**C++ code 9.8.28: Implementation of class KineticPropagator ↬ GitLab**

```
KineticPropagator::KineticPropagator(const SparseMatrixXd &A,
                                     const SparseMatrixXcd &M, double tau) {
  // Defeats the rationale of expression templates, but acceptable here, because
  // executed only once in the constructor.
  B_plus_ = M + 0.5 * tau * A.cast<std::complex<double>>();
  SparseMatrixXcd B_minus = M - 0.5 * tau * A.cast<std::complex<double>>();
  // This is the expensive step:  LU-factorization of a big sparse matrix.
  // Precomputation is essential
  solver_.compute(B_minus);
}

Eigen::VectorXcd KineticPropagator::operator()(
    const Eigen::VectorXcd &mu) const {
  // Cheap elimination steps operating on the LU-factors.  Effort is almost O(N)
  // thanks to sophisticated fill-in avoiding techniques employed by the sparse
  // solvers.
  return solver_.solve(B_plus_ * mu);
}
```