Class TestMatrix

```
java.lang.Object
  └─TestMatrix
```

```
public class TestMatrix
extends java.lang.Object
```

An implementation of basic matrix math which can be used with a library of linear algebra algorithms originally created in Python by Massimo Di Pierro and ported to Java. All code released under BSD licensing.

Version:
0.1

Author:
Ruthann Sudman

See Also:
`LinearAlgebra`, Code Repository

Field Summary

private int	**myCols**
private double[][]	**myData**
private int	**myRows**

Constructor Summary

TestMatrix(int rows, int columns) TestMatrix constructor, initializing the original matrix to all 0.

Method Summary

TestMatrix	**addMatrix**(double x) Add a value to all elements in the TestMatrix.
TestMatrix	**addMatrix**(TestMatrix otherData) Add two TestMatrices together.
void	**changeMe**(int row, int column, double myvalue)

	Updates a specific value in the myData.
double	**condition_number**() Return the condition number of myData.
TestMatrix	**copyMe**() Return a copy of myData.
TestMatrix	**divMatrix**(double x) Divide all elements in a TestMatrix by x.
int	**getColumns**() Get method that returns the number of columns in the TestMatrix.
double	**getMe**(int row, int column) Obtain a specific value in the myData.
int	**getRows**() Get method that returns the number of rows in the TestMatrix.
TestMatrix	**identity**() Construct a diagonal matrix identical in size to myData.
TestMatrix	**invMatrix**() The inverse of a TestMatrix object.
TestMatrix	**mulMatrix**(double x) Multiply all elements in a TestMatrix by a value.
TestMatrix	**mulMatrix**(TestMatrix otherData) Multiply two TestMatrices together.
double	**mulMatrixScalar**(TestMatrix B) Do a scalar multiplication of two matrices.
double	**norm**() Return the norm of myData
void	**printMe**() Print out the myData in a single line.
TestMatrix	**subMatrix**(double x) Subtract a specific value from all elements in the TestMatrix.
TestMatrix	**subMatrix**(TestMatrix otherData) Subtract one TestMatrix from another.
private void	**swapMe**(int r1, int c1, int r2, int c2) Swap two values in myData.
private void	**updateAddMe**(int row, int column, double myvalue) Adds a value to a specific value in the myData.
void	**updateSubMe**(int row, int column, double myvalue) Subtracts a specific value in the myData.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

myRows

```
private int myRows
```

---

myCols

```
private int myCols
```

---

myData

```
private double[][] myData
```

Constructor Detail

TestMatrix

```
public TestMatrix(int rows,
                  int columns)
```

TestMatrix constructor, initializing the original matrix to all 0.

Parameters:
`rows` - Number of rows in the TestMatrix.
`columns` - Number of Columns in the TestMatrix.
Exception(s):
`java.lang.ArrayIndexOutOfBoundsException` - TestMatrix need to have at least 1 row and 1 column.
`java.lang.ArithmeticException` - No known exceptions.

Method Detail

getRows

```
public int getRows()
```

Get method that returns the number of rows in the TestMatrix.

Returns:
Number of rows in the TestMatrix myRows.
Exception(s):
`java.lang.ArithmeticException` - No known exceptions.

---

getColumns

```
public int getColumns()
```

Get method that returns the number of columns in the TestMatrix.

Returns:
The number of columns in the TestMatrix myCols.
Exception(s):
`java.lang.ArithmeticException` - No known exceptions.

---

changeMe

```
public void changeMe(int row,
                     int column,
                     double myvalue)
```

Updates a specific value in the myData.

Parameters:
`row` - The row of the value to update.
`column` - The column of the value to update.
`myvalue` - The update value.
Exception(s):
`java.lang.ArrayIndexOutOfBoundsException` - row and column must be in the bounds of the matrix myData.
`java.lang.ArithmeticException` - No known exceptions.

---

updateAddMe

```
private void updateAddMe(int row,
                         int column,
                         double myvalue)
```

Adds a value to a specific value in the myData.

Parameters:
`row` - The row of the value to add to.
`column` - The column of the value to add to.
`myvalue` - The value to add to the original number.
Exception(s):
`ArrayIndexOutOfBoudsException` - row and column must be in the bounds of the matrix myData.
`java.lang.ArithmeticException` - No known exceptions.

---

updateSubMe

```
public void updateSubMe(int row,
                        int column,
                          double myvalue)
```

Subtracts a specific value in the myData.

Parameters:
`row` - The row of the value to subtract from.
`column` - The column of the value to subtract.
`myvalue` - The value to subtract from the original number.

Exception(s):
`java.lang.ArrayIndexOutOfBoundsException` - row and column must be in the bounds of the matrix myData.
`java.lang.ArithmeticException` - No known exceptions.

---

getMe

```
public double getMe(int row,
                    int column)
```

Obtain a specific value in the myData.

Parameters:
`row` - The row of the desired value.
`column` - The column of the desired value.

Returns:
The desired value to return from myData.

Exception(s):
`java.lang.ArrayIndexOutOfBoundsException` - row and column must be in the bounds of the matrix myData.
`java.lang.ArithmeticException` - No known exceptions

---

printMe

```
public void printMe()
```

Print out the myData in a single line.

Exception(s):
`java.lang.ArrayIndexOutOfBoundsException` - row and column must be in the bounds of the matrix myData.
`java.lang.ArithmeticException` - No known exceptions.
`Other` - Printing does not work well for TestMatrices with column = 1.

---

addMatrix

```
public TestMatrix addMatrix(TestMatrix otherData)
```

Add two TestMatrices together.

Parameters:

`otherData` - The TestMatrix to add to myData.

Returns:

The added TestMatrx.

Exception(s):

`java.lang.ArrayIndexOutOfBoundsException` - The rows and columns of both matrices must be equal.

`java.lang.ArithmeticException` - No known exceptions.

---

addMatrix

```
public TestMatrix addMatrix(double x)
```

Add a value to all elements in the TestMatrix.

Parameters:

`x` - The value to add to all elements of the TestMatrix.

Returns:

The TestMatrix with the addition of x performed.

Exception(s):

`java.lang.ArrayIndexOutOfBoundsException` - Improper index value used for array.

`java.lang.ArithmeticException` - No known exceptions

---

subMatrix

```
public TestMatrix subMatrix(TestMatrix otherData)
```

Subtract one TestMatrix from another.

Parameters:

`otherData` - The matrix to subtract from myData.

Returns:

The subtracted TestMatrix.

Exception(s):

`java.lang.ArrayIndexOutOfBoundsException` - The rows and columns of both matrices must be equal.

`java.lang.ArithmeticException` - No known exceptions

---

subMatrix

```
public TestMatrix subMatrix(double x)
```

Subtract a specific value from all elements in the TestMatrix.

Parameters:

`x` - The value to subtract from all elements in myData.

Returns:

myData with x subtracted.

Exception(s):

`java.lang.ArrayIndexOutOfBoundsException` - Improper index value used for array.
`java.lang.ArithmeticException` - No known exceptions.

---

mulMatrix

`public TestMatrix **mulMatrix**(double x)`

Multiply all elements in a TestMatrix by a value.

Parameters:
`x` - The value to multiply all elements in myData by.
Returns:
myData multiplied by x.
Exception(s):
`java.lang.ArrayIndexOutOfBoundsException` - Improper index value used for array.
`java.lang.ArithmeticException` - No known exceptions.

---

mulMatrixScalar

`public double **mulMatrixScalar**(TestMatrix B)`

Do a scalar multiplication of two matrices.

Parameters:
`B` - The TestMatrix to be multiplied with myData.
Returns:
The scalar multiplication of myData and TestMatrix B.
Exception(s):
`java.lang.ArrayIndexOutOfBoundsException` - Improper index value used for array.
`java.lang.ArithmeticException` - No known exceptions.
`Other` - The number of rows for both TestMatrices must be one and the number of columns must be equal OR the number of columns for both TestMatrices must be one and the number of rows must be equal.

---

mulMatrix

`public TestMatrix **mulMatrix**(TestMatrix otherData)`

Multiply two TestMatrices together.

Parameters:
`otherData` - The TestMatrix to multiply with myData.
Returns:
The multiplication of myData and TestMatrix otherData.
Exception(s):
`java.lang.ArrayIndexOutOfBoundsException` - Improper index value used for array.
`java.lang.ArithmeticException` - No known exceptions.
`Other` - The number of columns in myData must be equal to the number of rows in otherData.

divMatrix

```
public TestMatrix divMatrix(double x)
```

Divide all elements in a TestMatrix by x.

Parameters:
`x` - The value to divide all myData elements by.
Returns:
The muliplication of myData and x.
Exception(s):
`java.lang.ArrayIndexOutOfBoundsException` - Improper index value used for array.
`java.lang.ArithmeticException` - No known exceptions.

---

swapMe

```
private void swapMe(int r1,
                    int c1,
                    int r2,
                    int c2)
```

Swap two values in myData.

Parameters:
`r1` - The row of the first value to swap.
`c1` - The column of the first value to swap.
`r2` - The row of the second value to swap.
`c2` - The column of the second value to swap.
Exception(s):
`java.lang.ArrayIndexOutOfBoundsException` - Improper index value used for array.
`java.lang.ArithmeticException` - No known exceptions.

---

copyMe

```
public TestMatrix copyMe()
```

Return a copy of myData. One cannot simply return myData because that would be returning a double array and not a TestMatrix object.

Returns:
TestMatrix
Exception(s):
`java.lang.ArrayIndexOutOfBoundsException` - Improper index value used for array.
`java.lang.ArithmeticException` - No known exceptions.

---

invMatrix

`public TestMatrix invMatrix()`

The inverse of a TestMatrix object.

Returns:
The inverse of myData.
Exception(s):
`java.lang.ArrayIndexOutOfBoundsException` - Improper index value used for array.
`java.lang.ArithmeticException` - No known exceptions.

---

identity

`public TestMatrix identity()`

Construct a diagonal matrix identical in size to myData.

Returns:
The identity matrix for myData
Exception(s):
`java.lang.ArrayIndexOutOfBoundsException` - Improper index value used for array.
`java.lang.ArithmeticException` - No known exceptions.
`Other` - The size of the identity matrix will be identical to myData. Size is not customizable.

---

norm

`public double norm()`

Return the norm of myData

Returns:
Norm of matrix myData
Exception(s):
`Other` - This function is not properly implemented.

---

condition_number

`public double condition_number()`

Return the condition number of myData.

Returns:
The condition number of myData.
Exception(s):
`Other` - This function is not properly implemented.
