Design, Usage and Features of ⟨T⟩LAPACK
github.com/tlapack/tlapack

Weslley Pereira[1]

[1]National Renewable Energy Laboratory, Golden - CO

ICL Friday Lunch Talk
February 16, 2024

What is ⟨T⟩LAPACK ?

github.com/tlapack/tlapack

- C++ Template-based Linear Algebra Package.
- Routines are free functions using high-level abstraction.
- Interoperability with: SLATE, mdspan, Eigen3, StarPU.

```cpp
// User's code looks like:

auto A = std::mdspanA( A_ptr, 100, 100 );
auto B = Eigen::MatrixXd::Random(100,100).eval();

int infoA = getrf( A, piv );
int infoB = getrf( B, piv );
```

```cpp
// Inside potrf we find commands like:

A(piv[0], 0) = A(0, 0);
auto A00 = slice(A, range(0, k0), range(0, k0));
```

⟨T⟩LAPACK is a Work In Progress

Recent updates:

- Nonsymmetric eigenvalue solver (Apr-Jun 2022).
- LQ and bidiagonalization (Jun-Jul 2022).
- Continuous Testing using BLIS (Jul 2022).
- LU factorization and LU inversion (Sep-Oct 2022).
- Continuous Testing for Eigen::Matrix and mdspan (Nov 2022).
- Half precision and multiprecision types (Jan 2023)
- Reciprocal scaling with complex factor (Jun 2023).
- Workspace design and implementation (Oct 2022 - Oct 2023).
- Implicit QR SVD (Oct 2023).
- Multishift QZ (Oct 2023).
- Symmetric eigenvalue problem (current).
- Train of Givens rotations (current).

What is next?

- More functionality: eigenvector computation, multishift SVD, ...
- Contributions from students: EISPACK nonsymmetric eigenvalue solver, tests with 8-bit formats, matrix generators.
- More tests for performance, accuracy and reliability.
- More documentation about the design of the software.

Contributors

Senior Team Members

- Julien Langou, CU Denver
- Weslley Pereira, CU Denver
- Thijs Steel, KU Leuven

Undergrads, Grads, Postdocs (CU Denver)

- Heidi Meier, Lindsay Slager, Naomi Wilson, Natnael Tebeje, Yuxin Cai (Summer 2022, undergraduate)
- Ali Lotfi (Fall 2022, postdoc)
- Michael Pultorak, Racheal Asamoah, Skylar Johns (Spring 2023, undergraduate)
- Jonhathan Rhyne (AY 22-23, graduate)
- Alexander Hatle, Aslak Djupskas, Dominic Lioce, Hugh Kadhem, Seok-Ho Yoon, Sudhanva Kulkarni (Fall 2023, undergraduate)

Why ⟨T⟩LAPACK ?

LAPACK

- provides reference implementation for a wide set of state-of-the-art algorithms.
- is largely used for its efficiency and reliability.
- has been incorporated into several packages.

BLAS / LAPACK

- are not templated, currently support 4 data types.
- are not templated, replicate a single algorithm on 4 files.
- work on a few data layouts.

⟨T⟩LAPACK

- uses templates that encompass data types and layouts.
- interoperates with SLATE, mdspan and Eigen3.
- uses modern software standards.

Agenda

1. High-level design
2. Overloading for performance
3. Performance tests
4. Integration and Interaction

High-level design

Computational routines take

1. Runtime or compile-time parameters.
2. Matrices and vectors of a variety of types.
3. Optional arguments (e.g., workspace is optional).

```
1  /// Options for blocked Cholesky
2  struct PotrfOpts : public BlockedCholeskyOpts {
3      PotrfVariant variant = PotrfVariant::Blocked;
4  };
5
6  /// Declaration of blocked Cholesky
7  template <TLAPACK_UPLO uplo_t, TLAPACK_MATRIX matrix_t>
8  int potrf(
9      uplo_t uplo,
10     matrix_t& A,
11     const PotrfOpts& opts = {} );
```

High-level design

Computational routines take

1. Runtime or compile-time parameters.
2. Matrices and vectors of a variety of types.
3. Optional arguments (e.g., workspace is optional).

```
1  /// Options for blocked Cholesky
2  struct PotrfOpts : public BlockedCholeskyOpts {
3      PotrfVariant variant = PotrfVariant::Blocked;
4  };
5
6  /// Declaration of blocked Cholesky
7  template <TLAPACK_UPLO uplo_t, TLAPACK_MATRIX matrix_t>
8  int potrf(
9      uplo_t uplo,
10     matrix_t& A,
11     const PotrfOpts& opts = {} );
```

Let's look at the concepts and plugins!

Generalized Matrix-Vector multiplication in ⟨T⟩LAPACK

```
template< TLAPACK_MATRIX matrixA_t,
          TLAPACK_VECTOR vectorX_t,
          TLAPACK_VECTOR vectorY_t,
          TLAPACK_SCALAR alpha_t,
          TLAPACK_SCALAR beta_t >
void gemv( Op trans, const alpha_t& alpha, const matrix_t& A
    , const vectorX_t& x, const beta_t& beta, vectorY_t& y )
{
    using TA = type_t< matrix_t >;
    using TX = type_t< vectorX_t >;
    using idx_t = size_type< matrix_t >;
    const idx_t m = nrows(A);
    const idx_t n = ncols(A);
    const idx_t leny = size(y);
// (...)
    if (trans == Op::NoTrans ) {
        for (idx_t j = 0; j < n; ++j) {
            const scalar_type<alpha_t, TX> tmp = alpha*x[j];
            for (idx_t i = 0; i < m; ++i)
                y[i] += tmp * A(i, j);
// (...)
```

Recursive Cholesky in ⟨T⟩LAPACK I

```
1  template <TLAPACK_UPLO uplo_t, TLAPACK_SMATRIX matrix_t>
2  int potrf2(uplo_t uplo, matrix_t& A)
3  {
4    using T = type_t<matrix_t>;
5    using real_t = real_type<T>;
6    using idx_t = size_type<matrix_t>;
7    using range = pair<idx_t, idx_t>;
8
9    const real_t one( 1.0 );
10   const real_t zero(0);
11   const idx_t n = nrows(A);
12 // (...)
13   // Recursive part:
14   {
15     const idx_t n1 = n/2;
16
17     // Define A11 and A22
18     auto A11 = slice( A, range{0,n1}, range{0,n1} );
19     auto A22 = slice( A, range{n1,n}, range{n1,n} );
20
21     // Factor A11
22     int info = potrf2( uplo, A11 );
```

Recursive Cholesky in ⟨T⟩LAPACK II

```
23      if( info != 0 )
24        return info;
25
26      if( uplo == Uplo::Upper ) {
27        // Update and scale A12
28        auto A12 = slice( A, range{0,n1}, range{n1,n} );
29        trsm( LEFT_SIDE, UPPER_TRIANGLE, CONJ_TRANS,
        NON_UNIT_DIAG, one, A11, A12 );
30
31        // Update A22
32        herk( UPPER_TRIANGLE, CONJ_TRANS, -one, A12, one, A22
        );
33      }
34      else { /* ... */ }
35
36      // Factor A22
37      info = potrf2( uplo, A22 );
38
39      if( info == 0 ) return 0;
40      else return info + n1;
41    }
42  }
```

Optional arguments in ⟨T⟩LAPACK

- Encapsulated into a C++ struct.
- A function may have its own struct with options.
- Options may be derived from other options.
- Options can be used for input and output.

```
1  /// Options for blocked Cholesky
2  struct BlockedCholeskyOpts : public EcOpts {
3      constexpr BlockedCholeskyOpts(const EcOpts& opts = {}) :
        EcOpts(opts){};  ///< Use base constructor
4
5      size_t nb = 32;  ///< Block size
6  };
7
8  /// Declaration of blocked Cholesky
9  template <TLAPACK_UPLO uplo_t, TLAPACK_SMATRIX matrix_t>
10 int potrf_blocked( uplo_t uplo, matrix_t& A,
11                    const BlockedCholeskyOpts& opts );
```

Workspace in ⟨T⟩LAPACK

Workspace

- queries are separate routines.
- is an input in the *work interface*.
- is anything that can be reshaped into matrices and vectors.

```
1  template <class T,
2            TLAPACK_SMATRIX matrix_t, TLAPACK_VECTOR vector_t>
3  constexpr WorkInfo geqr2_worksize(const matrix_t& A,
4                                    const vector_t& tau)
5  {/* Computes the sizes of the workspace of type T */}
6
7  template <TLAPACK_SMATRIX matrix_t, TLAPACK_VECTOR vector_t>
8  int geqr2(matrix_t& A, vector_t& tau);
9  {/* Allocates space and call geqr2_work() */}
10
11 template <TLAPACK_SMATRIX matrix_t,
12           TLAPACK_VECTOR vector_t,
13           TLAPACK_WORKSPACE work_t>
14 int geqr2_work(matrix_t& A, vector_t& tau, work_t& work);
15 {/* Implementation of the algorithm is here! */}
```

Overloading for performance

Wrappers to optimized BLAS

⟨T⟩LAPACK uses BLAS++ and LAPACK++ from SLATE to access optimized BLAS and LAPACK libraries.

```
1  /// <T>LAPACK implementation
2  template<TLAPACK_VECTOR vector_t,
3           disable_if_allow_optblas_t< vector_t > = 0>
4  auto nrm2( const vector_t& x )
5  { ... }
6
7  #ifdef TLAPACK_USE_LAPACKPP
8
9      /// Wrapper to BLAS++
10     template<TLAPACK_LEGACY_VECTOR vector_t,
11              enable_if_allow_optblas_t< vector_t > = 0>
12     auto nrm2( const vector_t& x )
13     {
14         auto x_ = legacy_vector(x);
15         const auto& n = x_.n;
16         return ::blas::nrm2( n, x_.ptr, x_.inc );
17     }
18
19 #endif
```

Integration with StarPU

In a few lines:

- `starpu_data_handle_t`: handle to manage a piece of data. Splits data in a grid of nx-by-ny tiles.
- `tlapack::starpu::Matrix<T>`: matrix abstraction to `starpu_data_handle_t`. Allows for blocking that doesn't match with the nx-by-ny grid.
- ⟨T⟩LAPACK plugin: provides `slice`, `nrows`, `ncols`, etc.

```
1  int main() {
2    /* initialize StarPU here */
3    float *A_;
4    starpu_malloc((void**)&A_, 90 * 90 * sizeof(float));
5    genHermitianMatrix(A_, 90, 90); // 90x90 matrix
6    Matrix<float> A(A_, 90, 90, 20, 20); // 20x20 grid
7
8    potrf_opts_t<size_t> opts; opts.nb = 9; // 9x9 blocks
9    int info = potrf(upperTriangle, A, opts);
10   /* terminate StarPU here */
11 }
```

Performance tests

Integration with StarPU

- **Key aspect:** No need to modify top-level routines!
- Blocked Cholesky calls `gemm`, `herk`, `trsm` and `potf2`.
- Those four routines have a StarPU (tiled) implementation.

Preliminary results with $n = 7680$ in single precision:

1. Using only CPU workers:

MKL potrf: 589.0 GFLOPS,
StarPU Cholesky example: 519.3 GFLOPS,
⟨T⟩LAPACK potrf: 477.9 GFLOPS.

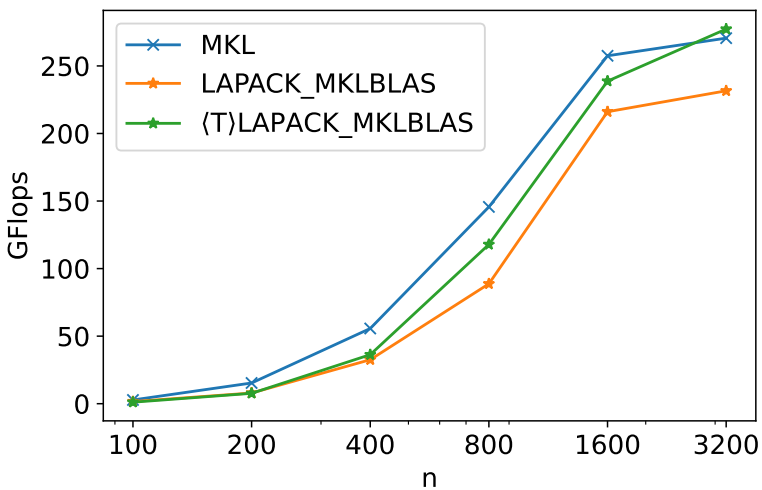
2. Using GPU and CPU workers:

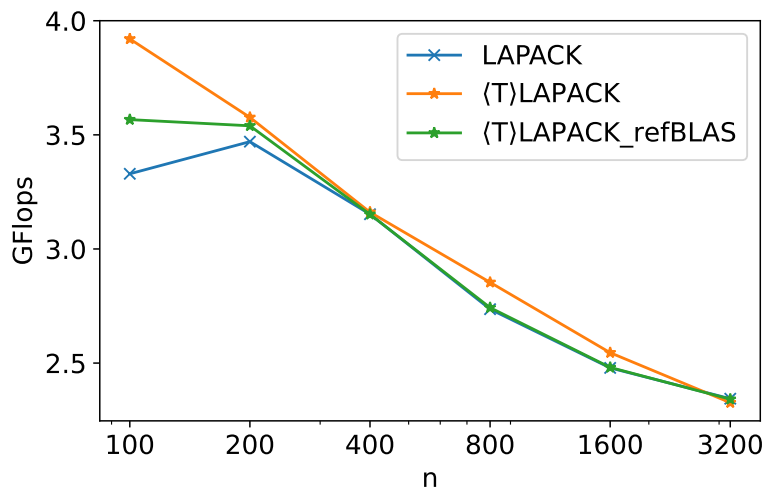
StarPU Cholesky example: 1835.1 GFLOPS,
⟨T⟩LAPACK potrf: 1445.6 GFLOPS.

Ubuntu 20.04.6 LTS x86_64. 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz (8 cores). NVIDIA GeForce RTX 3050 (2048 cores, 4GB). Driver version: 530.30.02. CUDA 11.0 (cuBLAS 11.2.0 cuSOLVER 10.6.0). MKL 2023.1.0. GNU compilers, version 9.4.0. StarPU scheduler: dmdas.

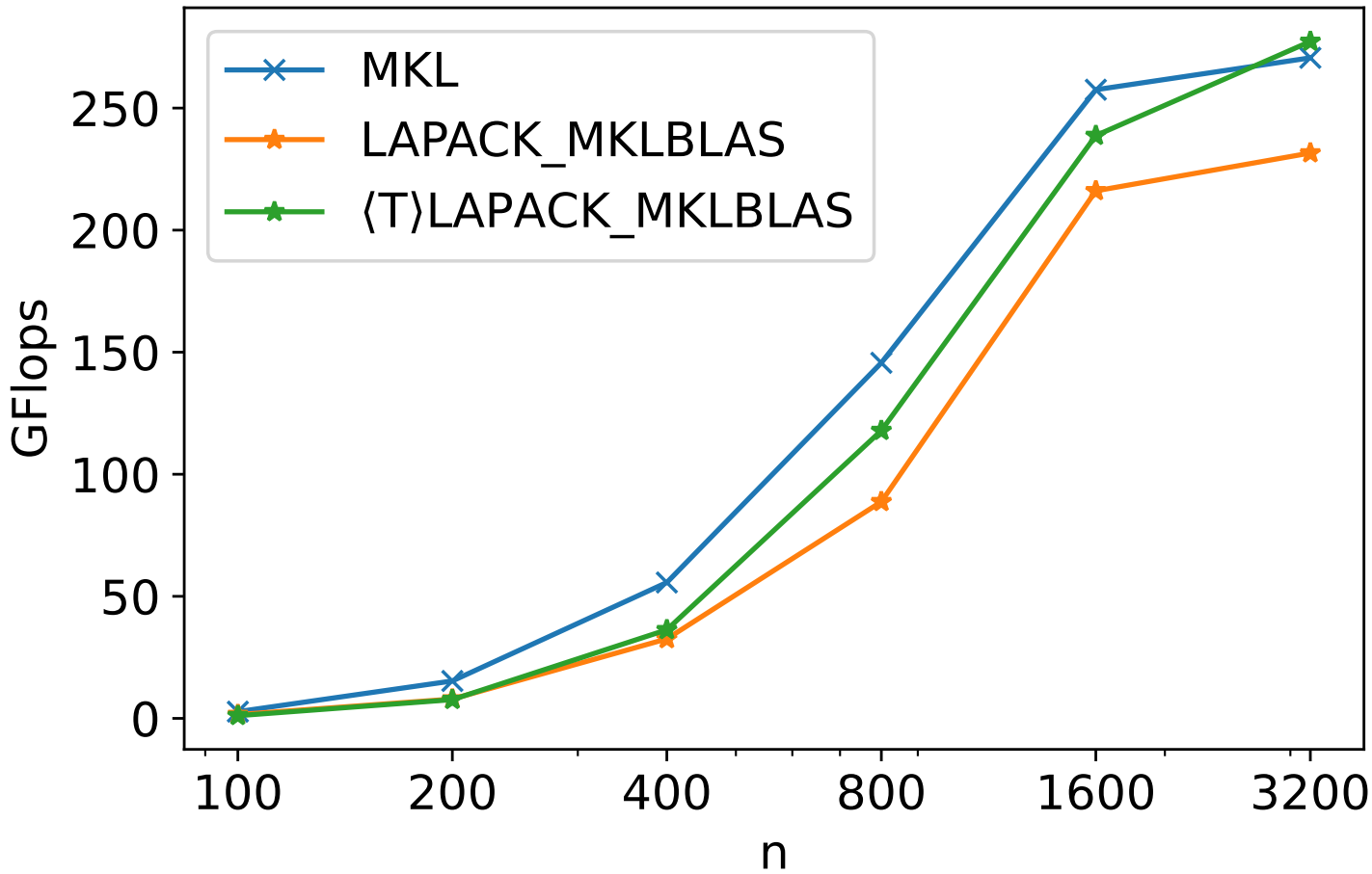
Performance test with recursive Cholesky

- 11th Gen i7-11800H @ 2.30GHz, 16 cores
- gcc version 9.3.0 (Ubuntu 9.3.0-17ubuntu1 20.04)
- LAPACK version 3.9.0
- oneAPI/MKL version 2022.0.2
- Main code: `github.com/wesslleyspereira/tlapack/blob/try-performance-example/examples/potrf/example_potrf.cpp`
- Double matrix of size n-by-n

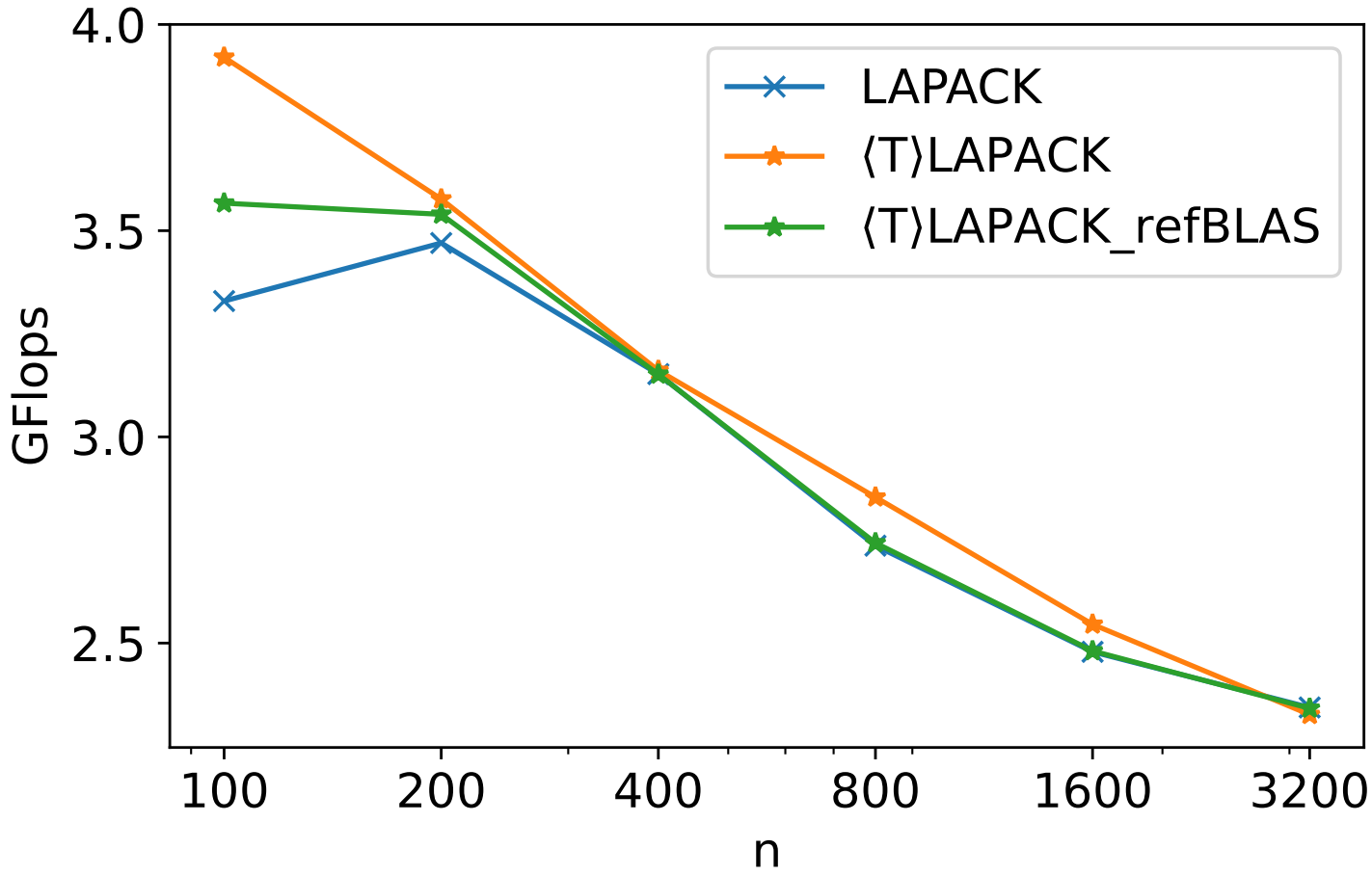




Performance test with recursive Cholesky



Performance test with recursive Cholesky



Performance of the real eigenvalue solver

Schur decomposition:

$$A = ZTZ^H$$

Two input scenarios:

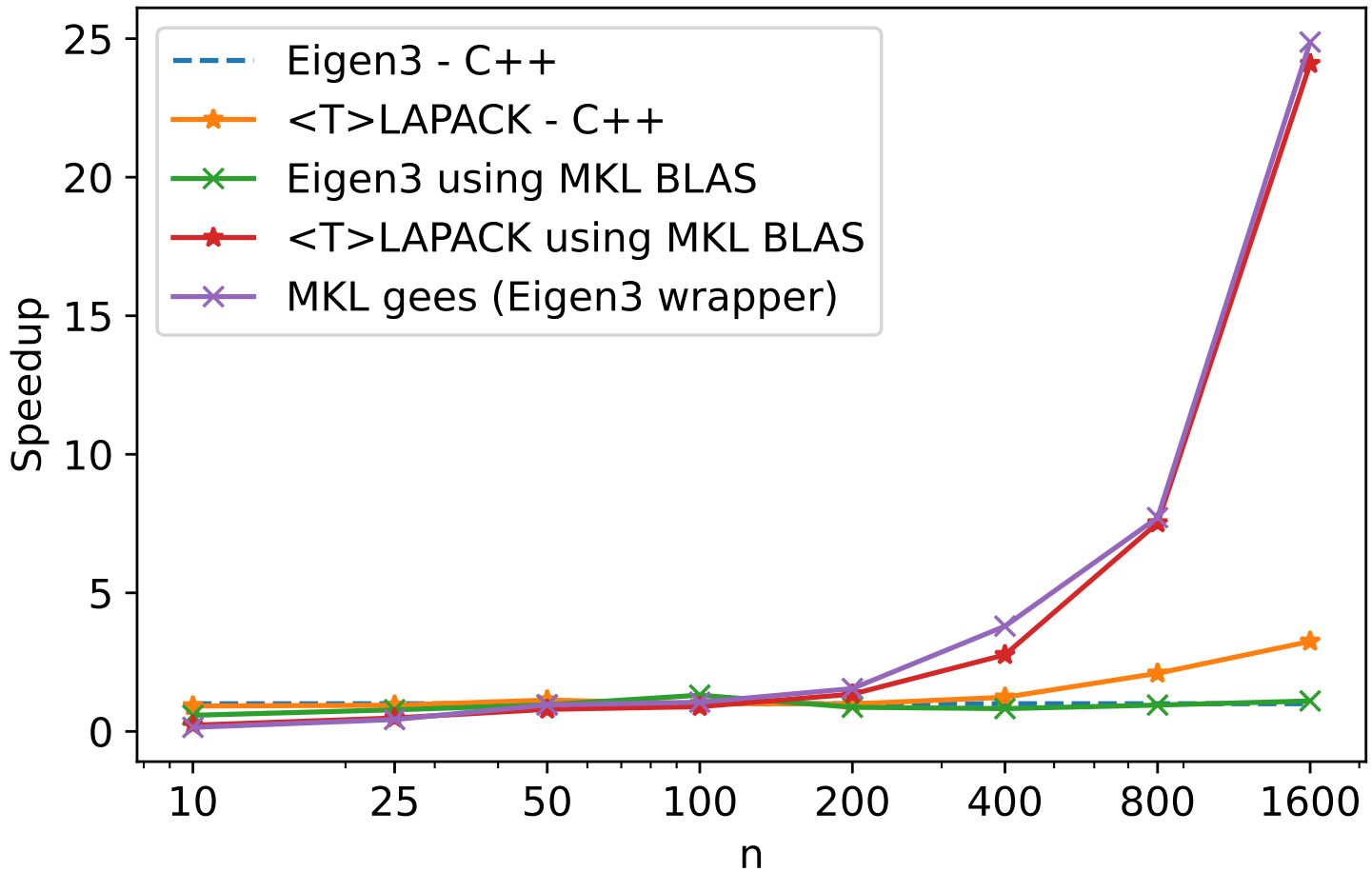
1. Random A, with entries taken uniformly in $[0, 1]$.
2. $A = (Q_1 D Q_2)\Lambda(Q_2^H D^{-1} Q_1^H)$, where
   - Λ is diagonal with entries taken uniformly in $[0, 1]$.
   - D is diagonal with $D_{ii} := 2^{(i \mod (54))}$.
   - Given A_i is random full rank, generate Q_i the Q-factor of the QR factorization of A_i, $i = 1, 2$.

Configurations in the following slides:

- Experiments in double precision.
- Package Eigen3 v3.3.7.
- Package mdspan v0.4.0.
- oneAPI/MKL version 2022.2.1.

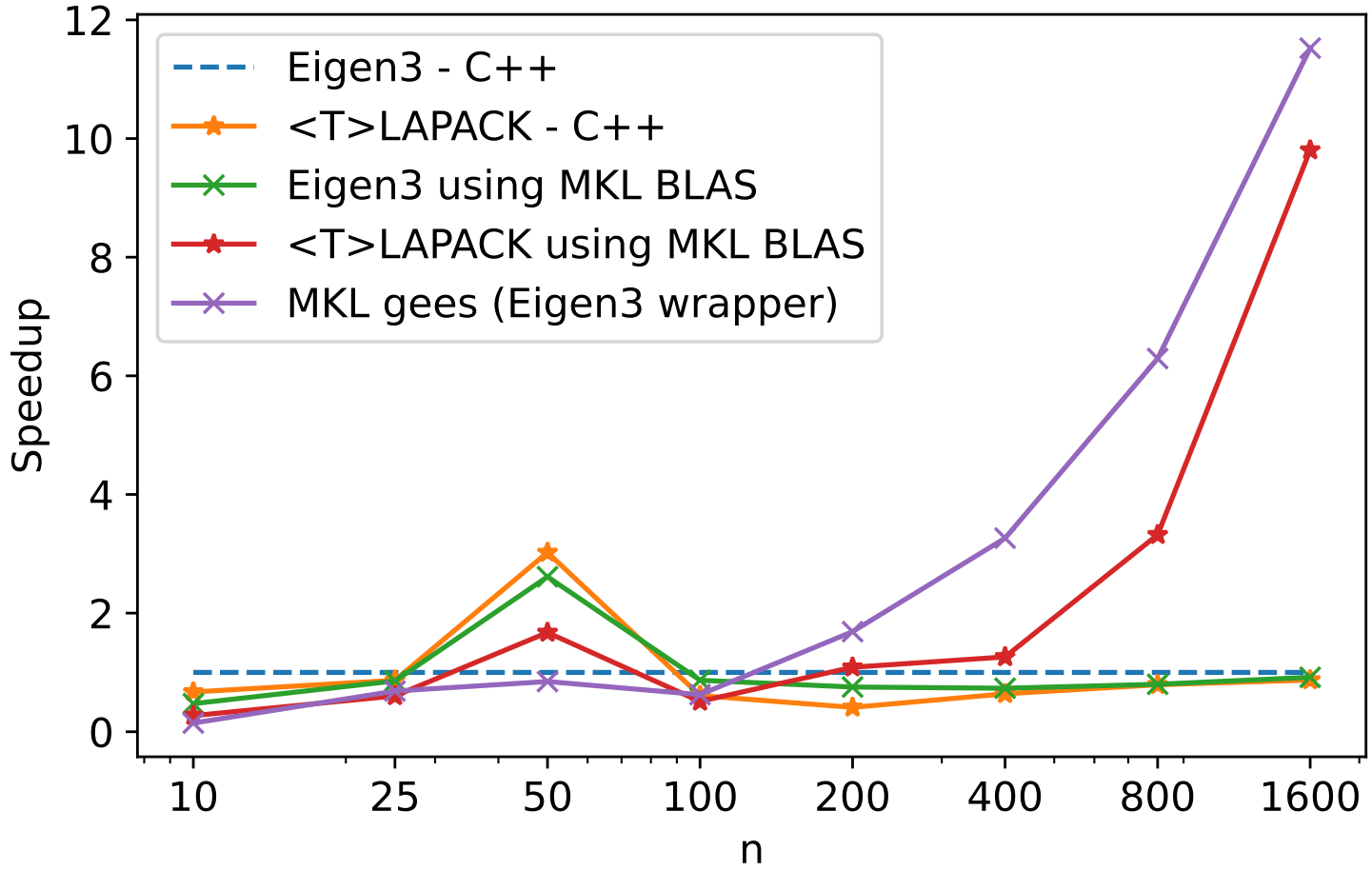
Performance in Scenario 1

Time to compute the eigenvalues compared to Eigen3.



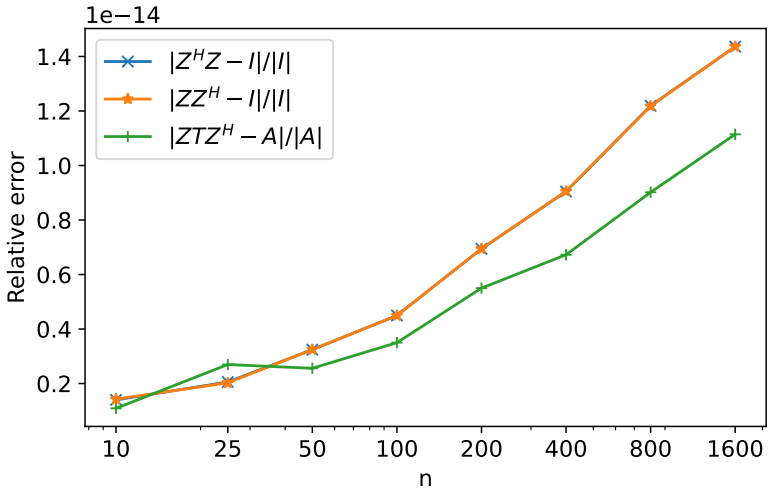
Performance in Scenario 2

Time to compute the eigenvalues compared to Eigen3.

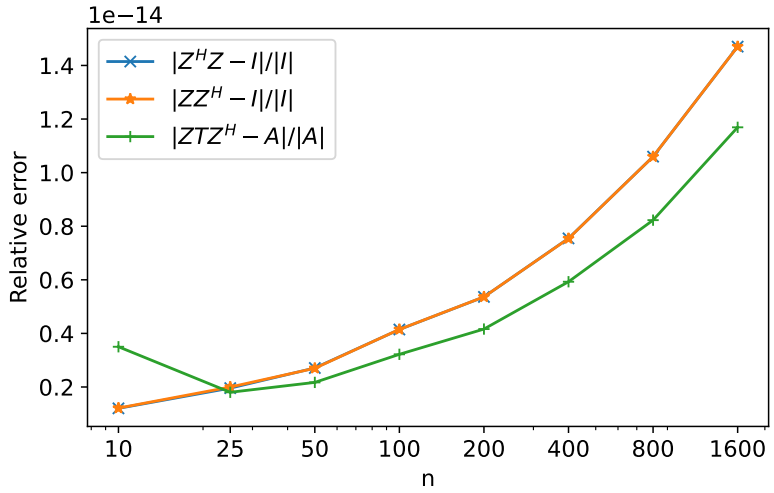


Stability analysis of the C++ codes

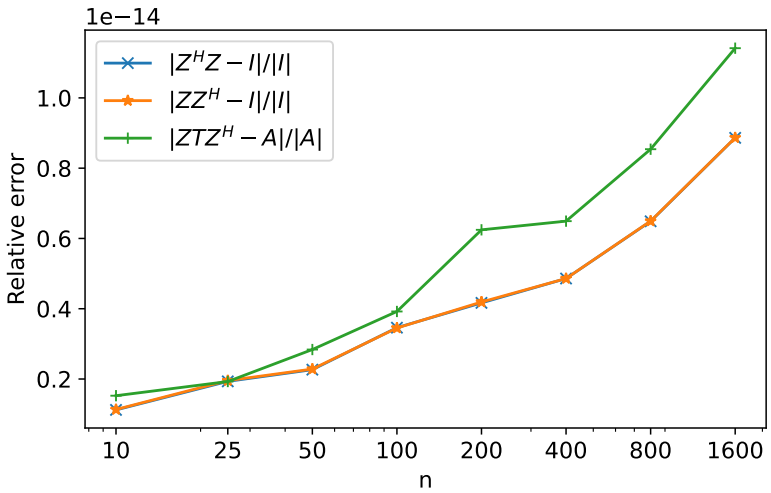
⟨T⟩LAPACK scenario 1



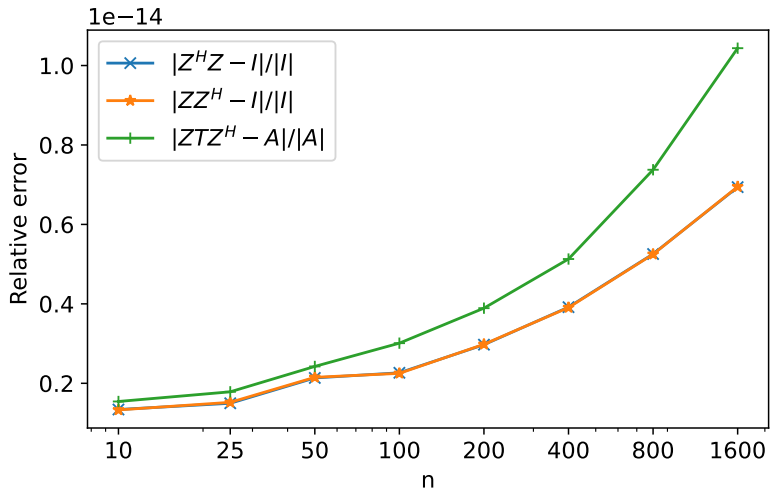
Eigen3 scenario 1



⟨T⟩LAPACK scenario 2

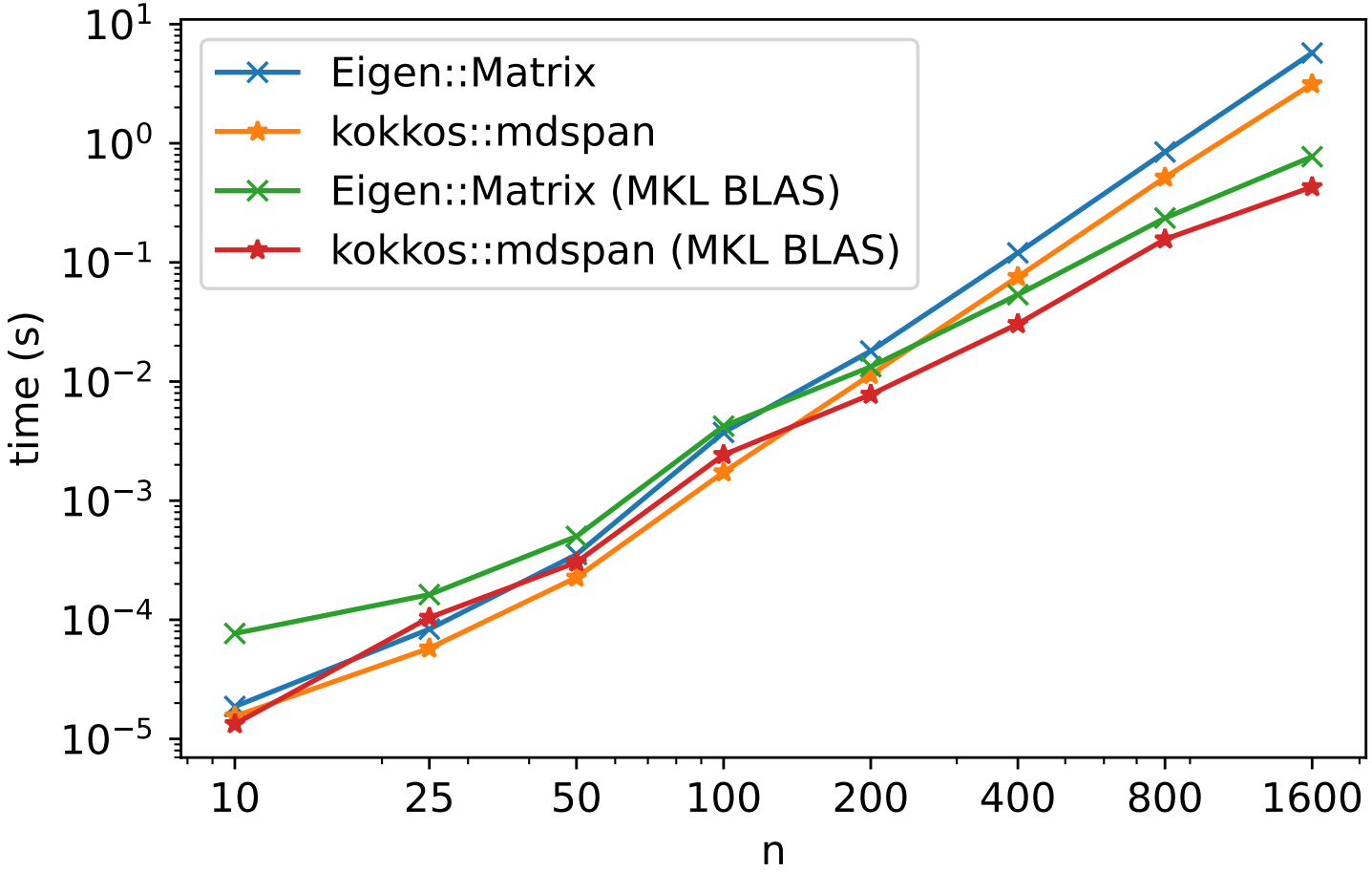


Eigen3 scenario 2



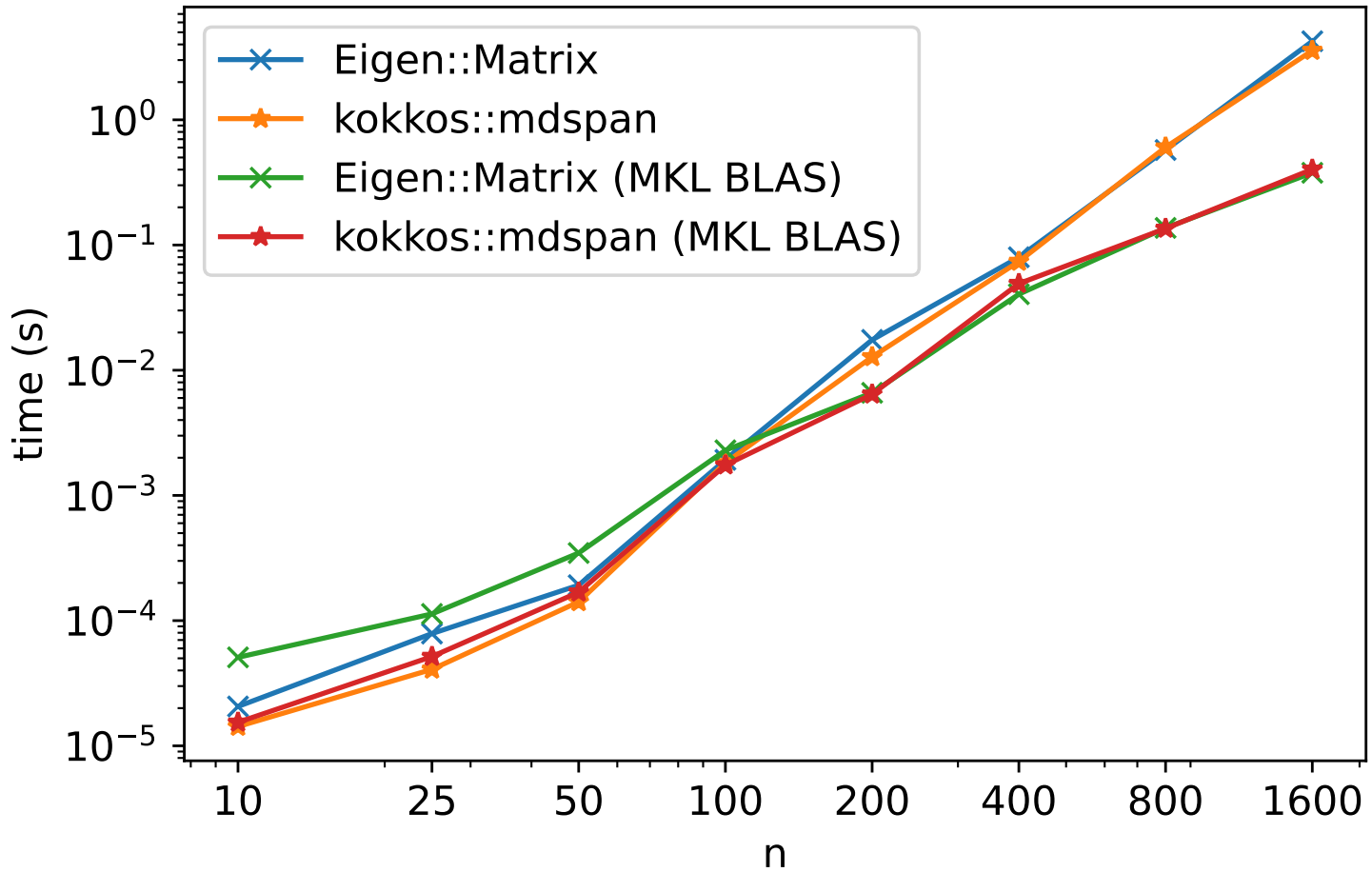
⟨T⟩LAPACK using mdspan from kokkos

Scenario 1 using different data structures with ⟨T⟩LAPACK



⟨T⟩LAPACK using mdspan from kokkos

Scenario 2 using different data structures with ⟨T⟩LAPACK



Code in the last example

Step-by-step (⟨T⟩LAPACK) x Straight-forward (Eigen3)

Example using ⟨T⟩LAPACK implementation:

```
// Hessenberg reduction
Eigen::Matrix<T,-1, 1> tau(n);
tlapack::gehrd( 0, n, A, tau );

// Discard reflectors
for (idx_t j = 0; j < n; ++j)
  for (idx_t i = j + 2; i < n; ++i)
    A(i,j) = T(0);

// Compute eigenvalues and Schur decomposition
Eigen::Matrix<std::complex<T>,-1,1> lambda(n);
Eigen::Matrix<T,-1,-1> Z(n,n); Z.setIdentity();
tlapack::multishift_qr( false, false, 0, n, A, lambda, Z );
```

Example using Eigen3 implementation:

```
Eigen::EigenSolver<decltype(A)> solver( A, false );
auto lambda = solver.eigenvalues();
```

Integration and Interaction

Continuous Testing for ⟨T⟩LAPACK

testBLAS: test corner cases and exception handling in the BLAS.

- Started in Jun 20, 2021.
- Used in Demmel, J. et.al. (2022). Proposed Consistent Exception Handling for the BLAS and LAPACK

BLAS++ and LAPACK++ testers for comparing with legacy code.

- Tests compare output and execution time with LAPACKE.

Tests inside the ⟨T⟩LAPACK project.

- Test different data types, layouts and matrix classes, e.g., legacyMatrix, Eigen::Matrix and mdspan.

Impact on other software

Related to Eigen:

- Commented on bug https://gitlab.com/libeigen/eigen/-/issues/408
- GCC complex was incompatible to Eigen::half. We proposed a PR that, we believe, is being evaluated: https://github.com/gcc-mirror/gcc/pull/8.

Related to StarPU:

- StarPU does not have the means to declare static constant codelets. https://github.com/starpu-runtime/starpu/pull/20.
- Bug on one of their examples: https://github.com/starpu-runtime/starpu/issues/18.
- Bug in cuSOLVER 11: https://github.com/starpu-runtime/starpu/issues/12. (Still need to report it)
- Makes StarPU compatible with CUDA 12: https://github.com/starpu-runtime/starpu/pull/10
- Task on elements: This is something outside StarPU in our opinion. They provide the means to do that. But they do not define a matrix class.

Related to SLATE, BLAS++, LAPACK++, and LAPACK:

- Test suite for Inf and NaN propagation and corner cases that works on both ⟨T⟩LAPACK and BLAS++: https://github.com/tlapack/testBLAS.
- Failures in the compilation of ⟨T⟩LAPACK with SLATE helps improving the latter, e.g., https://bitbucket.org/icl/lapackpp/pull-requests/14, https://github.com/icl-utk-edu/blaspp/pull/11, https://bitbucket.org/icl/blaspp/pull-requests/38.
- Reciprocal scaling: https://github.com/Reference-LAPACK/lapack/pull/749

Related to BLAS: • testBLAS to test and find inconsistencies in the use of the BLAS standard: https://arxiv.org/abs/2207.09281. Testing MKL, BLIS, OpenBLAS, and also BLAS++.

Happening Now

- Thijs Steel working on the symmetric eigenvalue solver. See `github.com/tlapack/tlapack/pulls`.
- Julien Langou, Johnathan Rhyne, Thijs Steel on SIAM LA'24. See `siam.org/conferences/cm/program/minitutorials/la24-minitutorials`
- Strategies for more sustainable software at NREL, running ⟨T⟩LAPACK in low precision. See `linkedin.com/jobs/search/?currentJobId=3826121686`. (Graduate Summer Internship)

Thank you!

Questions?

Weslley.daSilvaPereira@nrel.gov

Github repo:
`https://github.com/tlapack/tlapack`