Max Webinar

BigDFT

The operators of BigDFT code. From convolutions to Poisson Solver in High Performance Computing

Luigi Genovese

Laboratoire de Simulation Atomistique - L_Sim

November 12, 2020

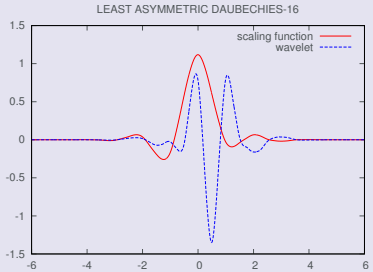
Virtual Room

Wavelet families used in BigDFT code

Daubechies $f(x) = \sum_\mu c_\mu \phi_\mu(x)$

Orthogonal set

$$c_\mu = \int \mathrm{d}x\, \phi_\mu(x) f(x) \equiv \langle \phi_\mu | f \rangle$$

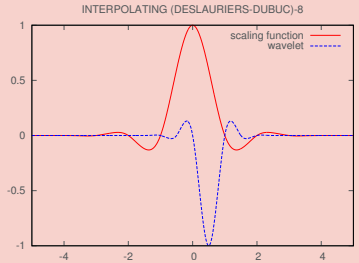


No need to calculate overlap matrix of basis functions
Used for wavefunctions, scalar products

Interpolating $f(x) = \sum_j f_j \varphi_j(x)$

Dual to dirac deltas (. . . is it?)

$$f_j = f(j)$$



The expansion coefficients are the point values on a grid
Used for charge density, function products

Magic Filter method (A.Neelov, S. Goedecker)

The passage between the two basis sets can be performed without losing accuracy $c_\mu = \sum_k w_{\mu-k} f_k$, $w_k = \langle \phi | L_k \rangle$.

Operations performed

The SCF cycle

Orbital scheme:

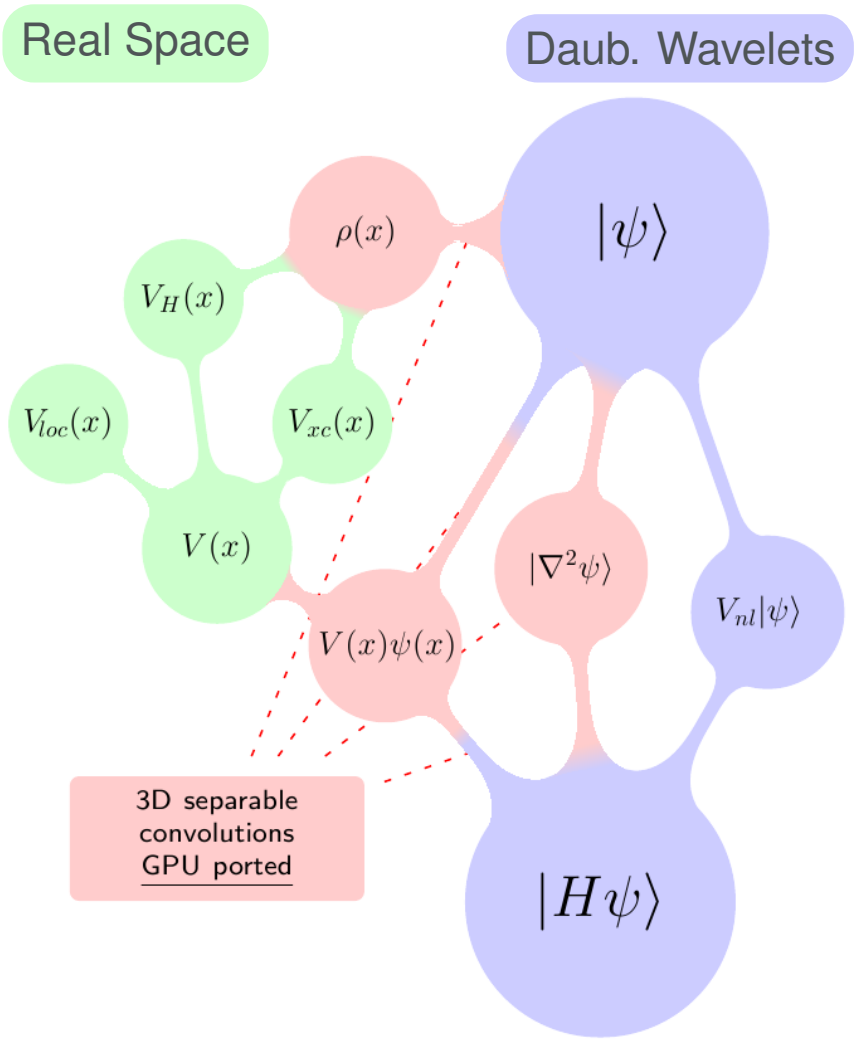
- Hamiltonian
- Preconditioner

Coefficient Scheme:

- Overlap matrices
- Orthogonalisation

Comput. operations

- Convolutions
- `BLAS` routines
- FFT (Poisson Solver)



Why not GPUs?

HPC approach of BigDFT

A DFT code conceived with a HPC mindset

- DFT calculations up to many thousands atoms

- ☛ An award-winning HPC code
- BigDFT has been conceived for massively parallel etherogeneous architectures since more than 10 years (MPI + OpenMP + GPU)

Code able to run routinely on different architectures

- GPGPU since the advent of double-precision (2009)
- Itanium, BG-P and BG-Q (Incite award 2015)
- ARM architectures (Mont Blanc I)
- KNL Accelerators (Marconi)
- K computer (RIKEN) Fugaku – preparatory
- European Supercomputers (Archer, IRENE-Rome, . . .)

Using GPUs in a given (DFT) code

Developer and user dilemmas

- Does my code fit well? For which systems?
- How much does porting costs?
- Should I always use GPUs?
- How can I interpret results?

Evaluating GPU convenience

Three levels of evaluation

1. Bare speedups: GPU kernels *vs.* CPU routines
   Are the operations suitable for GPU?
2. Full code speedup on one process
   Amdahl's law: are there hot-spot operations?
3. Speedup in a (massively?) parallel environment
   The MPI layer adds an extra level of complexity

Case study: 1D convolutions (BigDFT code)

Initially, naive routines (FORTRAN?)

$$y(j,I) = \sum_{\ell=L}^{U} h_\ell x(I+\ell, j)$$

- Easy to write and debug
- Define reference results

```
do j=1,ndat
  do i=0,n1
    tt=0.d0
    do l=lowfil,lupfil
tt=tt+x(i+l,j)*h(l)
    enddo
    y(j,i)=tt
  enddo
enddo
```

Optimisation can then start (2010: X5550,2.67 GHz)

Method	GFlop/s	% of peak	SpeedUp
Naive (FORTRAN)	0.54	5.1	1/(6.25)
Current (FORTRAN)	3.3	31	1
Best (C, SSE)	7.8	73	2.3
OpenCL (Fermi)	97	20	29 (12.4)

How to optimize?

A trade-off between benefit and effort

FORTRAN based

- ✔ Relatively accessible (loop unrolling)
- ✔ Moderate optimisation can be achieved relatively fast
- ✖ Compilers fail to use vector engine efficiently

Push optimisation at the best

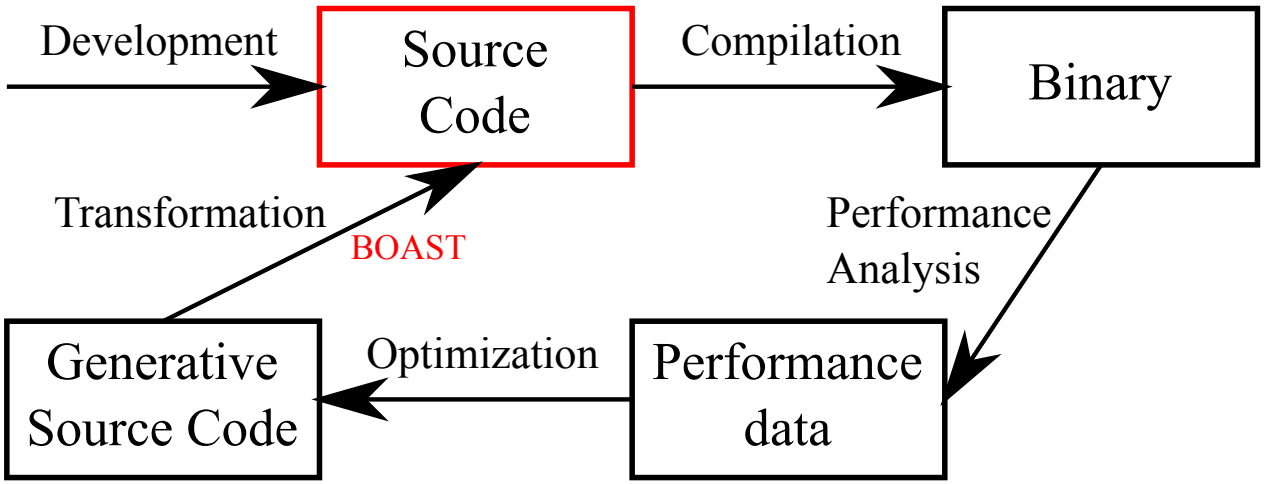
- Only one out of 3 convolution type has been dissected
- About 20 different patterns have been studied for one 1D convolution
- Tedious work, huge code $\longrightarrow$ Maintainability?

☛ Automatic code generation with BOAST engine

BOAST Workflow

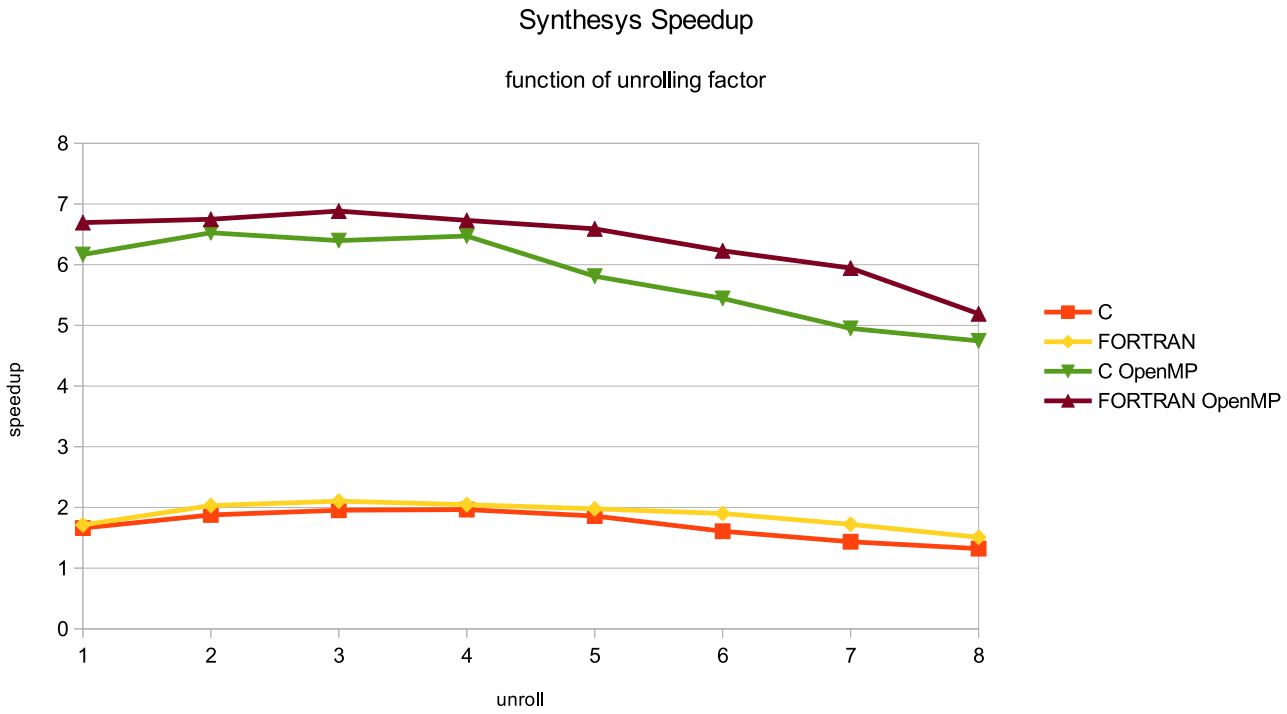
Metaprogramming the kernels

Optimization hand-in-hand with *design* of the operations



- Generate combination of optimizations
- C, OpenCL, FORTRAN and CUDA are supported
- Compilation and analysis are automated
- Selection of best version can also be automated

Example with a kernel (Wavelet Synthesis)



Take-home messages of the BOAST strategy

- Meta-programming of Hot-spot operations
- Optimization can be adapted to the characteristics
- Portability, best effort and maintainance may go together

☛ Towards a BLAS-like library for wavelets convolutions

Autotuning of Libraries (core level)

Not all codes can benefit from BLAS/LAPACK/FFT

- Domain specific libraries need to be optimized for the architecture
- Autotuning is possible through the usage of tools like BOAST
- Generality can be achieved throught meta-programming

Example: libconv

- BOAST can generate all wavelet families $\Rightarrow$ the library has more functionalities than the original code
- BOAST can adapt the routines to the architecture using OpenMP and vector instructions
- BOAST can target FORTRAN and C with OpenMP, CUDA, OpenCL

$O(N)$ BigDFT code is completely different

- Less data $\rightarrow$ lower number of flops per atom
- Communication scheduling completely different

Different problems ☛ new opportunities

- No hot spot operations anymore!
  Convolutions are a less important percentage of the run
- Linear algebra becomes sparse
  Extra completity in handling parallelisation

The time-to-solution is considerably improved

18 k Atoms $\rightarrow$ less than 20 minutes on 13k cores

- The computational physicist is happier
- What about the computer scientist?

Interpolating SF Poisson Solver

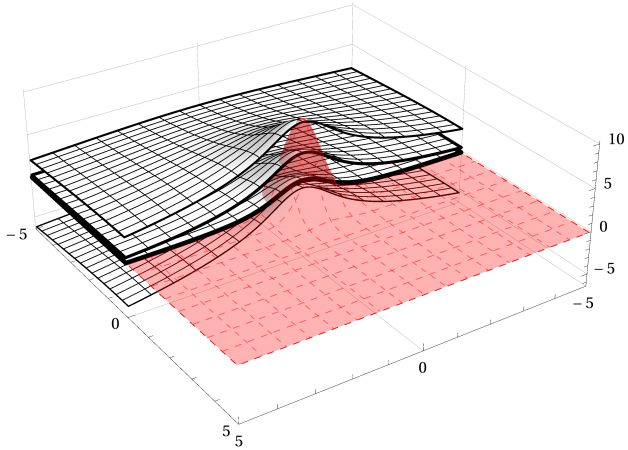
(Screened) Poisson Equation for any BC in vacuum

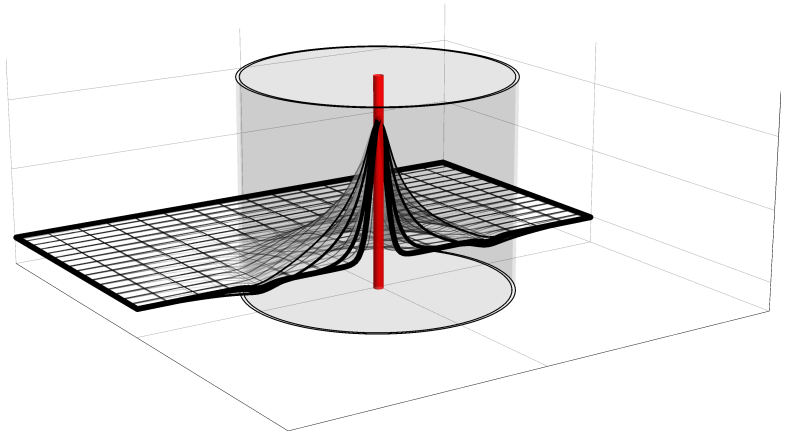
Non-orthorhombic cells (periodic, surface BC):

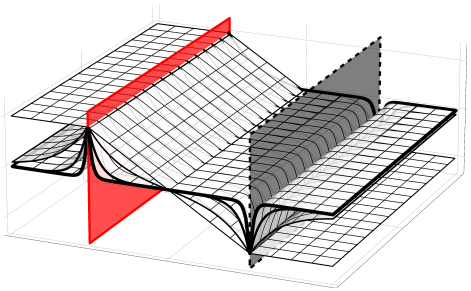
$$(\nabla^2 - \mu_0^2) V(x,y,z) = -4\pi \rho(x,y,z)$$

Machine-precision accuracy J. Chem. Phys. **137**, 13 (2012)

Extended to implicit solvents (JCP 144, 014103 (2016))







Future developments

Range-separated Coulomb operator

$$\frac{1}{r}\left[\operatorname{erf}\frac{r}{r_0} + \operatorname{erfc}\frac{r}{r_0}\right]$$

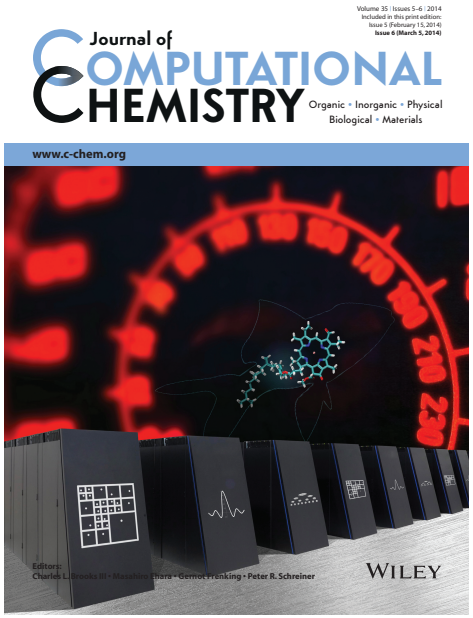
Precision and performances (JCC, 2014)

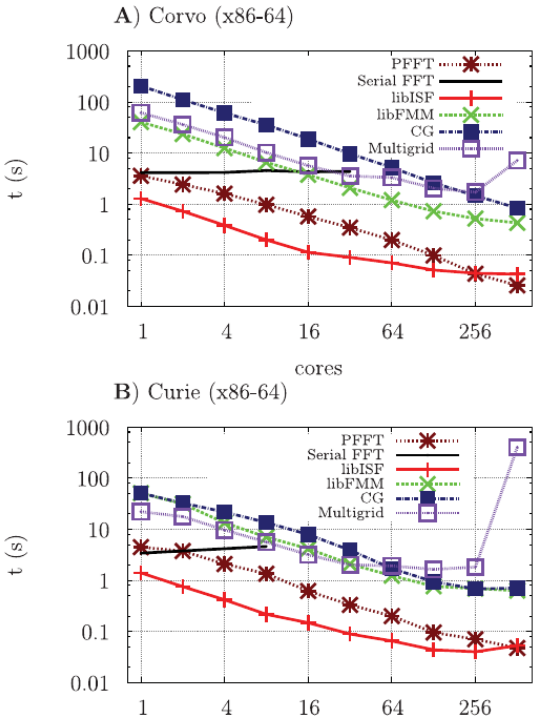
A Survey of the Parallel Performance and Accuracy of Poisson Solvers for Electronic Structure Calculations

Pablo García-Risueño,*[a,b,c] Joseba Alberdi-Rodriguez,[d,e] Micael J. T. Oliveira,[f] Xavier Andrade,[g] Michael Pippig,[h] Javier Muguerza,[d] Agustin Arruabarrena,[d] and Angel Rubio[e,i]

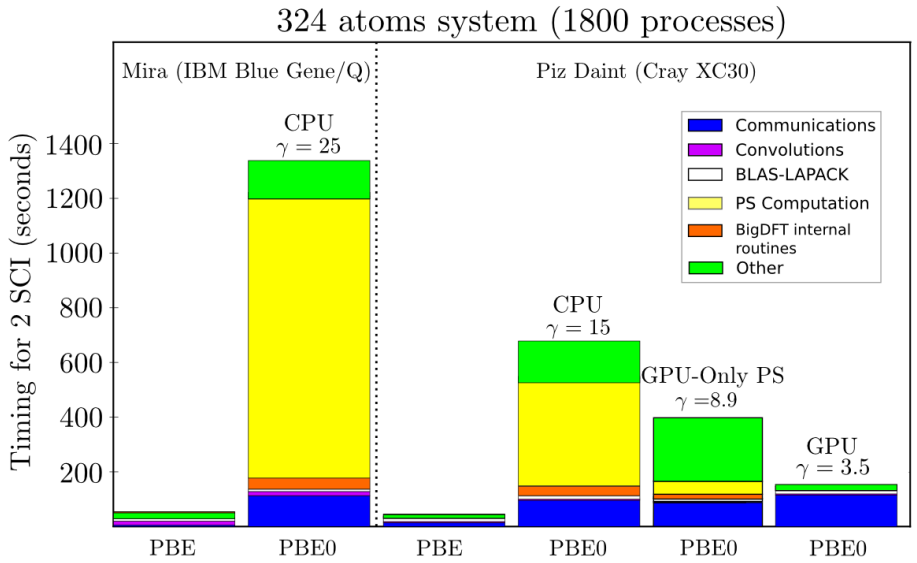
We present an analysis of different methods to calculate the classical electrostatic Hartree potential created by charge distributions. Our goal is to provide the reader with an estimation on the performance—in terms of both numerical complexity and accuracy—of popular Poisson solvers, and to give an intuitive idea on the way these solvers operate. Highly parallelizable routines have been implemented in a first-principle simulation code (OCTOPUS) to be used in our tests, so that reliable conclusions about the capability of methods to tackle large systems in cluster computing can be obtained from our work. © 2013 Wiley Periodicals, Inc.

DOI: 10.1002/jcc.23487

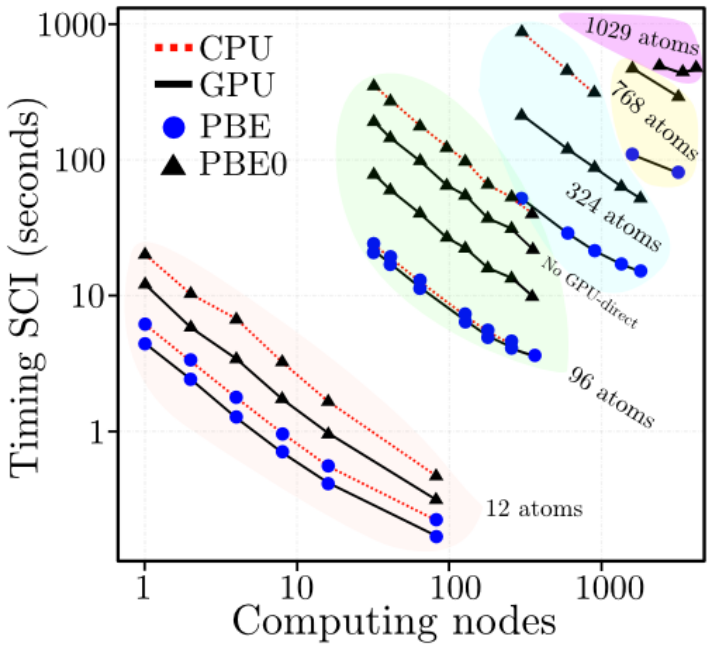


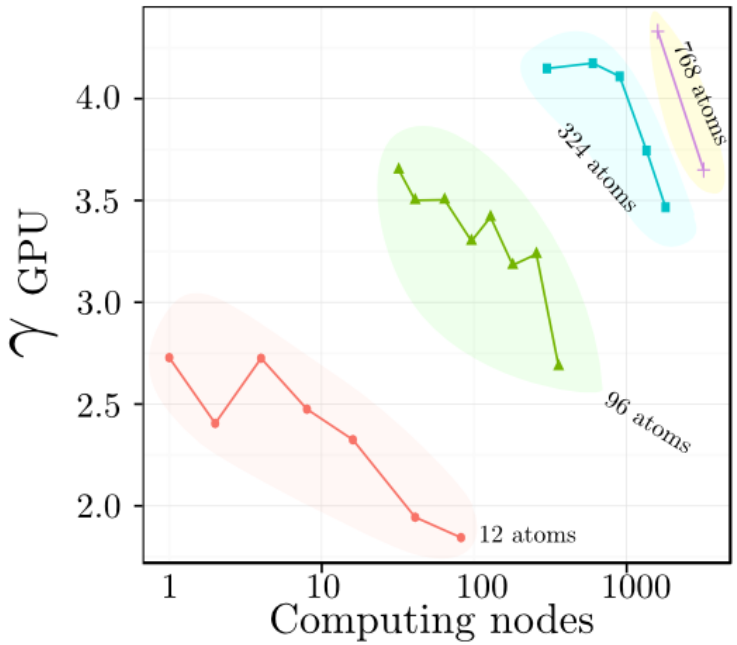


Hybrid Functionals (JPCM 30 (9),095901 (2018))



UO_2 systems:	
Atoms	Orbitals
12	200
96	1432
324	5400
768	12800
1029	17150





Polarizable Continuum Models

Poisson solver for implicit solvents JCP 144, 014103 (2016)

Allows an efficient and accurate treatment of implicit solvents

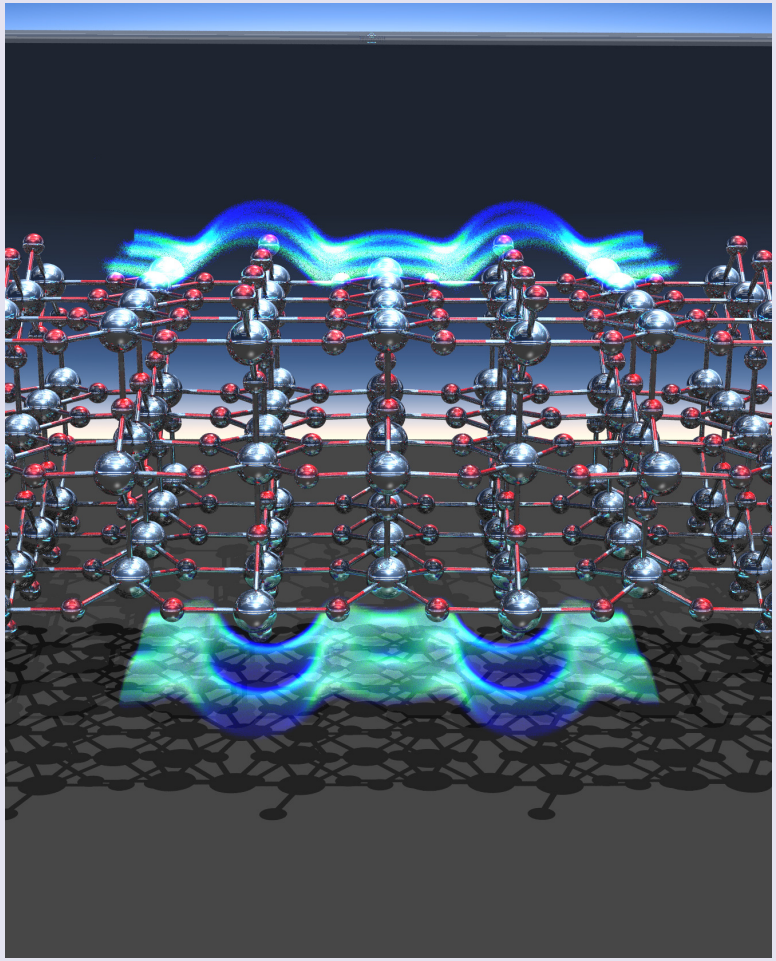
The dielectric function determine the cavity where the solute is defined.

The cavity can be

- rigid (PCM-like)
- determined from the Electronic Density (SCCS approach)

☛ Can treat various BC

(here TiO_2 surface)



Can be used in conjunction with $O(N)$ BigDFT

From vacuum to complex wet environments

Neutral or ionic wet environments

Generalized Poisson eq.

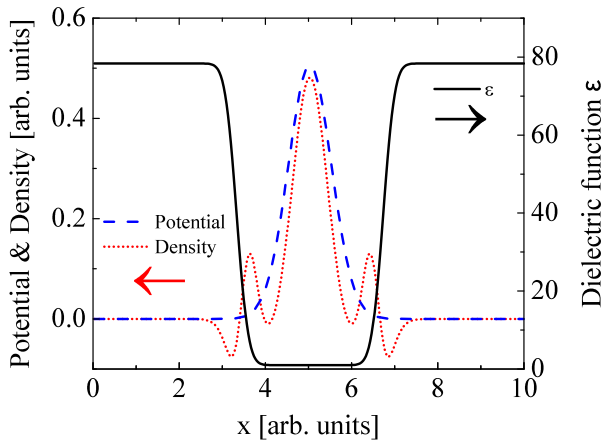
$$\nabla \cdot \varepsilon(\mathbf{r}) \nabla \phi(\mathbf{r}) = -4\pi\rho(\mathbf{r})$$

Poisson-Boltzmann eq.

$$\nabla \cdot \varepsilon(\mathbf{r}) \nabla \phi(\mathbf{r}) = -4\pi \left[\rho(\mathbf{r}) + \rho^{ions}[\phi](\mathbf{r})\right]$$

Various algorithms implemented

- Polarization Iteration (use gradient of polarization potential)
- Preconditioned Conjugate Gradient (need only function multiplications)

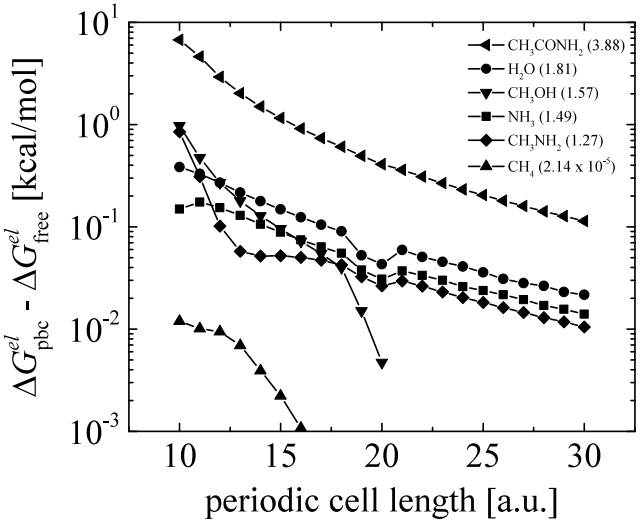


Solution found by iterative application of PS in vacuum BC

Modelling of Electrostatic Environment

Advantages of explicit BC

Difference between the electrostatic solvation energy computed with periodic and free boundary conditions as function of the periodic cell length.



Molecule dipole norm (D)

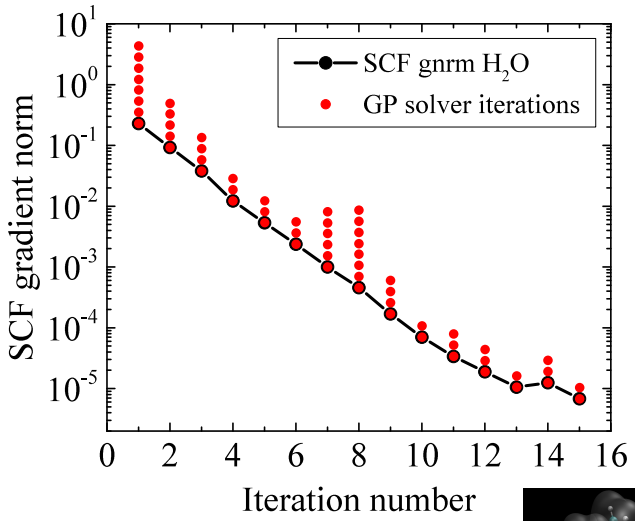
	vacuum	H_2O
CH_3CONH_2	3.88	5.76
H_2O	1.81	2.41
CH_3OH	1.57	2.14
NH_3	1.49	1.98
CH_3NH_2	1.27	1.78

Explicit BC avoid error due to supercell aliasing

Performances and timings for full DFT runs

Blackbox-like usage

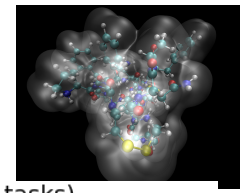
The Generalized PS only needs few iterations of the vacuum poisson solver

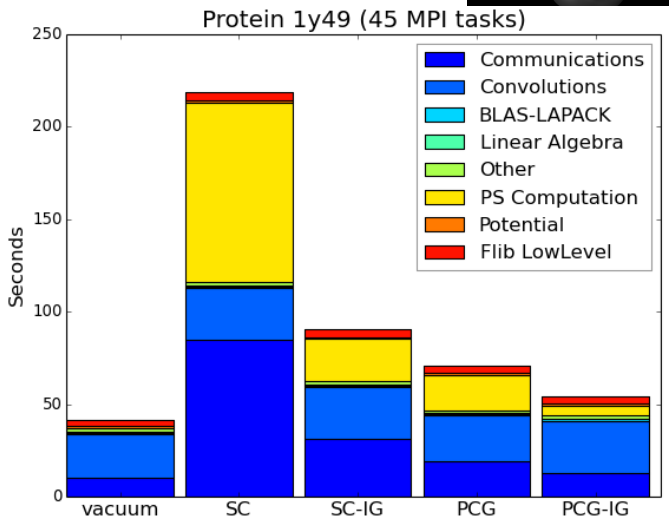


Time-to-solution

Timings for the protein PDB ID: 1y49 (122 atoms) in water

- Full SCF convergence 49 s
- Solvent/vacuum runtime ratio $\alpha = 1.16$





ISF: wavelet analysis meets real-space

Features of Systematicity

- ISF provide a rigorous framework to interpret real-space coefficients
- High interpolating power (precise results)
- Scaling relation $\rightarrow$ computational efficiency, arbitrary precision
- (Quasi) variational
- In conjunction with (Daubechies) wavelets offer a good formalism to get rid of uncertainties

Opportunities from new quadratures

- Generalize the concept of compensating charge (multipoles)
- Combine together cartesian and polar meshes