Numerical Linear Algebra

January 1, 2023

Matrix, View and Iterator

Object Oriented Representation

- Matrix: uniform type for matrices and vectors.
  - transparency - hidden implementation details
  - encapsulation - linear algebra functions
- View: lightwight “matrix”
- Expression: special type of views, syntactic sugar
- Iterator: efficient element access to contiguous memory

Matrix, View and Iterator

```
Matrix A(5,5);

// View: shared pointer to ``physical'' matrix A,
// but access only to the first row.
Matrix::View A1 = A.row(0);

// Matrix expressions: special type of views
// A*B: multiplication expression
// A*B + C: addition expression
Matrix D = A * B + C;

// A has contiguous memory and iterator support.
Matrix::iterator_support its = A.it_support();

// Iterators have efficient element access.
Matrix::iterator_traits::row_iterator it =
  its->row_begin(0);
```

Permutation

Row/column Interchanges

Let **P** be a permuation matrix, **M** be a regular matrix.

$$
\begin{aligned}
\mathbf{P} * \mathbf{M} &\rightarrow \text{row interchanged view of } \mathbf{M} \\
\mathbf{M} * \mathbf{P} &\rightarrow \text{column interchanged view of } \mathbf{M} \\
\mathbf{P}_n \cdots \mathbf{P}_1 * \mathbf{P}_0 * \mathbf{M} &\rightarrow \text{a sequence of row permutation on } \mathbf{M} \\
\mathbf{M} * \mathbf{P}_0 * \mathbf{P}_1 \cdots \mathbf{P}_n &\rightarrow \text{a sequence of column permutation on } \mathbf{M}
\end{aligned}
$$

Example

```
Matrix M(5,5);
// permutation of 2nd and 5th row/column interchange
Matrix::Permutation P(5, 1, 4);
Matrix::View PM=P * M;// row interchanged view of M
Matrix::View MP=M * P;// column interchanged view of M
```

Rotation

Givens

Let $\mathbf{G}$ be a Givens rotation matrix, $\mathbf{M}$ be a regular matrix.

$$
\begin{aligned}
\mathbf{G} * \mathbf{M} &\rightarrow \text{orthonormal transformation on 2 rows of } \mathbf{M} \\
\mathbf{M} * \mathbf{G} &\rightarrow \text{orthonormal transformation on 2 columns of } \mathbf{M} \\
\mathbf{G}_n \cdots \mathbf{G}_1 * \mathbf{G}_0 * \mathbf{M} &\rightarrow \text{a sequence of transformation on rows of } \mathbf{M} \\
\mathbf{M} * \mathbf{G}_0 * \mathbf{G}_1 \cdots \mathbf{G}_n &\rightarrow \text{a sequence of transformation on columns of } \mathbf{M}
\end{aligned}
$$

Example

```
Matrix M(5,5);
// rotation: r = sqrt(0.5^2 + 0.6^2),
// c = 0.6 / r, s = -0.6 / r.
Matrix::Givens G(1, 2, 0.5, 0.6);
G * M; // 2nd and 3rd rows of M are changed
M * G; // 2nd and 3rd columns of M are changed
```

Matrix Arithmetics and Factorization

Optimized BLAS/LAPACK (3rd party)

Multiplication, inverse, eigen, SVD, LU, QR

In-house Implementation

Full QR, Cholesky, reduced/modified/parital Cholesky

Matrix Arithmetics and Factorization

```
// LU decomposition of matrix A
Matrix L, U;
// P records the pivoting
Matrix::Permutation P;
A.lu(L, U, P);

Matrix B = P * L * U;
ftl::Assert(A.equals_to(B, 1.e-12));
```

Linear System

Different solvers based on various matrix forms, i.e. symmetric, positive definite, etc.

Optimized LAPACK (3rd party)

QR, SVD, LDL^T, LU, Cholesky

In-house Implementation

triangular based eliminations, tradiagonal, range space solver, conjugate gradient update, LU update

Linear System

```
// solve A x = b, A is a general matrix.
LinearSystem::solve(A, b, x, Matrix::GENERAL);

// solve A x = b, A is possibly a singular matrix.
LinearSystem::solve(A, b, x, Matrix::SINGULAR);

// solve A x = b, A is positive definite.
LinearSystem::solve(A, b, x, Matrix::POS_DEFINITE);
```

```
// Solve A x = b in conjugate gradient steps.
Matrix r = -b; // residual
Matrix p = b;  // step update

LinearSystem::cg_step_update(A, x, p, r);
LinearSystem::cg_step_update(A, x, p, r);
```