

# ZAP 2.1

Ben Ellingson  
CUP IX



OpenEye Scientific Software

$$\nabla \cdot \epsilon_0 \epsilon(\mathbf{r}) \nabla \phi(\mathbf{r}) = -\rho(\mathbf{r})$$

“Let us bask in the glory that is the Poisson Boltzmann equation” – Mark McGann



# Outline

- What can ZAP do for you?
- Searching for best practices
- Doing cool things with ZAP



# What ZAP can do

- ZAP is a fast, grid-based PB solver

- Calculates the electrostatic potential grid

$$\nabla \cdot \epsilon_0 \epsilon(\mathbf{r}) \nabla \phi(\mathbf{r}) = -\rho(\mathbf{r})$$

- Calculates surface area

- From this, quantities like vacuum-water transfer energies can be calculated

- The electrostatic energy is equal to:  $\sum q_i \phi_i$
- Transfer energies can be determined by calculating the universal  $\epsilon$  grid and a 80:1 grid.
- The electrostatic transfer energy is the difference in electrostatic energy.



# What ZAP can do (cont)

- How to do the same thing

```
OEZap zap;  
zap.SetMolecule(mol);  
float energy = zap.CalcSolvationEnergy();
```

- Useful API

- Solvation energy, potential grids: OEZap
- Accessible surface area: OEArea
- Binding properties: OEBind
- Electrostatic similarity: OEET



# Searching for Best Practices

- What are the best settings for ZAP?
  - Dielectric
  - Partial charges
  - Grid spacing



# Methodology

- Rizzo set
  - 201 small molecules with accurate hydration energies
- Ran through Omega2 with maxconfs=1
- Assigned Gasteiger, MMFF94, AM1, and AM1BCC charges
- Calculated hydration energies including a surface area term (10 cal/Å<sup>2</sup>) with grid spacings of 0.05Å to 1.0Å.



# Best dielectric and charges

- RMSE (kcal/mol) of hydration energy calculations vs experimental hydration energies
- Slightly modified version of transfer.cpp
  - Read in charges
  - Dielectric of 1.0 and 2.0

	Dielectric=1	Dielectric=2
AM1BCC	1.44	2.85
MMFF94	2.66	2.37
AM1	2.88	4.04
Gasteiger	3.53	4.47



# Default dielectric and charges

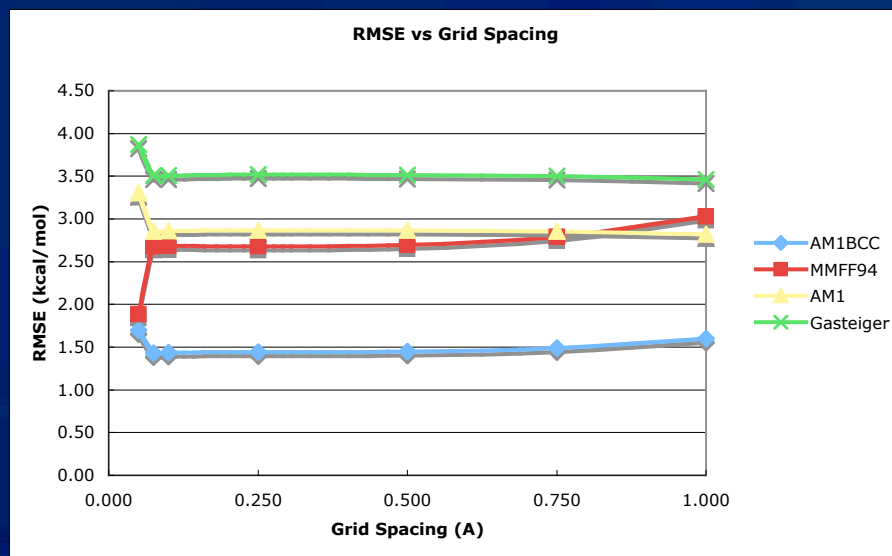
- Default inner dielectric has been changed from 2.0 to 1.0 in ZAP 2.1.
  - Can be set: `zap.SetInnerDielectric(float d);`
- Default charges are MMFF94.
  - MMFF94 charges are included in OEChem
  - AM1BCC charges are recommended and can be calculated with QuacPAC (molcharge)



# Optimal Grid Spacing

- Smaller than 0.05 requires too much memory
- Small grid spacings are silly
- The default value is 0.5
- May be set: `zap.SetGridSpacing(float spacing)`

spacing (A)	AM1BCC	MMFF94	AM1	Gasteiger	Time (s)
0.050	1.70	1.89	3.31	3.87	4109.00
0.075	1.43	2.68	2.85	3.51	1200.00
0.100	1.43	2.68	2.86	3.51	513.00
0.250	1.44	2.68	2.87	3.52	24.50
0.500	1.45	2.69	2.87	3.51	2.50
0.750	1.49	2.79	2.85	3.50	0.90
1.000	1.60	3.03	2.82	3.46	0.50



# Doing cool things with ZAP

- Calculating the induced charge outside the molecule and visualizing it

```
//OEGraphMol mol has 3D coords and charges  
OEAssignBondiVdWRadii(mol);  
OEZap zap;  
zap.SetOuterDielectric(1.0f); //default  
zap.SetInnerDielectric(80.0f); //default  
zap.SetMolecule(mol);  
zap.SetGridSpacing(0.25f); //for prettiness
```



# Calculate the potential grids

```
OEScalarGrid grid1, grid2;  
zap.CalcPotentialGrid(grid1);  
zap.SetOuterDielectric(innerrep);  
//Now calculate vacuum grid  
zap.CalcPotentialGrid(grid2);  
//The induced potential is the difference  
OESubtractGrid(grid1,grid2);  
//This subtracts grid2 from grid1 and  
//stores it in grid1
```



# Calculate Numerical Derivatives

$$\nabla \cdot \epsilon_0 \epsilon(\mathbf{r}) \nabla \phi(\mathbf{r}) = -\rho(\mathbf{r})$$

- We are solving for  $\rho(\mathbf{r})$
- We start by calculating  $\nabla \phi(\mathbf{r})$  (twice)

```
for (unsigned int k = mink; k < maxk; k++)  
  for (unsigned int j = minj; j < maxj; j++)  
    for (unsigned int i = mini; i < maxi; i++)  
      {  
        delphii1 = (grid1(i+1, j, k)-grid1(i, j, k))/spacing;  
        delphij1 = (grid1(i, j+1, k)-grid1(i, j, k))/spacing;  
        delphik1 = (grid1(i, j, k+1)-grid1(i, j, k))/spacing;  
        delphii2 = (grid1(i, j, k)-grid1(i-1, j, k))/spacing;  
        delphij2 = (grid1(i, j, k)-grid1(i, j-1, k))/spacing;  
        delphik2 = (grid1(i, j, k)-grid1(i, j, k-1))/spacing;
```



# Calculate Numerical Derivatives

$$\nabla\phi(\mathbf{r}) = E(\mathbf{r}), \varepsilon(\mathbf{r}) = 1$$
$$\varepsilon_0 \nabla \cdot E(\mathbf{r}) = -\rho(\mathbf{r})$$

- Now we calculate  $\nabla \cdot E(\mathbf{r})$  at each grid point

```
delEi = (delphi1-delfii2)/spacing;  
delEj = (delphi1-delfij2)/spacing;  
delEk = (delphi1-delfik2)/spacing;  
deldotE = delEi+delEj+delEk;
```



# Solve for Charge Density

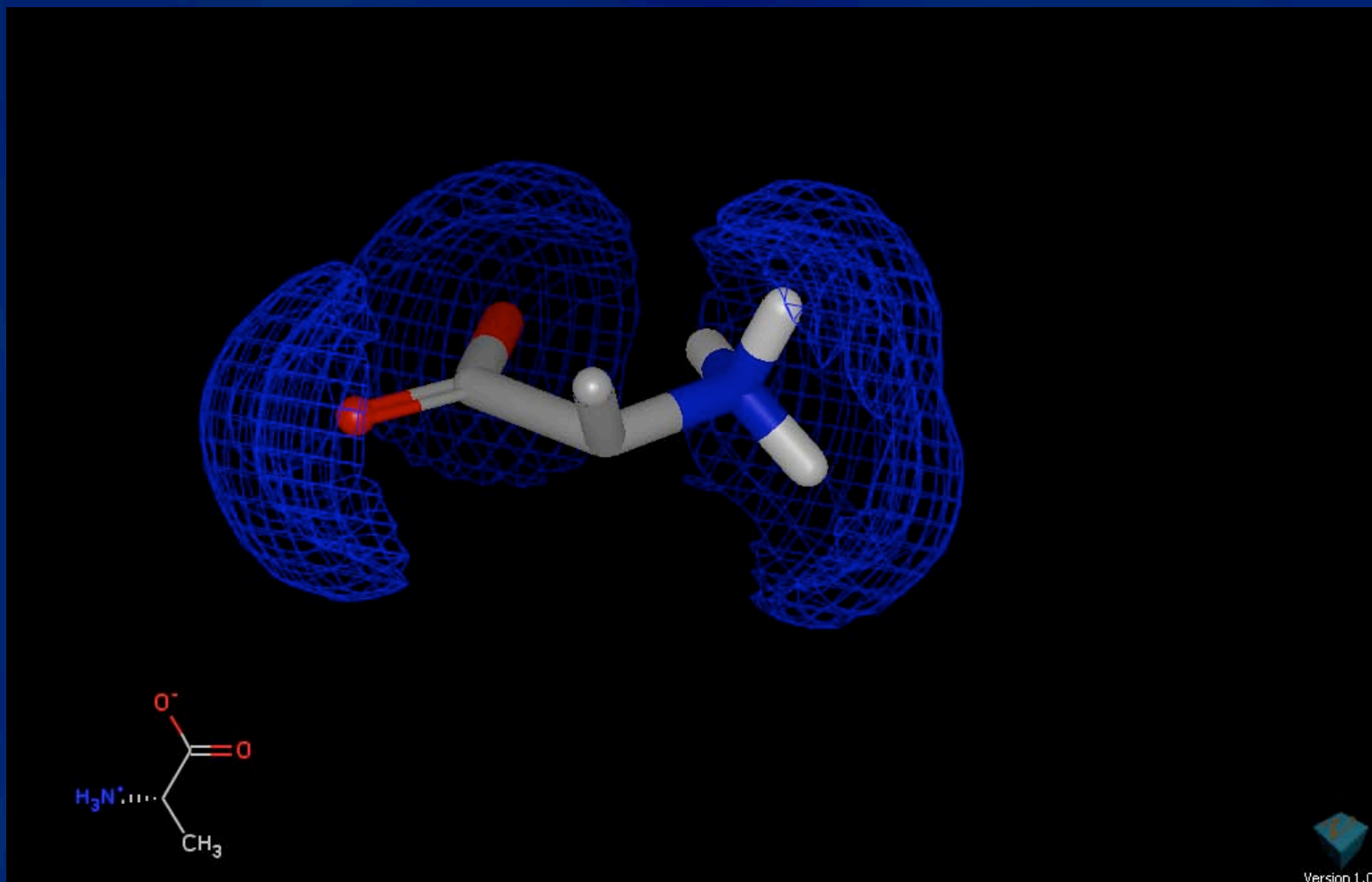
$$\rho(\mathbf{r}) = -\epsilon_0 \nabla \cdot \mathbf{E}(\mathbf{r})$$

- ZAP uses units of Angstrom, elementary charge, and kT
- $\epsilon_0$  is 0.000141916 in units of  $\frac{e^2}{kT \cdot A}$
- Write charge density and solvation energy to a grid

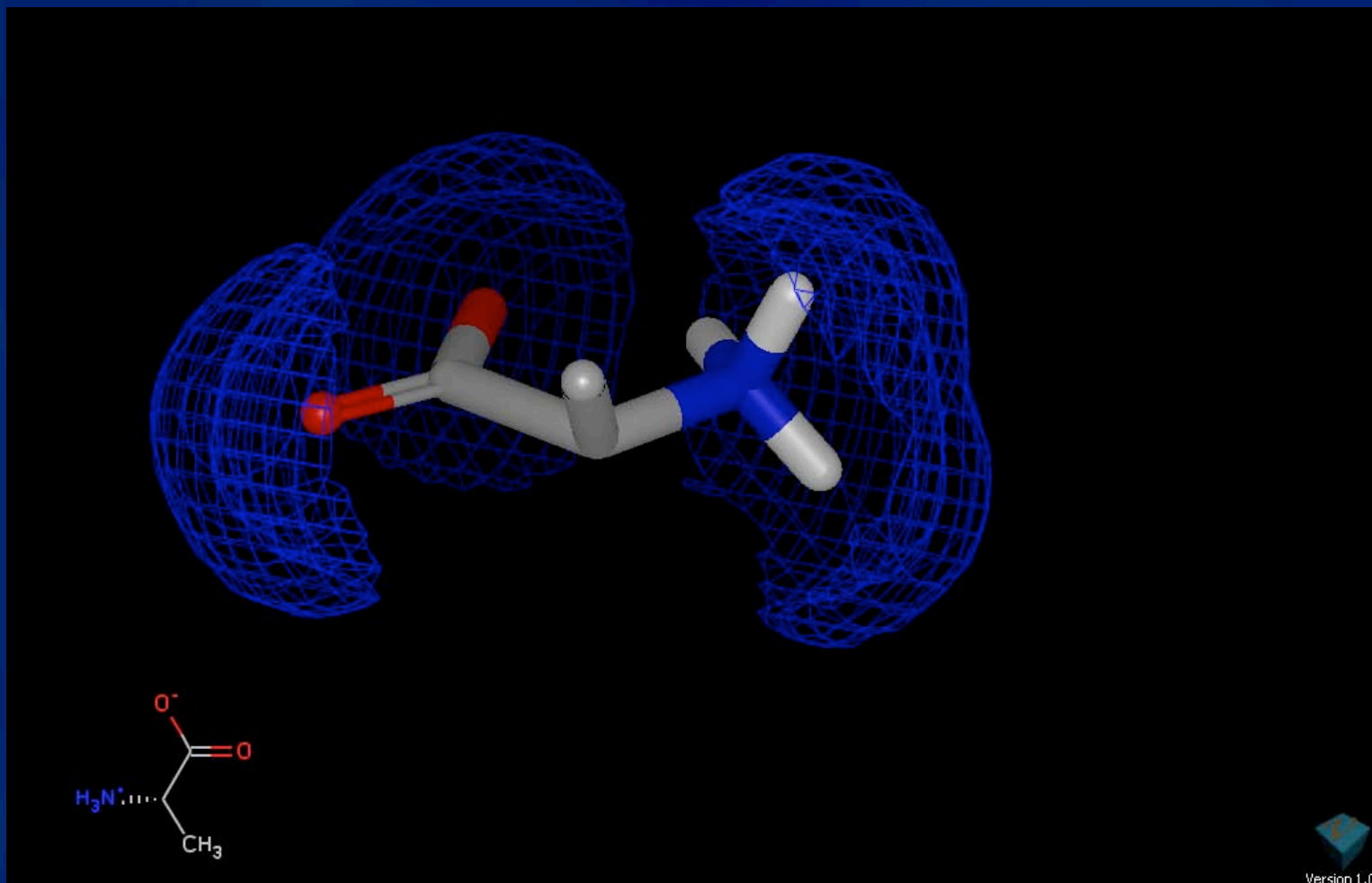
```
chargedensity = -1.0f * 0.000141916 * deldotE;  
ICgrid(i,j,k)=chargedensity;  
SEgrid(i,j,k)=chargedensity * grid1(i, j, k);  
} //closes triple loop
```



# Contour of Solvation Energy Density (2.5 kcal/mol per cubic Angstrom)



# Contour of Solvation Energy Density (2.5 kcal/mol per cubic Angstrom)



Version 1.0



# Conclusions

- The ZAP 2.1 API is user friendly
  - Error checking, proper setup
  - All API points have documentation
- The defaults have been updated to improve results
- ZAP toolkit has the flexibility to do cool things

