matOps

Generated by Doxygen 1.9.1

1 Class Index 1

1.1 Class List . . . 1

2 File Index 2

2.1 File List . . . 2

3 Class Documentation 3

3.1 Matrix Class Reference . . . 3

3.1.1 Detailed Description . . . 4

3.1.2 Constructor & Destructor Documentation . . . 4

3.1.2.1 Matrix() [1/2] . . . 4

3.1.2.2 Matrix() [2/2] . . . 5

3.1.3 Member Function Documentation . . . 5

3.1.3.1 determinant() . . . 5

3.1.3.2 extractMatrix() . . . 6

3.1.3.3 hStack() . . . 6

3.1.3.4 insertCol() [1/2] . . . 7

3.1.3.5 insertCol() [2/2] . . . 7

3.1.3.6 insertRow() [1/2] . . . 8

3.1.3.7 insertRow() [2/2] . . . 8

3.1.3.8 inverse() . . . 9

3.1.3.9 operator()() . . . 9

3.1.3.10 operator*() [1/2] . . . 10

3.1.3.11 operator*() [2/2] . . . 10

3.1.3.12 operator+() [1/2] . . . 11

3.1.3.13 operator+() [2/2] . . . 11

3.1.3.14 operator-() [1/2] . . . 12

3.1.3.15 operator-() [2/2] . . . 12

3.1.3.16 operator/() . . . 13

3.1.3.17 operator==() . . . 13

3.1.3.18 shape() . . . 14

3.1.3.19 transpose() . . . 14

3.1.3.20 vStack() . . . 14

3.1.4 Friends And Related Function Documentation . . . 15

3.1.4.1 operator* . . . 15

3.1.4.2 operator+ . . . 15

3.1.4.3 operator- . . . 16

3.1.4.4 operator<< . . . 16

4 File Documentation 18

4.1 src/matOps.hpp File Reference . . . 18

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Matrix
A simple linear algebra library for matrix operations 3

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

src/matOps.hpp 18

Chapter 3

Class Documentation

3.1 Matrix Class Reference

A simple linear algebra library for matrix operations.

`#include <matOps.hpp>`

Public Member Functions

- std::pair< size_t, size_t > shape () const

  Returns the dimensions of the matrix.
- Matrix (const std::vector< std::vector< double >> &container)

  Constructs a Matrix from a given 2D vector container.
- Matrix (size_t rows, size_t cols, double initialValue)

  Constructs a Matrix with specified dimensions and an initial value.
- Matrix operator+ (const Matrix &other) const

  Adds two matrices element-wise.
- Matrix operator+ (double scalar) const

  Adds a scalar value to each element of the matrix. (MATRIX + K)
- Matrix operator- (const Matrix &other) const

  Subtracts one matrix from another element-wise.
- Matrix operator- (double scalar) const

  Subtracts a scalar from each element of the matrix. (MATRIX - K)
- bool operator== (const Matrix &other) const

  Compares two matrices for equality.
- Matrix operator∗ (const Matrix &other) const

  Multiplies two matrices.
- Matrix operator∗ (double scalar) const

  Multiplies each element of the matrix by a scalar. (MATRIX ∗ K)
- Matrix operator/ (double scalar) const

  Divides each element of the matrix by a scalar.
- double & operator() (size_t row, size_t col)

  Accesses an element of the matrix at a specified row and column.
- Matrix transpose () const

  Transposes the matrix.

- double determinant () const

  Computes the determinant of the matrix.
- Matrix inverse () const

  Computes the inverse of the matrix.
- Matrix insertRow (std::vector< double > row, size_t idx) const

  Inserts a new row into the matrix.
- Matrix insertRow (double rowVal, size_t idx) const

  Inserts a new row filled with a constant value into the matrix.
- Matrix insertCol (std::vector< double > col, size_t idx) const

  Inserts a new column into the matrix.
- Matrix insertCol (double colVal, size_t idx) const

  Inserts a new column filled with a constant value into the matrix.
- Matrix hStack (const Matrix &other) const

  Horizontally concatenates two matrices.
- Matrix vStack (const Matrix &other) const

  Vertically concatenates two matrices.
- Matrix extractMatrix (std::pair< size_t, size_t > rowSlice, std::pair< size_t, size_t > colSlice) const

  Extracts a submatrix from the current Matrix.

Friends

- Matrix operator+ (double scalar, const Matrix &other)

  Adds a scalar to each element of a matrix. (K + MATRIX)
- Matrix operator- (double scalar, const Matrix &other)

  Subtracts each element of the matrix from a scalar. (K - MATRIX)
- Matrix operator∗ (double scalar, const Matrix &other)

  Multiplies a scalar by a matrix. (K ∗ MATRIX)
- std::ostream & operator<< (std::ostream &os, const Matrix &m)

  Outputs the matrix to an output stream.

3.1.1 Detailed Description

A simple linear algebra library for matrix operations.

This class provides basic matrix operations such as addition, subtraction, multiplication, transposition, determinant calculation, inversion, and row/column insertion.

Example Usage:

```
Matrix A({{1, 2}, {3, 4}});
Matrix B({{5, 6}, {7, 8}});
Matrix C = A + B; // Matrix addition
Matrix D = A * B; // Matrix multiplication
double detA = A.determinant(); // Determinant calculation
```

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Matrix() [1/2]

```
Matrix::Matrix (
            const std::vector< std::vector< double >> & container )  [inline]
```

Constructs a Matrix from a given 2D vector container.

Parameters

container	A 2D vector of doubles representing the matrix.

Exceptions

std::invalid_argument	if the container is empty or row sizes are inconsistent.

Note

The shape is determined by the size of the container.

3.1.2.2 Matrix() [2/2]

```
Matrix::Matrix (
            size_t rows,
            size_t cols,
            double initialValue )  [inline]
```

Constructs a Matrix with specified dimensions and an initial value.

Parameters

rows	Number of rows.
cols	Number of columns.
initialValue	The value to initialize each element.

Exceptions

std::invalid_argument	if rows or cols are 0.

Note

The shape is (rows x cols).

3.1.3 Member Function Documentation

3.1.3.1 determinant()

```
double Matrix::determinant ( ) const  [inline]
```

Computes the determinant of the matrix.

Returns

The determinant as a double.

Exceptions

std::invalid_argument	if the matrix is not square.

Note

The shape of the matrix remains unchanged.

3.1.3.2 extractMatrix()

```
Matrix Matrix::extractMatrix (
            std::pair< size_t, size_t > rowSlice,
            std::pair< size_t, size_t > colSlice ) const  [inline]
```

Extracts a submatrix from the current Matrix.

Given a pair of row indices and a pair of column indices, this function creates and returns a new Matrix containing the submatrix defined by the specified ranges. Both row and column ranges are inclusive, meaning that the elements at both the start and end indices are included in the result.

Parameters

rowSlice	A std::pair<size_t, size_t> representing the start and end row indices (inclusive).
colSlice	A std::pair<size_t, size_t> representing the start and end column indices (inclusive).

Returns

A new Matrix object containing the extracted submatrix.

Exceptions

std::out_of_range	If any indices are out of bounds or if the slice ranges are invalid.

Note

Indices are zero-based (i.e., valid indices range from 0 to size() - 1).

3.1.3.3 hStack()

```
Matrix Matrix::hStack (
            const Matrix & other ) const  [inline]
```

Horizontally concatenates two matrices.

This function creates and returns a new Matrix by appending the columns of the given matrix to the right of the calling matrix. Both matrices must have the same number of rows.

Parameters

other	The Matrix whose columns will be appended to the calling matrix.

Returns

A new Matrix representing the horizontal concatenation of the two matrices.

Exceptions

std::invalid_argument	if the two matrices do not have the same number of rows.

Note

This implementation reserves the necessary capacity before inserting to minimize reallocations.

3.1.3.4 insertCol() [1/2]

```
Matrix Matrix::insertCol (
            double colVal,
            size_t idx ) const  [inline]
```

Inserts a new column filled with a constant value into the matrix.

Parameters

colVal	The constant value to fill the new column.
idx	The index at which to insert the column.

Returns

A new Matrix with the column inserted.

Exceptions

std::invalid_argument	if idx is out of range.

3.1.3.5 insertCol() [2/2]

```
Matrix Matrix::insertCol (
            std::vector< double > col,
            size_t idx ) const  [inline]
```

Inserts a new column into the matrix.

Parameters

col	A vector representing the new column.
idx	The index at which to insert the column.

Returns

A new Matrix with the column inserted.

Exceptions

std::invalid_argument	if the column size is inconsistent or if idx is out of range.

3.1.3.6 insertRow() [1/2]

```
Matrix Matrix::insertRow (
            double rowVal,
            size_t idx ) const  [inline]
```

Inserts a new row filled with a constant value into the matrix.

Parameters

rowVal	The constant value to fill the new row.
idx	The index at which to insert the row.

Returns

A new Matrix with the row inserted.

Exceptions

std::invalid_argument	if idx is out of range.

3.1.3.7 insertRow() [2/2]

```
Matrix Matrix::insertRow (
            std::vector< double > row,
            size_t idx ) const  [inline]
```

Inserts a new row into the matrix.

Parameters

row	A vector representing the new row.
idx	The index at which to insert the row.

Returns

A new Matrix with the row inserted.

Exceptions

std::invalid_argument	if the row size is inconsistent or if idx is out of range.

3.1.3.8 inverse()

```
Matrix Matrix::inverse ( ) const  [inline]
```

Computes the inverse of the matrix.

Returns

A new Matrix representing the inverse.

Exceptions

std::runtime_error	if the matrix is singular (non-invertible).

Note

The shape remains unchanged.

3.1.3.9 operator()()

```
double& Matrix::operator() (
            size_t row,
            size_t col )  [inline]
```

Accesses an element of the matrix at a specified row and column.

Parameters

row	The row index.
col	The column index.

Returns

Reference to the value at the specified position. (modifiable)

Exceptions

std::out_of_range	if the indices are out of bounds.

3.1.3.10 operator∗() [1/2]

```
Matrix Matrix::operator* (
            const Matrix & other ) const  [inline]
```

Multiplies two matrices.

Parameters

other	The Matrix to multiply with.

Returns

A new Matrix resulting from matrix multiplication.

Exceptions

std::invalid_argument	if the number of columns of the first matrix does not match the number of rows of the second.

Note

The shape of the result is (nrows of first, ncols of second).

3.1.3.11 operator∗() [2/2]

```
Matrix Matrix::operator* (
            double scalar ) const  [inline]
```

Multiplies each element of the matrix by a scalar. (MATRIX ∗ K)

Parameters

scalar	The scalar value.

Returns

A new Matrix with each element multiplied by the scalar.

Note

The shape remains unchanged.

3.1.3.12 operator+() [1/2]

```
Matrix Matrix::operator+ (
            const Matrix & other ) const  [inline]
```

Adds two matrices element-wise.

Parameters

other	The Matrix to add.

Returns

A new Matrix representing the element-wise sum.

Exceptions

std::invalid_argument	if the dimensions of the two matrices do not match.

Note

The shape remains unchanged.

3.1.3.13 operator+() [2/2]

```
Matrix Matrix::operator+ (
            double scalar ) const  [inline]
```

Adds a scalar value to each element of the matrix. (MATRIX + K)

Parameters

scalar	A double value to add.

Returns

A new Matrix with the scalar added to each element.

Note

The shape remains unchanged.

3.1.3.14 operator-() [1/2]

```
Matrix Matrix::operator- (
            const Matrix & other ) const  [inline]
```

Subtracts one matrix from another element-wise.

Parameters

other	The Matrix to subtract.

Returns

A new Matrix representing the element-wise difference.

Exceptions

std::invalid_argument	if the dimensions of the two matrices do not match.

Note

The shape remains unchanged.

3.1.3.15 operator-() [2/2]

```
Matrix Matrix::operator- (
            double scalar ) const  [inline]
```

Subtracts a scalar from each element of the matrix. (MATRIX - K)

Parameters

scalar	The scalar value to subtract.

Returns

A new Matrix with each element reduced by the scalar.

Note

The shape remains unchanged.

3.1.3.16 operator/()

```
Matrix Matrix::operator/ (
            double scalar ) const  [inline]
```

Divides each element of the matrix by a scalar.

Parameters

scalar	The scalar value.

Returns

A new Matrix with each element divided by the scalar.

Exceptions

std::runtime_error	if scalar is zero.

Note

The shape remains unchanged.

3.1.3.17 operator==()

```
bool Matrix::operator== (
            const Matrix & other ) const  [inline]
```

Compares two matrices for equality.

Parameters

other	The Matrix to compare with.

Returns

True if the matrices are equal (within a tolerance), false otherwise.

3.1.3.18 shape()

```
std::pair<size_t, size_t> Matrix::shape ( ) const  [inline]
```

Returns the dimensions of the matrix.

Returns

A std::pair where first is the number of rows and second is the number of columns.

3.1.3.19 transpose()

```
Matrix Matrix::transpose ( ) const  [inline]
```

Transposes the matrix.

Returns

A new matrix of dim (mxn) for a calling matrix of dim (nxm) Example:

```
Matrix A({{1, 2}, {3, 4}});
A = A.transpose();
// A becomes:
// [
//   [1, 3],
//   [2, 4]
// ]
```

3.1.3.20 vStack()

```
Matrix Matrix::vStack (
            const Matrix & other ) const  [inline]
```

Vertically concatenates two matrices.

This function creates and returns a new Matrix by appending the rows of the given matrix below the rows of the calling matrix. Both matrices must have the same number of columns.

Parameters

other	The Matrix whose rows will be appended to the calling matrix.

Returns

A new Matrix representing the vertical concatenation of the two matrices.

Exceptions

std::invalid_argument	if the two matrices do not have the same number of columns.

Note

The function appends each row of the second matrix to the container of the first, and updates the total row count accordingly.

3.1.4 Friends And Related Function Documentation

3.1.4.1 operator*

```
Matrix operator* (
            double scalar,
            const Matrix & other )  [friend]
```

Multiplies a scalar by a matrix. (K * MATRIX)

Parameters

scalar	The scalar value.
other	The Matrix to multiply.

Returns

A new Matrix with each element multiplied by the scalar.

Note

The shape remains unchanged.

3.1.4.2 operator+

```
Matrix operator+ (
            double scalar,
            const Matrix & other )  [friend]
```

Adds a scalar to each element of a matrix. (K + MATRIX)

Parameters

scalar	The scalar value.
other	The Matrix to add the scalar to.

Returns

A new Matrix with the result.

Note

The shape remains unchanged.

3.1.4.3 operator-

```
Matrix operator- (
            double scalar,
            const Matrix & other )  [friend]
```

Subtracts each element of the matrix from a scalar. (K - MATRIX)

Parameters

scalar	The scalar value.
other	The Matrix whose elements are subtracted from the scalar.

Returns

A new Matrix with the result.

Note

The shape remains unchanged.

3.1.4.4 operator<<

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

Outputs the matrix to an output stream.

Parameters

os	The output stream.
m	The Matrix to output.

Returns

A reference to the output stream.

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

- src/matOps.hpp

Chapter 4

File Documentation

4.1 src/matOps.hpp File Reference

```
#include <iostream>
#include <vector>
```

Include dependency graph for matOps.hpp: