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled.
- docs** In the `openeye/docs` directory is a subdirectory for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled.
- examples** In the `openeye/examples` directory is a subdirectory for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled.

2.3.2 Windows

Installation of an OpenEye product on Windows causes an OpenEye group with an application specific subgroup to be added to the *Start* menu. One of the items in the application specific subgroup is a link to the uninstaller. Clicking on the uninstaller initiates a wizard which guides the user through uninstallation.

For GUI applications, uninstallation also removes the desktop link to the application as well as in the application specific subgroup of the *Start* menu.

2.3.3 Mac OS X

To uninstall a single distribution of a GUI application simply drag the application from /Applications/OpenEye/bin to the Trash can.

To uninstall a single distribution of a command line application you will need to delete the executable/folder from /Applications/OpenEye/arch/osx-10.6-x64/. For example, to delete or uninstall v1.0.0 of a product, delete the folder “1.0.0” located in /Applications/OpenEye/arch/osx-10.6-x64/<product_name>.

Associated documentation, data and example files for a single distribution can be uninstalled by deleting the subdirectory/folder from within the /Applications/OpenEye/data, /Applications/OpenEye/docs and /Applications/OpenEye/examples directories.

2.4 PVM

PVM or parallel virtual machine is a freely available library for running processes on more than one processor on one or more machines ([Geist-1994]). SZYBKI can take advantage of PVM to distribute jobs over multiple processors. To do this PVM must be installed on all the machines SZYBKI will be distributed over. The PVM source is freely available from http://www.csm.ornl.gov/pvm/pvm_home.html. However many Linux distributions, and some Unix versions, include PVM by default. SZYBKI is built with the current PVM version 3.4.5, but should also work with PVM version 3.4.4. SZYBKI does not support PVM under Windows.

To use SZYBKI with PVM you must do one of the following.

- Place a link to the `szybki` executable in `$PVM_ROOT/bin/$PVM_ARCH`
- Define the environment variable `PVM_PATH`, which points to the directory where the SZYBKI executable resides.

The environment variables `PVM_ROOT` and `PVM_ARCH` should be defined globally as part of the PVM installation. `PVM_PATH` is generally a user-defined environment variable, and must be defined for all shells (*i.e.*, you can't just define it in the shell you're launching SZYBKI in).

Note: There is no specific slave executable. The executable distributed for this program serves as both a master and slave PVM program as well as a single processor version.

2.4.1 PVM hosts file

After PVM and SZYBKI are installed, each SZYBKI job run via PVM must be supplied a file (via `-pvmconf`) that describes the hosts and number of CPU's to use. Each line in this file includes the word **host** followed by the hostname of the slave followed by the number of CPU's to use on that slave.

To use 10 slaves on a linux cluster (where nodes are named *linux1*, *linux2*, etc.)

```
host linux1 2
host linux2 2
host linux3 2
host linux4 2
host linux5 2
```

To use 32 CPU's on a single-image system like an SGI Altix, a single line can be used:

```
host altix 32
```

2.4.2 PVM Scaling

PVM jobs involve a certain of network traffic, sending multi-conformer molecules from the master to the slaves and sending results from the slaves back to the master. The single SZYBKI ligand optimization slave jobs are relatively fast compared to the network I/O time, so for scaling beyond 32 CPUs, additional effort is required in setting up a job. For low numbers of CPUs or for divide-and-conquer style applications, using `.oeb.gz` as the dbase file format is fine.

2.5 Open MPI

As of version 1.5.0, SZYBKI supports MPI-1, or Message Passing Interface 1, on all platforms except Microsoft Windows, Solaris & AIX. MPI, like PVM, enables distributing SZYBKI runs over multiple machines and processors.

SZYBKI uses the Open MPI implementation of MPI, found at <http://www.open-mpi.org>. Every version of SZYBKI includes a full Open MPI install, so no additional software is needed.

There are two requirements to run SZYBKI under Open MPI.

- Every machine in the cluster must have SZYBKI installed.
- The path to the top level `OpenEye bin` directory must be in the **PATH** environment variable on all machines, and before any other locations that may contain MPI executables (`orted`, `mpirun`, etc).

Note: While the OpenEye MPI install must be first in the path, it will not conflict with other installs. OpenEye puts two MPI related scripts in the top level `OpenEye bin` directory. The first, `oempirun`, is a replacement for `mpirun`. Given their different names, there will not be conflict. The `orted` command in the `OpenEye bin` directory is a wrapper script around the `orted` daemon. If MPI was not run with `oempirun` and an OpenEye application, it will fall back to the next instance of `orted` in the path.

2.5.1 Using Open MPI

Running SZYBKI under Open MPI makes use of a wrapper script `oempirun` located in the `OpenEye bin` directory.

To run SZYBKI under Open MPI:

```
prompt> oempirun [Open MPI options] szybki [SZYBKI Options]
```

For example, to run SZYBKI on 3 hosts `h1`, `h2` and `h3`:

```
prompt> oempirun -np 3 -host h1,h2,h3 szybki my_molecules.mol2
```

The above command line will run 3 SZYBKI processes on the hosts `h1`, `h2` and `h3`. The master machine, which needs a license, would be `h1`. It is important to note that the `szybki` passed to `oempirun` does not take a path.

2.5.2 Open MPI License

Most files in this release are marked with the copyrights of the organizations who have edited them. The copyrights below generally reflect members of the Open MPI core team who have contributed code to this release. The copyrights for code used under license from other parties are included in the corresponding files.

Copyright (c) 2004–2008 The Trustees of Indiana University and Indiana

University Research and Technology Corporation. All rights reserved.

Copyright (c) 2004-2008 The University of Tennessee and The University of Tennessee Research Foundation. All rights reserved.

Copyright (c) 2004-2008 High Performance Computing Center Stuttgart, University of Stuttgart. All rights reserved.

Copyright (c) 2004-2007 The Regents of the University of California. All rights reserved.

Copyright (c) 2006-2008 Los Alamos National Security, LLC. All rights reserved.

Copyright (c) 2006-2008 Cisco Systems, Inc. All rights reserved.

Copyright (c) 2006-2008 Voltaire, Inc. All rights reserved.

Copyright (c) 2006-2008 Sandia National Laboratories. All rights reserved.

Copyright (c) 2006-2008 Sun Microsystems, Inc. All rights reserved.
Use is subject to license terms.

Copyright (c) 2006-2008 The University of Houston. All rights reserved.

Copyright (c) 2006-2008 Myricom, Inc. All rights reserved.

Copyright (c) 2007-2008 UT-Battelle, LLC. All rights reserved.

Copyright (c) 2007-2008 IBM Corporation. All rights reserved.

Copyright (c) 1998-2005 Forschungszentrum Juelich, Juelich Supercomputing Centre, Federal Republic of Germany

Copyright (c) 2005-2008 ZIH, TU Dresden, Federal Republic of Germany

Copyright (c) 2007 Evergrid, Inc. All rights reserved.

Copyright (c) 2008 Institut National de Recherche en Informatique. All rights reserved.

Copyright (c) 2007 Lawrence Livermore National Security, LLC. All rights reserved.

Copyright (c) 2007-2008 Mellanox Technologies. All rights reserved.

Copyright (c) 2006 QLogic Corporation. All rights reserved.

Additional copyrights may follow

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer listed in this license in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

The copyright holders provide no reassurances that the source code provided does not infringe any patent, copyright, or any other intellectual property rights of third parties. The copyright holders disclaim any liability to any recipient for claims brought against recipient by any third party for infringement of that parties intellectual property rights.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR

A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

SZYBKI THEORY

3.1 SZYBKI Theory

3.1.1 Force Field

The MMFF potential expression is:

$$V_{MMFF} = \sum_b V_b + \sum_a V_a + \sum_s V_s + \sum_o V_o + \sum_t V_t + \sum_v V_v + \sum_c V_c$$

where the seven terms respectively describe bond stretching (*b*), angle bending (*a*), stretch-bending (*s*), out-of-plane bending (*o*), torsion (*t*), Van der Waals (*v*) and electrostatic (*c*) interactions. Their functional forms are given below.

Bond stretching

For a bond *b* between atoms *i* and *j* the stretching potential is:

$$V_b = 143.9325 \frac{k_{ij}}{2} \Delta r_{ij}^2 (1 + c_s^2 \Delta r_{ij} + 7/12 c^2 \Delta r_{ij}^2)$$

where k_{ij} is the force constant, Δr_{ij} is the difference between actual and reference bond lengths, and c_s is so called “cubic-stretch” constant for which the value is -2 \AA^{-1} .

Angle bending

The bending potential of a bond angle *a* made by the bonds between atoms *i, j* and atoms *j, k* is given by:

$$V_a = 0.043844 \frac{k_{ijk}}{2} \Delta \vartheta_{ijk}^2 (1 + c_b \Delta \vartheta_{ijk})$$

where k_{ijk} is the force constant, $\Delta \vartheta_{ijk}$ is the difference between actual and reference bond angles, and c_b is the so called “cubic-bend” constant for which the value is -0.4 rad^{-1} .

Stretch-bend interaction

The coupling between the stretching potential of two bonds forming a bond angle and bending that angle is described by:

$$V_s = 2.5121 (k_{ijk} \Delta r_{ij} + k_{kji} \Delta r_{kj}) \Delta \vartheta_{ijk}$$

where k_{ijk} and k_{kji} are force constants which couple stretches of *i-j* and *k-j* to the *i-j-k* bend respectively. Δr and $\Delta \vartheta$ are defined above.

Out-of-plane bending

For a trigonal center j , the potential of displacement for an atom l bonded to atom j out of plane i - j - k is:

$$V_o = 0.043844 \frac{k_{ijkl}}{2} \chi_{ijkl}^2$$

where k_{ijkl} is the force constant and χ_{ijkl} is an angle formed by the bond j - l and the plane i - j - k .

Torsion interactions

For every four bonded atoms i - j - k - l the torsion interaction is described by the term:

$$V_t = 0.5(V_1(1 + \cos \Phi) + V_2(1 - \cos 2\Phi) + V_3(1 + \cos 3\Phi))$$

where V_1 , V_2 and V_3 are constants depending on atoms i , j , k , l , and Φ is the dihedral angle formed by bonds i - j and k - l .

Van der Waals interactions

For a pair of atoms i and j separated by three or more bonds, where the distance between them is r_{ij} , MMFF adopts the following Van der Waals potential:

$$V_v = \epsilon_{ij} \left(\frac{1.07R_{ij}}{r_{ij} + 0.07R_{ij}} \right)^7 \left(\frac{1.12R_{ij}^7}{r_{ij}^7 + 0.12R_{ij}} - 2 \right)$$

where R_{ij} and ϵ_{ij} are defined as follows:

$$R_i = A_i \alpha_i^{0.25}$$

$$R_{ij} = 0.5(R_i + R_j)(1 + B(1 - \exp(-12\gamma_{ij}^2)))$$

$$\gamma_{ij} = (R_i - R_j)/(R_i + R_j)$$

$$\epsilon_{ij} = \frac{181.16G_iG_j\alpha_i\alpha_j}{(\alpha_i/N_i)^{0.5} + (\alpha_j/N_j)^{0.5}} \frac{1}{R_{ij}^6}$$

where α_i is atomic polarizability of atom i , B is 0.2 or 0.0 if one of the atoms is polar hydrogen, N_i and N_j are the Slater-Kirkwood effective numbers of valence electrons, G_i , G_j and A_i are scale factors.

Electrostatic interactions

The electrostatic interaction between two charged atoms i and j separated by at least three bonds is calculated from the standard Coulombic expression:

$$V_c = f \frac{332.0716q_iq_j}{D(r_{ij} + \delta)}$$

where D is the dielectric constant for which the default value is 1, q_i and q_j are the MMFF partial charges on atoms i and j , r_{ij} is the interatomic distance, δ is the "electrostatic buffering" constant of 0.05 Å. Scaling factor f is 0.75 for 1,4 interactions, and 1.0 otherwise.

3.1.2 Entropy evaluation

Ligand entropy is evaluated as a sum of configurational entropy (S_c) and solvation entropy (ΔS_s):

$$S = S_c + \Delta S_s$$

Configurational entropy

Configurational entropy is calculated as:

$$S_c = kN \left[1 + \ln \left(\frac{q}{N} \right) + \frac{T}{q} \frac{\partial q}{\partial T} \right]$$

where q is the conformation dependent partition function:

$$q = q_t \sum_{i=1}^{n_c} e^{-\frac{\epsilon_i}{kT}} q_{iv} q_{ir}$$

Here q_t , q_{ir} and q_{iv} are the translational, rotational and vibrational partition functions respectively, n_c is the number of unique conformations in the ensemble. All 3 partition functions are calculated from the classical statistical mechanics expressions which could be found in [McQuarrie-1976]. Vibrational frequencies for each conformation, needed for evaluation of q_{iv} are derived from diagonalization of a Hessian matrix obtained from Quasi-Newton optimization when convergence is achieved. Eigenvalues λ_i of the mass-weighted Hessian:

$$\mathbf{H}^m = \mathbf{M}^{-1/2} \mathbf{H} \mathbf{M}^{-1/2}$$

are converted into wavenumbers $\tilde{\nu}_i$ according to:

$$\tilde{\nu}_i = \frac{1}{2\pi c} \sqrt{\lambda_i}$$

Solvation entropy

Solvation entropy is split into electrostatic and hydrophobic parts:

$$\Delta S_s = \Delta S_{s,elec} + \Delta S_{s,hyd}$$

The electrostatic part of solvation entropy is divided into the bulk component and tight electrostatic polar solute - water interactions (hydrogen bonds). The bulk contribution is estimated from the temperature dependence of the solvent dielectric constant as:

$$\Delta S_{s,elec_bulk} = - \left(\frac{\partial \Delta G_s}{\partial \epsilon_{solv}} \right) \left(\frac{\partial \epsilon_{solv}}{\partial T} \right)$$

The second term of the electrostatic solvation entropy is estimated as a constant of 28 J/(mol K).

The hydrophobic term, $\Delta S_{s,hyd}$, is evaluated as:

$$\Delta S_{s,hyd} = - \left(\frac{\partial \Delta G_{s,hyd}}{\partial T} \right)$$

where $\Delta G_{s,hyd}$ consists of 3 components:

$$\Delta G_{s,hyd} = \Delta G_{cav} + \Delta G_{vdW} + \Delta G_{Ind}$$

describing the free energy of cavitation, solute-solvent van der Waals and inductive terms respectively. The cavity formation term is calculated from Scaled Particle Theory [Pierotti-1976]. Analytical expressions for ΔG_{vdW} and ΔG_{Ind} terms are also taken from the 1976 Pierotti review.

Protein-bound ligand entropy

Configurational entropy of a protein bound ligand is calculated totally as vibrational entropy for $3N$ modes, assuming that 3 rotational and 3 translational degrees of freedom of a solution ligand are transformed into low-vibrational degrees of freedom for the bound ligand.

Solvation entropy for a ligand in the active site is assumed to be a sum of its fractional value in solution determined by the percentage of the ligand surface exposed to the solvent, $f\Delta S_s$, and a partial desolvation entropy of the protein active site, ΔS_{des}

$$S_{protein} = S_v + f\Delta S_s + \Delta S_{des}$$

where f is the fraction of ligand surface exposed to the solvent. It is important to notice that $S_{protein}$ is not an experimentally measurable value, and that only the difference between $S_{protein}$ and $S = S_c + \Delta S_s$ might be compared with experimental binding entropy.

APPLICATION

4.1 Usage

4.1.1 Command Line Interface

A description of the command line interface can be obtained by executing SZYBKI with the `--help` option.

```
prompt> szybki --help
```

will generate the following output:

```
Help functions:
szybki --help simple      : Get a list of simple parameters
szybki --help all        : Get a complete list of parameters
szybki --help <parameter> : Get detailed help on a parameter
szybki --help html       : Create an html help file for this program
```

Required Parameters

The only required input parameter is defined in the option `-ligands`.

File Options

SZYBKI can handle molecular files in the following formats:

Table 4.1: SZYBKI file format support

File extension	Description
mol	MDL Mol File
mmd, mmod	Macromodel
mol2	Tripos Sybyl mol2 file
oeb	New Style OEBinary
pdb	Protein Databank PDB file
sd, sdf	MDL SD File
xyz	XMol XYZ format

-complex <filename>

Input file with 3D coordinates of a protein-ligand(s) complex in any format listed in the table above. All ligands contained in the molecule file `filename` will be optimized one by one in the presence of all others. If in

addition the option `-ligands` is used, all external ligands given with that option will be optimized as well. If there is spatial overlap with the internal ligand from the file, the internal ligand will be removed.

-fix_file <filename>

Text file `filename` should contain a list of molecule names followed by atom numbers to be fixed. An example of the command line for SZYBKI run with the use of this option is given below in the subsection *Examples of using SZYBKI*.

-flex_file <filename>

Text file `filename` should contain a list of molecule names followed by atom numbers to be optimized. All non-listed atoms will be fixed.

-heavy_rms

By default all atoms RMSD after optimization is reported in the log file. This option replaces the default with the heavy atoms RMSD.

-i <filename>

An alias to `-ligands`.

-in <filename>

An alias to `-ligands`.

-ligands <filename>

Molecular input file name containing 3D coordinates in any format supported by OEChem. In order to comply with the previous SZYBKI releases, an alias `-i` can be used instead of `-ligands`. This option might be used as a keyless parameter without the `-ligands` or `-i` keys but only when used last or next to last when the last option on the command line is `-out`. Warning: File name: `szybki_out.mol2` is used as the default output file name, so should be avoided as an input file name.

-loadPG <filename>

In the case when the electrostatic component of the protein-ligand interaction energy has been pre-calculated on a grid, this option forces SZYBKI to read the grid potential from the file `filename`, and use it for ligand optimization inside the protein.

-log <prefix>

Prefix of the log file name, `prefix.log`. If omitted, the prefix `szybki` is used by default. This option is aliased to `-l`.

-o <filename>

An alias to `-out`.

-out <filename>

Output file name, in any format supported by OEChem, for an optimized ligand. If not specified, `szybki_out.mol2` or `prefix_out.mol2` (when `-prefix` is used) will be generated. Alias `-o` can be used instead of `-out`. When options `-ligands` and `-complex` are used together, internal ligands from the protein-ligand complex will be output first. Can be used as a keyless parameter without the `-out` key when it is last on the command line just after the `-ligands` (or `-i`, or `-in`) option.

-out_complex <filename>

Protein-ligand complex for partially optimized protein will be saved in a file named `filename`

-out_protein <filename>

Partially optimized protein will be saved in a file named `filename`.

-param

Command line options will be read from the specified file. This file may have been generated from a previous run or may be constructed de novo. The default name of the file is `szybki.param`. Any parameter in the parameter setup file is superseded by the parameter on the command line. For example running `szybki -param szybki.param -i my.pdb` will perform calculations for the molecule `my.pdb` while using all other parameters taken from `szybki.param`.

-prefix <pn>

Replaces `szybki` prefix in `.log`, `.rpt` (report), `.status`, `.param` and `_out.mol2` files, with the input string `pn`.

-protein <filename>

Protein input file with 3D coordinates in any format supported by OEChem. Usually works in combination with the `-ligands` option. No removal of pre-existing ligands in file `filename` will be done as in the case of the `-complex` option. When used without `-ligands` option, two scenarios will be followed depending on the presence of ligands in the file. If ligands are present, all of them will be optimized as in the case of `-complex` option. If not, the optimization of the entire protein will be attempted. The latter is not recommended however, because for larger proteins extreme memory usage can occur causing crashes. This option can be aliased with `-p` for compatibility with previous SZYBKI releases.

-report

An output file in tabular form of final energy components for every molecule/conformer will be generated. The name of the file is fixed as: `prefix.rpt` where `prefix` is `szybki` by default or determined by the `-prefix` option.

-reportFile <filename>

Sets user selected file name, `filename`, for the tab file. Supersedes the `-report` option described above.

-savePG <filename>

Saves potential grid in the file named `filename`. Allows a significant saving in CPU time for runs when PB or Coulomb grids are used to optimize a number of ligands inside the same protein.

-sdtag <string>

In the case when the output file is in SD format an energy tag can be added. By default, the `string` is empty which means that all relevant tags are added. For single molecules, this means `Total_energy` and all MMFF terms and solvation terms that are not zero, and for a protein-ligand system this means `Ligand-Protein Energy` and all of its components. The following potential components can be used for `string`, exactly as shown:

```

1.MMFF VdW
2.MMFF Coulomb
3.MMFF Bond
4.MMFF Bend
5.MMFF StretchBend
6.MMFF Torsion
7.MMFF Improper Torsion
8.L MMFF VdW
9.L MMFF Coulomb
10.L MMFF Bond
11.L MMFF Bend
12.L MMFF StretchBend
13.L MMFF Torsion
14.L MMFF Improper Torsion
15.P MMFF VdW
16.P MMFF Coulomb
17.P MMFF Bond

```

- 18.P MMFF Bend
- 19.P MMFF StretchBend
- 20.P MMFF Torsion
- 21.P MMFF Improper Torsion
- 22.PL MMFF VdW
- 23.PL MMFF Coulomb
- 24.Ligand-Protein Energy
- 25.Sheffield Solvation
- 26.Constraint Potential
- 27.PB Solvent
- 28.Area Solvent
- 29.__VdW
- 30.__Coulomb
- 31.__Protein desolv
- 32.__Ligand desolv
- 33.__Solvent screening
- 34.__Grid Coulomb
- 35.__Exact Coulomb

If parameter string does not correspond to one of the available energy terms, no tag will be written. Tags starting with L, P, PL and __ are:

- L: ligand intra-molecular terms only
- P: protein intra-molecular terms only
- PL: protein-ligand inter-molecular terms only
- __: protein-ligand interaction terms

-silent

By default the names of the processed molecules are displayed. The use of the option *-silent* will suppress this output.

-verbose

An extensive general output will be generated containing initial and final energy and gradient data, as well as an optimization report. This option is available only for single-processor and MPI modes.

Potential Function Options

-MMFF94s

Modified set of MMFF94 parameters developed in 1999 by Halgren ([Halgren-1999-1], [Halgren-1999-2]) will be used.

-am1bcc

AM1BCC charges ([Jakalian-2002]) for PB or Sheffield are calculated and used for every conformation.

-exact_vdw

By default the VdW protein-ligand interaction is calculated with the use of lookup table in order to speedup the calculations. This option allows using the exact analytical VdW potential for the optimization of a ligand in the protein binding site.

-harm_constr1 <k>

Constrained potential of the form: kr^2 will be imposed on all heavy atoms, where k is the force constant. By default no constraint is applied ($k = 0$), and the upper allowed limit is 1000 kcal/(mol Å²).

-harm_constr2 <d>

Constrained potential of the form: $V = k(r - d)^2, r > d$ and $V = 0, r \leq d$ will be imposed on all heavy atoms, where d is the constraining distance in angstroms. Can be used only together with `-harm_constr1`. Default value is 0, while the upper allowed value is 5 Å.

-harm_smarts <p>

All atoms which belong to SMARTS pattern p will be constrained. For example input parameter “-harm_smarts cO” will result in constraining all aromatic carbon atoms and oxygen atoms bonded to them. Must be used in conjunction with `-harm_constr1`.

-inner_dielectric <d>

The default value for the protein or ligand dielectric constant of 1.0 can be changed to a user selected value d . This option is aliased as `-protein_dielectric`. The upper allowed limit is 20.0.

-mod_vdw

Regular MMFF *Van der Waals interactions* equation will be replaced with:

$$V_{vdw} = \begin{cases} \epsilon_{ij} \left(\frac{1.07R_{ij}}{r_{ij}+0.07R_{ij}} \right)^7 \left(\frac{1.12R_{ij}^7}{r_{ij}^7+0.12R_{ij}} - 2 \right) & \text{for } r_{ij} < R_{ij} \\ -\epsilon_{ij} & \text{for } r_{ij} \geq R_{ij} \end{cases}$$

in which no attractive VdW forces are present. This type of VdW potential prevents so called “hydrophobic collapse”.

-current_charges

During optimization of molecules in solution with the use of the Poisson-Boltzmann solvation model, free energy of solvation and solvent forces can be calculated with the use of atomic partial charges other than MMFF. Option `-current_charges` allows the use of partial charges read from the input molecular file. In the case where the ligand is optimized inside a protein, protein-ligand electrostatic interactions (Coulomb and/or Poisson-Boltzmann) could be calculated based on ligand and protein partial charges read from the molecular input file(s). When input partial charges are not found, the MMFF94 partial charges will be used. This option is aliased to `-mol2charges`

-mol2charges

An alias to `-current_charges`.

-neglect_frozen

After an optimization with frozen terms, the default behavior calls for a full single-point calculation of the whole system. The `-neglect_frozen` option will skip this final calculation of the whole system, thus yielding results for only the non-frozen pieces. When there is a large frozen section, this can drastically reduce memory and cpu usage.

-noCoulomb

Electrostatic terms defined in the *Electrostatic interactions* section will be excluded from the force field potential. This option might be useful to prevent generation of folded structures.

-protein_elec <m>

This option provides 4 choices for calculating protein-ligand interaction energies. In all cases the MMFF VdW potential is used. Option `None` (`none`) eliminates electrostatic interactions. Values of m set at `ExactCoulomb` and `GridCoulomb` result in the usage of MMFF Coulomb exact potential and digitized on the grid respectively. Option $m = PB$ provides a more realistic potential which accounts for solvent forces, calculated according to the

Poisson-Boltzmann (PB) model at every iteration step. This option requires substantially higher CPU time, particularly for large proteins. By default `m = None`.

-protein_vdw <r>

Calculation of VdW protein-ligand interaction energy will be limited to a sphere of radius `r`. The default value is 18.0 Angstroms. In many applications a value as small as 10 angstroms can be used with essentially no effect on the final optimized ligand geometry. Legal range is 5-500 Å.

-salt <c>

Allows to all PB calculations to be performed at specified salt concentration in M, up to 0.08M. By default salt concentration is zero.

-shefA <a>

Parameter `a` in the Sheffield solvation potential given in option `-sheffield`. If the option is not used a value of 1.553149 is assigned ([Grant-2007]).

**-shefB **

parameter `b` in the Sheffield solvation potential given in option `-sheffield`. If the option is not used a value of 0.735694 is assigned ([Grant-2007]).

-sheffield

Usage of this option will result in adding additional electrostatic term of the form: $f_\epsilon / (8\pi\epsilon_0) \sum_{i,j} q_i q_j / \sqrt{(ar^2 + bR_i R_j)}$ where $f_\epsilon = (1/\epsilon_{in} - 1/\epsilon_{out})$, q_i, q_j are partial charges and R_i, R_j are Van der Waals radii of atoms `i` and `j` in order to mimic the solution environment ([Grant-2007]). This option might help to preserve unfolded structures during optimization.

-solv_dielectric <d>

Allows to change the default value for solvent dielectric constant used for Poisson-Boltzmann and Sheffield solvation energy calculations is 80. Allowed range is 1.0 to 100.

-solventCA <s>

This option can only be used in combination with `-solventPB` or `-sheffield`. It causes inclusion of a molecular surface solvation term (sometimes called cavity solvation term) in the total energy. The value of parameter `s` (microscopic surface tension coefficient) is in the range 0.005 - 0.030 kcal/(mol Å²). This option is aliased as `-solventMA`, for compatibility with previous releases.

-solventPB

For optimization of small molecules in solution, the electrostatic part of molecule-solvent interactions will be calculated using Poisson-Boltzmann model.

-strict

Enforces strict atom typing. This is a default behavior. When the value of the flag is selected to be `false`, this enforcement is removed.

-vdw_cutoff

Sets distance for intramolecular VdW interactions. By default all atom pairs in the molecules contribute to the molecule's VdW energy. The legal range is 5 - 500 Å².

Optimization Options

-conj

Conjugate gradient optimization will replace the default quasi-Newton BFGS method.

-fix_smarts <p>

All atoms which belong to SMARTS pattern `p` will be fixed. For example, input parameter `p = "[!#1]"` will fix all heavy atoms and optimize all hydrogen atom positions.

-grad_conv <c>

Optimization is terminated when the rms gradient reaches the input value `c` unless is finished earlier because of

other reasons. When omitted the default convergence criteria is 0.1 on gradient vector norm: $\sqrt{\sum_i g_i g_i}$, where g_i is i -th gradient component.

-largest_part

Calculations are performed only for the largest fragment of the noncovalent input complex. For example when the input file contains coordinates for the salt, $[Large_cation^+] \cdot [Cl^-]$, the use of the *-largest_part* will cause the Cl^- anion to be ignored. By default calculations are done for the entire complex.

-max_iter <m>

Optimization will be terminated when the number of iteration cycles reaches input number m . The default value is 1000.

-no_opt

A single point calculation will be performed i.e. no optimization of ligands or the protein will be carried out.

-opt_cart

Optimization of Cartesian atomic coordinates will be done. This is the default optimization type for molecules in vacuum or in solution.

-opt_solid

Optimization of ligand position inside the protein receptor with frozen ligand internal ligand coordinates. Option is not valid and ignored without an input protein. Because it is the simplest and least expensive ligand optimization in the protein, it is the default behavior. The option is therefore only formal and redundant.

-opt_torsions

Optimization of free rotor torsions for molecules in vacuum or torsions and translational/rotational degrees of freedom for ligands inside protein receptors. In the case when the ligand has no rotor torsions, rigid-body optimization will be performed.

-polarH <r>

This option allows partial flexibility of a protein receptor in the optimization of a ligand inside the protein binding side. All polar protein hydrogens distanced by r from the ligand will adjust their position upon energy relaxation. Main chain amide NH hydrogens of peptide groups are not included. The molecular potential used for optimization consists of three components:

- 1.Ligand intra-molecular MMFF potential
- 2.Protein-ligand potential
 - 2.1. MMFF terms for interaction between polar protein hydrogens and the ligand
 - 2.2. Interaction between fixed protein atoms and the ligand
- 3.Protein intra-molecular potential
 - 3.1. MMFF terms involving polar protein hydrogens and atoms up to three bonds apart.
 - 3.2. Interaction between the rest of the protein and flexible polar hydrogens

The sum of components 2.2 and 3.2 might be called “protein-pseudoligand” interaction and is evaluated according to the model selected with the *-protein_elec* option above.

-residue <r>

Similar to polarH. This option allows for flexibility of complete residues having at least one atom within distance r of any atom of the ligand. There is no upper value of the r parameter, however using a large value will cause even distant residues to be flexible which could lead to poor results. The intention of this option is to allow only residue atoms interacting directly with the ligand to rearrange upon optimization.

-sideC <r>

Similar to polarH. This option allows for flexibility of all side chains having at least one atom within distance r of any atom of the ligand. Same comment as in the *-residue* above applies with respect to side chains atoms.

-strip_water

Causes removal of water molecules from the input protein when options *-protein* or *-complex* are used.

Thermodynamics calculations options**-entropy <type>**

Estimation of the entropy of a ligand will be done. Parameter *type* can take three values “None” (default), “QN” (Quasi-Newton Hessian) and “AN” (analytical). Input ligand is assumed to be in the form of a multiconformation molecule. The environment is determined by the option *-sheffield* (solution), *-protein* (in the binding site) or none of those two indicating a gas-phase.

-t

Sets the temperature (in C) of the system for entropy estimation. Default temperature is 25C (298K).

-sfp

Change the value of macroscopic surface tension coefficient from its default value of 6.0 cal/(mol Å²). This quantity is used for the calculation of the bound ligand entropy.

-rws

Removes explicit water molecules from solvent accessible surface calculation during the evaluation of the bound ligand entropy. By default explicit water molecules present in the protein binding site are treated as part of the protein receptor.

PVM Options**-pvmconf <hosts.txt>**

hosts.txt is a text file which specifies PVM processor configuration. For every host in the cluster it should contain line: *host host_name n* where *n* is the number of processors on the host.

4.2 Examples of using SZYBKI

This section has several examples of typical SZYBKI command-line executions.

4.2.1 Free molecule optimization

Protein preparation

X-ray protein structures rarely contain all hydrogen atoms coordinates. It is necessary therefore to prepare the protein structure before any protein-bound ligand calculations are done.

```
prompt> szybki -fix_smarts "[!#1]" -max_iter 2500 -prefix ex1 -conj -neglect_frozen protein.pdb
proteinH.mol2
```

In this command the input protein file *protein.pdb* is assumed to contain all heavy atoms. Option *-fix_smarts* fixes all heavy atoms at their positions given in the input file, while the option *-max_iter* increases in the total number of iterations beyond the default value of 1000 in order to make sure that the added hydrogen atoms are correctly optimized. Option *-conj* selects the conjugate gradient type of optimization because the default quasi-Newton method for large systems might be less efficient, or in the extreme cases of very large proteins might even fail if there is insufficient memory. Thus the usage of *-conj* is critical for such large systems. Option *-neglect_frozen* removes the single-point all-atoms energy calculation at the end of the run. The output protein file is *proteinH.mol2*. Note: All ionizable residues are assumed to be in their standard ionization states. If other than standard ionization states are needed, the input file has to be edited accordingly before the SZYBKI run is done.

Ligand optimization in solution

The equilibrium geometry of a molecule in solution is sometimes drastically different than in vacuum. This is because of solvent forces. The most efficient way to optimize a large set of ligands in solution is the usage of the `-sheffield` option.

```
prompt> szybki -sheffield -amlbcc -prefix ex2 ligands.sdf ligands_optimized.sdf
```

In the run above, a set of molecules from the input file `ligands.sdf` is optimized with the MMFF94 force field in solution. AM1BCC charges are optionally used for the Sheffield solvation model. Skipping the `-amlbcc` would result in the usage of MMFF94 atomic charges. Optimized structures are written to the `ligands_optimized.sdf` file. One can also optimize the ligand in solution using a physically more rigorous model by solving the Poisson-Boltzmann equation at every step. The command line below is equivalent to using the widely known PBSA model:

```
prompt> szybki -solventPB -solventCA 0.005 -prefix ex3 ligands.sdf ligands_optimizedPB.sdf
```

Optimization with fixed atoms

This example illustrates the usage of the option `-fix_file`.

```
prompt> szybki -fix_file f.txt -prefix ex4 mymolecule.mol2 molecule_opt.sdf
```

Text file `f.txt` contains molecules names followed by atom numbers in the (0,1...N-1) numbering convention. For example the `f.txt` file contains the following 6 lines:

```
Molecule1
0
15
Molecule2
5
9
```

informs SZYBKI that atom 0 and 15 for molecule `Molecule1` and atoms 5 and 21 for molecule `Molecule2` should be fixed during optimization. In the case a common substructure has to be fixed, one can use the option `-fix_smarts`:

```
prompt> szybki -fix_smarts c1cccc1 -prefix ex4a mymolecule.mol2 molecule_opt1.sdf
```

where `c1cccc1` is the SMARTS string which defines the substructure to be fixed.

4.2.2 Bound molecule optimization

Ligand optimization in the active site

We need to distinguish between two cases. One is screening of a large amount of docked ligands into a protein receptor, and the second is lead optimization. In the first case instead of cpu expensive optimization with the use of PB solvent forces, we recommend a two step SZYBKI calculation:

```
prompt> szybki -p p.mol2 -in lig.oeb -out optlig.oeb -opt_cart -prefix ex5
-protein_elec ExactCoulomb
prompt> szybki -p p.mol2 -in optlig.oeb -no_opt -protein_elec PB -prefix ex6 -log lig
```

where `p.mol2` is the protein file, `lig.oeb` and `optlig.oeb` are input and optimized output ligand files respectively, possibly with many conformations and poses. In the first run the input docked ligands in `lig.oeb` are minimized in the (Coulomb + VdW) MMFF94 potential in full Cartesian coordinates. In the second run, a single point calculation (option `-no_opt`) is done with PB protein-ligand electrostatics, which calculates protein-ligand interaction energy including solvent effects. The result of the calculation is stored in the `lig.log` file. The fragment of the log file showing protein-ligand interaction results is shown below. The entry marked `Lig-Protein Interaction Energy` is the sum of all lines starting from double underscore. All values are in kcal/mol.

```
Overall Ligand-Protein Interaction terms:
__VdW                               16.51126
__Coulomb diel=1.0                   -429.58420
__Protein desolv (PB)                 46.84836
__Ligand desolv (PB)                 67.28700
__Solvent screening (PB)             276.91150
-----
Overall Lig-Prot Interaction          -22.02608
```

In the case of lead optimization, a complete optimization which includes solvent forces and possibly partial relaxation of the protein residues in the direct proximity to the ligand is recommended:

```
prompt> szybki -p p.mol2 -in lig.oeb -out optligPB.oeb -residue 2 -prefix ex7 -protein_elec PB
-log ligPB
```

where 2 is the distance from the ligand (in Angstroms).

4.2.3 Entropy calculation

Ligand entropy calculation in solution

Assuming that the molecule conformations are given in the `molecule.oeb` file, the command line execution:

```
prompt> szybki -entropy QN -sheffield -prefix ex8 molecule.oeb
```

will result in the standard molar entropy estimation of the input molecule in solution. The meaning of the parameter value `QN` is explained in the description of the option `-entropy`. Option `-sheffield` informs SZYBKI that the calculation will be done in solution with the use of the Sheffield Solvation Model. The entropy results will be written in the `ex8.log` file. If the input file `molecule.oeb` does not contain the complete ensemble of conformations, the resultant entropy value may not be accurate. We recommend therefore that the input conformations are generated with the OpenEye tool OMEGA, with the `-rms` option set at 0.1 value, and `-maxconfs` set to at least 1000.

Protein-bound Ligand entropy calculation

Run:

```
prompt> szybki -p protein.mol2 -entropy QN -prefix ex9 ligand.mol2
```

will generate the estimation of ligand entropy in the active site of the protein. Input file `ligand.mol2` might be a docked structure or preferentially the X-ray structure.

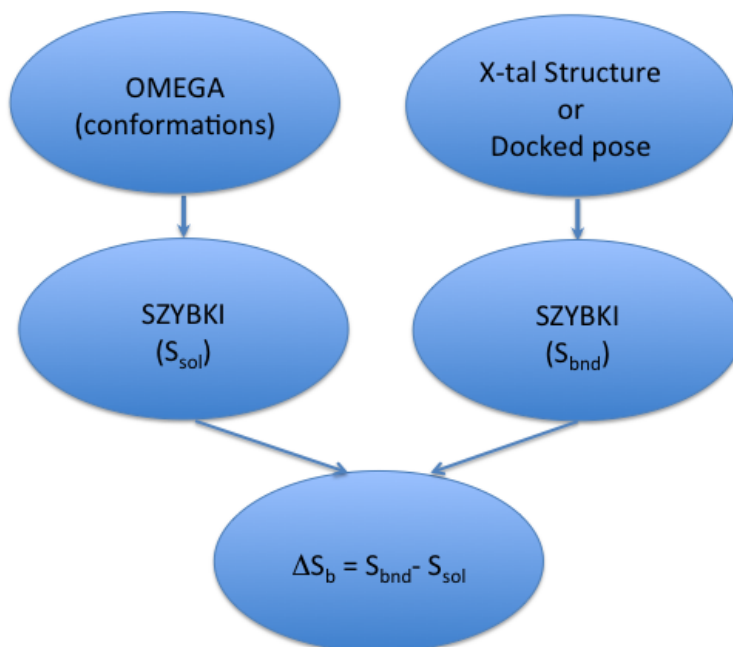


Figure 4.1: **Binding Entropy Estimation**
A basic workflow for binding entropy estimation.

Binding entropy calculation

A workflow for estimating binding entropy is shown in the figure below:

As Figure 4.1 shows, estimating binding entropy, ΔS_b , requires two separate SZYBKI runs: one for the estimation of ligand entropy in solution (S_{sol}) and another one which calculates protein-bound ligand entropy (S_{bnd}). Starting from the SZYBKI 1.7.0 release it is possible to calculate binding entropy (ΔS_b) in a single SZYBKI run:

```
prompt> szybki -complex complex.oeb -entropy QN -prefix ex10 -sheffield -in ligand.oeb
```

where `complex.oeb` is a molecular file of the protein-ligand complex and `ligand.oeb` contains an ensemble of conformations of the ligand in solution. Note: The docked ligand in the `complex.oeb` file must represent a single pose and must be the same molecule as in the `ligand.oeb` input file. In the case where two or more docked poses are used, calculation of binding entropy requires two separate runs as shown on Figure 4.1.

RELEASE NOTES

5.1 SZYBKI 1.7.0

5.1.1 New features

- An extension of the MMFF94 force field for three coordinated boron compounds is offered in this release. Most compounds containing B-X bonds where X=C,N,O,S, and H are covered with the following exceptions: X=N(imine),N(sulfonamide), N(pyridinium) and N(quaternary). Also not supported are compounds in which boron is bonded to X=F,Cl,Br,I,B and Si, or makes a bond angle BYX. Compounds in which boron is a part of four-membered rings of B1CCC1 type are also not available in the current parameterization because their existence is questionable: Ab initio calculations at the MP2/6-31G** level failed to identify stable structures for them (highly polar structures in which boron is four-coordinated are formed).
- New input flag `-solv_dielectric` is added. It controls solvent dielectric for all PB and Sheffield calculations. Default value is 80. Flag `-prot_dielectric` is changed into `-inner_dielectric`. Previous flag `-prot_dielectric` can be used.
- New option flag `-salt x` which sets the salt concentration for PB calculations is added. x is concentration in mM. Default value is 0.
- New option `-flex_file` which allow to specify only those atoms which will be optimized is added.
- Starting from the current release, binding entropy ($T\Delta S$), can be calculated in a single run:

```
szybki -entropy QN -complex pl_complex.oeb -ligands ligands.oeb
```

where `pl_complex.oeb` is a protein-ligand complex file, and `ligands.oeb` contains an ensemble of ligand conformations in solution.

- Entropy calculation of protein-bound ligands has been modified. Specifically, the fraction f of ligand surface exposed to the solvent used in the entropy component term: $f\Delta S_s$, is calculated currently as $f = A_{Lexp}/A_L$, where A_{Lexp} and A_L are ligand solvent accessible surfaces in the protein complex and solution respectively. The reason for a change is the recent OpenEye internal study which show that the above fraction f calculated according to the equation: $f = 0.5(A_L - A_P + A_{PL})/A_L$ ([Wlodek-2010]) is for many protein-ligand complexes overestimated.
- The default value of microscopic surface tension used for protein entropy desolvation calculation is now $6 \text{ cal}/(\text{mol}\text{A}^2)$ (previously default value was set at $5 \text{ cal}/(\text{mol}\text{A}^2)$). The flag `-sfp x`, where x is microscopic surface tension for protein desolvation entropy calculation, allows to overwrite the default value.
- Two more flags have been introduced: `-t x` and `-rws`. The first of them sets the temperature x in C for entropy calculations (the default value is 25C). The second flag removes water molecules from all solvent accessible surface calculations used for the evaluation of protein-bound ligand entropy, in the case the input protein contains water molecules.

- More entropy terms are reported in the log file. In earlier SZYBKI versions only the total entropy was reported. Starting from 1.7.0 release in addition to total entropy, configurational, solvation (or partial solvation) and protein desolvation entropy terms are reported.
- flag `-mol2charges` is replaced with `-current_charges`. `-mol2charges` is now an alias to the new flag.
- The format of the log file has been changed. Its readability is significantly improved.
- A new option `-out_complex` is added. It allows to output a protein-ligand complex after ligand optimization in partially flexible protein.

5.1.2 Bug fixes

- Optimization of polar hydrogens with the `-polarH` option when the input protein was in the PDB format resulted occasionally in mangling PDB atom names in the protein output file. This bug has been fixed.
- Combination of flags `-complex`, `-protein_elec PB` and `-exact_vdw` caused a crash on Windows platform. This bug has been fixed.
- Previous Szybki versions produced the parameter and status files even when Szybki was run without any user option. This problem has been fixed in the current release.
- Optimization of ligands in the partially flexible protein receptors in the case when the input file contained more than one ligand, stopped after processing the first ligand. This bug has been fixed.
- Information on the selected user options was written to the stdout. That was making piping impossible. This behavior has been eliminated in the current release.
- Input molecular files with badly specified geometry (with atom clashes), were not handled properly. Starting from the current release such an input is caught and the processing of such a molecule is skipped.
- Optimization of large molecules (e.g. optimization of hydrogen positions in very large proteins) can lead to memory exhaustion on some platforms, due to too large amount of VdW interactions or inability to allocate Hessian matrix in the quasi-Newton optimization. Previous Szybki releases terminated on such cases with the `bad_alloc` crash. This has been corrected by terminating the run and generating information on the mentioned above cases, so the user might decrease the number of VdW interactions or switch to conjugate gradient optimization method.

5.2 SZYBKI 1.5.1

5.2.1 Bug fixes

- Ligand optimization in the active site with the use of the flag `-protein_elec` set at `PB` ended prematurely. This problem has been fixed.
- In the previous (1.5.0) release the entropy calculation of the protein-bound ligand worked only when a single molecule in the ligand input file was present. In the case when the ligand input file contained two or more molecules, the calculation was performed for the first molecule and died on next one with segmentation fault. This bug is fixed.
- Calculated entropy of the protein-bound ligand was underestimated by up to 0.5 kcal/mol due to the underestimation of the buried protein surface area upon binding. This error is corrected in the current release.
- Using options: `-exact_vdw` and `-protein_elec` set at `PB` resulted in minimization with Coulomb and VdW potential only. This has been corrected in the current release.
- Parameter file was written without the user command line flags. This bug is fixed.

- A tiny memory leak in the calculations of protein-ligand interaction has been removed.
- Two subsections in documentation has been added. That includes the theory of entropy estimation adopted by SZYBKI and several examples of command line SZYBKI executions.

5.3 SZYBKI 1.5.0

5.3.1 New features

- Starting from the current release, certain classes of divalent selenium compounds can be handled by Szybki. Currently only some compounds in which selenium is bonded to the hydrogen or carbon atoms can be processed. Those compounds include selenols (RSeH) and selenides (RSeR1) in which Se is bonded to alkyl or aromatic carbon atoms, with the exception when Se makes a bond with alkyl C which belongs to the 3- or 4-membered ring.
- Previous Szybki releases attempted to assign the “closest” MMFF94 atom type to those atoms for which a unique MMFF94 type could not be found. Starting from the 1.5.0 release this feature is removed, so consequently the input molecules containing such atoms will not be processed.
- Ligand RMSD which is reported in the log file, can optionally refer to the heavy atoms only. The new option used for that purpose is `-heavy_rms`.
- Entropy calculations of a ligand in different environments can be performed based on the methods described in the recent paper of Wlodek *et. al.* ([Wlodek-2010]). This is done with the option `-entropy` described in the section 4.1 of this manual.
- Default VdW protein-ligand potential used for the optimization of a ligand inside the protein binding site is precalculated in the form of a lookup table. This is done for the purpose of speed. The exact VdW protein-ligand potential can be used with the option `-exact_vdw`.

5.3.2 Bug fixes

- Memory leak in ligand torsion optimization inside the partially flexible protein receptor was removed.

5.4 SZYBKI 1.3.4

5.4.1 New features

- Calculation of the constant potential terms at the end of adapted optimization can be optionally eliminated, by the use of the `-neglect_frozen` flag. This feature allows for less memory and cpu usage in the case of partial optimization of large molecules.
- The corresponding atom symbols, bond symbols, and torsion angles are now printed on a line under the matching smarts pattern.

5.4.2 Bug fixes

- Input proteins with only partial information about residues were not handled properly. As a result, none of the side chains which were supposed to be made flexible with the use of an option `-sideC` were optimized. This problem was fixed.
- A bug causing memory corruption in the case of ligand optimization in partially flexible protein receptor is fixed.

- Residues with non-standard name `CYX` were not properly handled in the situation when their side chains were included into the flexible part of the protein receptor. This bug has been fixed.

5.5 SZYBKI 1.3.3

5.5.1 Bug fixes

- Until current release, molecular output files in the `.oeb` format had MMFF aromaticity. In practice, such a situation lead to the visual change between the input and output aromaticity of some molecules. Starting from now OpenEye aromaticity model is used in the `.oeb` output files.

5.6 SZYBKI 1.3.2

5.6.1 Bug fixes

- A bug which occasionally caused errors in fixing a subset of atoms, was fixed.
- When using PVM, Szybki now runs a mechanism for checking versions used by master and slaves. This mechanism prevents the situation of different versions of Szybki running on the master and slaves.
- Molecules containing atoms for which MMFF types are not properly typed are not processed. In previous versions attempts were made to assign for such atom the closest MMFF type, however this procedure almost always lead to errors. This situation occurred for example in the case of bad `pdb` files. For proteins, better diagnostics (residue name and number, atom name) are included as output.
- Reading protein data in `.oeb` binary format containing additional untagged information (FRED receptor files) is allowed. Until now using such a file in `pvm` mode caused a fatal error.

5.7 SZYBKI 1.3.1

5.7.1 Bug fixes

- The combination of options `-complex if1` and `-ligands if2` run in the `pvm` mode caused incorrect behavior in the case when the preexisting ligand in the macromolecule (in the file `if1`) overlapped with one or more ligands from the input file `if2`.
- Run time license for the shape toolkit is added. It is required when the combination of options `-complex if1` and `-ligands if2` is used.
- Redundant warnings generated upon `-protein_elec none` and `-strip_water` were removed.
- Spelling in the log file was fixed.
- Year 2008 was added to the Szybki banner.
- In the documentation description for the `-fix_smarts` option an example was added which illustrates the adjustment of hydrogen atom positions.
- Examples of log files generated upon ligand optimization in the partially flexible protein followed by a single point PB calculations are given.

5.8 SZYBKI 1.3.0

5.8.1 New features

- Optimization using conjugate gradient method was added as an alternative to the default quasi-Newton method with the BFGS Hessian update scheme. In Szybki toolkit this is done by the usage of a new function `void SetOptimizerType(unsigned int type)`. In Szybki application a flag `-conj` has been added for that purpose.
- Default value of the protein dielectric constant is 1. Previous default value for that parameter was 2, however it was not consistent with the default value of MMFF94 force field.
- Protein-electrostatic models in Szybki application are selected with descriptive strings. Available options are:
 - `None` - no electrostatic potential used (default option)
 - `ExactCoulomb` - exact MMFF Coulomb potential
 - `GridCoulomb` - MMFF Coulomb potential on grid
 - `PB` - PP solvent forces used at every step

5.8.2 Bug fixes

- A bug causing crash in the case of combined usage of partial protein flexibility and protein electrostatic model `OEProteinElectrostatics::SolventPBForces` was fixed.
- A bug causing different behavior and/or occasional crash depending of the order of calls: `SetProtein(const OEChem::OEMolBase&)`, `SetRunType(unsigned int)` and `SetResidueFlexibility(double)` has been fixed.
- Modification of the potential function after the usage of coordinate adaptors could occasionally result in crash, for example when function `FixAtoms()` was followed by `SetRemoveCoulombTerms()`. This bug was fixed.
- Running in the `pvm` mode requires now setting the `PVM_PATH` variable, which points at the directory where the Szybki script is located. For example if Szybki was untared in the directory `mydir`, then using `bash` shell one can set the above variable with: `export PVM_PATH=mydir/openeye/bin`

5.9 SZYBKI 1.2.2

5.9.1 New features

- Two new functions are added to the API:

```
void SetResidueFlexibility(double dist); double GetResidueFlexibility() const;
```

These allow the user to introduce and query for the flexibility of entire residues around the ligand.

5.9.2 Bug fixes

- In the case when protein electrostatic model was set at `OEProteinElectrostatics::NoElectrostatics` and partial protein flexibility was allowed with the `void SetSideChainsFlexibility(double)` or `void SetPolarHFlexibility(double)` functions, Coulombic terms describing the interaction between the ligand and flexible protein atoms were included during optimization. This was inconsistent with the meaning of `NoElectrostatics`, therefore those Coulombic terms were eliminated in the current release

- Using the function `void SetSideChainsFlexibility(double)` resulted occasionally in double counting of some residues. This bug was fixed.
- Fixing atoms with a function `bool FixAtoms(std::string)` where the passed string is SMARTS pattern, was based on MMFF aromaticity types. Starting from the current release OE aromaticity model is used.

BIBLIOGRAPHY

- [Geist-1994] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R. and Sunderam, V., **PVM - Parallel Virtual Machine: A User's Guide and Tutorial for Networked Parallel Computing**; *MIT Press*, 1994
- [Halgren-1996-1] T.A. Halgren, **Merck Molecular Force Field: I. Basis, Form, Scope, Parameterization and Performance of MMFF94**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 490-519, 1996
- [Halgren-1996-2] T.A. Halgren, **Merck Molecular Force Field: II. MMFF94 van der Waals and Electrostatic Parameters for Intermolecular Interactions**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 520-552, 1996
- [Halgren-1996-3] T.A. Halgren, **Merck Molecular Force Field: III. Molecular Geometries and Vibrational Frequencies**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 553-586, 1996
- [Halgren-1996-4] T.A. Halgren, **Merck Molecular Force Field: IV. Conformational Energies and Geometries for MMFF94**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 587-615, 1996
- [Halgren-1996-5] T.A. Halgren, **Merck Molecular Force Field: V. Extension of MMFF94 using Experimental Data, Additional Computational Data and Empirical Rules**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 616-641, 1996
- [Halgren-1999-1] T.A. Halgren, **MMFF VI. MMFF94s Option for Energy Minimization Studies**, *Journal of Computational Chemistry*, Vol. 20, No. 5, pp. 720-729, 1999
- [Halgren-1999-2] T.A. Halgren, **MMFF VII. Characterization of MMFF94, MMFF94s and Other Widely Available Force Fields for Conformational Energies and for Intermolecular Interaction Energies and Geometries**, *Journal of Computational Chemistry*, Vol. 20, No. 5, pp. 730-748, 1999
- [Jakalian-2002] A. Jakalian, D.B. Jack and C.I. Bayly, **Fast, efficient generation of high-quality atomic charges. AM1-BCC model: II. Parameterization and validation**, *J. Comput. Chem.*, Vol. 23, pp. 1623-1641, 2002
- [Grant-2007] J.A. Grant, B.T. Pickup, M.J. Sykes, C.A. Kitchen and A. Nicholls, **A Simple Formula for Dielectric Polarization Energies: The Sheffield Solvation Model**, *Chem. Phys. Letters*, Vol. 441, pp. 163-166, 2007
- [Wlodek-2010] S. Wlodek, A.G. Skillman and A. Nicholls, **Ligand entropy in gas-phase, upon solvation and protein complexation. Fast estimation with Quasi-Newton Hessian**, *J. Chem. Theory Comput.*, Vol. 6, pp. 2140-2152, 2010
- [McQuarrie-1976] D.A. McQuarrie, **Statistical Mechanics** *Harper & Row*, 1976
- [Pierotti-1976] R.A. Pierotti, **A Scaled Particle Theory of Aqueous and Nonaqueous Solutions**, *Chem. Rev.*, Vol. 76, pp. 717-726, 1976
- [Nicholls-2010] A. Nicholls, S. Wlodek and J.A. Grant, **SAMPL2 and continuum modeling**, *J. Comput. Aided Mol. Des.*, Vol. 24, pp. 293-306, 2010

INDEX

Symbols

- MMFF94s
 - szybki command line option, 18
- am1bcc
 - szybki command line option, 18
- complex <filename>
 - szybki command line option, 15
- conj
 - szybki command line option, 20
- current_charges
 - szybki command line option, 19
- entropy <type>
 - szybki command line option, 22
- exact_vdw
 - szybki command line option, 18
- fix_file <filename>
 - szybki command line option, 16
- fix_smarts <p>
 - szybki command line option, 20
- flex_file <filename>
 - szybki command line option, 16
- grad_conv <c>
 - szybki command line option, 20
- harm_constr1 <k>
 - szybki command line option, 19
- harm_constr2 <d>
 - szybki command line option, 19
- harm_smarts <p>
 - szybki command line option, 19
- heavy_rms
 - szybki command line option, 16
- i <filename>
 - szybki command line option, 16
- in <filename>
 - szybki command line option, 16
- inner_dielectric <d>
 - szybki command line option, 19
- largest_part
 - szybki command line option, 21
- ligands <filename>
 - szybki command line option, 16
- loadPG <filename>
 - szybki command line option, 16
- log <prefix>
 - szybki command line option, 16
- max_iter <m>
 - szybki command line option, 21
- mod_vdw
 - szybki command line option, 19
- mol2charges
 - szybki command line option, 19
- neglect_frozen
 - szybki command line option, 19
- noCoulomb
 - szybki command line option, 19
- no_opt
 - szybki command line option, 21
- o <filename>
 - szybki command line option, 16
- opt_cart
 - szybki command line option, 21
- opt_solid
 - szybki command line option, 21
- opt_torsions
 - szybki command line option, 21
- out <filename>
 - szybki command line option, 16
- out_complex <filename>
 - szybki command line option, 16
- out_protein <filename>
 - szybki command line option, 16
- param
 - szybki command line option, 16
- polarH <r>
 - szybki command line option, 21
- prefix <pn>
 - szybki command line option, 16
- protein <filename>
 - szybki command line option, 17
- protein_elec <m>
 - szybki command line option, 19
- protein_vdw <r>
 - szybki command line option, 20

-pvmconf <hosts.txt>
 szybki command line option, 22

-report
 szybki command line option, 17

-reportFile <filename>
 szybki command line option, 17

-residue <r>
 szybki command line option, 21

-rws
 szybki command line option, 22

-salt <c>
 szybki command line option, 20

-savePG <filename>
 szybki command line option, 17

-sdtag <string>
 szybki command line option, 17

-sfp
 szybki command line option, 22

-shefA <a>
 szybki command line option, 20

-shefB
 szybki command line option, 20

-sheffield
 szybki command line option, 20

-sideC <r>
 szybki command line option, 21

-silent
 szybki command line option, 18

-solv_dielectric <d>
 szybki command line option, 20

-solventCA <s>
 szybki command line option, 20

-solventPB
 szybki command line option, 20

-strict
 szybki command line option, 20

-strip_water
 szybki command line option, 21

-t
 szybki command line option, 22

-vdw_cutoff
 szybki command line option, 20

-verbose
 szybki command line option, 18

A

APPNAME_OE_ARCH, 4

E

environment variable

APPNAME_OE_ARCH, 4
 OE_ARCH, 4
 OE_LICENSE, 3
 PATH, 4, 5, 7

PVM_ARCH, 6
 PVM_PATH, 6
 PVM_ROOT, 6

O

OE_ARCH, 4
 OE_LICENSE, 3

P

PATH, 4, 5, 7
 PVM_ARCH, 6
 PVM_PATH, 6
 PVM_ROOT, 6

S

szybki command line option

- MMFF94s, 18
- am1bcc, 18
- complex <filename>, 15
- conj, 20
- current_charges, 19
- entropy <type>, 22
- exact_vdw, 18
- fix_file <filename>, 16
- fix_smarts <p>, 20
- flex_file <filename>, 16
- grad_conv <c>, 20
- harm_constr1 <k>, 19
- harm_constr2 <d>, 19
- harm_smarts <p>, 19
- heavy_rms, 16
- i <filename>, 16
- in <filename>, 16
- inner_dielectric <d>, 19
- largest_part, 21
- ligands <filename>, 16
- loadPG <filename>, 16
- log <prefix>, 16
- max_iter <m>, 21
- mod_vdw, 19
- mol2charges, 19
- neglect_frozen, 19
- noCoulomb, 19
- no_opt, 21
- o <filename>, 16
- opt_cart, 21
- opt_solid, 21
- opt_torsions, 21
- out <filename>, 16
- out_complex <filename>, 16
- out_protein <filename>, 16
- param, 16
- polarH <r>, 21
- prefix <pn>, 16

- protein <filename>, 17
- protein_elec <m>, 19
- protein_vdw <r>, 20
- pvmconf <hosts.txt>, 22
- report, 17
- reportFile <filename>, 17
- residue <r>, 21
- rws, 22
- salt <c>, 20
- savePG <filename>, 17
- sdtag <string>, 17
- sfp, 22
- shefA <a>, 20
- shefB , 20
- sheffield, 20
- sideC <r>, 21
- silent, 18
- solv_dielectric <d>, 20
- solventCA <s>, 20
- solventPB, 20
- strict, 20
- strip_water, 21
- t, 22
- vdw_cutoff, 20
- verbose, 18