APBS*: Implicit Solvation Calculations within the CMDF Framework

Frank Ducheneaux, Jef Dodson, Markus Buehler, and William A. Goddard

Materials and Process Simulation Center, California Institute of Technology

August 23rd. 2005

*Baker, Nathan et al.

Implicit Solvation Methods

- Corresponds to solvation in an infinite volume of solvent
- Describe instantaneous solvent dielectric response
- Molecule can explore the available conformational space much faster
- Estimating energies of solvated structures is much more straightforward than with explicit water models

Background

- APBS: Adaptive Poisson-Boltzmann Solver
- Adaptive, multilevel, finite element method for solving the Poisson-Boltzmann equation
- Solution of the PBE:

$$-\nabla\cdot(\varepsilon(x)\nabla u(x))+\bar{k}^2(x)\sinh(u(x))=\frac{4\pi e_c^2}{k_B T}\sum_{i=1}^{Nm} z_i\delta(x-\bar{x}_i)$$

  - Electrostatic free energy of solvation (ΔG_{el})
  - Solute/Solvent interaction forces
- Literature* suggests a factor of 2 improvement in computation time over popular finite difference methods (e.g. Delphi software of Honig, et. al.)
  - We are currently investigating this claim

*Holst, M. et al, *J. Comp. Chem*, 21 (2000)

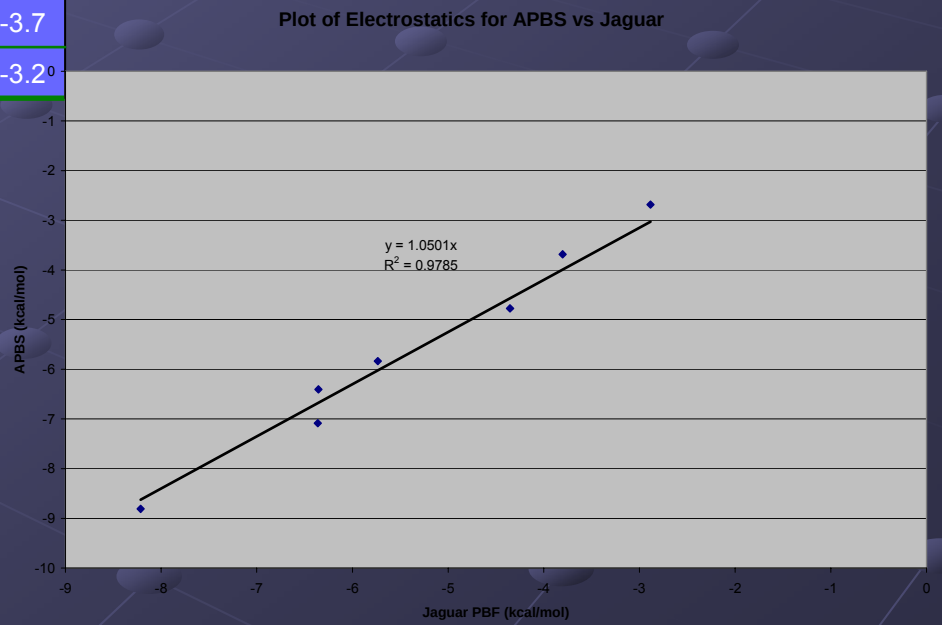
Background (contd)

- $\Delta G_{total} = \Delta G_{el} + \Delta G_{cavity}$
- APBS determines only ΔG_{el} implicitly
  - Solvent represented as a dielectric continuum (as opposed to the explicit representation as individual molecules)
  - Solute structure (protein, lipid membrane, etc.) represented implicitly via a dielectric constant and explicitly as a set of point charges
  - Continuum representation decreases computational time
- ΔG_{cavity} includes solute/solvent van der Waals interactions (both repulsive and dispersive) and the entropic penalty associated with reorganizing solvent around the solute

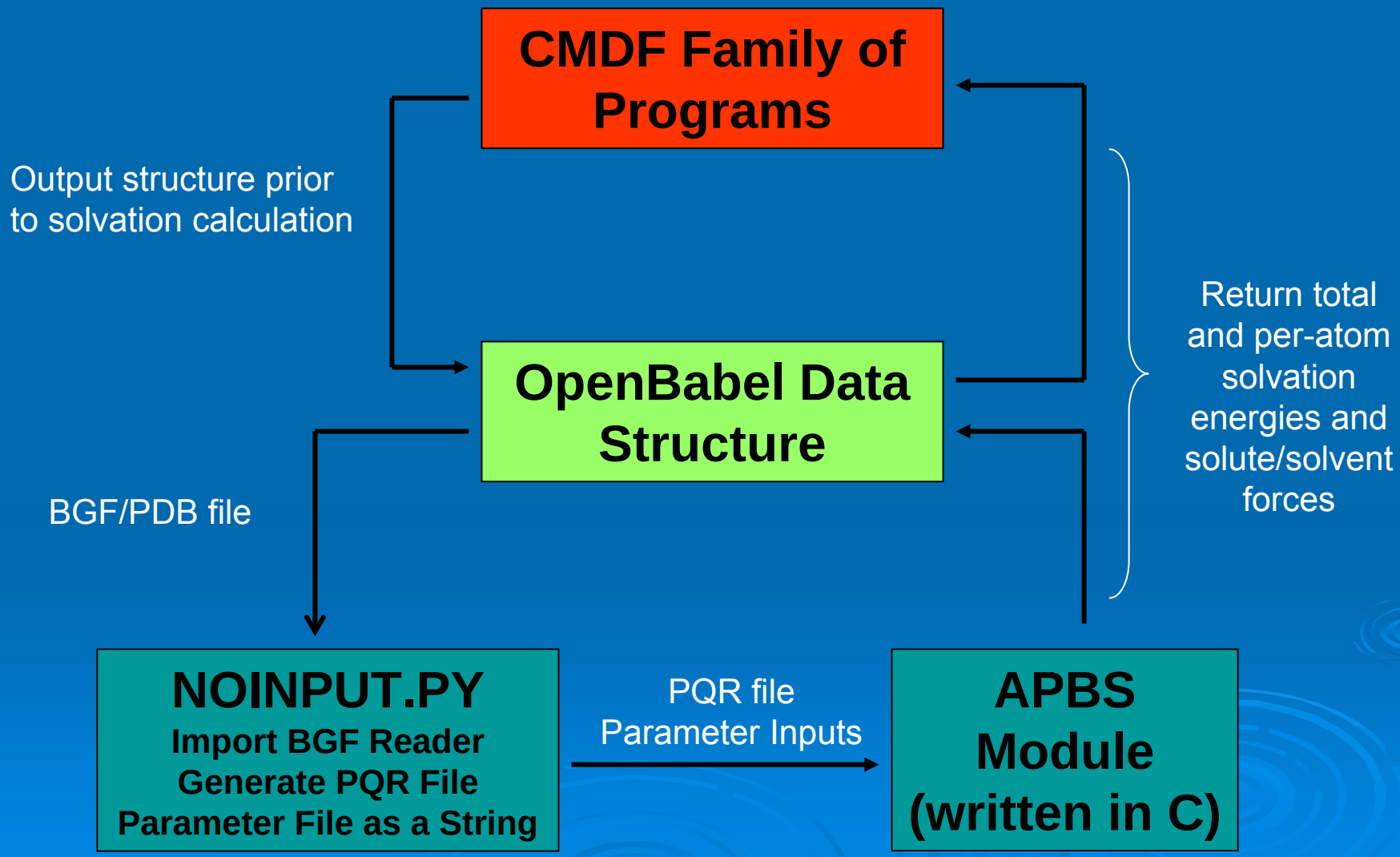
APBS Comparisons

	APBS	Jaguar	Deviation	Experimental
Acetic Acid	-7.1	-7.1	0	-6.7
Acetone	-4.9	-4.4	-0.5	-3.9
Benzaldehyde	-4.11	-4.1	-0.01	-4
Benzene	-0.07	-0.6	0.53	-0.9
Dimethylamine	-3.58	-4.4	0.82	-4.3
Ethane	1.97	1.8	0.17	1.8
Ethanol	-5.02	-5.1	0.08	-5
Ethyne	-0.65	-0.2	-0.45	0
Hexane	2.47	2.7	-0.23	2.6
Methylamine	-4.08	-4.6	0.52	-4.5
Nitroethane	-3.51	-3.8	0.29	-3.7
Trimethylamine	-3.02	-3.2	0.18	-3.2

- Compare APBS with Self Consistent Reaction Field (SCRF)
- APBS x20 times faster



General APBS Role in CMDF



Sample Standalone Parameter File

(as used in Baker's APBS version)

```
read
    mol pqr molecule.pqr      #read in structure
end

# SOLVATED STATE
elec name solvated_state
    mg-manual
    dime 17 17 17             #grid dimensions
    nlev 3                    #grid levels
    #grid 0.01 0.01 0.01      #grid spacings
    glen 10 10 10             #grid lengths
    gcent mol 1
    mol 1
    lpbe
    bcfl mdh                  #boundary conditions
    ion 1 0.000 2.0           #chg./conc./radius
    ion -1 0.000 2.0
    pdie 1.0                  #solute dielectric
    sdie 80.0                 #solvent dielectric
    chgm spl2
    srfm smol
    srad 1.4                  #solvent probe radius
    swin 0.3
    temp 298.15
    gamma 0.105
    calcenergy total
    calcforce no
end
```

```
# REFERENCE STATE
elec name reference_state
    mg-manual
    dime 17 17 17
    nlev 3
    #grid 0.01 0.01 0.01
    glen 10 10 10
    gcent mol 1
    mol 1
    lpbe
    bcfl mdh
    ion 1 0.000 2.0
    ion -1 0.000 2.0
    pdie 1.0
    sdie 1.0
    chgm spl2
    srfm smol
    srad 1.4
    swin 0.3
    temp 298.15
    gamma 0.105
    calcenergy comps
    calcforce comps
end

# solvation energy
print energy solvated_state - reference_state end
quit
```

Sample User Interface: ex_modbabel.py CMDF Script

```
#!/exec/python/bin/python

from noinput import *  #python wrapper

import bgf_reader

def main():
    OBtot=bgf_reader.parse_file("molecule.bgf") ## this is the complete sample

    (E, F) = RunAPBS_OBMol (OBtot, 0, [0.25, 0.25, 0.25], 1, 1);


if __name__ == "__main__": main()
```

- "molecule.bgf" obtained from the OpenBabel Data Structure
- RunAPBS_OBMol points to the function described in noinput.py module – the python wrapper for APBS

Noinput.py – Python Interface to C-coded APBS

```
def RunAPBS_OBMol ( OBMol, use_glen, grid, calcenergy, calcforce,
    mgtype="mg-manual", dime=[65,65,65], nlev=4, glen=[100,100,100],
    gcent=1, mol=1, bcfl="mdh",  ion_pos=[1.00,0.000,2.0],
    ion_neg=[-1,0.000,2.0], pdie=1.0, sdie=78.54, sdie_g=1.0,
    chgm="spl2", srfm="smol", srad=1.4, swin=0.3, temp=298.15, gamma=0.105,
    rflag="DREIDING_RADII" ):
```

- Currently, user must define input parameters here AND in the ex_modbabel.py
- DREIDING_RADII assigned to OpenBabel data structure within the noinput script
- Other force field radii (e.g. CHARMM) will be added

Noinput.py (continued)

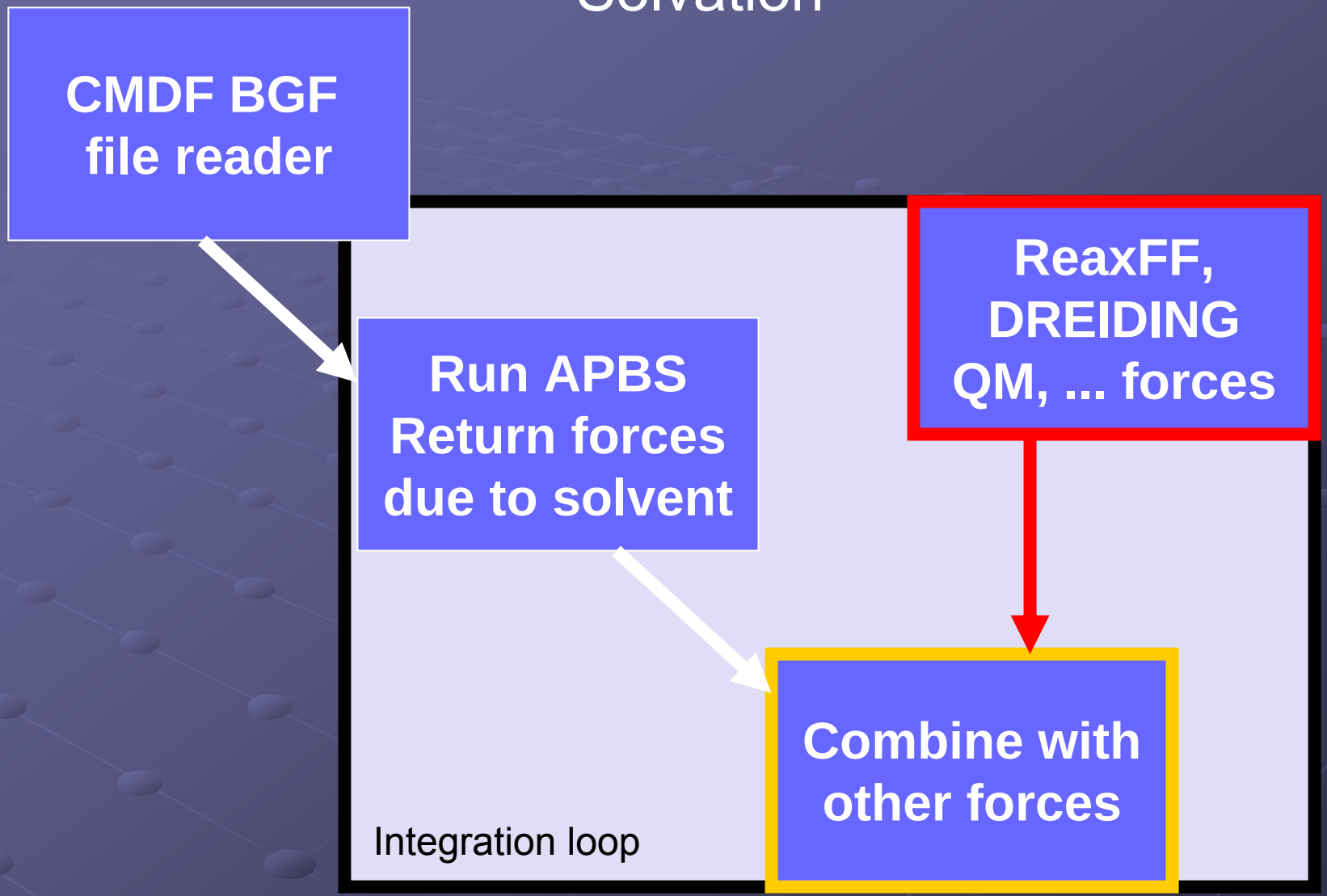
```
PQR = ""

   istring = ""
   istring = istring + "# SOLVATED STATE\n"
   istring = istring + "elec name solvated_state\n"
   istring = istring + "    " + mgtype + "\n"
   istring = istring + "    dime %10d %10d %10d" % (dime[0],dime[1],dime[2])
   istring = istring + "\n"
       .            .            .
       .            .            .
       .            .            .
    istring = istring + "quit“

   PQR = GetPQRInfoModBabel(OBMol, rflag)
```

- Variables defined in ex_modbabel and noinput modules combined with “istring” mimic the standalone input file in previous slide
- PQR file = BGF file (atomic info, x,y,z, atomic charges) + DREIDING radii

Applications and Impact of the CMDF APBS Module

- The CMDF APBS module enables us to model implicit solvents based on continuum theory within discrete atomistic systems
- This hybrid multi-scale approach represents enormous computational efficiency since explicit solvent molecules (e.g. water molecules) can be quite expensive for large volumes
- The CMDF APBS module can be combined with different force fields, including
  - DREIDING (e.g. in structural predictions of proteins in membranes)
  - ReaxFF (e.g. In modeling enzymes)
  - Mesoscale models (e.g. Tod Pascal‘s meso-DNA model, or Valeria Molinero‘s glucose meso-models)
- The CMDF APBS module allows for the development and training of new force fields and new modeling concepts based on hybrid modeling (e.g. spatially varying implicit solvent parameters at interfaces of membranes)

Overall Flow of Dynamics Program with Implicit Solvation



REAXTOOLS.ReaxForces (OB1, parms)
RunAPBS_OBMol (OB1, 0, [0.25, 0.25, 0.25], 1, 1, “add”)

Current Status/Upcoming Work

- APBS successfully implemented into CMDF framework
- Verified Python wrappers: CMDF version returns identical energy/force output values as standalone version (given identical sig. figs. for structural values, e.g. radii)
- Next step:
  - Further coding to return APBS output to the OpenBabel data structure for use by other programs
  - Improved user interface
  - Realiable method for including ΔG_{cavity}
- The next next step:
  - MD simulations/VAC analysis → determine rules for a location-dependent ε
  - Multiple ε representation in APBS (solvent, solute, lipid membrane)

Acknowledgements

- Tod Pascal (Caltech): for his help with the python coding and for doing my presentation while I am out of town
- Jef Dodson and Markus Buehler (Caltech): for assistance with “pythonizing” APBS and for general guidance in this project
- Prof. Nathan Baker (U. Wash.) for his authorship of the APBS code and to both Prof. Baker and his student, Todd Dolinsky, for their help in working out the bugs with the python coding