LinearC++

0.0.1

Generated by Doxygen 1.5.7

Wed Apr 14 23:15:04 2010

Contents

Chapter 1

LinearC++ Documentation

1.1 Introduction

This is the reference documentation for LinearC++, a very basic library for doing Linear Algebra using C++.

Chapter 2

Class Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Class Documentation

5.1 ColumnVector Class Reference

The ColumnVector class, which inherits from the general Matrix class.

`#include <Matrix.h>`

Inherits Matrix.

Collaboration diagram for ColumnVector:

Public Member Functions

- ColumnVector (int r)

  Creates a ColumnVector of the specified length.

- ColumnVector (std::vector< double > &a)

  Creates a ColumnVector from a vector of doubles.

- ColumnVector (double *a, int size)

  Creates a ColumnVector from an array of doubles, with the size of the array as the second argument.

- double length ()

  Calculates and returns the length of the vector.

- void appendCol ()

  This method is undefined for a ColumnVector since a ColumnVector can only have one column.

5.1.1 Detailed Description

The ColumnVector class, which inherits from the general Matrix class.

If you want to do vector-specific operations like get the length of the vector, use this class instead of the Matrix class.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 ColumnVector::ColumnVector (int *r*) [inline]

Creates a ColumnVector of the specified length.

5.1.2.2 ColumnVector::ColumnVector (std::vector< double > & *a*)

Creates a ColumnVector from a vector of doubles.

5.1.2.3 ColumnVector::ColumnVector (double * *a*, int *size*)

Creates a ColumnVector from an array of doubles, with the size of the array as the second argument.

5.1.3 Member Function Documentation

5.1.3.1 void ColumnVector::appendCol ()

This method is undefined for a ColumnVector since a ColumnVector can only have one column.

5.1.3.2 double ColumnVector::length ()

Calculates and returns the length of the vector.

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

- Matrix.h
- Matrix.cpp

5.2 Matrix Class Reference

The general Matrix class.

`#include <Matrix.h>`

Inherited by ColumnVector, and RowVector.

Public Member Functions

- Matrix (int r, int c)

  Creates a zero matrix of the given size.

- int rows ()

  Returns the number of rows in the matrix.

- int cols ()

  Returns the number of columns in the matrix.

- Matrix (std::vector< std::vector< double > > &a)

  Creates a Matrix object from a 2d vector of doubles.

- Matrix & populateRandom ()

  Populates the matrix with random integers in the range 0-9.

- Matrix & populateIdentity ()

  Makes this matrix an identity matrix.

- Matrix & transpose ()

  Returns a Matrix object that is the transpose of the current Matrix.

- double & operator() (int i, int j)

  This allows you to access the elements in the matrix using subscripts.

- virtual void appendRow (Matrix &b)

  Adds a row to the current Matrix object.

- virtual void appendRow (double *r, int size)

  Same as appendRow, but takes an array as the first argument and the length of the array as the second argument.

- virtual void appendRow (std::vector< double > &r)

  Same as appendRow, but takes a vector.

- virtual void appendCol (Matrix &b)

  Adds a column to the current Matrix object.

- virtual void appendCol (double *r, int size)

  Same as appendCol, but takes an array as the first argument and the length of the array as the second argument.

- virtual void appendCol (std::vector< double > &r)

  Same as appendCol, but takes a vector.

- void swapRows (int rowA, int rowB)

  Swaps the given rows in the Matrix.

- void swapCols (int colA, int colB)

  Swaps the given cols in the Matrix.

Protected Attributes

- std::vector< std::vector< double > > data

  Here's where the matrix is actually stored.

5.2.1 Detailed Description

The general Matrix class.

Defines several methods to create and populate matrices. Each element in a matrix is a double.

If you really want a vector instead, use RowVector or ColumnVector. This allows for additional operations such as length().

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Matrix::Matrix (int *r*, int *c*)

Creates a zero matrix of the given size.

Will throw an error if you try to create a matrix of negative dimensions.

5.2.2.2 Matrix::Matrix (std::vector< std::vector< double > > & *a*)

Creates a Matrix object from a 2d vector of doubles.

5.2.3 Member Function Documentation

5.2.3.1 void Matrix::appendCol (std::vector< double > & *r*) [virtual]

Same as appendCol, but takes a vector.

5.2.3.2 void Matrix::appendCol (double * *r*, int *size*) [virtual]

Same as appendCol, but takes an array as the first argument and the length of the array as the second argument.

5.2.3.3 void Matrix::appendCol (Matrix & *b*) [virtual]

Adds a column to the current Matrix object.

The new column must be the same length as all the other column in the matrix.

5.2.3.4 void Matrix::appendRow (std::vector< double > & *r*) [virtual]

Same as appendRow, but takes a vector.

5.2.3.5 void Matrix::appendRow (double * *r*, int *size*) [virtual]

Same as appendRow, but takes an array as the first argument and the length of the array as the second argument.

5.2.3.6 void Matrix::appendRow (Matrix & *b*) [virtual]

Adds a row to the current Matrix object.

The new row must be the same length as all the other rows in the matrix.

5.2.3.7 int Matrix::cols ()

Returns the number of columns in the matrix.

5.2.3.8 double & Matrix::operator() (int *i*, int *j*)

This allows you to access the elements in the matrix using subscripts.

Example: Matrix m(10,10); // create a 10x10 Matrix m(0,0) = 10; // set it's first element to 10 cout << m(9,9) << endl; // print it's last element

This is similar to MATLAB's notation for matrices, but our matrices are zero-indexed.

5.2.3.9 Matrix & Matrix::populateIdentity ()

Makes this matrix an identity matrix.

Throws an error if this is not a square matrix.

5.2.3.10 Matrix & Matrix::populateRandom ()

Populates the matrix with random integers in the range 0-9.

5.2.3.11 int Matrix::rows ()

Returns the number of rows in the matrix.

5.2.3.12 void Matrix::swapCols (int *colA*, int *colB*)

Swaps the given cols in the Matrix.

5.2.3.13 void Matrix::swapRows (int *rowA*, int *rowB*)

Swaps the given rows in the Matrix.

5.2.3.14 Matrix & Matrix::transpose ()

Returns a Matrix object that is the transpose of the current Matrix.

5.2.4 Member Data Documentation

5.2.4.1 std::vector<std::vector<double> > Matrix::data [protected]

Here's where the matrix is actually stored.

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

- Matrix.h
- Matrix.cpp

5.3 RowVector Class Reference

The RowVector class, which inherits from the general Matrix class.

`#include <Matrix.h>`

Inherits Matrix.

Collaboration diagram for RowVector:

Public Member Functions

- RowVector (int c)

  Creates a RowVector of the specified length.

- RowVector (std::vector< double > &a)

  Creates a RowVector from a vector of doubles.

- RowVector (double *a, int size)

  Creates a RowVector from an array of doubles, with the size of the array as the second argument.

- double length ()

  Calculates and returns the length of the vector.

- void appendRow ()

  This method is undefined for a RowVector since a RowVector can only have one row.

5.3.1 Detailed Description

The RowVector class, which inherits from the general Matrix class.

If you want to do vector-specific operations like get the length of the vector, use this class instead of the Matrix class.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 RowVector::RowVector (int *c*) [inline]

Creates a RowVector of the specified length.

5.3.2.2 RowVector::RowVector (std::vector< double > & *a*)

Creates a RowVector from a vector of doubles.

5.3.2.3 RowVector::RowVector (double * *a*, int *size*)

Creates a RowVector from an array of doubles, with the size of the array as the second argument.

5.3.3 Member Function Documentation

5.3.3.1 void RowVector::appendRow ()

This method is undefined for a RowVector since a RowVector can only have one row.

5.3.3.2 double RowVector::length ()

Calculates and returns the length of the vector.

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

- Matrix.h
- Matrix.cpp

Chapter 6

File Documentation

6.1 Matrix.cpp File Reference

```
#include <iostream>
#include <ctime>
#include <cstdlib>
#include <vector>
#include <cassert>
#include <cmath>
#include "Matrix.h"
```

Include dependency graph for Matrix.cpp:

Functions

- Matrix & operator∗ (Matrix &a, Matrix &b)

  Multiplies two matrices and returns the resulting Matrix object.

- Matrix & operator∗ (double s, Matrix &a)

  Multiplies a Matrix object by a scalar (double) and returns the resulting Matrix object.

- Matrix & operator∗ (Matrix &a, double s)

  Multiplies a Matrix object by a scalar (double) and returns the resulting Matrix object.

- Matrix & operator+ (Matrix &a, Matrix &b)

  Adds two Matrix objects elementwise and returns the resulting Matrix object.

- Matrix & operator- (Matrix &a, Matrix &b)

  Subtracts the second matrix from the first matrix returns the resulting Matrix object.

- ostream & operator<< (ostream &s, Matrix &m)
- bool operator== (Matrix &a, Matrix &b)

  Tests if two Matrix objects are equivalent.

6.1.1 Function Documentation

6.1.1.1 Matrix & operator∗ (Matrix & *a*, double *s*)

Multiplies a Matrix object by a scalar (double) and returns the resulting Matrix object.

6.1.1.2 Matrix & operator∗ (double *s*, Matrix & *a*)

Multiplies a Matrix object by a scalar (double) and returns the resulting Matrix object.

6.1.1.3 Matrix & operator∗ (Matrix & *a*, Matrix & *b*)

Multiplies two matrices and returns the resulting Matrix object.

6.1.1.4 Matrix & operator+ (Matrix & *a*, Matrix & *b*)

Adds two Matrix objects elementwise and returns the resulting Matrix object.

6.1.1.5 Matrix & operator- (Matrix & *a*, Matrix & *b*)

Subtracts the second matrix from the first matrix returns the resulting Matrix object.

6.1.1.6 ostream& operator<< (ostream & *s*, Matrix & *m*)

6.1.1.7 bool operator== (Matrix & *a*, Matrix & *b*)

Tests if two Matrix objects are equivalent.

6.2 Matrix.h File Reference

This file contains the declarations of the basic Linear Algebra classes and some methods that operate on them.

```
#include <iostream>
```

```
#include <vector>
```

Include dependency graph for Matrix.h:

This graph shows which files directly or indirectly include this file:

Classes

- class Matrix

  The general Matrix class.

- class RowVector

  The RowVector class, which inherits from the general Matrix class.

- class ColumnVector

  The ColumnVector class, which inherits from the general Matrix class.

Functions

- Matrix & operator∗ (Matrix &a, Matrix &b)

  Multiplies two matrices and returns the resulting Matrix object.

- Matrix & operator∗ (double s, Matrix &a)

  Multiplies a Matrix object by a scalar (double) and returns the resulting Matrix object.

- Matrix & operator∗ (Matrix &a, double s)

  Multiplies a Matrix object by a scalar (double) and returns the resulting Matrix object.

- Matrix & operator+ (Matrix &a, Matrix &b)

  Adds two Matrix objects elementwise and returns the resulting Matrix object.

- Matrix & operator- (Matrix &a, Matrix &b)

  Subtracts the second matrix from the first matrix returns the resulting Matrix object.

- std::ostream & operator<< (std::ostream &s, Matrix &m)

  Prints out a Matrix object.

- bool operator== (Matrix &a, Matrix &b)

  Tests if two Matrix objects are equivalent.

6.2.1 Detailed Description

This file contains the declarations of the basic Linear Algebra classes and some methods that operate on them.

6.2.2 Function Documentation

6.2.2.1 Matrix& operator∗ (Matrix & *a*, double *s*)

Multiplies a Matrix object by a scalar (double) and returns the resulting Matrix object.

6.2.2.2 Matrix& operator∗ (double *s*, Matrix & *a*)

Multiplies a Matrix object by a scalar (double) and returns the resulting Matrix object.

6.2.2.3 Matrix& operator∗ (Matrix & *a*, Matrix & *b*)

Multiplies two matrices and returns the resulting Matrix object.

6.2.2.4 Matrix& operator+ (Matrix & *a*, Matrix & *b*)

Adds two Matrix objects elementwise and returns the resulting Matrix object.

6.2.2.5 Matrix& operator- (Matrix & *a*, Matrix & *b*)

Subtracts the second matrix from the first matrix returns the resulting Matrix object.

6.2.2.6 std::ostream & operator<< (std::ostream & *s*, Matrix & *m*)

Prints out a Matrix object.

6.2.2.7 bool operator== (Matrix & *a*, Matrix & *b*)

Tests if two Matrix objects are equivalent.

6.3 MatrixFunctions.cpp File Reference

```
#include "MatrixFunctions.h"
#include "Matrix.h"
```

Include dependency graph for MatrixFunctions.cpp:

Functions

- int findNonZero (Matrix &m, int col)

  Given a Matrix and a column number, this functions returns the index of the first row which has a non-zero value in that column.

- ColumnVector & gaussJordan (Matrix &A, Matrix &b)

  Given two matrices A and b, solves the linear system of equations assuming the form Ax = b.

- ColumnVector & gaussianElimination (Matrix &A, Matrix &b)

  Given two matrices A and b, solves the linear system of equations assuming the form Ax = b.

6.3.1 Function Documentation

6.3.1.1 int findNonZero (Matrix & *m*, int *col*)

Given a Matrix and a column number, this functions returns the index of the first row which has a non-zero value in that column.

If all rows have zeroes, it returns -1. This function is used by both gaussJordan and gaussianElimination.

6.3.1.2 ColumnVector & gaussianElimination (Matrix & *A*, Matrix & *b*)

Given two matrices A and b, solves the linear system of equations assuming the form Ax = b.

Uses Gaussian Elimination with backsubstitution. Returns a ColumnVector object containing the solution.

6.3.1.3 ColumnVector & gaussJordan (Matrix & *A*, Matrix & *b*)

Given two matrices A and b, solves the linear system of equations assuming the form Ax = b.

Uses Gauss-Jordan. Returns a ColumnVector object containing the solution.

6.4 MatrixFunctions.h File Reference

This file contains the declarations of the basic Linear Algebra classes and some methods that operate on them.

```
#include <iostream>
#include <vector>
#include "Matrix.h"
```

Include dependency graph for MatrixFunctions.h:

This graph shows which files directly or indirectly include this file:

Functions

- int findNonZero (Matrix &m, int col)

  Given a Matrix and a column number, this functions returns the index of the first row which has a non-zero value in that column.

- ColumnVector & gaussJordan (Matrix &A, Matrix &b)

  Given two matrices A and b, solves the linear system of equations assuming the form Ax = b.

- ColumnVector & gaussianElimination (Matrix &A, Matrix &b)

  Given two matrices A and b, solves the linear system of equations assuming the form Ax = b.

6.4.1 Detailed Description

This file contains the declarations of the basic Linear Algebra classes and some methods that operate on them.

6.4.2 Function Documentation

6.4.2.1 int findNonZero (Matrix & *m*, int *col*)

Given a Matrix and a column number, this functions returns the index of the first row which has a non-zero value in that column.

If all rows have zeroes, it returns -1. This function is used by both gaussJordan and gaussianElimination.

6.4.2.2 ColumnVector& gaussianElimination (Matrix & *A*, Matrix & *b*)

Given two matrices A and b, solves the linear system of equations assuming the form Ax = b.

Uses Gaussian Elimination with backsubstitution. Returns a ColumnVector object containing the solution.

6.4.2.3 ColumnVector& gaussJordan (Matrix & *A*, Matrix & *b*)

Given two matrices A and b, solves the linear system of equations assuming the form Ax = b.

Uses Gauss-Jordan. Returns a ColumnVector object containing the solution.

6.5 test.cpp File Reference

```
#include <iostream>
#include "Matrix.h"
#include "MatrixFunctions.h"
#include <boost/progress.hpp>
```

Include dependency graph for test.cpp:

Functions

- int main (int argc, char ∗argv[ ])

6.5.1 Function Documentation

6.5.1.1 int main (int *argc*, char ∗ *argv*[ ])

Index