MatrixClassLib Reference Manual

Generated by Doxygen 1.3.8

Wed Jan 5 18:32:25 2005

Contents

Chapter 1

The MatrixClass Library

1.1 Introduction

This class library is designed to make use of matrices and thier associated functions using the C++ programming enviromnent. It is designed to be as flexible as possible in both implementation and usage.

1.2 Implementation

The matrix classes and thier derivatives come in three main types:

- **Stand-alone** (e.g. **Matrix**(p. 44), **SquareMatrix**(p. 197), **RowVector**(p. 147) ...)
- **Alias** (e.g. **MatrixAlias**(p. 49), *SquareMartixAlias*, **RowVectorAlias**(p. 152) ...)
- **Constant alias** (e.g. **MatrixAliasConstant**(p. 57), *SquareMartixAliasConstant*, **RowVectorAliasConstant**(p. 157) ...)

1.2.1 Stand-Alone Classes

The stand-alone classes allocate and free thier own memory. They can be instantiated in many ways:

```
Matrix A(2, 2);                // 2x2 matrix
RowVector x(4);                // 4 element row vector
ColumnVector y(ptr_to_data, 5); // 5 element column vector initiated with the values in an array
Matrix B(A);                   // New matrix the same size as A, and initiated with the same values
```

1.2.2 Alias Classes

The alias classes can be used to mask another matrix class. This can be useful if you wanted to reference a matrix in a different way, but it can be particularly useful if you already have some data (say in an array), and you want to implement the functionality of the matrix class on that data. For example:

```
void foo(double * data, int size)
{
    ColumnVectorAlias cva(data, size); // Use the data pointer to act as the storage for the new vector

    // operations on cva...
}
```

Remember that you will be allocating your own memory for alias classes. In the above example the `data` pointer must point to at least `size`∗`sizeof(double)` bytes of allocated memory.

1.2.3 Constant Alias Classes

In the situation where you are provided with read only data the constant alias class can be used to provive read only functionality. For example:

```
void foo(const double * data, int size)
{
    RowVectorAliasConstant rvac(data, size); // Use the data pointer to act as read only storage for the new vector

    // read operations on rvac...
}
```

1.3 Usage

The matrix class library has been designed to be as intuative to use as possible. The matrix classes, and thier derived classes have certain operations available to them. Operatons are inherited or overridden in derived classes (whichever make most sense), so an operation like `transpose()` which is available in the **Matrix**(p. 44) class will also be available in its derived class **SquareMatrix**(p. 197). However, a function like `inverse()` will be available to the **SquareMatrix**(p. 197) class, but not to the ordinary **Matrix**(p. 44) class.

To use an operator for a matrix, just call the associated function:

```
Matrix A(3,3);
Matrix B(3,3);

A.rand();           // Randomise the values of A

B = A.transpose();  // B becomes the transposed matrix of A
```

Although the matrix operations look to be member functions of the **Matrix**(p. 44) class, the are in fact member *objects*. They are a special kind of object where each object has has its `operator()` method overloaded. These special member objects are called "Matrix Operators", and they all inherit from just two base classes: **MatrixReadOperator**(p. 77) and **MatrixWriteOperator**(p. 85). The classes derived from **MatrixReadOperator**(p. 77) provide the read functions for the **Matrix**(p. 44) classes, and (you've guessed it!) the **MatrixWriteOperator**(p. 85) classes provide the write functions.

Further derived classes, for example **SquareMatrix**(p. 197), inherit the **Matrix**(p. 44) operations from **Matrix**(p. 44), but also include their own set of operator classes derived from **SquareMatrixReadOperator**(p. 215) and **SquareMatrixWriteOperator**(p. 218). Both of these inherit from **MatrixReadOperator**(p. 77) and **MatrixWriteOperator**(p. 85) respectively.

Obviously the AliasConstant classes only contain read operators, while the other classes contain both read and write operators.

For an exmaple, we can look at using a **Matrix**(p. 44) class, and a more specific **SquareMatrix**(p. 197) class. The **SquareMatrix**(p. 197) class contains both the **Matrix**(p. 44) read and write operators, and also the **SquareMatrix**(p. 197) read and write operations.

```
Matrix m(10,5);               // 10x5 matrix
SquareMatrix sm(5);           // 5x5 square matrix

m.set(1.0);                   // make all elements in m equal 1.0
m = m.multiply(5.34);         // multiply all elements in m by 5.34 and assign that back to m
sm = sm.inverse();            // make sm equal to its own inverse
sm = m.inverse();             // ERROR! m is not a square matrix
```

1.3.1 Standard Operators

There are many standard operators that have been overloaded to allow for a more readable code when using the matrix class library. These operators include:

- **Round brackets ()** - for accessing matrix elements
- **Arithmetic operators +, -, *, /** - for performing aritthmetic operations with matricies
- **Assignment operator=** - for assigning one matrix's values (and size) to another

Other operators may be overloaded for specific (derived) matrix types.

Example:

```
// Implementing the state-space formula:
//
//    dx = Ax + Bu
//
ColumnVector stateEquation(ColumnVector x, ColumnVector u)
{
    Matrix A(3,3);           // A matrix
    Matrix B(3,2);           // B matrix
    ColumnVector dx(3);      // state derivative vector

    // Create A & B matrices
    A(0,0) = 5; A(0,1) = 17; A(0,2) = 0.034;
    A(1,0) = 2.4;  // etc...

    dx = (A * x) + (B * u);  // Perform state equation

    return dx;               // Return state vector
}
```

Todo

DONE Add rest of operators
Create exception class and replace error() functions.
change return type for write operators (to return *this)
Test constructors fully
Test copy constructors fully.
Test assignment operators fully
Test SubMatrix class fully
Test/fix luDecomp and related functons (and cofactor())

Chapter 2

MatrixClassLib Hierarchical Index

2.1 MatrixClassLib Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

MatrixClassLib Class Index

3.1 MatrixClassLib Class List

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

Chapter 4

MatrixClassLib File Index

4.1 MatrixClassLib File List

Here is a list of all files with brief descriptions:

Chapter 5

MatrixClassLib Page Index

5.1 MatrixClassLib Related Pages

Here is a list of all related documentation pages:

Chapter 6

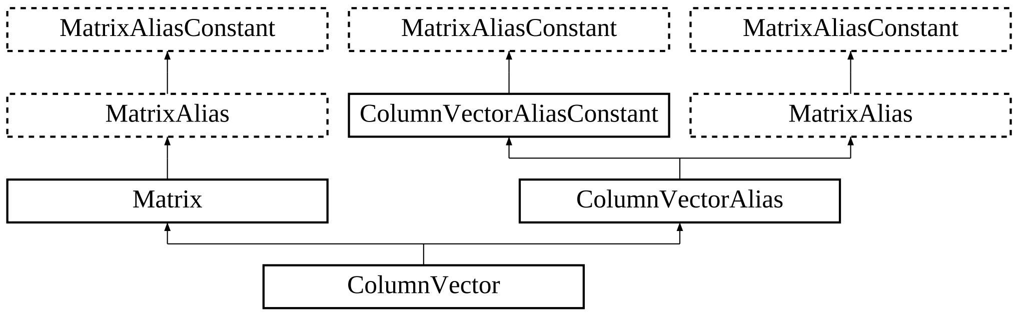
MatrixClassLib Class Documentation

6.1 ColumnVector Class Reference

The standard ColumnVector class.

`#include <matrix.h>`

Inheritance diagram for ColumnVector::



Public Member Functions

- **ColumnVector** (const unsigned int size)

  Sized constructor.

- **ColumnVector** (const **MatrixAliasConstant** &copy)

  Base class copy constructor.

- **ColumnVector** (**ColumnVector** &copy)

  Copy constructor.

- virtual ~**ColumnVector** ()

  ColumnVector Destructor.

- **ColumnVector** & **operator=** (const **MatrixAliasConstant** &copy)

  Base class assignment operator.

- **ColumnVector** & **operator=** (const **ColumnVector** &copy)

  Assignment operator.

Public Attributes

- **CVWO_EqualsElementCopyResize equals**

  Checks to see if the operand is compatable (i.e. a column vector) and then copies data in.

Protected Member Functions

- void **_constructColumnVector** (const unsigned int size)

  Sized constructor.

- void **_constructColumnVector** (const **MatrixAliasConstant** &copy)

  Copy constructor.

6.1.1 Detailed Description

The standard ColumnVector class.

Author:
Lee Netherton

The ColumnVector class provides the user with a pre made column vector. It will allocate its own memory, and is provided with a full complement of matrix and column vector operators.

Definition at line 1778 of file matrix.h.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 ColumnVector::ColumnVector (const unsigned int *size*) `[inline]`

Sized constructor.

Creates a new column vector of a given size.

Parameters:
size Number of rows matrix has

Definition at line 1796 of file matrix.h.

```
1796 {_constructColumnVector(size);}
```

6.1.2.2 ColumnVector::ColumnVector (const MatrixAliasConstant & *copy*) `[inline]`

Base class copy constructor.

Makes a copy of any another matrix.

Parameters:
copy Reaference to matrix to copy

Definition at line 1802 of file matrix.h.

```
1802 {_constructColumnVector(copy);}
```

6.1.2.3 ColumnVector::ColumnVector (ColumnVector & *copy*) `[inline]`

Copy constructor.

Makes a copy of another matrix.

Parameters:
copy Reaference to matrix to copy

Definition at line 1808 of file matrix.h.

```
1808 {_constructColumnVector(copy);}
```

6.1.2.4 ColumnVector::∼ColumnVector () `[virtual]`

ColumnVector Destructor.

Virtual - the lowest derived class will always need to be called, as they all allocate memory in ther own ways. Definition at line 1105 of file matrix.cpp.

```
1106 {
1107         #ifdef DEBUG_DESTRUCTOR
1108                 printf("Destructor: ~ColumnVector()\n");
1109         #endif
1110 }
```

6.1.3 Member Function Documentation

6.1.3.1 void ColumnVector::_constructColumnVector (const MatrixAliasConstant & *copy*) `[protected]`

Copy constructor.

Allocates some memory, and calls the **Matrix**(p. 44) constructor function and the **ColumnVectorAlias**(p. 20) blank constructor function. Finally, copys data in from copied matrix

Parameters:
copy **Matrix**(p. 44) to copy

Definition at line 1075 of file matrix.cpp.

```
{
        // Construct main base class
        _constructMatrix(copy.m_matrixContainer->getRows(),1);

        // Construct blank base class
        _constructColumnVectorAlias();

        // Construct operators
        _constructColumnVectorOperators();

        // Copy the information to this vector
        equals(copy);

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: ColumnVector::Copy Constructor\n");
        #endif
}
```

6.1.3.2 void ColumnVector::_constructColumnVector (const unsigned int *size*) [protected]

Sized constructor.

Allocates some memory, and calls the **Matrix**(p. 44) constructor function and the **Column-VectorAlias**(p. 20) blank constructor function.

Parameters:
size Number of rows matrix has

Definition at line 1056 of file matrix.cpp.

```
{
        // Construct main base class
        _constructMatrix(size,1);

        // Construct blank base class
        _constructColumnVectorAlias();

        // Construct operators
        _constructColumnVectorOperators();

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: ColumnVector::Sized Constructor\n");
        #endif
}
```

6.1.3.3 ColumnVector& ColumnVector::operator= (const ColumnVector & *copy*) [inline]

Assignment operator.

Definition at line 1855 of file matrix.h.

```
{return operator=((MatrixAliasConstant&)copy);}
```

6.1.3.4 ColumnVector& ColumnVector::operator= (const MatrixAliasConstant & *copy*) `[inline]`

Base class assignment operator.

Reimplemented from **Matrix** (p. 48).

Definition at line 1852 of file matrix.h.

```
1852 {equals(copy);return *this;}
```

6.1.4 Member Data Documentation

6.1.4.1 CVWO_EqualsElementCopyResize ColumnVector::equals

Checks to see if the operand is compatable (i.e. a column vector) and then copies data in.

Reimplemented from **Matrix** (p. 48).

Definition at line 1786 of file matrix.h.

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

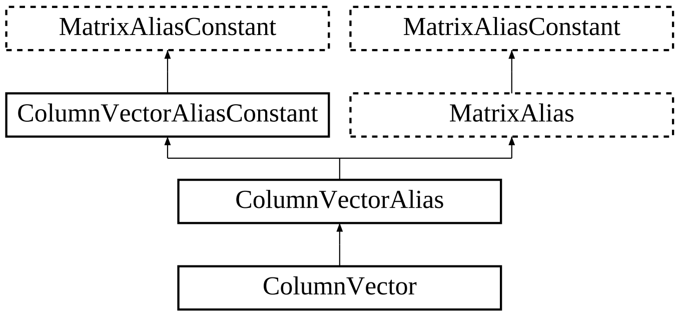
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix.cpp**

6.2 ColumnVectorAlias Class Reference

A ColumnVectorAlias class.

`#include <matrix.h>`

Inheritance diagram for ColumnVectorAlias::



Public Member Functions

- **ColumnVectorAlias** ()
  Default constructor.
- **ColumnVectorAlias** (const double ∗data, const unsigned int size)
  Pointer constructor.
- **ColumnVectorAlias** (const **MatrixAliasConstant** ∗alias)
  Alias constructor.
- **ColumnVectorAlias** (const **MatrixAliasConstant** &copy)
  Base class copy constructor.
- **ColumnVectorAlias** (const **ColumnVectorAlias** &copy)
  Copy constructor.
- virtual ∼**ColumnVectorAlias** ()
  ColumnVectorAlias Destructor.
- **ColumnVectorAlias** & **operator=** (const **MatrixAliasConstant** &copy)
  Base class assignment operator.
- **ColumnVectorAlias** & **operator=** (const **ColumnVectorAlias** &copy)
  Assignment operator.

Protected Member Functions

- void **_constructColumnVectorAlias** (const double ∗data, const unsigned int size)
  Pointer constructor.

- void **_constructColumnVectorAlias** (const **MatrixAliasConstant** &copy)

  Copy constructor.

- void **_constructColumnVectorAlias** ()

  Blank constructor.

6.2.1 Detailed Description

A ColumnVectorAlias class.

Author:
Lee Netherton

The ColumnVectorAlias class provides all the functionality from the **MatrixAlias**(p. 49) class, but add specific functions intended for column vectors. It also has specific column vector write functions.

Definition at line 1678 of file matrix.h.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 ColumnVectorAlias::ColumnVectorAlias () `[inline]`

Default constructor.

Creates a ColumnVectorAlias shell. The **MatrixContainer**(p. 70), **MatrixReadAccess**(p. 73) and **MatrixWriteAccess**(p. 81) handles can be set later using the constructor function **_-constructColumnVectorAlias()**(p. 23) Definition at line 1693 of file matrix.h.

```
1693 {}
```

6.2.2.2 ColumnVectorAlias::ColumnVectorAlias (const double ∗ *data*, const unsigned int *size*) `[inline]`

Pointer constructor.

To create a column vector that will access a pre-available data array.

Parameters:
data Pointer to data array. This will be the data storage for the matrix.

size Number of rows the vector has.

Definition at line 1700 of file matrix.h.

```
1700 {_constructColumnVectorAlias(data,size);}
```

6.2.2.3 ColumnVectorAlias::ColumnVectorAlias (const MatrixAliasConstant * *alias*) `[inline]`

Alias constructor.

To create a column vector that will alias another matrix.

Parameters:

alias Pointer to a matrix which this vector will alias.

Definition at line 1706 of file matrix.h.

```
1706 {_constructColumnVectorAlias(*alias);}
```

6.2.2.4 ColumnVectorAlias::ColumnVectorAlias (const MatrixAliasConstant & *copy*) `[inline]`

Base class copy constructor.

Used when creating a ColumnVectorAlias matrix from another matrix.

Parameters:

copy Reference to another matrix.

Definition at line 1712 of file matrix.h.

```
1712 {_constructColumnVectorAlias(copy);}
```

6.2.2.5 ColumnVectorAlias::ColumnVectorAlias (const ColumnVectorAlias & *copy*) `[inline]`

Copy constructor.

Used when creating a ColumnVectorAlias matrix from another. Calls base class copy constructor

Parameters:

copy Reference to another column vector.

Definition at line 1719 of file matrix.h.

```
1719 {_constructColumnVectorAlias(copy);}
```

6.2.2.6 ColumnVectorAlias::~ColumnVectorAlias () `[virtual]`

ColumnVectorAlias Destructor.

Virtual - the lowest derived class will always need to be called, as they all allocate memory in ther own ways. Definition at line 1035 of file matrix.cpp.

```
1036 {
1037         #ifdef DEBUG_DESTRUCTOR
1038                 printf("Destructor: ~ColumnVectorAlias()\n");
1039         #endif
1040 }
```

6.2.3 Member Function Documentation

6.2.3.1 void ColumnVectorAlias::_constructColumnVectorAlias () [protected]

Blank constructor.

Just constructs ColumnVectorAlias and **ColumnVectorAliasConstant**(p. 25) operators, and goes no further. Definition at line 1012 of file matrix.cpp.

```
{
        // Construct blank base classes and nothing else
        _constructColumnVectorAliasConstant();

        // Construct operators
        _constructColumnVectorAliasOperators();

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: ColumnVectorAlias::Blank Constructor\n");
        #endif
}
```

6.2.3.2 void ColumnVectorAlias::_constructColumnVectorAlias (const MatrixAliasConstant & *copy*) [protected]

Copy constructor.

Copies the pointers to the **MatrixContainer**(p. 70), **MatrixReadOperator**(p. 77) and **Matrix-WriteOperator**(p. 85) members Definition at line 993 of file matrix.cpp.

```
{
       // Construct main base class
       _constructMatrixAlias(copy.m_matrixContainer->getDataPointer(),copy.m_matrixContainer->getRows(),1);

       // Construct blank base class
       _constructColumnVectorAliasConstant();

        // Construct operators
        _constructColumnVectorAliasOperators();

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: ColumnVectorAlias::Copy Constructor\n");
        #endif
}
```

6.2.3.3 void ColumnVectorAlias::_constructColumnVectorAlias (const double * *data*, const unsigned int *size*) [protected]

Pointer constructor.

Sets the pointers to the **MatrixContainer**(p. 70), **MatrixReadOperator**(p. 77) and **Matrix-WriteOperator**(p. 85) members Definition at line 974 of file matrix.cpp.

```
{
       // Construct main base class
       _constructMatrixAlias(data,size,1);

       // Construct blank base class
       _constructColumnVectorAliasConstant();
```

```

        // Construct operators
        _constructColumnVectorAliasOperators();

        #ifdef DEBUG_CONSTRUCTOR
               printf("Constructed: ColumnVectorAlias::Pointer Constructor\n");
        #endif
}
```

6.2.3.4 ColumnVectorAlias& ColumnVectorAlias::operator= (const ColumnVectorAlias & *copy*) [inline]

Assignment operator.

Definition at line 1768 of file matrix.h.

```
{return operator=((MatrixAliasConstant&)copy);}
```

6.2.3.5 ColumnVectorAlias& ColumnVectorAlias::operator= (const MatrixAliasConstant & *copy*) [inline]

Base class assignment operator.

Reimplemented from **MatrixAlias** (p. 55).

Reimplemented in **ColumnVector** (p. 19).

Definition at line 1765 of file matrix.h.

```
{equals(copy);return *this;}
```

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

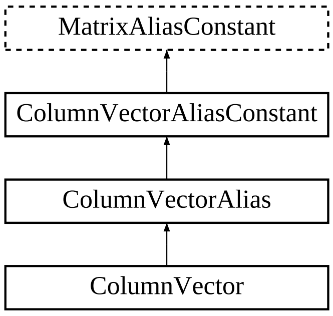
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix.cpp**

6.3 ColumnVectorAliasConstant Class Reference

A read-only **ColumnVectorAlias**(p. 20) class.

`#include <matrix.h>`

Inheritance diagram for ColumnVectorAliasConstant::



Public Member Functions

- **ColumnVectorAliasConstant** ()

  Default constructor.

- **ColumnVectorAliasConstant** (const double ∗data, const unsigned int size)

  Pointer constructor.

- **ColumnVectorAliasConstant** (const **MatrixAliasConstant** ∗alias)

  Alias constructor.

- **ColumnVectorAliasConstant** (const **MatrixAliasConstant** &copy)

  Base class copy constructor.

- **ColumnVectorAliasConstant** (const **ColumnVectorAliasConstant** &copy)

  Copy constructor.

- virtual ∼**ColumnVectorAliasConstant** ()

  ColumnVectorAliasConstant Destructor.

- **ColumnVectorAliasConstant** & **operator=** (const **MatrixAliasConstant** &copy)

  Base class assignment operator.

- **ColumnVectorAliasConstant** & **operator=** (const **ColumnVectorAliasConstant** &copy)

  Assignment operator.

Public Attributes

- **CVRO_CrossProduct cross**

  Returns the cross product of this vector.

- **CVRO_DotProduct dot**
  Returns the dot product of this vector.

- **CVRO_Modulus modulus**
  Returns the modulus of this vector.

Protected Member Functions

- void **_constructColumnVectorAliasConstant** (const double ∗data, const unsigned int size)
  Pointer constructor.

- void **_constructColumnVectorAliasConstant** (const **MatrixAliasConstant** &copy)
  Copy constructor.

- void **_constructColumnVectorAliasConstant** ()
  Blank constructor.

6.3.1 Detailed Description

A read-only **ColumnVectorAlias**(p. 20) class.

Author:
Lee Netherton

The ColumnVectorAliasConstant class provides all the functionality from the **MatrixAlias-Constant**(p. 57) class, but add specific functions intended for column vectors.

Definition at line 1567 of file matrix.h.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 ColumnVectorAliasConstant::ColumnVectorAliasConstant () `[inline]`

Default constructor.

Creates a ColumnVectorAliasConstant shell. The **MatrixContainer**(p. 70) and **MatrixRead-Access**(p. 73) handles can be set later using the constructor function **_constructColumn-VectorAliasConstant()**(p. 28) Definition at line 1594 of file matrix.h.

```
1594 {}
```

6.3.2.2 ColumnVectorAliasConstant::ColumnVectorAliasConstant (const double ∗ *data*, const unsigned int *size*) `[inline]`

Pointer constructor.

To create a read-only column vector that will access a pre-available data array.

Parameters:

data Pointer to data array. This will be the data storage for the matrix.

size Number of rows the matrix has.

Definition at line 1601 of file matrix.h.

```
1601 {_constructColumnVectorAliasConstant(data,size);}
```

6.3.2.3 ColumnVectorAliasConstant::ColumnVectorAliasConstant (const MatrixAliasConstant ∗ *alias*) `[inline]`

Alias constructor.

To create a read-only column vector that will alias another matrix.

Parameters:

alias Pointer to a matrix which this matris will alias.

Definition at line 1607 of file matrix.h.

```
1607 {_constructColumnVectorAliasConstant(*alias);}
```

6.3.2.4 ColumnVectorAliasConstant::ColumnVectorAliasConstant (const MatrixAliasConstant & *copy*) `[inline]`

Base class copy constructor.

Used when creating a ColumnVectorAliasConstant matrix from another.

Parameters:

copy Reference to another matrix.

Definition at line 1613 of file matrix.h.

```
1613 {_constructColumnVectorAliasConstant(copy);}
```

6.3.2.5 ColumnVectorAliasConstant::ColumnVectorAliasConstant (const ColumnVectorAliasConstant & *copy*) `[inline]`

Copy constructor.

Used when creating a **MatrixAliasConstant**(p. 57) matrix from another. Calls base class copy constructor

Parameters:

copy Reference to another matrix.

Definition at line 1620 of file matrix.h.

```
1620 {_constructColumnVectorAliasConstant(copy);}
```

6.3.2.6 ColumnVectorAliasConstant::∼ColumnVectorAliasConstant () `[virtual]`

ColumnVectorAliasConstant Destructor.

Virtual - the lowest derived class will always need to be called, as they all allocate memory in ther own ways. Definition at line 950 of file matrix.cpp.

```
{
        #ifdef DEBUG_DESTRUCTOR
                printf("Destructor: ~ColumnVectorAliasConstant()\n");
        #endif
}
```

6.3.3 Member Function Documentation

6.3.3.1 void ColumnVectorAliasConstant::_constructColumnVectorAliasConstant () `[protected]`

Blank constructor.

Just constructs ColumnVectorAliasConstant operators, and goes no further. Definition at line 926 of file matrix.cpp.

```
{
        // Construct operators
        _constructColumnVectorAliasConstantOperators();

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: ColumnVectorAliasConstant::Blank Constructor\n");
        #endif
}
```

6.3.3.2 void ColumnVectorAliasConstant::_constructColumnVectorAliasConstant (const MatrixAliasConstant & *copy*) `[protected]`

Copy constructor.

Copies the pointers to the **MatrixContainer**(p. 70) and **MatrixReadOperator**(p. 77) members Definition at line 910 of file matrix.cpp.

```
{
        // Construct main base class
        _constructMatrixAliasConstant(copy.m_matrixContainer->getDataPointer(),copy.m_matrixContainer->getRows(),1)

        // Construct operators
        _constructColumnVectorAliasConstantOperators();

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: ColumnVectorAliasConstant::Copy Constructor\n");
        #endif
}
```

6.3.3.3 void ColumnVectorAliasConstant::_constructColumnVectorAliasConstant (const double ∗ *data*, const unsigned int *size*) `[protected]`

Pointer constructor.

Sets the pointers to the **MatrixContainer**(p. 70) and **MatrixReadOperator**(p. 77) members Definition at line 894 of file matrix.cpp.

```
895 {
896         // Construct main base class
897         _constructMatrixAliasConstant(data,size,1);
898
899         // Construct operators
900         _constructColumnVectorAliasConstantOperators();
901
902         #ifdef DEBUG_CONSTRUCTOR
903                 printf("Constructed: ColumnVectorAliasConstant::Pointer Constructor\n");
904         #endif
905 }
```

6.3.3.4 ColumnVectorAliasConstant& ColumnVectorAliasConstant::operator= (const ColumnVectorAliasConstant & *copy*) `[inline]`

Assignment operator.

Definition at line 1667 of file matrix.h.

```
1667 {return operator=((MatrixAliasConstant&)copy);}
```

6.3.3.5 ColumnVectorAliasConstant& ColumnVectorAliasConstant::operator= (const MatrixAliasConstant & *copy*) `[inline]`

Base class assignment operator.

Reimplemented from **MatrixAliasConstant** (p. 66).

Reimplemented in **ColumnVectorAlias** (p. 24), and **ColumnVector** (p. 19).

Definition at line 1664 of file matrix.h.

```
1664 {m_matrixReadAccess->error("Tried to assign to a constant vector\n");return *this;}
```

6.3.4 Member Data Documentation

6.3.4.1 CVRO_CrossProduct ColumnVectorAliasConstant::cross

Returns the cross product of this vector.

Definition at line 1578 of file matrix.h.

6.3.4.2 CVRO_DotProduct ColumnVectorAliasConstant::dot

Returns the dot product of this vector.

Definition at line 1581 of file matrix.h.

6.3.4.3 CVRO_Modulus ColumnVectorAliasConstant::modulus

Returns the modulus of this vector.

Definition at line 1584 of file matrix.h.

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

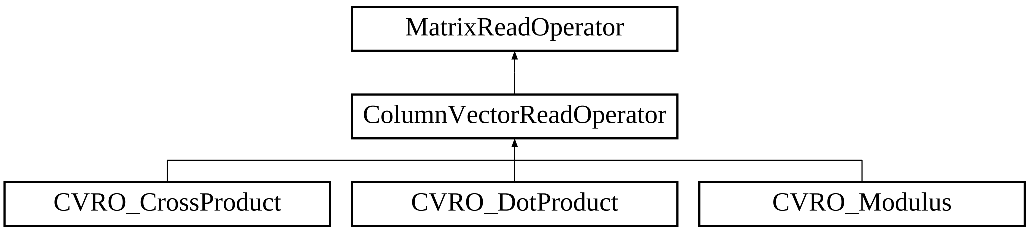
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix.cpp**

6.4 ColumnVectorReadOperator Class Reference

Base class for ColumnVectorReadOperators.

`#include <matrix_operator.h>`

Inheritance diagram for ColumnVectorReadOperator::



Public Member Functions

- **ColumnVectorReadOperator** ()

  Default Constructor.

- **ColumnVectorReadOperator** (**ColumnVectorAliasConstant** ∗columnVectorAlias-Constant)

  Full Constructor.

- void **_constructColumnVectorReadOperator** (**ColumnVectorAliasConstant** ∗columnVectorAliasConstant)

  Manual Constructor.

Protected Attributes

- **ColumnVectorAliasConstant** ∗ **m_thisMatrix**

  Pointer to owner matrix.

6.4.1 Detailed Description

Base class for ColumnVectorReadOperators.

Author:
Lee Netherton

Most importantly provides m_thisMatrix with the right kind of pointer.

Definition at line 329 of file matrix_operator.h.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 ColumnVectorReadOperator::ColumnVectorReadOperator () `[inline]`

Default Constructor.

Creates an empty operator class which can then be more full constructed using **_construct-ColumnVectorReadOperator()**(p. 32) Definition at line 341 of file matrix_operator.h.

```
341 {}
```

6.4.2.2 ColumnVectorReadOperator::ColumnVectorReadOperator (ColumnVectorAliasConstant ∗ *columnVectorAliasConstant*) `[inline]`

Full Constructor.

Creates an operator class which takes and stores a pointer to an owner matrix

Parameters:
columnVectorAliasConstant Pointer to owner matrix

Definition at line 347 of file matrix_operator.h.

```
347                                                                                            :
348                        MatrixReadOperator((MatrixAliasConstant *)columnVectorAliasConstant),
349                        m_thisMatrix(columnVectorAliasConstant)
350                        {}
```

6.4.3 Member Function Documentation

6.4.3.1 void ColumnVectorReadOperator::_constructColumnVectorReadOperator (ColumnVectorAliasConstant ∗ *columnVectorAliasConstant*) `[inline]`

Manual Constructor.

Constructs the class manually by setting the owner pointer

Parameters:
columnVectorAliasConstant Pointer to owner matrix

Definition at line 356 of file matrix_operator.h.

```
357                {
358                        _constructMatrixReadOperator((MatrixAliasConstant *)columnVectorAliasConstant);
359
360                        m_thisMatrix = columnVectorAliasConstant;
361                }
```

6.4.4 Member Data Documentation

6.4.4.1 ColumnVectorAliasConstant∗ ColumnVectorReadOperator::m_thisMatrix `[protected]`

Pointer to owner matrix.

Reimplemented from **MatrixReadOperator** (p. 80).

Definition at line 334 of file matrix_operator.h.

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

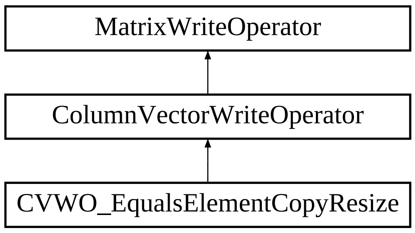
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**

6.5 ColumnVectorWriteOperator Class Reference

Base class for ColumnVectorWriteOperators.

`#include <matrix_operator.h>`

Inheritance diagram for ColumnVectorWriteOperator::



Public Member Functions

- **ColumnVectorWriteOperator** ()
  Default Constructor.
- **ColumnVectorWriteOperator** (**ColumnVectorAlias** ∗columnVectorAlias)
  Full Constructor.
- void **_constructColumnVectorWriteOperator** (**ColumnVectorAlias** ∗columnVector-Alias)
  Manual Constructor.

Protected Attributes

- **ColumnVectorAlias** ∗ **m_thisMatrix**
  Pointer to owner matrix.

6.5.1 Detailed Description

Base class for ColumnVectorWriteOperators.

Author:
Lee Netherton

Most importantly provides m_thisMatrix with the right kind of pointer.

Definition at line 370 of file matrix_operator.h.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 ColumnVectorWriteOperator::ColumnVectorWriteOperator () `[inline]`

Default Constructor.

Creates an empty operator class which can then be more full constructed using **_construct-ColumnVectorWriteOperator()**(p. 35) Definition at line 382 of file matrix_operator.h.

```
{}
```

6.5.2.2 ColumnVectorWriteOperator::ColumnVectorWriteOperator (ColumnVectorAlias * *columnVectorAlias*) [inline]

Full Constructor.

Creates an operator class which takes and stores a pointer to an owner matrix

Parameters:

columnVectorAlias Pointer to owner matrix

Definition at line 388 of file matrix_operator.h.

```
                                                                       :
                        MatrixWriteOperator((MatrixAlias *)columnVectorAlias),
                        m_thisMatrix(columnVectorAlias)
                        {}
```

6.5.3 Member Function Documentation

6.5.3.1 void ColumnVectorWriteOperator::_constructColumnVectorWriteOperator (ColumnVectorAlias * *columnVectorAlias*) [inline]

Manual Constructor.

Constructs the class manually by setting the owner pointer

Parameters:

columnVectorAlias Pointer to owner matrix

Definition at line 397 of file matrix_operator.h.

```
                {
                        _constructMatrixWriteOperator((MatrixAlias *)columnVectorAlias);

                        m_thisMatrix = columnVectorAlias;
                }
```

6.5.4 Member Data Documentation

6.5.4.1 ColumnVectorAlias* ColumnVectorWriteOperator::m_thisMatrix [protected]

Pointer to owner matrix.

Reimplemented from **MatrixWriteOperator** (p. 89).

Definition at line 375 of file matrix_operator.h.

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

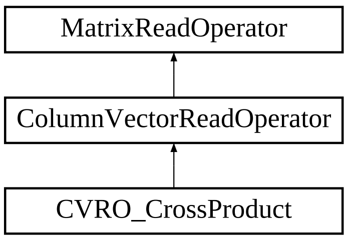
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**

6.6 CVRO_CrossProduct Class Reference

Returns the cross product of the vector and its operand.

`#include <matrix_operator.h>`

Inheritance diagram for CVRO_CrossProduct::



Public Member Functions

- **ColumnVector operator()** (const **ColumnVectorAliasConstant** &operand) const
  Returns the cross product of the vector and its operand.

6.6.1 Detailed Description

Returns the cross product of the vector and its operand.

Author:
Lee Netherton and Peter Mendham

Definition at line 1039 of file matrix_operator.h.

6.6.2 Member Function Documentation

6.6.2.1 ColumnVector CVRO_CrossProduct::operator() (const ColumnVectorAliasConstant & *operand*) const

Returns the cross product of the vector and its operand.

Parameters:
operand Operand to cross with.

Definition at line 1186 of file matrix_operator.cpp.

```
1186                                                                                {
1187
1188        ColumnVector result(getRows());
1189        SquareMatrix temp(getRows());
1190        unsigned int i;
1191
1192        if (operand.getRows() != getRows()) {
1193
1194                error("ColumnVectorAlias::cross : Dimensions are not consistent\n");
```

```

        } else if (operand.getRows() != 3) {

                error("ColumnVectorAlias::cross : Only vectors of length 3 are valid at this time\n");

        } else {

                for (i = 0; i < getRows(); i++) {
                        temp.element(1,i) = element(i);
                        temp.element(2,i) = operand.element(i);
                }

                for (i = 0; i < getRows(); i++) {
                        result.element(i) = temp.cofactor(0,i);
                }
        }

        return result;
}
```

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

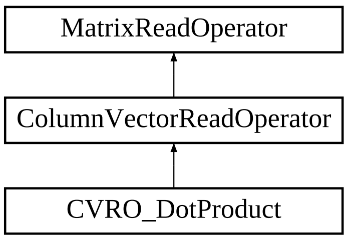
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.7 CVRO_DotProduct Class Reference

Returns the dot product of the vector and its operand.

`#include <matrix_operator.h>`

Inheritance diagram for CVRO_DotProduct::



Public Member Functions

- double **operator()** (const **ColumnVectorAliasConstant** &operand) const
  Returns the dot product of the vector and its operand.

6.7.1 Detailed Description

Returns the dot product of the vector and its operand.

Author:
Lee Netherton and Peter Mendham

Definition at line 1050 of file matrix_operator.h.

6.7.2 Member Function Documentation

6.7.2.1 double CVRO_DotProduct::operator() (const ColumnVectorAliasConstant & *operand*) const

Returns the dot product of the vector and its operand.

Parameters:
operand Operand to dot with.

Definition at line 1219 of file matrix_operator.cpp.

```
1219                                                                                           {
1220
1221         double result = 0;
1222         unsigned int i;
1223
1224         if (operand.getRows() != getRows()) {
1225
1226                 error("ColumnVectorAlias::dot : Dimensions are not consistent\n");
1227
```

```
        } else {

                for (i = 0; i < getRows(); i++) {
                        result += element(i) * operand.element(i);
                }
        }

        return result;
}
```

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

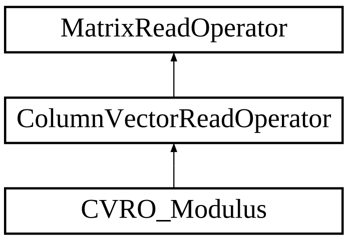
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.8 CVRO_Modulus Class Reference

Returns modulus of this vector.

`#include <matrix_operator.h>`

Inheritance diagram for CVRO_Modulus::



Public Member Functions

- double **operator()** () const

  Returns modulus of this vector.

6.8.1 Detailed Description

Returns modulus of this vector.

Author:
Lee Netherton and Peter Mendham

Definition at line 1061 of file matrix_operator.h.

6.8.2 Member Function Documentation

6.8.2.1 double CVRO_Modulus::operator() () const

Returns modulus of this vector.

Definition at line 1242 of file matrix_operator.cpp.

```
1242                                    {
1243
1244         unsigned int i;
1245         double mod = 0;
1246
1247         for (i = 0; i < getRows(); i++) {
1248                 mod += pow(element(i), 2);
1249         }
1250
1251         return sqrt(mod);
1252 }
```

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

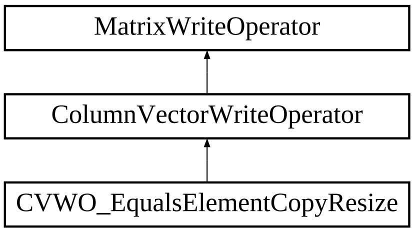
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.9 CVWO_EqualsElementCopyResize Class Reference

Copy elements (Resize if necessesary).

`#include <matrix_operator.h>`

Inheritance diagram for CVWO_EqualsElementCopyResize::



Public Member Functions

- void **operator()** (const **MatrixAliasConstant** &copy) const
  Checks to see if copy is a column vector, if so copies element in.

6.9.1 Detailed Description

Copy elements (Resize if necessesary).

Author:
Lee Netherton

Definition at line 1075 of file matrix_operator.h.

6.9.2 Member Function Documentation

6.9.2.1 void CVWO_EqualsElementCopyResize::operator() (const MatrixAliasConstant & *copy*) const

Checks to see if copy is a column vector, if so copies element in.

Parameters:
copy **Matrix**(p. 44) to copy (must be column vector)

Definition at line 1260 of file matrix_operator.cpp.

```
{
                unsigned int i;

                if(copy.isColumnVector() == 0)
                {
                        error("CVWO_EqualsElementCopyResize: Matrix to copy is not a column vector\n");
                        return;
                }


                if(copy.getRows() != getRows()) {
                        // Resize!!

                        // Change values
                        setRows(copy.getRows());

                        // Delete old memory
                        delete[] getDataPointer();

                        // Allocate new memory
                        setDataPointer(new double[getRows()]);
                }


                for (i = 0; i < getRows(); i++) {
                        element(i, 0) = copy.element(i, 0);
                }

}
```

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

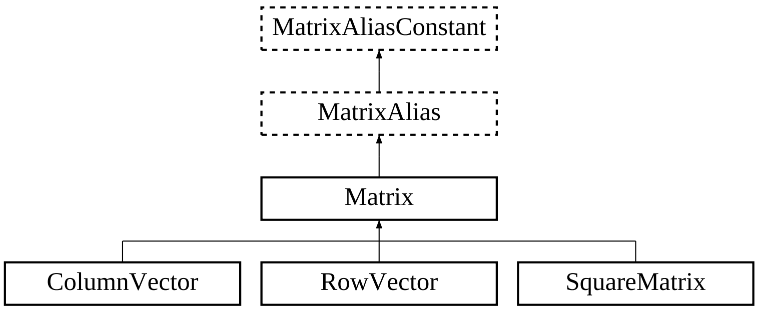
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.10 Matrix Class Reference

The standard matrix class.

`#include <matrix.h>`

Inheritance diagram for Matrix::



Public Member Functions

- **Matrix** ()

  Default constructor.

- **Matrix** (const unsigned int rows, const unsigned int columns)

  Sized constructor.

- **Matrix** (const **MatrixAliasConstant** &copy)

  Base class copy constructor.

- **Matrix** (const **Matrix** &copy)

  Copy constructor.

- virtual ~**Matrix** ()

  Matrix Destructor.

- **Matrix** & **operator=** (const **MatrixAliasConstant** &copy)

  Base class assignment operator.

- **Matrix** & **operator=** (const **Matrix** &copy)

  Assignment operator.

Public Attributes

- **MWO_EqualsElementCopyResize equals**

  Equals operator - allows matrix to resize if it is made equal to one of a different size.

Protected Member Functions

- void **_constructMatrix** (const unsigned int rows, const unsigned int columns)
  Sized constructor.
- void **_constructMatrix** (const **MatrixAliasConstant** &copy)
  Copy constructor.

6.10.1 Detailed Description

The standard matrix class.

Author:
Lee Netherton

The Matrix class provides the user with a pre made matrix. It will allocate its own memory, and is provided with a full complement of matrix operators.

For example:

```
Matrix A(3,3);        // Make a matrix called A
Matrix B(3,3);        // Make a matrix called B

// Fill the matrices with something useful

Matrix C = A * B;     // Matrix C is the product of the two matrices
```

Definition at line 586 of file matrix.h.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 Matrix::Matrix () `[inline]`

Default constructor.

Creates a **MatrixAlias**(p. 49) shell. The **MatrixContainer**(p. 70), **MatrixReadAccess**(p. 73), and **MatrixWriteAccess**(p. 81) handles can be set later using the constructor function **_-constructMatrixAlias()**(p. 54) Definition at line 606 of file matrix.h.

```
606 {}
```

6.10.2.2 Matrix::Matrix (const unsigned int *rows*, const unsigned int *columns*) `[inline]`

Sized constructor.

Creates a new matrix of a given size.

Parameters:
rows Number of rows matrix has
columns Number of columns matrix has

Definition at line 613 of file matrix.h.

```
613 {_constructMatrix(rows, columns);}
```

6.10.2.3 Matrix::Matrix (const MatrixAliasConstant & *copy*) `[inline]`

Base class copy constructor.

Makes a copy of any another matrix.

Parameters:
copy Reaference to matrix to copy

Definition at line 619 of file matrix.h.

```
619 {_constructMatrix(copy);}
```

6.10.2.4 Matrix::Matrix (const Matrix & *copy*) `[inline]`

Copy constructor.

Makes a copy of another matrix.

Parameters:
copy Reaference to matrix to copy

Definition at line 625 of file matrix.h.

```
625 {_constructMatrix(copy);}
```

6.10.2.5 Matrix::∼Matrix () `[virtual]`

Matrix Destructor.

Virtual - the lowest derived class will always need to be called, as they all allocate memory in ther own ways. Definition at line 276 of file matrix.cpp.

```
277 {
278         // Delete data array
279         delete[] m_matrixContainer->getDataPointer();
280
281         #ifdef DEBUG_DESTRUCTOR
282                 printf("Destructor: ~Matrix()\n");
283         #endif
284 }
```

6.10.3 Member Function Documentation

6.10.3.1 void Matrix::_constructMatrix (const MatrixAliasConstant & *copy*) `[protected]`

Copy constructor.

Allocates some memory, and then calls MatrixAlias' constructor function. Finally, copys data in from copied matrix

Parameters:

copy Matrix to copy

Definition at line 248 of file matrix.cpp.

```
{
        // Construct base class with new data size
        _constructMatrixAlias(new double[copy.getRows()*copy.getColumns()], copy.getRows(), copy.getColumns());

        // Constuct operators
        _constructMatrixWriteOperators();

        // Use equals operator to copy data in
        equals(copy);

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: Matrix::Copy Constructor\n");
        #endif
}
```

6.10.3.2 void Matrix::_constructMatrix (const unsigned int *rows*, const unsigned int *columns*) `[protected]`

Sized constructor.

Allocates some memory, and then calls MatrixAlias' constructor function

Parameters:

rows Number of rows matrix has

columns Number of columns matrix has

Definition at line 232 of file matrix.cpp.

```
{
        // Construct base class with new data size
        _constructMatrixAlias(new double[rows*columns], rows, columns);

        // Construct operators
        _constructMatrixWriteOperators();

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: Matrix::Sized Constructor\n");
        #endif
}
```

6.10.3.3 Matrix& Matrix::operator= (const Matrix & *copy*) `[inline]`

Assignment operator.

Definition at line 673 of file matrix.h.

```
{return operator=((MatrixAliasConstant&)copy);}
```

6.10.3.4 Matrix& Matrix::operator= (const MatrixAliasConstant & *copy*) `[inline]`

Base class assignment operator.

Reimplemented from **MatrixAlias** (p. 55).

Reimplemented in **SquareMatrix** (p. 200), **RowVector** (p. 150), and **ColumnVector** (p. 19).

Definition at line 670 of file matrix.h.

```
670 {equals(copy); return *this;}
```

6.10.4 Member Data Documentation

6.10.4.1 MWO_EqualsElementCopyResize Matrix::equals

Equals operator - allows matrix to resize if it is made equal to one of a different size.

Reimplemented from **MatrixAlias** (p. 55).

Reimplemented in **SquareMatrix** (p. 200), **RowVector** (p. 150), and **ColumnVector** (p. 19).

Definition at line 595 of file matrix.h.

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

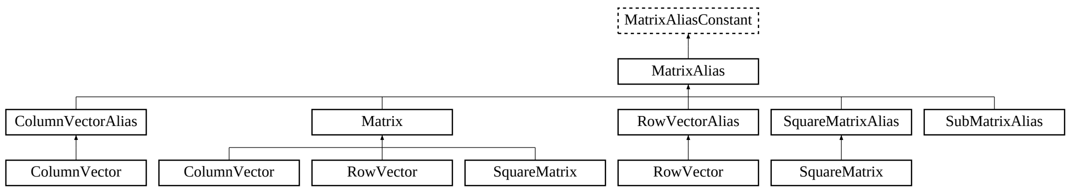
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix.cpp**

6.11 MatrixAlias Class Reference

An alias of a **Matrix**(p. 44) class, or to utilise a pre-available data array.

`#include <matrix.h>`

Inheritance diagram for MatrixAlias::



Public Member Functions

- **MatrixAlias** ()

  Default constructor.

- **MatrixAlias** (const double ∗data, const unsigned int rows, const unsigned int columns)

  Pointer constructor.

- **MatrixAlias** (const **MatrixAliasConstant** ∗alias)

  Alias constructor.

- **MatrixAlias** (const **MatrixAliasConstant** &copy)

  Copy constructor.

- **MatrixAlias** (const **MatrixAlias** &copy)

  Copy constructor.

- **MatrixAlias** (**MatrixContainer** ∗container, **MatrixReadAccess** ∗r_access, **Matrix-WriteAccess** ∗w_access)

  External container and access member constructor.

- virtual ∼**MatrixAlias** ()

  MatrixAlias Destructor.

- **MatrixAlias** & **operator=** (const **MatrixAliasConstant** &copy)

  Base class assignment operator.

- **MatrixAlias** & **operator=** (const **MatrixAlias** &copy)

  Assignment operator.

- double & **operator()** (const unsigned int row, const unsigned int column) const

  Element operator (two indexes).

- double & **operator()** (const unsigned int index) const

  Element operator (one index).

Public Attributes

- **MatrixWriteAccess ∗ m_matrixWriteAccess**

  Write access handle for the matrix.

- **MWO_Element element**

  Element operator.

- **MWO_EqualsElementCopy equals**

  Equals operator (makes the content of this matrix equal the values of another).

- **MWO_Zero zero**

  Zero all the elements in the matrix.

- **MWO_Set set**

  Sets all the values in the matrix equal to a specified value.

- **MWO_Randomise rand**

  Randomises all the values in the matrix to a values between 0 and specified value.

- **MWO_SubMatrixAlias subMatrix**

  Returns a subMatrixAlias of a portion of this matrix.

Protected Member Functions

- void **_constructMatrixAlias** (const double ∗data, const unsigned int rows, const unsigned int columns)

  Pointer constructor.

- void **_constructMatrixAlias** (const **MatrixAliasConstant** &copy)

  Copy constructor.

- void **_constructMatrixAlias** (**MatrixContainer** ∗container, **MatrixReadAccess** ∗r_-access, **MatrixWriteAccess** ∗w_access)

  Member constructor (external).

6.11.1 Detailed Description

An alias of a **Matrix**(p. 44) class, or to utilise a pre-available data array.

Author:
Lee Netherton

The MatrixAlias class provides a means to allow the aliasing of matrices, and also the flexibility to use external data storage made available be the user.

For example:

```
MatrixAlias giveMeAMatrix(int rows, int columns)
{
    double * dataPointer = new double[rows*columns];   // allocate some new memory

    MatrixAlias newMA(dataPointer, rows, columns);     // create new MatrixAlias

    return newMA
}
```

Definition at line 417 of file matrix.h.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 MatrixAlias::MatrixAlias () `[inline]`

Default constructor.

Creates a MatrixAlias shell. The **MatrixContainer**(p. 70), **MatrixReadAccess**(p. 73), and **MatrixWriteAccess**(p. 81) handles can be set later using the constructor function **_construct-MatrixAlias()**(p. 54) Definition at line 462 of file matrix.h.

```
462 {}
```

6.11.2.2 MatrixAlias::MatrixAlias (const double ∗ *data*, const unsigned int *rows*, const unsigned int *columns*) `[inline]`

Pointer constructor.

To create a matrix that will access a pre-available data array.

Parameters:

data Pointer to data array. This will be the data storage for the matrix.

rows Number of rows the matrix has.

columns Number of columns the matrix has.

Definition at line 470 of file matrix.h.

```
470 {_constructMatrixAlias(data, rows, columns);}
```

6.11.2.3 MatrixAlias::MatrixAlias (const MatrixAliasConstant ∗ *alias*) `[inline]`

Alias constructor.

To create a matrix that will alias another matrix.

Parameters:

alias Pointer to a matrix which this matris will alias.

Definition at line 476 of file matrix.h.

```
476 {_constructMatrixAlias(*alias);}
```

6.11.2.4 MatrixAlias::MatrixAlias (const MatrixAliasConstant & *copy*) `[inline]`

Copy constructor.

Used when creating a **MatrixAliasConstant**(p. 57) matrix from any other matrix.

Parameters:

copy Reference to another matrix.

Definition at line 482 of file matrix.h.

```
482 {_constructMatrixAlias(copy);}
```

6.11.2.5 MatrixAlias::MatrixAlias (const MatrixAlias & *copy*) `[inline]`

Copy constructor.

Used when creating a **MatrixAliasConstant**(p. 57) matrix from another.

```
 MatrixAliasConstant newMA(oldMA);
```

Parameters:

copy Reference to another matrix.

Definition at line 489 of file matrix.h.

```
489 {_constructMatrixAlias(copy);}
```

6.11.2.6 MatrixAlias::MatrixAlias (MatrixContainer ∗ *container*, MatrixReadAccess ∗ *r_ access*, MatrixWriteAccess ∗ *w_ access*) `[inline]`

External container and access member constructor.

Used when the **MatrixContainer**(p. 70) and matrix access members have been created externally (for example, in a submatrix). The MatrixAlias shell is populated with operators and the handles are set.

Parameters:

container Pointer to **MatrixContainer**(p. 70)

r_ access Pointer to **MatrixReadAccess**(p. 73)

w_ access Pointer to **MatrixWriteAccess**(p. 81)

Definition at line 499 of file matrix.h.

```
499 {_constructMatrixAlias(container, r_access, w_access);}
```

6.11.2.7 MatrixAlias::∼MatrixAlias () `[virtual]`

MatrixAlias Destructor.

Virtual - the lowest derived class will always need to be called, as they all allocate memory in ther own ways. Definition at line 207 of file matrix.cpp.

```
{
        // Delete write access object
        delete m_matrixWriteAccess;

        #ifdef DEBUG_DESTRUCTOR
                printf("Destructor: ~MatrixAlias()\n");
        #endif
}
```

6.11.3 Member Function Documentation

6.11.3.1 void MatrixAlias::_constructMatrixAlias (MatrixContainer ∗ *container*, MatrixReadAccess ∗ *r_ access*, MatrixWriteAccess ∗ *w_ access*) `[protected]`

Member constructor (external).

Sets the pointers to the **MatrixWriteOperator**(p. 85) member from an extenal source.

Parameters:

container Pointer to **MatrixContainer**(p. 70)

r_ access Pointer to **MatrixReadAccess**(p. 73)

w_ access Pointer to **MatrixWriteAccess**(p. 81)

Definition at line 173 of file matrix.cpp.

```
{
        // Construct base class with members
        _constructMatrixAliasConstant(container, r_access);

        // Set write access class pointer
        m_matrixWriteAccess = w_access;

        // Construct operators
        _constructMatrixWriteOperators();

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: MatrixAlias::Member Constructor\n");
        #endif

}
```

6.11.3.2 void MatrixAlias::_constructMatrixAlias (const MatrixAliasConstant & *copy*) `[protected]`

Copy constructor.

Sets the pointers to the **MatrixWriteOperator**(p. 85) member, and calls MatrixAliasConstant's constructor function with copy's values

Parameters:

copy Reference to the matrix to copy

Definition at line 154 of file matrix.cpp.

```
{
        // Construct base class
        _constructMatrixAliasConstant(copy.getDataPointer(), copy.getRows(), copy.getColumns());

        // Create new wrire access class
        m_matrixWriteAccess = new MatrixWriteAccess(this);

        // Construct operators
        _constructMatrixWriteOperators();

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: MatrixAlias::Copy Constructor\n");
        #endif
}
```

6.11.3.3 void MatrixAlias::_constructMatrixAlias (const double ∗ *data*, const unsigned int *rows*, const unsigned int *columns*) [protected]

Pointer constructor.

Sets the pointers to the **MatrixWriteOperator**(p. 85) member, and calls MatrixAliasConstant's constructor function

Parameters:

data Pointer to data array. This will be the data storage for the matrix.

rows Number of rows the matrix has.

columns Number of columns the matrix has.

Definition at line 135 of file matrix.cpp.

```
{
        // Construct base class
        _constructMatrixAliasConstant(data, rows, columns);

        // Create new wrire access class
        m_matrixWriteAccess = new MatrixWriteAccess(this);

        // Construct operators
        _constructMatrixWriteOperators();

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: MatrixAlias::Pointer Constructor\n");
        #endif
}
```

6.11.3.4 double& MatrixAlias::operator() (const unsigned int *index*) const [inline]

Element operator (one index).

Reimplemented from **MatrixAliasConstant** (p. 64).

Definition at line 563 of file matrix.h.

```
{return element(index);}
```

6.11.3.5 double& MatrixAlias::operator() (const unsigned int *row*, const unsigned int *column*) const `[inline]`

Element operator (two indexes).

Reimplemented from **MatrixAliasConstant** (p. 65).

Definition at line 560 of file matrix.h.

```
560 {return element(row,column);}
```

6.11.3.6 MatrixAlias& MatrixAlias::operator= (const MatrixAlias & *copy*) `[inline]`

Assignment operator.

Definition at line 557 of file matrix.h.

```
557 {return operator=((MatrixAliasConstant&)copy);}
```

6.11.3.7 MatrixAlias& MatrixAlias::operator= (const MatrixAliasConstant & *copy*) `[inline]`

Base class assignment operator.

Reimplemented from **MatrixAliasConstant** (p. 66).

Reimplemented in **Matrix** (p. 48), **SubMatrixAlias** (p. 224), **SquareMatrixAlias** (p. 206), **SquareMatrix** (p. 200), **RowVectorAlias** (p. 156), **RowVector** (p. 150), **ColumnVectorAlias** (p. 24), and **ColumnVector** (p. 19).

Definition at line 554 of file matrix.h.

```
554 {equals(copy);return *this;}
```

6.11.4 Member Data Documentation

6.11.4.1 MWO_Element MatrixAlias::element

Element operator.

Reimplemented from **MatrixAliasConstant** (p. 67).

Definition at line 436 of file matrix.h.

6.11.4.2 MWO_EqualsElementCopy MatrixAlias::equals

Equals operator (makes the content of this matrix equal the values of another).

Reimplemented in **Matrix** (p. 48), **SquareMatrix** (p. 200), **RowVector** (p. 150), and **ColumnVector** (p. 19).

Definition at line 439 of file matrix.h.

6.11.4.3 MatrixWriteAccess∗ MatrixAlias::m_matrixWriteAccess

Write access handle for the matrix.

Provides basic write funcions for all the write operators Definition at line 428 of file matrix.h.

6.11.4.4 MWO_Randomise MatrixAlias::rand

Randomises all the values in the matrix to a values between 0 and specified value.

Definition at line 448 of file matrix.h.

6.11.4.5 MWO_Set MatrixAlias::set

Sets all the values in the matrix equal to a specified value.

Definition at line 445 of file matrix.h.

6.11.4.6 MWO_SubMatrixAlias MatrixAlias::subMatrix

Returns a subMatrixAlias of a portion of this matrix.

Reimplemented from **MatrixAliasConstant** (p. 68).

Definition at line 451 of file matrix.h.

6.11.4.7 MWO_Zero MatrixAlias::zero

Zero all the elements in the matrix.

Definition at line 442 of file matrix.h.

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

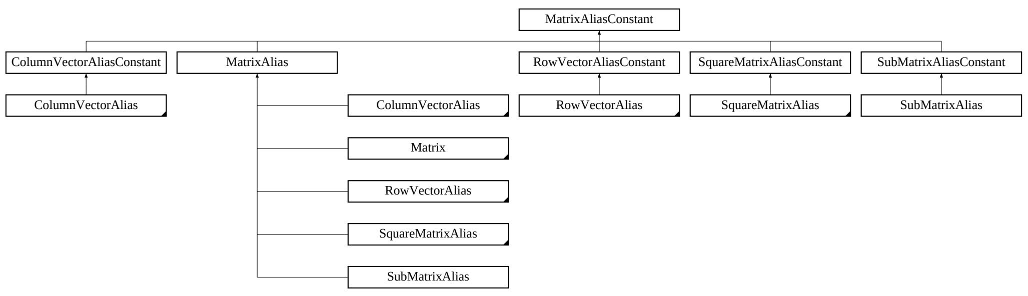
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix.cpp**

6.12 MatrixAliasConstant Class Reference

A read-only **MatrixAlias**(p. 49) class.

`#include <matrix.h>`

Inheritance diagram for MatrixAliasConstant::



Public Member Functions

- **MatrixAliasConstant** ()

  Default constructor.

- **MatrixAliasConstant** (const double ∗data, const unsigned int rows, const unsigned int columns)

  Pointer constructor.

- **MatrixAliasConstant** (const **MatrixAliasConstant** ∗alias)

  Alias constructor.

- **MatrixAliasConstant** (const **MatrixAliasConstant** &copy)

  Copy constructor.

- **MatrixAliasConstant** (**MatrixContainer** ∗container, **MatrixReadAccess** ∗access)

  External container and access member constructor.

- virtual ∼**MatrixAliasConstant** ()

  MatrixAliasConstant Destructor.

- const double & **operator()** (const unsigned int row, const unsigned int column) const

  Element operator (two indexes).

- const double & **operator()** (const unsigned int index) const

  Element operator (one index).

- **Matrix operator+** (const **MatrixAliasConstant** &operand) const

  Addition operator.

- **Matrix operator-** (const **MatrixAliasConstant** &operand) const

  Subtration operator.

- **Matrix operator-** () const

  Negative operator.

- **Matrix operator** ∗ (const **MatrixAliasConstant** &operand) const

  Multiplication of two matrices.

- **Matrix operator** ∗ (const double &operand) const

  Multiplication of matrix by a scalar.

- **Matrix operator**/ (const double &operand) const

  Division by a scalar.

- **MatrixAliasConstant** & **operator**= (const **MatrixAliasConstant** &copy)
- const unsigned int **getRows** () const

  Get the number of rows the matrix has.

- const unsigned int **getColumns** () const

  Get the number of columns the matrix has.

- const double ∗ **getDataPointer** () const

  Return a pointer to the data array.

Public Attributes

- **MatrixContainer** ∗ **m_matrixContainer**

  MatrixContainer(p. 70) *for this matrix.*

- **MatrixReadAccess** ∗ **m_matrixReadAccess**

  Read access handle for the matrix.

- **MRO_Element element**

  Element operator.

- **MRO_Add add**

  Addition operator.

- **MRO_Subtract subtract**

  Subtration operator.

- **MRO_Negative negative**

  Negative operator (reverses sign).

- **MRO_Multiply multiply**

  Multiplication operator.

- **MRO_Divide divide**

  Division operator.

- **MRO_Print print**

Prints matrix contents to screen.

- **MRO_PrintMatlabFriendly printM**

  Prints matrix contents in a form that can be pasted into MATLAB.

- **MRO_SubMatrixAliasConstant subMatrix**

  Return a subMatrixAliasConstant of a portion of this matrix.

- **MRO_SizeEqual sizeEqual**

  Return true if operand's size is equal to this matrix's size.

- **MRO_IsSquareMatrix isSquareMatrix**

  Returns true if matrix is square.

- **MRO_IsRowVector isRowVector**

  Returns true if matrix is square.

- **MRO_IsColumnVector isColumnVector**

  Returns true if matrix is square.

- **MRO_Transpose transpose**

  Returns the transpose of this matrix.

- **MRO_Absolute absolute**

  Returns an absolute version of this matrix.

- **MRO_RowSum rowSum**

  Returns a column vector which is the row sum of this matrix.

- **MRO_ColumnSum columnSum**

  Returns a row vector which is the column sum of this matrix.

- **MRO_Maximum maximum**

  Returns the maximum value in this matrix.

- **MRO_Minimum minimum**

  Returns the minimum value in this matrix.

- **MRO_InfinityNorm infinityNorm**

  Returns the infinity norm value of this matrix.

- **MRO_SquaredElements squaredElements**

  Returns a matrix of this matrix with its elements squared.

Protected Member Functions

- void **_constructMatrixAliasConstant** (const double ∗data, const unsigned int rows, const unsigned int columns)

  Pointer constructor.

- void **_constructMatrixAliasConstant** (const **MatrixAliasConstant** &copy)

  Copy constructor.

- void **_constructMatrixAliasConstant** (**MatrixContainer** ∗container, **MatrixReadAccess** ∗access)

  Member constructor (external).

Friends

- **Matrix operator** ∗ (const double &operand1, const **MatrixAliasConstant** &operand2)

  Multiplication of scalar by a matrix.

6.12.1 Detailed Description

A read-only **MatrixAlias**(p. 49) class.

Author:
Lee Netherton

The MatrixAliasConstant class provides a means to allow matrix functionality to otherwise unaccesible read-only data.

For example:

```
void foo(const double * readOnlyData, int rows, int columns)
{
    MatrixAliasConstant readOnlyMatrix(readOnlyData, rows, columns);


    // matrix operations on readOnlyMatrix....
}
```

Definition at line 179 of file matrix.h.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 MatrixAliasConstant::MatrixAliasConstant () `[inline]`

Default constructor.

Creates a MatrixAliasConstant shell. The **MatrixContainer**(p. 70) and **MatrixReadAccess**(p. 73) handles can be set later using the constructor function **_constructMatrixAliasConstant()**(p. 63) Definition at line 274 of file matrix.h.

```
274 {}
```

6.12.2.2 MatrixAliasConstant::MatrixAliasConstant (const double ∗ *data*, const unsigned int *rows*, const unsigned int *columns*) `[inline]`

Pointer constructor.

To create a read-only matrix that will access a pre-available data array.

Parameters:

data Pointer to data array. This will be the data storage for the matrix.

rows Number of rows the matrix has.

columns Number of columns the matrix has.

Definition at line 282 of file matrix.h.

```
282 {_constructMatrixAliasConstant(data, rows, columns);}
```

6.12.2.3 MatrixAliasConstant::MatrixAliasConstant (const MatrixAliasConstant ∗ *alias*) `[inline]`

Alias constructor.

To create a read-only matrix that will alias another matrix.

Parameters:

alias Pointer to a matrix which this matris will alias.

Definition at line 288 of file matrix.h.

```
288 {_constructMatrixAliasConstant(*alias);}
```

6.12.2.4 MatrixAliasConstant::MatrixAliasConstant (const MatrixAliasConstant & *copy*) `[inline]`

Copy constructor.

Used when creating a MatrixAliasConstant matrix from another.

```
 MatrixAliasConstant newMAC(oldMAC);
```

Parameters:

copy Reference to another matrix.

Definition at line 295 of file matrix.h.

```
295 {_constructMatrixAliasConstant(copy);}
```

6.12.2.5 MatrixAliasConstant::MatrixAliasConstant (MatrixContainer ∗ *container*, MatrixReadAccess ∗ *access*) `[inline]`

External container and access member constructor.

Used when the **MatrixContainer**(p. 70) and **MatrixReadAccess**(p. 73) members have been created externally (for example, in a submatrix). The MatrixAliasConstant shell is populated with operators and the handles are set.

Parameters:

container Pointer to **MatrixContainer**(p. 70)

access Pointer to **Matrix**(p. 44) ReadAccess

Definition at line 304 of file matrix.h.

```
304 {_constructMatrixAliasConstant(container, access);}
```

6.12.2.6 MatrixAliasConstant::∼MatrixAliasConstant () `[virtual]`

MatrixAliasConstant Destructor.

Virtual - the lowest derived class will always need to be called, as they all allocate memory in ther own ways. Definition at line 106 of file matrix.cpp.

```
107 {
108         // Delete container
109         delete m_matrixContainer;
110
111         // Delete read access object
112         delete m_matrixReadAccess;
113
114         #ifdef DEBUG_DESTRUCTOR
115                 printf("Destructor: ~MatrixAliasConstant()\n");
116         #endif
117 }
```

6.12.3 Member Function Documentation

6.12.3.1 void MatrixAliasConstant::_constructMatrixAliasConstant (MatrixContainer ∗ *container*, MatrixReadAccess ∗ *access*) `[protected]`

Member constructor (external).

Sets the pointers to the **MatrixContainer**(p. 70) and **MatrixReadOperator**(p. 77) members from an extenal source. Definition at line 58 of file matrix.cpp.

```
59 {
60         // Set container class pointer
61         m_matrixContainer = container;
62
63         // Set read access class pointer
64         m_matrixReadAccess = access;
65
66         // Construct operators
67         _constructMatrixReadOperators();
68
69         #ifdef DEBUG_CONSTRUCTOR
```

```
                printf("Constructed: MatrixAliasConstant::Member Constructor\n");
        #endif
}
```

6.12.3.2 void MatrixAliasConstant::_constructMatrixAliasConstant (const MatrixAliasConstant & *copy*) [protected]

Copy constructor.

Copies the pointers to the **MatrixContainer**(p. 70) and **MatrixReadOperator**(p. 77) members Definition at line 39 of file matrix.cpp.

```
{
        // Create new container from copy
        m_matrixContainer = new MatrixContainer(copy.getDataPointer(), copy.getRows(), copy.getColumns());

        // Create new read access object
        m_matrixReadAccess = new MatrixReadAccess(this);

        // Construct operators
        _constructMatrixReadOperators();

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: MatrixAliasConstant::Copy Constructor\n");
        #endif
}
```

6.12.3.3 void MatrixAliasConstant::_constructMatrixAliasConstant (const double ∗ *data*, const unsigned int *rows*, const unsigned int *columns*) [protected]

Pointer constructor.

Sets the pointers to the **MatrixContainer**(p. 70) and **MatrixReadOperator**(p. 77) members Definition at line 19 of file matrix.cpp.

```
{
        // Create new container
        m_matrixContainer = new MatrixContainer(data, rows, columns);

        // Create new read access object
        m_matrixReadAccess = new MatrixReadAccess(this);

        // Construct operators
        _constructMatrixReadOperators();


        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: MatrixAliasConstant::Pointer Constructor\n");
        #endif
}
```

6.12.3.4 const unsigned int MatrixAliasConstant::getColumns () const [inline]

Get the number of columns the matrix has.

Definition at line 390 of file matrix.h.

```
390 {return m_matrixContainer->getColumns();}
```

6.12.3.5 const double* MatrixAliasConstant::getDataPointer () const [inline]

Return a pointer to the data array.

Definition at line 393 of file matrix.h.

```
393 {return m_matrixContainer->getDataPointer();}
```

6.12.3.6 const unsigned int MatrixAliasConstant::getRows () const [inline]

Get the number of rows the matrix has.

Definition at line 387 of file matrix.h.

```
387 {return m_matrixContainer->getRows();}
```

6.12.3.7 Matrix MatrixAliasConstant::operator * (const double & *operand*) const [inline]

Multiplication of matrix by a scalar.

Definition at line 368 of file matrix.h.

```
368 {return multiply(operand);}
```

6.12.3.8 Matrix MatrixAliasConstant::operator * (const MatrixAliasConstant & *operand*) const [inline]

Multiplication of two matrices.

Definition at line 365 of file matrix.h.

```
365 {return multiply(operand);}
```

6.12.3.9 const double& MatrixAliasConstant::operator() (const unsigned int *index*) const [inline]

Element operator (one index).

Reimplemented in **MatrixAlias** (p. 54).

Definition at line 353 of file matrix.h.

```
353 {return element(index);}
```

6.12.3.10 const double& MatrixAliasConstant::operator() (const unsigned int *row*, const unsigned int *column*) const `[inline]`

Element operator (two indexes).

Reimplemented in **MatrixAlias** (p. 55).

Definition at line 350 of file matrix.h.

```
350 {return element(row,column);}
```

6.12.3.11 Matrix MatrixAliasConstant::operator+ (const MatrixAliasConstant & *operand*) const `[inline]`

Addition operator.

Definition at line 356 of file matrix.h.

```
356 {return add(operand);}
```

6.12.3.12 Matrix MatrixAliasConstant::operator- () const `[inline]`

Negative operator.

Definition at line 362 of file matrix.h.

```
362 {return negative();}
```

6.12.3.13 Matrix MatrixAliasConstant::operator- (const MatrixAliasConstant & *operand*) const `[inline]`

Subtration operator.

Definition at line 359 of file matrix.h.

```
359 {return subtract(operand);}
```

6.12.3.14 Matrix MatrixAliasConstant::operator/ (const double & *operand*) const `[inline]`

Division by a scalar.

Definition at line 374 of file matrix.h.

```
374 {return divide(operand);}
```

6.12.3.15 MatrixAliasConstant& MatrixAliasConstant::operator= (const MatrixAliasConstant & *copy*) [inline]

Assignment operator

Overload default assignment operation to stop bad things happening. Simply return object unchanged.

Reimplemented in **MatrixAlias** (p. 55), **Matrix** (p. 48), **SubMatrixAliasConstant** (p. 230), **SubMatrixAlias** (p. 224), **SquareMatrixAliasConstant** (p. 212), **SquareMatrixAlias** (p. 206), **SquareMatrix** (p. 200), **RowVectorAliasConstant** (p. 161), **RowVectorAlias** (p. 156), **RowVector** (p. 150), **ColumnVectorAliasConstant** (p. 29), **ColumnVectorAlias** (p. 24), and **ColumnVector** (p. 19).

Definition at line 380 of file matrix.h.

```
380 {m_matrixReadAccess->error("Tried to assign to a constant matrix\n");return *this;}
```

6.12.4 Friends And Related Function Documentation

6.12.4.1 Matrix operator * (const double & *operand1*, const MatrixAliasConstant & *operand2*) [friend]

Multiplication of scalar by a matrix.

Definition at line 371 of file matrix.h.

```
371 {return operand2.multiply(operand1);}
```

6.12.5 Member Data Documentation

6.12.5.1 MRO_Absolute MatrixAliasConstant::absolute

Returns an absolute version of this matrix.

Definition at line 245 of file matrix.h.

6.12.5.2 MRO_Add MatrixAliasConstant::add

Addition operator.

Definition at line 206 of file matrix.h.

6.12.5.3 MRO_ColumnSum MatrixAliasConstant::columnSum

Returns a row vector which is the column sum of this matrix.

Definition at line 251 of file matrix.h.

6.12.5.4 MRO_Divide MatrixAliasConstant::divide

Division operator.

Definition at line 218 of file matrix.h.

6.12.5.5 MRO_Element MatrixAliasConstant::element

Element operator.

Reimplemented in **MatrixAlias** (p. 55).

Definition at line 203 of file matrix.h.

6.12.5.6 MRO_InfinityNorm MatrixAliasConstant::infinityNorm

Returns the infinity norm value of this matrix.

Definition at line 260 of file matrix.h.

6.12.5.7 MRO_IsColumnVector MatrixAliasConstant::isColumnVector

Returns true if matrix is square.

Definition at line 239 of file matrix.h.

6.12.5.8 MRO_IsRowVector MatrixAliasConstant::isRowVector

Returns true if matrix is square.

Definition at line 236 of file matrix.h.

6.12.5.9 MRO_IsSquareMatrix MatrixAliasConstant::isSquareMatrix

Returns true if matrix is square.

Definition at line 233 of file matrix.h.

6.12.5.10 MatrixContainer* MatrixAliasConstant::m_matrixContainer

MatrixContainer(p. 70) for this matrix.

The containter holds all the storage information for the matrix. Definition at line 191 of file matrix.h.

6.12.5.11 MatrixReadAccess* MatrixAliasConstant::m_matrixReadAccess

Read access handle for the matrix.

Provides basic read funcions for all the operators Definition at line 196 of file matrix.h.

6.12.5.12 MRO_Maximum MatrixAliasConstant::maximum

Returns the maximum value in this matrix.

Definition at line 254 of file matrix.h.

6.12.5.13 MRO_Minimum MatrixAliasConstant::minimum

Returns the minimum value in this matrix.

Definition at line 257 of file matrix.h.

6.12.5.14 MRO_Multiply MatrixAliasConstant::multiply

Multiplication operator.

Definition at line 215 of file matrix.h.

6.12.5.15 MRO_Negative MatrixAliasConstant::negative

Negative operator (reverses sign).

Definition at line 212 of file matrix.h.

6.12.5.16 MRO_Print MatrixAliasConstant::print

Prints matrix contents to screen.

Definition at line 221 of file matrix.h.

6.12.5.17 MRO_PrintMatlabFriendly MatrixAliasConstant::printM

Prints matrix contents in a form that can be pasted into MATLAB.

Definition at line 224 of file matrix.h.

6.12.5.18 MRO_RowSum MatrixAliasConstant::rowSum

Returns a column vector which is the row sum of this matrix.

Definition at line 248 of file matrix.h.

6.12.5.19 MRO_SizeEqual MatrixAliasConstant::sizeEqual

Return true if operand's size is equal to this matrix's size.

Definition at line 230 of file matrix.h.

6.12.5.20 MRO_SquaredElements MatrixAliasConstant::squaredElements

Returns a matrix of this matrix with its elements squared.

Definition at line 263 of file matrix.h.

6.12.5.21 MRO_SubMatrixAliasConstant MatrixAliasConstant::subMatrix

Return a subMatrixAliasConstant of a portion of this matrix.

Reimplemented in **MatrixAlias** (p. 56).

Definition at line 227 of file matrix.h.

6.12.5.22 MRO_Subtract MatrixAliasConstant::subtract

Subtration operator.

Definition at line 209 of file matrix.h.

6.12.5.23 MRO_Transpose MatrixAliasConstant::transpose

Returns the transpose of this matrix.

Definition at line 242 of file matrix.h.

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

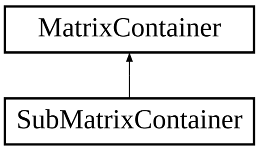
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix.cpp**

6.13 MatrixContainer Class Reference

Store for primative matrix information.

`#include <matrix_container.h>`

Inheritance diagram for MatrixContainer::



Public Member Functions

- **MatrixContainer** (const double ∗data, const unsigned int rows, const unsigned int columns)

  Sized constructor.

- **MatrixContainer** (const **MatrixContainer** &copy)

  Copy constructor.

- unsigned int **getRows** () const

  Returns the number of rows.

- unsigned int **getColumns** () const

  Retuens the number of columns.

- double ∗ **getDataPointer** () const

  Returns the data pointer.

- void **setRows** (unsigned int num)

  Sets the number of rows.

- void **setColumns** (unsigned int num)

  Sets the number of columns.

- void **setDataPointer** (double ∗ptr)

  Sets the data pointer.

6.13.1 Detailed Description

Store for primative matrix information.

Author:
Lee Netherton

The MatrixContainer class is used to store the matrix data. It holds a pointer to the data array, and also the size of the matrix. Each matrix will have a pointer to a matrix container within it.

Definition at line 13 of file matrix_container.h.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 MatrixContainer::MatrixContainer (const double ∗ *data*, const unsigned int *rows*, const unsigned int *columns*) `[inline]`

Sized constructor.

To create a MatrixContainer with appropriate values.

Parameters:

data Pointer to data array. Assigned to *m_ data*.

rows Number of rows. Assigned to *m_ rows*.

columns Number of columns. Assigned to *m_ columns*.

Definition at line 34 of file matrix_container.h.

```
34                                                                                         : m_data((c
35              {
36                      #ifdef DEBUG_CONSTRUCTOR
37                              printf("Constructor: MatrixContainer(const double * data, const unsigned int rows, co
38                      #endif
39              }
```

6.13.2.2 MatrixContainer::MatrixContainer (const MatrixContainer & *copy*) `[inline]`

Copy constructor.

To create a MatrixContainer from a copy of another

Parameters:

copy Reference to container to copy

Definition at line 45 of file matrix_container.h.

```
45                                                        : m_data(copy.getDataPointer()), m_rows(copy.getRows())
46              {
47                      #ifdef DEBUG_CONSTRUCTOR
48                              printf("Constructor: MatrixContainer(const MatrixContainer& copy)\n");
49                      #endif
50              }
```

6.13.3 Member Function Documentation

6.13.3.1 unsigned int MatrixContainer::getColumns () const `[inline]`

Retuens the number of columns.

Definition at line 58 of file matrix_container.h.

```
58 {return m_columns;}
```

6.13.3.2 double∗ MatrixContainer::getDataPointer () const `[inline]`

Returns the data pointer.

Definition at line 61 of file matrix_container.h.

```
61 {return m_data;}
```

6.13.3.3 unsigned int MatrixContainer::getRows () const `[inline]`

Returns the number of rows.

Definition at line 55 of file matrix_container.h.

```
55 {return m_rows;}
```

6.13.3.4 void MatrixContainer::setColumns (unsigned int *num*) `[inline]`

Sets the number of columns.

Definition at line 67 of file matrix_container.h.

```
67 {m_columns = num;}
```

6.13.3.5 void MatrixContainer::setDataPointer (double ∗ *ptr*) `[inline]`

Sets the data pointer.

Definition at line 70 of file matrix_container.h.

```
70 {m_data = ptr;}
```

6.13.3.6 void MatrixContainer::setRows (unsigned int *num*) `[inline]`

Sets the number of rows.

Definition at line 64 of file matrix_container.h.

```
64 {m_rows = num;}
```

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

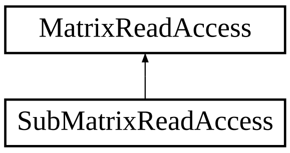
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_container.h**

6.14 MatrixReadAccess Class Reference

Read access for matrix classes.

`#include <matrix_access.h>`

Inheritance diagram for MatrixReadAccess::



Public Member Functions

- **MatrixReadAccess** (**MatrixAliasConstant** ∗mAC)

  Constructor.

- virtual const double & **readElement** (const unsigned int row, const unsigned int column) const
- virtual const double & **readElement** (const unsigned int index) const
- const unsigned int **getRows** () const

  Returns the number of rows the matrix has.

- const unsigned int **getColumns** () const

  Returns the number of columns the matrix has.

- const double ∗ **getDataPointer** () const

  Returns the a pointer to the matrix data array.

- void **error** (const char ∗str) const

  For flagging errors - to be depreciated.

Protected Attributes

- **MatrixAliasConstant** ∗ **m_matrixAliasConstant**

  Pointer to owner matrix.

6.14.1 Detailed Description

Read access for matrix classes.

Author:
Lee Netherton

The MatrixReadAccess class provides basic read operatons for the MatrixOperator classes, and the matrix class itself. Each matrix class will have a pointer to a MatrixReadAccess, or one if its derivatives.

Definition at line 19 of file matrix_access.h.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 MatrixReadAccess::MatrixReadAccess (MatrixAliasConstant ∗ *mAC*) `[inline]`

Constructor.

Sets pointer to owner matrix

Parameters:
mAC Pointer to owner matrix

Definition at line 32 of file matrix_access.h.

```
32                                                           : m_matrixAliasConstant(mAC)
33              {
34                      #ifdef DEBUG_CONSTRUCT
35                              printf("Constructor: MatrixReadAccess(MatrixAliasConstant * mAC)\n");
36                      #endif
37              }
```

6.14.3 Member Function Documentation

6.14.3.1 void MatrixReadAccess::error (const char ∗ *str*) const

For flagging errors - to be depreciated.

Definition at line 54 of file matrix_access.cpp.

```
54                                               {
55
56         // Just a printf for now
57         printf(str);
58 }
```

6.14.3.2 const unsigned int MatrixReadAccess::getColumns () const

Returns the number of columns the matrix has.

Definition at line 47 of file matrix_access.cpp.

```
47 {return m_matrixAliasConstant->getColumns();}
```

6.14.3.3 const double ∗ MatrixReadAccess::getDataPointer () const

Returns the a pointer to the matrix data array.

Definition at line 48 of file matrix_access.cpp.

```
48 {return m_matrixAliasConstant->m_matrixContainer->getDataPointer();}
```

6.14.3.4 const unsigned int MatrixReadAccess::getRows () const

Returns the number of rows the matrix has.

Definition at line 46 of file matrix_access.cpp.

```
46 {return m_matrixAliasConstant->getRows();}
```

6.14.3.5 const double & MatrixReadAccess::readElement (const unsigned int *index*) const [virtual]

Returns a read-only reference to the data member at specified position

Parameters:

index Row-wise position of element (zero-indexed)

Reimplemented in **SubMatrixReadAccess** (p. 238).

Definition at line 31 of file matrix_access.cpp.

```
32 {
33         if (index >= (getRows()*getColumns())) {
34                 error("MatrixAlias::element : Subscript out of range\n");
35                 return getDataPointer()[0];
36         }
37 
38         return getDataPointer()[index];
39 
40 }
```

6.14.3.6 const double & MatrixReadAccess::readElement (const unsigned int *row*, const unsigned int *column*) const [virtual]

Returns a read-only reference to the data member at specified position

Parameters:

row Row position of desired element (zero indexed)

column Column position of desired element (zero indexed)

Reimplemented in **SubMatrixReadAccess** (p. 238).

Definition at line 20 of file matrix_access.cpp.

```
21 {
22         if (row >= getRows() || column >= getColumns()) {
23                 error("MatrixAlias::element : Subscript out of range\n");
24                 return getDataPointer()[0];
25         }
26 
27         return getDataPointer()[row*(getColumns()) + column];
28 
29 }
```

6.14.4 Member Data Documentation

6.14.4.1 MatrixAliasConstant∗ MatrixReadAccess::m_matrixAliasConstant `[protected]`

Pointer to owner matrix.

Definition at line 24 of file matrix_access.h.

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

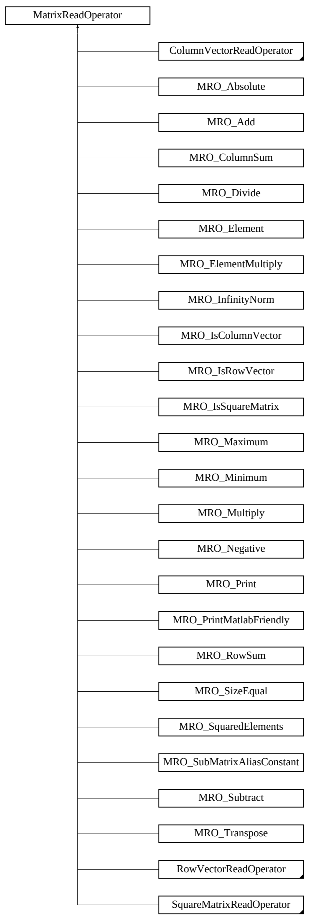
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_access.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_access.cpp**

6.15 MatrixReadOperator Class Reference

Base class for MatrixReadOperators.

`#include <matrix_operator.h>`

Inheritance diagram for MatrixReadOperator::



Public Member Functions

- **MatrixReadOperator** ()

  Default Constructor.

- **MatrixReadOperator** (**MatrixAliasConstant** ∗matrixAliasConstant)

  Full Constructor.

- void **_constructMatrixReadOperator** (**MatrixAliasConstant** ∗matrixAliasConstant)

  Manual Constructor.

- const unsigned int **getRows** () const

  Returns the number of rows the matrix has.

- const unsigned int **getColumns** () const

  Returns the number of columns the matrix has.

- const double ∗ **getDataPointer** () const

  Returns the a pointer to the matrix data array.

- const double & **element** (const unsigned int row, const unsigned int column) const
- const double & **element** (const unsigned int index) const
- void **error** (const char ∗str) const

  For flagging errors - to be depreciated.

Protected Attributes

- **MatrixAliasConstant** ∗ **m_thisMatrix**

  Pointer to owner matrix.

6.15.1 Detailed Description

Base class for MatrixReadOperators.

Author:
Lee Netherton

A MatrixReadOperator is a class which will perform a read operation on a matrix. The MatrixReadOperator class is never instantiated directly, but serves as a bass for derived operator classes. It provides basic read operatons for its derivatives.

Definition at line 40 of file matrix_operator.h.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 MatrixReadOperator::MatrixReadOperator () `[inline]`

Default Constructor.

Creates an empty operator class which can then be more full constructed using **_constructMatrixReadOperator()**(p. 79) Definition at line 54 of file matrix_operator.h.

```
54 {}
```

6.15.2.2 MatrixReadOperator::MatrixReadOperator (MatrixAliasConstant ∗ *matrixAliasConstant*) `[inline]`

Full Constructor.

Creates an operator class which takes and stores a pointer to an owner matrix

Parameters:
matrixAliasConstant Pointer to owner matrix

Definition at line 60 of file matrix_operator.h.

```
60 : m_thisMatrix(matrixAliasConstant) {}
```

6.15.3 Member Function Documentation

6.15.3.1 void MatrixReadOperator::_constructMatrixReadOperator (MatrixAliasConstant * *matrixAliasConstant*) [inline]

Manual Constructor.

Constructs the class manually by setting the owner pointer

Parameters:
matrixAliasConstant Pointer to owner matrix

Definition at line 66 of file matrix_operator.h.

```
66 {m_thisMatrix = matrixAliasConstant;}
```

6.15.3.2 const double & MatrixReadOperator::element (const unsigned int *index*) const

Returns a read-only reference to the data member at specified position

Parameters:
index Row-wise position of element (zero-indexed)

Definition at line 19 of file matrix_operator.cpp.

```
19 {return m_thisMatrix->m_matrixReadAccess->readElement(index);}
```

6.15.3.3 const double & MatrixReadOperator::element (const unsigned int *row*, const unsigned int *column*) const

Returns a read-only reference to the data member at specified position

Parameters:
row Row position of desired element (zero indexed)
column Column position of desired element (zero indexed)

Definition at line 18 of file matrix_operator.cpp.

```
18 {return m_thisMatrix->m_matrixReadAccess->readElement(row, column);}
```

6.15.3.4 void MatrixReadOperator::error (const char ∗ *str*) const

For flagging errors - to be depreciated.

Definition at line 20 of file matrix_operator.cpp.

```
20 {m_thisMatrix->m_matrixReadAccess->error(str);}
```

6.15.3.5 const unsigned int MatrixReadOperator::getColumns () const

Returns the number of columns the matrix has.

Definition at line 16 of file matrix_operator.cpp.

```
16 {return m_thisMatrix->getColumns();}
```

6.15.3.6 const double ∗ MatrixReadOperator::getDataPointer () const

Returns the a pointer to the matrix data array.

Definition at line 17 of file matrix_operator.cpp.

```
17 {return m_thisMatrix->m_matrixReadAccess->getDataPointer();}
```

6.15.3.7 const unsigned int MatrixReadOperator::getRows () const

Returns the number of rows the matrix has.

Definition at line 15 of file matrix_operator.cpp.

```
15 {return m_thisMatrix->getRows();}
```

6.15.4 Member Data Documentation

6.15.4.1 MatrixAliasConstant∗ MatrixReadOperator::m_thisMatrix `[protected]`

Pointer to owner matrix.

Reimplemented in **SquareMatrixReadOperator** (p. 217), **RowVectorReadOperator** (p. 164), and **ColumnVectorReadOperator** (p. 32).

Definition at line 45 of file matrix_operator.h.

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

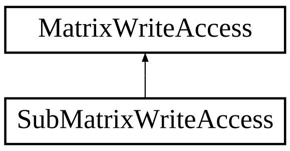
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.16 MatrixWriteAccess Class Reference

Write access for matrix classes.

`#include <matrix_access.h>`

Inheritance diagram for MatrixWriteAccess::



Public Member Functions

- **MatrixWriteAccess** (**MatrixAlias** ∗mA)

  Constructor.

- virtual double & **writeElement** (const unsigned int row, const unsigned int column) const
- virtual double & **writeElement** (const unsigned int index) const
- const unsigned int **getRows** () const

  Returns the number of rows the matrix has.

- const unsigned int **getColumns** () const

  Returns the number of columns the matrix has.

- double ∗ **getDataPointer** () const

  Returns the a pointer to the matrix data array.

- void **setDataPointer** (double ∗data) const

  Sets the data pointer.

- void **setRows** (const unsigned int num) const

  Sets the number of rows.

- void **setColumns** (const unsigned int num) const

  Sets the number of columns.

- void **error** (const char ∗str) const

  For flagging errors - to be depreciated.

Protected Attributes

- **MatrixAlias** ∗ **m_matrixAlias**

  Pointer to owner matrix.

6.16.1 Detailed Description

Write access for matrix classes.

Author:
Lee Netherton

The MatrixWriteAccess class provides basic write operatons for the **MatrixWrite-Operator**(p. 85) classes, and the matrix class itself. Each writable matrix class will have a pointer to a MatrixWriteAccess, or one if its derivatives.

Definition at line 128 of file matrix_access.h.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 MatrixWriteAccess::MatrixWriteAccess (MatrixAlias ∗ *mA*) [inline]

Constructor.

Sets pointer to owner matrix

Parameters:
mA Pointer to owner matrix

Definition at line 141 of file matrix_access.h.

```
141                                                   : m_matrixAlias(mA)
142             {
143                     #ifdef DEBUG_CONSTRUCT
144                             printf("Constructor: MatrixWriteAccess(MatrixAlias * mA)\n");
145                     #endif
146             }
```

6.16.3 Member Function Documentation

6.16.3.1 void MatrixWriteAccess::error (const char ∗ *str*) const

For flagging errors - to be depreciated.

Definition at line 108 of file matrix_access.cpp.

```
108                                                  {
109
110         // Just a printf for now
111         printf(str);
112 }
```

6.16.3.2 const unsigned int MatrixWriteAccess::getColumns () const

Returns the number of columns the matrix has.

Definition at line 96 of file matrix_access.cpp.

```
96 {return m_matrixAlias->getColumns();}
```

6.16.3.3 double * MatrixWriteAccess::getDataPointer () const

Returns the a pointer to the matrix data array.

Definition at line 97 of file matrix_access.cpp.

```
97 {return m_matrixAlias->m_matrixContainer->getDataPointer();}
```

6.16.3.4 const unsigned int MatrixWriteAccess::getRows () const

Returns the number of rows the matrix has.

Definition at line 95 of file matrix_access.cpp.

```
95 {return m_matrixAlias->getRows();}
```

6.16.3.5 void MatrixWriteAccess::setColumns (const unsigned int *num*) const

Sets the number of columns.

Definition at line 101 of file matrix_access.cpp.

```
101 {m_matrixAlias->m_matrixContainer->setColumns(num);}
```

6.16.3.6 void MatrixWriteAccess::setDataPointer (double * *data*) const

Sets the data pointer.

Definition at line 99 of file matrix_access.cpp.

```
99 {m_matrixAlias->m_matrixContainer->setDataPointer(data);}
```

6.16.3.7 void MatrixWriteAccess::setRows (const unsigned int *num*) const

Sets the number of rows.

Definition at line 100 of file matrix_access.cpp.

```
100 {m_matrixAlias->m_matrixContainer->setRows(num);}
```

6.16.3.8 double & MatrixWriteAccess::writeElement (const unsigned int *index*) const `[virtual]`

Returns a writable reference to the data member at specified position

Parameters:
index Row-wise position of element (zero-indexed)

Reimplemented in **SubMatrixWriteAccess** (p. 242).

Definition at line 80 of file matrix_access.cpp.

```
{
        if (index >= (getRows()*getColumns())) {
                error("MatrixAlias::element : Subscript out of range\n");
                return getDataPointer()[0];
        }

        return getDataPointer()[index];

}
```

6.16.3.9 double & MatrixWriteAccess::writeElement (const unsigned int *row*, const unsigned int *column*) const [virtual]

Returns a writable reference to the data member at specified position

Parameters:

row Row position of desired element (zero indexed)

column Column position of desired element (zero indexed)

Reimplemented in **SubMatrixWriteAccess** (p. 242).

Definition at line 69 of file matrix_access.cpp.

```
{
        if (row >= getRows() || column >= getColumns()) {
                error("MatrixAlias::element : Subscript out of range\n");
                return getDataPointer()[0];
        }

        return getDataPointer()[row*(getColumns()) + column];

}
```

6.16.4 Member Data Documentation

6.16.4.1 MatrixAlias∗ MatrixWriteAccess::m_matrixAlias [protected]

Pointer to owner matrix.

Definition at line 133 of file matrix_access.h.

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

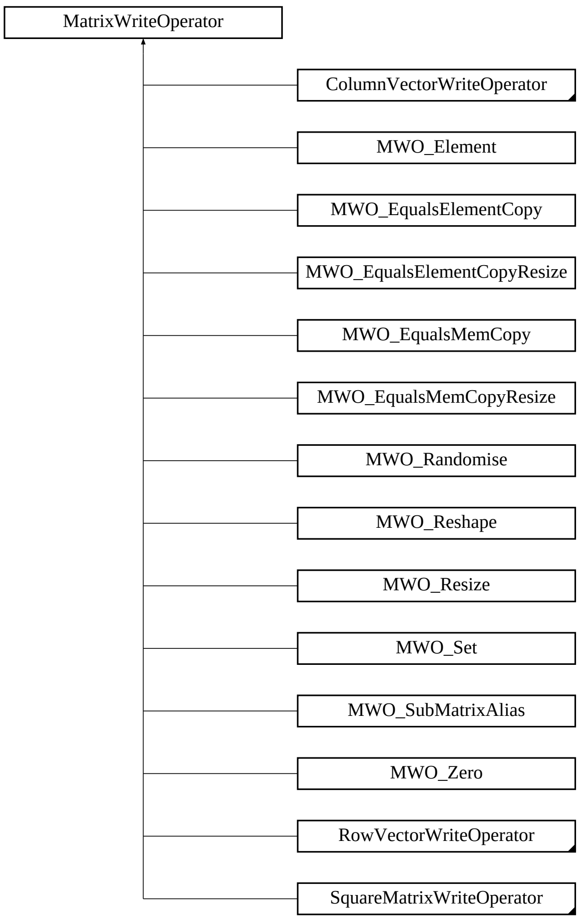
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_access.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_access.cpp**

6.17 MatrixWriteOperator Class Reference

Base class for MatrixWriteOperators.

`#include <matrix_operator.h>`

Inheritance diagram for MatrixWriteOperator::



Public Member Functions

- **MatrixWriteOperator** ()

  Default Constructor.

- **MatrixWriteOperator** (**MatrixAlias** ∗matrixAlias)

  Full Constructor.

- void **_constructMatrixWriteOperator** (**MatrixAlias** ∗matrixAlias)

  Manual Constructor.

- const unsigned int **getRows** () const

  Returns the number of rows the matrix has.

- const unsigned int **getColumns** () const

  Returns the number of columns the matrix has.

- double ∗ **getDataPointer** () const

  Returns the a pointer to the matrix data array.

- void **setDataPointer** (double ∗data) const

  Sets the data pointer.

- void **setRows** (const unsigned int num) const

  Sets the number of rows.

- void **setColumns** (const unsigned int num) const

  Sets the number of columns.

- double & **element** (const unsigned int row, const unsigned int column) const
- double & **element** (const unsigned int index) const
- void **error** (const char ∗str) const

  For flagging errors - to be depreciated.

Protected Attributes

- **MatrixAlias** ∗ **m_thisMatrix**

  Pointer to owner matrix.

6.17.1 Detailed Description

Base class for MatrixWriteOperators.

Author:
Lee Netherton

A MatrixWriteOperator is a class which will perform a write operation on a matrix. The MatrixWriteOperator class is never instantiated directly, but serves as a bass for derived operator classes. It provides basic write operatons for its derivatives.

Definition at line 100 of file matrix_operator.h.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 MatrixWriteOperator::MatrixWriteOperator () [inline]

Default Constructor.

Creates an empty operator class which can then be more full constructed using **_constructMatrixWriteOperator()**(p. 87) Definition at line 113 of file matrix_operator.h.

```
113 {}
```

6.17.2.2 MatrixWriteOperator::MatrixWriteOperator (MatrixAlias * *matrixAlias*) `[inline]`

Full Constructor.

Creates an operator class which takes and stores a pointer to an owner matrix

Parameters:

matrixAlias Pointer to owner matrix

Definition at line 119 of file matrix_operator.h.

```
119 : m_thisMatrix(matrixAlias) {}
```

6.17.3 Member Function Documentation

6.17.3.1 void MatrixWriteOperator::_constructMatrixWriteOperator (MatrixAlias * *matrixAlias*) `[inline]`

Manual Constructor.

Constructs the class manually by setting the owner pointer

Parameters:

matrixAlias Pointer to owner matrix

Definition at line 125 of file matrix_operator.h.

```
125 {m_thisMatrix = matrixAlias;}
```

6.17.3.2 double & MatrixWriteOperator::element (const unsigned int *index*) const

Returns a writable reference to the data member at specified position

Parameters:

index Row-wise position of element (zero-indexed)

Definition at line 34 of file matrix_operator.cpp.

```
34 {return m_thisMatrix->m_matrixWriteAccess->writeElement(index);}
```

6.17.3.3 double & MatrixWriteOperator::element (const unsigned int *row*, const unsigned int *column*) const

Returns a writable reference to the data member at specified position

Parameters:

row Row position of desired element (zero indexed)

column Column position of desired element (zero indexed)

Definition at line 33 of file matrix_operator.cpp.

```
33 {return m_thisMatrix->m_matrixWriteAccess->writeElement(row, column);}
```

6.17.3.4 void MatrixWriteOperator::error (const char * *str*) const

For flagging errors - to be depreciated.

Definition at line 35 of file matrix_operator.cpp.

```
35 {m_thisMatrix->m_matrixWriteAccess->error(str);}
```

6.17.3.5 const unsigned int MatrixWriteOperator::getColumns () const

Returns the number of columns the matrix has.

Definition at line 28 of file matrix_operator.cpp.

```
28 {return m_thisMatrix->getColumns();}
```

6.17.3.6 double * MatrixWriteOperator::getDataPointer () const

Returns the a pointer to the matrix data array.

Definition at line 29 of file matrix_operator.cpp.

```
29 {return m_thisMatrix->m_matrixWriteAccess->getDataPointer();}
```

6.17.3.7 const unsigned int MatrixWriteOperator::getRows () const

Returns the number of rows the matrix has.

Definition at line 27 of file matrix_operator.cpp.

```
27 {return m_thisMatrix->getRows();}
```

6.17.3.8 void MatrixWriteOperator::setColumns (const unsigned int *num*) const

Sets the number of columns.

Definition at line 32 of file matrix_operator.cpp.

```
32 {m_thisMatrix->m_matrixWriteAccess->setColumns(num);}
```

6.17.3.9 void MatrixWriteOperator::setDataPointer (double * *data*) const

Sets the data pointer.

Definition at line 30 of file matrix_operator.cpp.

```
30 {m_thisMatrix->m_matrixWriteAccess->setDataPointer(data);}
```

6.17.3.10 void MatrixWriteOperator::setRows (const unsigned int *num*) const

Sets the number of rows.

Definition at line 31 of file matrix_operator.cpp.

```
31 {m_thisMatrix->m_matrixWriteAccess->setRows(num);}
```

6.17.4 Member Data Documentation

6.17.4.1 MatrixAlias∗ MatrixWriteOperator::m_thisMatrix `[protected]`

Pointer to owner matrix.

Reimplemented in **SquareMatrixWriteOperator** (p. 219), **RowVectorWriteOperator** (p. 166), and **ColumnVectorWriteOperator** (p. 35).

Definition at line 105 of file matrix_operator.h.

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

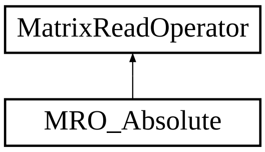
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.18 MRO_Absolute Class Reference

Returns an absolute matrix.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_Absolute::



Public Member Functions

- **Matrix operator()** () const
  Return a matrix containing the absolute values of this matrix.

6.18.1 Detailed Description

Returns an absolute matrix.

Author:
Lee Netherton and Peter Mendham

Definition at line 594 of file matrix_operator.h.

6.18.2 Member Function Documentation

6.18.2.1 Matrix MRO_Absolute::operator() () const

Return a matrix containing the absolute values of this matrix.

Definition at line 265 of file matrix_operator.cpp.

```
                                    {

        Matrix newMatrix(getColumns(), getRows());
        unsigned int i, j;

        for (i = 0; i < getRows(); i++) {
                for (j = 0; j < getColumns(); j++) {
                        newMatrix.element(i, j) = fabs(element(i, j));
                }
        }

        return newMatrix;
}
```

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

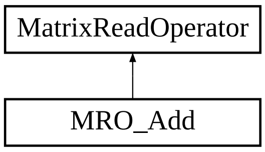
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.19 MRO_Add Class Reference

Add matrix.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_Add::



Public Member Functions

- **Matrix operator()** (const **MatrixAliasConstant** &operand) const
  Add matrix to another matrix.

6.19.1 Detailed Description

Add matrix.

Author:
Lee Netherton and Peter Mendham

Definition at line 457 of file matrix_operator.h.

6.19.2 Member Function Documentation

6.19.2.1 Matrix MRO_Add::operator() (const MatrixAliasConstant & *operand*) const

Add matrix to another matrix.

Parameters:
operand **Matrix**(p. 44) to add to

Definition at line 120 of file matrix_operator.cpp.

```
{
        Matrix result(getRows(), getColumns());
        unsigned int i;

        if (operand.getRows() != getRows() || operand.getColumns() != getColumns()) {
                error("MRO_Add : Dimensions are not consistent\n");
                return result; // Return empty matrix
        }

        for (i = 0; i < getRows()*getColumns(); i++) {
                result.element(i) = element(i) + operand.element(i);
        }

        return result;
}
```

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

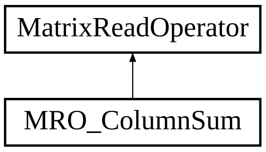
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.20 MRO_ColumnSum Class Reference

Returns a row vector which is the sum of all the columns.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_ColumnSum::



Public Member Functions

- **RowVector operator()** () const

  Adds columns to produce a row vector.

6.20.1 Detailed Description

Returns a row vector which is the sum of all the columns.

Author:
Lee Netherton and Peter Mendham

Definition at line 616 of file matrix_operator.h.

6.20.2 Member Function Documentation

6.20.2.1 RowVector MRO_ColumnSum::operator() () const

Adds columns to produce a row vector.

Definition at line 302 of file matrix_operator.cpp.

```
                                    {

        RowVector newVector(getColumns());
        unsigned int i, j;

        for (j = 0; j < getColumns(); j++) {
                newVector.element(j) = 0;
                for (i = 0; i < getRows(); i++) {
                        newVector.element(j) += element(i, j);
                }
        }

        return newVector;
}
```

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

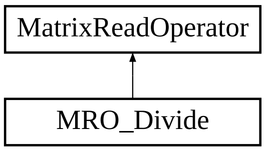
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.21 MRO_Divide Class Reference

Divide matrix.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_Divide::



Public Member Functions

- **Matrix operator()** (const double &operand) const
  Divide matrix by a scalar.

6.21.1 Detailed Description

Divide matrix.

Author:
Lee Netherton and Peter Mendham

Definition at line 492 of file matrix_operator.h.

6.21.2 Member Function Documentation

6.21.2.1 Matrix MRO_Divide::operator() (const double & *operand*) const

Divide matrix by a scalar.

Parameters:
operand Scalar to divide by

Definition at line 106 of file matrix_operator.cpp.

```
{
        Matrix result(getRows(), getColumns());
        unsigned int i, j;

        for (i = 0; i < getRows(); i++) {
                for (j = 0; j < getColumns(); j++) {
                        result.element(i, j) = element(i, j) / operand;
                }
        }

        return result;
}
```

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

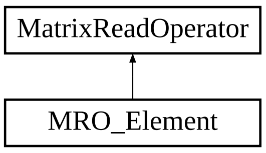
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.22 MRO_Element Class Reference

Read matrix element.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_Element::



Public Member Functions

- const double & **operator()** (const unsigned int row, const unsigned int column) const

  Read element given row, column position.

- const double & **operator()** (const unsigned int index) const

  Read element given row-wise index position.

6.22.1 Detailed Description

Read matrix element.

Author:
Lee Netherton

Definition at line 412 of file matrix_operator.h.

6.22.2 Member Function Documentation

6.22.2.1 const double& MRO_Element::operator() (const unsigned int *index*) const `[inline]`

Read element given row-wise index position.

Parameters:
index Index of element (zero-indexed)

Definition at line 422 of file matrix_operator.h.

```
422 {return element(index);}
```

6.22.2.2 const double& MRO_Element::operator() (const unsigned int *row*, const unsigned int *column*) const `[inline]`

Read element given row, column position.

Parameters:

row Row position (zero-indexed)

column Column position (zero-indexed)

Definition at line 418 of file matrix_operator.h.

```
418 {return element(row,column);}
```

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

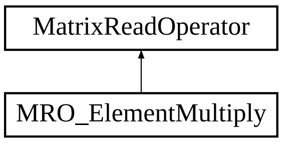
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**

6.23 MRO_ElementMultiply Class Reference

Perform element by element multiplication.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_ElementMultiply::



Public Member Functions

- **Matrix operator()** (const **MatrixAliasConstant** &operand) const

  Multiply matatix by a matrix.

6.23.1 Detailed Description

Perform element by element multiplication.

Author:

Lee Netherton and Peter Mendham

Definition at line 445 of file matrix_operator.h.

6.23.2 Member Function Documentation

6.23.2.1 Matrix MRO_ElementMultiply::operator() (const MatrixAliasConstant & *operand*) const

Multiply matatix by a matrix.

Parameters:

operand **Matrix**(p. 44) to multiply by

Definition at line 72 of file matrix_operator.cpp.

```
{
        Matrix result(getRows(), getColumns());
        unsigned int i, j;

        if (operand.getRows() != getRows() || operand.getColumns() != getColumns()) {
                error("MatrixAlias::elementMultiply : Dimensions are not consistent\n");
                return result; // Return empty matrix
        }

        for (i = 0; i < getRows(); i++) {
                for (j = 0; j < getColumns(); j++) {
                        result.element(i, j) = operand.element(i,j) * element(i, j);
                }
        }

        return result;
}
```

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

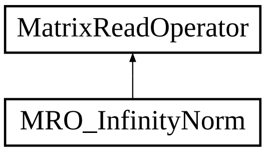
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.24 MRO_InfinityNorm Class Reference

Returns the infinity norm of this matrix.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_InfinityNorm::



Public Member Functions

- double **operator()** () const

  Returns the infinity norm of this matrix.

6.24.1 Detailed Description

Returns the infinity norm of this matrix.

Author:
Lee Netherton and Peter Mendham

Definition at line 649 of file matrix_operator.h.

6.24.2 Member Function Documentation

6.24.2.1 double MRO_InfinityNorm::operator() () const

Returns the infinity norm of this matrix.

Definition at line 361 of file matrix_operator.cpp.

```
361                                         {
362
363         ColumnVector newVector(getRows());
364         unsigned int i, j;
365
366         for (i = 0; i < getRows(); i++) {
367                 newVector.element(i) = 0;
368                 for (j = 0; j < getColumns(); j++) {
369                         newVector.element(i) += fabs(element(i, j));
370                 }
371         }
372
373         return newVector.maximum();
374 }
```

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

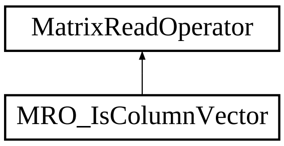
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.25 MRO_IsColumnVector Class Reference

Is this a column vector?.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_IsColumnVector::



Public Member Functions

- int **operator()** () const

  Returns 1 if this matrix's number of columns = 1.

6.25.1 Detailed Description

Is this a column vector?.

Author:
Lee Netherton

Definition at line 572 of file matrix_operator.h.

6.25.2 Member Function Documentation

6.25.2.1 int MRO_IsColumnVector::operator() () const

Returns 1 if this matrix's number of columns = 1.

Definition at line 233 of file matrix_operator.cpp.

```
{
        if(getColumns() == 1)
                return 1; // Matrix is column vector

        return 0; //  Matrix is NOT column vector
}
```

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

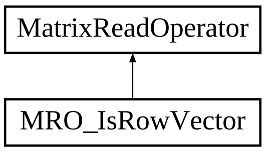
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.26 MRO_IsRowVector Class Reference

Is this a row vector?.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_IsRowVector::



Public Member Functions

- int **operator()** () const

  Returns 1 if this matrix's number of rows = 1.

6.26.1 Detailed Description

Is this a row vector?.

Author:
Lee Netherton

Definition at line 562 of file matrix_operator.h.

6.26.2 Member Function Documentation

6.26.2.1 int MRO_IsRowVector::operator() () const

Returns 1 if this matrix's number of rows = 1.

Definition at line 225 of file matrix_operator.cpp.

```
{
        if(getRows() == 1)
                return 1; // Matrix is row vector

        return 0; //  Matrix is NOT row vector
}
```

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

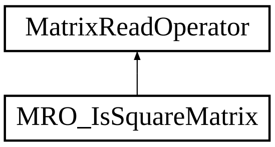
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.27 MRO_IsSquareMatrix Class Reference

Is this a square matrix?.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_IsSquareMatrix::



Public Member Functions

- int **operator()** () const
  Returns 1 if this matrix's rows = columns.

6.27.1 Detailed Description

Is this a square matrix?.

Author:
Lee Netherton

Definition at line 552 of file matrix_operator.h.

6.27.2 Member Function Documentation

6.27.2.1 int MRO_IsSquareMatrix::operator() () const

Returns 1 if this matrix's rows = columns.

Definition at line 217 of file matrix_operator.cpp.

```
{
        if(getRows() == getColumns())
                return 1; // Matrix is square

        return 0; //  Matrix is NOT square
}
```

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

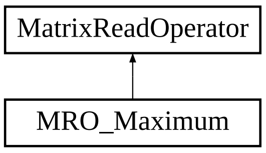
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.28 MRO_Maximum Class Reference

Returns the largest element.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_Maximum::



Public Member Functions

- double **operator()** () const
  Returns maximum value in matrix.

6.28.1 Detailed Description

Returns the largest element.

Author:
Lee Netherton and Peter Mendham

Definition at line 627 of file matrix_operator.h.

6.28.2 Member Function Documentation

6.28.2.1 double MRO_Maximum::operator() () const

Returns maximum value in matrix.

Definition at line 321 of file matrix_operator.cpp.

```
321                                       {
322 
323         unsigned int i, j;
324         double maxVal;
325 
326         maxVal = element(0);
327 
328         for (i = 0; i < getRows(); i++) {
329                 for (j = 0; j < getColumns(); j++) {
330                         maxVal = (element(i, j) > maxVal) ? element(i, j) : maxVal;
331                 }
332         }
333 
334         return maxVal;
335 }
```

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

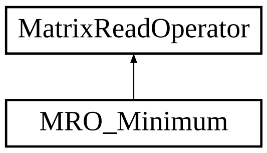
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.29 MRO_Minimum Class Reference

Returns the smallest element.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_Minimum::



Public Member Functions

- double **operator()** () const

  Returns minumum value in matrix.

6.29.1 Detailed Description

Returns the smallest element.

Author:
Lee Netherton and Peter Mendham

Definition at line 638 of file matrix_operator.h.

6.29.2 Member Function Documentation

6.29.2.1 double MRO_Minimum::operator() () const

Returns minumum value in matrix.

Definition at line 341 of file matrix_operator.cpp.

```
341                                     {
342 
343         unsigned int i, j;
344         double minVal;
345 
346         minVal = element(0);
347 
348         for (i = 0; i < getRows(); i++) {
349                 for (j = 0; j < getColumns(); j++) {
350                         minVal = (element(i, j) < minVal) ? element(i, j) : minVal;
351                 }
352         }
353 
354         return minVal;
355 }
```

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

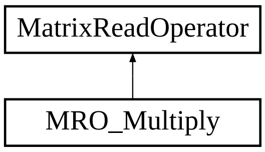
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.30 MRO_Multiply Class Reference

Multiply matrix.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_Multiply::



Public Member Functions

- **Matrix operator()** (const **MatrixAliasConstant** &operand) const

  Multiply matatix by a matrix.

- **Matrix operator()** (const double &operand) const

  Multiply matrix by a scalar.

6.30.1 Detailed Description

Multiply matrix.

Author:

Lee Netherton and Peter Mendham

Definition at line 429 of file matrix_operator.h.

6.30.2 Member Function Documentation

6.30.2.1 Matrix MRO_Multiply::operator() (const double & *operand*) const

Multiply matrix by a scalar.

Parameters:

operand Scalar to multiply by

Definition at line 92 of file matrix_operator.cpp.

```
{
        Matrix result(getRows(), getColumns());
        unsigned int i, j;

        for (i = 0; i < getRows(); i++) {
                for (j = 0; j < getColumns(); j++) {
                        result.element(i, j) = element(i, j) * operand;
                 }
         }

        return result;
}
```

6.30.2.2 Matrix MRO_Multiply::operator() (const MatrixAliasConstant & *operand*) const

Multiply matatix by a matrix.

Parameters:

operand **Matrix**(p. 44) to multiply by

Definition at line 43 of file matrix_operator.cpp.

```
{
        Matrix result(getRows(), operand.getColumns());
        double thisElement;
        unsigned int i, j, k;


        if (operand.getRows() != getColumns()) {
                error("MatrixAlias::operator* : Dimensions are not consistent\n");
                return result; // Return empty matrix
        }

        for (i = 0; i < getRows(); i++) {

                for (j = 0; j < operand.getColumns(); j++) {

                        thisElement = 0;

                        for (k = 0; k < getColumns(); k++) {
                                thisElement += element(i, k) * operand.element(k, j);
                        }

                        result.element(i, j) = thisElement;
                }
        }

        return result;
}
```

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

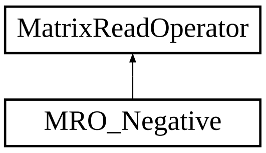
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.31 MRO_Negative Class Reference

Negative.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_Negative::



Public Member Functions

- **Matrix operator()** () const

  Returns the negative of this matrix.

6.31.1 Detailed Description

Negative.

Author:
Lee Netherton and Peter Mendham

Definition at line 481 of file matrix_operator.h.

6.31.2 Member Function Documentation

6.31.2.1 Matrix MRO_Negative::operator() () const

Returns the negative of this matrix.

Definition at line 154 of file matrix_operator.cpp.

```
155 {
156         Matrix result(getRows(), getColumns());
157 
158         unsigned int i, j;
159 
160         for (i = 0; i < getRows(); i++) {
161 
162                 for (j = 0; j < getColumns(); j++) {
163 
164                         result.element(i, j) = -1 * element(i, j);
165                 }
166         }
167 
168         return result;
169 }
```

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

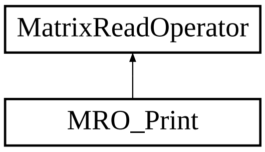
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.32 MRO_Print Class Reference

Print matrix to screen.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_Print::



Public Member Functions

- void **operator()** () const

  Prints the matrix to the screen.

6.32.1 Detailed Description

Print matrix to screen.

Author:

Lee Netherton and Peter Mendham

Definition at line 504 of file matrix_operator.h.

6.32.2 Member Function Documentation

6.32.2.1 void MRO_Print::operator() () const

Prints the matrix to the screen.

Definition at line 171 of file matrix_operator.cpp.

```
{

        unsigned int i, j;

        for (i = 0; i < getRows(); i++) {
                printf("[");
                for (j = 0; j < (getColumns()-1); j++) {
                        printf("%.2lf  ", element(i, j));
                }
                printf("%.2lf", element(i, getColumns()-1));
                printf("]\n");
        }
        printf("\n");
}
```

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

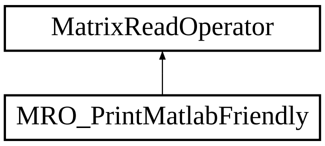
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.33 MRO_PrintMatlabFriendly Class Reference

Print matrix to screen (MATLAB Friendly).

`#include <matrix_operator.h>`

Inheritance diagram for MRO_PrintMatlabFriendly::



Public Member Functions

- void **operator()** () const

  Prints the matrix to the screen in a form that can be pasted into MATLAB.

6.33.1 Detailed Description

Print matrix to screen (MATLAB Friendly).

Author:
Lee Netherton

Definition at line 515 of file matrix_operator.h.

6.33.2 Member Function Documentation

6.33.2.1 void MRO_PrintMatlabFriendly::operator() () const

Prints the matrix to the screen in a form that can be pasted into MATLAB.

Definition at line 187 of file matrix_operator.cpp.

```
{

        unsigned int i, j;
        printf("[\n");
        for (i = 0; i < getRows(); i++) {
                for (j = 0; j < (getColumns()-1); j++) {
                        printf("%.2lf  ", element(i, j));
                }
                printf("%.2lf;", element(i, getColumns()-1));
                printf("\n");
        }
        printf("]\n");
}
```

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

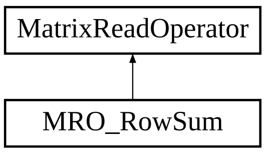
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.34 MRO_RowSum Class Reference

Returns a column vector which is the sum of all the rows.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_RowSum::



Public Member Functions

- **ColumnVector operator()** () const

  Adds rows to produce a column vector.

6.34.1 Detailed Description

Returns a column vector which is the sum of all the rows.

Author:

Lee Netherton and Peter Mendham

Definition at line 605 of file matrix_operator.h.

6.34.2 Member Function Documentation

6.34.2.1 ColumnVector MRO_RowSum::operator() () const

Adds rows to produce a column vector.

Definition at line 283 of file matrix_operator.cpp.

```
283                                     {
284 
285         ColumnVector newVector(getRows());
286         unsigned int i, j;
287 
288         for (i = 0; i < getRows(); i++) {
289                 newVector.element(i) = 0;
290                 for (j = 0; j < getColumns(); j++) {
291                         newVector.element(i) += element(i, j);
292                 }
293         }
294 
295         return newVector;
296 }
```

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

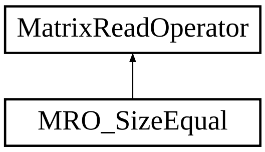
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.35 MRO_SizeEqual Class Reference

Comapre sizes.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_SizeEqual::



Public Member Functions

- int **operator()** (const **MatrixAliasConstant** &operand) const
  Returns 1 if the sizes of the matrices are equal, 0 if not.

6.35.1 Detailed Description

Comapre sizes.

Author:
Lee Netherton

Definition at line 541 of file matrix_operator.h.

6.35.2 Member Function Documentation

6.35.2.1 int MRO_SizeEqual::operator() (const MatrixAliasConstant & *operand*) const

Returns 1 if the sizes of the matrices are equal, 0 if not.

Parameters:
operand **Matrix**(p. 44) with which to compare sizes

Definition at line 209 of file matrix_operator.cpp.

```
{
        if(operand.getRows() == getRows() && operand.getColumns() == getColumns())
                return 1; // Size is equal

        return 0; // Size is NOT equal
}
```

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

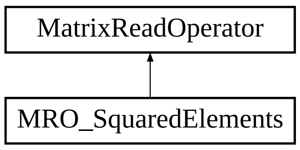
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.36 MRO_SquaredElements Class Reference

Returns a matrix with all the elements the square of this one's.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_SquaredElements::



Public Member Functions

- **Matrix operator()** () const
  Returns a matrix with all the elements the square of this one's.

6.36.1 Detailed Description

Returns a matrix with all the elements the square of this one's.

Author:
Lee Netherton and Peter Mendham

Definition at line 660 of file matrix_operator.h.

6.36.2 Member Function Documentation

6.36.2.1 Matrix MRO_SquaredElements::operator() () const

Returns a matrix with all the elements the square of this one's.

Definition at line 381 of file matrix_operator.cpp.

```
381                                         {
382 
383         Matrix newMatrix(m_thisMatrix);
384         unsigned int i, j;
385 
386         for (i = 0; i < getRows(); i++) {
387                 for (j = 0; j < getColumns(); j++) {
388                         newMatrix.element(i, j) *= element(i, j);
389                 }
390         }
391 
392         return newMatrix;
393 }
```

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

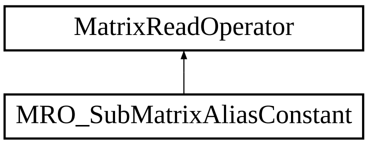
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.37 MRO_SubMatrixAliasConstant Class Reference

Get a SubMatrix.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_SubMatrixAliasConstant::



Public Member Functions

- **SubMatrixAliasConstant operator()** (const unsigned int rowStart, const unsigned int rowEnd, const unsigned int columnStart, const unsigned int columnEnd) const

  Returns a SubMatrix of this matrix give row/column start/end positions.

6.37.1 Detailed Description

Get a SubMatrix.

Author:
Lee Netherton

Definition at line 526 of file matrix_operator.h.

6.37.2 Member Function Documentation

6.37.2.1 SubMatrixAliasConstant MRO_SubMatrixAliasConstant::operator() (const unsigned int *rowStart*, const unsigned int *rowEnd*, const unsigned int *columnStart*, const unsigned int *columnEnd*) const

Returns a SubMatrix of this matrix give row/column start/end positions.

Parameters:
rowStart Starting row (zero-indexed)
rowEnd Finishing row (inclusive, zero-indexed)
columnStart Starting column (zero-indexed)
columnEnd Finishing column (inclusive, zero-indexed)

Definition at line 202 of file matrix_operator.cpp.

```
{
        SubMatrixAliasConstant subMatrixAliasConstant(m_thisMatrix, rowStart, rowEnd, columnStart, columnEnd);

        return subMatrixAliasConstant;
}
```

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

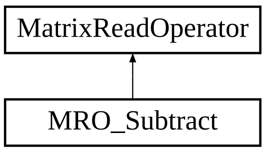
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.38 MRO_Subtract Class Reference

Subtract matrix.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_Subtract::



Public Member Functions

- **Matrix operator()** (const **MatrixAliasConstant** &operand) const
  Subtract another matrix.

6.38.1 Detailed Description

Subtract matrix.

Author:
Lee Netherton and Peter Mendham

Definition at line 469 of file matrix_operator.h.

6.38.2 Member Function Documentation

6.38.2.1 Matrix MRO_Subtract::operator() (const MatrixAliasConstant & *operand*) const

Subtract another matrix.

Parameters:
operand **Matrix**(p. 44) to subtract

Definition at line 137 of file matrix_operator.cpp.

```
{
        Matrix result(getRows(), getColumns());
        unsigned int i;

        if (operand.getRows() != getRows() || operand.getColumns() != getColumns()) {
                error("MRO_Subtract : Dimensions are not consistent\n");
                return result; // Return empty matrix
        }

        for (i = 0; i < getRows()*getColumns(); i++) {
                result.element(i) = element(i) - operand.element(i);
        }

        return result;
}
```

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

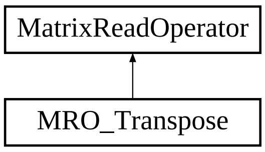
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.39 MRO_Transpose Class Reference

Returns a transposed matrix.

`#include <matrix_operator.h>`

Inheritance diagram for MRO_Transpose::



Public Member Functions

- **Matrix operator()** () const
  Return the transpose of this matrix.

6.39.1 Detailed Description

Returns a transposed matrix.

Author:
Lee Netherton and Peter Mendham

Definition at line 583 of file matrix_operator.h.

6.39.2 Member Function Documentation

6.39.2.1 Matrix MRO_Transpose::operator() () const

Return the transpose of this matrix.

Definition at line 247 of file matrix_operator.cpp.

```
                                    {

        Matrix newMatrix(getColumns(), getRows());
        unsigned int i, j;

        for (i = 0; i < getRows(); i++) {
                for (j = 0; j < getColumns(); j++) {
                        newMatrix.element(j, i) = element(i, j);
                }
        }

        return newMatrix;
}
```

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

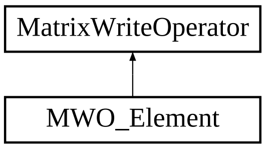
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.40 MWO_Element Class Reference

Writable matrix element.

`#include <matrix_operator.h>`

Inheritance diagram for MWO_Element::



Public Member Functions

- double & **operator()** (const unsigned int row, const unsigned int column) const

  Return writable element given row, column position.

- double & **operator()** (const unsigned int index) const

  Return writable given row-wise index position.

6.40.1 Detailed Description

Writable matrix element.

Author:
Lee Netherton

Definition at line 676 of file matrix_operator.h.

6.40.2 Member Function Documentation

6.40.2.1 double& MWO_Element::operator() (const unsigned int *index*) const `[inline]`

Return writable given row-wise index position.

Parameters:
index Index of element (zero-indexed)

Definition at line 686 of file matrix_operator.h.

```
686 {return element(index);}
```

6.40.2.2 double& MWO_Element::operator() (const unsigned int *row*, const unsigned int *column*) const [inline]

Return writable element given row, column position.

Parameters:

row Row position (zero-indexed)

column Column position (zero-indexed)

Definition at line 682 of file matrix_operator.h.

```
682 {return element(row,column);}
```

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

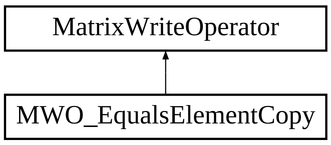
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**

6.41 MWO_EqualsElementCopy Class Reference

Copy elements (From matrix of same size).

`#include <matrix_operator.h>`

Inheritance diagram for MWO_EqualsElementCopy::



Public Member Functions

- void **operator()** (const **MatrixAliasConstant** &copy) const
  Copy element-by-element from `copy` *into this matrix.*

6.41.1 Detailed Description

Copy elements (From matrix of same size).

Author:
Lee Netherton

Definition at line 693 of file matrix_operator.h.

6.41.2 Member Function Documentation

6.41.2.1 void MWO_EqualsElementCopy::operator() (const MatrixAliasConstant & *copy*) const

Copy element-by-element from `copy` into this matrix.

Matrices must be of same dimentions as there is no resize capabilities.

Parameters:
copy **Matrix**(p. 44) to copy

Definition at line 403 of file matrix_operator.cpp.

```
{
                unsigned int i, j;

                //printf("MWO_EqualsElementCopy called\n");

                if(m_thisMatrix->sizeEqual(copy) == 0)
                {
                        error("MWO_EqualsElementCopy : Dimensions are not consistent\n");
                        return;
                }
```

```

        /*if(copy.getRows() != getRows() || copy.getColumns() != getColumns()) {
                error("MWO_EqualsElementCopy : Dimensions are not consistent\n");
                return;
        }*/

        for (i = 0; i < getRows(); i++) {
                for (j = 0; j < getColumns(); j++) {

                        element(i, j) = copy.element(i, j);
                }
        }
}
```

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

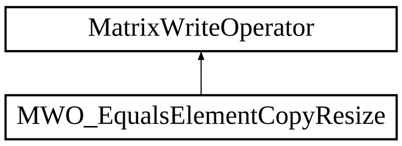
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.42 MWO_EqualsElementCopyResize Class Reference

Copy elements (Resize if necessesary).

`#include <matrix_operator.h>`

Inheritance diagram for MWO_EqualsElementCopyResize::



Public Member Functions

- void **operator()** (const **MatrixAliasConstant** &copy) const

    Copy element-by-element from `copy` *into this matrix.*

6.42.1 Detailed Description

Copy elements (Resize if necessesary).

Author:
Lee Netherton

Definition at line 721 of file matrix_operator.h.

6.42.2 Member Function Documentation

6.42.2.1 void MWO_EqualsElementCopyResize::operator() (const MatrixAliasConstant & *copy*) const

Copy element-by-element from `copy` into this matrix.

Matrix(p. 44) will be resized if there is a difference in dimentions.

Parameters:
copy **Matrix**(p. 44) to copy

Definition at line 428 of file matrix_operator.cpp.

```
{
                unsigned int i, j;

                //printf("MWO_EqualsElementCopyResize called\n");

                //printf("copy.getRows():%d\ngetRows():%d\ncopy.getColumns():%d\ncopy.getColumns():%d\n",copy.getRow

                if(copy.getRows() != getRows() || copy.getColumns() != getColumns()) {
                        // Resize!!

                        // Change values
                        setRows(copy.getRows());
                        setColumns(copy.getColumns());

                        // Delete old memory
                        delete[] getDataPointer();

                        // Allocate new memory
                        setDataPointer(new double[getRows()*getColumns()]);
                }


                for (i = 0; i < getRows(); i++) {
                        for (j = 0; j < getColumns(); j++) {
                                element(i, j) = copy.element(i, j);
                        }
                }

}
```

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

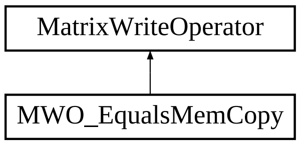
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.43 MWO_EqualsMemCopy Class Reference

Copy memory directly (From matrix of same size).

`#include <matrix_operator.h>`

Inheritance diagram for MWO_EqualsMemCopy::



Public Member Functions

- void **operator()** (const **MatrixAliasConstant** &copy) const
  Copy memory directly from `copy` *into this matrix.*

6.43.1 Detailed Description

Copy memory directly (From matrix of same size).

Author:
Lee Netherton

Definition at line 707 of file matrix_operator.h.

6.43.2 Member Function Documentation

6.43.2.1 void MWO_EqualsMemCopy::operator() (const MatrixAliasConstant & *copy*) const

Copy memory directly from `copy` into this matrix.

Matrices must be of same dimentions as there is no resize capabilities.

Parameters:
copy **Matrix**(p. 44) to copy

Definition at line 459 of file matrix_operator.cpp.

```
{
                unsigned int i, j;

                if(copy.getRows() != getRows() || copy.getColumns() != getColumns()) {
                        error("MWO_EqualsMemCopy : Dimensions are not consistent\n");
                        return;
                }

                // Perform a direct memory copy
                memcpy(getDataPointer(), copy.getDataPointer(), sizeof(double)*getRows()*getColumns());

}
```

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

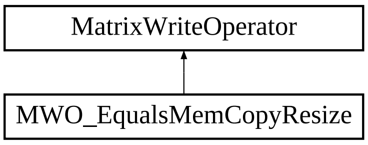
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.44 MWO_EqualsMemCopyResize Class Reference

Copy memory directly (Resize if necessesary).

`#include <matrix_operator.h>`

Inheritance diagram for MWO_EqualsMemCopyResize::



Public Member Functions

- void **operator()** (const **MatrixAliasConstant** &copy) const

    Copy memory directly from `copy` *into this matrix.*

6.44.1 Detailed Description

Copy memory directly (Resize if necessesary).

Author:
Lee Netherton

Definition at line 735 of file matrix_operator.h.

6.44.2 Member Function Documentation

6.44.2.1 void MWO_EqualsMemCopyResize::operator() (const MatrixAliasConstant & *copy*) const

Copy memory directly from `copy` into this matrix.

Matrix(p. 44) will be resized if there is a difference in dimentions.

Parameters:
copy **Matrix**(p. 44) to copy

Definition at line 473 of file matrix_operator.cpp.

```
{
                unsigned int i, j;

                if(copy.getRows() != getRows() || copy.getColumns() != getColumns()) {
                        // Resize!!

                        // Change values
                        setRows(copy.getRows());
                        setColumns(copy.getColumns());

                        // Delete old memory
                        delete[] getDataPointer();

                        // Allocate new memory
                        setDataPointer(new double[getRows()*getColumns()]);
                }

                // Perform a direct memory copy
                memcpy(getDataPointer(), copy.getDataPointer(), sizeof(double)*getRows()*getColumns());

}
```

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

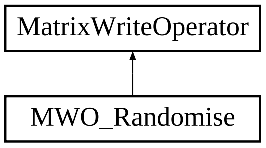
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.45 MWO_Randomise Class Reference

Randomise matrix.

`#include <matrix_operator.h>`

Inheritance diagram for MWO_Randomise::



Public Member Functions

- void **operator()** (const double val) const

  Make all the elements in this matrix equal some random value.

- void **operator()** () const

  Make all the elements in this matrix equal some random value (between 0 and 1).

6.45.1 Detailed Description

Randomise matrix.

Author:
Lee Netherton

Definition at line 801 of file matrix_operator.h.

6.45.2 Member Function Documentation

6.45.2.1 void MWO_Randomise::operator() () const

Make all the elements in this matrix equal some random value (between 0 and 1).

Definition at line 564 of file matrix_operator.cpp.

```
{
        unsigned int i;

        for (i = 0; i < getRows()*getColumns(); i++) {
                element(i) = _rand();
        }
}
```

6.45.2.2 void MWO_Randomise::operator() (const double *val*) const

Make all the elements in this matrix equal some random value.

Parameters:
val Max value

Definition at line 555 of file matrix_operator.cpp.

```
{
        unsigned int i;

        for (i = 0; i < getRows()*getColumns(); i++) {
                element(i) = _rand()*val;
        }
}
```

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

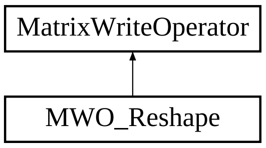
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.46 MWO_Reshape Class Reference

Reshape matrix (number of elements has to remain the same).

`#include <matrix_operator.h>`

Inheritance diagram for MWO_Reshape::



Public Member Functions

- void **operator()** (const unsigned int rows, const unsigned int columns) const

  Reshape matrix (number of elements has to remain the same).

6.46.1 Detailed Description

Reshape matrix (number of elements has to remain the same).

Author:

Lee Netherton

Definition at line 749 of file matrix_operator.h.

6.46.2 Member Function Documentation

6.46.2.1 void MWO_Reshape::operator() (const unsigned int *rows*, const unsigned int *columns*) const

Reshape matrix (number of elements has to remain the same).

Parameters:

rows Desired number of rows

columns Desired number of columns

Definition at line 496 of file matrix_operator.cpp.

```
{
        if((rows * columns) != (getRows() * getColumns())) {
                error("MWO_Reshape : Dimensions are not consistent\n");
                return;
        }

        // Change values
        setRows(rows);
        setColumns(columns);
}
```

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

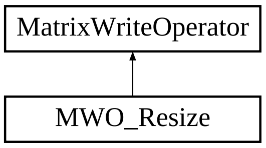
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.47 MWO_Resize Class Reference

Resize matrix (allocate new memory if number of elements changes).

`#include <matrix_operator.h>`

Inheritance diagram for MWO_Resize::



Public Member Functions

- void **operator()** (const unsigned int rows, const unsigned int columns) const
  Resize matrix (allocate new memory if number of elements changes).

6.47.1 Detailed Description

Resize matrix (allocate new memory if number of elements changes).

Author:
Lee Netherton

Definition at line 762 of file matrix_operator.h.

6.47.2 Member Function Documentation

6.47.2.1 void MWO_Resize::operator() (const unsigned int *rows*, const unsigned int *columns*) const

Resize matrix (allocate new memory if number of elements changes).

Parameters:
rows Desired number of rows
columns Desired number of columns

Definition at line 508 of file matrix_operator.cpp.

```
{
        // Change values
        setRows(rows);
        setColumns(columns);

        // Delete old memory
        delete[] getDataPointer();

        // Allocate new memory
        setDataPointer(new double[rows*columns]);
}
```

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

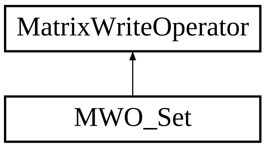
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.48 MWO_Set Class Reference

Set all matrix values.

`#include <matrix_operator.h>`

Inheritance diagram for MWO_Set::



Public Member Functions

- void **operator()** (const double val) const
  Make all the elements in this matrix equal some value.
- void **operator()** () const
  Make all the elements in this matrix equal unity.

6.48.1 Detailed Description

Set all matrix values.

Author:
Lee Netherton

Definition at line 786 of file matrix_operator.h.

6.48.2 Member Function Documentation

6.48.2.1 void MWO_Set::operator() () const

Make all the elements in this matrix equal unity.

Definition at line 539 of file matrix_operator.cpp.

```
{
        unsigned int i;

        for (i = 0; i < getRows()*getColumns(); i++) {
                element(i) = 1;
        }
}
```

6.48.2.2 void MWO_Set::operator() (const double *val*) const

Make all the elements in this matrix equal some value.

Parameters:
> ***val*** Value to make all elements equal

Definition at line 530 of file matrix_operator.cpp.

```
{
        unsigned int i;

        for (i = 0; i < getRows()*getColumns(); i++) {
                element(i) = val;
        }
}
```

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

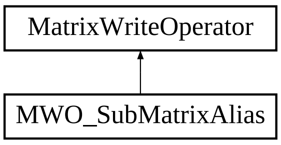
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.49 MWO_SubMatrixAlias Class Reference

Get a SubMatrix.

`#include <matrix_operator.h>`

Inheritance diagram for MWO_SubMatrixAlias::



Public Member Functions

- **SubMatrixAlias operator()** (const unsigned int rowStart, const unsigned int rowEnd, const unsigned int columnStart, const unsigned int columnEnd) const

  Returns a SubMatrix of this matrix give row/column start/end positions.

6.49.1 Detailed Description

Get a SubMatrix.

Author:
Lee Netherton

Definition at line 816 of file matrix_operator.h.

6.49.2 Member Function Documentation

6.49.2.1 SubMatrixAlias MWO_SubMatrixAlias::operator() (const unsigned int *rowStart*, const unsigned int *rowEnd*, const unsigned int *columnStart*, const unsigned int *columnEnd*) const

Returns a SubMatrix of this matrix give row/column start/end positions.

Parameters:

rowStart Starting row (zero-indexed)

rowEnd Finishing row (inclusive, zero-indexed)

columnStart Starting column (zero-indexed)

columnEnd Finishing column (inclusive, zero-indexed)

Definition at line 573 of file matrix_operator.cpp.

```
{
        SubMatrixAlias subMatrixAlias(m_thisMatrix, rowStart, rowEnd, columnStart, columnEnd);

        return subMatrixAlias;
}
```

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

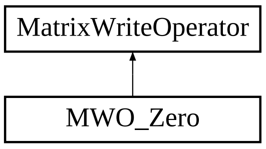
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.50 MWO_Zero Class Reference

Zero matrix.

`#include <matrix_operator.h>`

Inheritance diagram for MWO_Zero::



Public Member Functions

- void **operator()** () const

  Make all the elements in this matrix equal zero.

6.50.1 Detailed Description

Zero matrix.

Author:

Lee Netherton and Peter Mendham

Definition at line 775 of file matrix_operator.h.

6.50.2 Member Function Documentation

6.50.2.1 void MWO_Zero::operator() () const

Make all the elements in this matrix equal zero.

Definition at line 521 of file matrix_operator.cpp.

```
{
        unsigned int i;

        for (i = 0; i < getRows()*getColumns(); i++) {
                element(i) = 0;
        }
}
```

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

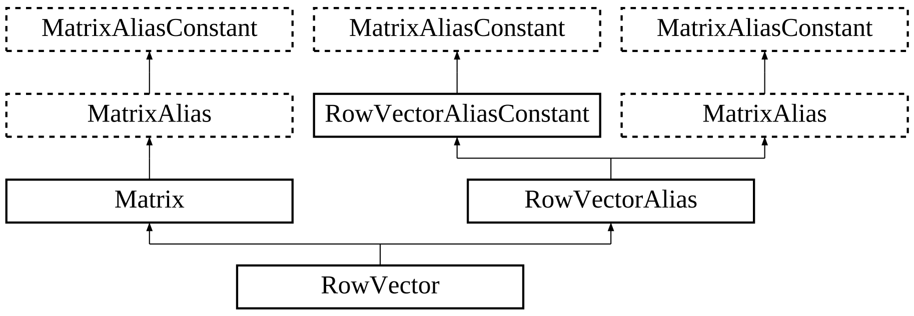
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.51 RowVector Class Reference

The standard RowVector class.

`#include <matrix.h>`

Inheritance diagram for RowVector::



Public Member Functions

- **RowVector** (const unsigned int size)

  Sized constructor.

- **RowVector** (const **MatrixAliasConstant** &copy)

  Base class copy constructor.

- **RowVector** (**RowVector** &copy)

  Copy constructor.

- virtual ~**RowVector** ()

  RowVector Destructor.

- **RowVector** & **operator=** (const **MatrixAliasConstant** &copy)

  Base class assignment operator.

- **RowVector** & **operator=** (const **RowVector** &copy)

  Assignment operator.

Public Attributes

- **RVWO_EqualsElementCopyResize equals**

  Checks to see if the operand is compatable (i.e. a row vector) and then copies data in.

Protected Member Functions

- void **_constructRowVector** (const unsigned int size)

  Sized constructor.

- void **_constructRowVector** (const **MatrixAliasConstant** &copy)

  Copy constructor.

6.51.1 Detailed Description

The standard RowVector class.

Author:

Lee Netherton

The RowVector class provides the user with a pre made row vector. It will allocate its own memory, and is provided with a full complement of matrix and row vector operators.

Definition at line 1479 of file matrix.h.

6.51.2 Constructor & Destructor Documentation

6.51.2.1 RowVector::RowVector (const unsigned int *size*) `[inline]`

Sized constructor.

Creates a new row vector of a given size.

Parameters:

size Number of columns matrix has

Definition at line 1497 of file matrix.h.

```
1497 {_constructRowVector(size);}
```

6.51.2.2 RowVector::RowVector (const MatrixAliasConstant & *copy*) `[inline]`

Base class copy constructor.

Makes a copy of any another matrix.

Parameters:

copy Reaference to matrix to copy

Definition at line 1503 of file matrix.h.

```
1503 {_constructRowVector(copy);}
```

6.51.2.3 RowVector::RowVector (RowVector & *copy*) `[inline]`

Copy constructor.

Makes a copy of another matrix.

Parameters:

copy Reaference to matrix to copy

Definition at line 1509 of file matrix.h.

```
1509 {_constructRowVector(copy);}
```

6.51.2.4 RowVector::∼RowVector () `[virtual]`

RowVector Destructor.

Virtual - the lowest derived class will always need to be called, as they all allocate memory in ther own ways. Definition at line 871 of file matrix.cpp.

```
872 {
873         #ifdef DEBUG_DESTRUCTOR
874                 printf("Destructor: ~RowVector()\n");
875         #endif
876 }
```

6.51.3 Member Function Documentation

6.51.3.1 void RowVector::_constructRowVector (const MatrixAliasConstant & *copy*) `[protected]`

Copy constructor.

Allocates some memory, and calls the **Matrix**(p. 44) constructor function and the **RowVector-Alias**(p. 152) blank constructor function. Finally, copys data in from copied matrix

Parameters:

copy **Matrix**(p. 44) to copy

Definition at line 841 of file matrix.cpp.

```
842 {
843         // Construct main base class
844         _constructMatrix(1,copy.m_matrixContainer->getColumns());
845
846         // Construct blank base class
847         _constructRowVectorAlias();
848
849         // Construct operators
850         _constructRowVectorOperators();
851
852         // Copy the information to this vector
853         equals(copy);
854
855         #ifdef DEBUG_CONSTRUCTOR
856                 printf("Constructed: RowVector::Copy Constructor\n");
857         #endif
858 }
```

6.51.3.2 void RowVector::_constructRowVector (const unsigned int *size*) `[protected]`

Sized constructor.

Allocates some memory, and calls the **Matrix**(p. 44) constructor function and the **RowVector-Alias**(p. 152) blank constructor function.

Parameters:

size Number of columns matrix has

Definition at line 822 of file matrix.cpp.

```
{
        // Construct main base class
        _constructMatrix(1,size);

        // Construct blank base class
        _constructRowVectorAlias();

        // Construct operators
        _constructRowVectorOperators();

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: RowVector::Sized Constructor\n");
        #endif
}
```

6.51.3.3 RowVector& RowVector::operator= (const RowVector & *copy*) `[inline]`

Assignment operator.

Definition at line 1556 of file matrix.h.

```
1556 {return operator=((MatrixAliasConstant&)copy);}
```

6.51.3.4 RowVector& RowVector::operator= (const MatrixAliasConstant & *copy*) `[inline]`

Base class assignment operator.

Reimplemented from **Matrix** (p. 48).

Definition at line 1553 of file matrix.h.

```
1553 {equals(copy);return *this;}
```

6.51.4 Member Data Documentation

6.51.4.1 RVWO_EqualsElementCopyResize RowVector::equals

Checks to see if the operand is compatable (i.e. a row vector) and then copies data in.

Reimplemented from **Matrix** (p. 48).

Definition at line 1487 of file matrix.h.

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

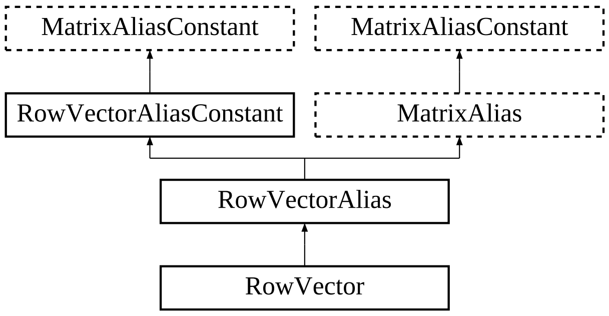
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix.cpp**

6.52 RowVectorAlias Class Reference

A RowVectorAlias class.

`#include <matrix.h>`

Inheritance diagram for RowVectorAlias::



Public Member Functions

- **RowVectorAlias** ()
  Default constructor.
- **RowVectorAlias** (const double ∗data, const unsigned int size)
  Pointer constructor.
- **RowVectorAlias** (const **MatrixAliasConstant** ∗alias)
  Alias constructor.
- **RowVectorAlias** (const **MatrixAliasConstant** &copy)
  Base class copy constructor.
- **RowVectorAlias** (const **RowVectorAlias** &copy)
  Copy constructor.
- virtual ∼**RowVectorAlias** ()
  RowVectorAlias Destructor.
- **RowVectorAlias** & **operator=** (const **MatrixAliasConstant** &copy)
  Base class assignment operator.
- **RowVectorAlias** & **operator=** (const **RowVectorAlias** &copy)
  Assignment operator.

Protected Member Functions

- void **_constructRowVectorAlias** (const double ∗data, const unsigned int size)
  Pointer constructor.

- void **_constructRowVectorAlias** (const **MatrixAliasConstant** &copy)

  Copy constructor.

- void **_constructRowVectorAlias** ()

  Blank constructor.

6.52.1 Detailed Description

A RowVectorAlias class.

Author:
Lee Netherton

The RowVectorAlias class provides all the functionality from the **MatrixAlias**(p. 49) class, but add specific functions intended for row vectors. It also has specific row vector write functions.

Definition at line 1379 of file matrix.h.

6.52.2 Constructor & Destructor Documentation

6.52.2.1 RowVectorAlias::RowVectorAlias () `[inline]`

Default constructor.

Creates a RowVectorAlias shell. The **MatrixContainer**(p. 70), **MatrixReadAccess**(p. 73) and **MatrixWriteAccess**(p. 81) handles can be set later using the constructor function **_construct-RowVectorAlias()**(p. 155) Definition at line 1394 of file matrix.h.

```
1394 {}
```

6.52.2.2 RowVectorAlias::RowVectorAlias (const double * *data*, const unsigned int *size*) `[inline]`

Pointer constructor.

To create a row vector that will access a pre-available data array.

Parameters:
data Pointer to data array. This will be the data storage for the matrix.

size Number of columns the vector has.

Definition at line 1401 of file matrix.h.

```
1401 {_constructRowVectorAlias(data,size);}
```

6.52.2.3 RowVectorAlias::RowVectorAlias (const MatrixAliasConstant ∗ *alias*) `[inline]`

Alias constructor.

To create a row vector that will alias another matrix.

Parameters:

alias Pointer to a matrix which this vector will alias.

Definition at line 1407 of file matrix.h.

```
1407 {_constructRowVectorAlias(*alias);}
```

6.52.2.4 RowVectorAlias::RowVectorAlias (const MatrixAliasConstant & *copy*) `[inline]`

Base class copy constructor.

Used when creating a RowVectorAlias matrix from another matrix.

Parameters:

copy Reference to another matrix.

Definition at line 1413 of file matrix.h.

```
1413 {_constructRowVectorAlias(copy);}
```

6.52.2.5 RowVectorAlias::RowVectorAlias (const RowVectorAlias & *copy*) `[inline]`

Copy constructor.

Used when creating a RowVectorAlias matrix from another. Calls base class copy constructor

Parameters:

copy Reference to another row vector.

Definition at line 1420 of file matrix.h.

```
1420 {_constructRowVectorAlias(copy);}
```

6.52.2.6 RowVectorAlias::∼RowVectorAlias () `[virtual]`

RowVectorAlias Destructor.

Virtual - the lowest derived class will always need to be called, as they all allocate memory in ther own ways. Definition at line 801 of file matrix.cpp.

```
802 {
803         #ifdef DEBUG_DESTRUCTOR
804                 printf("Destructor: ~RowVectorAlias()\n");
805         #endif
806 }
```

6.52.3 Member Function Documentation

6.52.3.1 void RowVectorAlias::_constructRowVectorAlias () [protected]

Blank constructor.

Just constructs RowVectorAlias and **RowVectorAliasConstant**(p. 157) operators, and goes no further. Definition at line 778 of file matrix.cpp.

```
{
        // Construct blank base classes and nothing else
        _constructRowVectorAliasConstant();

        // Construct operators
        _constructRowVectorAliasOperators();

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: RowVectorAlias::Blank Constructor\n");
        #endif
}
```

6.52.3.2 void RowVectorAlias::_constructRowVectorAlias (const MatrixAliasConstant & *copy*) [protected]

Copy constructor.

Copies the pointers to the **MatrixContainer**(p. 70), **MatrixReadOperator**(p. 77) and **Matrix-WriteOperator**(p. 85) members Definition at line 759 of file matrix.cpp.

```
{
        // Construct main base class
        _constructMatrixAlias(copy.m_matrixContainer->getDataPointer(),1,copy.m_matrixContainer->getColumns());

        // Construct blank base class
        _constructRowVectorAliasConstant();

        // Construct operators
        _constructRowVectorAliasOperators();

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: RowVectorAlias::Copy Constructor\n");
        #endif
}
```

6.52.3.3 void RowVectorAlias::_constructRowVectorAlias (const double ∗ *data*, const unsigned int *size*) [protected]

Pointer constructor.

Sets the pointers to the **MatrixContainer**(p. 70), **MatrixReadOperator**(p. 77) and **Matrix-WriteOperator**(p. 85) members Definition at line 740 of file matrix.cpp.

```
{
        // Construct main base class
        _constructMatrixAlias(data,1,size);

        // Construct blank base class
        _constructRowVectorAliasConstant();

        // Construct operators
        _constructRowVectorAliasOperators();

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: RowVectorAlias::Pointer Constructor\n");
        #endif
}
```

6.52.3.4 RowVectorAlias& RowVectorAlias::operator= (const RowVectorAlias & *copy*) `[inline]`

Assignment operator.

Definition at line 1469 of file matrix.h.

```
1469 {return operator=((MatrixAliasConstant&)copy);}
```

6.52.3.5 RowVectorAlias& RowVectorAlias::operator= (const MatrixAliasConstant & *copy*) `[inline]`

Base class assignment operator.

Reimplemented from **MatrixAlias** (p. 55).

Reimplemented in **RowVector** (p. 150).

Definition at line 1466 of file matrix.h.

```
1466 {equals(copy);return *this;}
```

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

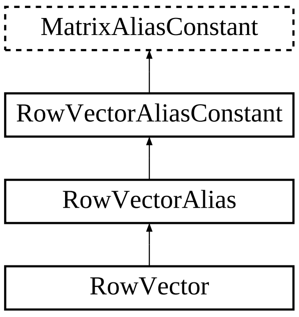
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix.cpp**

6.53 RowVectorAliasConstant Class Reference

A read-only **RowVectorAlias**(p. 152) class.

`#include <matrix.h>`

Inheritance diagram for RowVectorAliasConstant::



Public Member Functions

- **RowVectorAliasConstant** ()

  Default constructor.

- **RowVectorAliasConstant** (const double ∗data, const unsigned int size)

  Pointer constructor.

- **RowVectorAliasConstant** (const **MatrixAliasConstant** ∗alias)

  Alias constructor.

- **RowVectorAliasConstant** (const **MatrixAliasConstant** &copy)

  Base class copy constructor.

- **RowVectorAliasConstant** (const **RowVectorAliasConstant** &copy)

  Copy constructor.

- virtual ∼**RowVectorAliasConstant** ()

  RowVectorAliasConstant Destructor.

- **RowVectorAliasConstant** & **operator=** (const **MatrixAliasConstant** &copy)

  Base class assignment operator.

- **RowVectorAliasConstant** & **operator=** (const **RowVectorAliasConstant** &copy)

  Assignment operator.

Public Attributes

- **RVRO_CrossProduct cross**

  Returns the cross product of this vector.

- **RVRO_DotProduct dot**
  Returns the dot product of this vector.
- **RVRO_Modulus modulus**
  Returns the modulus of this vector.

Protected Member Functions

- void **_constructRowVectorAliasConstant** (const double *data, const unsigned int size)
  Pointer constructor.
- void **_constructRowVectorAliasConstant** (const **MatrixAliasConstant** &copy)
  Copy constructor.
- void **_constructRowVectorAliasConstant** ()
  Blank constructor.

6.53.1 Detailed Description

A read-only **RowVectorAlias**(p. 152) class.

Author:
Lee Netherton

The RowVectorAliasConstant class provides all the functionality from the **MatrixAliasConstant**(p. 57) class, but add specific functions intended for row vectors.

Definition at line 1268 of file matrix.h.

6.53.2 Constructor & Destructor Documentation

6.53.2.1 RowVectorAliasConstant::RowVectorAliasConstant () `[inline]`

Default constructor.

Creates a RowVectorAliasConstant shell. The **MatrixContainer**(p. 70) and **MatrixReadAccess**(p. 73) handles can be set later using the constructor function **_constructRowVectorAliasConstant()**(p. 160) Definition at line 1295 of file matrix.h.

```
1295 {}
```

6.53.2.2 RowVectorAliasConstant::RowVectorAliasConstant (const double * *data*, const unsigned int *size*) `[inline]`

Pointer constructor.

To create a read-only row vector that will access a pre-available data array.

Parameters:

data Pointer to data array. This will be the data storage for the matrix.

size Number of columns the matrix has.

Definition at line 1302 of file matrix.h.

```
1302 {_constructRowVectorAliasConstant(data,size);}
```

6.53.2.3 RowVectorAliasConstant::RowVectorAliasConstant (const MatrixAliasConstant ∗ *alias*) `[inline]`

Alias constructor.

To create a read-only row vector that will alias another matrix.

Parameters:

alias Pointer to a matrix which this matris will alias.

Definition at line 1308 of file matrix.h.

```
1308 {_constructRowVectorAliasConstant(*alias);}
```

6.53.2.4 RowVectorAliasConstant::RowVectorAliasConstant (const MatrixAliasConstant & *copy*) `[inline]`

Base class copy constructor.

Used when creating a RowVectorAliasConstant matrix from another.

Parameters:

copy Reference to another matrix.

Definition at line 1314 of file matrix.h.

```
1314 {_constructRowVectorAliasConstant(copy);}
```

6.53.2.5 RowVectorAliasConstant::RowVectorAliasConstant (const RowVectorAliasConstant & *copy*) `[inline]`

Copy constructor.

Used when creating a **MatrixAliasConstant**(p. 57) matrix from another. Calls base class copy constructor

Parameters:

copy Reference to another matrix.

Definition at line 1321 of file matrix.h.

```
1321 {_constructRowVectorAliasConstant(copy);}
```

6.53.2.6 RowVectorAliasConstant::∼RowVectorAliasConstant () [virtual]

RowVectorAliasConstant Destructor.

Virtual - the lowest derived class will always need to be called, as they all allocate memory in ther own ways. Definition at line 716 of file matrix.cpp.

```
{
        #ifdef DEBUG_DESTRUCTOR
                printf("Destructor: ~RowVectorAliasConstant()\n");
        #endif
}
```

6.53.3 Member Function Documentation

6.53.3.1 void RowVectorAliasConstant::_constructRowVectorAliasConstant () [protected]

Blank constructor.

Just constructs RowVectorAliasConstant operators, and goes no further. Definition at line 692 of file matrix.cpp.

```
{
        // Construct operators
        _constructRowVectorAliasConstantOperators();

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: RowVectorAliasConstant::Blank Constructor\n");
        #endif
}
```

6.53.3.2 void RowVectorAliasConstant::_constructRowVectorAliasConstant (const MatrixAliasConstant & *copy*) [protected]

Copy constructor.

Copies the pointers to the **MatrixContainer**(p. 70) and **MatrixReadOperator**(p. 77) members Definition at line 676 of file matrix.cpp.

```
{
        // Construct main base class
        _constructMatrixAliasConstant(copy.m_matrixContainer->getDataPointer(),1,copy.m_matrixContainer->getColumns

        // Construct operators
        _constructRowVectorAliasConstantOperators();

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: RowVectorAliasConstant::Copy Constructor\n");
        #endif
}
```

6.53.3.3 void RowVectorAliasConstant::_constructRowVectorAliasConstant (const double ∗ *data*, const unsigned int *size*) [protected]

Pointer constructor.

Sets the pointers to the **MatrixContainer**(p. 70) and **MatrixReadOperator**(p. 77) members Definition at line 660 of file matrix.cpp.

```
{
        // Construct main base class
        _constructMatrixAliasConstant(data,1,size);

        // Construct operators
        _constructRowVectorAliasConstantOperators();

        #ifdef DEBUG_CONSTRUCTOR
              printf("Constructed: RowVectorAliasConstant::Pointer Constructor\n");
        #endif
}
```

6.53.3.4 RowVectorAliasConstant& RowVectorAliasConstant::operator= (const RowVectorAliasConstant & *copy*) `[inline]`

Assignment operator.

Definition at line 1368 of file matrix.h.

```
{return operator=((MatrixAliasConstant&)copy);}
```

6.53.3.5 RowVectorAliasConstant& RowVectorAliasConstant::operator= (const MatrixAliasConstant & *copy*) `[inline]`

Base class assignment operator.

Reimplemented from **MatrixAliasConstant** (p. 66).

Reimplemented in **RowVectorAlias** (p. 156), and **RowVector** (p. 150).

Definition at line 1365 of file matrix.h.

```
{m_matrixReadAccess->error("Tried to assign to a constant vector\n");return *this;}
```

6.53.4 Member Data Documentation

6.53.4.1 RVRO_CrossProduct RowVectorAliasConstant::cross

Returns the cross product of this vector.

Definition at line 1279 of file matrix.h.

6.53.4.2 RVRO_DotProduct RowVectorAliasConstant::dot

Returns the dot product of this vector.

Definition at line 1282 of file matrix.h.

6.53.4.3 RVRO_Modulus RowVectorAliasConstant::modulus

Returns the modulus of this vector.

Definition at line 1285 of file matrix.h.

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

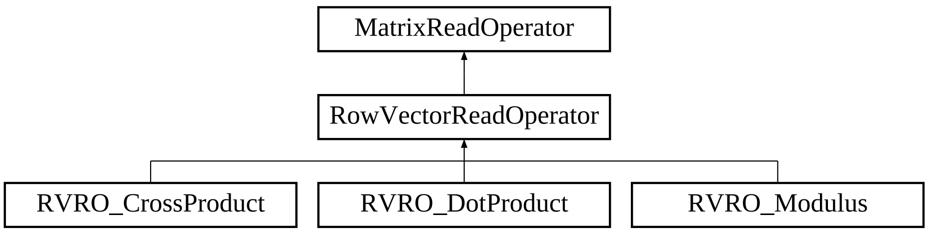
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix.cpp**

6.54 RowVectorReadOperator Class Reference

Base class for RowVectorReadOperators.

`#include <matrix_operator.h>`

Inheritance diagram for RowVectorReadOperator::



Public Member Functions

- **RowVectorReadOperator** ()

  Default Constructor.

- **RowVectorReadOperator** (**RowVectorAliasConstant** ∗rowVectorAliasConstant)

  Full Constructor.

- void **_constructRowVectorReadOperator** (**RowVectorAliasConstant** ∗rowVector-AliasConstant)

  Manual Constructor.

Protected Attributes

- **RowVectorAliasConstant** ∗ **m_thisMatrix**

  Pointer to owner matrix.

6.54.1 Detailed Description

Base class for RowVectorReadOperators.

Author:
Lee Netherton

Most importantly provides m_thisMatrix with the right kind of pointer.

Definition at line 247 of file matrix_operator.h.

6.54.2 Constructor & Destructor Documentation

6.54.2.1 RowVectorReadOperator::RowVectorReadOperator () `[inline]`

Default Constructor.

Creates an empty operator class which can then be more full constructed using **_constructRowVectorReadOperator()**(p. 164) Definition at line 259 of file matrix_operator.h.

```
259 {}
```

6.54.2.2 RowVectorReadOperator::RowVectorReadOperator (RowVectorAliasConstant * *rowVectorAliasConstant*) [inline]

Full Constructor.

Creates an operator class which takes and stores a pointer to an owner matrix

Parameters:
rowVectorAliasConstant Pointer to owner matrix

Definition at line 265 of file matrix_operator.h.

```
265                                                                                   :
266                         MatrixReadOperator((MatrixAliasConstant *)rowVectorAliasConstant),
267                         m_thisMatrix(rowVectorAliasConstant)
268                         {}
```

6.54.3 Member Function Documentation

6.54.3.1 void RowVectorReadOperator::_constructRowVectorReadOperator (RowVectorAliasConstant * *rowVectorAliasConstant*) [inline]

Manual Constructor.

Constructs the class manually by setting the owner pointer

Parameters:
rowVectorAliasConstant Pointer to owner matrix

Definition at line 274 of file matrix_operator.h.

```
275                 {
276                         _constructMatrixReadOperator((MatrixAliasConstant *)rowVectorAliasConstant);
277 
278                         m_thisMatrix = rowVectorAliasConstant;
279                 }
```

6.54.4 Member Data Documentation

6.54.4.1 RowVectorAliasConstant* RowVectorReadOperator::m_thisMatrix [protected]

Pointer to owner matrix.

Reimplemented from **MatrixReadOperator** (p. 80).

Definition at line 252 of file matrix_operator.h.

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

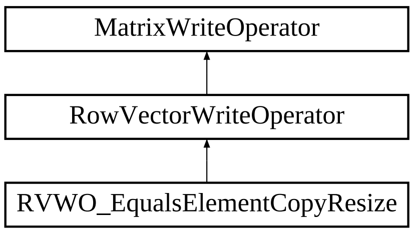
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**

6.55 RowVectorWriteOperator Class Reference

Base class for RowVectorWriteOperators.

`#include <matrix_operator.h>`

Inheritance diagram for RowVectorWriteOperator::



Public Member Functions

- **RowVectorWriteOperator** ()
  Default Constructor.
- **RowVectorWriteOperator** (**RowVectorAlias** ∗rowVectorAlias)
  Full Constructor.
- void **_constructRowVectorWriteOperator** (**RowVectorAlias** ∗rowVectorAlias)
  Manual Constructor.

Protected Attributes

- **RowVectorAlias** ∗ **m_thisMatrix**
  Pointer to owner matrix.

6.55.1 Detailed Description

Base class for RowVectorWriteOperators.

Author:
Lee Netherton

Most importantly provides m_thisMatrix with the right kind of pointer.

Definition at line 288 of file matrix_operator.h.

6.55.2 Constructor & Destructor Documentation

6.55.2.1 RowVectorWriteOperator::RowVectorWriteOperator () `[inline]`

Default Constructor.

Creates an empty operator class which can then be more full constructed using **_constructRowVectorWriteOperator()**(p. 166) Definition at line 300 of file matrix_operator.h.

```
{}
```

6.55.2.2 RowVectorWriteOperator::RowVectorWriteOperator (RowVectorAlias * *rowVectorAlias*) [inline]

Full Constructor.

Creates an operator class which takes and stores a pointer to an owner matrix

Parameters:
rowVectorAlias Pointer to owner matrix

Definition at line 306 of file matrix_operator.h.

```
                                                          :
                        MatrixWriteOperator((MatrixAlias *)rowVectorAlias),
                        m_thisMatrix(rowVectorAlias)
                        {}
```

6.55.3 Member Function Documentation

6.55.3.1 void RowVectorWriteOperator::_constructRowVectorWriteOperator (RowVectorAlias * *rowVectorAlias*) [inline]

Manual Constructor.

Constructs the class manually by setting the owner pointer

Parameters:
rowVectorAlias Pointer to owner matrix

Definition at line 315 of file matrix_operator.h.

```
                {
                        _constructMatrixWriteOperator((MatrixAlias *)rowVectorAlias);

                        m_thisMatrix = rowVectorAlias;
                }
```

6.55.4 Member Data Documentation

6.55.4.1 RowVectorAlias* RowVectorWriteOperator::m_thisMatrix [protected]

Pointer to owner matrix.

Reimplemented from **MatrixWriteOperator** (p. 89).

Definition at line 293 of file matrix_operator.h.

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

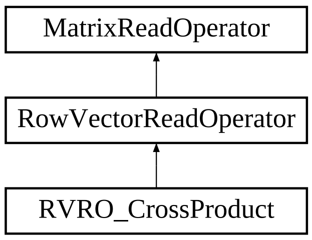
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**

6.56 RVRO_CrossProduct Class Reference

Returns the cross product of the vector and its operand.

`#include <matrix_operator.h>`

Inheritance diagram for RVRO_CrossProduct::



Public Member Functions

- **RowVector operator()** (const **RowVectorAliasConstant** &operand) const
  Returns the cross product of the vector and its operand.

6.56.1 Detailed Description

Returns the cross product of the vector and its operand.

Author:
Lee Netherton and Peter Mendham

Definition at line 980 of file matrix_operator.h.

6.56.2 Member Function Documentation

6.56.2.1 RowVector RVRO_CrossProduct::operator() (const RowVectorAliasConstant & *operand*) const

Returns the cross product of the vector and its operand.

Parameters:
operand Operand to cross with.

Definition at line 1063 of file matrix_operator.cpp.

```
                                                                            {

        RowVector result(getColumns());
        SquareMatrix temp(getColumns());
        unsigned int i;

        if (operand.getColumns() != getColumns()) {

                error("RowVectorAlias::cross : Dimensions are not consistent\n");

        } else if (operand.getRows() != 3) {

                error("RowVectorAlias::cross : Only vectors of length 3 are valid at this time\n");

        } else {

                for (i = 0; i < getColumns(); i++) {
                        temp.element(1,i) = element(i);
                        temp.element(2,i) = operand.element(i);
                }

                for (i = 0; i < getColumns(); i++) {
                        result.element(i) = temp.cofactor(0,i);
                }
        }

        return result;
}
```

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

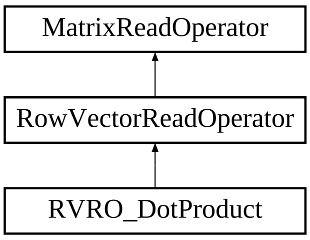
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.57 RVRO_DotProduct Class Reference

Returns the dot product of the vector and its operand.

`#include <matrix_operator.h>`

Inheritance diagram for RVRO_DotProduct::



Public Member Functions

- double **operator()** (const **RowVectorAliasConstant** &operand) const
  Returns the dot product of the vector and its operand.

6.57.1 Detailed Description

Returns the dot product of the vector and its operand.

Author:
Lee Netherton and Peter Mendham

Definition at line 991 of file matrix_operator.h.

6.57.2 Member Function Documentation

6.57.2.1 double RVRO_DotProduct::operator() (const RowVectorAliasConstant & *operand*) const

Returns the dot product of the vector and its operand.

Parameters:
operand Operand to dot with.

Definition at line 1096 of file matrix_operator.cpp.

```
                                                                             {

        double result = 0;
        unsigned int i;

        if (operand.getColumns() != getColumns()) {

                error("RowVectorAlias::dot : Dimensions are not consistent\n");

        } else {

                for (i = 0; i < getColumns(); i++) {
                        result += element(i) * operand.element(i);
                }
        }

        return result;
}
```

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

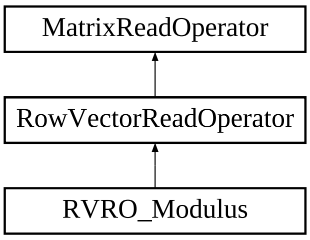
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.58 RVRO_Modulus Class Reference

Returns modulus of this vector.

`#include <matrix_operator.h>`

Inheritance diagram for RVRO_Modulus::



Public Member Functions

- double **operator()** () const

  Returns modulus of this vector.

6.58.1 Detailed Description

Returns modulus of this vector.

Author:
Lee Netherton and Peter Mendham

Definition at line 1002 of file matrix_operator.h.

6.58.2 Member Function Documentation

6.58.2.1 double RVRO_Modulus::operator() () const

Returns modulus of this vector.

Definition at line 1119 of file matrix_operator.cpp.

```
                                      {

        unsigned int i;
        double mod = 0;

        for (i = 0; i < getColumns(); i++) {
                mod += pow(element(i), 2);
        }

        return sqrt(mod);
}
```

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

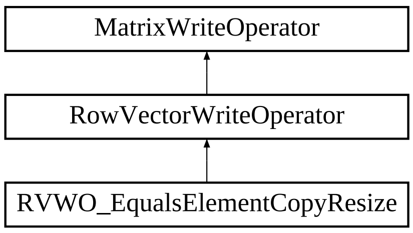
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.59 RVWO_EqualsElementCopyResize Class Reference

Copy elements (Resize if necessesary).

`#include <matrix_operator.h>`

Inheritance diagram for RVWO_EqualsElementCopyResize::



Public Member Functions

- void **operator()** (const **MatrixAliasConstant** &copy) const
  Checks to see if copy is a row vector, if so copies element in.

6.59.1 Detailed Description

Copy elements (Resize if necessesary).

Author:
Lee Netherton

Definition at line 1017 of file matrix_operator.h.

6.59.2 Member Function Documentation

6.59.2.1 void RVWO_EqualsElementCopyResize::operator() (const MatrixAliasConstant & *copy*) const

Checks to see if copy is a row vector, if so copies element in.

Parameters:
copy **Matrix**(p. 44) to copy (must be row vector)

Definition at line 1138 of file matrix_operator.cpp.

```
{
                unsigned int i;

                if(copy.isRowVector() == 0)
                {
                        error("RVWO_EqualsElementCopyResize: Matrix to copy is not a row vector\n");
                        return;
                }


                if(copy.getColumns() != getColumns()) {
                        // Resize!!

                        // Change values
                        setColumns(copy.getColumns());

                        // Delete old memory
                        delete[] getDataPointer();

                        // Allocate new memory
                        setDataPointer(new double[getColumns()]);
                }


                for (i = 0; i < getColumns(); i++) {
                        element(0, i) = copy.element(0, i);
                }

}
```

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

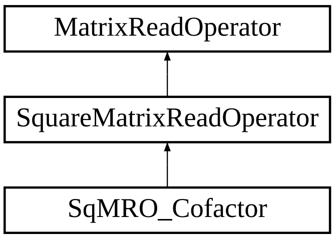
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.60 SqMRO_Cofactor Class Reference

Calculate the cofactor of an element.

`#include <matrix_operator.h>`

Inheritance diagram for SqMRO_Cofactor::



Public Member Functions

- double **operator()** (const unsigned int row, const unsigned int column) const
  Calculates the cofactor for an element.

6.60.1 Detailed Description

Calculate the cofactor of an element.

Author:
Lee Netherton and Peter Mendham

Definition at line 865 of file matrix_operator.h.

6.60.2 Member Function Documentation

6.60.2.1 double SqMRO_Cofactor::operator() (const unsigned int *row*, const unsigned int *column*) const

Calculates the cofactor for an element.

Parameters:
row Row of element (zero-indexed)
column Column of element (zero-indexed)

Definition at line 662 of file matrix_operator.cpp.

```
                                                                                   {

        SquareMatrix newSquareMatrix(getRows() - 1);
        double sign;

        if (row >= getRows() || column >= getColumns()) {
                error("SquareMatrixAlias::cofactor : Subscript out of range\n");
                return 0;
        }



        sign = (row + column) % 2 ? -1 : 1;

        // Top left quadrant
        if (row > 0 && column > 0) {
                newSquareMatrix.subMatrix(0, 0, row-1, column-1) = m_thisMatrix->subMatrix(0, 0, row-1, column-1);
        }
        // Top right quadrant
        if (row > 0 && column < (getColumns() - 1)) {
                newSquareMatrix.subMatrix(0, column, row-1, getColumns()-2) = m_thisMatrix->subMatrix(0, column+1, r
        }
        // Bottom left quadrant
        if (row < (getRows() - 1) && column > 0) {
                newSquareMatrix.subMatrix(row, 0, getRows()-2, column-1) = m_thisMatrix->subMatrix(row+1, 0, getRows
        }
        // Bottom right quadrant
        if (row < (getRows() - 1) && column < (getColumns() - 1)) {
                newSquareMatrix.subMatrix(row, column, getRows()-2, getColumns()-2) = m_thisMatrix->subMatrix(row+1
        }

        return sign * newSquareMatrix.determinant();
}
```

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

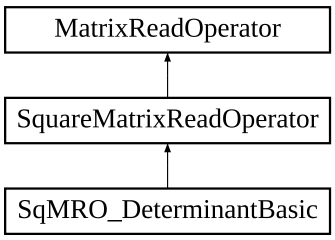
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.61 SqMRO_DeterminantBasic Class Reference

Assess matrix compataibility (is matrix square?). Calculate determinant of matrix.

`#include <matrix_operator.h>`

Inheritance diagram for SqMRO_DeterminantBasic::



Public Member Functions

- double **operator()** () const

  Calculates the determinant of this matrix. Uses basic method.

6.61.1 Detailed Description

Assess matrix compataibility (is matrix square?). Calculate determinant of matrix.

Author:

Lee Netherton and Peter Mendham

Definition at line 845 of file matrix_operator.h.

6.61.2 Member Function Documentation

6.61.2.1 double SqMRO_DeterminantBasic::operator() () const

Calculates the determinant of this matrix. Uses basic method.

Definition at line 601 of file matrix_operator.cpp.

```
                                           {

        unsigned int i;
        double det;

        if (getRows() == 1) {

                det = element(0);

        } else if (getRows() == 2) {

                det = element(0,0)*element(1,1) - element(0,1)*element(1,0);

        } else {

                det = 0;

                for (i = 0; i < getColumns(); i++) {
                        det     += element(i) * m_thisMatrix->cofactor(0, i);
                }
        }

        return det;
}
```

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

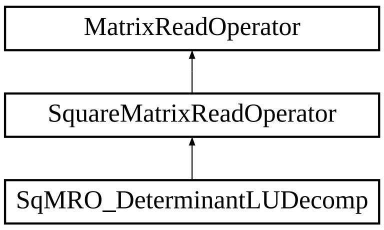
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.62 SqMRO_DeterminantLUDecomp Class Reference

Calculate determinant of matrix.

`#include <matrix_operator.h>`

Inheritance diagram for SqMRO_DeterminantLUDecomp::



Public Member Functions

- double **operator()** () const

  Calculates the determinant of this matrix. Uses LU decomposition method.

6.62.1 Detailed Description

Calculate determinant of matrix.

Author:

Lee Netherton and Peter Mendham

Definition at line 855 of file matrix_operator.h.

6.62.2 Member Function Documentation

6.62.2.1 double SqMRO_DeterminantLUDecomp::operator() () const

Calculates the determinant of this matrix. Uses LU decomposition method.

Definition at line 631 of file matrix_operator.cpp.

```
631                                                     {
632 
633         double det;
634 
635         if (getRows() == 1) {
636 
637                 det = element(0);
638 
639         } else if (getRows() == 2) {
640 
641                 det = element(0,0)*element(1,1) - element(0,1)*element(1,0);
642 
643         } else {
644 
645                 SquareMatrix temp(*m_thisMatrix);
```

```
                RowVector indices(getRows());
                unsigned int j;

                temp.luDecomposition(indices, &det);
                for (j = 0; j < getRows(); j++) {
                        det *= temp(j, j);
                }
        }

        return det;
}
```

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

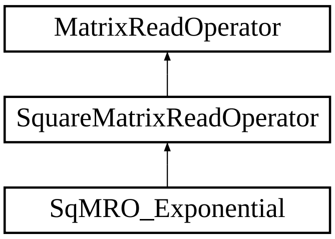
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.63 SqMRO_Exponential Class Reference

Matrix(p. 44) exponential.

`#include <matrix_operator.h>`

Inheritance diagram for SqMRO_Exponential::



Public Member Functions

- **SquareMatrix operator()** (double time) const
  Return the matrix exponential using irreducible Pade approximation.

6.63.1 Detailed Description

Matrix(p. 44) exponential.

Author:
Lee Netherton and Peter Mendham

Definition at line 887 of file matrix_operator.h.

6.63.2 Member Function Documentation

6.63.2.1 SquareMatrix SqMRO_Exponential::operator() (double *time*) const

Return the matrix exponential using irreducible Pade approximation.

Definition at line 751 of file matrix_operator.cpp.

```
751                                                     {
752 
753         SquareMatrix localCopy(getRows()), squaredCopy(getRows());
754         SquareMatrix I(getRows()), Q(getRows()), P(getRows()), result(getRows());
755         unsigned int nCoefs = 6;
756         RowVector padeCoef(nCoefs + 1);
757         unsigned int i, odd;
758         double k, scaleFactor;
759 
760         // Multiply intime factor
761         //localCopy = (*this) * time;
762         localCopy.multiply(time);
763 
764         // Setup Pade coefficients
765         padeCoef.element(0) = 1;
```

```
        for (i = 1; i <= nCoefs; i++) {
                k = i;
                padeCoef.element(i) = padeCoef.element(i-1) * ((nCoefs+1-k)/(k*(2*nCoefs+1-k)));
        }

        // Scale the matrix if necessary
        scaleFactor = m_thisMatrix->infinityNorm();
        if (scaleFactor > 0.5) {

                scaleFactor = log(scaleFactor)/log((double)2);
                scaleFactor = (scaleFactor > 0) ? floor(scaleFactor) : ceil(scaleFactor);
                scaleFactor += 2;
                scaleFactor = (scaleFactor > 0) ? scaleFactor : 0;

                localCopy = pow((double)2,-scaleFactor) * localCopy;
        }

        // Horner evaluation of the irreducible fraction
        I.identity();
        squaredCopy = localCopy * localCopy;
        Q = padeCoef.element(nCoefs) * I;
        P = padeCoef.element(nCoefs-1) * I;
        odd = 1;

        for (i = (nCoefs - 1); i > 0; i--) {

                if (odd == 1) {
                        Q = Q*squaredCopy + padeCoef.element(i-1)*I;
                } else {
                        P = P*squaredCopy + padeCoef.element(i-1)*I;
                }

                odd = 1 - odd;
        }

        if (odd == 1) {
                Q = Q*localCopy;
                Q = Q - P;
                result = -1*(I + 2 * (Q.inverse() * P));
        } else {
                P = P*localCopy;
                Q = Q - P;
                result = I + 2 * (Q.inverse() * P);
        }

        // Squaring
        for (i = 0; i < (unsigned int)scaleFactor; i++) {

                result = result * result;
        }

        return result;
}
```

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

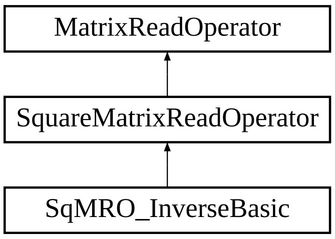
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.64 SqMRO_InverseBasic Class Reference

Matrix(p. 44) inverse.

`#include <matrix_operator.h>`

Inheritance diagram for SqMRO_InverseBasic::



Public Member Functions

- **SquareMatrix operator()** () const

  Returns the inverse of this matrix. Uses basic method.

6.64.1 Detailed Description

Matrix(p. 44) inverse.

Author:

Lee Netherton and Peter Mendham

Definition at line 877 of file matrix_operator.h.

6.64.2 Member Function Documentation

6.64.2.1 SquareMatrix SqMRO_InverseBasic::operator() () const

Returns the inverse of this matrix. Uses basic method.

Definition at line 700 of file matrix_operator.cpp.

```
                                             {

        SquareMatrix inv(getRows());
        unsigned int i, j;
        double det;

        det = m_thisMatrix->determinant();

        if (det == 0) {
                error("SquareMatrixAlias::inverse : MatrixAlias is rank defficient\n");
                return inv; // return blank matrix
        }

        for (i = 0; i < getRows(); i++) {
                for(j = 0; j < getColumns(); j++) {
                        inv.element(j, i) = m_thisMatrix->cofactor(i, j) / det;
                }
        }

        return inv;
}
```

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

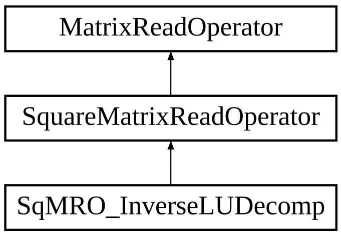
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.65 SqMRO_InverseLUDecomp Class Reference

Matrix(p. 44) inverse.

`#include <matrix_operator.h>`

Inheritance diagram for SqMRO_InverseLUDecomp::



Public Member Functions

- **SquareMatrix operator()** () const

  Returns the inverse of this matrix. Uses LU decomposition method.

6.65.1 Detailed Description

Matrix(p. 44) inverse.

Author:

Lee Netherton and Peter Mendham

Definition at line 898 of file matrix_operator.h.

6.65.2 Member Function Documentation

6.65.2.1 SquareMatrix SqMRO_InverseLUDecomp::operator() () const

Returns the inverse of this matrix. Uses LU decomposition method.

Definition at line 726 of file matrix_operator.cpp.

```
                                                {

        SquareMatrix inv(*m_thisMatrix);
        RowVector indices(getRows()), column(getRows());
        unsigned int i, j;
        double sign;

        inv.luDecomposition(indices, &sign);

        for (j = 0; j < getColumns(); j++) {
                column.zero();
                column(j) = 1;
                inv.luBackSubstitution(indices, column);
                for (i = 0; i < getRows(); i++) {
                        inv(i,j) = column(i);
                }
        }

        return inv;
}
```

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

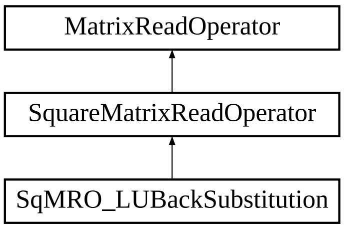
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.66 SqMRO_LUBackSubstitution Class Reference

Performs LU back substitution of matrix.

`#include <matrix_operator.h>`

Inheritance diagram for SqMRO_LUBackSubstitution::



Public Member Functions

- **SquareMatrix operator()** (**RowVector** &indx, **RowVector** &b) const

  Returns the LU back substitution of this matrix.

6.66.1 Detailed Description

Performs LU back substitution of matrix.

Author:

Lee Netherton and Peter Mendham

Definition at line 918 of file matrix_operator.h.

6.66.2 Member Function Documentation

6.66.2.1 SquareMatrix SqMRO_LUBackSubstitution::operator() (RowVector & *indx*, RowVector & *b*) const

Returns the LU back substitution of this matrix.

Definition at line 907 of file matrix_operator.cpp.

```
907                                                                                    {
908 
909         SquareMatrix a(*m_thisMatrix);
910         int n, i, j, ip, ii = -1;
911         double sum;
912 
913         n = a.getRows();
914 
915         for (i = 0; i < n; i++) {
916                 ip = (int)indx(i);
917                 sum = b.element(ip);
918                 b.element(ip) = b.element(i);
919                 if (ii != -1) {
920                         for (j = ii; j < i; j++) {
```

```
                                        sum -= a.element(i,j) * b.element(j);
                                }
                        } else if (sum) {
                                ii = i;
                        }
                        b.element(i) = sum;
                }
                for (i = (n - 1); i >= 0; i--) {
                        sum = b.element(i);
                        for (j = (i + 1); j < n; j++) {
                                sum -= a.element(i,j) * b.element(j);
                        }
                        b.element(i) = sum / a.element(i,i);
                }

                return a;
}
```

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

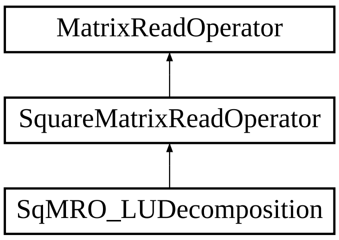
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.67 SqMRO_LUDecomposition Class Reference

Performs LU decomposition of this matrix.

`#include <matrix_operator.h>`

Inheritance diagram for SqMRO_LUDecomposition::



Public Member Functions

- **SquareMatrix operator()** (**RowVector** &indx, double ∗d) const

  Returns the LU decomposition ofthis matrix.

6.67.1 Detailed Description

Performs LU decomposition of this matrix.

Author:

Lee Netherton and Peter Mendham

Definition at line 908 of file matrix_operator.h.

6.67.2 Member Function Documentation

6.67.2.1 SquareMatrix SqMRO_LUDecomposition::operator() (RowVector & *indx*, double ∗ *d*) const

Returns the LU decomposition ofthis matrix.

Definition at line 825 of file matrix_operator.cpp.

```
825                                                                             {
826 
827         SquareMatrix a(*m_thisMatrix);
828         int n, i, imax, j, k;
829         double big, dum, sum, temp;
830         RowVector vv(a.getRows());
831 
832         n = a.getRows();
833 
834         // No row interchanges yet
835         *d = 1.0;
836 
837         // Loop over rows to get implicit scaling information
838         for (i = 0; i < n; i++) {
```

```
                big = 0.0;
                for (j = 0; j < n; j++) {
                        temp = fabs(a.element(i,j));
                        if (temp > big) {
                                big = temp;
                        }
                }
                if (big == 0.0) {
                        error("SquareMatrixAlias::luDecomp : MatrixAlias is rank defficient\n\n");
                        return a;
                }
                vv(i) = 1 / big;
        }

        // Loop over columns (Crout's method)
        for (j = 0; j < n; j++) {
                for (i = 0; i < j; i++) {
                        sum = a.element(i,j);
                        for (k = 0; k < i; k++) {
                                sum -= a.element(i,k) * a.element(k,j);
                        }
                        a.element(i,j) = sum;
                }
                big = 0;
                for (i = j; i < n; i++) {
                        sum = a.element(i,j);
                        for (k = 0; k < j; k++) {
                                sum -= a.element(i,k) * a.element(k,j);
                        }
                        a.element(i,j) = sum;

                        dum = vv(i) * fabs(sum);
                        if (dum >= big) {
                                big = dum;
                                imax = i;
                        }
                }
                if (j != imax) {
                        for (k = 0; k < n; k++) {
                                dum = a.element(imax,k);
                                a.element(imax,k) = a.element(j,k);
                                a.element(j,k) = dum;
                        }
                        *d = -(*d);
                        vv(imax) = vv(j);
                }
                indx.element(j) = imax;
                // Singularity may arise as aresult of rounding errors
                // Substitute in small values for zeros
                if (a.element(j,j) == 0) {
                        error("SquareMatrixAlias::luDecomp : MatrixAlias is rank defficient\n\n");
                        a.element(j,j) = 1e-100;
                }
                if (j != n) {
                        dum = 1 / a.element(j,j);
                        for (i = j+1; i < n; i++) {
                                a.element(i,j) *= dum;
                        }
                }
        }

        return a;
}
```

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

- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**

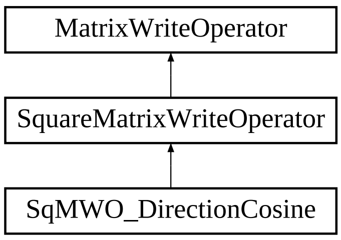
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.68 SqMWO_DirectionCosine Class Reference

Makes this matrix a direction cosine matrix.

`#include <matrix_operator.h>`

Inheritance diagram for SqMWO_DirectionCosine::



Public Member Functions

- void **operator()** (**ColumnVectorAlias** attitude)

  Use Column vector of angles.

- void **operator()** (double phi, double theta, double psi)

  Use direct angles.

6.68.1 Detailed Description

Makes this matrix a direction cosine matrix.

Author:

Lee Netherton and Peter Mendham

Definition at line 954 of file matrix_operator.h.

6.68.2 Member Function Documentation

6.68.2.1 void SqMWO_DirectionCosine::operator() (double *phi*, double *theta*, double *psi*)

Use direct angles.

Definition at line 1008 of file matrix_operator.cpp.

```
                                                                          {

        double sx, sy, sz;
        double cx, cy, cz;

        // Check size of square matrix is valid
        if (getRows() != 3) {
              error("Cannot compute direction cosine matrix for a matrix without 3 rows and columns\n");
              m_thisMatrix->zero();
              //return *this;
```

```
1018         }
1019 
1020         // Calculate sines and cosines
1021         sx = sin(phi);
1022         sy = sin(theta);
1023         sz = sin(psi);
1024         cx = cos(phi);
1025         cy = cos(theta);
1026         cz = cos(psi);
1027 
1028         // Set the elements of the matrix
1029         element(0,0) = cy*cz;
1030         element(0,1) = cy*sz;
1031         element(0,2) = -sy;
1032 
1033         element(1,0) = sx*sy*cz-cx*sz;
1034         element(1,1) = sx*sy*sz+cx*cz;
1035         element(1,2) = sx*cy;
1036 
1037         element(2,0) = cx*sy*cz+sx*sz;
1038         element(2,1) = cx*sy*sz-sx*cz;
1039         element(2,2) = cx*cy;
1040 
1041         // Return a reference to ourselves
1042         //return *this;
1043 }
```

6.68.2.2 void SqMWO_DirectionCosine::operator() (ColumnVectorAlias *attitude*)

Use Column vector of angles.

Definition at line 995 of file matrix_operator.cpp.

```
995                                                             {
996         if (attitude.getRows() != 3) {
997                 error("Cannot compute direction cosine matrix for an attitude vector without 3 rows\n");
998                 m_thisMatrix->zero();
999                 //return *this;
1000         }
1001         m_thisMatrix->directionCosine(attitude.element(0), attitude.element(1), attitude.element(2));
1002 }
```

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

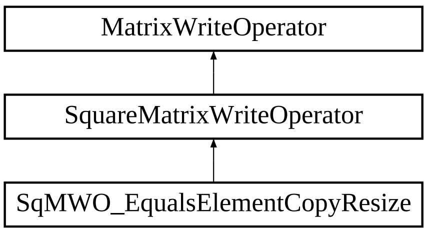
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.69 SqMWO_EqualsElementCopyResize Class Reference

Copy elements (Resize if necessesary).

`#include <matrix_operator.h>`

Inheritance diagram for SqMWO_EqualsElementCopyResize::



Public Member Functions

- void **operator()** (const **MatrixAliasConstant** &copy) const
  Checks to see if copy is a square matrix, if so copies element in.

6.69.1 Detailed Description

Copy elements (Resize if necessesary).

Author:
Lee Netherton

Definition at line 933 of file matrix_operator.h.

6.69.2 Member Function Documentation

6.69.2.1 void SqMWO_EqualsElementCopyResize::operator() (const MatrixAliasConstant & *copy*) const

Checks to see if copy is a square matrix, if so copies element in.

Parameters:
copy **Matrix**(p. 44) to copy (must be square)

Definition at line 945 of file matrix_operator.cpp.

```
{
                unsigned int i, j;

                if(copy.isSquareMatrix() == 0)
                {
                        error("SqMWO_EqualsElementCopyResize: Matrix to copy is not a square matrix\n");
                        return;
                }


                if(copy.getColumns() != getColumns()) {
                        // Resize!!

                        // Change values
                        setRows(copy.getRows());
                        setColumns(copy.getColumns());

                        // Delete old memory
                        delete[] getDataPointer();

                        // Allocate new memory
                        setDataPointer(new double[getRows()*getColumns()]);
                }


                for (i = 0; i < getRows(); i++) {
                        for (j = 0; j < getColumns(); j++) {
                                element(i, j) = copy.element(i, j);
                        }
                }
}
```

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

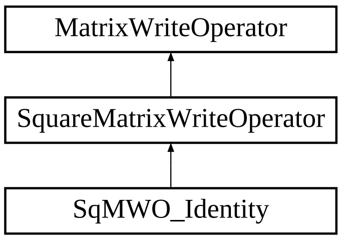
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.70 SqMWO_Identity Class Reference

Makes this matrix the identity matrix.

`#include <matrix_operator.h>`

Inheritance diagram for SqMWO_Identity::



Public Member Functions

- void **operator()** () const

  Make this matrix the identity matrix.

6.70.1 Detailed Description

Makes this matrix the identity matrix.

Author:

Lee Netherton and Peter Mendham

Definition at line 944 of file matrix_operator.h.

6.70.2 Member Function Documentation

6.70.2.1 void SqMWO_Identity::operator() () const

Make this matrix the identity matrix.

Definition at line 978 of file matrix_operator.cpp.

```
979 {
980         unsigned int i;
981 
982         m_thisMatrix->zero();
983 
984         for (i = 0; i < getRows(); i++) {
985                 element(i,i) = 1;
986         }
987 
988 }
```

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

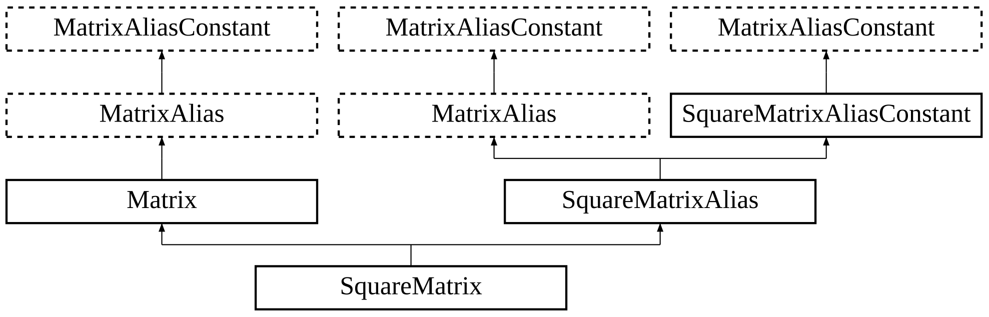
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_operator.cpp**

6.71 SquareMatrix Class Reference

The standard SquareMatrix class.

`#include <matrix.h>`

Inheritance diagram for SquareMatrix::



Public Member Functions

- **SquareMatrix** (const unsigned int size)
  Sized constructor.
- **SquareMatrix** (const **MatrixAliasConstant** &copy)
  Base class copy constructor.
- **SquareMatrix** (const **SquareMatrix** &copy)
  Copy constructor.
- virtual ~**SquareMatrix** ()
  SquareMatrix Destructor.
- **SquareMatrix** & **operator=** (const **MatrixAliasConstant** &copy)
  Base class assignment operator.
- **SquareMatrix** & **operator=** (const **SquareMatrix** &copy)
  Assignment operator.

Public Attributes

- **SqMWO_EqualsElementCopyResize equals**
  Checks to see if the operand is compatable (i.e. square) and then copies data in.

Protected Member Functions

- void **_constructSquareMatrix** (const unsigned int size)
  Sized constructor.

- void **_constructSquareMatrix** (const **MatrixAliasConstant** &copy)
  Copy constructor.

6.71.1 Detailed Description

The standard SquareMatrix class.

Author:
Lee Netherton

The SquareMatrix class provides the user with a pre made square matrix. It will allocate its own memory, and is provided with a full complement of matrix and square matrix operators.

Definition at line 1179 of file matrix.h.

6.71.2 Constructor & Destructor Documentation

6.71.2.1 SquareMatrix::SquareMatrix (const unsigned int *size*) `[inline]`

Sized constructor.

Creates a new square matrix of a given size.

Parameters:
size Number of rows and columns matrix has

Definition at line 1197 of file matrix.h.

```
1197 {_constructSquareMatrix(size);}
```

6.71.2.2 SquareMatrix::SquareMatrix (const MatrixAliasConstant & *copy*) `[inline]`

Base class copy constructor.

Makes a copy of any another matrix.

Parameters:
copy Reaference to matrix to copy

Definition at line 1203 of file matrix.h.

```
1203 {_constructSquareMatrix(copy);}
```

6.71.2.3 SquareMatrix::SquareMatrix (const SquareMatrix & *copy*) `[inline]`

Copy constructor.

Makes a copy of another matrix.

Parameters:

copy Reaference to matrix to copy

Definition at line 1209 of file matrix.h.

```
1209 {_constructSquareMatrix(copy);}
```

6.71.2.4 SquareMatrix::∼SquareMatrix () `[virtual]`

SquareMatrix Destructor.

Virtual - the lowest derived class will always need to be called, as they all allocate memory in ther own ways. Definition at line 638 of file matrix.cpp.

```
639 {
640         #ifdef DEBUG_DESTRUCTOR
641                 printf("Destructor: ~SquareMatrix()\n");
642         #endif
643 }
```

6.71.3 Member Function Documentation

6.71.3.1 void SquareMatrix::_constructSquareMatrix (const MatrixAliasConstant & *copy*) `[protected]`

Copy constructor.

Allocates some memory, and calls the **Matrix**(p. 44) constructor function and the **SquareMatrix-Alias**(p. 202) blank constructor function. Finally, copys data in from copied matrix

Parameters:

copy **Matrix**(p. 44) to copy

Definition at line 606 of file matrix.cpp.

```
607 {
608         // Construct main base class
609         _constructMatrix(copy.m_matrixContainer->getRows(),copy.m_matrixContainer->getColumns());
610
611         // Construct blank base class
612         _constructSquareMatrixAlias();
613
614         // Construct operators
615         _constructSquareMatrixOperators();
616
617         // Copy information to this matrix
618         equals(copy);
619
620         #ifdef DEBUG_CONSTRUCTOR
621                 printf("Constructed: SquareMatrix::Copy Constructor\n");
622         #endif
623 }
```

6.71.3.2 void SquareMatrix::_constructSquareMatrix (const unsigned int *size*) [protected]

Sized constructor.

Allocates some memory, and calls the **Matrix**(p. 44) constructor function and the **SquareMatrix-Alias**(p. 202) blank constructor function.

Parameters:

size Number of rows and columns matrix has

Definition at line 587 of file matrix.cpp.

```
588 {
589         // Construct main base class
590         _constructMatrix(size,size);
591
592         // Construct blank base class
593         _constructSquareMatrixAlias();
594
595         // Construct operators
596         _constructSquareMatrixOperators();
597
598         #ifdef DEBUG_CONSTRUCTOR
599                 printf("Constructed: SquareMatrix::Sized Constructor\n");
600         #endif
601 }
```

6.71.3.3 SquareMatrix& SquareMatrix::operator= (const SquareMatrix & *copy*) [inline]

Assignment operator.

Definition at line 1257 of file matrix.h.

```
1257 {return operator=((MatrixAliasConstant&)copy);}
```

6.71.3.4 SquareMatrix& SquareMatrix::operator= (const MatrixAliasConstant & *copy*) [inline]

Base class assignment operator.

Reimplemented from **Matrix** (p. 48).

Definition at line 1254 of file matrix.h.

```
1254 {equals(copy);return *this;}
```

6.71.4 Member Data Documentation

6.71.4.1 SqMWO_EqualsElementCopyResize SquareMatrix::equals

Checks to see if the operand is compatable (i.e. square) and then copies data in.

Reimplemented from **Matrix** (p. 48).

Definition at line 1187 of file matrix.h.

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

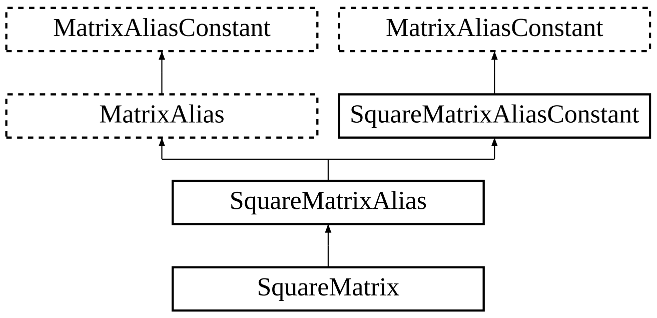
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix.cpp**

6.72 SquareMatrixAlias Class Reference

A SquareMatrixAlias class.

`#include <matrix.h>`

Inheritance diagram for SquareMatrixAlias::



Public Member Functions

- **SquareMatrixAlias** ()
  Default constructor.
- **SquareMatrixAlias** (const double ∗data, const unsigned int size)
  Pointer constructor.
- **SquareMatrixAlias** (const **MatrixAliasConstant** ∗alias)
  Alias constructor.
- **SquareMatrixAlias** (const **MatrixAliasConstant** &copy)
  Base class copy constructor.
- **SquareMatrixAlias** (const **SquareMatrixAlias** &copy)
  Copy constructor.
- virtual ∼**SquareMatrixAlias** ()
  SquareMatrixAlias Destructor.
- **SquareMatrixAlias** & **operator=** (const **MatrixAliasConstant** &copy)
  Base class assignment operator.
- **SquareMatrixAlias** & **operator=** (const **SquareMatrixAlias** &copy)
  Assignment operator.

Public Attributes

- **SqMWO_Identity identity**
  Makes this matrix an identity matrix.

- **SqMWO_DirectionCosine directionCosine**
  Makes this matrix direction cosine matrix.

Protected Member Functions

- void **_constructSquareMatrixAlias** (const double ∗data, const unsigned int size)
  Pointer constructor.
- void **_constructSquareMatrixAlias** (const **MatrixAliasConstant** &copy)
  Copy constructor.
- void **_constructSquareMatrixAlias** ()
  Blank constructor.

6.72.1 Detailed Description

A SquareMatrixAlias class.

Author:
Lee Netherton

The SquareMatrixAlias class provides all the functionality from the **MatrixAlias**(p. 49) class, but add specific functions intended for square matricies (like **inverse()**(p. 213), and **determinant()**(p. 213)). It also has specific square matrix write functions like **identity()**(p. 207).

Definition at line 1073 of file matrix.h.

6.72.2 Constructor & Destructor Documentation

6.72.2.1 SquareMatrixAlias::SquareMatrixAlias () `[inline]`

Default constructor.

Creates a SquareMatrixAlias shell. The **MatrixContainer**(p. 70), **MatrixReadAccess**(p. 73) and **MatrixWriteAccess**(p. 81) handles can be set later using the constructor function **_constructSquareMatrixAlias()**(p. 205) Definition at line 1094 of file matrix.h.

```
1094 {}
```

6.72.2.2 SquareMatrixAlias::SquareMatrixAlias (const double ∗ *data*, const unsigned int *size*) `[inline]`

Pointer constructor.

To create a square matrix that will access a pre-available data array.

Parameters:
data Pointer to data array. This will be the data storage for the matrix.

size Number of rows and columns the matrix has.

Definition at line 1101 of file matrix.h.

```
1101 {_constructSquareMatrixAlias(data,size);}
```

6.72.2.3 SquareMatrixAlias::SquareMatrixAlias (const MatrixAliasConstant ∗ *alias*) `[inline]`

Alias constructor.

To create a square matrix that will alias another matrix.

Parameters:

alias Pointer to a matrix which this matrix will alias.

Definition at line 1107 of file matrix.h.

```
1107 {_constructSquareMatrixAlias(*alias);}
```

6.72.2.4 SquareMatrixAlias::SquareMatrixAlias (const MatrixAliasConstant & *copy*) `[inline]`

Base class copy constructor.

Used when creating a SquareMatrixAlias matrix from another matrix.

```
 SquareMatrixAlias newSqMA(oldSqMA);
```

Parameters:

copy Reference to another matrix.

Definition at line 1114 of file matrix.h.

```
1114 {_constructSquareMatrixAlias(copy);}
```

6.72.2.5 SquareMatrixAlias::SquareMatrixAlias (const SquareMatrixAlias & *copy*) `[inline]`

Copy constructor.

Used when creating a SquareMatrixAlias matrix from another. Calls base class copy constructor

Parameters:

copy Reference to another matrix.

Definition at line 1121 of file matrix.h.

```
1121 {_constructSquareMatrixAlias(copy);}
```

6.72.2.6 SquareMatrixAlias::∼SquareMatrixAlias () `[virtual]`

SquareMatrixAlias Destructor.

Virtual - the lowest derived class will always need to be called, as they all allocate memory in ther own ways. Definition at line 565 of file matrix.cpp.

```
566 {
567         #ifdef DEBUG_DESTRUCTOR
568                 printf("Destructor: ~SquareMatrixAlias()\n");
569         #endif
570 }
```

6.72.3 Member Function Documentation

6.72.3.1 void SquareMatrixAlias::_constructSquareMatrixAlias () `[protected]`

Blank constructor.

Just constructs SquareMatrixAlias and **SquareMatrixAliasConstant**(p. 208) operators, and goes no further. Definition at line 538 of file matrix.cpp.

```
539 {
540         // Construct blank base classes and nothing else
541         _constructSquareMatrixAliasConstant();
542 
543         // Construct operators
544         _constructSquareMatrixAliasOperators();
545 
546 
547         #ifdef DEBUG_CONSTRUCTOR
548                 printf("Constructed: SquareMatrixAlias::Blank Constructor\n");
549         #endif
550 }
```

6.72.3.2 void SquareMatrixAlias::_constructSquareMatrixAlias (const MatrixAliasConstant & *copy*) `[protected]`

Copy constructor.

Copies the pointers to the **MatrixContainer**(p. 70), **MatrixReadOperator**(p. 77) and **Matrix-WriteOperator**(p. 85) members Definition at line 519 of file matrix.cpp.

```
520 {
521         // Construct main base class
522         _constructMatrixAlias(copy.m_matrixContainer->getDataPointer(),copy.m_matrixContainer->getRows(),copy.m_matr
523 
524         // Construct blank base class
525         _constructSquareMatrixAliasConstant();
526 
527         // Construct operators
528         _constructSquareMatrixAliasOperators();
529 
530         #ifdef DEBUG_CONSTRUCTOR
531                 printf("Constructed: SquareMatrixAlias::Copy Constructor\n");
532         #endif
533 }
```

6.72.3.3 void SquareMatrixAlias::_constructSquareMatrixAlias (const double * *data*, const unsigned int *size*) [protected]

Pointer constructor.

Sets the pointers to the **MatrixContainer**(p. 70), **MatrixReadOperator**(p. 77) and **MatrixWriteOperator**(p. 85) members Definition at line 500 of file matrix.cpp.

```
{
        // Construct main base class
        _constructMatrixAlias(data,size,size);

        // Construct blank base class
        _constructSquareMatrixAliasConstant();

        // Construct operators
        _constructSquareMatrixAliasOperators();

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: SquareMatrixAlias::Pointer Constructor\n");
        #endif
}
```

6.72.3.4 SquareMatrixAlias& SquareMatrixAlias::operator= (const SquareMatrixAlias & *copy*) [inline]

Assignment operator.

Definition at line 1168 of file matrix.h.

```
{return operator=((MatrixAliasConstant&)copy);}
```

6.72.3.5 SquareMatrixAlias& SquareMatrixAlias::operator= (const MatrixAliasConstant & *copy*) [inline]

Base class assignment operator.

Reimplemented from **MatrixAlias** (p. 55).

Reimplemented in **SquareMatrix** (p. 200).

Definition at line 1165 of file matrix.h.

```
{equals(copy);return *this;}
```

6.72.4 Member Data Documentation

6.72.4.1 SqMWO_DirectionCosine SquareMatrixAlias::directionCosine

Makes this matrix direction cosine matrix.

Definition at line 1084 of file matrix.h.

6.72.4.2 SqMWO_Identity SquareMatrixAlias::identity

Makes this matrix an identity matrix.

Definition at line 1081 of file matrix.h.

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

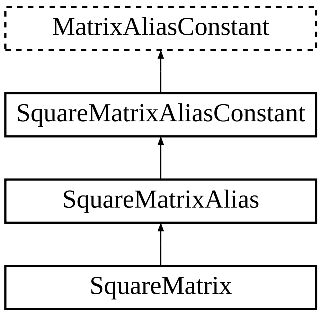
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix.cpp**

6.73 SquareMatrixAliasConstant Class Reference

A read-only **SquareMatrixAlias**(p. 202) class.

`#include <matrix.h>`

Inheritance diagram for SquareMatrixAliasConstant::



Public Member Functions

- **SquareMatrixAliasConstant** ()

  Default constructor.

- **SquareMatrixAliasConstant** (const double ∗data, const unsigned int size)

  Pointer constructor.

- **SquareMatrixAliasConstant** (const **MatrixAliasConstant** ∗alias)

  Alias constructor.

- **SquareMatrixAliasConstant** (const **MatrixAliasConstant** &copy)

  Base class copy constructor.

- **SquareMatrixAliasConstant** (const **SquareMatrixAliasConstant** &copy)

  Copy constructor.

- virtual ∼**SquareMatrixAliasConstant** ()

  SquareMatrixAliasConstant Destructor.

- **SquareMatrixAliasConstant** & **operator=** (const **MatrixAliasConstant** &copy)

  Base class assignment operator.

- **SquareMatrixAliasConstant** & **operator=** (const **SquareMatrixAliasConstant** &copy)

  Assignment operator.

Public Attributes

- **SqMRO_DeterminantLUDecomp determinant**

  Returns the determinant of this matrix (uses LU decomposition).

- **SqMRO_DeterminantBasic determinant2**

  Returns the determinant of this matrix (uses basic algorithm).

- **SqMRO_Cofactor cofactor**

  Returns the cofactor of an element.

- **SqMRO_InverseLUDecomp inverse**

  Returns the inverse of this matrix (uses LU decomposition).

- **SqMRO_InverseBasic inverse2**

  Returns the inverse of this matrix (uses basic algorithm).

- **SqMRO_Exponential exponential**

  Returns the matrix exponential using irreducible Pade approximation.

- **SqMRO_LUDecomposition luDecomposition**

  Returns an LU decomposition of this matrix.

- **SqMRO_LUBackSubstitution luBackSubstitution**

  Returns an LU back substitution of this matrix.

Protected Member Functions

- void **_constructSquareMatrixAliasConstant** (const double ∗data, const unsigned int size)

  Pointer constructor.

- void **_constructSquareMatrixAliasConstant** (const **MatrixAliasConstant** &copy)

  Copy constructor.

- void **_constructSquareMatrixAliasConstant** ()

  Blank constructor.

6.73.1 Detailed Description

A read-only **SquareMatrixAlias**(p. 202) class.

Author:

Lee Netherton

The SquareMatrixAliasConstant class provides all the functionality from the **MatrixAlias-Constant**(p. 57) class, but add specific functions intended for square matricies (like **inverse()**(p. 213), and **determinant()**(p. 213)).

Definition at line 943 of file matrix.h.

6.73.2 Constructor & Destructor Documentation

6.73.2.1 SquareMatrixAliasConstant::SquareMatrixAliasConstant () `[inline]`

Default constructor.

Creates a SquareMatrixAliasConstant shell. The **MatrixContainer**(p. 70) and **MatrixReadAccess**(p. 73) handles can be set later using the constructor function **_constructSquareMatrixAliasConstant()**(p. 211) Definition at line 986 of file matrix.h.

```
986 {}
```

6.73.2.2 SquareMatrixAliasConstant::SquareMatrixAliasConstant (const double * *data*, const unsigned int *size*) `[inline]`

Pointer constructor.

To create a read-only square matrix that will access a pre-available data array.

Parameters:

data Pointer to data array. This will be the data storage for the matrix.

size Number of rows and columns the matrix has.

Definition at line 993 of file matrix.h.

```
993 {_constructSquareMatrixAliasConstant(data,size);}
```

6.73.2.3 SquareMatrixAliasConstant::SquareMatrixAliasConstant (const MatrixAliasConstant * *alias*) `[inline]`

Alias constructor.

To create a read-only square matrix that will alias another matrix.

Parameters:

alias Pointer to a matrix which this matris will alias.

Definition at line 999 of file matrix.h.

```
999 {_constructSquareMatrixAliasConstant(*alias);}
```

6.73.2.4 SquareMatrixAliasConstant::SquareMatrixAliasConstant (const MatrixAliasConstant & *copy*) `[inline]`

Base class copy constructor.

Used when creating a SquareMatrixAliasConstant matrix from another.

```
 SquareMatrixAliasConstant newSqMAC(oldSqMAC);
```

Parameters:

copy Reference to another matrix.

Definition at line 1006 of file matrix.h.

```
1006 {_constructSquareMatrixAliasConstant(copy);}
```

6.73.2.5 SquareMatrixAliasConstant::SquareMatrixAliasConstant (const SquareMatrixAliasConstant & *copy*) [inline]

Copy constructor.

Used when creating a **MatrixAliasConstant**(p. 57) matrix from another. Calls base class copy constructor

Parameters:
    copy Reference to another matrix.

Definition at line 1013 of file matrix.h.

```
1013 {_constructSquareMatrixAliasConstant(copy);}
```

6.73.2.6 SquareMatrixAliasConstant::∼SquareMatrixAliasConstant () [virtual]

SquareMatrixAliasConstant Destructor.

Virtual - the lowest derived class will always need to be called, as they all allocate memory in ther own ways. Definition at line 477 of file matrix.cpp.

```
478 {
479         #ifdef DEBUG_DESTRUCTOR
480                 printf("Destructor: ~SquareMatrixAliasConstant()\n");
481         #endif
482 }
```

6.73.3 Member Function Documentation

6.73.3.1 void SquareMatrixAliasConstant::_constructSquareMatrixAliasConstant () [protected]

Blank constructor.

Just constructs SquareMatrixAliasConstant operators, and goes no further. Definition at line 447 of file matrix.cpp.

```
448 {
449         // Construct operators
450         _constructSquareMatrixAliasConstantOperators();
451
452         #ifdef DEBUG_CONSTRUCTOR
453                 printf("Constructed: SquareMatrixAliasConstant::Blank Constructor\n");
454         #endif
455 }
```

6.73.3.2 void SquareMatrixAliasConstant::_constructSquareMatrixAliasConstant (const MatrixAliasConstant & *copy*) `[protected]`

Copy constructor.

Copies the pointers to the **MatrixContainer**(p. 70) and **MatrixReadOperator**(p. 77) members Definition at line 431 of file matrix.cpp.

```
{
        // Construct main base class
        _constructMatrixAliasConstant(copy.m_matrixContainer->getDataPointer(),copy.m_matrixContainer->getRows(),co

        // Construct operators
        _constructSquareMatrixAliasConstantOperators();

        #ifdef DEBUG_CONSTRUCTOR
               printf("Constructed: SquareMatrixAliasConstant::Copy Constructor\n");
        #endif
}
```

6.73.3.3 void SquareMatrixAliasConstant::_constructSquareMatrixAliasConstant (const double ∗ *data*, const unsigned int *size*) `[protected]`

Pointer constructor.

Sets the pointers to the **MatrixContainer**(p. 70) and **MatrixReadOperator**(p. 77) members Definition at line 415 of file matrix.cpp.

```
{
        // Construct main base class
        _constructMatrixAliasConstant(data,size,size);

        // Construct operators
        _constructSquareMatrixAliasConstantOperators();

        #ifdef DEBUG_CONSTRUCTOR
               printf("Constructed: SquareMatrixAliasConstant::Pointer Constructor\n");
        #endif
}
```

6.73.3.4 SquareMatrixAliasConstant& SquareMatrixAliasConstant::operator= (const SquareMatrixAliasConstant & *copy*) `[inline]`

Assignment operator.

Definition at line 1060 of file matrix.h.

```
1060 {return operator=((MatrixAliasConstant&)copy);}
```

6.73.3.5 SquareMatrixAliasConstant& SquareMatrixAliasConstant::operator= (const MatrixAliasConstant & *copy*) `[inline]`

Base class assignment operator.

Reimplemented from **MatrixAliasConstant** (p. 66).

Reimplemented in **SquareMatrixAlias** (p. 206), and **SquareMatrix** (p. 200).

Definition at line 1057 of file matrix.h.

```
1057 {m_matrixReadAccess->error("Tried to assign to a constant square matrix\n");return *this;}
```

6.73.4 Member Data Documentation

6.73.4.1 SqMRO_Cofactor SquareMatrixAliasConstant::cofactor

Returns the cofactor of an element.

Definition at line 960 of file matrix.h.

6.73.4.2 SqMRO_DeterminantLUDecomp SquareMatrixAlias-Constant::determinant

Returns the determinant of this matrix (uses LU decomposition).

Definition at line 954 of file matrix.h.

6.73.4.3 SqMRO_DeterminantBasic SquareMatrixAliasConstant::determinant2

Returns the determinant of this matrix (uses basic algorithm).

Definition at line 957 of file matrix.h.

6.73.4.4 SqMRO_Exponential SquareMatrixAliasConstant::exponential

Returns the matrix exponential using irreducible Pade approximation.

Definition at line 969 of file matrix.h.

6.73.4.5 SqMRO_InverseLUDecomp SquareMatrixAliasConstant::inverse

Returns the inverse of this matrix (uses LU decomposition).

Definition at line 963 of file matrix.h.

6.73.4.6 SqMRO_InverseBasic SquareMatrixAliasConstant::inverse2

Returns the inverse of this matrix (uses basic algorithm).

Definition at line 966 of file matrix.h.

6.73.4.7 SqMRO_LUBackSubstitution SquareMatrixAliasConstant::luBack-Substitution

Returns an LU back substitution of this matrix.

Definition at line 975 of file matrix.h.

6.73.4.8 SqMRO_LUDecomposition SquareMatrixAliasConstant::luDecomposition

Returns an LU decomposition of this matrix.

Definition at line 972 of file matrix.h.

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

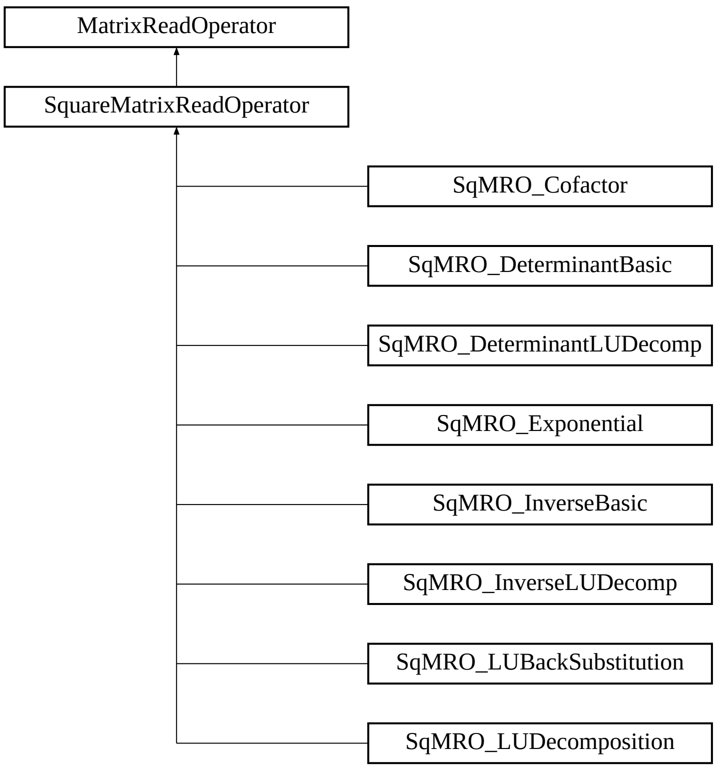
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix.cpp**

6.74 SquareMatrixReadOperator Class Reference

Base class for SquareMatrixReadOperators.

`#include <matrix_operator.h>`

Inheritance diagram for SquareMatrixReadOperator::



Public Member Functions

- **SquareMatrixReadOperator** ()

  Default Constructor.

- **SquareMatrixReadOperator** (**SquareMatrixAliasConstant** ∗squareMatrixAliasConstant)

  Full Constructor.

- void **_constructSquareMatrixReadOperator** (**SquareMatrixAliasConstant** ∗squareMatrixAliasConstant)

  Manual Constructor.

Protected Attributes

- **SquareMatrixAliasConstant** ∗ **m_thisMatrix**

  Pointer to owner matrix.

6.74.1 Detailed Description

Base class for SquareMatrixReadOperators.

Author:
Lee Netherton

Most importantly provides m_thisMatrix with the right kind of pointer.

Definition at line 165 of file matrix_operator.h.

6.74.2 Constructor & Destructor Documentation

6.74.2.1 SquareMatrixReadOperator::SquareMatrixReadOperator () `[inline]`

Default Constructor.

Creates an empty operator class which can then be more full constructed using **_construct-SquareMatrixReadOperator()**(p. 216) Definition at line 177 of file matrix_operator.h.

```
177 {}
```

6.74.2.2 SquareMatrixReadOperator::SquareMatrixReadOperator (SquareMatrixAliasConstant ∗ *squareMatrixAliasConstant*) `[inline]`

Full Constructor.

Creates an operator class which takes and stores a pointer to an owner matrix

Parameters:
squareMatrixAliasConstant Pointer to owner matrix

Definition at line 183 of file matrix_operator.h.

```
183                                                                                          :
184                         MatrixReadOperator((MatrixAliasConstant *)squareMatrixAliasConstant),
185                         m_thisMatrix(squareMatrixAliasConstant)
186                         {}
```

6.74.3 Member Function Documentation

6.74.3.1 void SquareMatrixReadOperator::_constructSquareMatrixReadOperator (SquareMatrixAliasConstant ∗ *squareMatrixAliasConstant*) `[inline]`

Manual Constructor.

Constructs the class manually by setting the owner pointer

Parameters:
squareMatrixAliasConstant Pointer to owner matrix

Definition at line 192 of file matrix_operator.h.

```
            {
                    _constructMatrixReadOperator((MatrixAliasConstant *)squareMatrixAliasConstant);

                    m_thisMatrix = squareMatrixAliasConstant;
            }
```

6.74.4 Member Data Documentation

6.74.4.1 SquareMatrixAliasConstant∗ SquareMatrixReadOperator::m_thisMatrix [protected]

Pointer to owner matrix.

Reimplemented from **MatrixReadOperator** (p. 80).

Definition at line 170 of file matrix_operator.h.

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

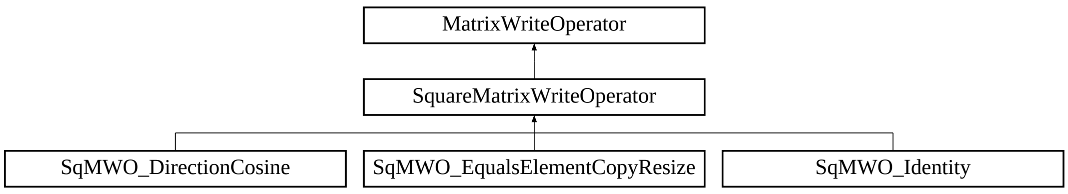
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**

6.75 SquareMatrixWriteOperator Class Reference

Base class for SquareMatrixWriteOperators.

`#include <matrix_operator.h>`

Inheritance diagram for SquareMatrixWriteOperator::



Public Member Functions

- **SquareMatrixWriteOperator** ()

  Default Constructor.

- **SquareMatrixWriteOperator** (**SquareMatrixAlias** ∗squareMatrixAlias)

  Full Constructor.

- void **_constructSquareMatrixWriteOperator** (**SquareMatrixAlias** ∗squareMatrix-Alias)

  Manual Constructor.

Protected Attributes

- **SquareMatrixAlias** ∗ **m_thisMatrix**

  Pointer to owner matrix.

6.75.1 Detailed Description

Base class for SquareMatrixWriteOperators.

Author:
Lee Netherton

Most importantly provides m_thisMatrix with the right kind of pointer.

Definition at line 206 of file matrix_operator.h.

6.75.2 Constructor & Destructor Documentation

6.75.2.1 SquareMatrixWriteOperator::SquareMatrixWriteOperator () `[inline]`

Default Constructor.

Creates an empty operator class which can then be more full constructed using **_construct-SquareMatrixWriteOperator()**(p. 219) Definition at line 218 of file matrix_operator.h.

```
218 {}
```

6.75.2.2 SquareMatrixWriteOperator::SquareMatrixWriteOperator (SquareMatrixAlias ∗ *squareMatrixAlias*) `[inline]`

Full Constructor.

Creates an operator class which takes and stores a pointer to an owner matrix

Parameters:

squareMatrixAlias Pointer to owner matrix

Definition at line 224 of file matrix_operator.h.

```
224                                                                          :
225                         MatrixWriteOperator((MatrixAlias *)squareMatrixAlias),
226                         m_thisMatrix(squareMatrixAlias)
227                         {}
```

6.75.3 Member Function Documentation

6.75.3.1 void SquareMatrixWriteOperator::_constructSquareMatrixWriteOperator (SquareMatrixAlias ∗ *squareMatrixAlias*) `[inline]`

Manual Constructor.

Constructs the class manually by setting the owner pointer

Parameters:

squareMatrixAlias Pointer to owner matrix

Definition at line 233 of file matrix_operator.h.

```
234                 {
235                         _constructMatrixWriteOperator((MatrixAlias *)squareMatrixAlias);
236 
237                         m_thisMatrix = squareMatrixAlias;
238                 }
```

6.75.4 Member Data Documentation

6.75.4.1 SquareMatrixAlias∗ SquareMatrixWriteOperator::m_thisMatrix `[protected]`

Pointer to owner matrix.

Reimplemented from **MatrixWriteOperator** (p. 89).

Definition at line 211 of file matrix_operator.h.

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

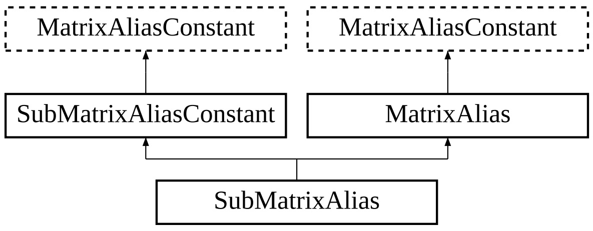
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_operator.h**

6.76 SubMatrixAlias Class Reference

Accesses just a portion of a matrix (and provides write access).

`#include <matrix.h>`

Inheritance diagram for SubMatrixAlias::



Public Member Functions

- **SubMatrixAlias** (const **MatrixAliasConstant** ∗original, const unsigned int rowStart, const unsigned int rowEnd, const unsigned int columnStart, const unsigned int columnEnd)

  Pointer constructor.

- **SubMatrixAlias** (const **MatrixAliasConstant** &original, const unsigned int rowStart, const unsigned int rowEnd, const unsigned int columnStart, const unsigned int columnEnd)

  Reference constructor.

- **SubMatrixAlias** (const **SubMatrixAlias** &copy)

  Copy constructor.

- virtual ∼**SubMatrixAlias** ()

  SubMatrixAlias Destructor.

- **SubMatrixAlias** & **operator=** (const **MatrixAliasConstant** &copy)

  Base class assignment operator.

- **SubMatrixAlias** & **operator=** (const **SubMatrixAlias** &copy)

  Assignment operator.

Protected Member Functions

- void **_constructSubMatrixAlias** (const **MatrixContainer** ∗original_container, const unsigned int rowStart, const unsigned int rowEnd, const unsigned int columnStart, const unsigned int columnEnd)

  Container constructor.

- void **_constructSubMatrixAlias** (const **SubMatrixContainer** ∗original_subContainer)

  SubContainer constructor.

6.76.1 Detailed Description

Accesses just a portion of a matrix (and provides write access).

Author:

Lee Netherton

The SubMatrixAlias class can be useful for "masking" out, all but the useful parts of a matrix. One particulary useful feature of the SubMatrixAlias class comes about when you use them in conjunction with the MatrixAlias::subMatrixAlias() operator function. Suppose you have two matrices, and you wanted to make one portion of one matrix equal to a portion of the other. The two matrices look like this:

```
  Matrix 1:          Matrix 2:

  [ a  b  c  d ]     [ q  r  s  t ]
  [ e  f  g  h ]     [ u  v  w  x ]
  [ i  j  k  l ]     [ y  z  1  2 ]
  [ m  n  o  p ]     [ 3  4  5  6 ]
```

If we wanted to make the top right portion of **Matrix**(p. 44) 1 equal to the top right corner of **Matrix**(p. 44) 2 we could do:

```
mat1.subMatrixAlias(0,1,2,3) = mat2.subMatrixAlias(0,1,2,3);
```

The matrices would then look like this:

```
  Matrix 1:          Matrix 2:

  [ a  b  s  t ]     [ q  r  s  t ]
  [ e  f  w  x ]     [ u  v  w  x ]
  [ i  j  k  l ]     [ y  z  1  2 ]
  [ m  n  o  p ]     [ 3  4  5  6 ]
```

Definition at line 854 of file matrix.h.

6.76.2 Constructor & Destructor Documentation

6.76.2.1 SubMatrixAlias::SubMatrixAlias (const MatrixAliasConstant ∗ *original*, const unsigned int *rowStart*, const unsigned int *rowEnd*, const unsigned int *columnStart*, const unsigned int *columnEnd*) `[inline]`

Pointer constructor.

Creates a SubMatrixAlias from a pointer to another matrix. The other values are the start and end rows and columns for the submatrix. See **SubMatrixAliasConstant()**(p. 226) for more details.

Parameters:

original Pointer to the matrix to be aliased

rowStart The index of the row to start the submatrix on

rowEnd The index of the row to stop the submatrix on

columnStart The index of the column to start the submatrix on

columnEnd The index of the column to stop the submatrix on

Definition at line 869 of file matrix.h.

```
869 {_constructSubMatrixAlias(original->m_matrixContainer, rowStart, rowEnd, columnStart, columnEnd);}
```

6.76.2.2 SubMatrixAlias::SubMatrixAlias (const MatrixAliasConstant & *original*, const unsigned int *rowStart*, const unsigned int *rowEnd*, const unsigned int *columnStart*, const unsigned int *columnEnd*) `[inline]`

Reference constructor.

Creates a SubMatrixAlias from a reference to another matrix. The other values are the start and end rows and columns for the submatrix.

Parameters:

original Reference to the matrix to be aliased

rowStart The index of the row to start the submatrix on

rowEnd The index of the row to stop the submatrix on

columnStart The index of the column to start the submatrix on

columnEnd The index of the column to stop the submatrix on

Definition at line 880 of file matrix.h.

```
880 {_constructSubMatrixAlias(original.m_matrixContainer, rowStart, rowEnd, columnStart, columnEnd);}
```

6.76.2.3 SubMatrixAlias::SubMatrixAlias (const SubMatrixAlias & *copy*) `[inline]`

Copy constructor.

Makes a copy of another submatrix.

Parameters:

copy Reaference to submatrix to copy

Definition at line 886 of file matrix.h.

```
886 {_constructSubMatrixAlias(copy.m_subMatrixContainer);}
```

6.76.2.4 SubMatrixAlias::~SubMatrixAlias () `[virtual]`

SubMatrixAlias Destructor.

Virtual - the lowest derived class will always need to be called, as they all allocate memory in ther own ways. Definition at line 390 of file matrix.cpp.

```
391 {
392         // SubMatrixContainer will be deleted by MatrixAliasConstant
393         // SubMatrixReadAccess will be deleted by MatrixAliasConstant
394         // SubMatrixWriteAccess will be deleted by MatrixAlias
395 
396         #ifdef DEBUG_DESTRUCTOR
397                 printf("Destructor: ~SubMatrixAlias()\n");
398         #endif
399 }
```

6.76.3 Member Function Documentation

6.76.3.1 void SubMatrixAlias::_constructSubMatrixAlias (const SubMatrixContainer ∗ *original_ subContainer*) `[protected]`

SubContainer constructor.

Creates a new **SubMatrixContainer**(p. 231) from a copy of an existing one, and a new **SubMatrixWriteAccess**(p. 240), then passes the handles back to **SubMatrixAlias-Constant**(p. 225)

Parameters:

original_ subContainer Pointer to the subMatrix's container class (to make a copy of)

Definition at line 374 of file matrix.cpp.

```
{
        // Create new SubMatrixContainer
        m_subMatrixContainer = new SubMatrixContainer(*original_subContainer);

        // Pass container to MatrixAlias
        _constructMatrixAlias(m_subMatrixContainer, new SubMatrixReadAccess(this), new SubMatrixWriteAccess(this));

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: SubMatrixAlias::SubContainer Constructor\n");
        #endif
}
```

6.76.3.2 void SubMatrixAlias::_constructSubMatrixAlias (const MatrixContainer ∗ *original_ container*, const unsigned int *rowStart*, const unsigned int *rowEnd*, const unsigned int *columnStart*, const unsigned int *columnEnd*) `[protected]`

Container constructor.

Creates a new **SubMatrixContainer**(p. 231) and **SubMatrixWriteAccess**(p. 240), and passes the handles back to **SubMatrixAliasConstant**(p. 225)

Parameters:

original_ container Pointer to the original matrix's container class

rowStart The index of the row to start the submatrix on

rowEnd The index of the row to stop the submatrix on

columnStart The index of the column to start the submatrix on

columnEnd The index of the column to stop the submatrix on

Definition at line 358 of file matrix.cpp.

```
{
        // Create new SubMatrixContainer
        m_subMatrixContainer = new SubMatrixContainer(original_container, rowStart, rowEnd, columnStart, columnEnd)

        // Pass container to MatrixAlias
        _constructMatrixAlias(m_subMatrixContainer, new SubMatrixReadAccess(this), new SubMatrixWriteAccess(this));

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: SubMatrixAlias::Container Constructor\n");
        #endif
}
```

6.76.3.3 SubMatrixAlias& SubMatrixAlias::operator= (const SubMatrixAlias & *copy*) [inline]

Assignment operator.

Definition at line 932 of file matrix.h.

```
932 {return operator=((MatrixAliasConstant&)copy);}
```

6.76.3.4 SubMatrixAlias& SubMatrixAlias::operator= (const MatrixAliasConstant & *copy*) [inline]

Base class assignment operator.

Reimplemented from **MatrixAlias** (p. 55).

Definition at line 929 of file matrix.h.

```
929 {equals(copy);return *this;}
```

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

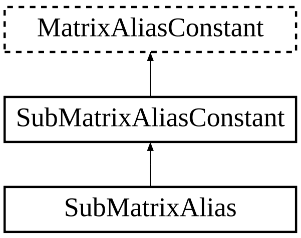
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix.cpp**

6.77 SubMatrixAliasConstant Class Reference

Accesses just a portion of a matrix.

`#include <matrix.h>`

Inheritance diagram for SubMatrixAliasConstant::



Public Member Functions

- **SubMatrixAliasConstant** ()

  Default constructor.

- **SubMatrixAliasConstant** (const **MatrixAliasConstant** ∗original, const unsigned int rowStart, const unsigned int rowEnd, const unsigned int columnStart, const unsigned int columnEnd)

  Pointer constructor.

- **SubMatrixAliasConstant** (const **MatrixAliasConstant** &original, const unsigned int rowStart, const unsigned int rowEnd, const unsigned int columnStart, const unsigned int columnEnd)

  Reference constructor.

- **SubMatrixAliasConstant** (const **SubMatrixAliasConstant** &copy)

  Copy constructor.

- virtual ∼**SubMatrixAliasConstant** ()

  SubMatrixAliasConstant Destructor.

- **SubMatrixAliasConstant** & **operator=** (const **MatrixAliasConstant** &copy)

  Base class assignment operator.

- **SubMatrixAliasConstant** & **operator=** (const **SubMatrixAliasConstant** &copy)

  Assignment operator.

Public Attributes

- **SubMatrixContainer** ∗ **m_subMatrixContainer**

  A pointer to the **SubMatrixContainer**(p. 231).

Protected Member Functions

- void **_constructSubMatrixAliasConstant** (const **MatrixContainer** *original_-container, const unsigned int rowStart, const unsigned int rowEnd, const unsigned int columnStart, const unsigned int columnEnd)

  Container constructor.

- void **_constructSubMatrixAliasConstant** (const **SubMatrixContainer** *original_-subContainer)

  SubContainer constructor.

6.77.1 Detailed Description

Accesses just a portion of a matrix.

Author:
Lee Netherton

The SubMatrixAliasConstant class can be useful for "masking" out, all but the useful parts of a matrix. Suppose you had a large matrix, but you only wanted to see the values in the top left corner of it:

```
void printJustASmallBit(Matrix& large_matrix)
{
    SubMatrixAliasConstant small(large_matrix,0,4,0,4); // SubMatrix is a 5x5 matrix aliasing
                                                        // the top corner of the large_matrix

    small.print();       // print out the small section of the large_matrix
}
```

Definition at line 694 of file matrix.h.

6.77.2 Constructor & Destructor Documentation

6.77.2.1 SubMatrixAliasConstant::SubMatrixAliasConstant () `[inline]`

Default constructor.

Creates a SubMatrixAliasConstant shell. The **SubMatrixContainer**(p. 231) and **SubMatrix-ReadAccess**(p. 236) handles can be set later using the constructor function **_constructSub-MatrixAliasConstant()**(p. 229) Definition at line 714 of file matrix.h.

```
714 {}
```

6.77.2.2 SubMatrixAliasConstant::SubMatrixAliasConstant (const MatrixAliasConstant * *original*, const unsigned int *rowStart*, const unsigned int *rowEnd*, const unsigned int *columnStart*, const unsigned int *columnEnd*) `[inline]`

Pointer constructor.

Creates a SubMatrixAliasConstant from a pointer to another matrix. The other values are the start and end rows and columns for the submatrix. For example, supposre you had an 8x8 matrix, and you wanted to make a submatrix alias the center 2x2 portion of it:

```
          Column Start (1)
                |Column End (2)
                |  |
    (0)  [ a | b  c | d ]
         [--------------]
    (1)  [ e | f  g | h ]-- Row Start (1)
         [   |      |   ]
    (2)  [ i | j  k | l ]-- Row End (2)
         [--------------]
    (3)  [ m | n  o | p ]

           (0) (1)(2) (3)
```

Remembering that the matricies are zero indexed, the code to prodice such a submatrix is as follows:

```
Matrix A(8,8);                         // Create our 8x8 matrix

SubMatrixAliasConstant subA(A,1,2,1,2); // Create the sub matrix

subA.print();                          // Will produce: [ f g ]
                                                        [ j k ]
```

Parameters:

original Pointer to the matrix to be aliased

rowStart The index of the row to start the submatrix on

rowEnd The index of the row to stop the submatrix on

columnStart The index of the column to start the submatrix on

columnEnd The index of the column to stop the submatrix on

Definition at line 755 of file matrix.h.

```
755 {_constructSubMatrixAliasConstant(original->m_matrixContainer, rowStart, rowEnd, columnStart, columnEnd);}
```

6.77.2.3 SubMatrixAliasConstant::SubMatrixAliasConstant (const MatrixAliasConstant & *original*, const unsigned int *rowStart*, const unsigned int *rowEnd*, const unsigned int *columnStart*, const unsigned int *columnEnd*) `[inline]`

Reference constructor.

Creates a SubMatrixAliasConstant from a reference to another matrix. The other values are the start and end rows and columns for the submatrix.

Parameters:

original Reference to the matrix to be aliased

rowStart The index of the row to start the submatrix on

rowEnd The index of the row to stop the submatrix on

columnStart The index of the column to start the submatrix on

columnEnd The index of the column to stop the submatrix on

Definition at line 766 of file matrix.h.

```
766 {_constructSubMatrixAliasConstant(original.m_matrixContainer, rowStart, rowEnd, columnStart, columnEnd);}
```

6.77.2.4 SubMatrixAliasConstant::SubMatrixAliasConstant (const SubMatrixAliasConstant & *copy*) [inline]

Copy constructor.

Makes a copy of another submatrix.

Parameters:

copy Reaference to submatrix to copy

Definition at line 772 of file matrix.h.

```
772 {_constructSubMatrixAliasConstant(copy.m_subMatrixContainer);}
```

6.77.2.5 SubMatrixAliasConstant::∼SubMatrixAliasConstant () [virtual]

SubMatrixAliasConstant Destructor.

Virtual - the lowest derived class will always need to be called, as they all allocate memory in ther own ways. Definition at line 333 of file matrix.cpp.

```
334 {
335         // SubMatrixContainer will be deleted by MatrixAliasConstant
336         // SubMatrixReadAccess will be deleted by MatrixAliasConstant
337
338         #ifdef DEBUG_DESTRUCTOR
339                 printf("Destructor: ~SubMatrixAliasConstant()\n");
340         #endif
341 }
```

6.77.3 Member Function Documentation

6.77.3.1 void SubMatrixAliasConstant::_constructSubMatrixAliasConstant (const SubMatrixContainer ∗ *original_subContainer*) [protected]

SubContainer constructor.

Creates a new **SubMatrixContainer**(p. 231) from a copy of an existing one, and a new **SubMatrixReadAccess**(p. 236), then passes the handles back to **MatrixAliasConstant**(p. 57)

Parameters:

original_subContainer Pointer to the subMatrix's container class (to make a copy of)

Definition at line 317 of file matrix.cpp.

```
{
        // Create new SubMatrixContainer
        m_subMatrixContainer = new SubMatrixContainer(*original_subContainer);

        // Pass container to MatrixAliasConstant
        _constructMatrixAliasConstant(m_subMatrixContainer, new SubMatrixReadAccess(this));

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: SubMatrixAliasConstant::SubContainer Constructor\n");
        #endif
}
```

6.77.3.2 void SubMatrixAliasConstant::_constructSubMatrixAliasConstant (const MatrixContainer * *original_container*, const unsigned int *rowStart*, const unsigned int *rowEnd*, const unsigned int *columnStart*, const unsigned int *columnEnd*) `[protected]`

Container constructor.

Creates a new **SubMatrixContainer**(p. 231) and **SubMatrixReadAccess**(p. 236), and passes the handles back to **MatrixAliasConstant**(p. 57)

Parameters:

original_container Pointer to the original matrix's container class

rowStart The index of the row to start the submatrix on

rowEnd The index of the row to stop the submatrix on

columnStart The index of the column to start the submatrix on

columnEnd The index of the column to stop the submatrix on

Definition at line 301 of file matrix.cpp.

```
{
        // Create new SubMatrixContainer
        m_subMatrixContainer = new SubMatrixContainer(original_container, rowStart, rowEnd, columnStart, columnEnd)

        // Pass container to MatrixAliasConstant
        _constructMatrixAliasConstant(m_subMatrixContainer, new SubMatrixReadAccess(this));

        #ifdef DEBUG_CONSTRUCTOR
                printf("Constructed: SubMatrixAliasConstant::Container Constructor\n");
        #endif
}
```

6.77.3.3 SubMatrixAliasConstant& SubMatrixAliasConstant::operator= (const SubMatrixAliasConstant & *copy*) `[inline]`

Assignment operator.

Definition at line 817 of file matrix.h.

```
{return operator=((MatrixAliasConstant&)copy);}
```

6.77.3.4 SubMatrixAliasConstant& SubMatrixAliasConstant::operator= (const MatrixAliasConstant & *copy*) `[inline]`

Base class assignment operator.

Reimplemented from **MatrixAliasConstant** (p. 66).

Reimplemented in **SubMatrixAlias** (p. 224).

Definition at line 814 of file matrix.h.

```
814 {m_matrixReadAccess->error("Tried to assign to a constant sub-matrix\n");return *this;}
```

6.77.4 Member Data Documentation

6.77.4.1 SubMatrixContainer∗ SubMatrixAliasConstant::m_subMatrixContainer

A pointer to the **SubMatrixContainer**(p. 231).

Only used to be able to make a copy of the **SubMatrixContainer**(p. 231). Definition at line 704 of file matrix.h.

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

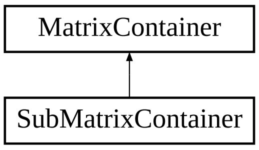
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix.cpp**

6.78 SubMatrixContainer Class Reference

Store for SubMatrix information.

`#include <matrix_container.h>`

Inheritance diagram for SubMatrixContainer::



Public Member Functions

- **SubMatrixContainer** (const **MatrixContainer** ∗original, unsigned int rowStart, unsigned int rowEnd, unsigned int columnStart, unsigned int columnEnd)

  Basic constructor.

- **SubMatrixContainer** (const **SubMatrixContainer** &copy)

  Copy constructor.

- const **MatrixContainer** ∗ **getOrigPointer** () const

  Resurns a pointer to the original matrix's **MatrixContainer**(p. 70)*.*

- unsigned int **getRowsOrig** () const

  Returns the number of rows in the original container.

- unsigned int **getColumnsOrig** () const

  Returns the number of rows in the original container.

- unsigned int **getRowStart** () const

  Returns the row start position.

- unsigned int **getRowEnd** () const

  Returns the row end position.

- unsigned int **getColumnStart** () const

  Returns the column start position.

- unsigned int **getColumnEnd** () const

  Returns the column end position.

- void **setRowStart** (const unsigned int num)

  Sets the row start position.

- void **setRowEnd** (const unsigned int num)

  Sets the row end position.

- void **setColumnStart** (const unsigned int num)

  Sets the column start position.

- void **setColumnEnd** (const unsigned int num)

  Sets the column end position.

6.78.1 Detailed Description

Store for SubMatrix information.

Author:
  Lee Netherton

The SubMatrixContainer class is used to store the sub matrix data as well as the usual matrix data. It holds a pointer to the original matrix's container, and also the start and end positions for the sub matrix with respect to the original.

Definition at line 82 of file matrix_container.h.

6.78.2 Constructor & Destructor Documentation

6.78.2.1 SubMatrixContainer::SubMatrixContainer (const MatrixContainer * *original*, unsigned int *rowStart*, unsigned int *rowEnd*, unsigned int *columnStart*, unsigned int *columnEnd*) `[inline]`

Basic constructor.

To create a SubMatrixContainer from an original **MatrixContainer**(p. 70) and start and end values for the rows and columns.

Parameters:
  original Pointer to the original matrix's container
  rowStart The starting row for the submatrix
  rowEnd The finishing row for the submatrix
  columnStart The starting column for the submatrix
  columnEnd The finishing column for the submatrix

Definition at line 113 of file matrix_container.h.

```
113
114                         MatrixContainer(original->getDataPointer(), (rowEnd-rowStart)+1, (columnEnd-columnStart)+1)
115                         m_rowStart(rowStart), m_rowEnd(rowEnd), m_columnStart(columnStart), m_columnEnd(columnEnd),
116                         m_originalContainer(original)
117                         {
118                                 #ifdef DEBUG_CONSTRUCTOR
119                                         printf("Constructor: SubMatrixContainer(MatrixContainer * original, int rowS
120                                 #endif
121                         }
```

6.78.2.2 SubMatrixContainer::SubMatrixContainer (const SubMatrixContainer & *copy*) `[inline]`

Copy constructor.

To create a SubMatrixContainer from a copy of another

Parameters:

copy Reference to SubMatrixContainer to copy

Definition at line 127 of file matrix_container.h.

```
127                                                 :
128                     MatrixContainer(copy.m_originalContainer->getDataPointer(), copy.getRows(), copy.getColumns
129                     m_rowStart(copy.getRowStart()), m_rowEnd(copy.getRowEnd()), m_columnStart(copy.getColumnSta
130                     m_originalContainer(copy.m_originalContainer)
131                     {
132                             #ifdef DEBUG_CONSTRUCTOR
133                                     printf("Constructor: SubMatrixContainer(SubMatrixContainer& copy)\n");
134                             #endif
135                     }
```

6.78.3 Member Function Documentation

6.78.3.1 unsigned int SubMatrixContainer::getColumnEnd () const `[inline]`

Returns the column end position.

Definition at line 159 of file matrix_container.h.

```
159 {return m_columnEnd;}
```

6.78.3.2 unsigned int SubMatrixContainer::getColumnsOrig () const `[inline]`

Returns the number of rows in the original container.

Definition at line 147 of file matrix_container.h.

```
147 {return m_originalContainer->getColumns();}
```

6.78.3.3 unsigned int SubMatrixContainer::getColumnStart () const `[inline]`

Returns the column start position.

Definition at line 156 of file matrix_container.h.

```
156 {return m_columnStart;}
```

6.78.3.4 const MatrixContainer∗ SubMatrixContainer::getOrigPointer () const [inline]

Resurns a pointer to the original matrix's **MatrixContainer**(p. 70).

Definition at line 141 of file matrix_container.h.

```
141 {return m_originalContainer;}
```

6.78.3.5 unsigned int SubMatrixContainer::getRowEnd () const [inline]

Returns the row end position.

Definition at line 153 of file matrix_container.h.

```
153 {return m_rowEnd;}
```

6.78.3.6 unsigned int SubMatrixContainer::getRowsOrig () const [inline]

Returns the number of rows in the original container.

Definition at line 144 of file matrix_container.h.

```
144 {return m_originalContainer->getRows();}
```

6.78.3.7 unsigned int SubMatrixContainer::getRowStart () const [inline]

Returns the row start position.

Definition at line 150 of file matrix_container.h.

```
150 {return m_rowStart;}
```

6.78.3.8 void SubMatrixContainer::setColumnEnd (const unsigned int *num*) [inline]

Sets the column end position.

Definition at line 171 of file matrix_container.h.

```
171 {m_columnEnd = num;}
```

6.78.3.9 void SubMatrixContainer::setColumnStart (const unsigned int *num*) [inline]

Sets the column start position.

Definition at line 168 of file matrix_container.h.

```
168 {m_columnStart = num;}
```

6.78.3.10 void SubMatrixContainer::setRowEnd (const unsigned int *num*) `[inline]`

Sets the row end position.

Definition at line 165 of file matrix_container.h.

```
165 {m_rowEnd = num;}
```

6.78.3.11 void SubMatrixContainer::setRowStart (const unsigned int *num*) `[inline]`

Sets the row start position.

Definition at line 162 of file matrix_container.h.

```
162 {m_rowStart = num;}
```

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

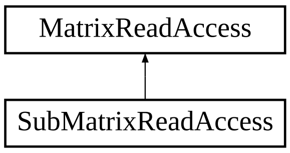
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_container.h**

6.79 SubMatrixReadAccess Class Reference

Read access for sub-matrix classes.

`#include <matrix_access.h>`

Inheritance diagram for SubMatrixReadAccess::



Public Member Functions

- **SubMatrixReadAccess** (**SubMatrixAliasConstant** ∗sMAC)

  Constructor.

- virtual const double & **readElement** (const unsigned int row, const unsigned int column) const
- virtual const double & **readElement** (const unsigned int index) const
- const unsigned int **getRowsOrig** () const

  Returns the number of rows in the original matrix.

- const unsigned int **getColumnsOrig** () const

  Returns the number of columns in the original matrix.

- const unsigned int **getRowStart** () const

  Returns the row start position.

- const unsigned int **getRowEnd** () const

  Returns the row end position.

- const unsigned int **getColumnStart** () const

  Returns the column start position.

- const unsigned int **getColumnEnd** () const

  Returns the column end position.

6.79.1 Detailed Description

Read access for sub-matrix classes.

Author:

Lee Netherton

The SubMatrixReadAccess class provides basic read operatons for the MatrixOperator classes, and the sub matrix class itself. Each sub matrix class will have a pointer to a SubMatrixReadAccess.

Definition at line 69 of file matrix_access.h.

6.79.2 Constructor & Destructor Documentation

6.79.2.1 SubMatrixReadAccess::SubMatrixReadAccess (SubMatrixAliasConstant * *sMAC*) `[inline]`

Constructor.

Sets pointer to owner matrix and constructs base class.

Parameters:
 sMAC Pointer to owner matrix

Definition at line 82 of file matrix_access.h.

```
82                                                      :
83                      MatrixReadAccess((MatrixAliasConstant *)sMAC),
84                      m_subMatrixAliasConstant(sMAC)
85                      {
86                              #ifdef DEBUG_CONSTRUCT
87                                      printf("Constructor: SubMatrixReadAccess(SubMatrixAliasConstant * sMAC)\n");
88                              #endif
89                      }
```

6.79.3 Member Function Documentation

6.79.3.1 const unsigned int SubMatrixReadAccess::getColumnEnd () const

Returns the column end position.

Definition at line 155 of file matrix_access.cpp.

```
155 {return m_subMatrixAliasConstant->m_subMatrixContainer->getColumnEnd();}
```

6.79.3.2 const unsigned int SubMatrixReadAccess::getColumnsOrig () const

Returns the number of columns in the original matrix.

Definition at line 151 of file matrix_access.cpp.

```
151 {return m_subMatrixAliasConstant->m_subMatrixContainer->getColumnsOrig();}
```

6.79.3.3 const unsigned int SubMatrixReadAccess::getColumnStart () const

Returns the column start position.

Definition at line 154 of file matrix_access.cpp.

```
154 {return m_subMatrixAliasConstant->m_subMatrixContainer->getColumnStart();}
```

6.79.3.4 const unsigned int SubMatrixReadAccess::getRowEnd () const

Returns the row end position.

Definition at line 153 of file matrix_access.cpp.

```
{return m_subMatrixAliasConstant->m_subMatrixContainer->getRowEnd();}
```

6.79.3.5 const unsigned int SubMatrixReadAccess::getRowsOrig () const

Returns the number of rows in the original matrix.

Definition at line 150 of file matrix_access.cpp.

```
{return m_subMatrixAliasConstant->m_subMatrixContainer->getRowsOrig();}
```

6.79.3.6 const unsigned int SubMatrixReadAccess::getRowStart () const

Returns the row start position.

Definition at line 152 of file matrix_access.cpp.

```
{return m_subMatrixAliasConstant->m_subMatrixContainer->getRowStart();}
```

6.79.3.7 const double & SubMatrixReadAccess::readElement (const unsigned int *index*) const [virtual]

Returns a read-only reference to the data member at specified position

Parameters:
index Row-wise position of element (zero-indexed)

Reimplemented from **MatrixReadAccess** (p. 75).

Definition at line 135 of file matrix_access.cpp.

```
{
        if (index >= (getRows()*getColumns())) {
                error("Matrix::element : Subscript out of range\n");
                return getDataPointer()[0];
        }

        return readElement(index / getColumns(), index % getColumns());

}
```

6.79.3.8 const double & SubMatrixReadAccess::readElement (const unsigned int *row*, const unsigned int *column*) const [virtual]

Returns a read-only reference to the data member at specified position

Parameters:

row Row position of desired element (zero indexed)

column Column position of desired element (zero indexed)

Reimplemented from **MatrixReadAccess** (p. 75).

Definition at line 124 of file matrix_access.cpp.

```
{
        if (row >= getRows() || column >= getColumns()) {
                error("Matrix::element : Subscript out of range\n");
                return getDataPointer()[0];
        }

        return getDataPointer()[(row+getRowStart())*(getColumnsOrig()) + (column+getColumnStart())];

}
```

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

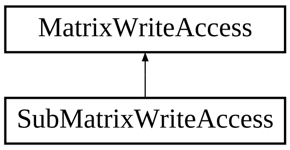
- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_access.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_access.cpp**

6.80 SubMatrixWriteAccess Class Reference

Write access for sub-matrix classes.

`#include <matrix_access.h>`

Inheritance diagram for SubMatrixWriteAccess::



Public Member Functions

- **SubMatrixWriteAccess** (**SubMatrixAlias** ∗sMA)

  Constructor.
- virtual double & **writeElement** (const unsigned int row, const unsigned int column) const
- virtual double & **writeElement** (const unsigned int index) const
- const unsigned int **getRowsOrig** () const

  Returns the number of rows in the original matrix.
- const unsigned int **getColumnsOrig** () const

  Returns the number of columns in the original matrix.
- const unsigned int **getRowStart** () const

  Returns the row start position.
- const unsigned int **getRowEnd** () const

  Returns the row end position.
- const unsigned int **getColumnStart** () const

  Returns the column start position.
- const unsigned int **getColumnEnd** () const

  Returns the column end position.

6.80.1 Detailed Description

Write access for sub-matrix classes.

Author:

Lee Netherton

The SubMatrixWriteAccess class provides basic write operatons for the MatrixOperator classes, and the sub matrix class itself. Each writable sub matrix class will have a pointer to a SubMatrix-WriteAccess.

Definition at line 187 of file matrix_access.h.

6.80.2 Constructor & Destructor Documentation

6.80.2.1 SubMatrixWriteAccess::SubMatrixWriteAccess (SubMatrixAlias * *sMA*) `[inline]`

Constructor.

Sets pointer to owner matrix

Parameters:
- ***sMA*** Pointer to owner matrix

Definition at line 200 of file matrix_access.h.

```
200                                                         :
201                         MatrixWriteAccess((MatrixAlias *)sMA),
202                         m_subMatrixAlias(sMA)
203                         {
204                                 #ifdef DEBUG_CONSTRUCT
205                                         printf("Constructor: SubMatrixWriteAccess(SubMatrixAlias * sMA)\n");
206                                 #endif
207                         }
```

6.80.3 Member Function Documentation

6.80.3.1 const unsigned int SubMatrixWriteAccess::getColumnEnd () const

Returns the column end position.

Definition at line 198 of file matrix_access.cpp.

```
198 {return m_subMatrixAlias->m_subMatrixContainer->getColumnEnd();}
```

6.80.3.2 const unsigned int SubMatrixWriteAccess::getColumnsOrig () const

Returns the number of columns in the original matrix.

Definition at line 194 of file matrix_access.cpp.

```
194 {return m_subMatrixAlias->m_subMatrixContainer->getColumnsOrig();}
```

6.80.3.3 const unsigned int SubMatrixWriteAccess::getColumnStart () const

Returns the column start position.

Definition at line 197 of file matrix_access.cpp.

```
197 {return m_subMatrixAlias->m_subMatrixContainer->getColumnStart();}
```

6.80.3.4 const unsigned int SubMatrixWriteAccess::getRowEnd () const

Returns the row end position.

Definition at line 196 of file matrix_access.cpp.

```
196 {return m_subMatrixAlias->m_subMatrixContainer->getRowEnd();}
```

6.80.3.5 const unsigned int SubMatrixWriteAccess::getRowsOrig () const

Returns the number of rows in the original matrix.

Definition at line 193 of file matrix_access.cpp.

```
193 {return m_subMatrixAlias->m_subMatrixContainer->getRowsOrig();}
```

6.80.3.6 const unsigned int SubMatrixWriteAccess::getRowStart () const

Returns the row start position.

Definition at line 195 of file matrix_access.cpp.

```
195 {return m_subMatrixAlias->m_subMatrixContainer->getRowStart();}
```

6.80.3.7 double & SubMatrixWriteAccess::writeElement (const unsigned int *index*) const `[virtual]`

Returns a writable reference to the data member at specified position

Parameters:
index Row-wise position of element (zero-indexed)

Reimplemented from **MatrixWriteAccess** (p. 83).

Definition at line 179 of file matrix_access.cpp.

```
180 {
181         if (index >= (getRows()*getColumns())) {
182                 error("Matrix::element : Subscript out of range\n");
183                 return getDataPointer()[0];
184         }
185 
186         return writeElement(index / getColumns(), index % getColumns());
187 
188 }
```

6.80.3.8 double & SubMatrixWriteAccess::writeElement (const unsigned int *row*, const unsigned int *column*) const `[virtual]`

Returns a writable reference to the data member at specified position

Parameters:

row Row position of desired element (zero indexed)

column Column position of desired element (zero indexed)

Reimplemented from **MatrixWriteAccess** (p. 84).

Definition at line 168 of file matrix_access.cpp.

```
{
        if (row >= getRows() || column >= getColumns()) {
                error("Matrix::element : Subscript out of range\n");
                return getDataPointer()[0];
        }

        return getDataPointer()[(row+getRowStart())*(getColumnsOrig()) + (column+getColumnStart())];

}
```

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

- Desktop/ltn100/Shared/MatrixClassLib/code/include/**matrix_access.h**
- Desktop/ltn100/Shared/MatrixClassLib/code/src/**matrix_access.cpp**

Chapter 7

MatrixClassLib File Documentation

7.1 Desktop/ltn100/Shared/MatrixClassLib/code/include/matrix.h File Reference

```
#include "matrix_container.h"
#include "matrix_operator.h"
#include "matrix_access.h"
#include <iostream.h>
```

Classes

- class **MatrixAliasConstant**

  A read-only **MatrixAlias**(p. 49) *class.*

- class **MatrixAlias**

  An alias of a **Matrix**(p. 44) *class, or to utilise a pre-available data array.*

- class **Matrix**

  The standard matrix class.

- class **SubMatrixAliasConstant**

  Accesses just a portion of a matrix.

- class **SubMatrixAlias**

  Accesses just a portion of a matrix (and provides write access).

- class **SquareMatrixAliasConstant**

  A read-only **SquareMatrixAlias**(p. 202) *class.*

- class **SquareMatrixAlias**

  A SquareMatrixAlias class.

- class **SquareMatrix**

The standard SquareMatrix class.

- class **RowVectorAliasConstant**

  A read-only **RowVectorAlias**(p. 152) *class.*

- class **RowVectorAlias**

  A RowVectorAlias class.

- class **RowVector**

  The standard RowVector class.

- class **ColumnVectorAliasConstant**

  A read-only **ColumnVectorAlias**(p. 20) *class.*

- class **ColumnVectorAlias**

  A ColumnVectorAlias class.

- class **ColumnVector**

  The standard ColumnVector class.

7.2 Desktop/ltn100/Shared/MatrixClassLib/code/include/matrix_access.h File Reference

```
#include <iostream.h>
```

Classes

- class **MatrixReadAccess**

  Read access for matrix classes.

- class **SubMatrixReadAccess**

  Read access for sub-matrix classes.

- class **MatrixWriteAccess**

  Write access for matrix classes.

- class **SubMatrixWriteAccess**

  Write access for sub-matrix classes.

7.3 Desktop/ltn100/Shared/MatrixClassLib/code/include/matrix_container.h File Reference

`#include <iostream.h>`

Classes

- class **MatrixContainer**

  Store for primative matrix information.

- class **SubMatrixContainer**

  Store for SubMatrix information.

7.4 Desktop/ltn100/Shared/MatrixClassLib/code/include/matrix_operator.h File Reference

```
#include "matrix_access.h"
#include <iostream.h>
```

Classes

- class **MatrixReadOperator**

  Base class for MatrixReadOperators.

- class **MatrixWriteOperator**

  Base class for MatrixWriteOperators.

- class **SquareMatrixReadOperator**

  Base class for SquareMatrixReadOperators.

- class **SquareMatrixWriteOperator**

  Base class for SquareMatrixWriteOperators.

- class **RowVectorReadOperator**

  Base class for RowVectorReadOperators.

- class **RowVectorWriteOperator**

  Base class for RowVectorWriteOperators.

- class **ColumnVectorReadOperator**

  Base class for ColumnVectorReadOperators.

- class **ColumnVectorWriteOperator**

  Base class for ColumnVectorWriteOperators.

- class **MRO_Element**

  Read matrix element.

- class **MRO_Multiply**

  Multiply matrix.

- class **MRO_ElementMultiply**

  Perform element by element multiplication.

- class **MRO_Add**

  Add matrix.

- class **MRO_Subtract**

  Subtract matrix.

- class **MRO_Negative**

Negative.

- class **MRO_Divide**

  Divide matrix.

- class **MRO_Print**

  Print matrix to screen.

- class **MRO_PrintMatlabFriendly**

  Print matrix to screen (MATLAB Friendly).

- class **MRO_SubMatrixAliasConstant**

  Get a SubMatrix.

- class **MRO_SizeEqual**

  Comapre sizes.

- class **MRO_IsSquareMatrix**

  Is this a square matrix?.

- class **MRO_IsRowVector**

  Is this a row vector?.

- class **MRO_IsColumnVector**

  Is this a column vector?.

- class **MRO_Transpose**

  Returns a transposed matrix.

- class **MRO_Absolute**

  Returns an absolute matrix.

- class **MRO_RowSum**

  Returns a column vector which is the sum of all the rows.

- class **MRO_ColumnSum**

  Returns a row vector which is the sum of all the columns.

- class **MRO_Maximum**

  Returns the largest element.

- class **MRO_Minimum**

  Returns the smallest element.

- class **MRO_InfinityNorm**

  Returns the infinity norm of this matrix.

- class **MRO_SquaredElements**

  Returns a matrix with all the elements the square of this one's.

- class **MWO_Element**

  Writable matrix element.

- class **MWO_EqualsElementCopy**

  Copy elements (From matrix of same size).

- class **MWO_EqualsMemCopy**

  Copy memory directly (From matrix of same size).

- class **MWO_EqualsElementCopyResize**

  Copy elements (Resize if necessesary).

- class **MWO_EqualsMemCopyResize**

  Copy memory directly (Resize if necessesary).

- class **MWO_Reshape**

  Reshape matrix (number of elements has to remain the same).

- class **MWO_Resize**

  Resize matrix (allocate new memory if number of elements changes).

- class **MWO_Zero**

  Zero matrix.

- class **MWO_Set**

  Set all matrix values.

- class **MWO_Randomise**

  Randomise matrix.

- class **MWO_SubMatrixAlias**

  Get a SubMatrix.

- class **SqMRO_DeterminantBasic**

  Assess matrix compataibility (is matrix square?). Calculate determinant of matrix.

- class **SqMRO_DeterminantLUDecomp**

  Calculate determinant of matrix.

- class **SqMRO_Cofactor**

  Calculate the cofactor of an element.

- class **SqMRO_InverseBasic**

  Matrix(p. 44) *inverse.*

- class **SqMRO_Exponential**

  Matrix(p. 44) *exponential.*

- class **SqMRO_InverseLUDecomp**

  Matrix(p. 44) *inverse.*

- class **SqMRO_LUDecomposition**
  Performs LU decomposition of this matrix.
- class **SqMRO_LUBackSubstitution**
  Performs LU back substitution of matrix.
- class **SqMWO_EqualsElementCopyResize**
  Copy elements (Resize if necessesary).
- class **SqMWO_Identity**
  Makes this matrix the identity matrix.
- class **SqMWO_DirectionCosine**
  Makes this matrix a direction cosine matrix.
- class **RVRO_CrossProduct**
  Returns the cross product of the vector and its operand.
- class **RVRO_DotProduct**
  Returns the dot product of the vector and its operand.
- class **RVRO_Modulus**
  Returns modulus of this vector.
- class **RVWO_EqualsElementCopyResize**
  Copy elements (Resize if necessesary).
- class **CVRO_CrossProduct**
  Returns the cross product of the vector and its operand.
- class **CVRO_DotProduct**
  Returns the dot product of the vector and its operand.
- class **CVRO_Modulus**
  Returns modulus of this vector.
- class **CVWO_EqualsElementCopyResize**
  Copy elements (Resize if necessesary).

7.5 Desktop/ltn100/Shared/MatrixClassLib/code/src/matrix.cpp File Reference

```
#include "../include/matrix.h"
#include "../include/matrix_container.h"
#include "../include/matrix_access.h"
#include "../include/matrix_operator.h"
```

7.6 Desktop/ltn100/Shared/MatrixClassLib/code/src/matrix_access.cpp File Reference

```
#include "../include/matrix_access.h"
#include "../include/matrix_container.h"
#include "../include/matrix.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
```

7.7 Desktop/ltn100/Shared/MatrixClassLib/code/src/matrix_operator.cpp File Reference

`#include "../include/matrix_operator.h"`

`#include "../include/matrix.h"`

`#include <stdio.h>`

`#include <stdlib.h>`

`#include <string.h>`

`#include <math.h>`

Functions

- double **_rand** ()

7.7.1 Function Documentation

7.7.1.1 double _rand ()

Definition at line 548 of file matrix_operator.cpp.

```
{
        int big_num=100000;

        return (double)random(big_num) / (double)big_num;
}
```

Chapter 8

MatrixClassLib Page Documentation

8.1 Todo List

page The MatrixClass Library(p. 1) **DONE** Add rest of operators

Create exception class and replace error() functions.

change return type for write operators (to return ∗this)

Test constructors fully

Test copy constructors fully.

Test assignment operators fully

Test SubMatrix class fully

Test/fix luDecomp and related functons (and cofactor())

Index