Class RunMe

```
java.lang.Object
  └ RunMe
```

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

Used to demonstrate the functionality of the mathematical library. 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:
TestMatrix, LinearAlgebra, TestFunctionAbstract, TestFunction, TestFunction2, TestFunction3, TestFunction4, TestFunction5, TestFunction6, TestFunction7, Code Repository

Field Summary

private static LinearAlgebra	**LA**
private static TestFunction3	**P**
private static TestFunction4	**Q**
private static java.text.DecimalFormat	**twelveD**
private static java.text.DecimalFormat	**twoD**
private static TestFunction	**Y**
private static TestFunction2	**Z**

Constructor Summary

RunMe()

Method Summary

static void	**main**(java.lang.String[] args) Runs all test methods.
static void	**Test1**() Tests inverse matrix as implemented in class using c++.
static void	**Test10**() Tests optimize bisection for a function extended from TestFunctionAbstract.
static void	**Test11**() Tests optimize newton for a function extended from TestFunctionAbstract.
static void	**Test12**() Tests optimize secant for a function extended from TestFunctionAbstract.
static void	**Test13**() Tests optimize newton stabilized for a function extended from TestFunctionAbstract.
static void	**Test14**() Tests optimize golden search for a function extended from TestFunctionAbstract.
static void	**Test15**() Tests first and second derivatives for a function extended from TestFunctionAbstract.
static void	**Test16**() Tests for basic TestMatrix math functionality.
static void	**Test2**() Tests Cholesky as implemented in test096 from Massimo Ei Pierro's numeric.py.
static void	**Test3**() Tests Markovitz as implemented in the orignal Markovitz by Massimo Di Pierro in numeric.py
static void	**Test35**() Tests the condition number and square root for doubles.
static void	**Test4**() Tests fit least squares for TestFunctionAbstract array of functions.
static void	**Test5**() Tests solve fixed point for a function extended from TestFunctionAbstract.
static void	**Test6**() Tests solve bisection for a function extended from TestFunctionAbstract.
static void	**Test7**() Tests solve solve newton for a function extended from TestFunctionAbstract.
static void	**Test8**() Tests solve secant for a function extended from TestFunctionAbstract.
static void	**Test9**() Tests solve newton stabilized for a function extended from TestFunctionAbstract.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

Y

```
private static TestFunction Y
```

Z

```
private static TestFunction2 Z
```

P

```
private static TestFunction3 P
```

Q

```
private static TestFunction4 Q
```

LA

```
private static LinearAlgebra LA
```

---

twoD

```
private static java.text.DecimalFormat twoD
```

---

twelveD

```
private static java.text.DecimalFormat twelveD
```

Constructor Detail

RunMe

```
public RunMe()
```

Method Detail

Test1

`public static void **Test1**()`

Tests inverse matrix as implemented in class using c++.

Exception(s):

`java.lang.ArithmeticException` - Fails when method is incorrect.

See Also:

`TestMatrix`, `TestMatrix.invMatrix()`, `TestMatrix.mulMatrix(TestMatrix)`

Test2

`public static void **Test2**()`

Tests Cholesky as implemented in test096 from Massimo Ei Pierro's numeric.py.

Exception(s):

`java.lang.ArithmeticException` - Fails when method is incorrect.

See Also:

`TestMatrix`, `LinearAlgebra`, `LinearAlgebra.Cholesky(TestMatrix)`

Test3

`public static void **Test3**()`

Tests Markovitz as implemented in the orignal Markovitz by Massimo Di Pierro in numeric.py

Exception(s):

`java.lang.ArithmeticException` - Fails when method is incorrect.

See Also:

`TestMatrix`, `LinearAlgebra`, `LinearAlgebra.Markovitz(TestMatrix, TestMatrix, double)`, `LinearAlgebra.getMarkovitzPortfolio()`, `LinearAlgebra.getMarkovitzPortfolioReturn()`, `LinearAlgebra.getMarkovitzPortfolioReturn()`

Test35

`public static void **Test35**()`

Tests the condition number and square root for doubles.

Exception(s):

`java.lang.ArithmeticException` - Fails when method is incorrect. The condition number for test matrix is not implemented.

See Also:

`TestMatrix`, `TestMatrix.condition_number()`

Test4

```
public static void Test4()
```

Tests fit least squares for TestFunctionAbstract array of functions.

Exception(s):

`java.lang.ArithmeticException` - Not yet implemented.

See Also:

`TestFunctionAbstract`, `TestFunctionAbstract.fit_least_squares()`

Test5

```
public static void Test5()
```

Tests solve fixed point for a function extended from TestFunctionAbstract.

Exception(s):

`java.lang.ArithmeticException` - Fails when method is incorrect.

See Also:

`TestFunctionAbstract`, `TestFunctionAbstract.solve_fixed_point(double)`, `TestFunction3`

Test6

```
public static void Test6()
```

Tests solve bisection for a function extended from TestFunctionAbstract.

Exception(s):

`java.lang.ArithmeticException` - Fails when method is incorrect.

See Also:

`TestFunctionAbstract`, `TestFunctionAbstract.solve_bisection(double, double)`, `TestFunction4`

Test7

```
public static void Test7()
```

Tests solve solve newton for a function extended from TestFunctionAbstract.

Exception(s):

`java.lang.ArithmeticException` - Fails when method is incorrect.

See Also:

`TestFunctionAbstract`, `TestFunctionAbstract.solve_newton(double)`, `TestFunction4`

Test8

`public static void Test8()`

Tests solve secant for a function extended from TestFunctionAbstract.

Exception(s):

`java.lang.ArithmeticException` - Fails when method is incorrect.

See Also:

`TestFunctionAbstract`, `TestFunctionAbstract.solve_secant(double)`, `TestFunction4`

Test9

`public static void Test9()`

Tests solve newton stabilized for a function extended from TestFunctionAbstract.

Exception(s):

`java.lang.ArithmeticException` - Fails when method is incorrect.

See Also:

`TestFunctionAbstract`, `TestFunctionAbstract.solve_newton_stabilized(double, double)`, `TestFunction4`

Test10

`public static void Test10()`

Tests optimize bisection for a function extended from TestFunctionAbstract.

Exception(s):

`java.lang.ArithmeticException` - Fails when method is incorrect.

See Also:

`TestFunctionAbstract`, `TestFunctionAbstract.optimize_bisection(double, double)`, `TestFunction4`

Test11

`public static void Test11()`

Tests optimize newton for a function extended from TestFunctionAbstract.

Exception(s):

`java.lang.ArithmeticException` - Fails when method is incorrect.

See Also:

`TestFunctionAbstract`, `TestFunctionAbstract.optimize_newton(double)`, `TestFunction4`

Test12

`public static void Test12()`

Tests optimize secant for a function extended from TestFunctionAbstract.

Exception(s):

`java.lang.ArithmeticException` - Fails when method is incorrect.

See Also:

TestFunctionAbstract, TestFunctionAbstract.optimize_secant(double), TestFunction4

Test13

`public static void` **`Test13`**`()`

Tests optimize newton stabilized for a function extended from TestFunctionAbstract.

Exception(s):

`java.lang.ArithmeticException` - Fails when method is incorrect.

See Also:

TestFunctionAbstract, TestFunctionAbstract.optimize_newton_stabilized(double, double), TestFunction4

Test14

`public static void` **`Test14`**`()`

Tests optimize golden search for a function extended from TestFunctionAbstract.

Exception(s):

`java.lang.ArithmeticException` - Fails when method is incorrect.

See Also:

TestFunctionAbstract, TestFunctionAbstract.optimize_golden_search(double, double), TestFunction4

Test15

`public static void` **`Test15`**`()`

Tests first and second derivatives for a function extended from TestFunctionAbstract.

Exception(s):

`java.lang.ArithmeticException` - Fails when method is incorrect.

See Also:

TestFunctionAbstract, TestFunctionAbstract.f(double), TestFunctionAbstract.Df(double), TestFunctionAbstract.DDf(double), TestFunction2

Test16

`public static void` **`Test16`**`()`

Tests for basic TestMatrix math functionality.

Exception(s):

`java.lang.ArithmeticException` - Fails when method is incorrect.

See Also:

`TestMatrix`, `TestMatrix.addMatrix(double)`, `TestMatrix.addMatrix(TestMatrix)`, `TestMatrix.changeMe(int, int, double)`, `TestMatrix.condition_number()`, `TestMatrix.copyMe()`, `TestMatrix.divMatrix(double)`, `TestMatrix.invMatrix()`, `TestMatrix.mulMatrix(double)`, `TestMatrix.mulMatrix(TestMatrix)`, `TestMatrix.mulMatrixScalar(TestMatrix)`, `TestMatrix.printMe()`, `TestMatrix.subMatrix(double)`, `TestMatrix.subMatrix(TestMatrix)`

---

main

```
public static void main(java.lang.String[] args)
```

Runs all test methods.

Parameters:

`args` - Default for Java.

Exception(s):

`java.lang.ArithmeticException` - Fails for incorrect methods.
