Read Me Document for Matrix.vb file (VB.NET 2005 Matrix Class)

Written by: Ryan Brazeal
Date: February, 2008
Version 1.2

This document provides instructions on how to implement and use this VB.NET Matrix Class within a project. Specific syntax examples are given for all the available properties and methods of the class.

To include this Matrix class as part of your project you simply need to add the Matrix.vb file to your project (add existing item…)

DECLARATION AND INSTANTIATION

Declaring a Matrix object within your project is as simple as:

```
Dim A As Matrix    or    Dim B,C As Matrix
```

Instantiating a Matrix object is then handled by 1 of 2 class constructors

Constructor 1 – creates a 1x1 matrix with a value = 0, `A = New Matrix()`
Constructor 2 – creates an MxN matrix with all values = 0, `B = New Matrix(6,8)`

Inline declaration and instantiation can be performed at the same time of course by:

```
Dim A As New Matrix()    or    Dim B As New Matrix(6,8)
```

MATRIX PROPERTIES

Writing and Reading Matrix Elements

To write (set) or read (get) individual matrix element values you use the .data(row, column) property. This property has been designed to be base 1 instead of base 0 (base 0 is how all VB.NET arrays are set). This means that the matrix element situated at row 1, column 1 is accessed with the .data(1,1) property and NOT the .data(0,0) property. An **ERROR** will occur if you try and get or set an element out of range, a message will be displayed and then your application will terminate!

Ex. Set → `A.data(3, 6) = 3`
Get → `myDecimal = B.data(4, 5)`

Number of Rows and Columns in a Matrix

There are 2 other properties associated with this Matrix Class that simply read (get) the number of rows the matrix has .nRows() and the number of columns the matrix has .nCols(). These are read-only properties and hence you cannot manually change the size of the matrix.

Ex.
```
numNewRowsDecimal = C.nRows()
numNewColumnsDecimal = C.nCols()
```

MATRIX OPERATIONS AND METHODS

Scalar Multiplication and Division

If you want to multiply or divide a Matrix by a scalar term you can simply use the * or / operator as you normally would to multiply or divide any two numbers. The order of the terms is important. For multiplication the scalar must be listed BEFORE the Matrix object. For division the scalar must be listed AFTER the Matrix object An **ERROR** will occur if you try and divide by zero (0), a message will be displayed and then your application will terminate!

Ex.
```
A = 2.6 * A
B = B / 3.5
```

Matrix Addition and Subtraction

If you want to add or subtract 2 Matrices you can simply use the + or – operators as you normally would to add or subtract any two numbers. An **ERROR** will occur if you try and add or subtract matrices of different sizes, a message will be displayed and then your application will terminate!

Ex.
```
A = A – B
B = A + C
```

Matrix Multiplication

If you want to multiply 2 Matrices together you can simply use the * operator as you normally would to multiply any two numbers. An **ERROR** will occur if you try and multiply matrices of non-compatible sizes (ie. # columns in Matrix 1 does NOT equal # rows in Matrix 2), a message will be displayed and then your application will terminate!

Ex.
```
C = A * B
```

Matrix Inversion

If you want to compute the inverse of a Matrix you can use the .inverse() method of this Matrix Class. This inverse method uses Gaussian elimination to compute the inverse of any square matrix. An **ERROR** will occur if you try and inverse a matrix that is not square or a matrix is singular (ie. determinant = 0), a message will be displayed and then your application will terminate!

Ex.
```
B = A.Inverse()
```

Matrix Transpose

If you want to transpose a Matrix you can use the .transpose() method of this Matrix Class.

Ex.
```
A = B.Transpose()
```

Identity Matrix

If you want to create an Identity matrix you can use the .makeIdentity() method of this Matrix Class. A matrix of the desired size must already exist, the easiest way to do this is to simply instantiate a new matrix with the desired size. Then you can use the .makeIdentity() method to create the desired identity matrix. Notice this works with all matrices not only square matrices. In the case of a non-square matrix this method will produce a pseudo identity matrix.

Ex.
```
Dim B As New Matrix(4, 4)
B = B.makeIdentity()
```

Clearing (Resetting) a Matrix

If you want to clear all the elements within a matrix and set all the values to = 0 you can use the .clear() method of this Matrix Class.

Ex.
```
C = C.clear()
```

Printing (Viewing) a Matrix

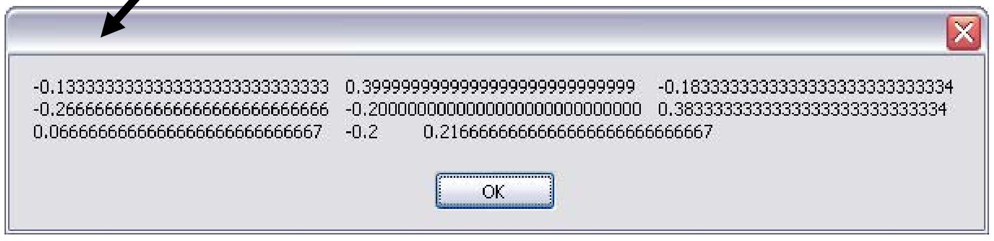
If at any point within your application you want to view all the elements of a matrix you can use the .printAll([matrixName],[formatOutput],[decimals]) method of this Matrix Class. This will help with debugging your application. There are 3 optional arguments to this method.

matrixName is a string value that is used to display a Name for the matrix inside the Title of a message box

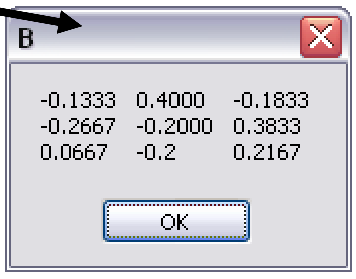
formatOutput is a Boolean value = True or False Only
- if you want to set the number of decimal places, set = TRUE
- if you want to see ALL decimal places, leave it empty or set = FALSE

decimals is an Integer value which is used only if formatOutput = TRUE and it specifies the number of decimal places to show for each matrix element (if applicable)

Ex.
```
A.printAll()
A.printAll("A")
B.printAll(,True, 4)
B.printAll("B",True, 4)
```



```
-0.1333333333333333333333333333  0.3999999999999999999999999999  -0.1833333333333333333333333334
-0.2666666666666666666666666666  -0.2000000000000000000000000000  0.3833333333333333333333333334
0.0666666666666666666666666667  -0.2       0.2166666666666666666666666667
```



```
B

-0.1333  0.4000   -0.1833
-0.2667  -0.2000  0.3833
0.0667   -0.2     0.2167
```

Convert 1x1 matrix to a scalar

If you have a 1x1 matrix and want to use it as a scalar for multiplication or division you can use the .toScalar() method of this Matrix Class. An **ERROR** will occur if you try and convert a matrix that is not 1 row by 1 column in size, a message will be displayed and then your application will terminate!

Ex.
```
Dim apriori As Decimal
apriori = A.toScalar()
```

Get Diagonal Elements of a square matrix

If you want to get all the diagonal elements of a square matrix you can use the .getDiagonal() method of this Matrix Class. The matrix returned will be a single column matrix with the same number of rows as the original square matrix. An **ERROR** will occur if you try and get the diagonal elements of a matrix that is not square, a message will be displayed and then your application will terminate!

Ex.
```
Variances = Cx.getDiagonal()
```

Solve the square root of all the Elements of a matrix

If you want to solve for the square root of all the elements of a matrix you can use the .Sqrt() method of this Matrix Class. The matrix returned will be a matrix with the same size as the original matrix. If an element is negative (less than zero) the absolute value of the element will be used to solve the square root.

Ex.
```
StdDevs = Variances.Sqrt()
```

Re-dimension (re-size) a Matrix

If you want to make an existing matrix a different size (ie. change the number of rows and/or columns) you can use the .matrixReDim(rows, cols, [Preserve]) method of this Matrix Class. There are 2 required and 1 optional arguments for this method.

rows is an integer value that is used to specify the NEW number of rows of the current matrix

cols is an integer value that is used to specify the NEW number of columns of the current matrix

Preserve is a Boolean value that is used to specify if the existing data that is in the current matrix is to be kept or if the re-dimensioned matrix data is to be cleared.

- Preserve = False (is the default value used if the Preserved argument is not specified), will clear all the data (all reset to zero) in the re-dimensioned matrix

- Preserve = True, will retain the existing data that is within the new row/column sizes of the re-dimensioned matrix. If the new row/column size is larger than the existing row/column size the re-dimensioned matrix will contain data values equal to zero for those new individual elements. If the new row/column size is smaller than the existing row/column size the re-dimensioned matrix will contain data values equal to the original matrix that are less than or equal to new row/column size.

Ex. `newMatrix = oldMatrix.matrixReDim(4,4,True)`

Get an individual Row of a Matrix

If you want to get all the elements of a matrix that are in a particular row you can use the .getRow(row) method of this Matrix Class. The matrix returned will be a matrix with the size of 1 row by the number of columns equal to the number of columns of the original matrix. There is 1 required argument for this method. An **ERROR** will occur if you try and get a row of a matrix that does not exist, a message will be displayed and then your application will terminate!

row is an integer value that is used to specify the row number of a matrix that is desired

Ex. `row1 = observationMatrix.getRow(4)`

Get an individual Column of a Matrix

If you want to get all the elements of a matrix that are in a particular column you can use the .getColumn(column) method of this Matrix Class. The matrix returned will be a matrix with the size of number of rows equal to the number of rows of the original matrix by 1 column. There is 1 required argument for this method. An **ERROR** will occur if you try and get a column of a matrix that does not exist, a message will be displayed and then your application will terminate!

column is an integer value that is used to specify the column number of a matrix that is desired

Ex. `column1 = observationMatrix.getColumn(3)`