Chapter 3

Class Documentation

3.1 Matrix Class Reference

`#include <Matrix.hpp>`

Public Member Functions

- Matrix (const Matrix &m)
- Matrix (int row, int col)
- Matrix (int n)
- ~Matrix ()
- void setValue (double value, int row, int col) const
- double & getValue (int row, int col) const
- int getNumRow () const
- int getNumCol () const
- double det () const
- Matrix & operator= (const Matrix &m)
- double & operator() (int i, int j)
- Matrix T () const
- double rowMax (int row, int col=0, bool abs_true=true) const
- double colMax (int col, int row=0, bool abs_true=true) const
- int argRowMax (int row, int col=0, bool abs_true=true) const
- int argColMax (int col, int row=0, bool abs_true=true) const
- void swapRow (int row1, int row2) const
- void swapCol (int col1, int col2) const
- void swap (int row1, int col1, int row2, int col2) const
- double norm (int p) const
- double norm (std::string p) const

Protected Attributes

- double * mData
- const int mRow
- const int mCol
- const int mSize

Friends

- Matrix operator+ (const Matrix &m1, const Matrix &m2)
- Matrix operator+ (const Matrix &m, const double &a)
- Matrix operator+ (const double &a, const Matrix &m)
- Matrix operator- (const Matrix &m1, const Matrix &m2)
- Matrix operator- (const Matrix &m, const double &a)
- Matrix operator- (const double &a, const Matrix &m)
- Matrix operator∗ (const Matrix &m1, const Matrix &m2)
- Matrix operator∗ (const Matrix &m, const double &a)
- Matrix operator∗ (const double &a, const Matrix &m)
- Matrix operator/ (const Matrix &m, const double &a)
- double dot (const Matrix &m1, const Matrix &m2)
- Matrix gaussianElimination (const Matrix &A_orig, const Matrix &b)
- Matrix cgs (const Matrix &A, const Matrix &b)
- Matrix operator/ (const Matrix b, const Matrix A)
- std::tuple< Matrix, Matrix, Matrix, int > lu (const Matrix &A)
- double det (const Matrix &A)
- std::tuple< Matrix, Matrix > qr (const Matrix &A, bool reduced=false)
- Matrix lsq (const Matrix &A, const Matrix &b)
- Matrix hessenbergReduction (const Matrix &A)
- Matrix eigenVal (const Matrix &A, double tol=1.0e-10)
- Matrix operator- (const Matrix &m)
- bool operator== (const Matrix &m1, const Matrix &m2)
- bool operator!= (const Matrix &m1, const Matrix &m2)
- Matrix transpose (const Matrix &m)
- const Matrix eye (int row, int col)
- const Matrix eye (int n)
- const Matrix rand (int row, int col)
- const Matrix rand (int n)
- void print (const Matrix &m)
- std::ostream & operator<< (std::ostream &output, const Matrix &m)
- double norm (Matrix &m, int p)
- double norm (Matrix &m, std::string p)
- int ∗ size (const Matrix &m)

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Matrix() [1/3]

```
Matrix::Matrix (
            const Matrix & m )
```

Copy constructor for the matrix class

Creates a copy of matrix object with the same dimensions and element values.

Parameters

m	The matrix object from which a copy is made

Returns

Matrix object with desired number of row and columns

3.1.1.2 Matrix() [2/3]

```
Matrix::Matrix (
            int row,
            int col )
```

Constructor for the matrix class

Creates a matrix object with the desired number of rows and columns with element values initialised as zero. Inputs row and col become attributes mRow and mCol. mSize := mRow ∗ mSize.

Parameters

row	The desired number of row in the matrix
col	The desired number of row in the matrix

Returns

Matrix object with desired number of row and columns

3.1.1.3 Matrix() [3/3]

```
Matrix::Matrix (
            int n )
```

Constructor for the matrix class

Creates a square matrix object with the desired number of rows and columns with element values initialised as zero. Input n become attributes mRow and mCol. mSize := mRow ∗ mSize.

Parameters

n	The desired number of rows and columns in the square matrix

Returns

Matrix object with desired number of row and columns

3.1.1.4 ~Matrix()

```
Matrix::~Matrix ( )
```

Destructor for the matrix class

Once a matrix object goes out of scope the destructor is called. It deletes the memory array used to store the matrix element data.

3.1.2 Member Function Documentation

3.1.2.1 argColMax()

```
int Matrix::argColMax (
            int col,
            int row = 0,
            bool abs_true = true ) const
```

Find argument of maximum value in column, by default this is the absolute max

Find column coordinate with maximum value in column, and optionally from a starting row (default is the first row).

Parameters

col	Column to check for max
row	Optional: Rows to start checking for max from
abs_true	Optional: return absolution max or regular max

Returns

Row position of column max value.

3.1.2.2 argRowMax()

```
int Matrix::argRowMax (
            int row,
            int col = 0,
            bool abs_true = true ) const
```

Find argument of maximum value in row, by default this is the absolute max

Find row coordinate with maximum value in column, and optionally from a starting column (default is the first column).

Parameters

row	Row to check for max
col	Optional: Columns to start checking for max from
abs_true	Optional: return absolution max or regular max

Returns

Column positon of row max value.

3.1.2.3 colMax()

```
double Matrix::colMax (
            int col,
            int row = 0,
            bool abs_true = true ) const
```

Find maximum value in column, by default this is the absolute max

Find maximum value in column, and optionally from a starting row (default is the first row).

Parameters

col	Column to check for max
row	Optional: Row to start checking for max from
abs_true	Optional: return absolution max or regular max

Returns

Row max value.

3.1.2.4 det()

```
double Matrix::det ( ) const
```

Calculate determinant of square matrix

Find the determinant of a matrix using LU decomposition

Returns

the determinant of self/this

3.1.2.5 getNumCol()

```
int Matrix::getNumCol ( ) const
```

Getter: Gets the number of columns in Matrix object.

Returns

The number of columns in matrix object

3.1.2.6 getNumRow()

```
int Matrix::getNumRow ( ) const
```

Getter: Gets the number of rows in Matrix object.

Returns

The number of rows in matrix object

3.1.2.7 getValue()

```
double & Matrix::getValue (
            int row,
            int col ) const
```

Getter: Used to get values of matrix elements.

Given valid coordinates this method gets the value at the coordinate location (using 0-indexing)

Parameters

row	The row position of the element to be returned
col	The row position of the element to be returned

Returns

Value of element at specified coordinate location

3.1.2.8 norm() [1/2]

```
double Matrix::norm (
            int p ) const
```

Calculates matrix norm

Calculates induces matrix 1-norm and 2-norm (for symmetric matrices). If matrices are vectors (1 row or col) calculates vector norm.

Parameters

p	choice of norm. For matrices this is 1 or 2-norm, for vectors this can be any positive integer

Returns

Norm calculate for self/this

3.1.2.9 norm() [2/2]

```
double Matrix::norm (
            std::string p ) const
```

Calculates matrix norm

Calculates induces matrix frobenius norm and infinity -norm .

Parameters

p	choice of norm.

Returns

Norm calculate for self/this

3.1.2.10 operator()()

```
double & Matrix::operator() (
            int i,
            int j )
```

Getter: Allows MATLAB like access of Matrix elements (1-indexing)

Allows matrix elements to be accessed and indexed in a MATLAB fashion i.e. A_ij = A(i,j)

Parameters

i	the row of desired element (in 1-indexing)
j	the column of the desired element (in 1-indexing)

Returns

The element at positon (i,j) (in 1-indexing)

3.1.2.11 operator=()

```
Matrix & Matrix::operator= (
            const Matrix & m )
```

Assignment operator

Assigns one matrix object to another, however, they are not linked. Essential creates a copy.

Parameters

m	Matrix object we wish to "copy" to self/this.

Returns

Updated self/this object with values of m.

3.1.2.12 rowMax()

```
double Matrix::rowMax (
            int row,
            int col = 0,
            bool abs_true = true ) const
```

Find maximum value in row, by default this is the absolute max

Find maximum value in row, and optionally from a starting column (default is the first column).

Parameters

row	Row to check for max
col	Optional: Column to start checking for max from
abs_true	Optional: return absolution max or regular max

Returns

Row max value.

3.1.2.13 setValue()

```
void Matrix::setValue (
            double value,
            int row,
            int col ) const
```

Setter: Used to set values of matrix elements.

Given a value and valid coordinates this method sets the value at the coordinate location to the given value.

Parameters

value	The value assign to the specified coordinates
row	The row position of the element to be updated
col	The row position of the element to be updated

3.1.2.14 swap()

```
void Matrix::swap (
            int row1,
            int col1,
            int row2,
            int col2 ) const
```

Swap two elements in matrix

Parameters

row1	Element 1 row coordinate
col1	Element 1 column coordinate
row2	Element 2 row coordinate
col2	Element 2 column coordinate

3.1.2.15 swapCol()

```
void Matrix::swapCol (
            int col1,
            int col2 ) const
```

Swap two columns in matrix

Parameters

col1	Row 1 position
col2	Row 2 position

3.1.2.16 swapRow()

```
void Matrix::swapRow (
            int row1,
            int row2 ) const
```

Swap two columns in matrix

Parameters

row1	Row 1 position
row2	Row 2 position

3.1.2.17 T()

```
Matrix Matrix::T ( ) const
```

Transposes matrix

Create transpose matrix by swapping rows and columns.

Returns

transposed matrix

3.1.3 Friends And Related Function Documentation

3.1.3.1 cgs

```
Matrix cgs (
            const Matrix & A,
            const Matrix & b )  [friend]
```

3.1.3.2 det

```
double det (
            const Matrix & A )  [friend]
```

Calculate determinant of square matrix

Find the determinant of a matrix using LU decomposition

Parameters

A	target matrix to find determinant of

Returns

the determinant of self/this

3.1.3.3 dot

```
double dot (
            const Matrix & m1,
            const Matrix & m2 )  [friend]
```

Dot product between two matrices (vectors) with one column or row

Performs the vector dot product on matrix or vector objects (i.e those with one column or row)

Parameters

m1	First vector (matrix)
m2	Second vector (matrix)

Returns

Dot product of vector m1 and vector m2.

3.1.3.4 eigenVal

```
Matrix eigenVal (
            const Matrix & A,
            double tol = 1.0e-10 )  [friend]
```

Finds eigenvalues of symmetric matrix using QR Algorithm

Uses QR Algorithm with shift, hessenberg reduction, and deflation to finds eigenvalues.

Parameters

A	Target matrix for eigenvalues
tol	Tolerance for when eigenvalue has been found. Determined by H(k,k-1) approx 0.

Returns

eigVals: Vecotr of eigenvalues of A.

3.1.3.5 eye [1/2]

```
const Matrix eye (
            int n )  [friend]
```

Square identity matrix constructor

Creates a square identity matrix with n rows and n columns.

Parameters

n	The desired number of rows and columns in the square identity matrix

Returns

Square identity matrix with desired number of row and columns

3.1.3.6 eye [2/2]

```
const Matrix eye (
            int row,
            int col )  [friend]
```

Identity matrix constructor

Creates an identity matrix with the desired number of rows and columns with ones on the diagonal and zeros everywhere else

Parameters

row	The desired number of row in the matrix
col	The desired number of row in the matrix

Returns

Matrix object with desired number of row and columns with ones on the diagonal

3.1.3.7 gaussianElimination

```
Matrix gaussianElimination (
            const Matrix & A_orig,
            const Matrix & b )  [friend]
```

Solves full-rank square system Ax = b with Gaussian Elimination.

Parameters

A	Matrix of the linear system
b	vector of the linear system

Returns

vector x: solution to the linear system

3.1.3.8 hessenbergReduction

```
Matrix hessenbergReduction (
            const Matrix & A )  [friend]
```

Finds upper Hessenberg reduction of matrix A

Compute the upper Hessenberg reduction of A using Householder reflections

Parameters

A	Target matrix for upper Hessenberg reduction

Returns

Matrix H, upper Hessenberg reduction of A.

3.1.3.9 lsq

```
Matrix lsq (
            const Matrix & A,
            const Matrix & b )  [friend]
```

Finds least squares solutio to overdetermined system |Ax-b|_2

It attempts to solve the least squares solution x that minimizes norm(b-A∗x,2). Uses reduced QR factorisation and then gaussian elimination.

Parameters

A	Overdetermined matrix in Ax = b
b	Vector b

Returns

Solution x to least-squares problem

3.1.3.10 lu

```
std::tuple<Matrix, Matrix, Matrix, int> lu (
            const Matrix & A )  [friend]
```

Compute LU decompositon with partial pivoting of square matrix

Parameters

A	Target matrix for LU decompositon

Returns

Matrix P - permutation matrix, Matrix L - lower triangular matrix Matrix U - upper triangular matrix, int sign - number of permutation made

3.1.3.11 norm [1/2]

```
double norm (
            Matrix & m,
            int p = 2 )  [friend]
```

3.1.3.12 norm [2/2]

```
double norm (
            Matrix & m,
            std::string p )  [friend]
```

3.1.3.13 operator"!=

```
bool operator!= (
            const Matrix & m1,
            const Matrix & m2 )  [friend]
```

non-equality check between two matrices

Overloads non-equality boolean operator to check non-equality element-wise between two matrices

Parameters

m1	First matrix
m2	Second matrix

Returns

True or false

3.1.3.14 operator* [1/3]

```
Matrix operator* (
            const double & a,
            const Matrix & m )  [friend]
```

Multiplication between doubles and Matrix object

Overloads multiplication operator to support element-wise multiplication of a double number.

Parameters

m	Matrix object
a	double number

Returns

Matrix object with element updated by multiplication of double value to element values,

3.1.3.15 operator* [2/3]

```
Matrix operator* (
            const Matrix & m,
            const double & a )  [friend]
```

Multiplication between doubles and Matrix object

Overloads multiplication operator to support element-wise multiplication of a double number.

Parameters

m	Matrix object
a	double number

Returns

Matrix object with element updated by multiplication of double value to element values,

3.1.3.16 operator* [3/3]

```
Matrix operator* (
            const Matrix & m1,
            const Matrix & m2 )  [friend]
```

Matrix-matrix multiplication

Overloads multiplication operator for matrix-matrix multiplication. Supports non-square matrices.

Parameters

m1	Left matrix
m2	Right matrix

Returns

Matrix multiplication of m1 * m2

3.1.3.17 operator+ [1/3]

```
Matrix operator+ (
            const double & a,
            const Matrix & m )  [friend]
```

Addition between doubles and Matrix object

Overloads addition operator to support element-wise addition of a double number.

Parameters

m	Matrix object
a	double number

Returns

Matrix object with element updated by addition of double value

3.1.3.18 operator+ [2/3]

```
Matrix operator+ (
            const Matrix & m,
            const double & a )  [friend]
```

Addition between doubles and Matrix object

Overloads addition operator to support element-wise addition of a double number.

Parameters

m	Matrix object
a	double number

Returns

Matrix object with element updated by addition of double value

3.1.3.19 operator+ [3/3]

```
Matrix operator+ (
            const Matrix & m1,
            const Matrix & m2 )  [friend]
```

Addition operator for matrix-matrix addition

Overloads addition operator to support addition of two matrix objects with the same dimensions

Parameters

m1	First matrix object
m1	Second matrix object

Returns

Matrix object that results from element-wise addition of m1 and m2.

3.1.3.20 operator- [1/4]

```
Matrix operator- (
            const double & a,
            const Matrix & m )  [friend]
```

Subtraction between doubles and Matrix object

Overloads subtraction operator to support element-wise subtraction of a double number.

Parameters

m	Matrix object
a	double number

Returns

Matrix object with element updated by addition of double value to negative element values, i.e. a - m[0,0]

3.1.3.21 operator- [2/4]

```
Matrix operator- (
            const Matrix & m )  [friend]
```

Return element-wise negative of input matrix

Overloads unary operator to perform element-wise negation.

Parameters

m	Input matrix

Returns

Negative of input matrix

3.1.3.22 operator- [3/4]

```
Matrix operator- (
            const Matrix & m,
            const double & a )  [friend]
```

Subtraction between doubles and Matrix object

Overloads subtraction operator to support element-wise subtraction of a double number.

Parameters

m	Matrix object
a	double number

Returns

Matrix object with element updated by subtraction of double from element values.

3.1.3.23 operator- [4/4]

```
Matrix operator- (
            const Matrix & m1,
            const Matrix & m2 )  [friend]
```

Subraction operator for matrix-matrix subtraction

Overloads subtraction operator to support subtraction of two matrix objects with the same dimensions

Parameters

m1	First matrix object
m1	Second matrix object

Returns

Matrix object that results from element-wise subtraction of m1 and m2.

3.1.3.24 operator/ [1/2]

```
Matrix operator/ (
            const Matrix & m,
            const double & a )  [friend]
```

Division between doubles and Matrix object

Overloads division operator to support element-wise division of a double number.

Parameters

m	Matrix object
a	double number

Returns

Matrix object with element updated by multiplication of double value to element values,

3.1.3.25 operator/ [2/2]

```
Matrix operator/ (
            const Matrix b,
            const Matrix A )  [friend]
```

3.1.3.26 operator<<

```
std::ostream& operator<< (
            std::ostream & output,
            const Matrix & m )  [friend]
```

Print with in std::cout;

Overloads << operator so that matrices and correctly formatted when using std::cout

Parameters

output	current stream to cout
m	matrix to be printed

Returns

Updated stream with matrix formating

3.1.3.27 operator==

```
bool operator== (
            const Matrix & m1,
            const Matrix & m2 )  [friend]
```

Equality check between two matrices

Overloads equaility boolean operator to check equality element-wise between two matrices

Parameters

m1	First matrix
m2	Second matrix

Returns

True or false

3.1.3.28 print

```
void print (
            const Matrix & m )  [friend]
```

Prints matrix object in formatted way

Parameters

m	Matrix to be printed

3.1.3.29 qr

```
std::tuple<Matrix, Matrix> qr (
            const Matrix & A,
            bool reduced = false )  [friend]
```

Calculates QR factorisation of matrix

Performs a qr factorisation on m-by-n matrix A such that A = Q∗R. The factor R is an m-by-n upper triangular matrix and Q is an m-by-m orthogonal matrix.

Parameters

A	Target matrix for QR factorisation
reduce	Optional: by default false, if true returns reduced QR

Returns

Matrix Q - orthogonal matrix, Matrix R - upper triangular

3.1.3.30 rand [1/2]

```
const Matrix rand (
            int n )  [friend]
```

Random square matrix

Creates square matrix with dimensions n x n, elements are given values sampled from uniform distribution between 0 and 1.

Parameters

n	Number of rows of square matrix

Returns

Matrix object with random element values between 0 and 1 on a uniform distribution.

3.1.3.31 rand [2/2]

```
const Matrix rand (
            int row,
            int col )  [friend]
```

Random matrix with desired number of rows and columns

Creates matrix with dimensions row x col, elements are given values sampled from uniform distribution between 0 and 1.

Parameters

row	Number of rows
col	Number of columns

Returns

Matrix object with random element values between 0 and 1 on a uniform distribution.

3.1.3.32 size

```
int* size (
            const Matrix & m )  [friend]
```

Gets number of rows and columns

Parameters

m	Matrix from which number of rows and columns want to be known

Returns

Array of size 2 [mRow, mCol];

3.1.3.33 transpose

```
Matrix transpose (
            const Matrix & m )  [friend]
```

Transposes matrix

Create transpose matrix by swapping rows and columns.

Parameters

m	Matrix to be transposed

Returns

transposed matrix

3.1.4 Member Data Documentation

3.1.4.1 mCol

```
const int Matrix::mCol  [protected]
```

3.1.4.2 mData

```
double* Matrix::mData  [protected]
```

3.1.4.3 mRow

```
const int Matrix::mRow  [protected]
```

3.1.4.4 mSize

```
const int Matrix::mSize  [protected]
```

The documentation for this class was generated from the following files:

- include/Matrix.hpp
- Matrix.cpp