Optimized Implementation of Coupling Matrix Computation in Time Dependent Density Functional Theory

Shivaraju B. Gowda

Master's thesis defense

October 7th, 2004

Motivation

- Study the properties of materials at atomic scale:

  Applications : Crystal growth,

  Semiconductor Integrated circuits, etc.

- For steady state material properties :

  ab initio pseudopotentials within Density Functional Theory (DFT)

- For time-dependent material properties :

  Ex: electronic excitations, optical spectra

  Time-Dependent DFT (TDDFT), (Tested and validated with experiments for hundreds of atoms)

- The main computational bottleneck of TDDFT :

  Coupling matrix computation ⟵ Our problem of interest

Motivation (contd.)

- Task already accomplished in Real space.
- Sample problem, Si34H36 --- takes 15hrs on 8 processors
- Goal now : gain on speed and memory….
- Go beyond Si275H172… using.

    Better algorithms

    Optimized implementation

    Suitable approximations

    Sparsity of the wave functions

    Faster computer (power3 $\rightarrow$ power4, Itanium2)

Outline

- Formalism
    - TDDFT
    - Coupling matrix construction
    - Real-space implementation
- Implementation
    - Solution of Poisson equation
    - Efficient Integration
    - Parallel Implementation
- Numerical Examples
    - Validation
    - Performance
    - Faster Machines
- Conclusion

Formalism- TDDFT

Three steps to calculating the optical spectra

1. Solve time-independent formulation of the pseudopotential-DFT method.

$$\left(-\frac{\triangle}{2}+\sum_{R_a} V_{ps}(\mathbf{r}-\mathbf{R}_a)+V_H(\rho(\mathbf{r}))+V_{xc}(\rho(\mathbf{r}))\right)\psi_n(\mathbf{r})=\epsilon_n\psi_n(\mathbf{r}).$$

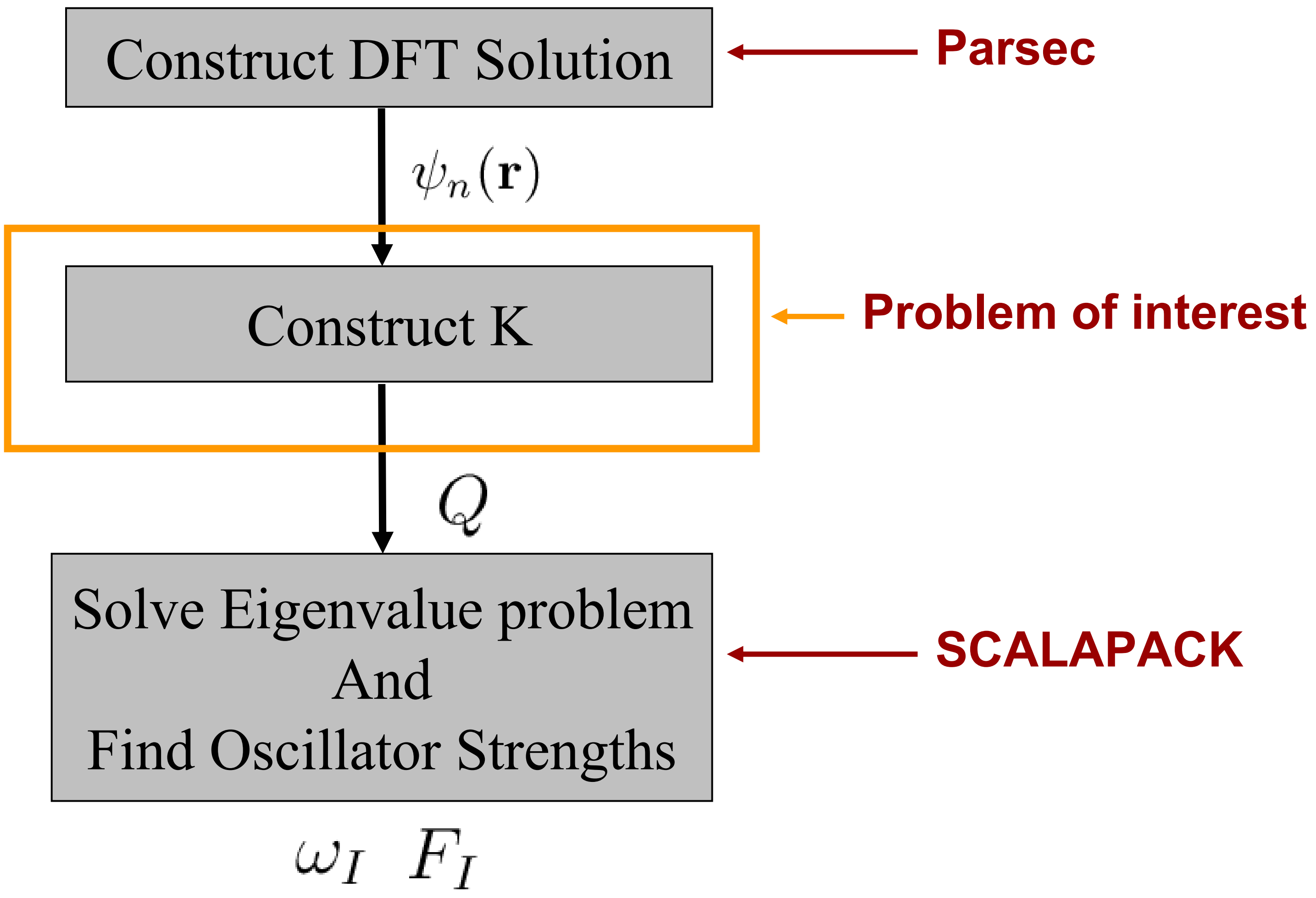
2. Construct the coupling matrix in the form.

$$K_{ij,kl}=2\int_\Omega\int_\Omega \bar{\psi}_i(\mathbf{r})\psi_j(\mathbf{r})\left(\frac{1}{|\mathbf{r}-\mathbf{r}'|}+\frac{dV_{xc}(\rho(\mathbf{r}))}{d\rho(\mathbf{r})}\delta(\mathbf{r}-\mathbf{r}')\right)\psi_k(\mathbf{r}')\bar{\psi}_l(\mathbf{r}')d\mathbf{r}d\mathbf{r}'$$

$$Q_{ij,kl}=\delta_{ik}\delta_{jl}\omega_{kl}^2+2\sqrt{\lambda_{ij}\omega_{ij}}K_{ij,kl}\sqrt{\lambda_{kl}\omega_{kl}}$$

3. Solve for eigenvalues of Q and find absorption strength

$$QF_I=\omega_I^2F_I, \qquad f_I=\frac{2}{3}\sum_{\beta=\{x,y,z\}}|B_\beta^T R^{1/2}F_I|^2.$$



$$f_I = \frac{2}{3} \sum_{\beta=\{x,y,z\}} |B_\beta^T R^{1/2} F_I|^2 .$$

Coupling matrix Construction

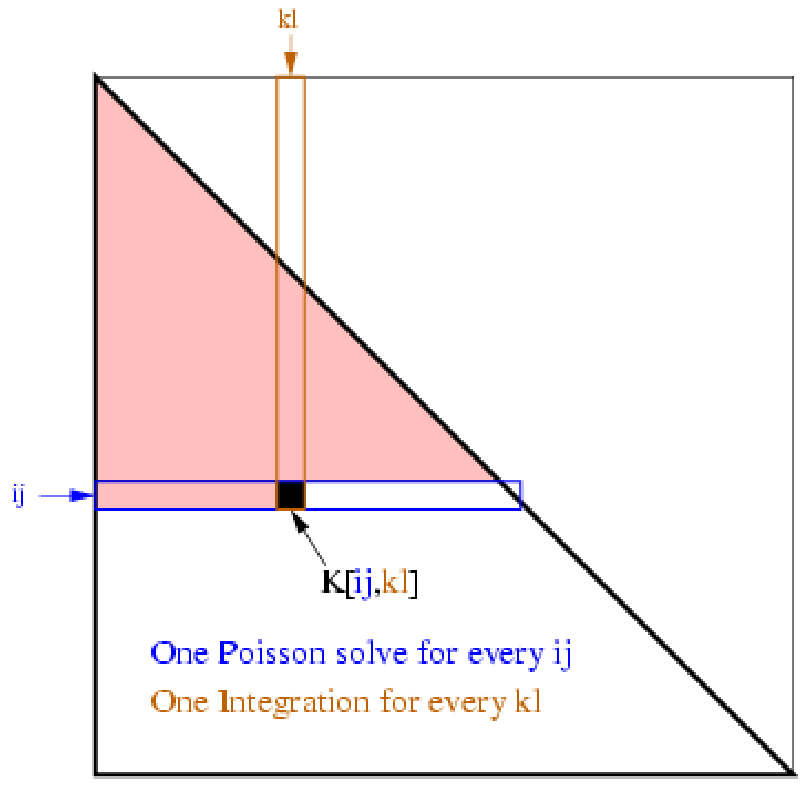
$$K_{ij,kl} = 2\int_\Omega\int_\Omega \bar{\psi}_i(\mathbf{r})\psi_j(\mathbf{r})\left(\frac{1}{|\mathbf{r}-\mathbf{r}'|} + \frac{dV_{xc}(\rho(\mathbf{r}))}{d\rho(\mathbf{r})}\delta(\mathbf{r}-\mathbf{r}')\right)\psi_k(\mathbf{r}')\bar{\psi}_l(\mathbf{r}')d\mathbf{r}d\mathbf{r}'$$

Define:

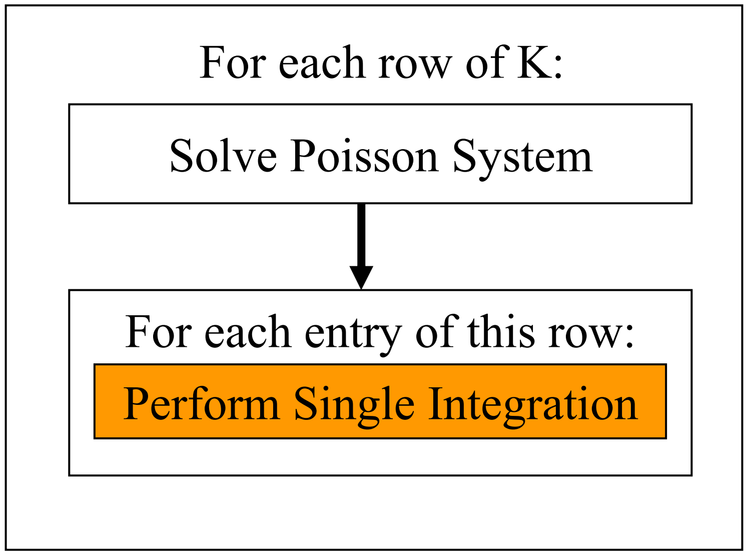
$$\rho_{ij}(\mathbf{r}) \equiv \bar{\psi}_i(\mathbf{r})\psi_j(\mathbf{r})$$

$$\Phi_{ij}(\mathbf{r}') \equiv \int d\mathbf{r}\frac{1}{|\mathbf{r}-\mathbf{r}'|}\bar{\psi}_i(\mathbf{r})\psi_j \longrightarrow \nabla^2\Phi_{ij}(\mathbf{r}') = -4\pi\rho_{ij}(\mathbf{r}').$$

$$K_{ij,kl} = 2\int_\Omega d\mathbf{r}\left[\Phi_{ij}(\mathbf{r}) + \rho_{ij}(\mathbf{r})\frac{dV_{xc}[\rho(\mathbf{r})]}{d\rho(\mathbf{r})}\right]\rho_{lk}(\mathbf{r}).$$



Coupling Matrix K

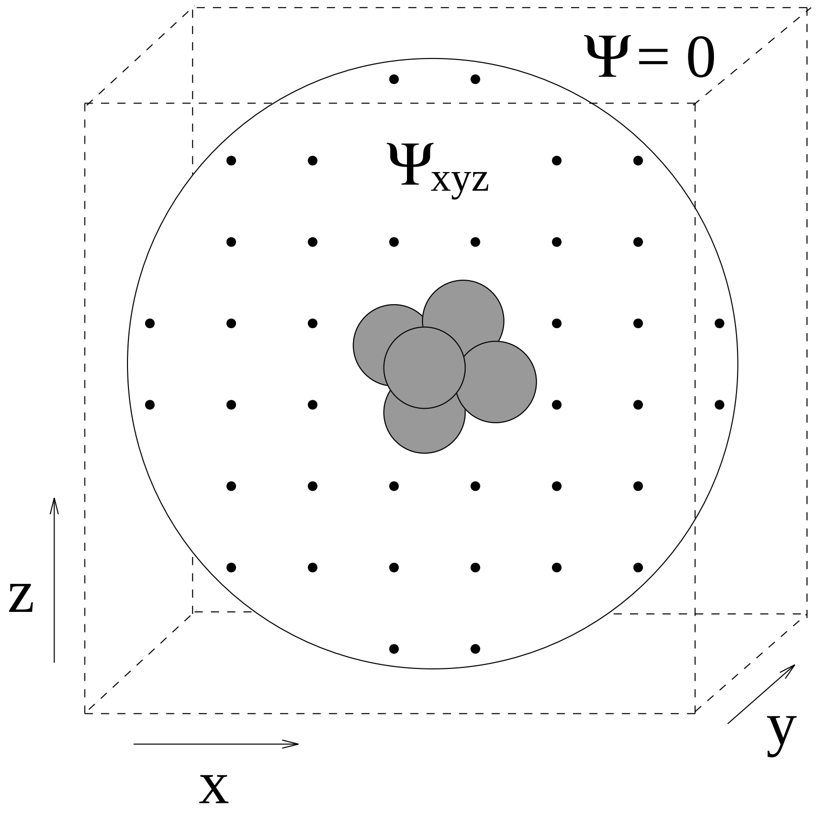


$$K_{ij,kl} = 2\int_{\Omega} d\mathbf{r} \left[\Phi_{ij}(\mathbf{r}) + \rho_{ij}(\mathbf{r})\frac{dV_{xc}[\rho(\mathbf{r})]}{d\rho(\mathbf{r})}\right]\rho_{lk}(\mathbf{r}).$$

- K is of the order of the square of the number of states considered.
- K is symmetric, only lower triangular part is assembled.
- Calculation of each row is independent of others, embarrassingly parallel.
- Poisson equation is solved for different RHS.
- Solve one Poisson equation for each row of K then use direct summation to evaluate the single integral for each element of K.
- Wave functions are localized in real space, i.e., have small support (sparse). This property can be used for optimized integration.

Real-space Implementation

- Potentials and wave functions are set up on a simple 3-D grid with spherical domain.
- The Laplacian is approximated by a high-order finite-difference expansion.
- The potential does not necessarily vanish outside the sphere so they are evaluated on the spherical surface using multipole expansion.
- The corresponding boundary conditions are applied to the linear system.
- The linear system is solved using Preconditioned Conjugate Gradient (PCG) solver.



Si34H36

Total = 15:30 Hrs

Poisson solution time = 6hrs

Real-space Implementation vs. Fourier-space Implementation

	Real-space	Fourier-space
Poisson equation solver	Preconditioned Conjugate Gradient (PCG)	FFT based solver
Parallel Implementation	"Master-slave" model	Reuse the real-space model
Language	Fortran77	Fortran90
Approximations	No approximations with respect to the bounded support of the wave-functions	Framework to allow the user to set the desired approximations
Math libraries	Doesn't significantly make use of math libraries	Use math libraries for all computationally intensive loops

Solution of Poisson equation

- Regular 3D grid with constant grid spacing
- Constant coefficients
- Finite boundary conditions
- Main problem solved in parallel but subsidiary problem on one processor so don't need parallelism.

Method	Complexity	Memory
Gaussian Elimination	N^3	N^2
PCG	$N^{3/2}$	$N^{1/2} log N$
FFT	$N log N$	$log N$
Multigrid	N	$log^2 N$

Multigrid methods : Efficient implementation is complicated due to non-uniform memory access pattern

Uncorrected solution of Poisson equation

$$\Phi_{ij}(\mathbf{r}) = 4\pi\mathcal{F}^{-1}\left[\,\mathcal{F}(\Psi_i\bar{\Psi}_j)(\mathbf{k})/\|\mathbf{k}\|^2\,\right](\mathbf{r}).$$

$$f^{cut} \longrightarrow \frac{1}{\|\mathbf{k}\|^2}$$

$$\Phi_{ij}(\mathbf{r}) = 4\pi\mathcal{F}^{-1}\left[\,\mathcal{F}(\Psi_i\bar{\Psi}_j)(\mathbf{k})\cdot f^{cut}\,\right](\mathbf{r}).$$

Algorithm 1.3.1: Poisson Equation Solution

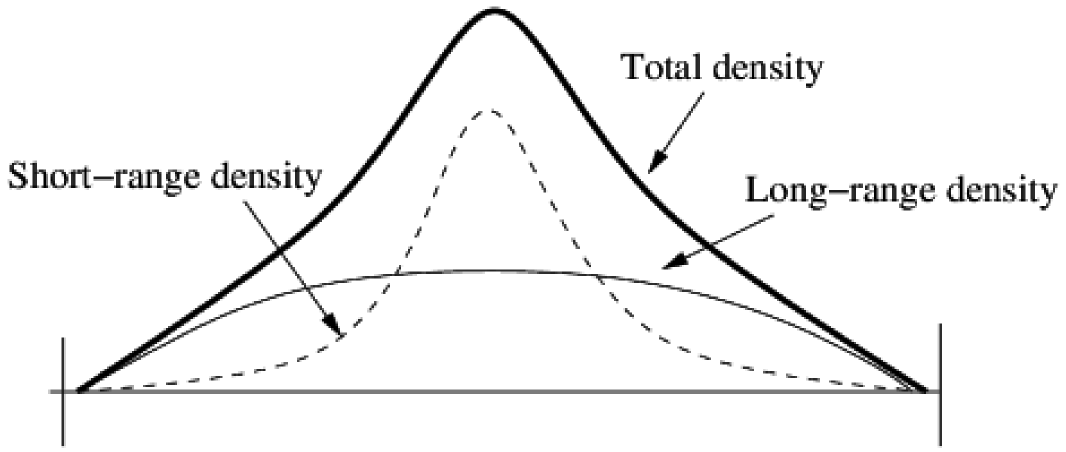
```
1:  Compute $f^{cut} \longrightarrow \frac{1}{\|\mathbf{k}\|^2}$
2:  Start of Coupling Matrix Assembly Loop
3:  for all the transitions $ij$ considered do
4:      $\rho_{ij} = \psi_i\psi_j$
5:      C ——— Start Poisson Equation Solver ————-
6:      Apply Forward Fourier Transform: $\mathcal{F}(\rho_{ij})(\mathbf{k})$
7:      Multiply by $f^{cut}$ : $\mathcal{F}(\rho_{ij})(\mathbf{k})f^{cut}$.
8:      Apply Inverse Fourier Transform: $\mathcal{F}^{-1}(\mathcal{F}(\rho_{ij})(\mathbf{k})f^{cut})$
9:      C ——— End Poisson Equation Solver ————-
10:     Assemble the remaining elements of the row corresponding to $ij$
11: end for
12: End of Coupling Matrix Assembly Loop.
```

- Only Suitable for periodic systems
- For localized systems involves spurious affects of periodic images

Long-range and short-range splitting

$$-\triangle\Phi_{ij}^{(\mathrm{long})}(\mathbf{r}) = 4\pi\rho_{ij}^{(\mathrm{long})}(\mathbf{r}),$$

$$-\triangle\Phi_{ij}^{(\mathrm{short})}(\mathbf{r}) = 4\pi\rho_{ij}^{(\mathrm{short})}(\mathbf{r}) = 4\pi\left(\rho_{ij}(\mathbf{r}) - \rho_{ij}^{(\mathrm{long})}(\mathbf{r})\right)$$



1. Solve the long-range part analytically using multipole expansion
2. Solve the short-range part using FFT
3. Sum up both potentials to obtain total potential

$$\Phi_{ij} = \Phi_{ij}^{(long)} + \Phi_{ij}^{(short)}$$

Algorithm 3.3.2: Long-range and short-range splitting

```
1: Compute and store $\rho_l$ for all $l \leq l_{cut}$.
2: Solve the one dimensional equations to obtain $\Phi_l$.
3: C ——- Start of Coupling Matrix Assembly Loop. ————
4: for all the transitions $ij$ considered do
5:     $\rho_{ij} = \psi_i \psi_j$.
6:     C ——— Start Poisson Equation Solver ————-
7:     Compute $q_{lm}$, the spherical multipole moments.
8:     Compute $\rho_{ij}^{(long)}$
9:     Substract $\rho_{ij}^{(long)}$ from $\rho$ to obtain $\rho_{ij}^{(short)}$
10:    Apply Forward Fourier Transform: $\mathcal{F}(\rho_{ij}^{(short)})(\mathbf{k})$.
11:    Multiply by $f^{cut}$ : $\mathcal{F}(\rho_{ij}^{(short)})(\mathbf{k}) f^{cut}$.
12:    Apply Inverse Fourier Transform: $\mathcal{F}^{-1}(\mathcal{F}(\rho_{ij}^{(short)})(\mathbf{k}) f^{cut})$.
13:    Compute $\Phi_{ij}^{(long)}$ and then the total potential, $\Phi_{ij}$.
14:    C ——- End of Coupling Matrix Assembly Loop. ————
15:    Assemble the remaining elements of the row corresponding to $ij$.
16: end for
17: End of Coupling Matrix Assembly Loop.
```

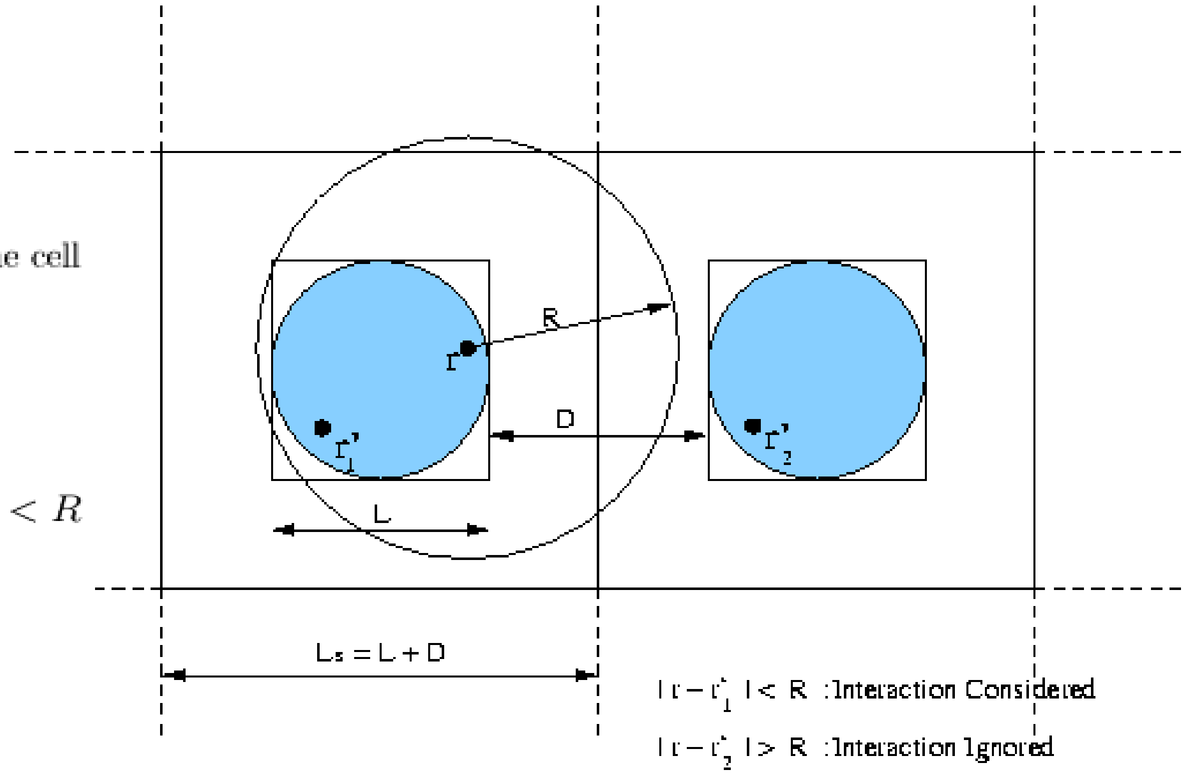
Cut-off based methods

$$\Phi^{cut}(\mathbf{r}) = \int_{cell} \frac{\rho(\mathbf{r}')}{\|\mathbf{r}-\mathbf{r}'\|} d\mathbf{r}'.$$

$$f^{cut}(\mathbf{r},\mathbf{r}') = \begin{cases} \dfrac{1}{|\mathbf{r}-\mathbf{r}'|} & \text{For } \mathbf{r},\ \mathbf{r}' \text{ in same cell} \\ 0 & \text{Otherwise} \end{cases}$$

$$f^{cut}(\mathbf{r},\mathbf{r}') = \begin{cases} \dfrac{1}{|\mathbf{r}-\mathbf{r}'|} & \text{For } |\mathbf{r}-\mathbf{r}'| < R \\ 0 & \text{Otherwise} \end{cases}$$

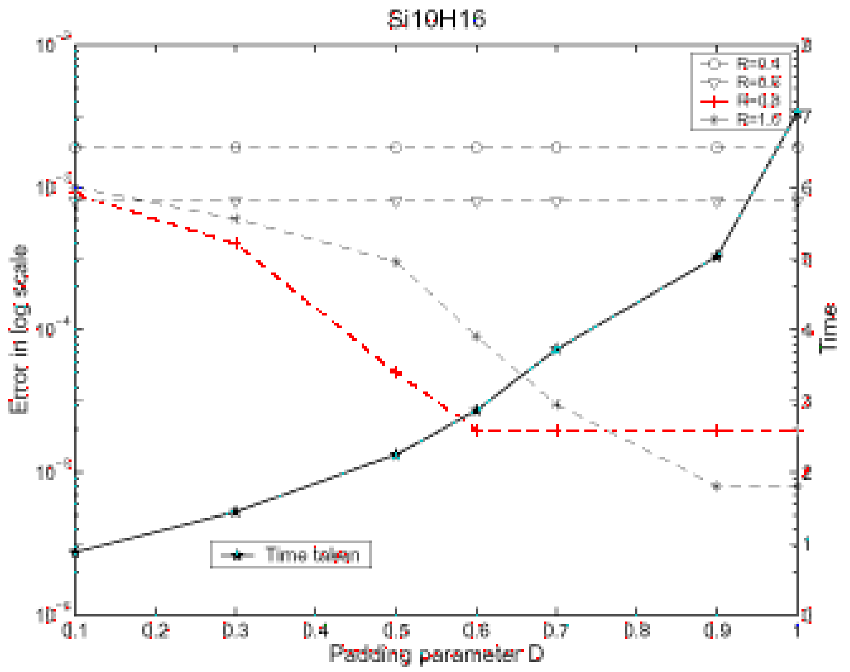
$$f^{cut} \rightarrow \frac{1}{|\mathbf{k}|^2}\left[1-\cos(\mathbf{k}R)\right]$$

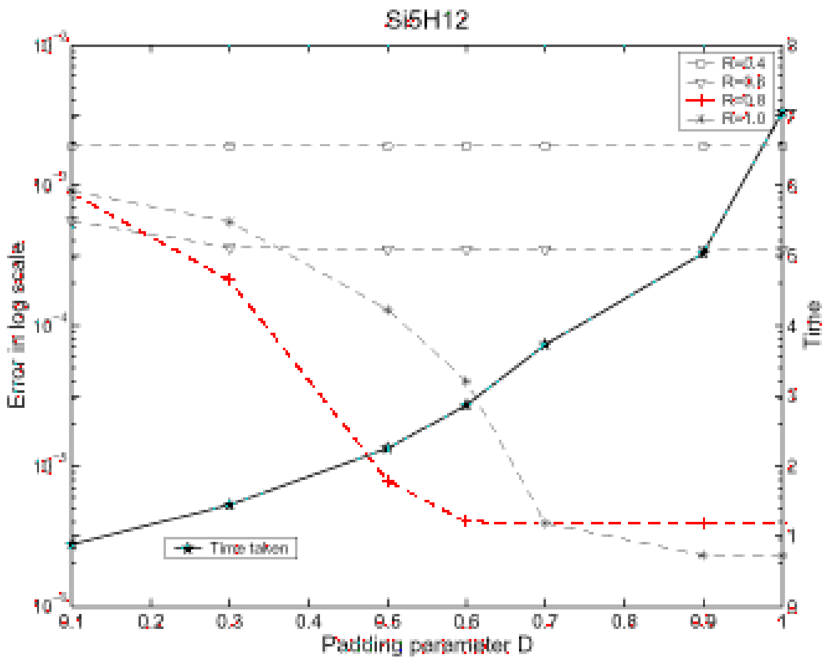

R
r
D
r'1
r'2
L
Ls = L + D
| r − r'1 | < R :Interaction Considered
| r − r'2 | > R :Interaction Ignored

Algorithm 1.3.3: Poisson Equation Solution with cut-off

```
1: Compute $f^{cut} \longrightarrow \frac{1-\cos(\mathbf{k}R)}{\|\mathbf{k}\|^2}$ for the Super-Cell
2: Start of Coupling Matrix Assembly Loop
3: for all the transitions $ij$ considered do
4:     $\rho_{ij} = \psi_i \psi_j$
5:     C ——— Start Poisson Equation Solver ————-
6:     Build a super-cell twice the size of the Original Cell
7:     Project the Original cell to the center of the Super-Cell
8:     Apply Forward Fourier Transform: $\mathcal{F}(\rho_{ij})(\mathbf{k})$
9:     Multiply by $f^{cut}$ : $\mathcal{F}(\rho_{ij})(\mathbf{k}) f^{cut}$.
10:    Apply Inverse Fourier Transform: $\mathcal{F}^{-1}(\mathcal{F}(\rho_{ij})(\mathbf{k}) f^{cut})$
11:    Collect the values of the potential from the Super-Cell
12:    C ——— End Poisson Equation Solver ————-
13:    Assemble the remaining elements of the row corresponding to $ij$
14: end for
15: End of Coupling Matrix Assembly Loop.
```

Parametric study for R and D





- Error indicates error in potential with respect to Real-space code
- D and R normalized with respect to L
- Time in seconds for 100 Poisson equation solutions
- R=0.8 and D= 0.6 optimum choice

Assembling the Coupling matrix

$$K_{ij,kl} = 2\int_{\Omega} \left[\Phi_{ij}(\mathbf{r}) + \Psi_i(\mathbf{r})\bar{\Psi}_j(\mathbf{r})\frac{dV_{\text{XC}}[\rho(\mathbf{r})]}{d\rho(\mathbf{r})}\right]\Psi_k(\mathbf{r})\bar{\Psi}_l(\mathbf{r})d\mathbf{r}.$$

$\rho(\mathbf{r})$, constant throughout the construction

$\Rightarrow$ $\dfrac{dV_{\text{XC}}[\rho(\mathbf{r})]}{d\rho(\mathbf{r})}$ Independent of *ij, kl*

1. Compute $\text{ker}_{ij}(\mathbf{r}) = \left[\Phi_{ij}(\mathbf{r}) + \rho_{ij}(\mathbf{r})\frac{dV_{xc}[\rho(\mathbf{r})]}{d\rho(\mathbf{r})}\right].$

2. Assemble $K_{ij,kl} = \sum_{\mathbf{r}} \left[\text{ker}_{ij}(\mathbf{r})\right]\psi_k(\mathbf{r})\bar{\psi_l}(\mathbf{r}).$

Direct summation used for integration

Real-space implementation

Algorithm 1.4: Coupling matrix assembly in Real-space code

```
 1: Compute $\frac{dV_{xc}[\rho(\mathbf{r})]}{d\rho(\mathbf{r})}$
 2: Start of Coupling Matrix Assembly Loop
 3: for all the transitions $ij$ considered do
 4:    $\rho_{ij}(\mathbf{r}) = \psi_i(\mathbf{r})\bar{\psi}_j(\mathbf{r})$
 5:    Solve for $\Phi_{ij}(\mathbf{r})$ by using PCG with $\rho_{ij}(\mathbf{r})$
 6:    Compute $kl$ independent kernel : $\mathrm{ker}_{ij}(\mathbf{r}) = \left[\Phi_{ij}(\mathbf{r}) + \rho_{ij}(\mathbf{r})\frac{dV_{xc}[\rho(\mathbf{r})]}{d\rho(\mathbf{r})}\right]$.
 7:    for all $k$ do
 8:       for all $l$ do
 9:          $K_{ij,kl} = \sum_{\mathbf{r}} \boxed{\mathrm{ker}_{ij}(\mathbf{r})\psi_k(\mathbf{r})}\bar{\psi}_l(\mathbf{r})$.
10:       end for
11:    end for
12: end for
13: End of Coupling Matrix Assembly Loop.
```

Implemented with "do loops"

Independent of *l*

Fourier-space implementation

Algorithm 1.4: Coupling matrix assembly in Fourier space code

```
Compute $\frac{dV_{xc}[\rho(\mathbf{r})]}{d\rho(\mathbf{r})}$
Start of Coupling Matrix Assembly Loop
for all the transitions $ij$ considered do
    $\rho_{ij}(\mathbf{r}) = \psi_i(\mathbf{r})\bar{\psi}_j(\mathbf{r}) \longleftarrow$ Use DVEM
    Solve for $\Phi_{ij}(\mathbf{r})$ by using FFT with $\rho_{ij}(\mathbf{r})$
    $\text{ker}_{ij}(\mathbf{r}) = \left[\Phi_{ij}(\mathbf{r}) + \rho_{ij}(\mathbf{r})\frac{dV_{xc}[\rho(\mathbf{r})]}{d\rho(\mathbf{r})}\right] \longleftarrow$ Use DVEM
    for all $k$ do
        $\text{tmp}_{ij}(\mathbf{r}) = \text{ker}_{ij}(\mathbf{r})\psi_k(\mathbf{r}) \longleftarrow$ Use DVEM
        for all $l$ do
            $K_{ij,kl} = \sum_{\mathbf{r}} \text{tmp}_{ij}(\mathbf{r})\bar{\psi}_l(\mathbf{r}) \longleftarrow$ Use DDOT
        end for
    end for
end for
End of Coupling Matrix Assembly Loop.
```

Using bounded support of the wave-functions

Wave-functions have bounded support

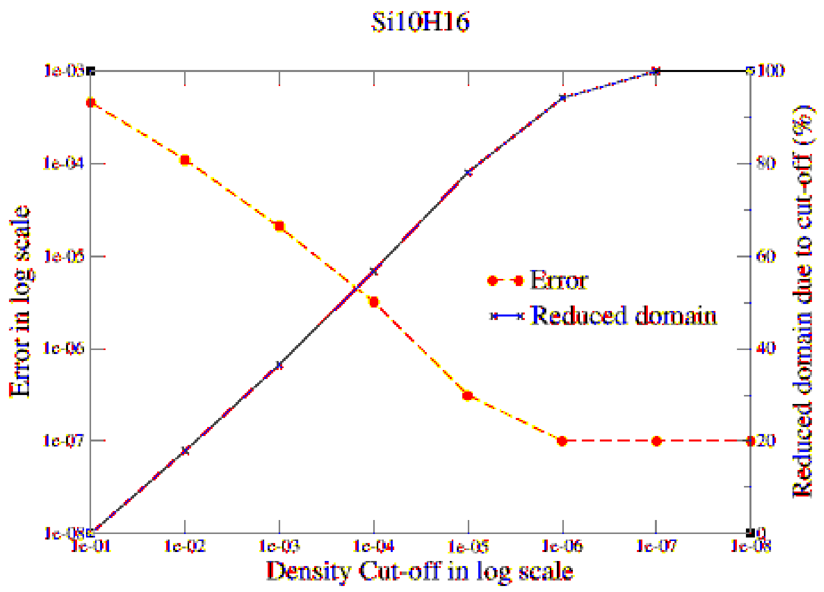
→ Only necessary to sum over $\mathrm{Supp}(\Psi_k) \cap \mathrm{Supp}(\Psi_l) \subset \Omega$

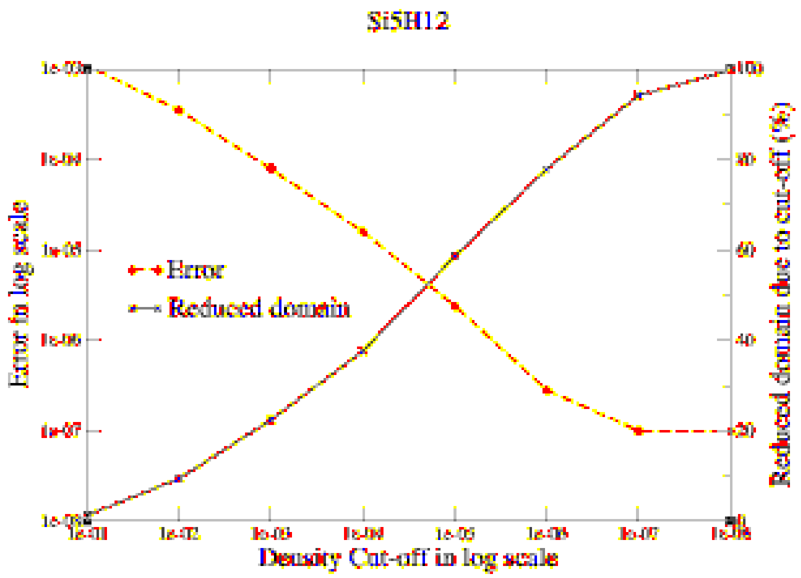
$$K_{ij,kl} = \sum_{\mathbf{r}} \left[\mathrm{ker}_{ij}(\mathbf{r}) \right] \psi_k(\mathbf{r}) \bar{\psi}_l(\mathbf{r}).$$

$$K_{ij,kl} = \int_{\mathrm{Supp}(\Psi_k) \,\cap\, \mathrm{Supp}(\Psi_l)} \left(\Psi_i(\mathbf{r}) \bar{\Psi}_j(\mathbf{r}) \frac{dV_{\mathrm{XC}}(\mathbf{r})}{d\rho(\mathbf{r})} + \Phi_{ij}(\mathbf{r}) \right) \Psi_k(\mathbf{r}) \bar{\Psi}_l(\mathbf{r}) d\mathbf{r}.$$

$$K_{ij,kl} = \int_{\{\mathbf{r} \;|\; \rho(\mathbf{r}) > \varepsilon\}} \left(\Psi_i(\mathbf{r}) \bar{\Psi}_j(\mathbf{r}) \frac{dV_{\mathrm{XC}}(\mathbf{r})}{d\rho(\mathbf{r})} + \Phi_{ij}(\mathbf{r}) \right) \Psi_k(\mathbf{r}) \bar{\Psi}_l(\mathbf{r}) d\mathbf{r}.$$

Parametric study for dcut (density cut-off)





- Error indicates error in coupling matrix with respect to Real-space code
- dcut = 1e-6 Optimum choice

Implementation Details

- Platforms : IBM SP and Linux
- Computer Language : Fortran90
    - easy array manipulation.
    - easy to read
    - reuse fortran 77
- Array storage: 3 dimensional for Fourier transform, 1 dimensional for wavefunctions.
- FFT Libraries : ESSL, MKL and FFTw
- Math Libraries : ESSL and MKL.
- Parallelization : Dynamic, “Master-slave” approach

Numerical Examples

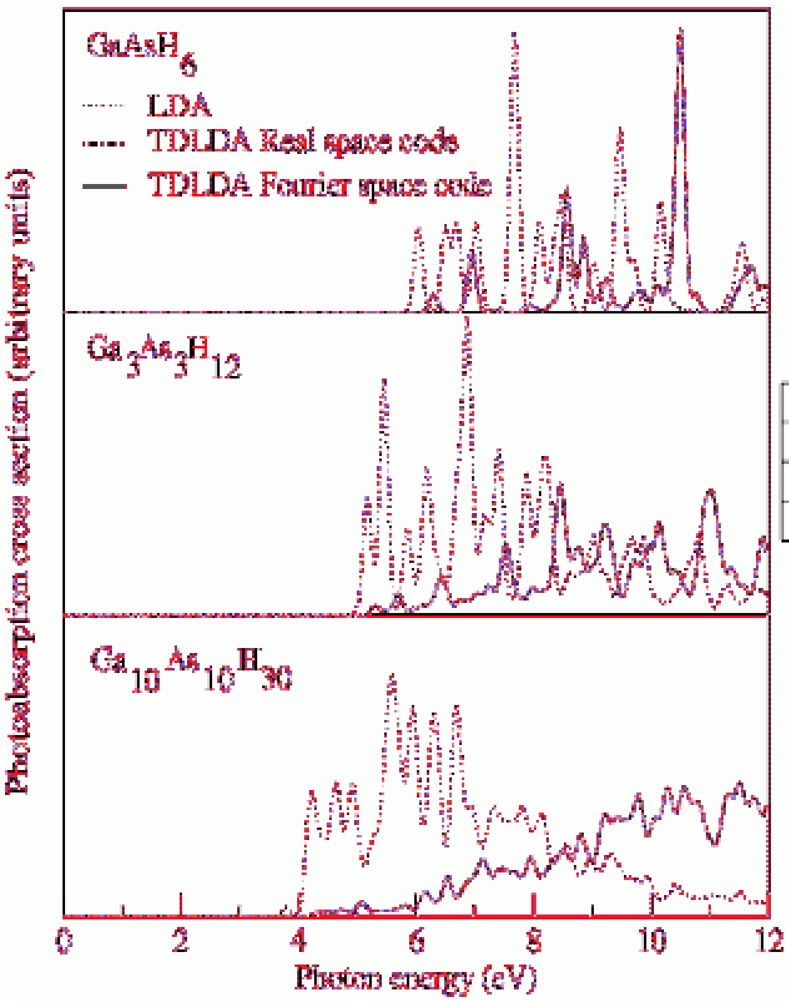
Test cases : Galium Arsenide, Hydrogenated silicon

Machine : IBM power3, 375 MHz, -O3 and -qtune=pwr3

Parameters : R=0.8, D=0.6, dcut = 1e-6

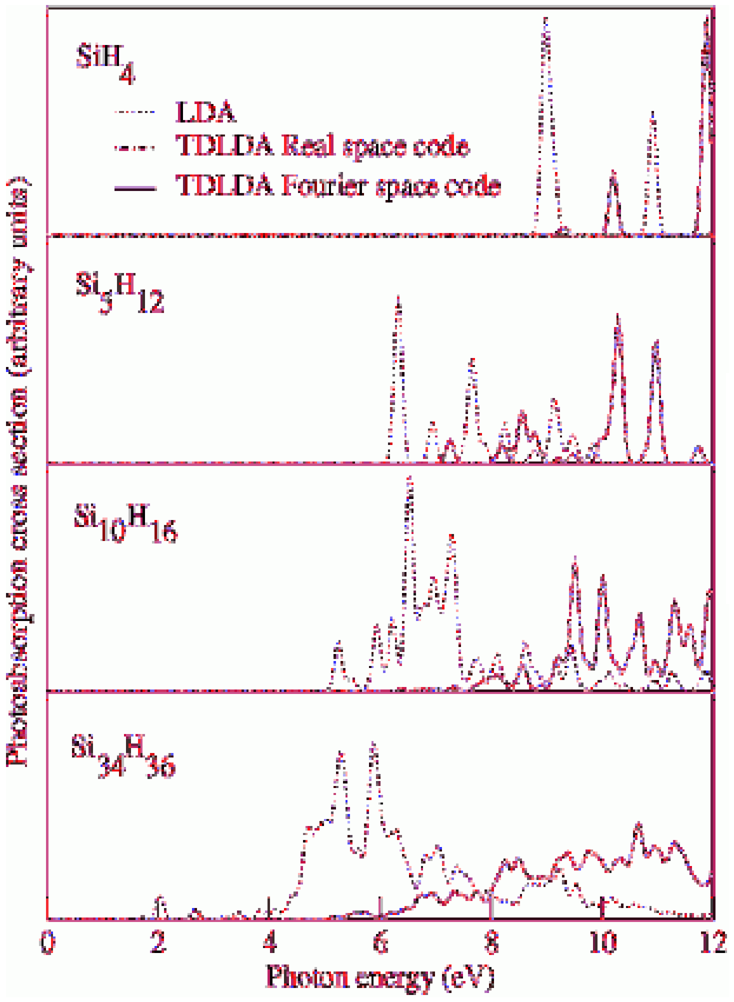
FFT library : FFTw   Math Library : ESSL

Galium Arsenide clusters



	Grid	Grid spacing	Radius	nstate	nocc	kdim
$GaAsH_6^{'}$	$49 \times 49 \times 49$	0.45 a.u	11 a.u	55	7	336
$Ga_3As_3H_{12}^{'}$	$47 \times 47 \times 47$	0.6 a.u	14 a.u	82	18	1152
$Ga_{10}As_{10}H_{30}^{'}$	$61 \times 61 \times 61$	0.6 a.u	18 a.u	151	55	5280

Hydrogenated silicon clusters



	Grid	Grid spacing	Radius	nstate	nocc	kdim
SiH_4	$21 \times 21 \times 21$	0.75 a.u	7.5 a.u	15	4	44
Si_5H_{12}	$33 \times 33 \times 33$	0.75 a.u	12 a.u	30	16	224
$Si_{10}H_{16}$	$33 \times 33 \times 33$	0.75 a.u	12 a.u	60	28	896
$Si_{34}H_{36}$	$65 \times 65 \times 65$	0.75 a.u	24 a.u	240	86	13244

Performance

Time for completion :

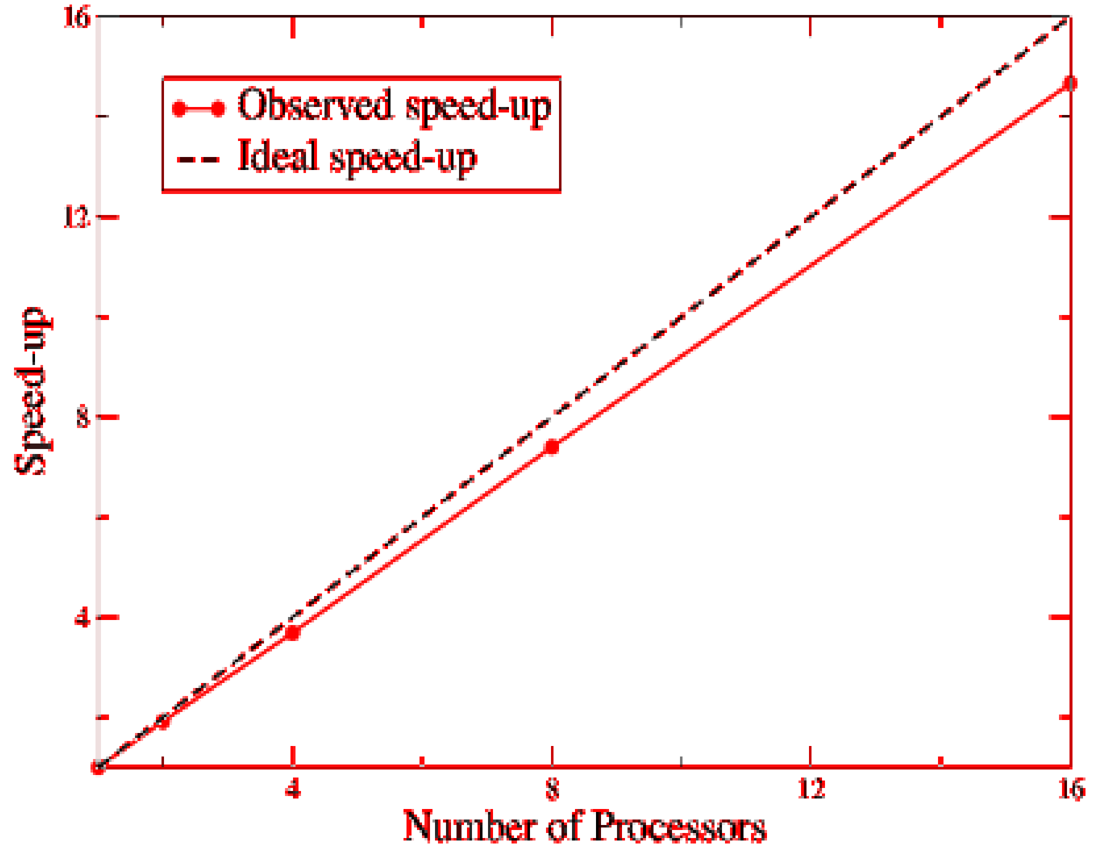
	SiH_4	Si_5H_{12}	$Si_{10}H_{16}$	$Si_{34}H_{36}$
Number of Processors	1	1	1	8
Real-space Code	6 sec	121 sec	9:35 min	15:30 hrs
Fourier-space Code	1 sec	15 sec	1:29 min	1:35 hrs
Speed-up	6.0 times	8.0 times	6.5 times	9.8 times

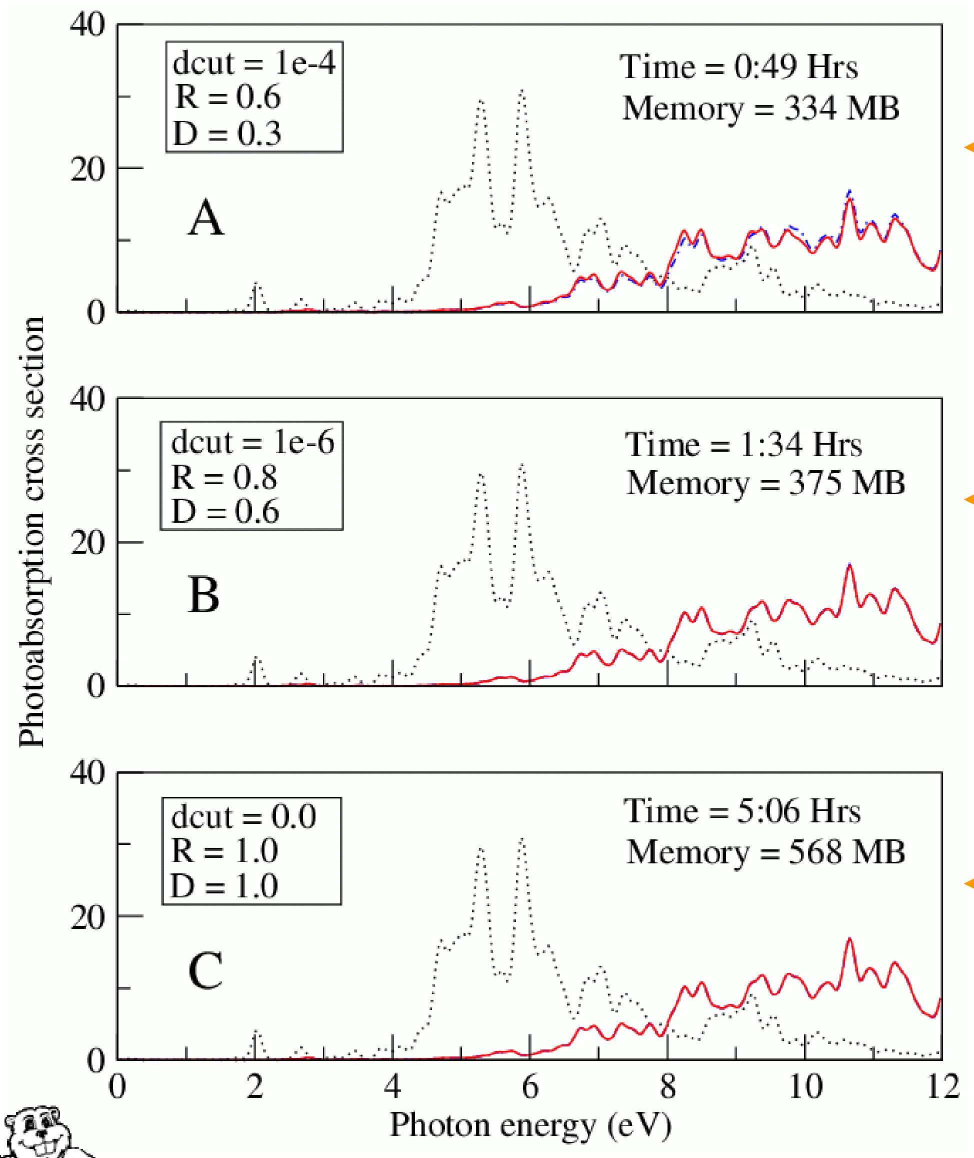
Memory space required :

	SiH_4	Si_5H_{12}	$Si_{10}H_{16}$	$Si_{34}H_{36}$
Real-space Code	17 MB	33 MB	38 MB	420 MB
Fourier-space Code	11 MB	21 MB	29 MB	375 MB
Percent reduction	54%	57%	31%	12%

Scalability :

Test case: Si34H36





Best case with respect to time

With suggested parameters

Best case with respect to Accuracy

Hardware performance

	IBM p655	SGI Altix
Processor	Power4	Intel Itanium2
Processor Speed	1.5GHz	1.5GHz
Memory requested	500 MB/Processor	500 MB/Processor
Compiler	xlf_r	ifort 8.0
Compiler options	-O3 -qtune="pwr4"	-O3 -ftz
FFT Library	FFTw	FFTw
BLAS Library	ESSL	SCSL

	Setup Time	Poisson solver time	Total CPU time	Wall-Clock Time
IBM Regatta	70 secs	2151 secs	19093 secs	42:00 mins
SGI Altix	28 secs	3447 secs	15660 secs	35:24 mins

Conclusion

- Reduced the total CPU time by an order of magnitude for the construction of the coupling matrix in TDDFT
- Some approximations involved: suggested optimal parameters after parametric studies
- Even with no approximations, present implementation 3 times faster than the previous implementation
- Compared the performance of the Itanium2 and Power4 systems.

Future work

- Sophisticated integration techniques for assembly
- Assemble block of elements at a time
- Eigen-value problem(5hrs for Si34H36)
- FFT based methods for calculation of Hartree potential in DFT part

Thank You