faer-rs, a Linear Algebra Library For The Rust Programming Language

Sarah El Kazdadi

February 20, 2024

Why Rust?

- Memory safety
- Extremely competitive performance
- Generic SIMD support with runtime dispatch
- Robust error handling
- Excellent compiler errors and runtime diagnostic capabilities

Why Rust?

- Memory safety
- Extremely competitive performance
- Generic SIMD support with runtime dispatch
- Robust error handling
- Excellent compiler errors and runtime diagnostic capabilities
- 💖 Cargo my beloved 💖

Features

- Vectorized and optimized for x86 (AVX2/AVX512) and ARM64
- Fine control over multithreading
- Matrix arithmetic
- Matrix decompositions (Cholesky, QR, LU), with optional pivoting
- Simplicial and supernodal sparse decompositions
- Solving linear systems
- Computing the SVD/EVD (currently dense only)

Example (dense)

```rust
fn main() {
    let a = Mat::from_fn(4, 4, |i, j| i as f64 + j as f64);
    let b = mat![[1.0], [2.0], [3.0], [4.0]];

    // solving a linear system (selfadjoint)
    let lblt = a.lblt(faer::Side::Lower);
    let x = lblt.solve(&b);

    // [src/main.rs:65:5] (&a * &x - &b).norm_l2() = 0.0
    dbg!((&a * &x - &b).norm_l2());
}
```

Example (dense)

```
fn main() {
    let a = Mat::from_fn(4, 4, |i, j| i as f64 + j as f64);
    // computing eigenvalues
    let complex_eigenvalues = a.eigenvalues::<c64>();
    let real_eigenvalues = a.selfadjoint_eigenvalues(Lower);

    dbg!(&complex_eigenvalues);
    dbg!(&real_eigenvalues);
}
```

Example (sparse)

```rust
fn main() {
    let a = SparseColMat::<usize, f64>::try_new_from_triplets(
            4, 4,
            &[
                (0, 0, 10.0), (1, 1, 20.0), (2, 2, 30.0), (3, 3, 40.0),
                (0, 1, 3.0), (1, 0, 2.0), (3, 2, 1.0),
            ],
    ).unwrap();

    let b = mat![[1.0], [2.0], [3.0], [4.0]];

    // solving a linear system
    let lu = a.as_ref().sp_lu().unwrap();
    let x = lu.solve(&b);

    dbg!((&a * &x - &b).norm_l2());
}
```

Example (sparse)

```rust
fn main() {
    let a = SparseColMat::<usize, f64>::try_new_from_triplets(
            4, 4,
            &[
                (0, 0, 10.0), (1, 1, 20.0), (2, 2, 30.0), (3, 3, 40.0),
                (0, 1, 3.0), (1, 0, 2.0), (3, 2, 1.0),
            ],
    ).unwrap();

    let b = mat![[1.0], [2.0], [3.0], [4.0]];

    // splitting up the solve into symbolic and numeric parts
    // the symbolic part is cheap to copy.
    let lu_symbolic = solvers::SymbolicLu::try_new(a.as_ref().symbolic()).unwrap();

    let lu_numeric = solvers::Lu::try_new_with_symbolic(lu_symbolic.clone(), a.as_ref()).unwrap();
    let x = lu_numeric.solve(&b);
    dbg!((&a * &x - &b).norm_l2());
}
```

Error diagnostic examples

Adding a matrix to a column

```
error[E0277]: the trait bound `Dense: MatAdd<DenseCol>` is not satisfied
  --> src/main.rs:17:9
   |
17 |     lhs + rhs
   |         ^ the trait `MatAdd<DenseCol>` is not implemented for `Dense`
   |
   = help: the trait `MatAdd<Dense>` is implemented for `Dense`
   = help: for that trait implementation, expected `Dense`, found `DenseCol`
   = note: required for `Matrix<DenseRef<'_, f64>>` to implement
`std::ops::Add<Matrix<DenseColRef<'_, f64>>>`
```

Error diagnostic examples

Returning a dangling reference to a local variable

```rust
fn returning_dangling_reference<'a>(lhs: MatRef<'a, f64>, rhs: MatRef<'a, f64>) -> MatRef<'a, f64> {
    let mut m = lhs + rhs;
    m.as_ref()
}
```

```
error[E0515]: cannot return value referencing local variable `m`
  --> src/main.rs:31:5
   |
31 |     m.as_ref()
   |     -^^^^^^^^^
   |     |
   |     returns a value referencing data owned by the current function
   |     `m` is borrowed here
```

Error diagnostic examples

Trying to read and mutate a matrix at the same time.

```rust
fn breaking_no_alias_guarantees(mut dst: MatMut<'_, c64>) {
    let a = dst.as_ref();
    faer_core::mul::matmul(
        dst.as_mut(), a, a.adjoint(), None, c64::new(1.0, 0.0), faer::Parallelism::None
    );
}
```

```
error[E0502]: cannot borrow `dst` as mutable because it is also borrowed as immutable
  --> src/main.rs:47:9
   |
45 |     let a = dst.as_ref();
   |             --- immutable borrow occurs here
46 |     faer_core::mul::matmul(
47 |         dst.as_mut(),
   |         ^^^^^^^^^^^^ mutable borrow occurs here
48 |         a,
   |         - immutable borrow later used here
```

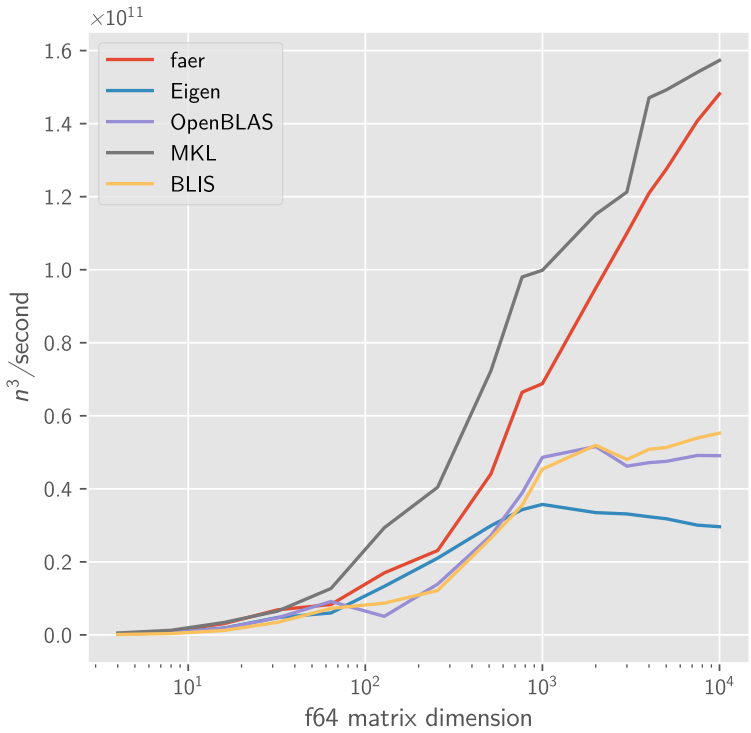
Error diagnostic examples

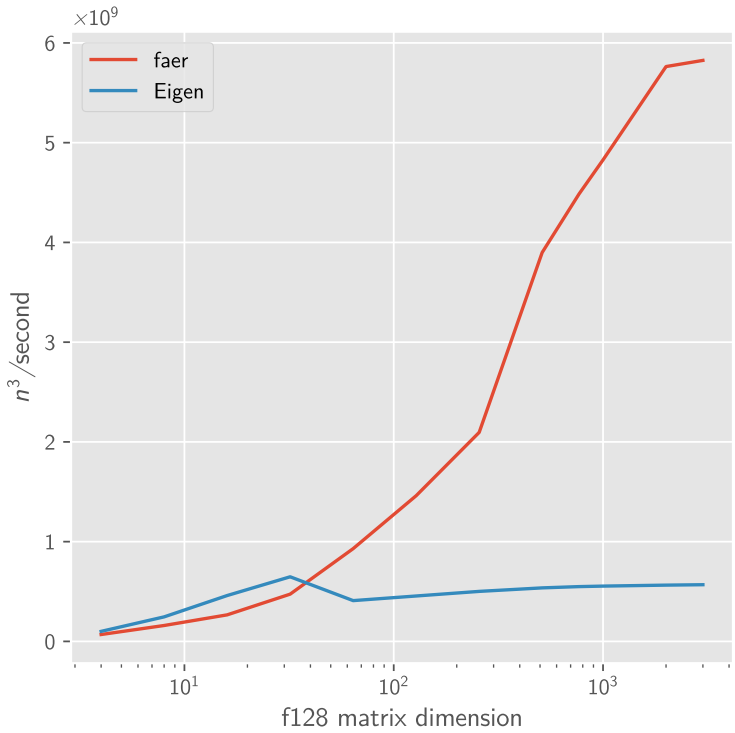
Mismatching dimensions at runtime.

```rust
let a = Mat::<f64>::zeros(5, 4);
let b = Mat::<f64>::zeros(4, 5);
let _ = &a + &b;
```

```
thread 'main' panicked at src/main.rs:127:17:
Assertion failed at /home/sarah/.cargo/git/checkouts/faer-rs-40fd231bb00bdbde/fa5e486/faer-libs/
faer-core/src/matrix_ops.rs:1522:9
Assertion failed: lhs.nrows() == rhs.nrows()
- lhs.nrows() = 5
- rhs.nrows() = 4
Assertion failed: lhs.ncols() == rhs.ncols()
- lhs.ncols() = 4
- rhs.ncols() = 5
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
```

Benchmarks (dense)





Benchmarks (sparse)

```
SSLSQ / NYPA_Maragal_6_lasso:
    Clarabel / QDLDL : 23.5s
    Clarabel / faer  : 4.13s
    MOSEK            : 4.73s

SSLSQ / ANSYS_Delor295K_huber:
    Clarabel / QDLDL : 8.47s
    Clarabel / faer  : 8.04s
    MOSEK            : 12.5s

NETLIB / nug15:
    Clarabel / QDLDL : 133s
    Clarabel / faer  : 16.5s
    MOSEK            : 9.44s
```

```
MAROS / EXDATA:
    Clarabel / QDLDL : 9.66s
    Clarabel / faer  : 2.58s
    MOSEK            : 2.34s (failed)

MAROS / CONT-300:
    Clarabel / QDLDL : 24.3s
    Clarabel / faer  : 11.8s
    MOSEK            : 5.96s

SUITESPARSE MATRIX COLLECTION / ND3K:
    SuiteSparse / CHOLMOD : 2.2s
    faer                  : 1.0s
    Eigen                 : ∞ (timeout)
```

Thank you!