# The ELSI Infrastructure to Solve or Circumvent the Kohn-Sham Eigenvalue Problem

## An NSF SI2-SSI Supported Software Infrastructure Project


Victor Wen-Zhe Yu[1], Fabiano Corsetti[2], William Paul Huhn[1], Mathias Jacquelin[3], Weile Jia[4], Björn Lange[1], Lin Lin[3,4], Jianfeng Lu[5], Wenhui Mi[1], Álvaro Vázquez-Mayagoita[6], Chao Yang[3], Haizhao Yang[5], and Volker Blum[1]

[1] Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708, USA; [2] Department of Materials, Imperial College London, London SW7 2AZ, United Kingdom
[3] Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA; [4] Department of Mathematics, University of California, Berkeley, Berkeley, CA 94720, USA
[5] Department of Mathematics, Duke University, Durham, NC 27708, USA; [6] Argonne National Laboratory, Argonne, IL 60349, USA


elsi-interchange.org

## The O(N³) Kohn-Sham Eigenvalue Problem

**Kohn-Sham Density-Functional Theory (DFT)** [1]:
Predictive Theory for Materials and Molecules

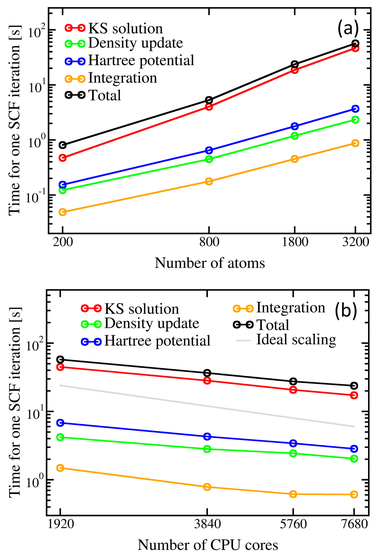
- ~30,000 scientific publications/year report results of DFT-based calculations and DFT method developments.
- Frequently among top applications on world's highest performance supercomputing infrastructure.
- Large ecosystem of communities centered around different codes, approximations, use cases.

**"Cubic Wall"**:
K.-S. Eigenvalue Problem - O(N³) with System Size N

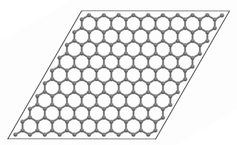
**Example system**: Supercell models of graphene monolayer.

**Electronic structure method**: DFT-PBE [2], FHI-aims [3] all-electron simulation, "light" settings, ELPA [4,5] library for massively parallel eigenvalue solutions.

**Scalability limit**: O(N³) solution of the eigenvalue problem must dominate over remaining O(N) grid-based operations by Amdahl's law (generic to K.-S. theory).

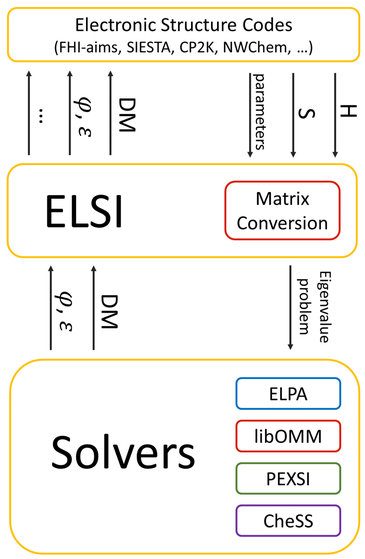



Graphene 10x10 supercell



(a) Scalability of number of atoms shown by timings in one SCF iteration. Systems of 200, 800, 1800, 3200 atoms are graphene 10x10, 20x20, 30x30, and 40x40 supercell models. Computation done with 1920 CPU cores.

(b) Scalability of CPU cores shown by timings in one SCF iteration. System studied is graphene 40x40 supercell model (3200 atoms).




## The ELSI Project: ELectronic Structure Infrastructure

**Supported Methods**

**Direct diagonalization:** Eigenvalue soLvers for Petaflop Applications (**ELPA**) solves the generalized or standard eigenvalue problem using dense linear algebra, including Cholesky transformation, reduction to banded and tridiagonal form, tridiagonal solver, and efficient back-transformations.

**Density matrix methods:** Circumventing the direct solution of the eigenvalue problem. Orbital Minimization Method (**libOMM**) [6-8] solves the K.-S. equation as a generalized eigenvalue problem within a density matrix formalism. Its basic strategy is to find the set of Wannier functions (WFs) describing the occupied subspace by direct unconstrained minimization of an appropriately-constructed functional. The density matrix can then be calculated from the WFs. Pole EXpansion and Selected Inversion (**PEXSI**) [9,10] efficiently evaluates certain selected elements of matrix functions, e.g., the Fermi-Dirac function of the Hamiltonian, which yields the density matrix. The sparsity of matrices is exploited to reduce the scaling to **O(N)-O(N²)**, depending on the dimensionality of the system.

**ELSI Software**

**ELSI Interface** provides an easy-to-use API to simplify the implementation of the supported solver libraries into any electronic structure code. Based on an analysis of the problem size and sparsity, the optimal method for a given eigenvalue problem will be suggested by ELSI. Efficient and scalable matrix format conversion is automatically performed when necessary.

## ELSI Running on High Performance Computing (HPC) Facilities

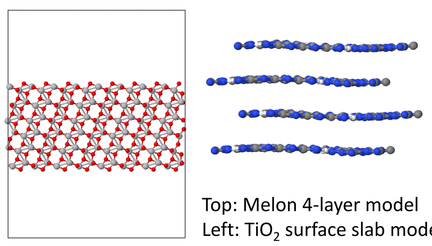
**Portability**

Successful compilation and execution of ELSI have been proven on a broad list of leading supercomputers/architectures.

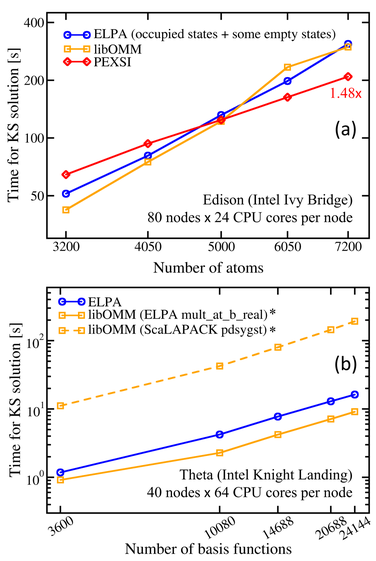
- Titan: Cray XK7 @ Oak Ridge National Laboratory (ORNL). Opteron 6274 (16 CPU cores) + NVidia K20x GPU.
- Cori: Cray XC40 @ National Energy Research Scientific Computing Center (NERSC). Intel Xeon Phi 7250 (68 CPU cores).
- Mira: IBM BlueGene/Q @ Argonne National Laboratory (ANL). Power BQC (16 CPU cores).
- And more: Theta @ ANL, Edison @ NERSC, mid-sized computer clusters, even laptops.

**Benchmarks**

**Electronic structure method**: DFT-PBE calculation using FHI-aims all-electron simulation package based on numeric atom-centered orbitals with variable basis set size.



Top: Melon 4-layer model
Left: $TiO_2$ surface slab model




**(a)** Supercell models of graphene monolayer containing 3200 ~ 7200 atoms. Calculation done on Edison @ NERSC using 80 nodes x 24 CPU cores per node. PEXSI becomes faster than ELPA and libOMM for models consisting more than 5000 atoms. With a smaller scaling exponent, PEXSI offers the opportunity to perform (extra-) large calculations.

**(b)** Melon (polymerized heptazine) 4-layer model containing 288 atoms. Calculation done on Theta @ ANL using 40 nodes x 64 CPU cores per node. ScaLAPACK subroutine pdsygst is used in libOMM to transform the generalized eigen-problem to the standard form. A speed-up of at least 15x is achieved by replacing pdsygst with ELPA subroutines.
* libOMM time corresponds to one CG iteration

**(c)** $TiO_2$ surface slab model containing 3456 atoms. Calculation done on Titan @ ORNL using 80 nodes x (16 CPU cores + 1 GPU) per node. GPU-accelerated ELPA shows a speed-up of 1.25 over the SSE assembler kernel, which has the best performance among all the ELPA CPU kernels. Active development on further improvement of GPU speed-up and scalability is work-in-progress.

## Development and Improvements of Solver Libraries

**Scalability of ELPA and PEXSI on Next Generation Many-Core Processors**

ELSI and its ingredients are demonstrated to be promising candidates on Intel's Many Integrated Core (MIC) architecture. Preliminary benchmarks of ELPA and PEXSI on supercomputers equipped with Intel Knight Landing processors are shown.

**(a)** Scalability of ELPA on Theta @ ANL for matrices of small, medium, and large sizes. The data point in the top right corner (cyan) represents an eigenvalue problem of the size 1 million x 1 million, solved by ELPA within 3 hours using **~200,000** CPU cores.

**(b)** Scalability comparison between MPI and OpenMP implementations of the parallel selected inversion (**PSelInv**) [11,12] in PEXSI on Cori @ NERSC. The OpenMP parallelization displays strong scalability up to 16 threads on Intel KNL, allowing memory footprint reduction compared to the MPI-only code. A **hybrid MPI + OpenMP** implementation of the PSelInv for sparse **symmetric** matrices is being developed.

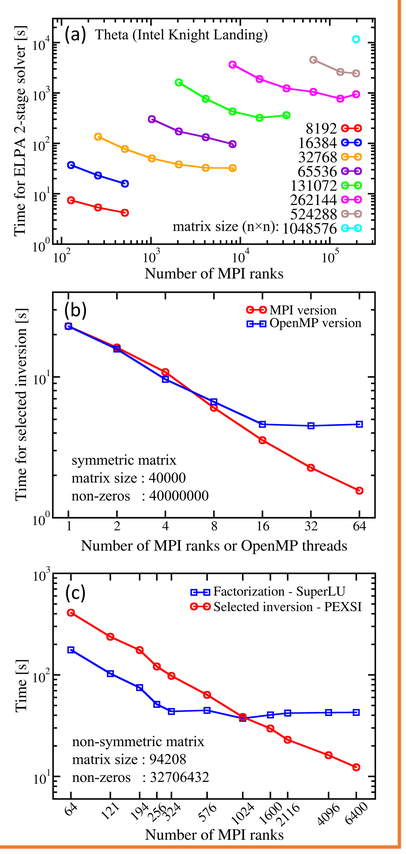
**(c)** In order to cope with the increasingly more expensive communication costs, a new parallel selected inversion algorithm for sparse **non-symmetric** matrices exploiting asynchronous computations and communications has been introduced. Shown here is scalability of the two key steps in PEXSI - factorization by SuperLU and selected inversion by PSelInv for the non-symmetric implementation.

**Parallel Sparse BLAS (pspBLAS) Implementation in libOMM**

Distributed-memory sparse linear algebra library **pspBLAS** is integrated in libOMM, offering sparse vector-matrix and sparse matrix-matrix multiplications with high scalability and load balance via a 2D block-cyclic distribution. Various dense and sparse matrix formats have been supported in MatrixSwitch, the low-level matrix manipulation layer in libOMM that can be used as the matrix format converter in ELSI. Future work will include improvement of the sparse matrix-matrix multiplication kernel in pspBLAS.

**Chebyshev Sparse Solvers (CheSS): New K.-S. Approach in ELSI**

**CheSS** provides various high-level operations based on Chebyshev expansions for sparse matrices. It includes an **O(N)** implementation of the Fermi operator expansion method to calculate the density matrix in K.-S. DFT calculations. Integration of CheSS into ELSI is ongoing.




## The ELSI Interface: Connecting User Codes with Solver Libraries

**Application Programming Interface (API)**

Example: ELSI used in a normal self-consistent field (SCF) solver routine

**elsi_init** (…)

Choose one specific solver to use or simply let ELSI decide the optimal method based on an analysis of the given problem; specify the matrices storage format used in the user code.

**elsi_set_mpi** (…)
**elsi_set_blacs** (…)

Pass MPI/BLACS information to ELSI.

SCF loop
  **elsi_customize_{elpa|omm|pexsi}** (…)

Customize parameters used in the chosen solver. ELSI default settings provided otherwise.

  user code prepares Hamiltonian and overlap matrices
  **elsi_{dm|ev}_{real|compelx}** (H,S,D or ev)

Compute the density matrix or eigenvalues/eigenvectors.

  user code checks convergence criterion
  exit loop if converged
end SCF loop
**elsi_finalize** ( )

**Easy to implement**: Designed to be flexible for rapid integration into a variety of K.-S. DFT codes.

**Versatile**: Includes density matrix and eigensystem formalisms on equal footing.

**Flexible**: All technical settings in solver libraries are adjustable for experienced users.

**Successful connection** has been demonstrated between solver libraries supported in ELSI and electronic structure code (FHI-aims, as the first step). Refer to the above block for benchmarks.

## The ELSI Interchange: Community Website and Services


The ELSI community website, elsi-interchange.org , includes:

- **Code repositories** (GitLab): Master repository for ELSI interface software, with **flexible access** to allow simple download vs. development activities. **37** users already registered, showing interests from FHI-aims, SIESTA, CP2K, BigDFT, Quantum Espresso, NWChem, and ATK.
- Technical support available to the community in forms of wiki, forum, webinar, workshop, and code integration assistance.

## References and Acknowledgements


[1] W. Kohn and L.J. Sham, Phys. Rev., 140: A1133-A1138 (1965).
[2] J.P. Perdew, K. Burke, and M. Ernzerhof, Phys. Rev. Lett., 77: 3865–3868 (1996).
[3] V. Blum, R. Gehrke, F. Hanke, P. Havu, V. Havu, X. Ren, K. Reuter, and M. Scheffler, Comput. Phys. Commun., 180: 2175–2196 (2009).
[4] T. Auckenthaler, V. Blum, H.J. Bungartz, T. Huckle, R. Johanni, L. Krämer, B. Lang, H. Lederer, and P.R. Willems, Parallel Computing, 37: 783–794 (2011).
[5] A. Marek, V. Blum, R. Johanni, V. Havu, B. Lang, T. Auckenthaler, A. Heinecke, H.J. Bungartz, and H. Lederer, J. Phys.: Condens. Matter, 26: 213201 (2014).
[6] F. Mauri, G. Galli, and R. Car, Phys. Rev. B, 47: 9973 (1993).
[7] P. Ordejon, D.A. Drabold, R.M. Martin, and M.P. Grumbach, Phys. Rev. B, 51: 1456 (1995).
[8] F. Corsetti, Comput. Phys. Commun., 185: 873–883 (2014).
[9] L. Lin, J. Lu, L. Ying, R. Car and W. E, Commun. Math. Sci., 7: 755 (2009).
[10] L. Lin, M. Chen, C. Yang and L. He, J. Phys. Condens. Matter, 25: 295501 (2013).
[11] M. Jacquelin, L. Lin, N. Wichmann and C. Yang, IEEE IPDPS, 192 (2016).
[12] M. Jacquelin, L. Lin and C. Yang, ACM Trans. Math. Software, 43: 21 (2017 ).



This work is supported by the National Science Foundation under Grant Number 1450280. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the National Science Foundation.

The authors gratefully acknowledge the following organizations: