ExaNLA Survey Response Report

Generated on: 10/17/2025 at 12:48:23 PM

Submission Details

Library Name: FHI-aims
Version: 250822
Contact Name: Volker Blum
Email: volker.blum@duke.edu
Organization: Duke University

Selected NLA Operations

1. Quasi-Hermitian (BSE) Eigenvalue Problems
2. Symmetric/Hermitian Eigenvalue Problems
3. Matrix-Matrix Multiplication (GEMM)

1. Codes Information

Basic information about your application/simulation codes.

Library Name:
  FHI-aims

Current Version:
  250822

Contact Information:
  Not specified

  Name:
    Volker Blum

  Email:
    volker.blum@duke.edu

  Organization:
    Duke University

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, Excited-state dynamics, Crystal structure prediction, Defect calculations, Surface science

    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?:
Groubd stte DFT

Library Description:
General purpose code for simulations in materials science and chemistry.

2. Matrix-Matrix Multiplication (GEMM)

Matrix-Matrix Multiplication (GEMM):
Yes

Matrix Properties:
Not specified

Matrix Structure:
Dense matrices, Distributed matrices

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

Matrix Storage Format:
Dense (column-major/row-major), Compressed Sparse Row (CSR/CRS), Compressed Sparse Column (CSC/CCS), Multiple formats (conversion as needed)

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

Typical Dimensions:
Not specified

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

Typical Matrix Shapes:
Square matrices (m "H n "H k)

Batch Size:
Not specified

Distributed-Memory NLA Library Usage:
Not specified

General Distributed Memory Libraries (CPU/GPU):
ScaLAPACK, ELPA, COSMA

Special/Advanced Implementations:
Not specified

Are there any NLA libraries you are interested in using (but have not yet adopted)?:
cuBLASMp (NVIDIA distributed-memory GPUs)

Future Requirements:
Not specified

Desired Features:
Improved sparse-dense multiplication, Hardware-specific optimizations, Other: simply, efficiency of all steps.

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), Both strong and weak scaling needed

Working Precision:
Double precision (64-bit)

3. Standard Eigenvalue Problems (Ax = »x)

Symmetric/Hermitian

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

Matrix Properties and Structure:
Dense

Matrix Properties:
Not specified

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

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

Positive definiteness:
Varies depending on the problem

Eigenvalue distribution:
Varies

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

Computation Requirements:
Not specified

Percentage of eigenvalues:
Varies

What to compute:
Varies

Eigenvalue location:
Varies

Required tolerance/precision:
Not specified

Residual tolerance type:
Absolute residual (||Ax - »x||)

Absolute residual tolerance:
Machine precision

Relative residual tolerance:
Not specified

Hybrid residual tolerance:
Not specified

Orthogonality tolerance:
Machine precision

Working Precision:
Double precision (64-bit)

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), 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

Iterative Eigensolvers:
Not specified

High-Level & Interface Libraries:
ELSI – Abstraction layer for eigenvalue solvers

Are there any NLA libraries you are interested in using (but have not yet adopted)?:
SLEPc, PRIMME, ChASE

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)

Quasi-Hermitian (BSE) - HÈ = EÈ, where H = (A B; -B* -A*), A =

Matrix Properties and Structure:
Dense (standard full matrix)

Matrix Properties:
Not specified

Eigenvalue distribution:
Varies

Matrix scale/size:
Large (10,000 - 100,000)

Computation Requirements:
Not specified

Percentage of eigenvalues:
Varies

What to compute:
Varies

Eigenvalue location:
Varies

Required tolerance/precision:
Not specified

Residual tolerance type:
Absolute residual (||Ax - »x||)

Absolute residual tolerance:
Machine precision

Relative residual tolerance:
Not specified

Hybrid residual tolerance:
Not specified

Orthogonality tolerance:
Machine precision

Working Precision:
Double precision (64-bit)

Workload Characteristics:
Not specified

Computation Pattern: capability or capacity:
Large-scale single problems (e.g., one large BSE matrix at a time, using significant computational resources)

Distributed-Memory NLA Library Usage:
Not specified

General Non-Hermitian Eigensolvers:

Solvers Targeting Full BSE Problems:
BSEPACK – Dedicated solver library for full BSE eigenproblems, ChASE – Chebyshev Accelerated Subspace Eigensolver, extended to BSE with custmoized filters and rayleigh-ritz

Are there any NLA libraries you are interested in using (but have not yet adopted)?:
ChASE

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)

4. Generalized Eigenvalue Problems (Ax = »Bx)

Symmetric/Hermitian A, SPD B

Matrix Structure:
A is dense, B is dense, A is banded, B is banded, Complex valued, Real valued

Reduction to Standard Eigenproblem (using B):
Not specified

Reduction to Standard Eigenproblem:
Yes, always

Reduction Method:
C h o l e s k y   f a c t o r i z a t i o n   o f   B   (B   =   L L W   o r   B   =   L * L)

Matrix Properties:
Not specified

Eigenvalue distribution:
Varies

Problem Scale:
Large (10,000 - 100,000)

Computation Requirements:
Not specified

Percentage of eigenvalues:
Varies

What to compute:
Varies

Eigenvalue location:
Varies

Required tolerance/precision:
Not specified

Residual tolerance type:
A b s o l u t e   r e s i d u a l   (| | A x   -   » x | |)

Absolute residual tolerance:
Machine precision

Relative residual tolerance:
Not specified

Hybrid residual tolerance:
Not specified

Orthogonality tolerance:
Machine precision

Working Precision:
Double precision (64-bit)

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), Repeated similar-sized problems (e.g., time evolution or parameter sweeps)

Distributed-Memory NLA Library Usage:
Not specified

Distributed-Memory Dense Linear Algebra:
ELPA, ScaLAPACK

Iterative Eigensolvers:
Not specified

High-Level & Interface Libraries:
ELSI – Abstraction layer for eigenvalue solvers (e.g., used by SIESTA, FHI-aims)

Are there any NLA libraries you are interested in using (but have not yet adopted)?:
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:
Strong scaling (fixed total problem size)