ELPA Eigensolvers

Autotunable, **E**nergy-efficient, **O**ptimized

ELPA-AEO

http://elpa-aeo.mpcdf.mpg.de

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

BMBF Projekt 01IH15001
Feb 2016 - Jan 2019

Project Partners

Lehrstuhl für Angewandte Informatik
Prof. B. Lang (BUW)

Prof. M. Scheffler, Dr. Ch. Carbogno (FHI)

Dr. H. Lederer, Dr. A. Marek (MPCDF)

Lehrstuhl für Informatik mit Schwerpunkt
Wissenschaftliches Rechnen (TUM-SCCS)
Prof. H.-J. Bungartz, Prof. Th. Huckle

Lehrstuhl für Theoretische Chemie
Prof. K. Reuter, Dr. Ch. Scheurer (TUM-CH)

Stepwise Approach for the Eigenproblem

Solution of the generalized matrix eigenproblem generally proceeds in five steps:

(I) Transformation to a dense standard eigenproblem (e.g., by Cholesky decomposition of S),
$H_{KS}c_I = \varepsilon_I S c_I \rightarrow A q_A = \lambda q_A, \quad \lambda = \varepsilon_I$

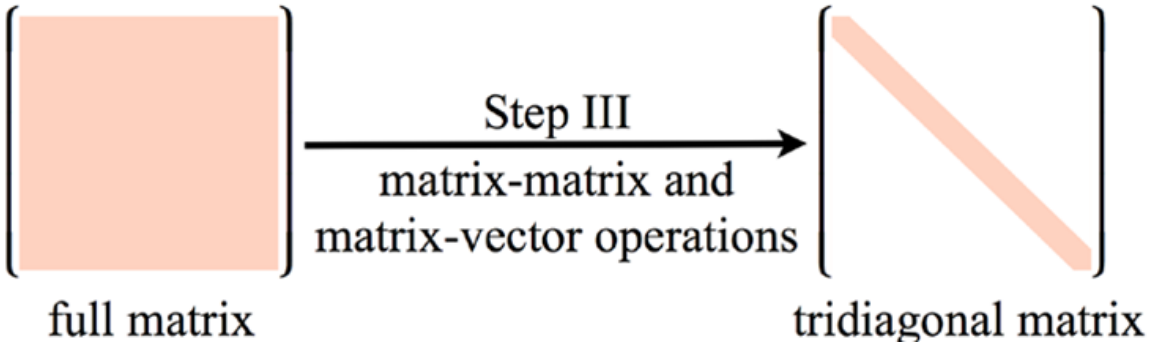
(II) Reduction to tridiagonal form, A -> T;

(III) Solution of the tridiagonal problem for k eigenvalues and -vectors, $T q_T = \lambda q_T$;

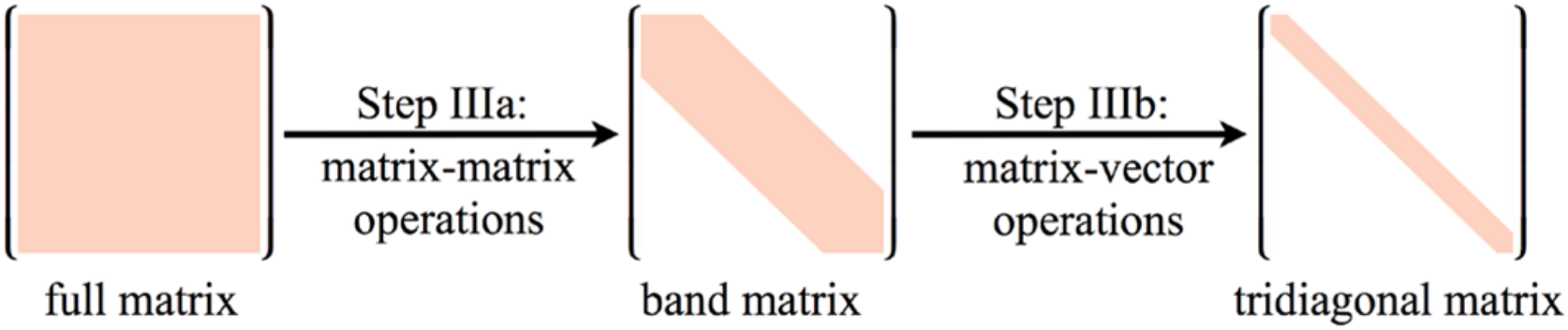
(IV) Back transformation of k eigenvectors to dense orthonormal form, $q_T \rightarrow q_A$;

(V) Back transformation to the original, non-orthonormal basis, $q_A \rightarrow c_I$.

ELPA1: one-step direct solver



ELPA2: two-step direct solver



Pros: *allows full use of matrix–matrix products and sparse matrix-vector products*
Cons*: gives rise to one extra back transformation step of the eigenvectors*

Eigensolver performance comparison on Cray XC systems

Brandon Cook, Thorsten Kurth, Jack Deslippe, Pierre Carrier, Nick Hill, Nathan Wichmann

First published: 25 September 2018 | https://doi.org/10.1002/cpe.4997

Authors from NERSC & Cray Inc

PDF TOOLS SHARE

Summary

Hermitian (symmetric) eigenvalue solvers are the core constituents of electronic structure, quantum-chemistry, and other HPC applications such as Quantum ESPRESSO, VASP, CP2K, and NWChem to name a few. Our understanding of the performance of

From the conclusions: „we found that ELPA outperforms SCALAPACK on all architectures tested and all matrix sizes and concurrencies. In addition, ELPA performs significantly better than SCALAPACK in scenarios where multiple OpenMP threads are used for each MPI rank. The performance gap between ELPA and SCALAPACK widens as function of threads used."

Autotuning
Monitoring
Steering

- New object-oriented interface (back compatible)
- Measure and store timings of different components
- Reporting back to calling program
- Steering through calling program
- Easy „set“ and „ get“ steering parameters through „key value“ pairs

Autotuning

Task: Automatic finding and usage of the most efficient ELPA variant for a given problem and a given compute architecture

- All parameters in the ELPA library are classified either „tunable“ or „not-tunable“
- For each tunable parameter, its potential values get defined and whether the problem is real or complex.
- With autotuning activated, a decision tree is being created which contains all tunable parameters and the values to be tested. This tree can be restricted by the user.
- During the autotuning runs at each solver call one combination of parameters /values gets tested and timed. Autotuning stores the sets of actually best parameter combinations.
- The actual state of the autotuning procedure can be read anytime and stored if wanted.
- Thus autotuning can be continued later or the best settings of a completed autotuning can be loaded used for a further simulation.

ELPA-AEO

Parameters for Autotuning

MPCDF Max Planck Computing & Data Facility

Parameter name	Description	# of options
solver	1-step or 2-step solver ?	2
gpu	GPU usage (potentially only advantageous for part of calc.)?	256
kernel	For 2-step solvers: which kernel for back transformation?	Up to 18
blocking_in_band_to_full	Optimal blocking factor inside the back transformation?	10
Stripewidth	Stripe width for sub matrices?	17
max_stored_rows	Optimal number of stored rows?	8
omp_threads	Optimal number of used OpenMP threads per MPI task	Architecture dependent
Cannon	Usage of Cannon algorithm in transformation of the generalized eigenproblem advantageous?	2

Performance Gain by Autotuning

For a given hardware configuration:

Automatic selection of the most efficient procedure

For a given problem

(matrix size and number of eigenvalues needed):

Comparing the different available procedures in consecutive SCF cycles, identification and automatic selection of the most efficient one for all further steps.

For elpa2

Selection of the best block size and the best compute-intenisve kernels.
Inclusion of GPUs in case of availability

Tuning and Monitoring

Options for calling Main program:

Autotuning: enable/disable (default: disable)

User guided tuning:

User can preselect a subset of settings –
then all other options are skipped

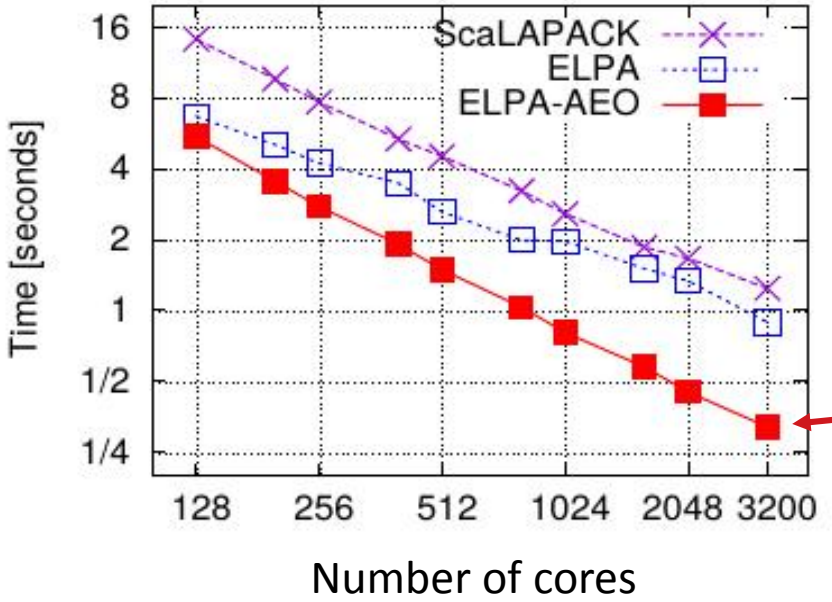
Monitoring:

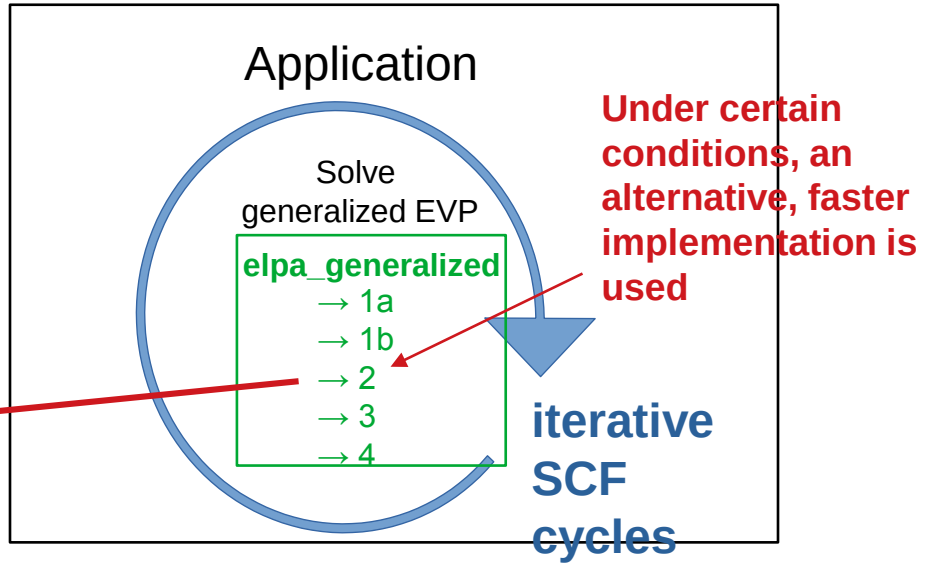
Enable/disable timing measurements
Queries:
- timing results
- number of calls of a subroutine
- performance in GigaFlop/s

Example of Tuning

Improvements to the generalized eigenvalue problem AX = BXμ under certain conditions

ELPA automatically detects whether the improved method can be applied





Steering:

Solving an eigenproblem of a band-structured matrix with ELPA 2

Five Steps:

a) Matrix transformation to band structure

b) Transformation from banded to tridiagonal structure

c) Solving the eigenproblem of the tridiagonal matrix

d) Back transformation of the found eigenvectors into the banded strucured system

e) Back transformation of the eigenvectors into the original system

Old:

In case the original matrix was already band structured, calculation a) was done nevertheless since ELPA had no knowledge about it

New:

In case the original matrix was already band structured, this can be indicated (through a setting) and calculations a) and e) are skipped

Steering:

Repeated solving of a generalized eigenproblem with constant matrix B inside a convergence loop

Normal procedure:

a) Cholesky decomposition of matrix B
b) Explicit calculation of the inverse values of the Cholesky factors
c) Transformation of A to the standard eigenvalue problem,
   through multiplication with the inverse Cholesky factors
d) Solving the standard eigenproblem
e) Back transformation of the eigenvectors into the generalized system

For certain **applications as often in materials science**, inside one iteration loop a large number of generalized eigenproblems have to be solved and **the B matrix only changes after some dozens of iteration steps.**

Optimization:
The calling program can tell ELPA whether the B matrix has changed.
In case of no change, steps a) and b) get skipped and the inverse Cholesky factors already calculated in a previous step are used. **This can lead to a performance gain up to 40%.**

Steering:

Precision: Acceleration of a convergence iteration through partially calculating in single precision

Often at the beginning of a convergence loop solving of the generalized eigenproblem double precision is not needed, since the physical system is still far from converging.

Therefore now a procedure allows doing the first 10 bis 20 iteration steps partially in single precision; only the last steps are done in double precision.

Code available to allow for calculations in single precision and internally convert between double and single precision.

The advanced user can now determine the precision for each calculation of the steps a) to e),
potentially significantly accelerating the convergence loop.

Recent publications

P. Kus et al.:
GPU Optimization of Large-Scale Eigenvalue Solver, Numerical Mathematics and Advanced Applications ENUMATH 2017, Lecture Notes in Computational Science and Engineering 126, Springer Berlin **2019** (ISBN 978-3-319-96414-0)

P. Kus et al.:
Optimizations of the Eigensolvers in the ELPA Library
Parallel Computing 85, pp 167-177 (**2019**), doi: 10.1016/j.parco.2019.04.003

A Alvermann et al.:
Benefits from using mixed precision computations in the ELPA-AEO and ESSEX-II eigensolver projects, Japan Journal of Industrial and Applied Mathematics, 36(2), 699-717 (**2019**), doi: 10.1007/s13160-019-00360-8

Supported Architectures

X86
Intel Xeon Phi
with AVX512

measured up to matrix size
1 M on 200k cores

X86
Intel Xeon
with AVX512

measured up to matrix
size 1 M up to 150k cores

X86
AMD EPYC
with AVX2

NVIDIA V100
GPUs in Intel
Xeon CPU hosts
via PCIe

NVIDIA V100
GPUs in Power 9
hosts via NVLINK 2

IBM Power
8 + 9

NEC SX-Aurora
vector processors

ARM - 32
ARM - 64

MAC OS

Latest ELPA Releases

ELPA-Release 2018.11 – Dec 2018

- Automatic checkpoint/restart for autotuning
  -> Option to start new simulation with restart file from previous autotuning run
- Improved transformation from generalized to standard problem

ELPA-Release 2019.05 – June 2019

- new kernels for ARM arch64 added
- elpa_print_kernels supports GPU usage
- new simple real kernels: block4 and block6
- users can define the default-kernel at build time
- allow measurements with the likwid tool
- c functions can be build with optional arguments if compiler supports it (configure option)
- ELPA versioning number is provided in the C header files
- fix an out-of-bound-error in elpa2
- fix an error if PAPI measurements are activated

Next ELPA Release
- Preview -

ELPA-Release 2019.11 – planned still in 2019

- Support for a high performance solver
  for skew-symmetric eigenvalue problems

  Solving the Bethe-Salpeter Eigenvalue Problem
  important for applications in quantum chemistry

ELPA vs Intel MKL 2019.5

- In summer 2018 Intel announced at a conference in the US to develop an own two-step direct eigensolver for MKL to be released in 2019

- In MKL 2019.5 (Sep 13, 2019) improved solvers were provided

MKL PDSYEVR

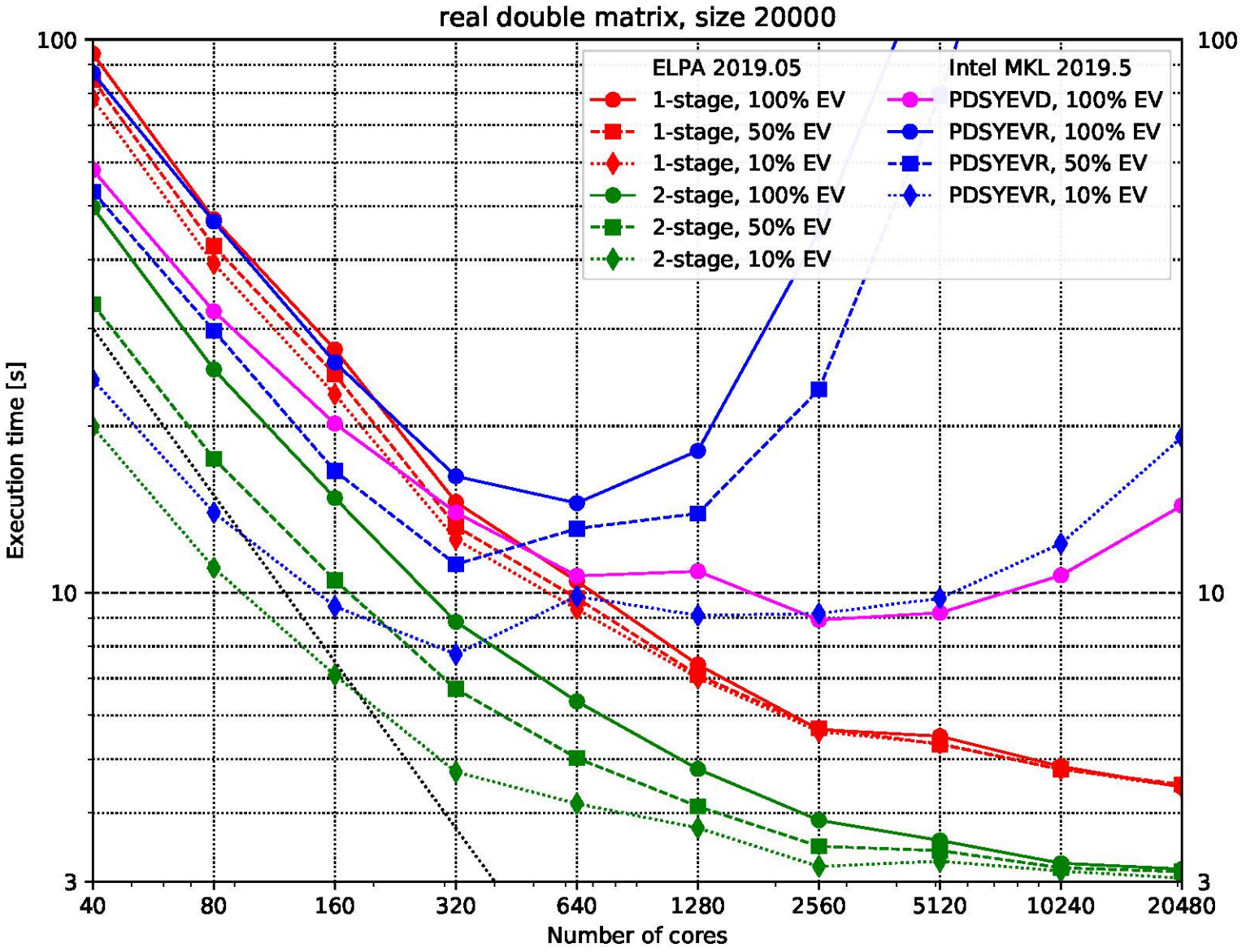
Computes selected eigenvalues and, optionally, eigenvectors of a real symmetric matrix First, the matrix *A* is reduced to real symmetric tridiagonal form. Then, the eigenproblem is solved using the parallel MRRR algorithm. Last, if eigenvectors have been computed, a backtransformation is done.

MKL PDSYEVD

Computes all eigenvalues and eigenvectors of a real symmetric matrix by using a divide and conquer algorithm.

Both new solvers have been compared to ELPA1 and ELPA2 on an Intel SkyLake processor based architecture (Cobra at MPCDF with Xeon Gold 6148))

ELPA vs Intel MKL 2019.5



Conclusions

- ELPA solvers continue to keep their role as performance leaders both on a single compute node as with respect to scalability

- Elpa solvers are highly energy-efficient

  - Automatic autotuning saves compute time

  - Unique: Interaction between the calling main program and ELPA solvers („monitoring“ and „steering“) saves compute time by avoiding unnecessary computations

- All major architectures are supported