The ELSI Infrastructure
for Scalable Electronic Structure Theory

Victor Yu
Volker Blum

Duke University
Department of Mechanical Engineering and Materials Science
Durham, NC, USA
http://aims.pratt.duke.edu

The ELSI Infrastructure
for Scalable Electronic Structure Theory

\[ \hat{h}_{\text{eff}} \phi_l(r) = \epsilon_l \phi_l(r) \quad \leftrightarrow \quad n(r) = \sum_l f_l |\phi_l(r)|^2 \]

\[ H, S \quad \quad \quad \quad \quad \quad \quad n \]

Victor Yu

https://www.elsi-interchange.org

Hands-On Workshop Density Functional Theory and Beyond - Barcelona, August 29, 2019
Nucleus: Emilio Artacho, 2014: “Dear all, There will be a workshop in CECAM at Lausanne … aiming to kick-start an electronic structure library. … I hope you are interested”
**ELSI - Acknowledgments**

*Nucleus*: Emilio Artacho, 2014: “*Dear all, There will be a workshop in CECAM at Lausanne … aiming to kick-start an electronic structure library. … I hope you are interested*”

*NSF-SI2 - ACI-1450280:*
Volker Blum, Jianfeng Lu, Lin Lin, Chao Yang, Alvaro Vazquez-Mayagoitia, Fabiano Corsetti
NSF-SI2 - ACI-1450280:
Volker Blum, Jianfeng Lu, Lin Lin, Chao Yang, Alvaro Vazquez-Mayagoitia, Fabiano Corsetti

Why ELSI Works:

ELSI - Acknowledgments

Nucleus: Emilio Artacho, 2014: “Dear all, There will be a workshop in CECAM at Lausanne … aiming to kick-start an electronic structure library. … I hope you are interested”

NSF-SI2 - ACI-1450280: Volker Blum, Jianfeng Lu, Lin Lin, Chao Yang, Alvaro Vazquez-Mayagoitia, Fabiano Corsetti

Why ELSI Works:

- Victor Yu (Duke)
- Yingzhou Li (Duke)
- Will Huhn (Duke)
- ELPA, PEXSI, NTPoly, Slepc, …
- Micael Oliveira, Yann Pouillon, Fabiano Corsetti, Nick Papior, many more.

Electronic Structure Library:

https://esl.cecam.org
https://gitlab.com/ElectronicStructureLibrary
ELSI, Fdict, Futile, libfdf, libgridxc, libpsml, libxc, Psolver, pspio, xmlf90, …
Materials Properties from Theory?

E.g., Organic-Inorganic Hybrid Semiconductors - Tailored Properties?

Component 1

Component 2

New Functional Material?

3D crystal

400+ atoms per unit cell

E.g., Organic-Inorganic Hybrid Semiconductors - Tailored Properties?

Pathway 1: Make material, find out its properties. 

Materials Properties from Theory?

**E.g., Organic-Inorganic Hybrid Semiconductors - Tailored Properties?**

Pathway 1: Make material, find out its properties.  

Pathway 2: Accurate computational prediction?

\[ \hat{H} \Psi = E \Psi \]

The theory we have is accurate enough.

But need to implement sufficiently good approximations on a real computer.

AE4T-PbBr$_4$: Energy Levels - Impact of Spin-Orbit Coupling

AE4T-PbBr$_4$: Energy Levels - Impact of Spin-Orbit Coupling

SOC changes the character of conduction band minimum ("electrons")

Holes on organic component, electrons on inorganic component:
Type IIb Quantum Well
SOC changes the character of conduction band minimum ("electrons")

Holes on organic component, electrons on inorganic component: Type IIb Quantum Well

... but how exactly were these calculations run?

**Component 1**

**Component 2**

**New Functional Material?**

3D crystal

400+ atoms per unit cell

424 atoms (light and heavy)

**Geometry:**

DFT-PBE+vdW\(^{TS}\)  
(Tkatchenko, Scheffler, PRL 2009)  
tight settings  
(12,384 basis functions)  
several tens to hundreds of steps for geometry optimization

**Electronic Structure:**

DFT-HSE06+SOC  
intermediate settings  
(10,224 basis functions)

(AE4T)PbBr₄ - Computational Details

424 atoms (light and heavy)

Geometry:
DFT-PBE+vdW^{TS}  
(Tkatchenko, Scheffler, PRL 2009)
tight settings
(12,384 basis functions)
several tens to hundreds of steps for geometry optimization

Electronic Structure:
DFT-HSE06+SOC
intermediate settings
(10,224 basis functions)

(AE4T)PbBr$_4$ - Computational Details

Which computer to use?

424 atoms (light and heavy)

Geometry:
- DFT-PBE+vdW$^{TS}$
  (Tkatchenko, Scheffler, PRL 2009)
- tight settings
  (12,384 basis functions)
- several tens to hundreds of steps for geometry optimization

Electronic Structure:
- DFT-HSE06+SOC
- intermediate settings
  (10,224 basis functions)

(AE4T)PbBr$_4$ - Computational Details

Which computer to use?

424 atoms (light and heavy)

Geometry:
DFT-PBE+vdW$^{TS}$ (Tkatchenko, Scheffler, PRL 2009)
tight settings
(12,384 basis functions)
several tens to hundreds of steps for geometry optimization

Electronic Structure:
DFT-HSE06+SOC
intermediate settings
(10,224 basis functions)

Which computer to use?

424 atoms (light and heavy)

Geometry:
DFT-PBE+vdW$^{TS}$ (Tkatchenko, Scheffler, PRL 2009)
tight settings
(12,384 basis functions)
several tens to hundreds of steps for geometry optimization

Electronic Structure:
DFT-HSE06+SOC
intermediate settings
(10,224 basis functions)

https://www.alcf.anl.gov/theta
Intel KNL
9.8 PFlops (231,424 cores)
1,024 - 2,048 cores
comfortably perform these tasks

However, How to Go Much Larger?

1. Real space grid operations

\[ h_{ij} = \int d^3 r \varphi_i(r) \hat{h}_{KS} \varphi_j(r) \]

Basis functions, Hamiltonian, Kohn-Sham potential etc.

2. Matrix algebra (basis space)

\[ \hbar c_k = \epsilon_k s_c c_k \]

Kohn-Sham eigenvalue problem
However, How to Go Much Larger?

1. Real space grid operations

\[ h_{ij} = \int d^3r \varphi_i(r) \hat{h}_{KS} \varphi_j(r) \]

Basis functions, Hamiltonian, Kohn-Sham potential etc.

2. Matrix algebra (basis space)

\[ \hbar c_k = \epsilon_k s c_k \]

Kohn-Sham eigenvalue problem

- Large “prefactor:” Dominant for standard problems
- Mature algorithms (Delley, others)
- \( O(N) \) scalability possible in all steps
- relatively simple parallelization

However, How to Go Much Larger?

1. Real space grid operations

\[ h_{ij} = \int d^3r \varphi_i(r) \hat{h}_{\text{KS}} \varphi_j(r) \]

Basis functions, Hamiltonian, Kohn-Sham potential etc.

2. Matrix algebra (basis space)

\[ \hbar c_k = \epsilon_k \underline{s} \underline{c}_k \]

Kohn-Sham eigenvalue problem

• Large “prefactor:” Dominant for standard problems
• Mature algorithms (Delley, others)
• \( O(N) \) scalability possible in all steps
• relatively simple parallelization

“Conventional” solvers (Lapack-like):

• Small prefactor for NAO’s: affordable up to \( \geq 1,000 \) atoms
• Robust, general (metals!)
• \( O(\text{size}^3) \) scalability inevitable
• Massively parallel scalability not out of the box


How far can we push such solvers?
Typical Scaling - $O(N^3)$ Wall

$H\varphi = \epsilon S\varphi$

Graphene monolayer, 4050 atoms
FHI-aims, PBE, “light” settings

Number of basis functions
56700
70000
84700
100800

Time for one SCF iteration [s]
500
100
20

EDISON (Intel Ivy Bridge)
80 nodes x 24 CPU cores per node

$O(N^3)$

$O(N)$

Generic problem for any Kohn-Sham DFT code ... solution strategies?

ELPA Library
http://elpa.rzg.mpg.de
Calculations for millions of atoms with density functional theory: linear scaling shows its potential

D R Bowler\textsuperscript{1,2,3} and T Miyazaki\textsuperscript{4}
In Principle, Much Larger Systems Are Possible

Calculations for millions of atoms with density functional theory: linear scaling shows its potential

D R Bowler¹,²,³ and T Miyazaki

Hybrid MPI-OpenMP Parallelism in the ONETEP Linear-Scaling Electronic Structure Code: Application to the Delamination of Cellulose Nanofibrils

Karl A. Wilkinson,† Nicholas D. M. Hine,‡ and Chris-Kriton Skylaris*†

Linear Scaling Self-Consistent Field Calculations with Millions of Atoms in the Condensed Phase

Joost VandeVondele,*† Urban Borštnik,‡,§ and Jürg Hutter‡

†Department of Materials, ETH Zurich, Wolfgang-Pauli-Strasse 27, 8093 Zurich, Switzerland
‡Physical Chemistry Institute, University of Zurich, Winterthurerstrasse 190, CH-8057 Zurich, Switzerland

ABSTRACT: In this work, the applicability and performance of a linear scaling algorithm is investigated for three-dimensional condensed phase systems. A simple but robust approach based on the matrix sign function is employed together with a thresholding matrix multiplication that does not require a prescribed sparsity pattern. Semiempirical methods and density functional theory have been tested. We demonstrate that self-consistent calculations with 1 million atoms are feasible for simple systems. With this approach, the computational cost of the calculation depends strongly on basis set quality. In the current
In Principle, Much Larger Systems Are Possible

Calculations for millions of atoms with density functional theory: linear scaling shows promise

D R Bowler¹,²,³

Joost Vandeven²

Department of Chemistry

Physical Chemistry

ABSTRACT

Linear scaling for condensed phase systems is a simple but robust approach based on the matrix sign function. It employs techniques within thresholding matrix multiplication that does not require a prescribed sparsity pattern. Semiempirical methods and density functional theory have been tested. We demonstrate that self-consistent calculations with 1 million atoms are feasible for simple systems. With this approach, the computational cost of the calculation depends strongly on the basis set quality. In the current

However:

• System- and/or density-functional specific algorithms
• The optimal algorithms changes from small to large scale
• Complex, algorithm-specific parallelization (matrix storage distribution, communication patterns)
• Very different supercomputing paradigms - must be addressed at low level

→ seamless, easy-to-use, general implementation from laptop to supercomputer not straightforward
Agenda for the Remainder of this Talk

• Making an Eigenvalue Solver Scalable (ELPA library)

• Bypassing the Eigenvalue Problem - Density-Matrix Based Approaches

• ELSI Infrastructure
A Massively Parallel Dense Eigensolver: “ELPA”

\[ h c_k = \epsilon_k S c_k \]

Given a matrix \( H \) and metric \( S \) (dimension \( N \)), find \( M \) eigenvalue/eigenvector pairs \( \epsilon_k/c_k \)

http://elpa.rzg.mpg.de

A. Marek, V. Blum, R. Johanni, V. Havu, B. Lang, T. Auckenthaler, A. Heinecke, H.-J. Bungartz, H. Lederer, 
A Massively Parallel Dense Eigensolver: “ELPA”

\[ h \mathbf{c}_k = \epsilon_k \mathbf{S} \mathbf{c}_k \]

Given a matrix \( \mathbf{H} \) and metric \( \mathbf{S} \) (dimension \( N \)), find \( M \) eigenvalue/eigenvector pairs \( \epsilon_k/\mathbf{c}_k \)

Original Goals:
- Scalable, Scalapack-compatible “drop-in enhancement”
- Pure MPI-based implementation
- Detailed rewrite based on proven robust/general algorithms

http://elpa.rzg.mpg.de

A. Marek, V. Blum, R. Johanni, V. Havu, B. Lang, T. Auckenthaler, A. Heinecke, H.-J. Bungartz, H. Lederer,
A Massively Parallel Dense Eigensolver: “ELPA”

\[ \hbar c_k = \epsilon_k s c_k \]

Given a matrix \( H \) and metric \( S \) (dimension \( N \)), find \( M \) eigenvalue/eigenvector pairs \( \epsilon_k/c_k \)

Original Goals:
- Scalable, Scalapack-compatible “drop-in enhancement”
- Pure MPI-based implementation
- Detailed rewrite based on proven robust/general algorithms

Today:
- Standalone open-source (LGPL) library, used in other major codes (cp2k, Quantum Espresso, VASP, ...)
- Optional support for shared-memory systems (OpenMP)

http://elpa.rzg.mpg.de

A. Marek, V. Blum, R. Johanni, V. Havu, B. Lang, T. Auckenthaler, A. Heinecke, H.-J. Bungartz, H. Lederer,
Taking Apart the Eigenvalue Problem

Generalized (non-orthogonal) eigenvalue problem:
• Transform to orthogonal form: $U^{-THU^{-1}}$
• Transform orthogonal $H'$ to *tridiagonal* form
• Solve *tridiagonal* eigenproblem
• Backtransform (1) solution to standard form
• Backtransform (2) standard to general form

$h \epsilon_k c_k = \epsilon_k s c_k$
Taking Apart the Eigenvalue Problem

Generalized (non-orthogonal) eigenvalue problem:
- Transform to orthogonal form: $U^{-THU^{-1}}$
- Transform orthogonal $H'$ to tridiagonal form
- Solve tridiagonal eigenproblem
- Backtransform (1) solution to standard form
- Backtransform (2) standard to general form

\[
\hat{h} c_k = \epsilon_k s c_k
\]
A Careful Rewrite Can Improve Scaling ("ELPA I")

\[ \alpha\text{-helical Polyalanine Ala}_{100} \]

\[ N=27069, M=3410 \]

NAO basis set (FHI-aims)
A Careful Rewrite Can Improve Scaling ("ELPA I")

α-helical Polyalanine Ala_{100}

$N=27069$, $M=3410$

NAO basis set (FHI-aims)

---

**Elapsed time [s]**

- **Intel/Infiniband**
- **BlueGene/P**
- **Scalapack**
- **rewrite(1)**

**Number of cores**

- 64
- 128
- 256
- 512
- 1K
- 2K
- 4K
- 8K
- 16K
- 32K
- 64K
- 128K
- 256K

Auckenthaler et al.,
Parallel Computing (2011)
Algorithmic Improvement: 2-Step Tridiagonalization

Remaining chief bottleneck: Tridiagonalization

“Conventional” reduction:

\[
\begin{align*}
\text{full matrix} & \quad \xrightarrow{\text{Householder transform}} \quad \text{tridiagonal matrix} \\
& \quad \xrightarrow{\text{matrix-vector operations}} \\
\end{align*}
\]
Algorithmic Improvement: 2-Step Tridiagonalization

Remaining chief bottleneck: Tridiagonalization

“Two-step” reduction:

But extra back transform necessary - benefit shrinks for $M$ approaching $N$

Auckenthaler, Blum, Bungartz, Huckle, Johanni, Krämer, Lang, Lederer, Willems, Parallel Computing (2011)
Algorithmic Improvement: 2-Step Tridiagonalization

Remaining chief bottleneck: Tridiagonalization

“Two-step” reduction:

\[
\begin{align*}
\text{full matrix} & \xrightarrow{\text{Step 1:}} \text{band matrix} & \xrightarrow{\text{Step 2:}} \text{tridiagonal matrix}
\end{align*}
\]

But extra back transform necessary - benefit shrinks for \( M \) approaching \( N \)

Massively parallel two-step tridiagonalization:
- 2-dimensional data layout for eigenvectors
- Heavily optimized backtransform steps for eigenvectors (adaptive data layout, architecture-specific linear algebra kernels - cache blocking)

Auckenthaler, Blum, Bungartz, Huckle, Johanni, Krämer, Lang, Lederer, Willems, Parallel Computing (2011)
ELPA, Two-Step Solver

$\alpha$-helical Polyalanine Ala$_{100}$

$N=27069, M=3410$

NAO basis set (FHI-aims)

![Graph showing elapsed time vs. number of cores for different architectures.](image)

Auckenthaler et al., Parallel Computing (2011)
Actively developed high-performance library (Kus, Marek, others):

ELPA dense eigensolver
http://elpa.mpcdf.mpg.de

- High-performance, massively parallel two-stage tri-diagonalization
- Scales to over 10k CPU cores
- Optimized for present-day supercomputing architectures
- Drop-in enhancement to ScaLAPACK standard and generalized eigensolvers

Fig. 3, Kus et al., Comput. Phys. Commun. 2019
Matrix size 20,000, Intel MKL version 2018

Auckenthaler et al., Parallel Comput. 2011
Kus et al., Comput. Phys. Commun. 2019

But How to Tackle Larger System Sizes?
Plan A: Have Big Computer, Push the Eigensolver

ELPA Eigenvalue Solver

- Efficient full → band → tridiagonal reduction & backtransform
- Dense linear algebra up to full spectrum

Auckenthaler, Blum, Bungartz, Huckle, Johanni, Krämer, Lang, Lederer, Willems, Parallel Computing 37, 783 (2011).


Benchmark: Alvaro Vazquez-Mayagoitia, ANL
Plan B: Density Matrix Based Approaches

**Straightforward Solution: Eigenvalue Solver, \(O(N^3)\)**

\[ \psi_k(r) = \sum_i c_{ik} \varphi_i(r) \quad \rightarrow \quad \hbar c_k = e_k s c_k \quad \rightarrow \quad n(r) = \sum_k f_k |\psi_k(r)|^2 \]
Plan B: Density Matrix Based Approaches

**Straightforward Solution: Eigenvalue Solver, \(O(N^3)\)**

\[
\psi_k(r) = \sum_i c_{ik} \varphi_i(r) \quad \rightarrow \quad \hbar c_k = e_k s c_k \quad \rightarrow \quad n(r) = \sum_k f_k |\psi_k(r)|^2
\]

However, in DFT we need \(n(r)\) - eigenvectors are not needed:

\[
n(r) = \sum_k f_k \sum_{ij} c_{ik} c_{jk} \varphi_i(r) \varphi_j(r) = \sum_{ij} \varphi_i(r) n_{ij} \varphi_j(r)
\]

Density matrix - sparse!

Decays with distance between basis functions \(i,j\)

\[
\rightarrow \text{Construct density matrix in less than } O(N^3) \text{ for given } h, s?\]
Many Solution Strategies to Kohn-Sham Problem

\[ \hbar c_k = \epsilon_k s c_k \]

Different use cases (basis sets, physics), different “solvers”. Solve (eigenvectors, $O(N^3)$) or circumvent (density matrix)?
Many Solution Strategies to Kohn-Sham Problem

\[ \hbar c_k = \epsilon_k s c_k \]

Different use cases (basis sets, physics), different “solvers".
Solve (eigenvectors, \(O(N^3)\)) or circumvent (density matrix)?

**Exact solvers**
- Lapack
- Scalapack
- ELPA
- EigenExa
- Magma
- ...

**Iterative solvers**
- Davidson
- Projected Preconditioned Conjugate Gradient
- Chebychev Filtering
- Slepc-SIPS
- ...

**DM: \(O(N)\) solvers**
- NTPoly
- Various code-internal and/or proprietary implementations

**Other DM-based approaches**
- PEXSI
- Orbital Minimization Method
- FEAST
- ...

\[ \hbar c_k = \epsilon_k s c_k \]
Many Solution Strategies to Kohn-Sham Problem

\[ \hbar c_k = \epsilon_k s c_k \]

Different use cases (basis sets, physics), different “solvers”. Solve (eigenvectors, \(O(N^3)\)) or circumvent (density matrix)?

**Exact solvers**
- Lapack
- Scalapack
- ELPA
- EigenExa
- Magma
- ...

**Iterative solvers**
- Davidson
- Projected Preconditioned Conjugate Gradient
- Chebychev Filtering
- Slepc-SIPS
- ...

**DM: \(O(N)\) solvers**
- NTPoly
- Various code-internal and/or proprietary implementations

**Other DM-based approaches**
- PEXSI
- Orbital Minimization Method
- FEAST
- ...

**Robust General**
- (Essentially) robust \(N_{basis} >> N_{ev}\)

**Sparse H, S**
- Nonmetallic systems

**Sparse H, S can depend on XC**
## Many Solution Strategies to Kohn-Sham Problem

$$\hbar \mathbf{c}_k = \mathbf{\epsilon}_k \mathbf{s}_c \mathbf{c}_k$$

Different use cases (basis sets, physics), different “solvers”. Solve (eigenvectors, $O(N^3)$) or circumvent (density matrix)?

<table>
<thead>
<tr>
<th>Exact solvers</th>
<th>Iterative solvers</th>
<th>DM: $O(N)$ solvers</th>
<th>Other DM-based approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lapack</td>
<td>Davidson</td>
<td>NTPoly</td>
<td>PEXSI</td>
</tr>
<tr>
<td>Scalapack</td>
<td>Projected</td>
<td>Various code-internal and/or proprietary implementations</td>
<td></td>
</tr>
<tr>
<td>ELPA</td>
<td>Preconditioned</td>
<td></td>
<td>Orbital</td>
</tr>
<tr>
<td>EigenExa</td>
<td>Conjugate Gradient</td>
<td></td>
<td>Minimization Method</td>
</tr>
<tr>
<td>Magma</td>
<td>Chebychev Filtering</td>
<td></td>
<td>FEAST</td>
</tr>
<tr>
<td></td>
<td>Slepc-SIPS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Robust General

(Essentially) robust $N_{\text{basis}} >> N_{\text{ev}}$

Sparse $H, S$

Nonmetallic systems

Sparse $H, S$ can depend on XC
Expand density matrix $P$ in terms of rational functions (poles)

$$P = \sum_l \text{Im} \left( \frac{w_l}{H - (z_l + \mu)S} \right)$$

- $P$: Density matrix
- $H$: Hamiltonian matrix
- $S$: Overlap matrix
- $z_l$: Shift (pole)
- $w_l$: Weight
- $\mu$: Chemical potential

- **Computational steps:**
  1. Determine pole expansion parameters $\{z_l, w_l\}$
  2. Compute all poles in parallel by process groups
  3. Sum over poles to get density matrix and other quantities

- Applicable to insulators, semiconductors as well as metals
- Selected inversion: Evaluate needed elements of $(H - (z_l - \mu)S)^{-1}$

Pole Expansion and Selected Inversion (PEXSI)

Lv. 1
Pole expansion (20~100 poles)

Lv. 2
Chemical potential search (a few trial μ points)

Lv. 3
Parallel selected inversion (1,000+ CPU cores)

PEXSI sparse density matrix solver
http://www.pexsi.org

\[ P = \sum \text{Im} \left( \frac{w_l}{H - (z_l + \mu)S} \right) \]

- Computational complexity (semi-local XC)
  - 1D system: O(N)
  - 2D system: O(N^{1.5})
  - 3D system: O(N^2)

- Scales to over 100k CPU cores

NTPoly: \(O(N)\) Approaches

- Zero-temperature density matrix \(P\) must be:
  - (assuming orthogonal basis)
  - Hermitian: \(P = P^*\)
  - Normalized: \(\text{Tr}(P) = N_{\text{electron}}\)
  - Idempotent: \(P = P^2\)

- Computational steps:
  1) Convert to orthogonal basis
  2) Construct initial guess of density matrix
  3) Iteratively purify density matrix to satisfy the above three conditions
  4) Convert back to non-orthogonal basis

- Established methods for linear scaling electronic structure theory: One million atoms possible

Goedecker, Rev. Mod. Phys. 1999
NTPoly: $O(N)$ Approaches

- Purification function $f(\tilde{P}_n)$ is often a matrix polynomial of order $m$
  - Canonical purification ($m = 3$)
  - Trace resetting purification ($m = 2, 3, 4, ...$)
  - Generalized canonical purification ($m = 3$)

- Example: 2$^\text{nd}$ order trace resetting purification
  \[
  \sigma_n = \text{sign}(N_{\text{electron}} - \text{Tr}(P_n))
  \]
  \[
  P_{n+1} = P_n + \sigma_n(I - P_n)P_n
  \]

- Sparse matrix algebra essential for linear scaling

References:
- Goedecker, Rev. Mod. Phys. 1999
Many Different Solvers - How to Unify Access?

- Replicated infrastructure to implement solvers efficiently
- Conversion between a variety of matrix storage formats
- Complexity in solver selection for different problems

Electronic structure codes

Solvers:
- ELPA
- SLEPc-SIPs
- NTPoly
- PEXSI
- libOMM
- Many more
ELSI: Connecting Electronic Structure Codes and Solvers

Yu et al., Comput. Phys. Commun. 2018
http://elsi-interchange.org
http://git.elsi-interchange.org/elsi-devel/elsi-interface

Electronic structure codes

Unified interface connecting KS-DFT codes and solvers

Fast, automatic matrix format conversion and redistribution

Recommendation of optimal solver based on benchmarks
ELSI: Connecting Electronic Structure Codes and Solvers

Electronic structure codes

Currently:
FHI-aims, Siesta, DFTB+, DGDFT

Unified interface connecting KS-DFT codes and solvers

Fast, automatic matrix format conversion and redistribution

Recommendation of optimal solver based on benchmarks

Solvers
- ELPA
- PEXSI
- SLEPe-SIPs
- libOMM
- NTPoly
- Many more

Yu et al., Comput. Phys. Commun. 2018
http://elsi-interchange.org
http://git.elsi-interchange.org/elsi-devel/elsi-interface
Portability, Extendability, Sustainability

- Designed for rapid integration into a variety of electronic structure codes
- Compatible with common workflows
  - Single self-consistent field (SCF)
  - Multiple SCF cycles (geometry relaxation or molecular dynamics)
- Supports density matrix solvers and eigensolvers on equal footing
- Technical settings adjustable for experienced users
- Object-oriented: Concurrent instances

**ELSI API**

```plaintext
elsi_init
elsi_set_parameters
while (geometry not converged) do
  while (SCF not converged) do
    elsi_{ev|dm}
  end while
elsi_reinit
end while
elsi_finalize
```
Git commit to add NTPoly support into FHI-aims: 10 lines of code

Victor Yu

```f90
--- a/read_control.f90
+++ b/read_control.f90
@@ -12581,6 +12581,12 @@ subroutine read_control
  + case("elsi_ntpoly_method")
  +   read(inputline,*,end=88,err=99) desc_str,elsi_ntpoly_method
  + case("elsi_ntpoly_tol")
  +   read(inputline,*,end=88,err=99) desc_str,elsi_ntpoly_tol
  + case("elsi_ntpoly_filter")
  +   read(inputline,*,end=88,err=99) desc_str,elsi_ntpoly_filter

--- a/elsi_wrapper.f90
+++ b/elsi_wrapper.f90
@@ -265,6 +265,10 @@ subroutine aims_init_elsi
  + call elsi_set_ntpoly_method(eh,elsi_ntpoly_method)
  + call elsi_set_ntpoly_tol(eh,elsi_ntpoly_tol)
  + call elsi_set_ntpoly_filter(eh,elsi_ntpoly_filter)
```
ELSI Functionality

Functionalities

- Eigensolvers: ELPA, EigenExa, SLEPc, LAPACK
- Density matrix solvers: libOMM, PEXSI, NTPoly
- Parallel solution for spin-polarized and periodic systems
- Dense and sparse matrix formats, arbitrary distribution
- Parallel matrix format conversion and I/O

Portability

- CMake build system supports Cray, GNU, IBM, Intel, PGI compilers
- From laptops to supercomputers (Cobra, Cori, Mira, Sierra, Theta, …)
- Provides Fortran, C, C++ programming interfaces
- Part of CECAM Electronic Structure Library (ESL): Distribution of shared open-source libraries in the electronic structure community

Code freely available at: http://elsi-interchange.org
Performance: Solver Benchmarks on Equal Footing

Edison Cray XC30
Processor: Intel Ivy Bridge
Interconnect: Cray Aries
5,586 compute nodes
134,064 processing cores
2.57 Petaflops

Cori-Haswell Cray XC40
Processor: Intel Haswell
Interconnect: Cray Aries
2,388 compute nodes
76,416 processing cores
2.81 Petaflops

Cori-KNL Cray XC40
Processor: Intel Knights Landing
Interconnect: Cray Aries
9,688 compute nodes
658,784 processing cores
29.5 Petaflops

http://www.nersc.gov/edison
http://www.nersc.gov/cori

Victor Yu
Performance: Solver Benchmarks on Equal Footing

1D
(a) Ge

2D
(b) MoS$_2$

3D
(c) Cu$_2$BaSnS$_4$

Sparsity of matrices
Example: FHI-aims Basis Sets - ELPA vs. PEXSI

- DFT-PBE
- FHI-aims (all-electron)
- 2,560 CPU cores on Cori-Haswell

PEXSI faster for large low-dimensional (sparse) systems

PEXSI: Semilocal DFT, $O(N) - O(N^2)$ for large systems

Performance: Solver Benchmarks on Equal Footing

1D
(a) CNT

2D
(b) Graphene

3D
(c) Graphite

Sparsity of matrices
Example: Siesta Basis Sets - ELPA vs. PEXSI

<table>
<thead>
<tr>
<th>System</th>
<th>Number of Basis Functions</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1D CNT</strong></td>
<td>800, 1600, 3200, 6400</td>
<td>10^0, 10^1, 10^2, 10^3</td>
</tr>
<tr>
<td>ELPA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PEXSI</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2D graphene</strong></td>
<td>800, 1600, 3200, 6400</td>
<td>10^0, 10^1, 10^2, 10^3</td>
</tr>
<tr>
<td>ELPA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PEXSI</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3D graphite</strong></td>
<td>800, 1600, 3200, 6400</td>
<td>10^0, 10^1, 10^2, 10^3</td>
</tr>
<tr>
<td>ELPA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PEXSI</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PEXSI: Semilocal DFT, $O(N)$ - $O(N^2)$ for large systems

Performance: Solver Benchmarks on Equal Footing

3D Periodic Systems:

(a) H₂O

(b) Si
Example: DFTB+ (semiempirical) - ELPA, PEXSI, NTPoly

- **NTPoly** settings:
  - 4th order TRS method
  - 10^{-5} truncation threshold
  - 10^{-2} convergence criterion

- **NTPoly** accuracy: Band structure energies agree with ELPA within 10^{-5} eV/atom

- DFTB
- **DFTB+** (highly sparse matrices)
- 2,560 CPU cores on Cori-Haswell

---

**NTPoly**: Sparse Matrix Algebra, O(N) Solvers

Dawson, Nakajima, Computer Physics Communications 225, 154-165 (2018)
ELSI Decision Layer (Beginnings)

1st layer: Quick decision
- Choose ELPA
  ✓ n_basis < 20,000
- Choose PEXSI
  ✓ dimensionality < 3
  ✓ sparsity > 0.95
- Choose NTPoly
  ✓ n_basis > 100,000
  ✓ sparsity > 0.999
  ✓ have_gap

2nd layer: Direct comparison
- Exclude PEXSI
  ✓ dimensionality == 3
  ✓ n_cpu < 20,000
- Exclude NTPoly
  ✓ n_basis < 50,000
  ✓ sparsity < 0.99
  ✓ NOT have_gap
- Try available solvers (one in an SCF step)
- Choose fastest solver
What About Iterative Solvers? (Plane Waves)

If $N_{\text{basis}} >> N_{\text{ev}}$, “full matrix” solvers are not competitive (time and memory).

Alternative: Iterative - e.g., Davidson

- $\Psi$
- Solve Rayleigh Ritz problem $(\Psi^*H\Psi, \Psi^*\Psi)$ for smallest eigenpairs $\Lambda, Q$
- $R = H\Psi Q - \Psi Q\Lambda$
- If $\|R\| < tol$, converged
- Approximately solve $(H - \Lambda_i I)V_i = R_i$ for all $V_i$

$\Psi = [\Psi V]$
**Problem:** Data Structures, Distribution in Different Codes Can Vary Widely.

**Solution:** Ask Code to Perform Detailed Operations, Drive Sequence of Steps.

**RCI - Generic Iterative Solvers for Existing Codes**

"Code"

- **Initialization**
  - Initialized
  - The instruction is accomplished
  - Post-calculation
  - Converged

**RCI**

**ELSI-RCI**

An instruction
ijob = RCI_INIT_IJOB

Do as the instruction

end do
ELSI-RCI - Proof of Principle

- Si supercell
- ONCV pseudopotential
- $E_{\text{cut}}=20$ Ha
- Kerker preconditioner
- Initial random wave function, Hamiltonian from converged SCF

→ Efficient prototyping, implementation of solvers for different purposes (e.g., BSE)
Conclusion and Acknowledgments

ELSI offers a unified interface to various eigensolvers and density matrix solvers
- Small-to-medium calculations: ELPA
- Large 1D/2D geometries: PEXSI
- Large 3D geometries: NTPoly, PEXSI
- More solvers to come

- Adopted by DFTB+, DGDFT, FHI-aims, SIESTA
- Ongoing efforts
  - RCI framework for iterative solvers
  - Solver optimization targeting GPUs

ELSI is an NSF SI²-SSI supported project under grant number 1450280. Any opinions, findings, and conclusions or recommendations expressed here are those of the author(s) and do not necessarily reflect the views of NSF.