DelPhi v.6- The New Macromolecular Electrostatics Modeling Package

This manual describes the main features of the old program, as well as the new features. Whenever possible, we have preserved compatibility with previous versions of DelPhi. People who are used to older versions of DelPhi should not encounter any difficulties in this one. DelPhi is a software package that calculates electrostatic potentials in and around macromolecules or geometrical objects. It can solve the non-linear and linear forms of the Poisson Boltzmann equation using finite difference methods on a GSZxGSZxGSZ cubical lattice. The user can specify the size of the ion exclusion (or Stern) layer around the molecule and a variable probe radius to define the solvent accessible surface. Different objects and molecules (or a combination of them) can be specified using their own dielectric constant. Various boundary conditions such as periodic and focusing can be used to model different systems like long periodic molecules or cell membranes. The output from the program can be used to calculate molecular interactions, changes in pKa, solvation energies and many other properties of interest.

Authors:

Delphi is maintained and developed by Delphi team:
email: delphi@clemson.edu

References:

The following references should be quoted if the use of the DelPhi v.6 results to a publication.

In particular, reference 1 describes some of the new features introduced in DelPhi v.6 and references 2 and 3 describe the implementations of parallel conmputing in parallel DelPhi V.6.

- L. Li, C. Li, S. Sarkar, J. Zhang, S. Witham, Z. Zhang, L. Wang, N. Smith, M. Petukh, E. Alexov, "DelPhi: a comprehensive suite for DelPhi software and associated resources", BMC, Biophys, (2012) May 14; 4(1):9.
- Smith N, Witham S, Sarkar S, Zhang J, Li L, Li C, Alexov E. "DelPhi Web Server v2: Incorporating atomic-style geometrical figures into the computational protocol", Bioinformatics. 2012 Apr 23.
- L. Li, C. Li, Z. Zhang, E. Alexov, "On the Dielectric Constant of Proteins: Smooth Dielectric Function for Macromolecular Modeling and its Implementation in DelPhi", J. Chem, Theory Comput. 2013 Apr 9; 9(4): 2126-2136.
- C. Li, L. Li, J. Zhang, E. Alexov, "Highly efficient and exact method for parallelization of grid-based algorithms and its implementation in DelPhi", J. Comput Chem, 2012 Sep 15: 33(24): 1960-1966.
- C. Li, M. Petukh, L. Li, E. Alexov, "Continuous Development of Schemes for Parallel Computing of the Electrostatics in Biological Systems: Implementation in DelPhi", J comput chem (2013), Article first published online: 3 JUN 2013 DOI: 10.1002/jcc.23340.
- Rocchia, W.; Alexov, E.; Honig, B. "Extending the applicability of the nonlinear Poisson-Boltzmann equation: Multiple dielectric constants and multivalent ions" J Phys. Chem. B 105, 6507-6514 (2001) (pdf)
- W. Rocchia, S. Sridharan, A. Nicholls, E. Alexov, A. Chiabrera and B. Honig "Rapid Grid-based Construction of the Molecular Surface for both Molecules and Geometric Objects: Applications to the Finite Difference Poisson-Boltzmann Method" *J. Comp. Chem.* 23, 128-137,2002

Additional references:

- Klapper, I., Hagstrom, R., Fine, R., Sharp, K., Honig, B. (1986). Focusing of electric fields in the active site of Cu-Zn Superoxide Dismutase: Effects of ionic strength and amino-acid modification. Proteins 1, p 47.
- K.A. Sharp, M.K.Gilson, R.M.Fine and B.H. Honig. (1987). Electrostatic interactions in proteins. UCLA Symposium on Molecular and Cell Biology, Vol 69: Protein Structure, Folding and Design, Ed. D.L. Oxender, p235.
- Gilson, M., Sharp, K., Honig, B. Calculating electrostatic interactions in bio-molecules: Method and error assessment. J. Computational Chem. 9, pp327-335.
- Gilson, M., Honig, B. Total Electrostatic Energy of a Protein. Proteins, 4, p7 (1988).

- B. Jayaram, K.A.Sharp and B.H.Honig. The electrostatic potential of B-DNA. Biopolymers, 28, p975 (1989).
- K. Sharp, and B. Honig. Lattice Models of Electrostatic Interactions: The Finite Difference Poisson-Boltzmann Method. Chemica Scripta, 29A:71 (1989)
- K. Sharp, and B. Honig. Electrostatic Interactions in Macromolecules: Theory and Applications. Ann. Rev. Biophys. Chem. 19:301-32 (1990).

The original reference to the use of the finite difference method for macromolecular electrostatics is:
- J. Warwicker and H.C. Watson, J. Mol. Biol., 157, p671 (1982).

Table of Contents

1. INTRODUCTION

DelPhi takes as input a Brookhaven database coordinate file format of a molecule or equivalent data for geometrical objects and/or charge distributions and calculates the electrostatic potential in and around the system, using a finite difference solution to the Poisson-Boltzmann equation. This can be done for any given concentration of any two different salts. The system can be composed of different parts having different dielectrics.

Return to TOC

2. INSTALLATION

DelPhi v.6 is distributed in four versions:

IRIX version, compiled under IRIX 6.5 Operating System, 32bits, using f77 and cc compilers.

IRIX version, compiled under IRIX 6.5 Operating System, 64bits, using f77 and cc compilers.

LINUX version, compiled under Red Hat 7.1, kernel 2.4.2 Operating System, using GNU gfortran compilers,

PC version, compiled under Windows Operating System, using Microsoft Developer Studio C++ and Fortran compilers.

Their way of working is very similar; however, unexpected differences may appear due to different numerical precision or to the porting of the software to different architectures; for example, at present, the elapsed time in the PC version is not calculated.

Each distribution contains one executable, named delphi or delphi.exe, the source codes with corresponding makefile when needed, and some worked examples.

3.   BASIC TUTORIAL

This section provides an overview of using DelPhi in an energy calculation. A quick introduction is given in the first section, and then details are given in the sections that follow.

Briefly, running DelPhi 1 consists of the following steps:

Prepare run parameters file   (named fort.10 or [namefile].prm)

If at least one molecule is going to be introduced then prepare at least three more files containing:

	Default convention	Alternative convention
Atom coordinates	fort.13	[namefile].pdb
Atomic radii	fort.11	[namefile].siz
Atomic charges	fort.12	[namefile].crg

Several sets of sample parameter files are provided with the distribution, so it is not necessary to generate them from scratch. These include PARSE, CHARMM and Amber charge and radii files. They all are developed for Brookhaven protein databank (pdb) files with hydrogens. Thus, for successful modeling, the input pdb file should be protonated prior to running DelPhi. If the accuracy of the calculations is not crucial, then using the unprotonated pdb file is possible, using the proper charge (crg) and radii (siz) files. Importantly, the names of the atoms and residues should be consistent between the pdb, crg and siz files.

In the simplest case, DelPhi is applied to a single molecule in a pdb file. To do this, the pdb file can be renamed to fort.13 in the directory where DelPhi will be run, or the following line can be added to the run parameters (prm) file:

```
in(pdb,file="[namefile].pdb")
```

Likewise, the crg, siz, and other parameter files should be set up in a similar manner. This will be discussed in more detail in Statement and Functions.

After the input and the parameter files have been properly set up, DelPhi can be run from the command line. For instance if one wants to use the parameter file "test.prm" as the parameter file one types:

```
delphi test.prm
```

Typing only:

```
delphi
```

defaults to `fort.10`. Any additional parameters after test.prm are ignored.

Run the program DELPHI in batch or interactively directing the output to unit 6 (standard output) or into a log file, if necessary. Example:

```
delphi > out.log
```

Analyze the results. The primary output file from the program is a three dimensional array of potentials calculated at the lattice points. This is a large file (grid size)^3 and is written in binary to save space and time. Much more information from the run can be extracted and saved in suitable files. Delphi prints out the grid energy, reaction field energy, coulombic interaction energy. These energies can be used for variety of biophysical applications.

As an option, the site coordinates (frc) file can be provided in order to collect the potential and electrostatic field components at specific positions. It has the same format as a pdb file. The calculated potential and the components of electrostatic field will be reported at the positions of atoms given in the frc file. (Warning – do not charge the atoms specified in frc file).

The most important file is the parameter file. It contains the parameters that control the run and output files. Lines within the parameter file can be either Statement or Functions.

This manual will describe meaning and structure of the Statements and Functions, together with a description of input/output file naming and format, energy calculation, and a description of the new features available in DelPhi v.6. We also offer various advices on choosing parameters and using DelPhi.

Note: Older versions of the program provided utilities for file format conversion together with specific flags for output format. Unfortunately, not all these options have been tested and updated in the new version. However, most of them are expected to work properly, at least if the input is a single molecule with only one dielectric constant.

OVERALL PROGRAM FLOW:

- Header with time and date is written.

- Parameters are read from fort.10 or prm file and echoed to output.

- Radius data read from fort.11 or siz file and stored in hash table for efficient look up.

- Charge data read from fort.12 or crg file and stored in hash table for efficient look up.

- Atomic coordinates are read from fort.13 or pdb file and scaling is computed. In accordance with charge and size files, radius and charge are assigned to each atom. Distribution of dielectric values, ionic strength parameters and charge values over the lattice are determined from the coordinate/charge/radius data files.

- Arrays that describe 3D distribution of dielectric and ion accessibility in space are initialized.

- Atom file with charge and radii records are outputted to fort.19 if requested.

- Centres of + and - charge distributions, and net charge calculated for check on charge distribution.

- Arrays are set up for the difference eqn. iteration.

- Boundary values are set, either through analytical expressions or interpolated from the potential map read from fort.18.

- Linear then non-linear iterative relaxations are done and convergence histories are printed out as simple log/lin line plots, if requested.

- Potentials are converted to concentrations if requested.

- Potentials and fields are calculated at the coordinates of the atoms read from fort.15, and outputted to fort.16 if requested.

- Grid of potentials outputted to fort.14, if requested.

- Energy contributions and overall surface induced polarization charge are printed out.

- The dielectric map is outputted to fort.17, if requested.

[Return to TOC](#)

4. Statements and Functions

DelPhi uses a command interpreter that allows commands to be used in the parameter file. The concept of the command in DelPhi comes in two forms, statements and functions. Statements have the form:

```
variable-name=value
```

e.g.

```
Scale=2.0
grid size=65
pbx=t
```

Functions have the form:

```
operation(specifier,file="xxx.yyy",format="abc")
```

e.g.

```
in(pdb,file="lys.pdb")
```

```
out(phi,unit=20,format=2)

center(file="test.pdf")
```

Statements simply set values or flags.

Functions tell DelPhi to perform an immediate operation using the specifiers in the function as parameters for the operation.

4.1   Syntax

In general, statements and functions can each be placed on a new line. Since this is the clearest way to organize statements, functions and comments, this is what we would recommend. However, several statements and functions can be placed on the same line, separated by commas ",", vertical bars "|" or colons ":". Comments can also be included on the same line as functions or statements. These are set apart by surrounding them with a pair of exclamation points "!". If a comment extends to the end of the line, then only a single exclamation point "!" is necessary. Spaces and capitalization are ignored only if they appear outside of quotation marks. A very long line can be split into two lines using the backslash "\". This is illustrated in the following examples:

```
scale=2.0, gridsize=65, center (file="mid.pdb")

in(pdb,file="lys.pdb") !this is a comment at the end of a line

In(pdb,file="Lys.pdb") !this line reads the file Lys.pdb, not lys.pdb

Scale=1.5 !this is a comment surrounded by two statements! probe
radius=1.4
```

Note that in the last example, both scale and probe radius will be set by DelPhi. Please be careful with the slightly unconventional use of the comment.

We have tried to anticipate some input errors and to inform the user of them, but the hardest part of every complex program is error handling. At the moment DelPhi will only pipe back to you what it doesn't understand and continue on with the program. If DelPhi does not understand

a command, it will attempt to use a default value and continue running anyway. Therefore, it would be worthwhile to pay attention to syntax to avoid running an unintended calculation.

Return to TOC

4.2   Shorthand and Longhand Statements

Many statements have abbreviated names. These may come in various forms from three to six letters long. Although longer descriptions are easier to read, the shorter forms are easier to type and as such they may be less prone to typing error. They are a matter of taste. A complete listing of abbreviations and full names appears in the Index of Statements.

Yes, No, Maybe:

When setting logical values the following are case insensitive and equivalent:

```
yes, on, true, t

no, off, false, f
```

Return to TOC

4.3   Functions in detail:

The present set of allowed functions is:

CENTER

ACENTER

READ/IN (Equivalent)

WRITE/OUT (Equivalent)

ENERGY

BUFFZ

SITE

QINCLUDE

We shall cover these one by one since they vary somewhat more in format than statements. But first, some of the common features:

```
Function(file="test.file)
```

will open the file test.file, whether for centering, output or input.

```
Function(unit=14)
```

will do the same but with fort.14 or whatever is linked to it.

```
Function(format=abc)
```

will perform operations on files with a particular format, or in a specified way. The default format is always zero (*i.e.* "0"). The format can be a number or a string. Users are advised not to change the format and to use the default settings.

Center

```
Center(0.2,3,2)
```

will offset the molecule by 0.2 grid units in the *x* direction, 3 in the *y* and 2 in the *z*. Center was created as a function to allow the following possibility:

```
Center(unit=15)
```

This opens `fort.15` (usually called frc file), reads its atoms and centers the current calculation using the geometrical center of the atoms in the file.

An alias for opening fort.15 and take is center as the system center is Center(999,0,0).

To read just the first atom of a file and use its coordinates use the following,

```
Center(file="whatever",an=1)
```

Center(999,999,0) is an equivalent of Center(unit=15,an=1)

Note that an=1 is a string and that an=n is not going to take the n-th atom position as the center.

Other aliases are : Center(777,0,0) for Center(unit=27) and Center(777,777,0) for Center(unit=27,an=1)

This function is used to specify the offset (expressed in grid units) with respect to the lattice center at which the center of the molecule [*pmid(3)*] is placed. This will influence what point in the real

space (expressed in Angstroms) is placed at the center of the grid [oldmid(3)]. The relationship between real space $r(i)$ and grid $g(i)$ coordinates for a grid size of *igrid*, with a scale of *gpa* grids/angstrom is as follows:

The centre of the grid is:

midg = (igrid+1)/2
oldmid(i) = pmid(i) - OFFSET(i)/gpa
g(i) = (r(i) - pmid(i))*gpa + midg + OFFSET(i)
r(i) = (g(i) - midg)/gpa + oldmid(3)

The scale, the system center and the shift are printed in the logfile.

Note that a certain error inevitably results from the mapping of the molecule onto the grid. By moving the molecule slightly (changing CENTER offset between 0,0,0 and 1,1,1) and repeating the calculations, it is possible to see whether the results are sensitive to the particular position on the grid, and if so, to improve the accuracy by averaging (this is related to rotational averaging, discussed in the J. Comp Chem paper of Gilson et al.). However using a larger scale is a more effective way of improving accuracy than averaging.

Acenter

`Acenter` takes three absolute coordinates, *i.e.* in Å and uses those as the center, so:

```
    Acenter(1.0,5.6,7.0)
```

centers the grid box at $x=1.0$Å, $y=5.6$Å, $z=7.0$Å.

Read/In

This function allows files to be read as input. It comes with several specifiers, namely:

SIZ: for the radius file

CRG: for the charge file

PDB: for the pdb structure file (possible alternative formats: frm=UN and frm=MOD)

MODPDB4: for the modified pdb structure file (possible alternative formats: frm=PQR and frm=MOD) which contain charges and radii values with 4 digits precession after decimal points.

FRC: for positions of site potentials.

PHI: for the phimap used in focusing

The main use, at present will be to give the user flexibility to specify the file name or unit number of any of these files. Note that the default files for all read (and write) operations are the standard DelPhi files.

Example:

```
in(modpdb4, file="test.mod",format=mod)
```

Read a mod file called "test.mod", which contains charge and radius value in 4 digits after decimal

```
in(modpdb4, file="test.pqr",format=pqr)
```

Read a pqr file called "test.pqr", which contains charge and radius value in 4 digits after decimal. Using this option, delphi can directly read PQR files which are generated by other programs (such like pdb2pqr program).

```
in(frc,file="namefile")
```

opens the file "namefile" and logically assigns to it the unit 15 (see Files for details)

Write/Out

Equally obviously this deals with output. The specifiers are:

PHI : for phimaps (possible other formats: frm=BIOSYM, frm=GRASP, frm=CUBE; see Unit14 in Files)

FRC : for site potentials (possible other formats: frm=RC, frm=R, frm=UN;)

EPS : for epsmaps

MODPDB: for [modified pdb files](#)

MODPDB4: modified pdb files that contain charges and radii with 4 digits precession after decimal points

UNPDB: for unformatted pdb file

UNFRC : for unformatted frc files

ENERGY: writes the file "energy.dat" containing energy data. (Example: out(energy))

-Note that this is different from the Energy function!

As an example of use,

```
write(eps)
```

writes an epsmap, to the file fort.17, format described [here](#).

```
out(modpdb, file="test.out")
```

writes a modified pdb file called **"test.out"**

```
out(modpdb4, file="test.mod",format=mod)
```

Writes a modified pdb file called "test.mod", which contains charge and radius value in 4 digits after decimal

```
out(modpdb4, file="test.pqr",format=pqr)
```

Writes a pqr file called "test.pqr", which contains charge and radius value in 4 digits after decimal

Note that all writes are turned OFF initially, including `eps` and `phi`.

Energy

At present it takes as its argument any of the following:

G or **GRID** for the grid energy,

S or **SOL** or **SOLVATION** for the corrected reaction field energy

C or **COULOMBIC** or **COU** for the coulombic energy

ION or **IONIC** or **IONIC_C** for the direct ionic contribution (see [Ionic direct Contribution](#))

separated by commas. (As always there is no case sensitivity here.)

So, for example,

```
energy(s,g,Cou,ion)
```

gives the solvation, coulombic, grid energies and ionic contribution.

Note that the calculation of the non linear contributions are automatically turned on whenever non-linear PBE solver is invoked.

For the energy definition we recommend the Rocchia et al. J. Phys. Chem, however a brief explanation is given below:

The **grid energy** is obtained from the product of the potential at each point on the grid and the charge at that point, summed over all points on the grid. However, the potential computed for each charge on the grid includes not only the potentials induced by all other charges, but also the "self" potential. The effect is caused by the partitioning of the real charges into the grid points. Thus, two neighboring grid points might have partial charges that originate from the same real charge. Since the product of a charge with its own potential is not a true physical quantity, the grid energy should not be taken as a physically meaningful number by itself. Instead, the grid energy is only meaningful when comparing two DelPhi runs with exactly the same grid conditions (e.g constant structure and constant scale). The difference can then be used to extract solvation energies, salt effects, and others.

The **coulombic energy** is calculated using Coulomb's law. It is defined as the energy required to bring charges from infinite distance to their resting positions within the dielectric specified for the molecule. This term has been revised in the new DelPhi to be consistent with the new multiple dielectric model. For the most recent definition, we again refer the reader to the previously mentioned paper.

The **reaction field energy** (also called the solvation energy) is obtained from the product of the potential due to induced surface charges with all fixed charges of the solute molecule. This includes any fixed charge in the molecule that happens to be outside of the grid box. The induced surface charges are calculated at each point on the boundary between two dielectrics, e.g. the surface of the molecule. If the entire molecule lies within the box and salt is absent, this energy is the energy of transferring the molecule from a medium equal to the interior dielectric of the molecule into a medium of external dielectric of the solution. Depending on the physical process being described, this may be the actual solvation energy, but in general the solvation energy is obtained by taking the difference in reaction field energies between suitable reference states - hence we make the distinction between this physical process and our calculated energy term.

For other Energy contributions, see [here](#).

<u>Site</u>

```
SITE(argument)
```

Reports the potentials and electrostatic field components at the positions of the subset of atoms specified in the frc file. The atoms specified in frc file should not be charged in the delphi run.

The argument is a list of identifiers that can be:

Atom or A
Charge or Q
Potential or P
Field or F
Reaction or R
Coulomb or C
Coordinates or X
Salt or I
Total or T

Examples:

```
SITE(atom,potentials)

Site(a,p) - specifies what printed to frc file (see above).
```

Buffz

Defines a box with sides parallel to grid unit vectors that the reaction field energy will then be calculated using ONLY the polarization charges contained in that box.

The fixed format is BUFFZ(6i3).

Example:

```
BUFFZ(001002003004005006) will fill a matrix:
```

```
Bufz(1,1)=1          distance in grid units from the negative x side

Bufz(2,1)=2          distance in grid units from the negative y side

Bufz(3,1)=3          distance in grid units from the negative z side

Bufz(1,2)=4          distance in grid units from the positive x side

Bufz(2,2)=5          distance in grid units from the positive y side

Bufz(3,2)=6          distance in grid units from the positive z side
```

Qinclude

The `qinclude` function is a feature that has not been tested in the latest versions of DelPhi, so it may behave a bit differently from expected. It works in the same way as an include statement works in `FORTRAN` or `C`, *i.e.*, it inserts lines from another file into the current one. For instance, suppose we have the following files:

`test.prm`:

```
scale=3.0, write(frc),write(modpdb,file="test.out")

acenter(0.123,4.55,2.34)
```

```
test2.prm:

        boundary type=2, read(pdb,file="test.pdb")
```

then the file:

```
        scale=3.0, write(frc),write(modpdb,file="test.out")

        qinclude(test2.prm)

        acenter(0.123,4.55,2.34)
```

is equivalent to:

```
        scale=3.0, write(frc),write(modpdb,file="test.out")

        boundary type=2, read(pdb,file="test.pdb")

        acenter(0.123,4.55,2.34)
```

or one could even write:

```
        qinclude(test1.prm)

        qinclude(test2.prm)
```

Clearly the motivation behind this form is to allow the user to create his/her own default file and qinclude this file at the beginning of subsequent parameter file. One then needs only a `qinclude` statement plus and lines indicating those parameters that need to be changed from the default file. Note that qinclude is immediate, *i.e.* it includes the lines from the indicated file at the position of the qinclude command. This is important to remember that if you define a quantity multiple times, then only the last instance is used. In other words, a file containing

```
        scale=2.0

        scale=3.0
```

tells DelPhi to set the scale to 3 grids/Å. This is the reason we include a `write(specifier,off)` command. If you have a default file which *enables* a write, you can still turn it off without modifying the default file.

Can a qinclude file contain a qinclude file? But of course. At present one can nest qinclude files up to ten deep. If a qinclude file does not exist DelPhi will tell you so and move on to the next command. If there is no file passed to qinclude, *i.e.*

```
    qinclude()
```

then, if it exists, the default include file `~/qpref.prm` is passed. Qinclude is a special command and as such *always requires its own line*, *i.e.* do NOT add more commands to a line which start with a qinclude command (not even comments).

<u>INSOBJ</u>(Removed and OBJECTS are no longer supported! Instead users are suggested to use Protein_Nano Object Integrator (ProNOI) http://compbio.clemson.edu/downloadDir/ProNO_Integrator.zip to create, visualize and manipulate atomistic-style objects and use them in conjunction with standard Protein Data Bank files.)

This function is somehow different from the others in the sense that it doesn't have any argument, if it is written in a line of a prm file, it launches the routine that allows the user to insert objects, charge distributions etc. (see [description](#))

[Return to TOC](#)

4.4   Index of Statements and their shorthand

Statement Long Form	Short	2L abr	Default Value
AUTOCON AUTOCONVERGENCE AUTOMATICCONVERGENCE	[AUTOC](#)	AC	TRUE
BOUNDARYCONDITION BOUNDARYCONDITIONS	[BNDCON](#)	BC	2(=DIPOLAR)
BOXFILL PERCENTFILL	[PERFIL](#)	PF	80

PERCENTBOXFILL			
CHEBIT	CHEBIT	CI	FALSE
CLCSRF	CLCSRF	CS	FALSE
CONVERGENCEFRACTION	CONFRA	CF	1
CONVERGENCEINTERVAL	CONINT	CI	10
EXITUNIFORMDIELECTRIC	EXITUN	XU	FALSE
EXTERIORDIELECTRIC EXTERNALDIELECTRIC	EXDI	ED	80
FANCYCHARGE SPHERICALCHARGEDISTRIBUTION	FCRG	FC	FALSE
GRIDCONVERGENCE	GRDCON	GC	0.0
GRIDSIZE	GSIZE	GS	AUTOMATIC
INTERIORDIELECTRIC	INDI	ID	2.0
IONICSTRENGTH SALTCONC SALTCONCENTRATION	SALT	IS	0.0
IONRADIUS	IONRAD	IR	0.0/2.0
ITERATION ITERATIONS LINEARITERATION	LINIT	LI	AUTOMATIC
LOGFILECONVERGENCE	LOGGRP	LG	FALSE
LOGFILEPOTENTIALS	LOGPOT	LP	FALSE
MAXC (new!)	MAXC	XC	0.
MEMBRANEDATA	NOT USED	MD	FALSE
NONLINEARITERATION	NONIT	NI	0

NONLINEARITERATIONS			
PERIODICBOUNDARYX	PBX	PX	FALSE
PERIODICBOUNDARYY	PBY	PY	FALSE
PERIODICBOUNDARYZ	PBZ	PZ	FALSE
PHICON	PHICON		FALSE
PROBERADIUS	PRBRAD	PR	1.4
RADPLOEXT (new!)	RADPOL	RL	1.
RADPR2 (new!)	RADPR2	R2	PRBRAD
RELAXATIONFACTOR	RELFAC	RF	0.9975
RELPAR (new!)	RELPAR	RR	1.
RMSC (new!)	RMSC	MC	0.
SALT2 (new!)	SALT2	S2	0.
SCALE	SCALE	SC	1.2
SOLVPB	SOLVPB	SP	TRUE
VAL+1 and similar (new!)	VAL+1	+1	1

4.5.1 Full list and Statement description

GSIZE: An odd integer number of points per side of the cubic lattice, min=5, max=571 (=NGRID, platform dependent). A larger grid size will in general mean a better resolution representation of the molecule on the lattice. This will results in more accurate potentials, but will require more time. The number of iterations required to reach a certain convergence will increase approximately linearly with parameter GS. Since the time per iteration will go up as the cube of this parameter the amount of calculation will thus increase at about the fourth power of GS. Example: `gsize=65 or gs=65`.

SCALE: The reciprocal of one grid spacing (grids/angstrom). Example: `scale=1.2 or sc=1.2`.

PERFIL: A percentage of the object longest linear dimension to the lattice linear dimension. This will affect the scale of the lattice (grids/angstrom). The percentage fill of the lattice will depend on the application. A large percentage fill will provide a more detailed mapping of the molecular shape onto the lattice. A <u>perfil less than 20% is not usually necessary or advisable</u>. A very large filling will bring the dielectric boundary of the molecule closer to the lattice edge. This will cause larger errors arising from the boundary potential estimates, <u>which are set to zero or approximated by coulombic/Debye-Huckel-type functions using a uniform solvent dielectric</u>. The error will be minimal for higher salt concentrations or weakly charged molecules. Smaller percentages will increase the accuracy of the boundary conditions, but result in a coarser representation of the molecule. Higher resolution can be achieved more efficiently using <u>focusing</u>.
Example: `perfil=40 or pf=40`.

NOTES:

If the molecule is not centered in the origin of the coordinate system, the perfil reflects the percentage of the system that is actually contained in the lattice. For example, if the maximum dimension of a molecule is 100Å, there is no offset and perfil is 50%, then the box side will be 200Å; but if there is an offset of 20Å in the maximum dimension direction, then the box side will be 280Å. (new!)

Scale, grid size and perfil are not independent variables so they cannot all be assigned simultaneously in a single run. In any quantitative calculation, the largest possible scale should be

used, preferably greater than 2 grids/angstrom. Without focusing, a perfil of around 50% or 60% is reasonable. For example, if scale is set to 2 and perfil is set to 50, the grid size is calculated automatically given the size of the structure. For larger molecules this could mean a prohibitively large memory requirement. In this case a compromise must be found or focusing could be used. Regardless of grid scale, calculations should be repeated at different scales to assess the size of lattice resolution errors.

A good approach to the calculation could start with a small percentage, say 20%, using *Debye-Huckel* boundary conditions, and then focus in to say 90% or more, in one (or two) stages, using focusing boundary conditions for the second (and third) runs. It is not necessary for the molecule to lie completely within the grid although then the potential boundary conditions must be generated by focusing. **However when calculating solvation energies with box fills of > 100% remember that unexpected results may be obtained since parts of the surface, (and perhaps some charges) are not included in the grid**.

INDI: The internal (molecules) dielectric constant. It is used only in single molecule systems for compatibility with the old version. A value of INDI=1 corresponds to a molecule with no polarizability- the state assumed in most molecular mechanics applications. INDI=2 represents a molecule with only electronic polarizability (i.e. assuming no reorientation of fixed dipoles, peptide bonds, etc). A value of 2 is based on the experimentally observed high frequency dielectric behavior of essentially all organic materials. INDI=4-6 represents a process where some small reorganization of molecular dipoles occurs which is not represented explicitly (for example in modeling the effects of site directed mutagenesis experiments, when the structure of the wild type, but not mutant protein is known). According to M.K. Gilson and B. Honig, *Biopolymers*, 25:2097 (1986) for instance, materials having similar dipole density, dipole moment and flexibility as globular proteins have a dielectric between 4 and 6. In modeling any process where large reorientations of dipoles, or large conformational change occurs, *i.e.* upon folding or denaturation, using a simple dielectric constant for the molecule would be inappropriate, and the change in conformation should be modeled explicitly.

Example: `indi=2 or id=2.`

EXDI: The external (solution) dielectric constant. A value of EXDI=1 corresponds to the molecule in vacuum, EXDI=80 to the molecule in water. Depending on the application runs with EXDI equal to either of these values may be used to represent different states in a thermodynamic cycle. Example: `exdi=80 or ed=80.`

PRBRAD: A radius (Å) of probe molecule that will define solvent accessible surface in the Lee and Richard's sense. In combination with the atomic van der Waals radii in the siz file, PRBRAD determines the regions of space, and hence the lattice points, that are inaccessible to solvent molecules (water). Suggested value is PRBRAD 1.4 for water. To understand how these parameters work, you should be familiar with the concepts of contact and solvent accessible surface, as discussed by *Lee and Richards,* and by *Mike Connolly*. For the purpose of DelPhi, any region of space that is accessible to any part of a solvent (water) molecule is considered as having a dielectric of EXDI. A value of zero for PRBRAD used with a siz file containing the standard van der Waals radii values will assign any region of space not inside any atom's van der Waals sphere to the solvent. For more details, please refer to Rocchia et al. *J. Comp. Chem.* paper.

IONRAD: The thickness of the ion exclusion layer around molecule (Å). IONRAD, in combination with the atomic van der Waals radii in the siz file, determines the regions of space, and hence the lattice points, which are inaccessible to solvent ions. Suggested values is IONRAD = 2.0 for sodium chloride. For the purpose of DelPhi, a <u>solvent ion is considered as a point charge</u>, which can approach no closer than its ionic radius, IONRAD, to any atoms van der Waals surface. The ion excluded volume is thus bounded by the contact surface, which is the locus of the ion centre when in van der Waals contact with any accessible atom of the molecule. A zero value for IONRAD will just yield the van der Waals surface. A non zero value of IONRAD will thus introduce a Stern, or ion exclusion layer, around the molecule where the solvent ion concentration will be zero and whose dielectric constant is that of the solvent, EXDI. Example: `ionrad=2 or ir=2.`

SALT: The concentration of first kind of salt,(moles/liter). In the case of a single 1:1 salt, it coincides with ionic strength. Example: `salt=0.14 or is=0.14.`

BNDCON: An integer flag specifying the type of boundary condition imposed on the edge of the lattice. Example: bndcond=4 or bc=4. Allowed options:

(1) - potential is zero.

(2) – dipolar. The boundary potentials are approximated by the Debye-Huckel potential of the equivalent dipole to the molecular charge distribution. Phi is the potential estimated at a given lattice boundary point, q+ (q-) is the sum of all positive (negative) charges, and $r_+(r_-)$ is distance from the point to the center of positive (negative) charge, lambda is the Debye length.

$$\phi = q_+ \frac{e^{-\frac{r_+}{\lambda_D}}}{\varepsilon_{solv} r_+} + q_- \frac{e^{-\frac{r_-}{\lambda_D}}}{\varepsilon_{solv} r_-}$$

(3) – focusing. The potential map from a previous calculation is read in unit 18, and values for the potential at the lattice edge are interpolated from this map- clearly the first map should have been generated with a coarser grid (greater distance between lattice points) and positioned such that current lattice lies completely within old lattice or the program will protest. For focusing boundary conditions, the program reads in a potential map from a previous run, and compares the scale of the focusing map with that for the current run. If they are the same, it assumes that this is a continuation of a previous run, and iteration of the potentials contained in the previous potential map is continued. If the scale is not the same, it checks to ensure that the new lattice lies completely within the old lattice before interpolating the boundary conditions.

(4) – coulombic. They are approximated by the sum of Debye-Huckel potentials of all the charges. q_i is the i'th charge, and r_i is the distance from the lattice boundary point to the charge.

$$\phi = \sum_i q_i \frac{e^{-\frac{r_i}{\lambda_D}}}{\varepsilon_{solv} r_i}$$

LINIT: An integer number (> 3) of iterations with linear equation. The convergence behavior of the finite difference procedure is reported in the log file as both the mean and maximum absolute change in potential at the grid points between successive iterations. The latter is probably more important since it puts an upper bound on how much the potential is changing at the grid points. It is suggested that sufficient iterations be performed to give a final maximum change of less than 0.001 kT/e. The number of iterations per se is not important, as long as its sufficient to give the required convergence. The convergence behavior can also be judged from the slope of the semi-log

plot of the mean and max changes given in the log file. LINIT is best determined by experience, since the convergence rate depends on several factors. Start with say 100 iterations, and then increase the number of iterations until sufficient. Note that a run can be restarted by using focusing boundary conditions with exactly the same SCALE, PERFIL and ACENTER values (see note 5). Some guidelines are: The number of iterations needed will increase with grid size (GSIZE). It will decrease with decreasing PERFIL, since the potentials converge more rapidly in the solvent. It will decrease with increasing ionic strength. The number is fairly insensitive to the size and number of charges on the molecule. Example: `linit=400 or li=400`.

NONIT: An integer number ($> = 0$) of non-linear iterations. If linear PB equation only is required, NONIT is set to be 0. Example: `nonit=400 or ni=400`.

FCRG: A flag, normally set to false indicating a linear cubic interpolation of charges to grid points; set to true this turns on a spherical charge interpolation. If an atomic charge does not lie exactly on a grid point, then it must somehow be distributed onto the grid points. If this flag is set false, the standard algorithm is used which distributes a charge to the nearest 8 grid points (quick and simple, see the Proteins paper of Klapper et al.). If this flag is set true, then an algorithm is used which gives a more spherically symmetric charge distribution, although the charge is now spread over a wider region of space. For certain cases this gives higher accuracy for potentials less than 3 grid units from a charge (see Gilson et al. J.Comp. Chem paper), although this point has not been exhaustively explored.

LOGPOT: A flag that activates the potential listing during the run.
Example: `logpot=t or lp=t or logfilepotentials=t`.

LOGGRP: A flag that activates the convergence plot during the run.
Example: `loggrp=t or logfileconvergence=t or lg=t`.

CONINT: A flag that determines at what iteration interval convergence is checked, by default it equals 10.(usually not modified from default) The idea behind this parameter is to allow

convergence to be checked less frequently to reduce the amount of time spent. Example: conint=10 or ci=10 or convergenceinterval=10.

CONFRA: A flag that determines the convergence fraction. Iit decides what fraction of grid points are used in assessing convergence (1=all, 2=half, 5=fifth etc). By default it equals 1 (usually not modified from default). Example: `confra=10 or cf=1 or convergencefraction=1`.

PBX,PBY,PBZ: They are the three logical flags (t/f) for periodic boundary conditions for the x,y,z edges of the lattice respectively. Note that periodic boundary conditions will override other boundary conditions on edges to which they are applied. Periodic boundary conditions can be applied in one or more of the x, y or z directions. When applied, the potential at each periodic lattice boundary point is iterated by supplying its missing neighbor(s) from the corresponding point on the opposite edge of the lattice. This can be used for example to model an infinite length of DNA. Assume that the helical axis of the DNA in the pdb file is aligned along the Z axis. The periodic boundary flags are set to false, false, true, and the percent fill of the box, PERFIL, is adjusted so that an integral number of turns just fill the box in the Z direction. Normal boundary conditions are applied to the X,Y boundaries. By setting two, or three of the boundary flags to true, one can simulate 2 dimensional or 3 dimensional cubic lattices of molecules. Example: `pbx=t or px=t or periodicboundaryx=t`.

AUTOC: A flag for automatic convergence. The program by default will automatically calculate the number of iterations needed to attain convergence. It is automatically set if no number of iteration is specified otherwise. See also LINIT and GC options Example: `autoc=t or automaticconvergence=t or autoconvergence=t or autocon=t or ac=t`.

EXITUN: A flag to terminate the program if uniform dielectric is present (INDI=EXDI). By default it is false. (usually not modified) Example: `exitun=f or exituniformdielectric=f or xu=f`.

GRDCON: The value for grid convergence. When set, the criterion used to stop the iterative process is the difference on values of grid energy, this option might slow down the calculation a bit,

but provides a very strong criterion. Example: `grdcon=0.001 or gc=0.001 or gridconvergence=0.001`.

RELFAC: The externally assigned value for spectral radius (define spectral radius). (usually not modified from default) Example: `relfac=0.9975 or relaxationfactor=0.9975 or rf=0.9975`

CHEBIT: A flag, that if it is true the relaxation parameter for linear convergence process is set equal to 1. (usually not modified from default) Example: `chebit=t or ci=t` .

SOLVPB: A flag, which controls the Poisson-Boltzmann solver. Normally DelPhi will invoke the Poisson-Boltzmann solver but if you are interested in using DelPhi for other things such as calculating surface area or producing a GRASP viewable surface file, you can turn off the solver using this option. Example: `solvpb=t or sp=t`.

CLCSRF: A flag, that when set to true, outputs a GRASP viewable surface file in the name grasp.srf. Example: `clcsrf=t or cs=t`.

PHICON: A flag, that maps charge density in a .phi file, with a procedure that is equivalent to the one that saves the potential map. phicon=f produces standard potential output in kT/e (approximately equal to 25.6 mV at 25oC, or to 0.593 kcal/mole of charge). phicon=t will give net solvent ion concentration output in M/l, where for every lattice point inside the molecule the concentration is 0, and the outside concentration is obtained from: (-ionic strength*2*sinh(potential)) or its linearized version if linear PBE is used. Example: `phicon=t.`

RADPOLEXT: A default radius for point charges in a continuum (only in objects) (see self-reaction field energy), Example: `radpolext=1 or radpol=1 or rl=1`

RELPAR: A manually assigned value for relaxation parameter in non-linear iteration convergence process. (see non-linear equation convergence) Example: `relpar=0.8 or rr=0.8`

SALT2: The concentration of second salt (if present) expressed in Moles/liters. (see [multi-salt option](#)) Example: `salt2=0.2 or s2=0.2`

RADPR2: The value for effective probe radius relative to the part of the molecule which is internal to an object. (see [geometric objects](#)) (This option is still under testing on Jan 2002) Example: `radpr2=2 or r2=2.`

VAL+1: (VAL-1 VAL+2 VAL-2) A number > 0, valence of positive (negative) ion constituting salt one (two). (see [multi-salt option](#)) Example: `val+1=1 or +1=1.`

RMSC: The convergence threshold value based on root mean square change of potential. (see [convergence hints](#)) Example rmsc=0.0001 or mc=0.0001

MAXC: The convergence threshold value based on maximum change of potential (suggested). (see [convergence hints](#)) Example maxc=0.0001 or xc=0.0001

[Return to TOC](#)

5.   FILES

Many files are used to input and output data. Each type of data or output has its own file. The default name for the **unit [number]** file is **fort.[number]**. Here follows a description and format for each of them.

INPUT FILES:

UNIT 10

Default extension prm. Contains input parameters. See the section on [Statements and Functions](#) for full details.

UNIT 11

Default extension *siz*. List describing the van der Waals radii to be assigned to each atom/residue pdb record type. A sample file is provided together with the code. Note the atom and residue fields ignore case and leading blanks. The residue field may be left blank (wild card), causing a match with the given atom type of any residue. ONLY if the residue field is left blank, the LAST 5 characters of the atom record may be left blank. In this case all atom types beginning with the letter in column 1 will be matched. Records of greater specificity override those of less specificity. Beware of ambiguities like calcium (ca) and alpha carbon! All atoms of an input *pdb* file must be assigned a radius through the *siz* file, even if it is 0, or the output will be flagged with a warning.

UNIT 12

Default extension crg. List of the atomic charges to be assigned to each atom/residue/number/chain pdb record type. A sample file is provided together with the code. The ascii fields for atom, residue, number and chain ignore case and leading blanks. Any field except the atom name may be left blank and will be treated as a wild card. Records of greater specificity override those of lesser specificity as for the *siz* file above.

search order:

```
atom_res_num_chain

atom_res_num_____

atom_res_____chain

atom_res_____

atom_____num_chain

atom_____num_____

atom_____chain

atom_____
```

Atoms that do not find a match in the *crg* file will be neutral (q=0.0)

file must have a line: atom__resnumbc_charge_

Examples:

A line as shown below will charge only the N atom of ALA residues.

```
N     ALA      -0.400
```

A line as show below will charge all N atoms.

```
N                  -0.400
```

Note that position of text phases and numbers is strictly determined and can't be changed!

UNIT 13

A Brookhaven protein data bank standard format file containing atom labels and coordinates, or a modified OBJECTFILE. Only records starting with ATOM or HETATM are read; if objects or multi-dielectric option are used, also the keywords MEDIA, OBJECT, CRGDST, DATA are also read. The default extension is *pdb*. The precise format is essential; using Fortran syntax, (6A1,I5,1X,A4,A1,A3,1X,A1,I4,A1,3X,3F8.3,2F6.2,1X,I3) is used for the atom record. From left to right, the fields contain 'ATOM--' or 'HETATM' atom serial number, atom name, alternate location indicator, residue name, chain identifier, residue sequence number, residue insertion code, x, y, and z coordinates, occupancy, temperature factor, footnote number. Note that the program treats the residue number as an ascii string, not as an integer. As a warning to the user, there are many variations, and even outright errors found in the format of pdb files obtained from the web. It would be wise to double-check the contents of a file to save any heartache.

UNIT 15

Default extension: *pdb* or *frc*. List of coordinates where site potentials are output in Unit 16. Format as for Unit 13.

UNIT 18

Default extension phi, potential map for focusing boundary conditions. Potentials are in kT/e (25.6mV, 0.593 kcal/mole/charge at 25 ℃).

The format of the file is given below in case that the user wants to adopt the file to its own software. If the users wants to visualize the file with Grasp or Insight, no action should be taken.

```
unformatted (binary file)

character*20 uplbl

character*10 nxtlbl,character*60 toplbl

real*4 phi(65,65,65)

character*16 botlbl

real*4 scale,oldmid(3)
```

uplbl, nxtlbl, toplbl, botlbl are ascii information. Phi is the 3D array containing values of potential for all the lattice points. Index order is *x,y,z*. Scale is lattice scale in *grid/Å*. Oldmid is the *x,y,z* coordinates in real space (angstroms) of the centre of the lattice: thus the real space coordinates x,y,z of the lattice point for phi(IX,IY,IZ), for the case where IGRID = 65, are:

```
x = (IX - 33)/scale + oldmid(1)

y = (IY - 33)/scale + oldmid(2)

z = (IZ - 33)/scale + oldmid(3)
```

where 33 = (65+1)/2 is the middle point of the grid.

OUTPUT FILES:

UNIT 6

Output from the program, including error messages and convergence history. When run interactively, appears on standard output. Default extension log when run in batch

UNIT 14

If the flag IBIOS (BIOSYM) is false, then output is in DELPHI format, default extension phi. The output can be either a potential map or a concentration map, with format same as for unit 18 above. The output phi map has the same scale as used in the calculation (i.e, variable) *unless* `format=grasp` is specified. The grasp-style phi map format will always interpolate to a 65 x 65 x 65 grid for use in Grasp (or other hardwired display/analysis programs).

If the flag IBIOS (BIOSYM) is true, then output is in INSIGHT format, default extension ins. This is an unformatted (binary) file. As it was explained above, the format is provided only for completeness in case that one wants to visualize the file with different than Insight software.

```
character*132  toplbl     !ascii header
integer*4  ivary          !0  =>  x  index  varys  most rapidly
integer*4 nbyte           !=4, # of bytes in data
integer*4 inddat          !=0, floating point data
real*4 xang,yang,zang     !=90,90,90 unit cell angles
integer*4 intx,inty,intz  !=igrid-1, # of intervals/grid side
real*4 extent             !maximum extent of grid
real*4 xstart,xend        !beginning, end of grid sides
real*4 ystart,yend        !in fractional
real*4 zstart,zend        !units of extent
write(14)toplbl
write(14)ivary, nbyte, intdat, extent, extent, extent,
xang, yang, zang, xstart, xend, ystart,  yend,  zstart,
     zend, intx, inty, intz
do k = 1,igrid
    do j = 1,igrid
        write(14)(phimap(i,j,k),i=1,igrid)
    end do
end do
```

Note that for grid sizes less than 65, INSIGHT format files will occupy less disk space than the corresponding DELPHI files. ins files are designed as input to a Biosym Corp. stand alone utility called CONTOUR, supplied with INSIGHT Version 2.4. This program will produce contour files for display with INSIGHT. 33

If the flag CUBE is true, then output is in CUBE format (Gaussian Cube). Example:
`Out(phi,file='phimap.txt',form='cube')` – this command creates file 'phimap.txt' in the cube-format.

There is a source code for saving in the 'cube-format'.

```
write(6,*)' Potential map in cube format '
write(6,*)'written to file',filnam
write(14,*)'qdiffxs4 with an improved surfacing routine'
write(14,*) 'Gaussian cube format phimap'
coeff=0.5291772108
stepsize=1.0/scale
do i=1,3
origin(i)=oldmid(i)-stepsize*(igrid-1)/2/coeff
enddo
write(14,'(i5,3f12.6)') 1, (origin(i),i=1,3)
write(14,'(i5,3f12.6)') igrid, stepsize/coeff,0.0,0.0
```

```
write(14,'(i5,3f12.6)') igrid, 0.0,stepsize/coeff,0.0
write(14,'(i5,3f12.6)') igrid, 0.0,0.0,stepsize/coeff
write(14,'(i5,4f12.6)') 1,0.0,0.0,0.0,0.0
do i = 1,igrid
      do j = 1,igrid
            write(14,'(6E13.5)')(phimap(i,j,k),k=1,igrid)
      end do
end do
```

UNIT 16

Default extension *frc*. A list of potentials and fields at coordinates in *pdb* file read on unit 15.

Format: 12 lines of ascii header information, followed by a variable number of records written as:

```
230       format(8G10.3)

          write(16,230)xo,chrgv,phiv,fx,fy,fz
```

where *xo(3)* are *x ,y ,z* coordinates of charge, *chrgv* is the charge value, *phiv* is the potential (*in kT/e*) at that point, and *fx, fy, fz* are the field components (in *kT/e/Å*). The last line of the file is the sum of chrgv*phiv/2 over all the charges in the file. This quantity can be used for calculating solvation and interaction energies.

UNIT 17

Dielectric bit map, default extension: *eps*. If grid size=65, there are 3*65*65*65 lines joining neighboring grid points, 65*65*65 each in of the *x,y,z* directions. The midpoint of each line is given a value of 1 if it lies within the solvent accessible volume of the system, 0 if outside. This defines the shape of the molecule and separates the space into different dielectric regions. The format of the output files is described below in case that the user wants to build own software to visualize the map. For compact output purposes the array of INTEGER*4, epsmap(65,65,65,3), is compressed into an INTEGER*2 array, neps(5,65,65), by bit-mapping: the first index of epsmap, range 1-65 is compressed into the first index of neps, range 1-5, where the indices 1-16 go into bits 0-15 of the word with index 1, indices 17-32 -> bits 0-15 of word with index 2 etc. The array neps is then written to an unformatted binary file:

```
write (17) imap, scale, oldmid

write (17) neps
```

where *imap* is an unused *integer*4* flag and scale, oldmid(3) are real*4 scaling information as above.

Note (new!) in the case that the solute is composed of more than one dielectric media, in this release (v.6 up to rel. 1.1) the additional information is not included in the fort.17, in order to maintain compatibility with software packages that take it as an input.

UNIT 19

If the "modified pdb file" option is activated in a WRITE/OUT function, a logical flag (t/f), iatout, will be set to true and will produce a modified PDB file written on unit 19, containing the:
radius and charge assigned to each atom written after the coordinates, in the fields used for occupancy and B factor. It is recommended that this option be set initially so that the user can check that all the radius and charge assignments are correct. An additional check on the charge assignment can be made by looking at the total charge written to the log file.

Return to TOC

6. Other Features In DelPhi v.6

Here below are described some of the new features that have been introduced to DelPhi v.6:

1.   MULTI-DIELECTRIC CODE
2.   GEOMETRIC OBJECTS
3.   GEOMETRIC-SHAPED CHARGE DISTRIBUTIONS
4.   NON-LINEAR PBE SOLVING ROUTINE
5.   MULTI-SALT CODE

1.   MULTI-DIELECTRIC CODE

Earlier versions of DelPhi handle a scenario where a single molecule is immersed in a solution. In other words, only a "two-media" world was considered. In the present version, a system with *many different objects* having different dielectric constant can be modeled. These objects can be either sets of atoms obtained from a pdb file or geometric objects.

Note: In order to gain better accuracy for reaction field energy, the location of polarization charges is normally projected onto the molecular surface. If a molecule is immersed in an object, the molecular surface would be built inside the object. If two molecules with different dielectric constants come in contact or overlap, the molecular surface at their interface is not built. (In fact, it doesn't make sense to do so in that case.) Instead, the polarization charges are not projected anywhere but are left at their own grid point locations.

2   GEOMETRIC OBJECTS

A complete description of a molecule is not always the most convenient way to represent it; this could be because a detailed calculation is too computationally expensive, or because the user prefers a simpler, schematic description of a system for computing semi-quantitative results. Therefore we added to DelPhi a new way to insert geometrical dielectric objects and charge distributions of simple geometrical shapes. The user can now insert various dielectric objects of different types, where a molecule is just one of the possible choices. In order to accomplish this, a set of data structures has been built that contains:

•) an ordinal number, that labels the object.

•) a field that contains the dielectric constant of the medium constituting the object.

•) an integer called "objectype" that tells if the object is a molecule (0), a sphere (1), a cylinder (2), a cone (3) or a box (4).

•) the parameters related to its dimension and position, which are:

for a ***molecule*** : name of the .pdb file,

for a ***sphere*** (a): center C (described by coordinates *x, y, z*) and radius, R. Example: 0.0,0.0,0.0,5 for a sphere centered in the origin and having a radius of 5Å.

for a ***cylinder*** (b): centers of the two bases (A and B) and radius. Example: 0.0,0.0,0.0,0.0,0.0,1.0,10.0 for a cylinder where A=(0,0,0), B=(0,0,1) and R=10Å.

for a ***cone*** (c): center of the base A, vertex B, opening angle (in degrees). Example: 0,0,0,0,0,1,30 for a cone where A=(0,0,0), B=(0,0,1) and angle=30 °.
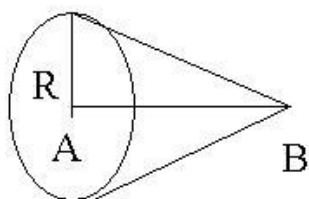
for a ***box*** (d): four adjacent vertices disposed as the origin and the three vectors of a right-handed vector basis (x, y and z, respectively). Example: 0,0,0,1,0,0,0,2,0,0,0,3 for a box where A=(0,0,0),B=(1,0,0),C=(0,2,0) and D=(0,0,3).
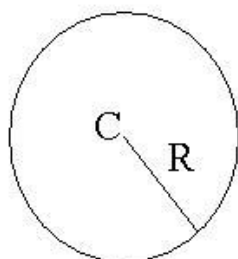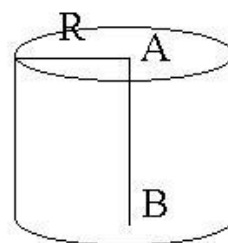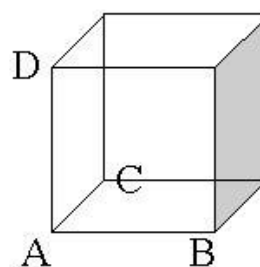
(a)　　　　　　　　(b)

(c)　　　　　　　　(d)

There is also the possibility to create more complex shapes: if two objects turn out to be overlapping, the overlapping region will have the dielectric constant of the object assigned later. In fact, **the dielectric constant of the last assigned object overrides the dielectric constant of all the objects assigned beforehand.** Exploiting this fact, the user can build unions or intersections of the mentioned objects, each of them having the desired dielectric constant, ending up with a more complex scenario.

As a rule, **all molecules superimpose on the other objects regardless of the order in which they have been inserted**.

We are currently testing the possibility of using a different probe radius to build the molecular surface (MS) on the part of any molecule that is buried within an object. This value is assigned through the radpr2=[value] statement. By default, it is set equal to the "regular" probe radius.

At present, boundary conditions cannot be computed for a grid that contains an object that is not completely immersed in the cube. If the user is interested to such a system, he must start with a low percentage filling (e.g. 50% or 60%) where the usual boundary conditions are sufficient. Then he must increase the *perfil* until it gets higher than 100% using the focusing method.

Return to new features

3   GEOMETRIC-SHAPED CHARGE DISTRIBUTIONS

In the new DelPhi v.6 program, various types of uniform charge distributions can be inserted and the user can specify if these belong to an object or not. This is really useful when we are dealing with a charge very close to an interface between different dielectric media and we want to be sure the program assigns the right dielectric to it.

The charge distributions that have been taken into consideration may have the same overall shapes of the objects plus some other additional ones.

In fact, the data structure is built to contain:

- •) an ordinal number, that labels the distribution,

- •) a field that tells if the distribution is within the volume or only onto the surface,

- •) a field that tells if the distribution is "*free*" or belongs to one of the previously inserted objects, if it is free, the medium where it is located is decided based on epsilon map (grid discretized), if it is "linked" to some object, its medium will be the one of the object (see note below). This is useful for charges close to the boundary between different media to avoid erroneous assignments,

- •) a field which contains the total charge amount,

- •) a number called "*distrtype*" that tells if the shape is *spherical (1), cylindrical (2), conic (3), box-shaped (4), point charge (8), linear (9), disk-shaped (10), rectangular (11)*,

- •) the parameters related to its dimension and position, which are the same as corresponding objects if "*distrtype*" is less or equal to 4, and:

  the point (x,y,z) coordinates for **number 8**, Ex.: 1,0,0 for the point (1,0,0)

  the two extremes A and B for **number 9**, Ex.: 0,0,0,1,0,0 for A(0,0,0) and B(1,0,0)

the disk center and three points disposed as the three vectors of a right-handed vector basis (x, y and z, respectively), and the disk radius for **number 10**, Ex.: 0,0,0,1,0,0,0,1,0,0,0,1,5 for a disk lying on the x-y plane, centered in (0,0,0), and having radius R=5Å.

three adjacent vertices disposed as the origin and the two vectors of a right-handed two dimensional vector basis (x and y, respectively), for **number 11**. Ex.: 0,0,0,1,0,0,0,1,0 for a rectangle having the following vertices: A(0,0,0),B(1,0,0),C(0,1,0),D(1,1,0)

Note: the distinction between "free" and "bound" or "linked" charge is a consequence of discretization. Because of it, a charge can be assigned in a location where there is a different medium from the one where the charge was originally set to be. If the user knows which is the medium embedding the charge, then he should link the charge to the correct object, otherwise, the charge will be classified as "free".

[Return to new features](#)

4. <u>NON-LINEAR PBE SOLVING ROUTINE</u>

An algorithm has been implemented that can solve the non-linear PBE, as described in Rocchia et. al. Journal of Physical Chemistry B [paper](#). A good relaxation parameter must be used for fast and reliable convergence of the non-linear equation. A heuristic algorithm is used to estimate it; nonetheless, for some particularly pathological conditions, it is possible that the choice is not optimal and that the convergence is very slow or even absent.

In order to deal also with those systems, the new statement relpar=[value] can be put in the parameter file to manually assign the relaxation parameter.

A **general set of guidelines** to the choice of the **relaxation parameter** follows:

The optimal relaxation parameter for a non-linear system generally turns out to be less than 1. Pathological conditions like a very high charge in proximity to the surface, ionic strength close to zero but not null and some particular scale values, can require a parameter much lower, down to 0.001 or even lower. On the other hand, a too low value has the drawback of slowing down the

convergence and, in the worst case, to give only an illusion of convergence. Thus, the advice is to start with an intermediate value and to decrease it until the convergence is reached.

Convergence hints again:

The hardest part for the convergence process is at the beginning, when the non-linearity of the equation is slowly added. In this phase the relaxation parameter could be quite low, and thus can hinder the continuation of the convergence process. This is the rationale behind the heuristic algorithm.

A good convergence is characterized by a low max change in the potential together with a relaxation parameter not too close to zero, say greater than 0.1 or similar. If the convergence is absent or not satisfying, the user can decide to use a fixed, but suitably tuned, **relpar** which should be as high as possible, provided it allows for convergence.

Two more statements have been added, that are valid for both linear and non linear cases: rmsc=potential root mean square change threshold,[kT/e] and maxc= potential maximum change threshold, [kT/e] that allows two new convergence criteria. The first concerns the rms change and the second the maximum change of grid potential between successive iterations. The second one is of particular interest because it sets an upper bound on the error in potential throughout the grid. Both criteria can be assigned together with the number of iterations, linit or nonit; the program will stop iterating as soon as one of the assigned criteria is satisfied.

Return to new features

5. MULTI-SALT CODE

Including the effects of salts of different valences makes a large difference when calculating the electric field of a solute in solution. Handling multiple valences properly requires using the more general expression for the charge in the original PBE. The ionic strength is then calculated accordingly to its general definition: $I = \frac{1}{2} \sum_i z_i^2 C_i^{bulk}$; the concentration of each ion is related to

the one of the salt assuming electro-neutrality for each salt, that is: $\sum_i z_i C_i^{bulk} = 0$ for the ions of every single salt in solution.

Some new statements have been added to handle this new situation:

salt=[conc first kind of salt, moles/liter], val+1=[valence of positive ion in salt type 1], val-1=[valence of negative ion in salt type 1];

salt2=[conc second kind of salt, moles/liter], val+2=[valence of positive ion in salt type 2], val-2=[valence of negative ion in salt type 2];

Return to new features

6. ENERGY PARTITIONING

A new partitioning of the energy is considered and described in details in the article of the Journal of Physical Chemistry B paper. Accordingly, the electrostatic energy is subdivided into: *coulombic, reaction field, self-reaction field, external ion contribution, osmotic pressure term, "rho*phi" (electrostatic stress)*. The last three terms only appear when the ionic strength is greater than zero. The last two terms cancel out in the linear PB equation.

Coulombic and *reaction field* energies are calculated using Coulomb's law and Gauss's theorem as described in the J. Phys. Chem. paper.

The *osmotic* and *electrostatic stress* terms, defined in the same paper, are calculated whenever the program is asked to run nonlinear iterations; their contribution arises from the solvent *inside* the box.

The calculation of the ***external ion contribution*** is toggled by the flag ion in the energy statement, it calculates the direct interaction of ions to real charges.

This direct calculation needs some comments: it calculates the coulombic interaction between the ions in solution (screened by the surrounding polarized water). First of all, it can be computationally expensive, especially if the grid size is very high. Secondly, this energy neglects all of the ionic contribution due to the ions located outside the box. This can result in a

underestimation of the real contribution, especially if the percentage filling is high. In the linear case this problem has been reduced almost completely by a mixed numerical/analytical technique. The analytical contribution of the first term in a spherical expansion of the molecular system outside the box is calculated. So the direct ionic contribution is better estimated by the sum of two terms: the one internal to the box and the one external to the box.

For the non-linear case this is not so easy, because an analytical solution of the non-linear PBE for the spherical case is not available. The use of the linear approximation routine would result in an overestimation of the ionic contribution.

Another way to get an estimation of the same energy term is to subtract the grid and reaction field energy in two cases: with and without salt. This means running the program twice on the system.

In order to help the user decide if these two runs are necessary, the program estimates and reports the number of Debye lengths bounded within the grid. This number, together with the ionic strength, aids the user in estimating the error due to neglecting the ionic contribution outside the box.

Note: If periodic boundary conditions are used, the estimated external ion contribution will be taken only in the directions where periodicity is NOT invoked. This is done in order to give an estimate of energy *per periodic cell*.

The **self-reaction field energy** is calculated whenever the usual reaction field energy is calculated. It is an attempt to estimate the energy that the system gains when a point charge is moved from vacuum to a dielectric medium. In fact, when a charge moves from vacuum to a dielectric a polarization charge of opposite sign builds up around it. This results in an energetically favorable process. The higher the dielectric constant of the medium, the higher is the amount of polarization charge and the more energetically favorable is the process. In our model, it is assumed that polarization charge builds up spherically around any real point charge, at a distance the user can decide via the statement radpolext=[radius], which by default is set to be 1Å. This information is no attempt to give a quantitative estimation of this energy but just to take into account in a more detailed way the fact that the most favorable place for a charge is within a high dielectric medium. This necessity arises when a system with many different

dielectric regions is studied.<u>SELECTING OUTPUT FORMAT USING "ideveloper" OPTION (DelPhi V.6 only)</u>

7. <u>SELECTING OUTPUT FORMAT USING "ideveloper" OPTION (DelPhi v.6 only)</u>

Users now are able to select the output format either in single or in double precision by flipping over the logical switch "ideveloper" in subroutine defprm. When ideveloper is set to be .true., the output values have more decimals printed out, which would be useful to have a closer insight of the changes of certain values.

<u>Return to new features</u>

<u>Return to TOC</u>

Sample parameter file

!example prm file for DelPhi v.6
scale=1.5
perfil=80
in(pdb,file="ala.pdb")
bndcon=4
indi=2
exdi=80
ionrad=2.0
prbrad=1.4
salt=0.001
!out(frc)
!out(modpdb)
!out(phi)
logfileconvergence=true

logfilepotentials=true

energy(s,g,c,ion)