ExaNLA Survey Response Report

Generated on: 10/17/2025 at 12:49:00 PM

Submission Details

Library Name: libNEGF
Version: 1.3
Contact Name: Alessandro Pecchia
Email: alessandro.pecchia@cnr.it
Organization: Consiglio Nazionale delle Ricerche

Selected NLA Operations

1. Matrix Inversion
2. Linear System Solvers
3. Matrix-Matrix Multiplication (GEMM)

1. Codes Information

Basic information about your application/simulation codes.

Library Name:
  libNEGF

Current Version:
  1.3

Contact Information:
  Not specified

  Name:
    Alessandro Pecchia

  Email:
    alessandro.pecchia@cnr.it

  Organization:
    Consiglio Nazionale delle Ricerche

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?:
      Quantum transport

    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?:
Transport calculations in nanostructures

Library Description:
libNEGF is a general-purpose library to perform quantum transport calculations. It is agnostic of the underlying Hamiltonian formulation, provided it is a local basis representation (FD, FEM, LCAO, etc.). In recent years there was a big development of the code thanks to the Energy-Oriented Center of Excellence, involving a tight collaboration with JSC. Currently the codes relays on block-dense linear algebra also ported to GPUs. There is an activity to further improve scalability by distributing the spatial domain by Nested-Dissection or similar approaches. Mixed-precision and compression by low-rank approximations are also under scrutiny.

2. Matrix Inversion

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

Purpose and Use Cases of Matrix Inversion:
Green's function calculation (ÉI-H-£(É)){^{1}

Matrix Properties:
Not specified

Matrix Structure:
Dense, Low-rank updates (A + UCV@)

Matrix Distribution:
Distribution optimized for selected inversion

Matrix Storage Format:
Dense (column-major/row-major)

Mathematical Properties:
Complex valued

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

Accuracy Requirements:
Not specified

Inverse Accuracy:
Medium accuracy (10^-6)

Linear System Accuracy:
Medium accuracy (10^-6)

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

Workload Characteristics:
Not specified

Computation Pattern: capability or capacity:
Many independent smaller inversions (e.g., batch processing multiple matrices simultaneously), Part of larger computation (e.g., Green's function calculation, preconditioner construction)

Dense Linear Algebra Libraries:
Not specified

Currently Used Libraries:
ScaLAPACK, Other: cuSolver

Interested in Using:
cuSolverMp

Sparse or Iterative Solver Libraries:
Not specified

Currently Used Libraries:
SuperLU / SuperLU_DIST

Interested in Using:
Not specified

Specialized and Domain-Specific Libraries:
Not specified

Currently Used Libraries:
Domain-specific GPU libraries (e.g., custom Green’s function solvers)

Interested in Using:
Not specified

Benchmarking Requirements:
Not specified

Benchmark Input Types:
Real matrices from application workloads

Can You Provide Data or Mini-apps?:
Yes, both matrices and mini-apps

Scaling Requirements:
Strong scaling (fixed total problem size), Weak scaling (fixed problem size per process/node)

3. Linear System Solvers

Linear System Solvers:
Yes

Matrix Properties:
Not specified

Matrix Structure:
Not specified

Matrix Properties:
Not specified

Matrix Distribution:
Not specified

Matrix Storage Format:
Not specified

Matrix Size:
Not specified

Performance Requirements:
Not specified

Accuracy Requirements:
Not specified

Working Precision:
Not specified

Scaling Requirements:
Not specified

Parallelization Requirements:
Not specified

Workload Characteristics:
Not specified

Computation Pattern: capability or capacity:
Not specified

Library Usage:
Not specified

Dense Solver Libraries:
Not specified

Sparse Solver Libraries:
Not specified

Benchmarking Requirements:
Not specified

Benchmark Input Types:
Not specified

Can You Provide Data or Mini-apps?:
Not specified

Scaling Requirements:
Not specified

4. Matrix-Matrix Multiplication (GEMM)

Matrix-Matrix Multiplication (GEMM):
Yes

Matrix Properties:
Not specified

Matrix Structure:
Dense matrices, Distributed matrices, Block-structured matrices

Matrix Distribution:
Custom domain decomposition

Matrix Storage Format:
Dense (column-major/row-major)

Which types of matrix multiplications do you perform?:
Full GEMM (±AB + ^{2}C), Triple product (ABC), Mixed precision

Typical Dimensions:
Not specified

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

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

Batch Size:
Not applicable

Distributed-Memory NLA Library Usage:
Not specified

General Distributed Memory Libraries (CPU/GPU):
Custom distributed implementation

Special/Advanced Implementations:
Algorithm-specific implementations (e.g., Strassen, communication-avoiding algorithms)

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

Future Requirements:
Not specified

Desired Features:
Better mixed precision support, More flexible memory layouts

Benchmarking Requirements:
Not specified

Benchmark Input Types:
Real matrices from application workloads

Can You Provide Data or Mini-apps?:
Yes, both matrices and mini-apps

Scaling Requirements:
Strong scaling (fixed total problem size), Weak scaling (fixed problem size per process/node)

Working Precision:
Single precision (32-bit), Double precision (64-bit), Mixed precision (e.g., FP32 multiplication with FP64 accumulation), Tensor Core compatible precisions