Institute for the Theory of Advanced Materials in Information Technology

Computational Challenges and Solution Methods in DFT and TDDFT

Yousef Saad

University of Minnesota

Department of Computer Science and Engineering

ITAMIT Workshop, Aug. 6-7, 2004

Current and recent members of the team: (CS side)

Shiv Gowda	Masters student - Scientific computing
Efi Kokiopoulou	PhD student – Computer Science
Emmanuel Lorin de La Grandmaison	Post-doc
Costas Bekas	Post-Doc
Yunkai Zhou	Post-Doc
Susanne Shontz	Post-Doc [starts in Sept. 04]

Main issues of interest:

- Solve the eigenvalue problem efficiently [specificity: large number of eigenvalues]
- Find alternatives [avoid eigenvectors, eigenvalues]
- Solve various related computational problems [TDDFT, computation of dielectric matrix, ...]

Challenges in DFT

- DFT leads to a one-electron Shrödinger equation of the form

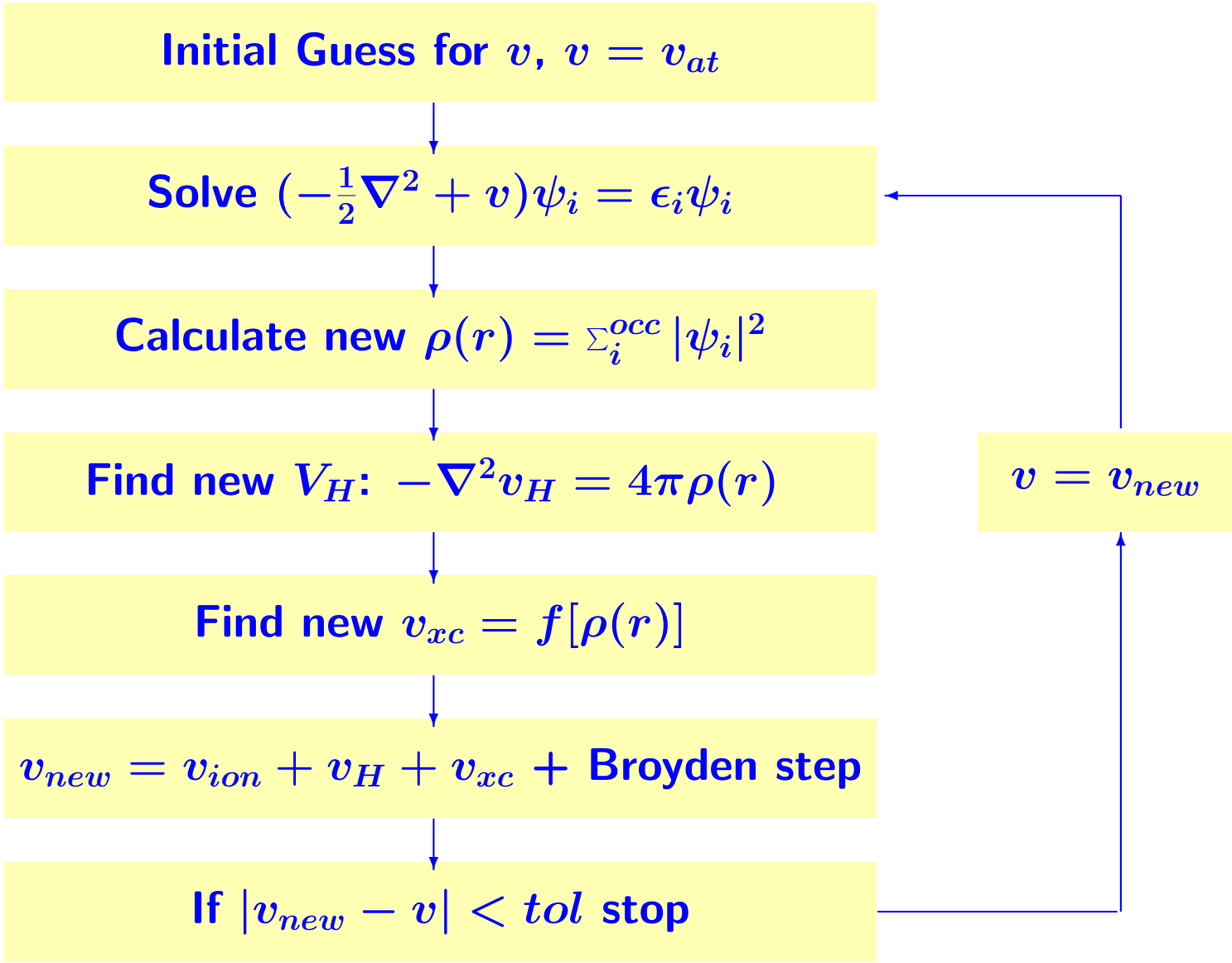
$$\left[-\frac{h^2}{2m}\nabla^2 + \underbrace{V_{ion} + V_H + V_{xc}}_{V_{tot}}\right]\Psi(r) = E\Psi(r)$$

- V_H = Hartree potential — local
- V_{xc} = Exchange & Correlation potential — local (LDA)
- V_{ion} = Ionic potential — Non-local

- Electron Density:

$$\rho(r) = \Sigma_i^{occup} |\Psi_i(r)|^2$$

- Above problem can be viewed as a nonlinear eigenvalue problem.


Initial Guess for v, $v = v_{at}$
Solve $(-\frac{1}{2}\nabla^2 + v)\psi_i = \epsilon_i \psi_i$
Calculate new $\rho(r) = \Sigma_i^{occ} |\psi_i|^2$
Find new V_H: $-\nabla^2 v_H = 4\pi\rho(r)$
Find new $v_{xc} = f[\rho(r)]$
$v_{new} = v_{ion} + v_H + v_{xc}$ + Broyden step
If $|v_{new} - v| < tol$ stop
$v = v_{new}$

- Most time-consuming part = computing eigenvalues / eigenvectors.

- Difficulty: large number of eigenvalues/eigenvectors to compute [number of occupied states]. For example, in real space, matrix size can be $N = 1,000,000$ and the number of eigenvectors could be 2,000.

- Self-consistent loop takes a few iterations (typically 4 to 12)

Standard methods ('The old')

- Lanczos method. Basis generated from 3-term recurrence:

$$\beta_{j+1} v_{j+1} = A v_j - \alpha_j v_j - \beta_j v_{j-1}$$

- Add preconditioning: Davidson.

$$v_{j+1} = M_j^{-1}(A u_j - \theta_j u_j); \quad [u_j, \theta_j] = \text{current eigenpair}$$

- Add implicit restart – clever way of restarting Lanczos process.

- Software. Best-known code = ARPACK

Issues

Preconditioning

- No clear-cut convincing approach for large numbers of eigenvalues.
- Simplest method: filtering by averaging..

Orthogonalization

- Becomes problematic for large number of eigenvalues -
- Idea of spectrum sclicing may be attractive but hard to implement

The outer loop

- Each self consistent loop should utilize previous information when starting – not easy to do with standard packages such as ARPACK

Current work on eigenvalue algorithms ('The new')

Focus:

(1) AMLS and related methods

(2) Block versions of restarted Lanczos

Motivation:

(1) Excellent success of AMLS in structural engineering. Similarity: large number of eigenvectors to compute

(2) Standard packages (ARPACK) do not easily take advantage of self-consistent loop. Also: not specialized for large number of eigenvalues.

Block-Lanczos – advantages

- Basic principle of the Block Lanczos algorithm: operate on block of b columns instead of only one column as in standard Lanczos.

Advantages:

- Can exploit a block of several initial guesses of eigenvectors
- Deals well with clustered or multiple ('degenerate') eigenvalues
- Can yield better cache performance (BLAS 3 instead of BLAS 2)

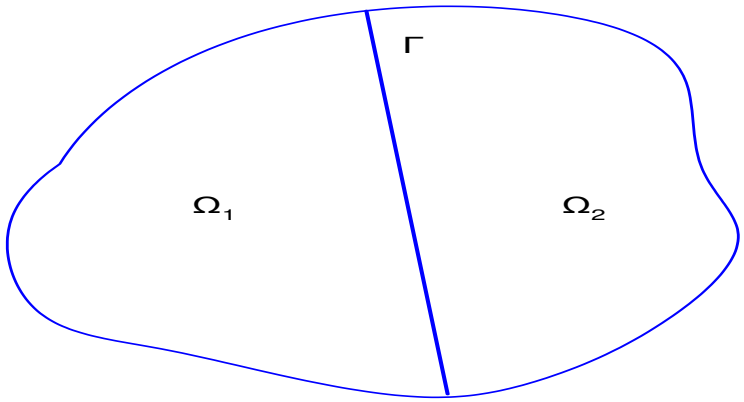
Issues:

- How to implement implicit restarts?
- Important to dynamically adapt block size

Automatic Multi-Level Substructuring

Origin: Extention of substructuring for eigenvalue problems.

Background: Domain decomposition. Let $A \in \mathbb{C}^{n\times n}$, Hermitian



$$\rightarrow A = \begin{pmatrix} B & E \\ E^* & C \end{pmatrix} \quad B \in \mathbb{C}^{(n-p)\times(n-p)}$$

Note: B is block-diagonal

Main Reference:

J. K. Bennighof and R. B. Lehoucq, **An automated multilevel substructuring method for eigenspace computation in linear elastodynamics, To appear in SIAM. J. Sci. Comput.**

Basic idea of the method for two levels

First step: eliminate the blocks E, E^*.

$$U = \begin{pmatrix} I & -B^{-1}E \\ 0 & I \end{pmatrix} \rightarrow U^*AU = \begin{pmatrix} B & 0 \\ 0 & S \end{pmatrix} ; \quad S = C - E^*B^{-1}E.$$

Original problem is equivalent to $U^*AUu = \lambda U^*Uu \rightarrow$

$$\begin{pmatrix} B & 0 \\ 0 & S \end{pmatrix} u = \lambda \begin{pmatrix} I & -B^{-1}E \\ -E^*B^{-1} & M_S \end{pmatrix} u \; ; M_S = I + E^*B^{-2}E$$

Second step: neglect the coupling in right-hand side matrix:

$$Bv = \mu \, v$$

$$Sw = \eta \, M_S w.$$

- Compute a few of the smallest engenvalues of the above problem.

Third step: Build a 'good' subspace to approximate to eigenfunctions of original problem. The basis used for this projection is of the form

$$\left\{ \hat{v}_i = \begin{pmatrix} v_i \\ 0 \end{pmatrix} \quad i = 1, \ldots, m_B; \qquad \hat{w}_j = \begin{pmatrix} 0 \\ w_j \end{pmatrix} \quad j = 1, \ldots, m_S \right\},$$

where $m_B < (n-p)$ and $m_S < p$.

Then use this subspace for a Rayleigh-Ritz projection applied to

$$\begin{pmatrix} B & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} u^B \\ u^S \end{pmatrix} = \lambda \begin{pmatrix} I & -B^{-1}E \\ -E^* B^{-1} & M_S \end{pmatrix} \begin{pmatrix} u^B \\ u^S \end{pmatrix}$$

(Note: not the original problem.)

Final step: exploit recursion –

NOTE: algorithm does only one shot of descent - ascent (no iterative improvement).

Issues investigated

- AMLS is a one shot algorithm - accuracy unlikely to be sufficient for electronic structure calculations. Problem: develop an improved version.

- AMLS can be viewed as Domain-Decomposition applied to shift-and-invert – with one shift at zero. Problem: develop a multi-shift version.

- Example: use AMLS with shift σ – obtain approximate eigenvectors (one shot 'descend and ascend'). Keep in basis. Change σ, add new eigenvectors to basis, etc..

Avoiding the eigenvalue problem

Recall: Density matrix,

$$\rho(r,r') = \sum_j f(E_j)\Psi_j(r)\overline{\Psi_j(r')} \quad \text{with} f(\lambda) = \text{occupancy factor}$$
$$\equiv f(H)$$

Main observation: Charge density ρ can be viewed as the diagonal of the density matrix: When $\psi_j(r)$ is discretized w.r.t. r then $\rho(r_i,r_i) \equiv \rho_{ii}$, the diagonal entry of $\rho(r,r')$, is the charge density at location r_i. Standard methods compute this by computing explicitly the eigenfunctions. ➨ Any orthogonal basis of the same space can be used. Is the eigenbasis an overkill?

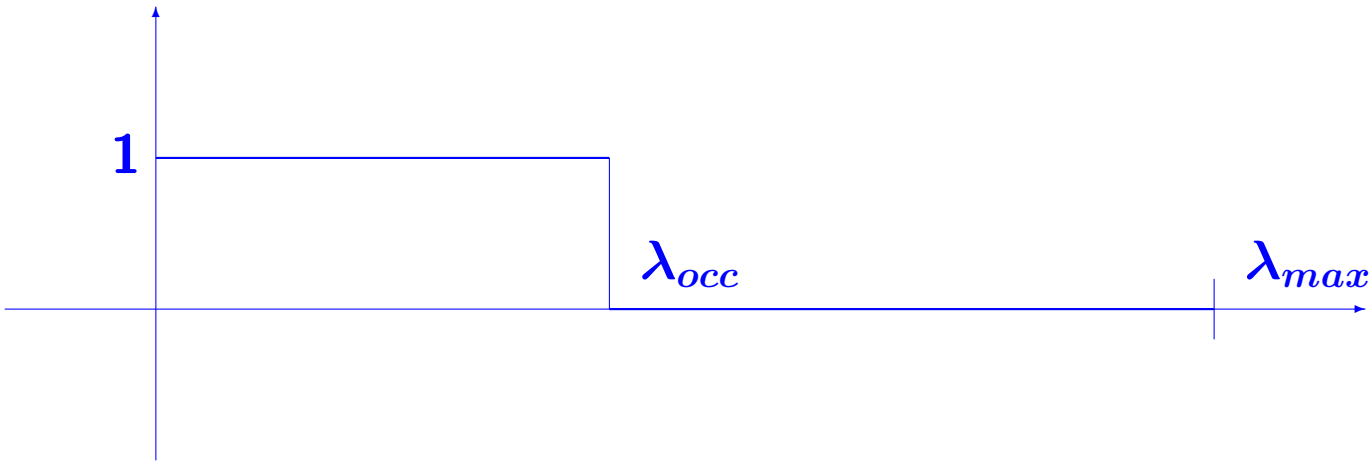
➨ 'Order n methods' find approximations to $\rho(r,r')$ by exploiting its decay properties

Previous work - Use of Chebychev polynomials ('the old')

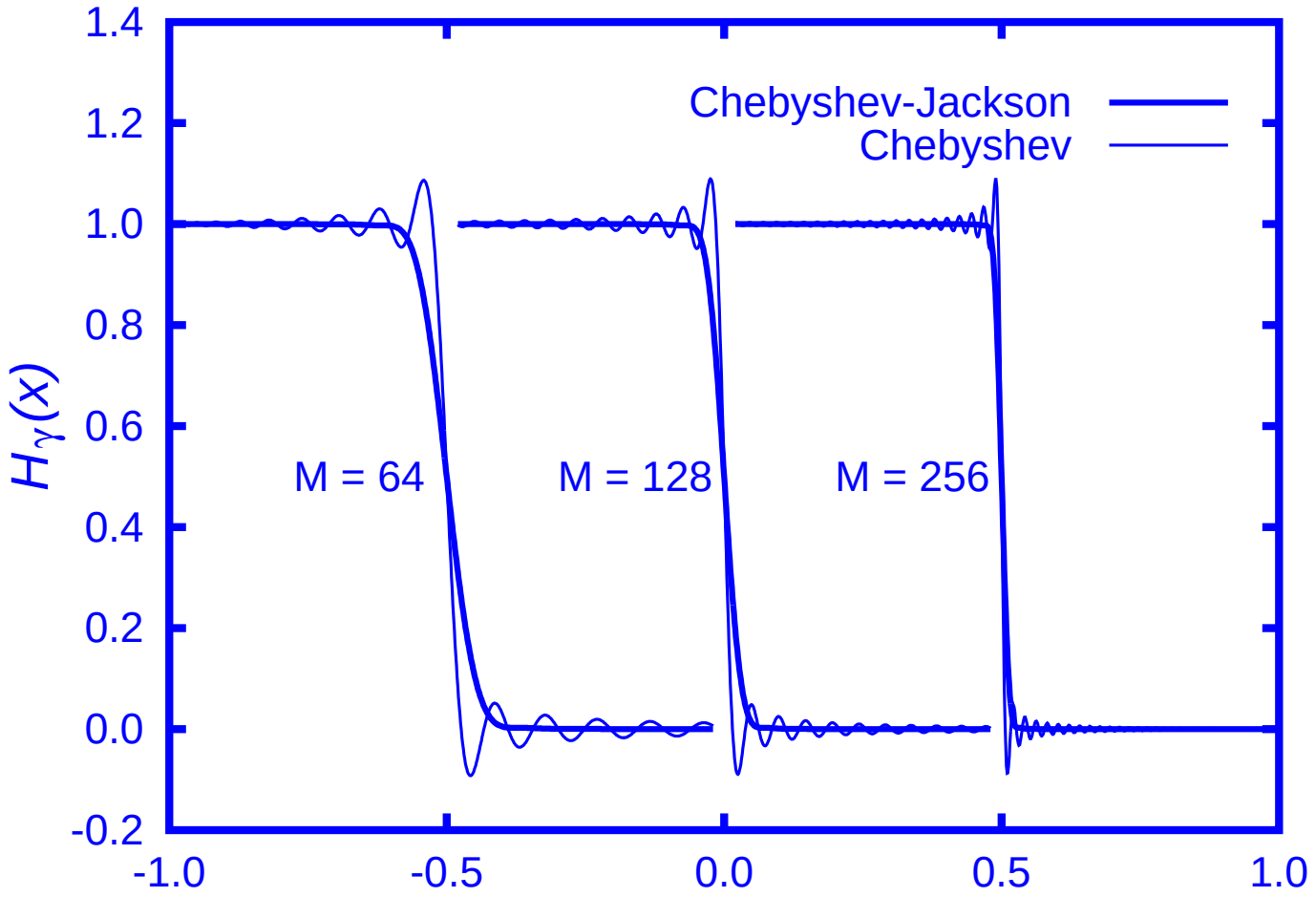
Main idea: Call H the original Hamiltonian matrix. Then

$$P = f(H)$$

where $f(t)$ is the Heaviside function:



- Replace f by a polynomial using Chebyshev expansions.
- This is now done in the planewave space not real space.



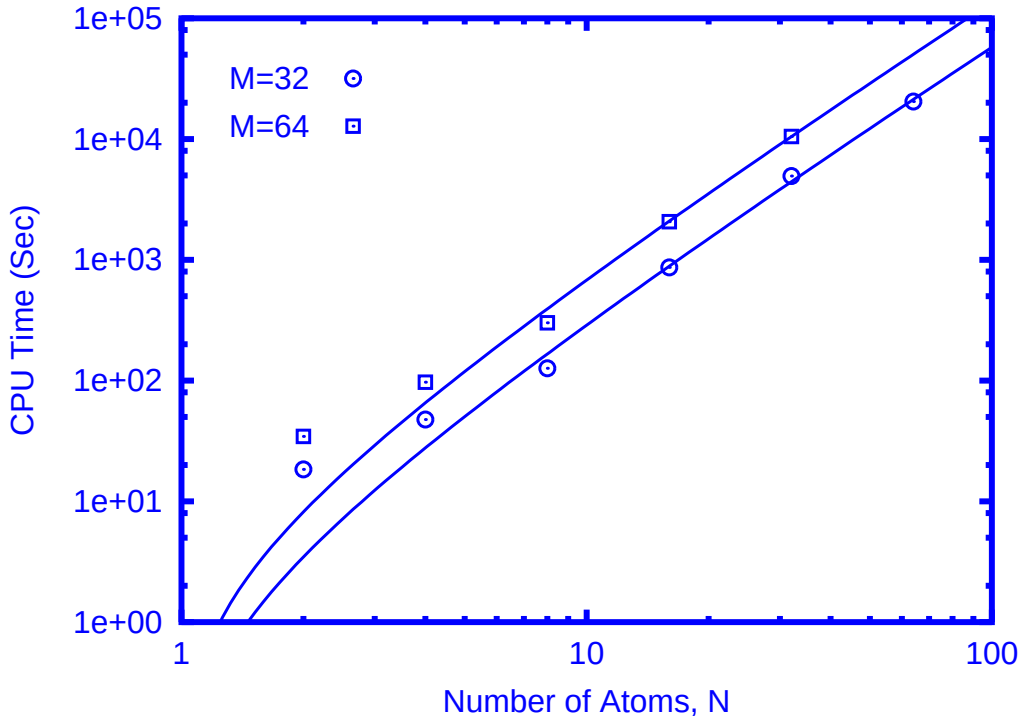
Chebyshev and Jackson polynomials of degree 64, 128, and 258

[Jackson expansions are modifications of the least-squares Chebyshev expansions that avoid Gibbs oscillations.]

- P is approximated by $p_m(H)X$ where X is a matrix of size $n \times p$ [e.g., 1st p columns of identity]

- Exploit recurrence relations of Chebyshev polynomials and near bandedness of P in planewave basis

- Test : with (Crystalline silicon) [See L. Jay et al. 1998]
- Times scale like $n^2 \log n$
- Cost is still high but : (1) can avoid eigenvectors completely, and (2) can still iterate to self-consistency



Current Approaches ('the new')

$$P = f(H)$$

where f is a step function. Approximate f by, e.g., a polynomial

- Result: can obtain columns of P inexpensively via:

$$Pe_j \approx p_k(H)e_j$$

- Exploit sparsity of P (especially in planewave basis)- ideas of "probing" allow to compute several columns of P at once.

- Statistical approach: work of Hutchinson for estimating trace of a matrix [used in image processing] adapted to estimating diagonals.

- Many variants currently being investigated

Example 1 : Statistical approach

- Let a sequence of random vectors $v^1, \ldots, v^s$ with entries satisfying a normal distribution. Diagonal of a matrix B can be approximated by

$$D^s = \left[\sum_{k=1}^{s} v^k \odot Bv^k\right] \oslash \left[\sum_{k=1}^{s} v^k \odot v^k\right]$$

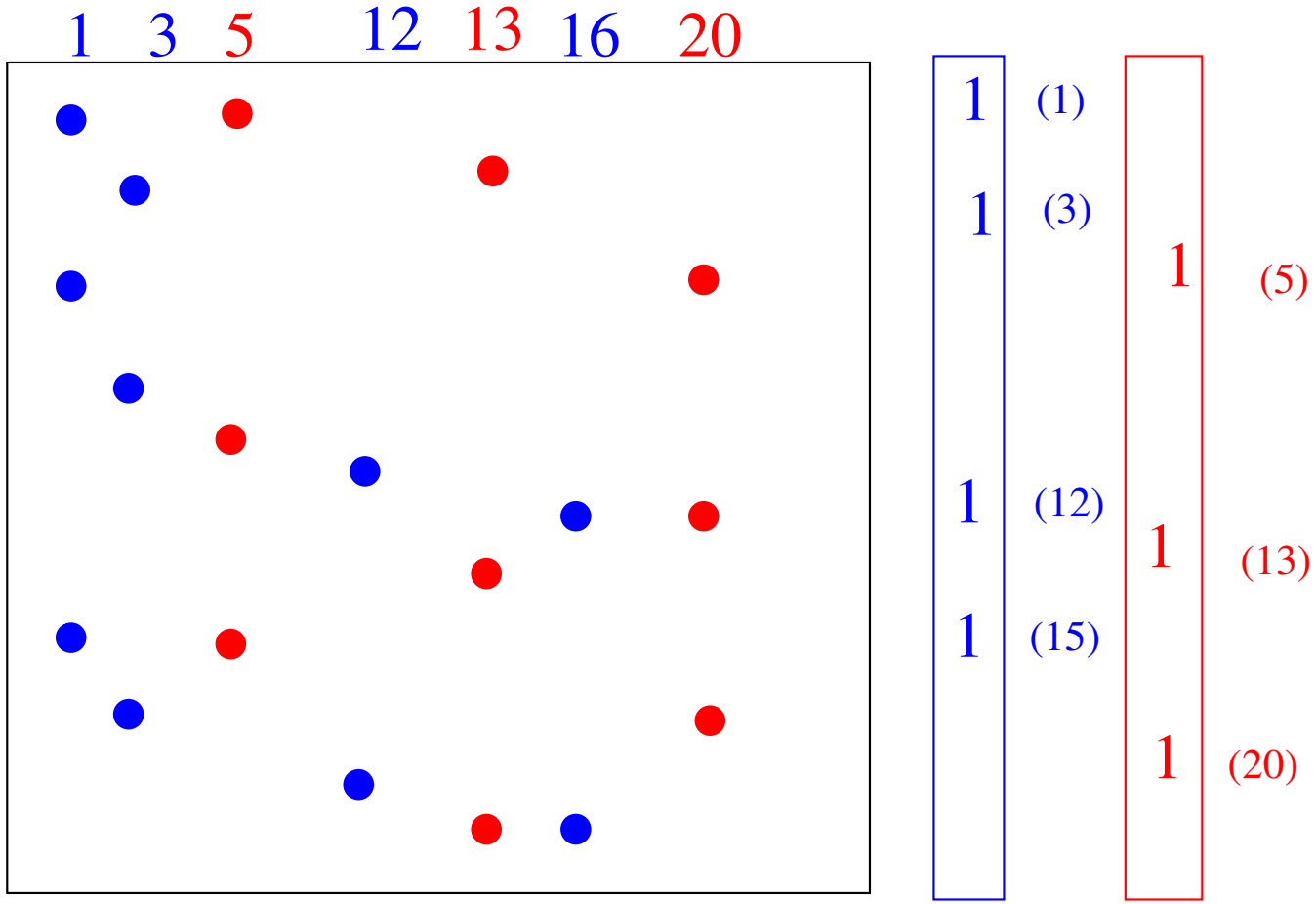
in which $\odot$ is a componentwise product of vectors, and similarly $\oslash$ represents a componentwise division of vectors.

- Deterministic approach: For a banded matrix (bandwidth p), there exists p vectors such the above formula yields the exact diagonal.

- These methods would require computing $p_k(H)v$ for several v's. Generally: method is expensive unless bandwith is small.

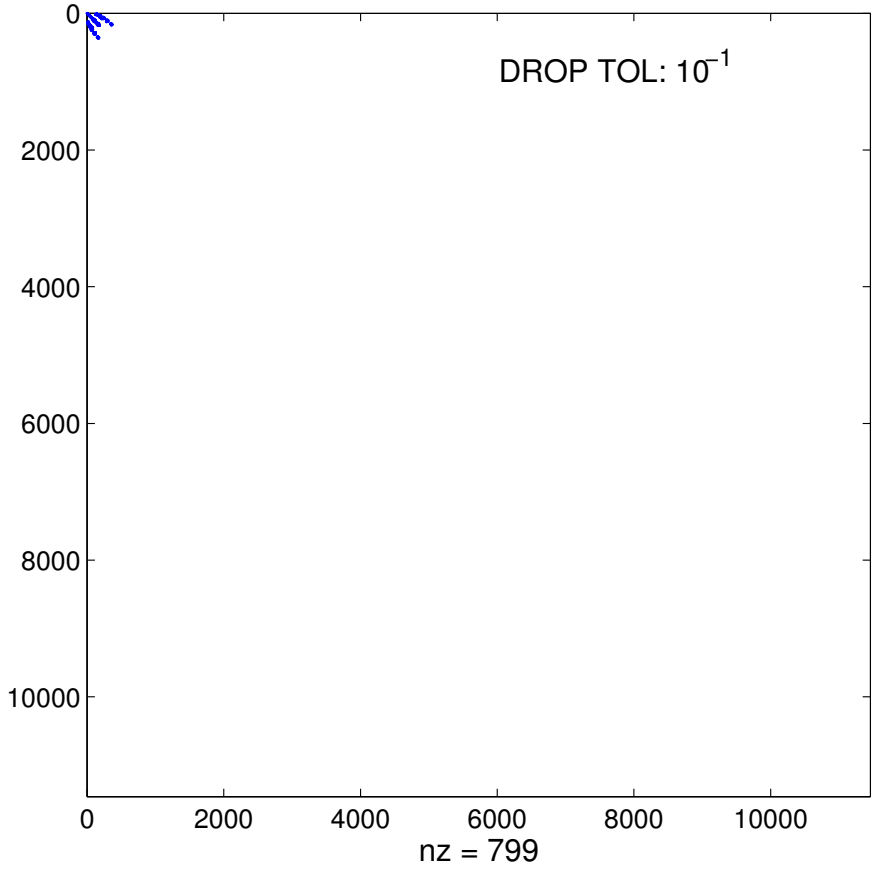
Example 2 : Sparsifying the density matrix in PW basis

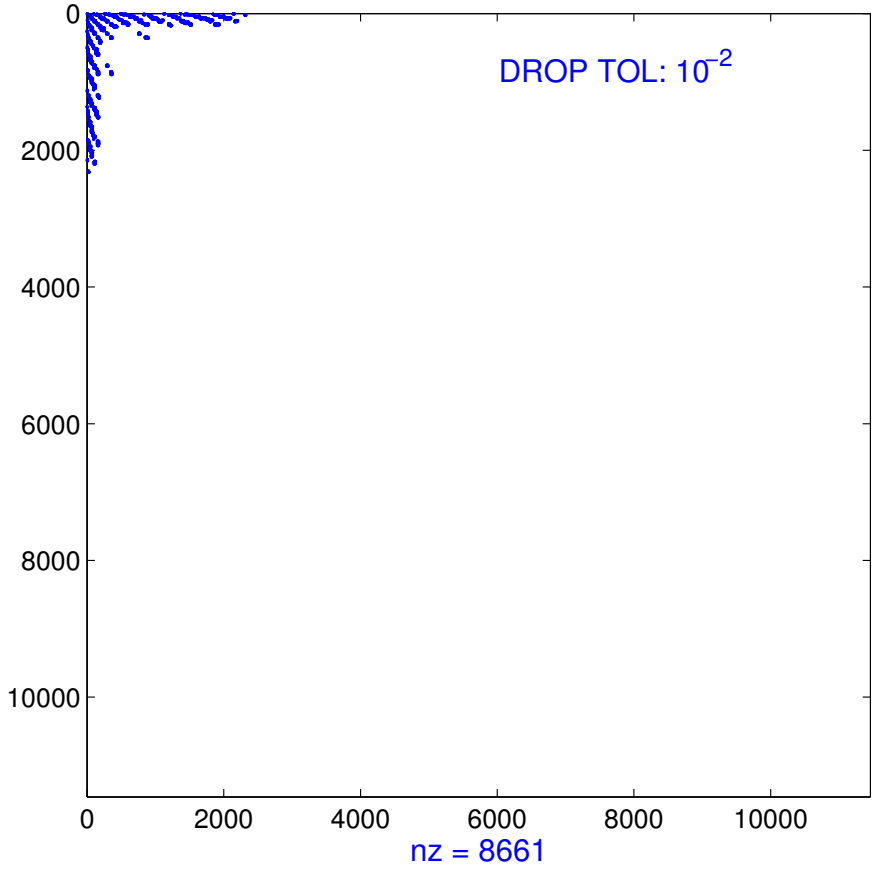
- Recall $P = f(H) \;=\; \{\rho(r, r')\}$
- Consider the expression in the G-basis
- Most entries are small –
- Idea: use technique of "probing" or "CPR" or "Sparse Jacobian" estimators ....

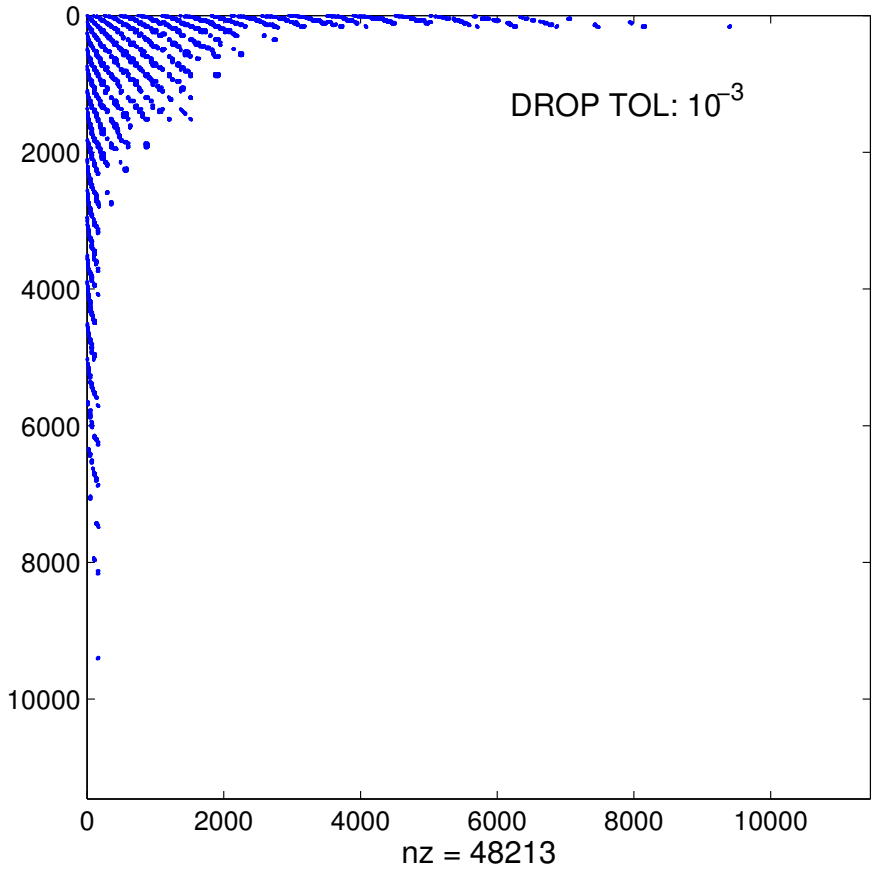


Probing in action: blue columns can be computed at once by one matrix-vector product. Then red columns can be coumputed the same way

Density matrix for Si64




DROP TOL: 10^-2
0
2000
4000
6000
8000
10000
nz = 8661


DROP TOL: 10^{-3}
0
2000
4000
6000
8000
10000
nz = 48213

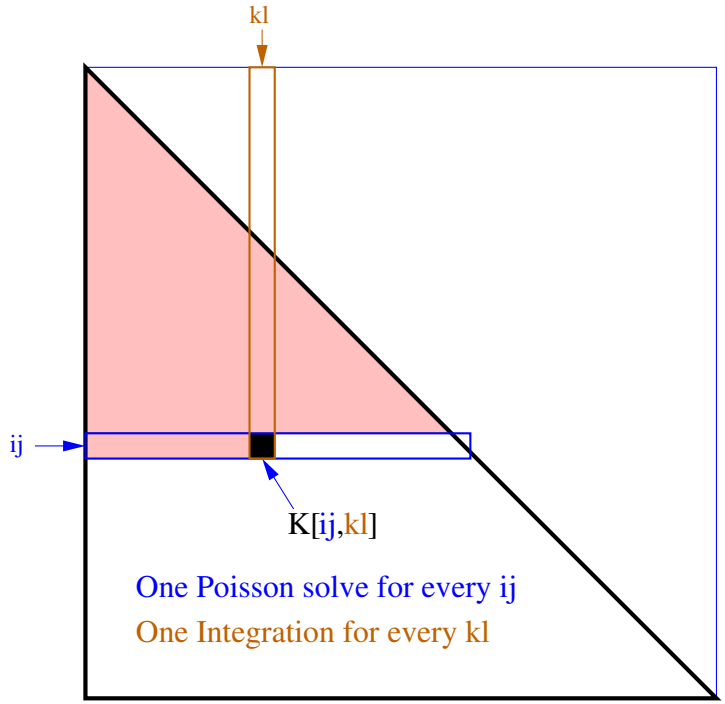
TIME DEPENDENT DFT

TDLDA: Use of planewave bases and FFT

Recall :

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

With $\triangle \Phi_{ij}(\mathrm{r}) = -4\pi \Psi_i \bar{\Psi}_j(\mathrm{r}).$



Coupling Matrix K

- Previous work [our group] : work in real space + use CG to solve Poisson's equation.

- Real space approach does not exploit specific features of the physics when solving Poisson's equation.

- Idea is to use FFTs: (In essence: Use "fast Poisson solvers")

- Expand each wavefunction in planewave basis:

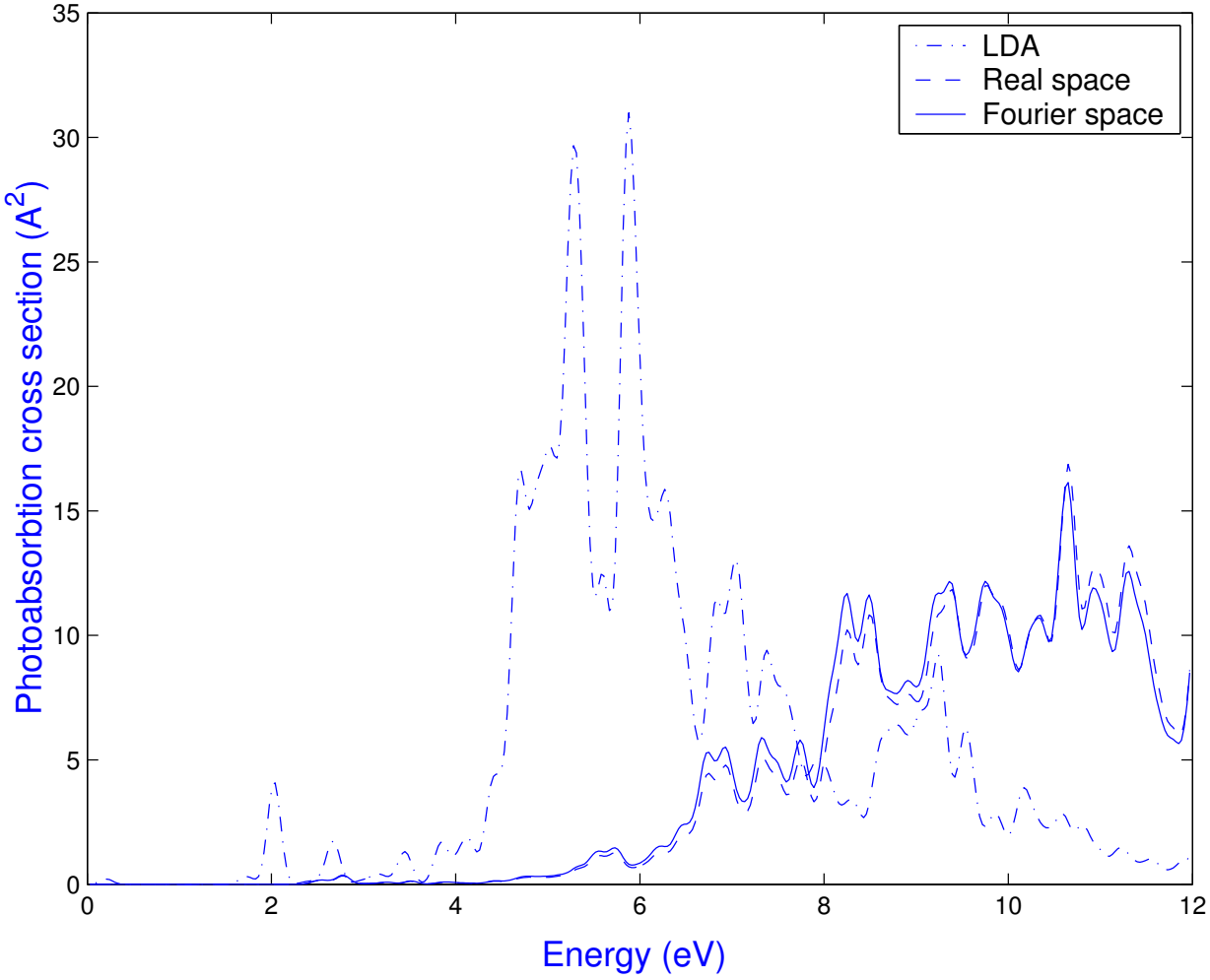
$$\Psi_j(\mathrm{r}) = \sum_{\mathrm{l}} \psi_{\mathrm{l}}^j \exp i(\mathrm{l}.\mathrm{r}) \longrightarrow \Phi_{ij}(\mathrm{r}) = 4\pi \sum_{(\mathrm{l},\mathrm{l}')\mathrm{l}\neq\mathrm{l}'} \frac{\psi_{\mathrm{l}}^i \bar{\psi}_{\mathrm{l}'}^j}{\|\mathrm{l}-\mathrm{l}'\|^2} e^{i(\mathrm{l}-\mathrm{l}').\mathrm{r}}.$$

- Many improvements can now be made. For example, in practice meaningful 'support' of $\psi_i\psi_j$ is small

$$\mathcal{F}(\Psi_i\bar{\Psi}_j)(\mathrm{k}) = \sum_{\mathrm{r}} e^{i\mathrm{k}.\mathrm{r}}(\Psi_i\bar{\Psi}_j)(\mathrm{r}) = \sum_{\mathrm{r}\,\in\,\mathbf{Supp}(\Psi_i\bar{\Psi}_j)} e^{i\mathrm{k}.\mathrm{r}}(\Psi_i\bar{\Psi}_j)(\mathrm{r}).$$

Results

- **Compare Real space code with planewave code for $Si34H36$**



- Compare times for Real space code and planewave code [for $Si34H36$]

Method	Wall-Clock Time (hours)
Real Space Code	15:30
PW: Initial Implementation	3:30
PW: Optimized load balancing	2:30

Wall-clock time of the parallel TDLDA code using Fourier space and Real Space for the Si34H36 test case running on 8 processors

Note: Gain a factor of 5-6 wrt to optimized version of TDLDA code. Compound with another factor of 3-4 from original to optimized real-space code $\longrightarrow$ 15 to 24 faster than [Vasiliev et al. 2000]

- More to come!

Specific Plans (TDDFT)

- 1. Optimize PW-based code – more opportunities for improvements;
- 2. Develop an efficient library for TDDFT
- 3. Do a large challenging new calculation;
- 4. In TDLDA, is it possible to better exploit the special nature of right-hand sides in Poisson's equation:

$$\nabla^2 \Phi_{ij\sigma}(\mathrm{r}) = -4\pi \psi_{i\sigma}(\mathrm{r}) \psi_{j\sigma}(\mathrm{r}).$$

More general plans

- 1. Most costly computation currently is still: the eigenvalue problem. Q: Can the success of AMLS [Automatic Multi-Level Substrututing] be extended from structural engineering to electronic structures calculations..

- 2. Challenge number 1 is: Can we avoid eigenvalue calculations altogether and still get good accuracy? [at lower cost?]