Numerical Linear Algebra

Copyright ⓒ Mathwrist LLC 2023

January 1, 2023

(Copyright ©Mathwrist LLC 2023)

Mathwrist Presentation Series

January 1, 2023

1/9

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 A(5,5);
1
3
   // View: shared pointer to ''physical'' matrix A,
4
   // but access only to the first row.
5
   Matrix::View A1 = A.row(0);
7
   // Matrix expressions: special type of views
8
   // A*B: multiplication expression
9
   // A*B + C: addition expression
10
   Matrix D = A * B + C;
12
   // A has contiguous memory and iterator support.
13
   Matrix::iterator_support its = A.it_support();
15
   // Iterators have efficient element access.
16
   Matrix::iterator_traits::row_iterator it =
17
     its->row_begin(0);
```

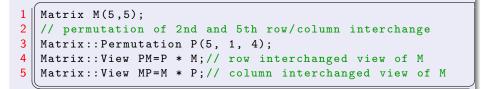
Permutation

Row/column Interchanges

Let \mathbf{P} be a permuation matrix, \mathbf{M} be a regular matrix.

P * M	\rightarrow	row interchanged view of ${f M}$
M * P	\rightarrow	column interchanged view of ${\bf M}$
$\mathbf{P}_n \cdots \mathbf{P}_1 * \mathbf{P}_0 * \mathbf{M}$	\rightarrow	a sequence of row permutation on ${\bf M}$
$\mathbf{M} * \mathbf{P}_0 * \mathbf{P}_1 \cdots \mathbf{P}_n$	\rightarrow	a sequence of column permutation on ${\bf M}$

Example



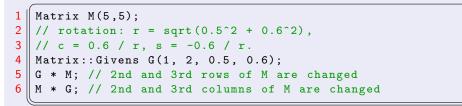
Rotation

Givens

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

${f G} * {f M} o$	orthonormal transformation on 2 rows of ${f M}$
$M * G \ o$	orthonormal transformation on 2 columns of ${\boldsymbol{M}}$
$\mathbf{G}_n \cdots \mathbf{G}_1 * \mathbf{G}_0 * \mathbf{M} ightarrow$	a sequence of transformation on rows of ${\bf M}$
$\mathbf{M} * \mathbf{G}_0 * \mathbf{G}_1 \cdots \mathbf{G}_n ightarrow$	a sequence of transformation on columns of ${\bf M}$

Example



Optimized BLAS/LAPACK (3rd party)

Multiplication, inverse, eigen, SVD, LU, QR

In-house Implementation

Full QR, Cholesky, reduced/modified/parital Cholesky

```
1 // LU decomposition of matrix A
2 Matrix L, U;
3 // P records the pivoting
4 Matrix::Permutation P;
5 A.lu(L, U, P);
7 Matrix B = P * L * U;
8 ftl::Assert(A.equals_to(B, 1.e-12));
```

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

```
1 // solve A x = b, A is a general matrix.
2 LinearSystem::solve(A, b, x, Matrix::GENERAL);
4 // solve A x = b, A is possibly a singular matrix.
5 LinearSystem::solve(A, b, x, Matrix::SINGULAR);
7 // solve A x = b, A is positive definite.
8 LinearSystem::solve(A, b, x, Matrix::POS_DEFINITE);
```

```
1 // Solve A x = b in conjugate gradient steps.
2 Matrix r = -b; // residual
3 Matrix p = b; // step update
5 LinearSystem::cg_step_update(A, x, p, r);
6 LinearSystem::cg_step_update(A, x, p, r);
```