Scalable Algorithms for Electronic Structure Calculations on Petascale Computers

François Gygi
University of California, Davis
fgygi@ucdavis.edu
http://eslab.ucdavis.edu

Supported by NSF ITR-HECURA-0749217 and DOE-SciDAC

RANMEP2008 Workshop, NCTS, Taiwan, Jan 6, 2008
Outline

• First-Principles simulations
• Eigenvalue problems in electronic structure calculations
• Localized representations of solutions and simultaneous diagonalization problem
• Data compression through simultaneous diagonalization
First-Principles Simulations

- **Goal:** Simulate molecules, solids, liquids, from first principles, without input from experiments
- **The approach: Molecular dynamics:** an atomic-scale simulation method
  - Compute the trajectories of all atoms
  - Extract statistical information from the trajectories

Atoms move according to Newton’s law:

\[ m_i \ddot{\mathbf{R}}_i = \mathbf{F}_i \]
First-Principles Simulations

- Why “First-Principles”?
  - Avoid empirical models and adjustable parameters
    - Goal: applications to extreme conditions (high pressure, etc.) where no experimental data is available
  - Use fundamental principles: Quantum Mechanics
  - Must describe ions and electrons consistently and simultaneously

At each time step:
1) Compute the electronic structure
2) Derive interatomic forces
3) Move atoms
First-Principles Simulations

• Applications
  - Chemistry
  - Nanotechnology
  - Semiconductors
  - Biochemistry
  - High-pressure physics

Growth of a carbon nanotube on an iron catalyst

Biotin on silicon carbide

Ice-water interface

Silicon quantum dot
First-Principles Simulations

- The computation of the electronic structure is the most expensive part of the simulation

At each time step:
1) Compute the electronic structure
2) Derive interatomic forces
3) Move atoms

>99% of CPU time
First-principles simulations require large computing resources

- Cost of one time step scales as $O(n^3)$
  - $n$: number of electrons
- Many time steps required / long simulations
- Requires use of large-scale parallel platforms
  - target: $O(10^4)$ to $O(10^5)$ CPUs
- Focus on scalable algorithms
  - communication cost is primary concern
Using large computers: BlueGene/L

- 65,536 nodes, 128k CPUs
- 3D torus network
- 512 MB/node
- 367 TFlop peak
Computing the electronic structure

- Kohn-Sham equations
  - solutions $\varphi_i$ represent electronic wavefunctions (one per electron)
    $\varphi_i \in L^2(\mathbb{R}^3)$

\[
\begin{align*}
H \varphi_i &= -\Delta \varphi_i + V(\rho, \mathbf{r}) \varphi_i = \varepsilon_i \varphi_i \quad i = 1 \ldots n \\
V(\rho, \mathbf{r}) &= V_{\text{ion}}(\mathbf{r}) + \int \frac{\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}' + V_{\text{XC}}(\rho(\mathbf{r}), \nabla \rho(\mathbf{r})) \\
\rho(\mathbf{r}) &= \sum_{i=1}^{n} |\varphi_i(\mathbf{r})|^2 \\
\int \varphi_i^*(\mathbf{r}) \varphi_j(\mathbf{r}) d\mathbf{r} &= \delta_{ij}
\end{align*}
\]
Computing the electronic structure

• Solutions are represented as Fourier series

\[ \varphi_j(r) = \sum_{|q|^2 < E_{\text{cut}}} c_{q, j} e^{iq \cdot r} \]

• A set of solutions is represented by an (orthogonal) \((m \times n)\) matrix of complex Fourier coefficients

\[ Y_{ij} = c_{q_i, j} \]

• Dimensions of \(Y\): \(10^6 \times 10^4\)

• Note: typically \(m/n \sim 100\)
Computing the electronic structure

- The energy is invariant under unitary transformations of $Y$

$$E(Y) = \text{tr}(Y^T H Y) + F[\rho]$$

$$\rho(r) = \sum_j |\varphi_j(r)|^2$$

$$E(Y) = E(YQ), \quad Q \text{ unitary}$$
Electronic structure calculation: (with fixed potential)

- Invariant subspace computation

Find $Y$ such that:

$$HY = Y\Lambda$$

- $H \in \mathbb{R}^{m \times m}$, $Y \in \mathbb{R}^{m \times n}$, $\Lambda \in \mathbb{R}^{n \times n}$

  - $H$ is sparse
  - Cost of computing $Hx$: $O(m \log m)$ (involves Fast Fourier Transforms)
Electronic structure calculation: (with fixed potential)

- Iterative methods for invariant subspace computations
  - Variants of Jacobi-Davidson
  - DIIS (a.k.a. Anderson acceleration)

- Simple, diagonal preconditioning works well
- Robustness of eigensolvers is key
Preconditioned steepest descent

1) correction

\[ Y := Y + \beta K \left( I - YY^T \right) HY \]

2) orthogonalization
Preconditioned DIIS

1) descent direction
\[ \Delta_k = K \left( I - Y_k Y_k^T \right) H Y_k \]

2) update
\[ \theta = \frac{\text{tr} \Delta_k^T (\Delta_k - \Delta_{k-1})}{\|\Delta_k - \Delta_{k-1}\|_F} \]
\[ \bar{Y}_k = Y_k + \theta (Y_k - Y_{k-1}) \]
\[ \bar{\Delta}_k = \Delta_k + \theta (\Delta_k - \Delta_{k-1}) \]
\[ Y_{k+1} = \bar{Y}_k + \beta \bar{\Delta}_k \]

3) orthogonalization
Self-consistent electronic structure computation

- $H$ depends non-linearly on the solution $Y$ (through $\rho$)
- Fixed point iteration:
  repeat
  { 
    1) compute charge density $\rho_i = (YY^T)_{ii}$
    2) solve
    $$H(\rho)Y = Y \Lambda$$
  }
  until converged (i.e. $\rho$ does not change)

- Convergence can be accelerated using various charge-mixing schemes (e.g. Broyden)
Molecular Dynamics: solve the SCF problem at each time step

- $H$ is time-dependent (depends on positions of atoms)

for each time step $t$
{
  repeat
  {
    1) compute charge density $\rho_i = (YY^T)_{ii}$
    2) solve $H(\rho, t)Y(t) = Y(t)\Lambda(t)$
  } until converged
  compute forces, move atoms
}
Molecular Dynamics: using previous solutions optimally

- Computing $Y(t)$
  - The previous solution $Y(t-dt)$ is “close” to $Y(t)$, can be used as initial guess for iterative calculation of $Y(t)$
Molecular Dynamics: using previous solutions optimally

- Computing $Y(t)$
  - The previous solution $Y(t-dt)$ is “close” to $Y(t)$, can be used as initial guess for iterative calculation of $Y(t)$
  - The extrapolated subspace $\tilde{Y} = 2Y_k - Y_{k-1}$ is a better initial guess
Molecular Dynamics: using previous solutions optimally

- Subspace alignment
  - The eigensolver introduces arbitrary rotations in $Y(t)$
  - Extrapolation must be preceded by subspace alignment
  - Orthogonal Procrustes problem

\[
\min_Q \left\| Y_k - Y_{k-1}Q \right\| \quad Q^T Q = I
\]

\[
\tilde{Y} = 2Y_k - Y_{k-1}Q
\]
Orthogonal Procrustes problem: minimize \( \| Y_k - Y_{k-1} Q \| \)

1) Compute the polar decomposition

\[
Y_k^T Y_{k-1} \equiv A = UH
\]

where \( U \) is unitary, \( H \) hermitian.

2) rotation of \( Y_{k-1} \)

\[
Y_{k-1} := Y_{k-1} U
\]
Polar decomposition

Polar decomposition $A = UH$ (Higham ‘86)

$$X_0 = A$$

$$X_{k+1} = \frac{1}{2} \left( X_k + (X_k^{-1})^* \right)$$

converges quadratically to the unitary polar factor $U$

Need better, inverse-free, scalable algorithm
Outline

• First-Principles simulations
• Eigenvalue problems in electronic structure calculations
• Localized representations of solutions and simultaneous diagonalization problem
• Data compression through simultaneous diagonalization
Localized representations of the invariant subspace

- Linear combinations of electronic wavefunctions that minimize the spatial spread are called “Maximally Localized Wannier Functions” (MLWF)
  \[ \sigma^2 = \langle x - \langle x \rangle \rangle^2 \]
- MLWFs are used to compute the electronic polarization in crystals
- Computing MLWFs during a molecular dynamics simulation yields the infrared absorption spectrum

Spread Functionals

• Spread of a wavefunction associated with an operator $\hat{A}$

$$\sigma_{\hat{A}}^2 (\phi) = \left\langle \phi \left| \left( \hat{A} - \langle \phi | \hat{A} | \phi \rangle \right)^2 \right| \phi \right\rangle = \left\langle \phi | \hat{A}^2 | \phi \right\rangle - \left\langle \phi | \hat{A} | \phi \right\rangle^2$$

• Spread of a set of wavefunctions associated with an operator $\hat{A}$

$$\sigma_{\hat{A}}^2 \left( \{ \phi_i \} \right) = \sum_i \sigma_{\hat{A}}^2 (\phi_i)$$
Spread Functionals

- The spread is *not* invariant under orthogonal transformations

\[ \psi_i = \sum_j x_{ij} \phi_j \quad X \in \mathbb{R}^{n \times n} \text{ orthogonal} \]

\[ \sigma_\hat{A}^2(\{\psi_i\}) \neq \sigma_\hat{A}^2(\{\phi_i\}) \]

- There exists a matrix \(X\) that minimizes the spread
Spread Functionals

- Let

\[ A, B \in \mathbb{R}^{n \times n} \quad a_{ij} = \langle i \mid \hat{A} \mid j \rangle \quad b_{ij} = \langle i \mid \hat{A}^2 \mid j \rangle \]

\[ \sigma_A^2(\{\psi_i\}) = \text{tr} \left( X^T BX \right) - \sum_{i=1}^{n} \left( X^T AX \right)_{ii}^2 \]

- Minimize the spread = maximize \[ \sum_{i=1}^{n} \left( X^T AX \right)_{ii}^2 \]
  = diagonalize \( A \)
Spread Functionals

- Case of multiple operators

  operators \( \hat{A}^{(k)} \) \( k = 1, \ldots, m \)

  matrices \( A^{(k)} \) \( k = 1, \ldots, m \)

  \[
  \sigma^2_{\hat{A}}(\{\psi_i\}) = \sum_i \sum_k \sigma^2_{\hat{A}^{(k)}}(\psi_i)
  \]

- Minimize the spread = maximize \[
  \sum_{i=1}^{n} \sum_k \left( X^T A^{(k)} X \right)_{ii}^2
  \]

  = joint approximate diagonalization of the matrices \( A^{(k)} \)
Spread Functionals

- Example of multiple operators

\[ \hat{A}^{(1)} = \hat{X} \]
\[ (\hat{X}\phi)(x, y, z) \equiv x\phi(x, y, z) \]

\[ \hat{A}^{(2)} = \hat{Y} \]
\[ (\hat{Y}\phi)(x, y, z) \equiv y\phi(x, y, z) \]

\[ \hat{A}^{(3)} = \hat{Z} \]
\[ (\hat{Z}\phi)(x, y, z) \equiv z\phi(x, y, z) \]

- The matrices \( A^{(k)} \) do not necessarily commute, even if the operators \( \hat{A}^{(k)} \) do commute
Computation of MLWFs by simultaneous diagonalization

- Jacobi algorithm for simultaneous diagonalization

\[
\begin{align*}
\text{repeat} & \\
\text{for each pair } i, j & \\
\text{compute Jacobi rotation } R(i, j) & \\
A^{(k)} & \leftarrow R^T A^{(k)} R & \forall k \\
\text{until converged}
\end{align*}
\]

Computation of MLWFs by simultaneous diagonalization

\[ R(i, j) = \begin{pmatrix} r_{ii} & r_{ij} \\ r_{ji} & r_{jj} \end{pmatrix} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \]

\( c, s \in \mathbb{C} \)

\[ |c|^2 + |s|^2 = 1 \]

\[ c = \frac{\sqrt{x + r}}{2r} \quad s = \frac{y - iz}{\sqrt{2r(x + r)}} \quad r = \sqrt{x^2 + y^2 + z^2} \]

Computation of MLWFs by simultaneous diagonalization

\[
\begin{pmatrix}
x \\
y \\
z
\end{pmatrix}
\text{eigenvector of } G \equiv \text{Re}\left(\sum_k h(A^{(k)})h^H(A^{(k)})\right)
\]

\[
h(A) = \begin{pmatrix}
a_{ii} - a_{jj} \\
a_{ij} + a_{ji} \\
i(a_{ji} - a_{ij})
\end{pmatrix}
\]

Calculation of MLWFs

- In periodic systems

\[
\begin{align*}
\hat{A}^{(1)} &= \hat{C}_x \equiv \cos \frac{2\pi}{L_x} \hat{x} \\
\hat{A}^{(2)} &= \hat{S}_x \equiv \sin \frac{2\pi}{L_x} \hat{x} \\
\hat{A}^{(3)} &= \hat{C}_y \equiv \cos \frac{2\pi}{L_x} \hat{y} \\
\hat{A}^{(4)} &= \hat{S}_y \equiv \sin \frac{2\pi}{L_x} \hat{y} \\
\hat{A}^{(5)} &= \hat{C}_z \equiv \cos \frac{2\pi}{L_x} \hat{z} \\
\hat{A}^{(6)} &= \hat{S}_z \equiv \sin \frac{2\pi}{L_x} \hat{z}
\end{align*}
\]

Calculation of MLWFs

- The spread is minimized by simultaneous diagonalization of the matrices $C_x, S_x, C_y, S_y, C_z, S_z$

- Positions of the center of mass of the localized solutions ("Wannier centers")

$$\tau_i = \begin{pmatrix} L_x \frac{\theta_i^x}{2\pi} \\ L_y \frac{\theta_i^y}{2\pi} \\ L_z \frac{\theta_i^z}{2\pi} \end{pmatrix}$$

$$\theta_i^x = \arctan \left( \frac{(S_x)_{ii}}{(C_x)_{ii}} \right)$$

- Spreads

$$\left( \sigma_i^2 \right)_x = L_x^2 \left( 1 - (C_x)_{ii}^2 - (S_x)_{ii}^2 \right)$$

Simultaneous diagonalization algorithms

- The cost of simultaneous diagonalization is $O(N^3)$
- Conventional parallel implementations do not scale beyond $N_{CPU} \sim N_{st}/2$
Parallel Jacobi algorithm

Each processor holds a pair of columns

Limited to n/2 processors for n columns

One-sided Jacobi algorithms

Conventional Jacobi

\[
\text{repeat } \\
A \leftarrow R^T AR \\
U \leftarrow UR \\
\text{until converged}
\]

One-sided Jacobi

\[
\text{repeat } \\
A \leftarrow AR \\
U \leftarrow UR \\
\text{until converged}
\]


The one-sided Jacobi algorithm reduces the amount of global communication
Scalability of simultaneous diagonalization

- One-sided Jacobi simultaneous diagonalization algorithm:
  - 64-node dual-dual-core AMD Opteron/Infinipath cluster (256 CPUs)
  - 1 rack ANL BlueGene/L (1024 CPUs)
- Slight superlinear scaling due to:
  - cache effects
  - size-dependent MPI protocols
  - node mapping
Large-scale calculations of maximally localized Wannier functions

- Computation of MLWF starting from KS eigenvectors
- Simultaneous diagonalization of 6 \((n \times n)\) matrices
  - 64-node (256 CPUs) Opteron platform

<table>
<thead>
<tr>
<th>System</th>
<th>(n)</th>
<th>(m)</th>
<th>time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\text{HfSiO}<em>4)</em>{32})</td>
<td>1216</td>
<td>68723</td>
<td>38</td>
</tr>
<tr>
<td>((\text{H}<em>2\text{O})</em>{512})</td>
<td>4096</td>
<td>534310</td>
<td>440</td>
</tr>
</tbody>
</table>
Outline

- First-Principles simulations
- Eigenvalue problems in electronic structure calculations
- Localized representations of solutions and simultaneous diagonalization problem
- Data compression through simultaneous diagonalization
Finding reduced representations of Kohn-Sham subspaces

- A KS invariant subspace is represented as an orthogonal matrix \( Y \in \mathbb{R}^{m \times n} \)
- basis set size = \( m \)
- number of solutions = \( n \)
- In general, \( Y \) is dense

- Saving and reloading \( Y \) is costly (file size up to 250 GB, grows as \( O(nm) \) )
- Data compression is needed for storage of \( Y \)
Subspace bisection

- **Subspace Bisection**
  - divide the simulation domain into two subdomains (e.g. “left” and “right”) $\Omega_L, \Omega_R$
  - try to find an orthogonal transformation $V$ that makes every function either
    - localized on the left domain
    - localized on the right domain
    - extended across both domains
Subspace Bisection

\[ \Omega_L \]

\[ \Omega_R \]

\[ Y \]

\[ YV \]
The CS decomposition

- A matrix $Y$ having orthogonal columns, can be decomposed as

$$ Y = \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} = \begin{pmatrix} U_1 \Sigma_1 V^T \\ U_2 \Sigma_2 V^T \end{pmatrix} $$

where $U_1$, $U_2$, $V$ are orthogonal matrices,

$$ \Sigma_1 = \begin{pmatrix} C \\ 0 \end{pmatrix} \quad \Sigma_2 = \begin{pmatrix} S \\ 0 \end{pmatrix} $$

$$ C = \text{diag}(c_1,\ldots,c_n) \quad S = \text{diag}(s_1,\ldots,s_n) $$

$$ c_i^2 + s_i^2 = 1 $$

Stewart (1982)
CS Decomposition

\[
\begin{align*}
\Omega_L & \quad \Omega_R \\
Y & \quad YV
\end{align*}
\]
CS Decomposition

\[
\begin{align*}
\Omega_L \quad \Omega_R \\
Y & \rightarrow \text{norm} = c_i^2 \\
YV & \rightarrow \text{norm} = s_i^2 = 1 - c_i^2
\end{align*}
\]
CS Decomposition

\[ \Omega_L \quad \Omega_R \quad Y \]

- \( S_i^2 < \varepsilon \) localized in \( \Omega_L \)
- \( c_i^2 < \varepsilon \) localized in \( \Omega_R \)
- Extended states
Subspace Bisection Algorithm

1. Choose the acceptable error $\varepsilon$
2. Perform a CS decomposition of the matrix $Y$
3. For each vector of $YV$:
   - if $c_i^2 < \varepsilon$ state localized in $\Omega_R$ (truncate in $\Omega_L$)
   - else if $s_i^2 < \varepsilon$ state localized in $\Omega_L$ (truncate in $\Omega_R$)
   - else state is extended

Ideal limit: 2-fold data reduction
Subspace Bisection Algorithm

- When does the CS decomposition lead to efficient data compression?
  - in some systems (insulators, where the spectrum of $H$ has a gap) most of the solutions can be localized in $\Omega_L$ or $\Omega_R$
  - in some systems (metals, not spectral gap) fewer states can be localized
Subspace Bisection Algorithm

$C_i^2$

**Fig. 2.** Singular values of $F_{1024}$. computed with MATLAB.

$C_i^2$

**Fig. 3.** Singular values of a $256 \times 256$ section of a random $1024 \times 1024$ unitary matrix, computed with MATLAB.

Y matrix = eigenvectors of the Laplacian

Y matrix = random unitary matrix

Multiple Subspace Bisection

- The Subspace Bisection algorithm can be applied simultaneously in three directions
Multiple Subspace Bisection

- Each solution is associated with a triplet

\[
\begin{pmatrix}
C_x^2 \\
C_y^2 \\
C_z^2
\end{pmatrix}
\]

Ideal limit: 8-fold data reduction

Simultaneous CS decomposition: Requires simultaneous diagonalization of three matrices
Applications

• \((H_2O)_{512}\) \(\epsilon = 10^{-3}\)

Data size reduction: 4.03

<table>
<thead>
<tr>
<th></th>
<th>(N_1)</th>
<th>(N_{1/2})</th>
<th>(N_{1/4})</th>
<th>(N_{1/8})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>48</td>
<td>276</td>
<td>844</td>
<td>880</td>
</tr>
</tbody>
</table>

2048 functions

• (19x0) Carbon nanotube (304 atoms) \(\epsilon = 10^{-3}\)

Data size reduction: 2.72

<table>
<thead>
<tr>
<th></th>
<th>(N_1)</th>
<th>(N_{1/2})</th>
<th>(N_{1/4})</th>
<th>(N_{1/8})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>108</td>
<td>80</td>
<td>183</td>
<td>237</td>
</tr>
</tbody>
</table>

608 functions
\((\text{H}_2\text{O})_{512}\) states after bisection

localized

extended
CNT(19x0) states after bisection

localized

extended
Distribution of singular values

- \((\text{H}_2\text{O})_{512}\)
Distribution of singular values

- (19x0) CNT $C_{304}$

\[
\begin{pmatrix}
  c_x^2 \\
  c_y^2 \\
  c_z^2
\end{pmatrix}
\]

\[
\begin{pmatrix}
  \log c_x^2 (1 - c_x^2) \\
  \log c_y^2 (1 - c_y^2) \\
  \log c_z^2 (1 - c_z^2)
\end{pmatrix}
\]
Distribution of singular values

- BCC Mo$_{54}$

\[
\begin{pmatrix}
  c_x^2 \\
  c_y^2 \\
  c_z^2
\end{pmatrix}
\]

\[
\begin{pmatrix}
  \log c_x^2 \left(1 - c_x^2\right) \\
  \log c_y^2 \left(1 - c_y^2\right) \\
  \log c_z^2 \left(1 - c_z^2\right)
\end{pmatrix}
\]
Remaining open issues

- Uniqueness of minimum of spread functionals
- Exploration of other simultaneous diagonalization methods
- Recursive implementation of the subspace bisection algorithm
Summary

- Electronic structure calculations involve large invariant subspace computations
- Extrapolation of invariant subspaces is performed using subspace alignment (Procrustes problem)
- Localized representations of solutions of the Kohn-Sham equations can be obtained via simultaneous diagonalization
- A subspace bisection algorithm provides a reduced representation of Kohn-Sham invariant subspaces with controlled accuracy