ExaNLA Survey Response Report

Generated on: 10/17/2025 at 12:47:39 PM

Submission Details

Library Name: CP2K
Version: v2025.2
Contact Name: Thomas Kühne
Email: t.kuehne@hzdr.de
Organization: Helmholtz-Zentrum Dresden-Rossendorf

Selected NLA Operations

1. Matrix Inversion
2. Symmetric/Hermitian Eigenvalue Problems
3. Matrix-Matrix Multiplication (GEMM)
4. Cholesky Factorization

1. Codes Information

Basic information about your application/simulation codes.

Library Name:
CP2K

Current Version:
v2025.2

Contact Information:
Not specified

Name:
Thomas Kühne

Email:
t.kuehne@hzdr.de

Organization:
Helmholtz-Zentrum Dresden-Rossendorf

Application Domain:
Not specified

What is the primary application domain of your codes?:
Materials Science

Materials Science:
Not specified

What are the main functionalities of your code?:
Ground state DFT, Time-dependent DFT, Many-body perturbation theory (GW, BSE), Molecular dynamics, Quantum transport, Surface science, Defect calculations, Phase transitions, Excited-state dynamics

If you selected "Other", please specify::
Not specified

Climate/Weather Modeling:
Not specified

What are the main functionalities of your code?:
Not specified

If you selected "Other", please specify::
Not specified

Fluid Dynamics:
Not specified

What are the main functionalities of your code?:
Not specified

If you selected "Other", please specify::
Not specified

Other Domain Functions:
Not specified

What are the main functionalities of your code?:
Not specified

Use Case Information:
Not specified

Does your codes have multiple distinct use cases?:
Yes, multiple distinct use cases

Which use case are you describing in this submission?:
Ab-initio Molecular dynamics

Library Description:
Not specified

2. Matrix Inversion

Matrix Inversion (A^{-1}):
Yes

Purpose and Use Cases of Matrix Inversion:
**Solving linear systems (Ax = b), Shift-and-invert operations ((H-
transformation/orthogonalization (e.g., Löwdin, $S^{-1/2}$), Precondit
(M^{-1}"HA^{-1}), Density matrix construction/purification, Kohn-Sham**

Matrix Properties:
Not specified

Matrix Structure:
Sparse, Block sparse, Dense

Matrix Distribution:
Block cyclic distribution (e.g., ScaLAPACK style), Block row/column distribution

Matrix Storage Format:
Dense (column-major/row-major), Compressed Sparse Row (CSR/CRS), Block CSR/CSC

Mathematical Properties:
Positive definite, Hermitian/Symmetric

Matrix Dimensions:
Large (10,000 - 100,000)

Accuracy Requirements:
Not specified

Inverse Accuracy:
Low accuracy (10^-3)

Linear System Accuracy:
Low accuracy (10^-3)

Working Precision:
Mixed precision (e.g., FP32 inversion with FP64 refinement), Double precision (64-bit), Low precision (e.g., FP16, BF16), Single precision (32-bit)

Workload Characteristics:
Not specified

Computation Pattern: capability or capacity:
Large-scale single inversions (e.g., one large matrix at a time, using significant computational resources), Repeated inversions of similar matrices (e.g., during time evolution or parameter sweeps)

Dense Linear Algebra Libraries:
Not specified

Currently Used Libraries:
ScaLAPACK, cuSolverMp

Interested in Using:
ScaLAPACK, cuSolverMp

Sparse or Iterative Solver Libraries:
Not specified

Currently Used Libraries:
Not specified

Interested in Using:
Not specified

Specialized and Domain-Specific Libraries:
Not specified

Currently Used Libraries:
PEXSI (Selected Inversion for DFT)

Interested in Using:
H2Lib / HLIBpro (H-matrix methods)

Benchmarking Requirements:
Not specified

Benchmark Input Types:
Real matrices from application workloads

Can You Provide Data or Mini-apps?:
Yes, matrices only

Scaling Requirements:
Not specified

3. Matrix-Matrix Multiplication (GEMM)

Matrix-Matrix Multiplication (GEMM):
Yes

Matrix Properties:
Not specified

Matrix Structure:
Dense matrices, Sparse matrices, Tall-and-skinny matrices, Distributed matrices, Small batched matrices

Matrix Distribution:
Block cyclic distribution (e.g., ScaLAPACK style), Hybrid CPU-GPU distribution, Cannon/SUMMA-style distribution

Matrix Storage Format:
Dense (column-major/row-major), Compressed Sparse Row (CSR/CRS), Block CSR/CSC

Which types of matrix multiplications do you perform?:
Standard multiplication (AB), Transpose multiplication (A@B + ²C)

Typical Dimensions:
Not specified

Matrix Size Range:
Small (< 100), Medium (100 - 1,000), Large (1,000 - 10,000), Very Large (10,000 - 100,000), Extreme (> 100,000)

Typical Matrix Shapes:
Not specified

Batch Size:
Not specified

Distributed-Memory NLA Library Usage:
Not specified

General Distributed Memory Libraries (CPU/GPU):
ScaLAPACK, COSMA, ELPA, cuBLASMp (NVIDIA distributed-memory GPUs), Other: DLA-Future

Special/Advanced Implementations:
Block-sparse or block-structured multiplication, Algorithm-specific implementations (e.g., Strassen, communication-avoiding algorithms)

Are there any NLA libraries you are interested in using (but have not yet adopted)?:
COSMA, ScaLAPACK, ELPA, cuBLASMp (NVIDIA distributed-memory GPUs), Other: DLA-Future

Future Requirements:
Not specified

Desired Features:
Improved sparse-dense multiplication, Better tensor contraction support, Auto-tuning capabilities, Hardware-specific optimizations

Benchmarking Requirements:
Not specified

Benchmark Input Types:
Real matrices from application workloads, Both synthetic and real data

Can You Provide Data or Mini-apps?:
Yes, matrices only

Scaling Requirements:
Both strong and weak scaling needed

Working Precision:
Mixed precision (e.g., FP32 multiplication with FP64 accumulation), Double precision (64-bit)

4. Cholesky Factorization

Cholesky Factorization (A = LL^T):
Yes

Diagonal Dominance:
Not sure

Condition Number:
Well-conditioned (< 10^3)

Matrix Properties and Structure:
Sparse

Matrix Distribution:
Block cyclic distribution (e.g., ScaLAPACK style)

Matrix Storage Format:
Dense (column-major/row-major), Compressed Sparse Row (CSR/CRS), Block CSR/CSC

Matrix Dimensions:
Medium (1,000 – 10,000)

Factorization Tolerance:
Medium accuracy (10^-6)

Working Precision:
Double precision (64-bit), Mixed precision (e.g., FP32 factorization with FP64 refinement)

Workload Characteristics:
Not specified

Computation Pattern: capability or capacity:
Large-scale single factorizations (e.g., one large matrix at a time, using significant computational resources), Many independent smaller factorizations (e.g., batch processing multiple matrices simultaneously), Mix of large and small factorizations (varying resource requirements), Repeated factorizations of similar matrices (e.g., during iterative refinement or optimization)

Distributed-Memory Dense NLA Library Usage:
Not specified

Currently Used Libraries:
ScaLAPACK, cuSoverMp, DLA-Future, ELPA

Interested in Using, but not currently using:
Not specified

Specialized Libraries (Sparse/Structured/Hierarchical):
Not specified

Currently Used Libraries:
Not specified

Interested in Using, but not currently using:
Not specified

Benchmarking Requirements:
Not specified

Benchmark Input Types:
Real matrices from application workloads

Can You Provide Data or Mini-apps?:
Yes, matrices only

Scaling Requirements:
Weak scaling (fixed problem size per process/node), Strong scaling (fixed total problem size), Both strong and weak scaling needed

5. Standard Eigenvalue Problems (Ax = »x)

Symmetric/Hermitian

Primary Use Cases:
Kohn–Sham equations (standard DFT), GW quasiparticle calculations, Bethe–Salpeter equation (Tamm-Dancoff approximation), Tight-binding models

Matrix Properties and Structure:
Dense, Sparse, Block Sparse, Block tridiagonal/Block Diagonal

Matrix Properties:
Not specified

Matrix Distribution:
Block cyclic distribution (e.g., ScaLAPACK style), Hybrid CPU-GPU distribution

Matrix Storage Format:
Dense (column-major/row-major), Compressed Sparse Row (CSR/CRS)

Positive definiteness:
Always positive definite

Eigenvalue distribution:
Not specified

Problem Scale:
Small (< 1,000), Medium (1,000 - 10,000), Large (10,000 - 100,000)

Computation Requirements:
Not specified

Percentage of eigenvalues:
90-100%

What to compute:
Eigenvalues and eigenvectors

Eigenvalue location:
Eigenvalues near a target shift

Required tolerance/precision:
Not specified

Residual tolerance type:
Not specified

Absolute residual tolerance:
Not specified

Relative residual tolerance:
Not specified

Hybrid residual tolerance:
Not specified

Orthogonality tolerance:
Low (10^-3)

Working Precision:
Double precision (64-bit), Mixed precision (e.g., FP32/FP64 combination)

Workload Characteristics:
Not specified

Computation Pattern: capability or capacity:
Large-scale single problems (e.g., one large matrix at a time, using significant computational resources), Many independent smaller problems (e.g., batch processing multiple matrices simultaneously), Mix of large and small problems (varying resource requirements), Repeated similar-sized problems (e.g., time evolution or parameter sweeps)

Distributed-Memory NLA Library Usage:
Not specified

Distributed-Memory Dense Linear Algebra:
ScaLAPACK, ELPA, DLA-Future

Iterative Eigensolvers:
Not specified

High-Level & Interface Libraries:
Not specified

Are there any NLA libraries you are interested in using (but have not yet adopted)?:
ScaLAPACK, ELPA, ChASE, DLA-Future

Benchmarking Requirements:
Not specified

Benchmark Input Types:
Real matrices from application workloads

Can You Provide Data or Mini-apps?:
Yes, matrices only

Scaling Requirements:
Both strong and weak scaling needed

6. Generalized Eigenvalue Problems (Ax = »Bx)

Symmetric/Hermitian A, SPD B

Matrix Structure:
Not specified

Reduction to Standard Eigenproblem (using B):
Not specified

Reduction to Standard Eigenproblem:
No, solver works directly with generalized form (e.g., contour integration, FEAST, iterative solvers)

Reduction Method:
Approximate inverse of B (e.g., polynomial or iterative approximation)

Matrix Properties:
Not specified

Eigenvalue distribution:
Not specified

Problem Scale:
Medium (1,000 - 10,000)

Computation Requirements:
Not specified

Percentage of eigenvalues:
90-100%

What to compute:
Eigenvalues and eigenvectors

Eigenvalue location:
Not specified

Required tolerance/precision:
Not specified

Residual tolerance type:
Not specified

Absolute residual tolerance:
Not specified

Relative residual tolerance:
Not specified

Hybrid residual tolerance:
Not specified

Orthogonality tolerance:
Low (10^-3)

Working Precision:
Double precision (64-bit), Mixed precision (e.g., FP32/FP64 combination)

Workload Characteristics:
Not specified

Computation Pattern: capability or capacity:
Large-scale single problems (e.g., one large generalized eigenproblem at a time, using significant computational resources), Many independent smaller problems (e.g., batch processing multiple generalized eigenproblems simultaneously), Mix of large and small problems (varying resource requirements), Repeated similar-sized problems (e.g., time evolution or parameter sweeps)

Distributed-Memory NLA Library Usage:
Not specified

Distributed-Memory Dense Linear Algebra:
ScaLAPACK, ELPA, DLA-Future

Iterative Eigensolvers:
Not specified

High-Level & Interface Libraries:
Not specified

Are there any NLA libraries you are interested in using (but have not yet adopted)?:
ScaLAPACK, ChASE, ELPA, DLA-Future

Benchmarking Requirements:
Not specified

Benchmark Input Types:
Real matrices from application workloads

Can You Provide Data or Mini-apps?:
Yes, matrices only

Scaling Requirements:
Strong scaling (fixed total problem size), Weak scaling (fixed problem size per process/node), Both strong and weak scaling needed