

# **iAPBS Interface: Users's Guide**

**Robert Konecny**  
University of California San Diego  
National Biomedical Computation Resource  
Center for Theoretical Biological Physics

[rok@ucsd.edu](mailto:rok@ucsd.edu)

## **iAPBS Interface: Users's Guide**

by Robert Konecny

Published Mar 2011

Copyright © 2005-2009 UC Regents

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

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

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

# Table of Contents

<b>1. Introduction</b> .....	<b>1</b>
1.1. iAPBS interface.....	1
1.2. Amber/iAPBS Module.....	1
1.3. NAMD/iAPBS Module.....	1
1.4. CHARMM/iAPBS Module.....	1
<b>2. Implementation of the iAPBS interface</b> .....	<b>3</b>
<b>3. Installing the iAPBS Interface</b> .....	<b>4</b>
3.1. Requirements.....	4
3.2. Building iAPBS interface.....	4
3.3. Building of Amber/APBS module.....	5
3.4. Building of NAMD/APBS module.....	6
3.5. Building of CHARMM/APBS module.....	8
<b>4. Using the iAPBS Interface</b> .....	<b>11</b>
4.1. Using Amber/APBS module.....	11
4.2. Using NAMD/APBS module.....	14
4.3. Using CHARMM/APBS module.....	17
4.4. Notes.....	20
<b>5. Examples</b> .....	<b>21</b>
5.1. Examples of using the Amber/APBS module.....	21
5.2. Examples of using the NAMD/APBS module.....	23
5.3. Examples of using the CHARMM/APBS module.....	27

# List of Tables

4-1. APBS and Amber/APBS keywords.....	11
4-2. APBS and NAMD/APBS keywords.....	14
4-3. APBS and CHARMM/APBS keywords.....	17

# Chapter 1. Introduction

## 1.1. iAPBS interface

APBS<sup>1</sup> (Adaptive Poisson-Boltzmann Solver) is a software package for the numerical solution of the Poisson-Boltzmann equation (PBE), one of the most popular continuum models for describing electrostatic interactions between molecular solutes in salty, aqueous media. APBS was designed to efficiently evaluate electrostatic properties for a wide range of length scales to enable the investigation of molecules with tens to millions of atoms.

The iAPBS package provides a C/C++ and Fortran interface to the APBS library through a single function call. The interface code can be compiled as a library (`libiapbs.a`) which can be linked with a Fortran or C/C++ application thus making most of APBS functionality available from within any C/C++/Fortran code.

## 1.2. Amber/iAPBS Module

The Amber/APBS interface makes most of the APBS functionality available to Amber users. The APBS interface in sander is accessible through the `&apbs` keyword. Additionally, the `igb` keyword must be set to 6. This combination of keywords means that a calculation will be performed on a given system in vacuum and then APBS calculated solvation energies and forces will be added to the total energy and forces.

In addition to minimization and molecular dynamics simulation with APBS calculated implicit solvent contribution the Amber/APBS module can write out calculated electrostatic potential to a file. This file can be visualized using third party applications, including VMD, Pymol, PMV/Vision and OpenDX.



The current version of the Amber/APBS module supports serial execution only. Parallel capability will be added in later versions.

## 1.3. NAMD/iAPBS Module

The NAMD/APBS interface allows access to APBS functionality from NAMD. The APBS NAMD module can be used to perform implicit solvent MD simulation with NAMD, to write out electrostatic maps for visualization purposes and also to perform MM-PBSA calculations directly with NAMD.

## 1.4. CHARMM/iAPBS Module

The CHARMM/APBS interface makes most of the APBS functionality available to CHARMM users. The interface was designed as an extension to the CHARMM's PBEQ module. The CHARMM/APBS module is invoked by `APBS`

keyword inside of the PBEQ section. The module's keywords are identical or similar to APBS keywords to allow for easy transition between the codes.



The current version of the CHARMM/APBS module supports serial execution only. Parallel capability will be added in later versions.

## **Notes**

1. <http://apbs.sourceforge.net/>

# Chapter 2. Implementation of the iAPBS interface

The iAPBS interface is implemented as a single wrapper function to the APBS C library. This function (`apbsdrv`) can be called from a C, C++ or Fortran code. The function accepts several input parameters - coordinates of atoms, their charges and radii and APBS calculation parameters. The function returns calculated electrostatic energies and forces. Various calculated properties like electrostatic potential, solvent accessible surface and charge can also be written to files and subsequently visualized by external applications.

For detailed description of required input and output parameters please see `src/apbs_driver.h` and `src/wrapper_inc.f` and iAPBS Programmer's Guide<sup>1</sup>.

The iAPBS distribution also includes a simple reference Fortran code (`src/wrapper.f`) which reads input data from an external file, calls the `apbsdrv` wrapper function and prints out calculated electrostatic energy.

If you plan to include APBS functionality in your C, C++ or Fortran code through iAPBS interface the following steps are required:

- Write a C, C++ or Fortran code which prepares all necessary APBS input parameters (for example by parsing an input file). When all input parameters are set in the code call the `apbsdrv` function and process the output data.
- Compile your code and link it with the APBS, MALOC, BLAS and iAPBS libraries.

For example, if you have a code called `application.f` the following would compile and link it with the iAPBS and other required libraries:

```
ifort -o application application.f -L${IAPBS_LIB} -liapbs \  
      -L${APBS_LIB} -lapbsmainroutines -lapbs -lmalloc -lapbsblas
```

## Notes

1. <http://mccammon.ucsd.edu/iapbs/programmer/>

# Chapter 3. Installing the iAPBS Interface

## 3.1. Requirements

To compile iAPBS interface two packages are required:

- APBS<sup>1</sup> (version 1.0.0 or later).
- iAPBS<sup>2</sup>

## 3.2. Building iAPBS interface

Building of the iAPBS interface requires compilation and installation of MALOC and APBS libraries. The following describes each step separately (assumes `bash` as login script, modify appropriately for `t/csh`).

### 1. Setup the environment.

```
mkdir iAPBS-build
cd iAPBS-build
export APBS_PREFIX=`pwd`
export CC=icc
export F77=ifort
tar xvzf iapbs-1.2.0.tgz
```

### 2. Compile and install APBS (for details please see the APBS manual):

```
cd $APBS_PREFIX
tar xvzf apbs-1.0.0.tar.gz
cd apbs-1.0.0
export APBS_SRC=`pwd`
export CFLAGS="${CFLAGS} -DVAPBSQUIET"
./configure --prefix=${APBS_PREFIX} --enable-abps-quiet
make && make install

cd $APBS_PREFIX/include
ln -s $APBS_SRC/contrib/include/maloc
cd $APBS_PREFIX/lib
cp $APBS_SRC/contrib/lib/libmaloc.a .
cp $APBS_SRC/contrib/blas/.libs/libapbsblas.a .
```

### 3. Build the iAPBS interface library:

```
cd $APBS_PREFIX
cd iapbs-1.2.0
./setup.sh
./configure --prefix=${APBS_PREFIX} --disable-openmp
```

```
make && make install
```

After this last step the `${APBS_PREFIX}/lib` (or your machine specific) directory should contain all necessary libraries for Fortran/C/C++ iAPBS/APBS linking, including the `libiapbs.a` library.

Optionally you can now test the iAPBS library by compiling a wrapper program and running it on a set of examples:

```
cd src
make wrapper
cd ../test
# if testing OpenMP version
# ulimit -s unlimited
./test.sh
```

### 3.3. Building of Amber/APBS module

Currently the Amber/APBS module is supported on i386 and x86\_64 Linux platforms only. The Intel compiler is recommended.

1. Amber9 source code must be first patched for use with iAPBS. No patching is necessary for Amber10.

Amber9:

```
export APBS_LIB=${APBS_PREFIX}/lib
cd $AMBERHOME/src
./configure ifort_ia32
cd $AMBERHOME/src/sander
make clean
cp ${APBS_PREFIX}/iapbs-1.2.0/modules/Amber/apbs.f90 ./apbs.f
cp ${APBS_PREFIX}/iapbs-1.2.0/modules/Amber/apbs_vars.f90 ./apbs_vars.f
patch -b -z .orig -p1 < \
    ${APBS_PREFIX}/iapbs-1.2.0/modules/Amber/patches/amber9_sander.patch
make depend
unset CC F77
unset CFLAGS FFLAGS LDFLAGS
```

Amber10:

```
export APBS_LIB=${APBS_PREFIX}/lib
cd $AMBERHOME/src
./configure_at icc
./configure_amber ifort
cd $AMBERHOME/src/sander
make clean
make depend
unset CC F77
unset CFLAGS FFLAGS LDFLAGS
```

2. After this, the sander source code is ready for building and linking with iAPBS:

```
make -e AMBERBUILD_FLAGS="-DAPBS" sander.APBS
```

After this last step the directory `AMBERHOME/src/sander` should contain `sander.APBS` executable.

3. To verify that the built binary produces correct results run tests in `AMBERHOME/test` directory:

```
export TESTsander=${AMBERHOME}/src/sander/sander.APBS
cd ${AMBERHOME}/test/iapbs_radi
./Run.ion.min
```

## 3.4. Building of NAMD/APBS module

Currently the NAMD/APBS module is supported on x86\_64 Linux platform only. The Intel compiler is recommended. The following building steps have been verified for Intel 11.0 and gcc 4.4.3 compilers and Linux Ubuntu 10.04 (Linux kernel 2.6.32-29 x86\_64).

Download APBS, iAPBS and NAMD latest source code into a directory and untar it there. This directory will be `APBS_PREFIX`:

```
export APBS_PREFIX=`pwd`
```

**Build APBS:**

```
cd $APBS_PREFIX/apbs
export APBS_SRC=`pwd`
./configure --prefix=${APBS_PREFIX} --enable-abps-quiet --disable-openmp
make && make install
```

```
cd $APBS_PREFIX/include
ln -s $APBS_SRC/contrib/include/malloc
cd $APBS_PREFIX/lib
cp $APBS_SRC/contrib/lib/libmalloc.a .
cp $APBS_SRC/contrib/blas/.libs/libapbsblas.a .
cp $APBS_SRC/contrib/zlib/.libs/libz.a .
```

**Build iAPBS**

```
cd $APBS_PREFIX
ln -s iapbs-1.2.0 ./iapbs
cd $APBS_PREFIX/iapbs
./setup.sh
./configure --prefix=${APBS_PREFIX} --disable-openmp
make && make install
```

**Build charm**

```
cd $APBS_PREFIX/namd2
export NAMD_SRC=`pwd`
wget -O - http://charm.cs.uiuc.edu/autobuild/bin/charm_src.tar.gz |tar xzf -
cd $NAMD_SRC/charm
./build charm++ net-linux-x86_64 tcp --no-build-shared -O -DCMK_OPTIMIZE=1
```

```
cd $NAMD_SRC
wget -O - http://www.ks.uiuc.edu/Research/namd/libraries/fftw-linux-x86_64.tar.gz | tar xzf -
mv linux-x86_64 fftw
wget -O - http://www.ks.uiuc.edu/Research/namd/libraries/tcl-linux-x86_64.tar.gz | tar xzf -
mv linux-x86_64 tcl
```

### Build NAMD

```
cd $NAMD_SRC
patch -p0 < $APBS_PREFIX/iapbs/modules/NAMD/patches/namd2-apbs-20110315.patch
cd src
ln -s $APBS_PREFIX/iapbs/modules/NAMD/GlobalMasterAPBS.* .
cd ../arch
ln -s $APBS_PREFIX/iapbs/modules/NAMD/*.apbs .
```

```
cd $NAMD_SRC
cat >Make.charm <<END
CHARMBASE = $NAMD_SRC/charm
END
```

```
cd arch
arch=Linux-x86_64
cat >${arch}.fftw <<END
FFTDIR=${NAMD_SRC}/fftw
FFTINCL=-I\$(FFTDIR)/include
FFTLIB=-L\$(FFTDIR)/lib -lsrfftw -lsfftw
FFTFLAGS=-DNAMD_FFTW
FFT=\$(FFTINCL) \$(FFTFLAGS)
END
```

```
cat > ${arch}.tcl <<END
TCLDIR=${NAMD_SRC}/tcl
TCLINCL=-I\$(TCLDIR)/include
TCLLIB=-L\$(TCLDIR)/lib -ltcl8.3 -ldl
TCLFLAGS=-DNAMD_TCL
TCL=\$(TCLINCL) \$(TCLFLAGS)
END
```

```
cd $NAMD_SRC

./config apbs ${arch}-g++ --charm-arch net-linux-x86_64-tcp
cd ${arch}-g++
make -j4
```

If the last step finishes without errors you should see the namd2 binary in your \$NAMD\_SRC/\${arch}-g++ directory. To test it you can run these jobs:

```
./namd2 src/alanin
./charmrun ++local +p2 ./namd2 src/alanin

cd $APBS_PREFIX/iapbs/modules/NAMD/test/dipeptide
$NAMD_SRC/${arch}-g++/namd2 apbs-solvation.conf
```

```
cd ../ubq/
$NAMD_SRC/${arch}-g++/namd2 ubq.conf
```

## 3.5. Building of CHARMM/APBS module

Currently, building of CHARMM/APBS module is supported on i386 and x64 Linux platforms only (gnu target in the CHARMM installation process). Both GNU and Intel compilers have been tested. The build process is a bit complex at this time, future releases will include more automated compilation.

First, make sure you have these packages and versions:

- CHARMM version 34b2 or later (<http://www.charmm.org/>).
- APBS version 1.0.0 or later (<http://apbs.sourceforge.net/>).
- iAPBS version 1.2.0 (<http://mccammon.ucsd.edu/iapbs>).

Compile and install APBS (for details please see the APBS manual):

```
export APBS_PREFIX=/home/soft/iAPBS
mkdir -p $APBS_PREFIX
cd $APBS_PREFIX
tar xvzf apbs-1.0.0.tar.gz
cd ${APBS_PREFIX}/apbs-0.1.0
export CC=icc
export F77=ifort
export F90=ifort
export FC=ifort
export CFLAGS="${CFLAGS} -DVAPBSQUIET"
./configure --prefix=${APBS_PREFIX}
make && make install

cd $APBS_PREFIX/include
ln -s $APBS_PREFIX/apbs/contrib/include/malloc
cd $APBS_PREFIX/lib
cp $APBS_PREFIX/apbs/contrib/lib/libmalloc.a .
```

Build iAPBS interface library:

```
cd $APBS_PREFIX
tar xvzf iapbs-1.2.0.tgz
cd iapbs-1.2.0
./setup.sh
./configure --prefix=${APBS_PREFIX}
make && make install
```

After this last step the `$APBS_PREFIX/lib` directory should contain `iapbs.a` library which can be linked with CHARMM.

### 3.5.1. Compiling the CHARMM/APBS module

The following instructions assume your CHARMM version supports the CHARMM/APBS module (v.34b2 and later). If you are using version which does not you have to patch it first before following the steps below. Please download appropriate patch pack from the iAPBS site<sup>6</sup> and follow included directions. Your CHARMM distribution must be patched before proceeding with the following steps.

Build and install APBS and iAPBS libraries as described above.

Compile CHARMM with CHARMM/APBS module enabled

```
cd your_CHARMM_dir
# patch your charmm installation
cd c35b2/source/misc/
rm apbs.src
ln -s $APBS_PREFIX/iapbs/modules/CHARMM/apbs.f ./apbs.src
cd ../fcm
rm apbs.fcm
ln -s $APBS_PREFIX/iapbs/modules/CHARMM/apbs_fcm.f ./apbs.fcm
cd $APBS_PREFIX/c35b2/
patch -p0 < $APBS_PREFIX/iapbs/modules/CHARMM/patches/c35b2/pbeq.patch
patch -p0 < $APBS_PREFIX/iapbs/modules/CHARMM/patches/c35b2/install.patch
patch -p0 < $APBS_PREFIX/iapbs/modules/CHARMM/patches/c35b2/Makefile_gnu.patch
export APBS_LIB=$APBS_PREFIX/lib
cd $APBS_LIB
objcopy -N daxpy_ -N dcopy_ -N ddot_ -N dnm2_ \
$APBS_PREFIX/apbs/contrib/blas/.libs/libapbsblas.a ./libapbsblasmod.a
export APBS_BLAS="-L${APBS_LIB} -lapbsblasmod"
cd $APBS_PREFIX/c35b2/
./install.com gnu medium APBS IFORT

# test it
cd $APBS_PREFIX/iapbs/modules/CHARMM/examples/
$APBS_PREFIX/c35b2/exec/gnu/charmm < apbs_elstat.inp
```

After the last step your `exec/gnu` directory should contain `charmm.exe` executable which has APBS compiled in.

## Notes

1. <http://apbs.sourceforge.net/>
2. <http://mccammon.ucsd.edu/iapbs>
3. <http://www.charmm.org/>
4. <http://apbs.sourceforge.net/>
5. <http://mccammon.ucsd.edu/iapbs>

6. <http://mccammon.ucsd.edu/iapbs>

# Chapter 4. Using the iAPBS Interface

The iAPBS interface provides access to APBS functionality so understanding theory and implementation behind APBS is essential. Good place to start is APBS documentation<sup>1</sup> and APBS Tutorial<sup>2</sup>.

## 4.1. Using Amber/APBS module

The APBS module in sander offers an alternative to the built-in PB sander module. The APBS module can be used for example for implicit solvent minimization and dynamics, calculation and visualization of miscellaneous electrostatic biomolecular properties. Please see the Examples section of this User's Guide.

The following table lists all Amber/APBS keywords with their description. The left side of the table also lists corresponding APBS keywords which Amber/APBS keywords mimic very closely. For detailed discussion of APBS keywords please see APBS documentation<sup>3</sup>.

### 4.1.1. Amber/APBS Module Keywords

Table 4-1. APBS and Amber/APBS keywords

APBS keyword	Amber/APBS keyword	Description
	apbs	Enters the APBS module
mg-auto/mg-para	calc_type [0]	0: manual MG; 1: autoMG; 2: parallel MG
lpbe/nbpe	nonlin [0]	Linear/full Poisson-Boltzmann equation: 0: linear; 1: non-linear; 4: size-dependent PBE
bcfl	bcfl [1]	Boundary condition method: 0: zero; 1: sdh; 2: mdh; 4: focus
srfm	srfm [1]	Surface calculation method: 0: mol; 1: smol; 2: spl2; 3: spl4
pdie	pdie [2.0]	Solute dielectric
sdie	sdie [78.4]	Solvent dielectric
sdens	sdens [10.0]	Vacc sphere density
srad	srad [1.4]	Solvent radius
swin	swin [0.3]	Cubic spline window
temp	temp [298.15]	Temperature (in K)
gamma	gamma [0.105]	Surface tension for apolar energies/forces (in kJ/mol/A <sup>2</sup> )
chgm	chgm [1]	Charge discretization method: 0: spl0; 1: spl2; 2: spl4

APBS keyword	Amber/APBS keyword	Description
vol	smvolume	The parameter smvolume controls the lattice size (in Angstroms <sup>3</sup> ) used in the SMPBE formalism.
size	smsize	The parameter smsize controls the relative size of the ions (in Angstroms) such that each lattice site can contain a single ion of volume radius <sup>3</sup> or size ions of volume radius <sup>3</sup> /size.
calcenergy	calcenergy [1]	Energy calculation flag: 0: Do not perform energy calculation; 1: Calculate total energy only; 2: Calculate per-atom energy components
calcforce	calcforce [0]	Atomic forces calculation: 0: Do not perform force calculation; 1: Calculate total force only; 2: Calculate per-atom force components
write pot	wpot	Writes electrostatic potential data to <code>iapbs-pot.dx</code> in DX format
write charge	wchg	Writes charge data to <code>iapbs-charge.dx</code> in DX format
write smol	wsmol	Writes molecular surface data to <code>iapbs-smol.dx</code> in DX format
write kappa	wkappa	Writes the ion-accessibility kappa map to <code>iapbs-kappa.dx</code> in DX format
write diel	wdiel	Writes dielectric maps to <code>iapbs-diel[x,y,z].dx</code> in DX format
read charge	rchg	Reads charge data from <code>iapbs-charge.dx</code> in DX format
read kappa	rkappa	Reads the ion-accessibility kappa map from <code>iapbs-kappa.dx</code> in DX format
read diel	rdiel	Reads dielectric maps from <code>iapbs-diel[x,y,z].dx</code> in DX format
	nion	Number of counterions
ionq	ionq	Counterion charges (in e)
ionc	ionc	Counterion concentrations (in M)
ionr	ionrr	Counterion radii (in A)

APBS keyword	Amber/APBS keyword	Description
dime	dime	Grid dimensions (in x, y and z)
cmeth	cmeth [1]	Centering method: 0: Center on a point 1: Center on a molecule
gcent	center	Grid center if <code>cmeth=0</code>
ccmeth	ccmeth [1]	Coarse grid centering method: 0: Center on a point 1: Center on a molecule
cgcent	ccenter	Coarse grid center if <code>ccmeth=0</code>
fcmeth	fcmeth [1]	Fine grid centering method: 0: Center on a point 1: Center on a molecule
fgcent	fcenter	Fine grid center if <code>fcmeth=0</code>
grid	grid	Grid spacings
glen	glen	Grid side lengths
cglen	cglen	Coarse grid side lengths
fglen	fglen	Fine grid side lengths
pdime	pdime	Grid of processors to be used in calculation
ofrac	ofrac [0.1]	Overlap fraction between processors
	radiopt [0]	Radii optimization: 0: use prmtop for both charge and radius; 1: read charge and radius information from 'pbparamsin' file (format: atom_label charge radius); 2: Read charge/radius information from a PQR file; 3: Read radius information from a PQR file.
	pqr	PQR file name
	calcnpenergy [1]	Calculate nonpolar energy [0, 1]. 0: Don't calculate; 1: SASA-based apolar energy calculation.
	sp_apbs [.FALSE.]	Perform a single point (a single electrostatic evaluation) APBS calculation



Values in square brackets [] are defaults.

## 4.1.2. Amber/APBS Notes

See note about disabling creation of `io.mc` file below before attempting any extensive minimization or MD simulation with the APBS module.

## 4.2. Using NAMD/APBS module

The APBS module in NAMD adds implicit solvent simulation environment to the NAMD capabilities. The APBS module can be used for example for implicit solvent minimization and dynamics, calculation and visualization of miscellaneous electrostatic biomolecular properties. Please see the Examples section of this User's Guide.

The following table lists all NAMD/APBS keywords with their description. The left side of the table also lists corresponding APBS keywords which NAMD/APBS keywords mimic very closely. For detailed discussion of APBS keywords please see APBS documentation<sup>4</sup>.

### 4.2.1. NAMD/APBS Module Keywords

**Table 4-2. APBS and NAMD/APBS keywords**

APBS keyword	NAMD/APBS keyword	Description
	apbsForces [off]	Turns on the APBS module.
	apbsPQRFile	PQR file name to be read.
mg-auto/mg-para	calc_type [0]	0: manual MG; 1: autoMG; 2: parallel MG
lpbe/nbpe	nonlin [0]	Linear/full Poisson-Boltzmann equation: 0: linear; 1: non-linear; 4: size-dependent PBE
bcfl	bcfl [1]	Boundary condition method: 0: zero; 1: sdh; 2: mdh; 4: focus
srfm	srfm [1]	Surface calculation method: 0: mol; 1: smol; 2: spl2; 3: spl4
pdie	pdie [2.0]	Solute dielectric
sdie	sdie [78.4]	Solvent dielectric
sdens	sdens [10.0]	Vacc sphere density
srad	srad [1.4]	Solvent radius
swin	swin [0.3]	Cubic spline window
temp	temp [298.15]	Temperature (in K)
gamma	gamma [0.105]	Surface tension for apolar energies/forces (in kJ/mol/A <sup>2</sup> )
chgm	chgm [1]	Charge discretization method: 0: spl0; 1: spl2; 2: spl4

APBS keyword	NAMD/APBS keyword	Description
vol	smvolume [10.0]	The parameter smvolume controls the lattice size (in Angstroms <sup>3</sup> ) used in the SMPBE formalism.
size	smsize [1000.0]	The parameter smsize controls the relative size of the ions (in Angstroms) such that each lattice site can contain a single ion of volume radius <sup>3</sup> or size ions of volume radius <sup>3</sup> /size.
calcenergy	calcenergy [1]	Energy calculation flag: 0: Do not perform energy calculation; 1: Calculate total energy only; 2: Calculate per-atom energy components
calcforce	calcforce [0]	Atomic forces calculation: 0: Do not perform force calculation; 1: Calculate total force only; 2: Calculate per-atom force components
	calcnpenenergy [1]	Calculate nonpolar energy [0, 1]. 0: Don't calculate; 1: SASA-based apolar energy calculation.
write pot	wpot [off]	Writes electrostatic potential data to <code>iapbs-pot.dx</code> in DX format
write charge	wchg [off]	Writes charge data to <code>iapbs-charge.dx</code> in DX format
write smol	wsmol [off]	Writes molecular surface data to <code>iapbs-smol.dx</code> in DX format
write kappa	wkappa [off]	Writes the ion-accessibility kappa map to <code>iapbs-kappa.dx</code> in DX format
write diel	wdiel [off]	Writes dielectric maps to <code>iapbs-diel[x,y,z].dx</code> in DX format
read charge	rchg [off]	Reads charge data from <code>iapbs-charge.dx</code> in DX format
read kappa	rkappa [off]	Reads the ion-accessibility kappa map from <code>iapbs-kappa.dx</code> in DX format
read diel	rdiel [off]	Reads dielectric maps from <code>iapbs-diel[x,y,z].dx</code> in DX format

APBS keyword	NAMD/APBS keyword	Description
ion	ion charge conc radius	Counterion charges (in e), Counterion concentrations (in M), Counterion radii (in A)
dime	dime	Grid dimensions (in x, y and z)
cmeth	cmeth [1]	Centering method: 0: Center on a point 1: Center on a molecule
gcent	center	Grid center if <code>cmeth=0</code>
ccmeth	ccmeth [1]	Coarse grid centering method: 0: Center on a point 1: Center on a molecule
cgcent	ccenter	Coarse grid center if <code>ccmeth=0</code>
fcmeth	fcmeth [1]	Fine grid centering method: 0: Center on a point 1: Center on a molecule
fgcent	fcenter	Fine grid center if <code>fcmeth=0</code>
grid	grid	Grid spacings
glen	glen	Grid side lengths
cglen	cglen	Coarse grid side lengths
fglen	fglen	Fine grid side lengths
pdime	pdime	Grid of processors to be used in parallel calculation
ofrac	ofrac [0.1]	Overlap fraction between processors
	debug [0]	Debugging flag [0-5]
	verbose [0]	Print (verbosity) flag [0-5]
	sp_apbs [off]	Perform a single point (a single electrostatic evaluation) APBS calculation
	recalculateGrid [off]	Recalculate the grid dimensions on the fly.



Values in square brackets [] are defaults.

### 4.2.2. NAMD/APBS Notes

When the `recalculateGrid` and `grid` keywords are set then the grid size parameters (`cglen`, `fglen` and `dime`) are automatically recalculated on the fly during the simulation. For example if `grid` is set to [0.5 0.5 0.5] Angstroms the grid size will be automatically adjusted to fit the molecule as the system's dimensions are changed during the simulation (the requested grid spacing 0.5 A will be maintained). This is the recommended option for most of simulations since this prevents the solute to "escape" the pre-set grid.

See note about disabling creation of `io.mc` file below before attempting any extensive minimization or MD simulation with the APBS module.

## 4.3. Using CHARMM/APBS module

Keyword `APBS` inside of the `PBEQ` section in the CHARMM input file enters the CHARMM/APBS module. The module's keywords are as similar to the original APBS keywords as possible. However, there are several exceptions. Please read the following section carefully and also take a look at the Examples section of this User's Guide.

The following table lists all CHARMM/APBS keywords with their description. The left side of the table also lists corresponding APBS keywords which CHARMM/APBS keywords mimic very closely. For detailed discussion of APBS keywords please see APBS documentation<sup>5</sup>.

### 4.3.1. CHARMM/APBS Module Keywords

Table 4-3. APBS and CHARMM/APBS keywords

APBS keyword	CHARMM/APBS keyword	Description
	APBS	Enters the APBS module
mg-auto	MGAUTO	Automatically-configured sequential focusing multigrid calculation
mg-para	MGPARA	Automatically-configured parallel focusing multigrid calculation
mg-manual	MGMANUAL	Manually-configured multigrid calculation
lpbe/npbe	LPBE/NPBE	Linear/full Poisson-Boltzmann equation
bcfl	BCFL [1]	Boundary condition method: 0: zero; 1: sdh; 2: mdh; 4: focus
srfm	SRFM [1]	Surface calculation method: 0: mol; 1: smol; 2: spl2
pdie	PDIE [2.0]	Solute dielectric
sdie	SDIE [78.54]	Solvent dielectric
sdens	SDENS [10.0]	Vacc sphere density
srad	SRAD [1.4]	Solvent radius
swin	SWIN [0.3]	Cubic spline window
temp	TEMP [298.15]	Temperature (in K)
gamma	GAMMA [0.105]	Surface tension for apolar energies/forces (in kJ/mol/A <sup>2</sup> )
chgm	CHGM [1]	Charge discretization method: 0: spl0; 1: spl2

APBS keyword	CHARMM/APBS keyword	Description
calcenergy	CALCE [1]	Energy calculation flag: 0: Do not perform energy calculation; 1: Calculate total energy only; 2: Calculate per-atom energy components
calcforce	CALCF [0]	Atomic forces calculation: 0: Do not perform force calculation; 1: Calculate total force only; 2: Calculate per-atom force components
write pot	WPOT	Writes electrostatic potential data to <code>iapbs-pot.dx</code> in DX format
write charge	WCHG	Writes charge data to <code>iapbs-charge.dx</code> in DX format
write smol	WSMOL	Writes molecular surface data to <code>iapbs-smol.dx</code> in DX format
write kappa	WKAPPA	Writes the ion-accessibility kappa map to <code>iapbs-kappa.dx</code> in DX format
write diel	WDIEL	Writes dielectric maps to <code>iapbs-diel[x,y,z].dx</code> in DX format
read charge	RCHG	Reads charge data from <code>iapbs-charge.dx</code> in DX format
read kappa	RKAPPA	Reads the ion-accessibility kappa map from <code>iapbs-kappa.dx</code> in DX format
read diel	RDIEL	Reads dielectric maps from <code>iapbs-diel[x,y,z].dx</code> in DX format
ionq	IONQ{1 2}	Counterion charges (in e)
ionc	IONC{1 2}	Counterion concentrations (in M)
ionr	IONR{1 2}	Counterion radii (in Å)
dime	DIMX [65] DIMY [65] DIMZ [65]	Grid dimensions (in x, y and z)
cmeth	CMET [1]	Centering method: 0: Center on a point 1: Center on a molecule
center	CNTX CNTY CNTZ	Grid center if <code>CMET 0</code>
ccmeth	CCME [1]	Coarse grid centering method: 0: Center on a point 1: Center on a molecule
ccenter	CCNX CCNY CCNZ	Coarse grid center if <code>CCME 0</code>

APBS keyword	CHARMM/APBS keyword	Description
fcmeth	FCME [1]	Fine grid centering method: 0: Center on a point 1: Center on a molecule
fcenter	FCNX FCNY FCNZ	Fine grid center if FCME 0
grid	GRDX GRDY GRDZ	Grid spacings
glen	GLNX GLNY GLNZ	Grid side lengths
cglen	CGLX CGLY CGLZ	Coarse grid side lengths
fglen	FGLX FGLY FGLZ	Fine grid side lengths
pdime	PDIX PDIY PDIZ	Grid of processors to be used in calculation
ofrac	OFRA [0.1]	Overlap fraction between processors
	DEBUG [0]	Debugging flag
	UPDATE [1]	How often are solvation forces calculated during minimization or MD. Defaults to 1 which means solvation forces are updated every step.
	UMETHOD [1]	Method by which are solvation forces updated, when using UPDATE keyword. 0: no solvation forces are included between updates 1: forces from previous APBS calculation are used between updates
	SFORCE	Requests calculation of solvation forces (this keyword must be present when doing MD or minimization with APBS calculated solvation forces).



Values in square brackets [] are defaults.

### 4.3.2. CHARMM/APBS Notes

Each occurrence of the `APBS` keyword triggers APBS calculation. This may not be desirable in all cases, for example when setting up a molecular dynamics simulation with APBS calculated solvation forces. In such a case keyword `SKIP` in the `APBS` section will prevent from starting APBS calculation, the section will just initialize all APBS parameters. The actual APBS calculation will be performed when `DYNAMICS` section starts the simulation.

When the `SFORCE` keyword is specified the APBS calculation is initialized (all parameters are set) with the `APBS` keyword inside of the `PBEQ` section but the actual electrostatic calculation starts during minimization or MD step.

When performing solvation forces calculation, for example during minimization or molecular dynamics, the keywords `SRFM` and `CHGM` must be both set to 2. Also, no counterions must be present.

## 4.4. Notes

Every time iAPBS module is executed it creates and writes to an auxiliary file named `io.mc`. This file can grow quite big when iAPBS is executed many times in succession, for example during minimization and molecular dynamics in implicit solvent. To disable writing to this file set `MCSH_HOME` environment variable to `/dev/null` before using CHARMM with iAPBS module. For example if using bash shell use this before running your calculation:

```
export MCSH_HOME=/dev/null
```

## Notes

1. <http://apbs.sourceforge.net/doc>
2. <http://apbs.sourceforge.net/doc/tutorial/>
3. <http://apbs.sourceforge.net/doc/user-guide/>
4. <http://apbs.sourceforge.net/doc/user-guide/>
5. <http://apbs.sourceforge.net/doc/user-guide/>

# Chapter 5. Examples

## 5.1. Examples of using the Amber/APBS module

The APBS module is initialized when `&apbs` keyword is encountered in the input file. Also, `igb` must be set to 6 in the `&cntrl` section. The `&apbs` keyword can be then followed by additional `apbs` keywords, for details see Amber/APBS keywords.

### 5.1.1. Solvation energy calculation

This example executes a solvation energy calculation and prints out total solvation energy for the given molecule (in energy terms EPB and ENPOLAR for polar and non-polar solvation terms, respectively).

```
iAPBS solvation energy example
&cntrl
  maxcyc=0, imin=1,
  cut=12.0,
  igb=6, ntb=0,
  ntp=1,
/
&apbs
  apbs_print=1
  grid=0.5, 0.5, 0.5,
  calc_type=0, cmeth=1,
  bcfl=2, srfm=1, chgm=1,
  pdie=1.0, sdie=78.54,
  sradi = 1.4,
  nion=2,
  ionq = 1.0, -1.0,
  ionc = 0.15, 0.15,
  ionrr = 2.0, 2.0,
  radiopt = 3, pqr = 'my.pqr',
  calcforce=0, calcnpenergy=1,
&end
```

### 5.1.2. Electrostatic energy calculation

This example executes a single point electrostatic energy calculation and prints out the calculated energy (in energy terms EPB and ENPOLAR for polar and non-polar terms, respectively).

```
APBS electrostatic energy example
&cntrl
  maxcyc=0, imin=1,
  cut=12.0,
```

```

    igb=6, ntb=0,
    ntp=1,
/
&apbs
  apbs_print=1
  grid = 0.1, 0.5, 0.5,
  calc_type=0, cmeth=1,
  sp_apbs=.true.,
&end

```

### 5.1.3. Molecular dynamics in implicit solvent using APBS

This examples shows how to use the Amber/APBS module for carrying out a molecular dynamics simulation in implicit solvent with APBS.

Example of APBS implicit solvent dynamics

```

&cntrl
  ntx=1,  irest=0,  imin=0,
  ntp=10, ntwx=500, nscm=100, ntwr=5000,
  dt=0.001, nstlim=50,
  temp0=300, tempi=0, ntt=1, tautp=0.1,
  igb=6, cut=12.0, ntb=0,
  ntc=2, ntf=2, tol=0.000001
/
&apbs
  apbs_print=0,
  grid = 0.5, 0.5, 0.5,
  calc_type=0,
  cmeth=1,
  bcfl=2, srfm=2, chgm=1,
  pdie=2.0, sdie=78.54,
  radiopt=2, pqr='my.pqr',
  calcforce=2, calcnpenergy=1,
  nion=2,
  ionq  = 1.0, -1.0,
  ionc  = 0.15, 0.15,
  ionrr = 2.0, 2.0,
&end

```

### 5.1.4. Calculation and visualization of electrostatic potential

When this input file is run three DX files will be created. These can be visualized using several applications (for details please see APBS visualization guide<sup>1</sup>). The DX files will be `iapbs-pot.dx`, `iapbs-smol.dx` and `iapbs-charge.dx` which will contain electrostatic potential, solvent accessible surface and charge information, respectively.

```

APBS visualization example
&cntrl
  maxcyc=0, imin=1,
  cut=12.0,
  igb=6, ntb=0
  ntp=1,
/
&apbs
  apbs_print=1,
  grid= 0.5, 0.5, 0.5,
  calc_type = 0,
  srad = 0.7,
  wpot = 1, wchg = 1, wsmol =1,
  sp_apbs=.true.,
&end

```

### 5.1.5. Notes

To determine the correct size of the numerical grid enveloping the studied molecule (parameters `cglen` and `fglen`) you can use `psize.py` tool from the APBS distribution.

Parameter `sp_apbs=.true.` triggers only a single APBS calculation (SP, single point energy). The calculated electrostatic energy contains self-energy terms thus it is not very useful by itself. This is however useful when, for instance, generating DX files for visualization. If the parameter `sp_apbs` is omitted (or set to `.false.`, which is the default) then two APBS calculations (one for a system in solvent and then in vacuum) are performed every time solvation energy and forces are recalculated. This is used during minimization and MD simulations.

## 5.2. Examples of using the NAMD/APBS module

The APBS module is initialized with keyword `apbsForces`. The APBS calculation set up is specified in `apbsForcesConfig` section. Please note that charges and radii definition for the electrostatic calculation must be read from a PQR file. See APBS documentation for PQR file format and description. APBS distribution also contains tools for generating valid PQR files (from a PDB file, for example). The order of atoms in the PQR file must be the same as in the associated `.top` file. For list of APBS-related keywords see NAMD/APBS keywords table.

## 5.2.1. Solvation energy calculation

This is an example of single point solvation energy calculation. The electrostatic calculation is done on  $0.5^3$  Å numerical grid. The external charges and radii are read from dipeptide.pqr file. The final solvation energy is printed in the 'MISC' energy column (as a sum of electrostatic and non-polar energies).

```

amber on
parmfile dipeptide.top
ambercoor dipeptide.crd
temperature 300
exclude scaled1-4
1-4scaling 0.8333333

switching on
switchDist 9
cutoff 10
pairListDist 11

outputname output
outputEnergies 1
outputTiming 100
dcdFreq 500
restartFreq 500
wrapWater on
wrapNearest on

langevin on
langevinDamping 2
langevinHydrogen no
langevinTemp 300

apbsForces on
apbsPQRFile dipeptide.pqr
apbsForcesConfig {

    calc_type 0 # mg-manual
    grid 0.5 0.5 0.5
    recalculateGrid on
    srfm 2
    chgm 1
    bcfl 1
    debug 1
    verbose 5
    pdie 2.0000
    sdie 78.5400
    sdens 10.00
    srad 1.40
    swin 0.30
    temp 298.15
    gamma 0.105
    sp_apbs off
    wpot off
}

```

```
minimize 0
```

## 5.2.2. Molecular dynamics in implicit solvent using APBS

This calculation performs a MD simulation in implicit solvent (water in this case). The charges and radii definition is read from an external file (dipeptide.pqr) and the electrostatic calculation is done on  $33^3$  points of numerical grid with the grid dimensions specified in the input file.

```
amber on
parmfile dipeptide.top
ambercoor dipeptide.crd
temperature 300
exclude scaled1-4
1-4scaling 0.8333333

switching on
switchDist 9
cutoff 10
pairListDist 11

outputname output
outputEnergies 1
outputTiming 100
dcdFreq 500
restartFreq 500
wrapWater on
wrapNearest on

langevin off
langevinDamping 2
langevinHydrogen no
langevinTemp 300

apbsForces on
apbsPQRFile dipeptide.pqr
apbsForcesConfig {
  dime 33 33 33
  cglen 17.0071 13.8706 12.3012
  fglen 17.0071 13.8706 12.3012
  srfm 2
  chgm 1
  bcfl 1
  debug 0
  pdie 2.0000
  sdie 78.5400
  sdens 10.00
  srad 1.40
  swin 0.30
  temp 298.15
  gamma 0.105
  sp_apbs off
```

```

    wpot off
}
numsteps 100

```

### 5.2.3. Calculation and visualization of electrostatic potential

This calculation writes out the calculated electrostatic potential to a file (`iapbs-pot.dx`). This potential can be visualized using `vmd` and `pymol`. The grid dimensions are  $1.0^3$  Å.

```

amber on
parmfile dipeptide.top
ambercoor dipeptide.crd

temperature 300

exclude scaled1-4
1-4scaling 0.8333333

switching on
switchDist 9
cutoff 10
pairListDist 11

outputname output
outputEnergies 1
outputTiming 100
dcdFreq 500
restartFreq 500
wrapWater on
wrapNearest on

langevin on
langevinDamping 2
langevinHydrogen no
langevinTemp 300

apbsForces on
apbsPQRFile dipeptide.pqr
apbsForcesConfig {
    dime 0 0 0
    grid 1.0 1.0 1.0
    srfm 2
    chgm 1
    bcfl 1
    debug 0
    verbose 2
    pdie 2.0000
    sdie 78.5400
    sdens 10.00
    srad 1.40
    swin 0.30
}

```

```

temp 298.15
gamma 0.105
sp_apbs on
wpot on
}
minimize 0

```

## 5.3. Examples of using the CHARMM/APBS module

The following examples demonstrate how APBS functionality can be used from within CHARMM. Using CHARMM/APBS module is very similar to using the PBEQ module so examples provided for the PBEQ module can be easily adapted for CHARMM/APBS. The example files are located in CHARMM/examples in iAPBS distribution directory.

### 5.3.1. Solvation energy calculation

This example executes a solvation energy calculation and prints out total solvation energy for the given molecule.

```

* sp_elstat.inp
* external files: top_all22_prot.inp, par_all22_prot.inp and radius.str
*

!if ?apbs .ne. 1 then stop

stream datadir.def

open read card unit 11 name @0top_all22_prot.inp
read rtf card unit 11
close unit 11

open read card unit 11 name @0par_all22_prot.inp
read para card unit 11
close unit 11

... read in or generate structure

coord orient

scalar charge show

PBEQ
  stream @0radius.str
  set factor 0.939
  set sw      0.4
  scalar wmain add @sw
  scalar wmain mult @factor
  scalar wmain set 0.0 sele type H* end
  scalar wmain show

```

```

APBS mgauto lpbe dimx 65 dimy 65 dimz 65 -
cglx 30 cgly 30 cglz 30 fglx 15 fgly 15 fglz 15 -
srfm 2 debug 1 -
calcene 1 calcforce 1 sforce -
sele all END

END

skip all excl pbelec pbnp
ener
set elstatenergy = ?ENPB
set apolarEN = ?ENNP
stop

```

### 5.3.2. Molecular dynamics in implicit solvent using APBS

This examples shows how to use the CHARMM/APBS module for performing molecular dynamics simulation in implicit solvent with APBS.

```

* md.inp
* external files: top_all122_prot.inp, par_all122_prot.inp and radius.str
*

if ?apbs .ne. 1 then stop

stream datadir.def

open read card unit 11 name @0top_all122_prot.inp
read rtf card unit 11
close unit 11

open read card unit 11 name @0par_all122_prot.inp
read para card unit 11
close unit 11

... read in or generate structure

coor orient

PBEQ
  set factor 0.939
  set sw      0.4
  stream @0radius.str
  scalar wmain add @sw
  scalar wmain mult @factor
  scalar wmain set 0.0 sele type H* end

```

```

APBS mgauto lpbe -
grdx 0.5 grdy 0.5 grdz 0.5 -
swin @sw srfm 2 -
calcene 2 calcfor 2 debug 0 -
sforce -
sele all END
END

skip none

dynamics leap verlet strt nstep 150 timestep 0.001 -
firstt 100.0 finalt 300.0 teminc 100.0 -
twindh 10.0

stop

```

### 5.3.3. Calculation and visualization of electrostatic potential

When this input file is run three DX files will be created. These can be visualized using several applications (for details please see APBS visualization guide<sup>2</sup>). The DX files will be `iapbs-pot.dx`, `iapbs-smol.dx` and `iapbs-charge.dx` which will contain electrostatic potential, solvent accessible surface and charge information, respectively.

```

* visualization.inp
* external files: top_all22_prot.inp, par_all22_prot.inp and radius.str
*

!if ?apbs .ne. 1 then stop

stream datadir.def

open read card unit 11 name @0top_all22_prot.inp
read rtf card unit 11
close unit 11

open read card unit 11 name @0par_all22_prot.inp
read para card unit 11
close unit 11

... read in or generate structure

coord orient

PBEQ
  stream @0radius.str
  set factor 0.939
  set sw 0.4
  scalar wmain add @sw

```

```
scalar wmain mult @factor
scalar wmain set 0.0 sele type H* end
scalar wmain show

APBS mgauto lpbe dimx 65 dimy 65 dimz 65 -
cglx 30 cgly 30 cglz 30 fglx 15 fgly 15 fglz 15 -
calcene 1 calcforce 0 -
ionq1 1.0 ionc1 0.15 ionr1 2.0 ionq2 -1.0 ionc2 0.15 ionr2 2.0 -
wpot wsmol wchg -
debug 1 -
sele all END
END

set elstatenergy = ?ENPB

stop
```

## Notes

1. <http://apbs.sourceforge.net/doc/user-guide/x2443.html#visualization-sect>
2. <http://apbs.sourceforge.net/doc/user-guide/x2443.html#visualization-sect>