Homework Assignment 009 (2020)

Matrix and Vector Algebra

Create a Win application project named as < student ID><your name>Ass009. Remember to rename your Form1.cs file to having a meaningful class name. The form class is used as a user interface to test the functionalities of your matrix and vector class.

Define a utility class (namely, e.g., MatrixAndVector) for operations of matrix and vector algebra. Provide the following functions:

`static public string VectorString( double[] vector)`: returns a string for displaying the vector.

`static public string MatrixString( double[,] matrix)`: returns a string for displaying the matrix, rows are separated with escape character \n.

`static public double[] ToVector( string vectorString )`: returns a double array from a string, where elements are separated with blank space character.

`static public double[,] ToMatrix( string matrixString )`: returns a two dimensional array from a string where rows of the matrix are separated with the escape character \n.

`static public double[,] ToMatrix( string[] rowStrings )`: returns a two dimensional array from an array of string, where each string is a row vector with elements separated with blank space characters.

`static public double Norm( double[] vector )`: calculates the norm of a given vector.

`static public double[] Addition( double[] v1, double[] v2 )`: returns the addition of two vectors

`static public double[] Subtraction( double[] v1, double[] v2 )`: calculates the subtraction of two vectors.

`static public double InnerProduct( double[] v1, double[] v2 )`: calculates the inner product of two vectors.

`static public double[] UnitVector( double[] vector )`: returns the unit vector of a given vector.

`static public double[,] Addition( double[,] m1, double[,] m2 )`: calculates the addition of two matrices in the same dimension.

`static public double[,] Subtraction( double[,] m1, double[,] m2 )`:

calculates the subtraction of two matrices in the same dimension.

`static public double[,]Multiplication( double[,] m1, double[,] m2 )`: calculates the product of two matrices that share the same inner dimension. For example $A_{(m\times n)}\cdot B_{(n\times p)}=C_{(m\times p)}$. Mathematically, $C_{(m\times p)}=\lfloor c_{i,j}\rfloor_{(m\times p)}\ c_{i,j}=\sum_{k=1}^{n}a_{i,k}\cdot b_{k,j}, i=1,2,\cdots,m; j=1,2,\cdots,p.$

`static public double[,] TransposedMatrix( double[,] matrix )`: returns the transpose matrix from a given matrix.

`static public double Determinant( double[,] matrix )`: calculates the determinant (行列式) of a square matrix. (The dimension should be smaller or equal to 3, for now.)

`static public double[] MatrixTimeVector ( double[,] matrix, double[] columnVector)`: returns the resulting vector from a multiplication of a matrix and a (column) vector. Note that the column dimension of the matrix must be equal to the dimension of the vector. Mathematically, $A_{(m\times n)}\cdot\mathbf{v}_{(n\times 1)}=\mathbf{u}_{(m\times 1)}; A_{(m\times n)}=\lfloor a_{i,j}\rfloor_{(m\times n)};\ \mathbf{v}=\begin{bmatrix}v_1\\v_2\\\vdots\\v_n\end{bmatrix},\quad \mathbf{u}=\begin{bmatrix}u_1\\u_2\\\vdots\\u_m\end{bmatrix}$, and then

$$u_i=\sum_{k=1}^{n}a_{i,k}\cdot v_k, i=1,2,\cdots,m$$

`static public double[] VectorTimeMatrix(double[] rowVector, double[,] matrix)`: returns the resulting vector of a multiplication of a (row) vector and a matrix. Note that the row dimension of the matrix must be equal to the dimension of the vector.

Mathematically, $\mathbf{v}_{(1\times m)}A_{(m\times n)}=\mathbf{u}_{(1\times n)}; A_{(m\times n)}=\lfloor a_{i,j}\rfloor_{(m\times n)};\ \mathbf{v}=[v_1v_2\cdots v_m],\quad \mathbf{u}=[u_1u_2\cdots u_n]$, and then

$$u_j=\sum_{i=1}^{m}v_i\cdot a_{i,j}, j=1,2,\cdots,n$$

In your form class, provide input UIs, command UIs, and display UIs for the user to specify values of vectors and matrices to test each function of your vector & matrix class. To facilitate the functionality test, you can define data fields of vectors and matrices and hard code them in your form class.

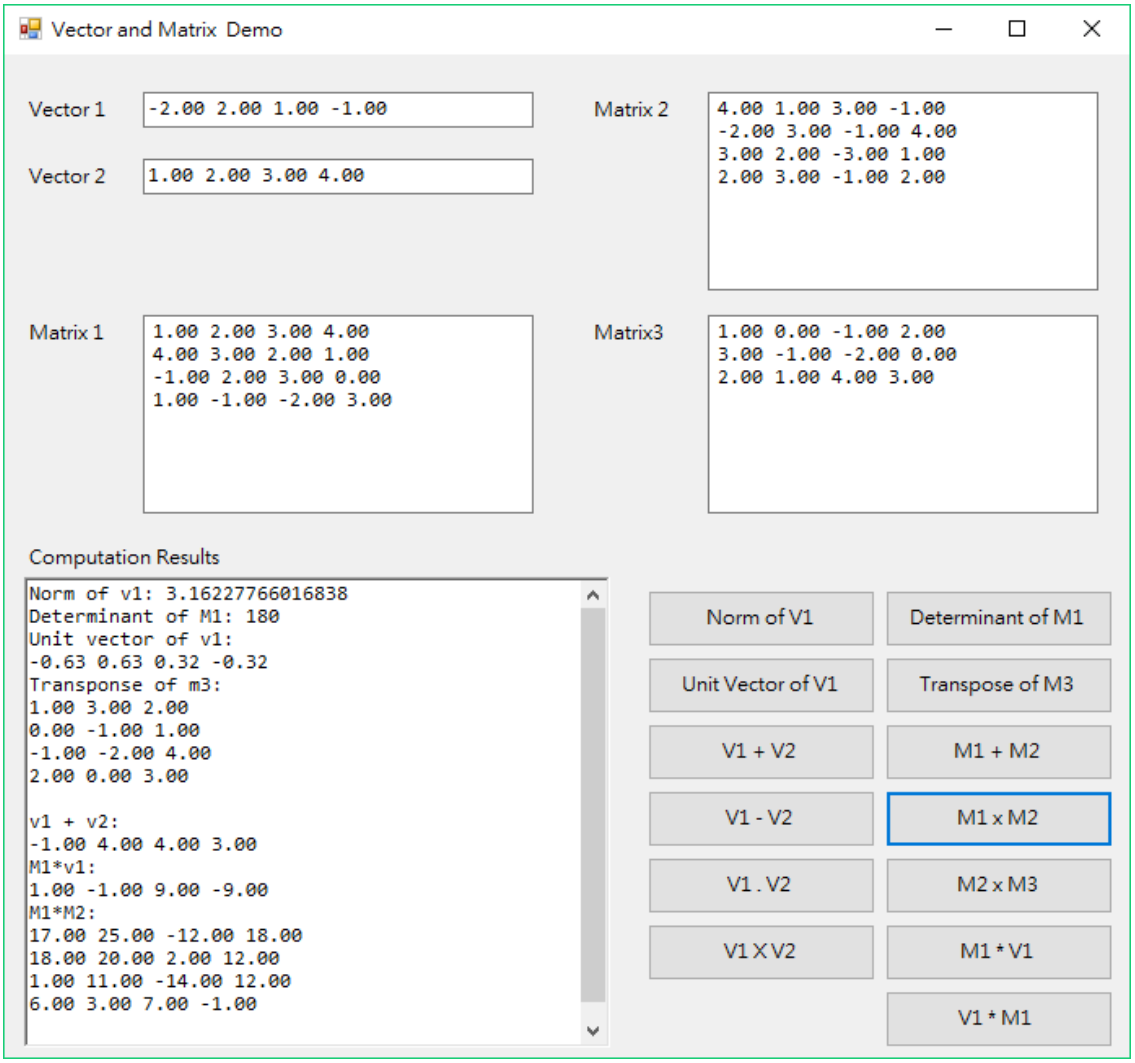
For example:

```
double[] vector1 = { -2, 2, 1, -1 };
double[] vector2 = { 1, 2, 3, 4 };
double[,] matrix1 = { { 1, 2, 3, 4 }, { 4, 3, 2, 1 }, { -1, 2, 3, 0 }, { 1, -1, -2, 3 } };
double[,] matrix2 = { { 4, 1, 3, -1 }, { -2, 3, -1, 4 }, { 3, 2, -3,
```

```
1 }, { 2, 3, -1, 2 } };
double[,] matrix3 = { { 1, 0, -1, 2 }, { 3, -1, -2, 0 }, { 2, 1, 4,
3 } };

void initializeUserInterfaces()
{ } // Set value properties of controls from all vectors and matrices
void updateV1()
{ } // Set Vector1 from controls. Called before using V1.
void updateV2()
{ } // Set Vector2 from controls. Called before using V2.
void updateM1()
{ } // Set Matrix1 from controls. Called before using M1.
void updateM2()
{ } // Set Matrix2 from controls. Called before using M2.
void updateM3()
{ } // Set Matrix3 from controls. Called before using M3.
```

Snapshot of a sample application:



Appendix: Determinant

The determinant of a matrix of arbitrary size can be defined by the Leibniz formula or the Laplace formula. The Leibniz formula for the determinant of an n×n matrix A is

$$\det(A) = \sum_{\boldsymbol{\sigma} \in S_n} \left(\operatorname{sgn}(\boldsymbol{\sigma}) \prod_{i=1}^{n} a_{i,\sigma_i} \right)$$

Here the sum is computed over all permutations **σ** of the set {1, 2, ..., n}. A permutation is a function that reorders this set of integers. The value in the ith position after the reordering **σ** is denoted by σ_i. For example, for n = 3, the original sequence 1, 2, 3 might be reordered to **σ** = [2, 3, 1], with $\sigma_1 = 2$, $\sigma_2 = 3$, and $\sigma_3 = 1$. The set of all such permutations (also known as the symmetric group on n elements) is denoted by S_n. For each permutation **σ**, sgn(**σ**) denotes the signature of **σ**, a value that is +1 whenever the reordering given by **σ** can be achieved by successively interchanging two entries an even number of times, and −1 whenever it can be achieved by an odd number of such interchanges.

2x2 matrices:

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

3x3 matrices:

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix} = aei + bfg + cdh - ceg - bdi - afh$$

4x4 matrices:

$$\begin{vmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{vmatrix} = a \begin{vmatrix} f & g & h \\ j & k & l \\ n & o & p \end{vmatrix} - b \begin{vmatrix} e & g & h \\ i & k & l \\ m & o & p \end{vmatrix} + c \begin{vmatrix} e & f & h \\ i & j & l \\ m & n & p \end{vmatrix} - b \begin{vmatrix} e & f & g \\ i & j & k \\ m & n & o \end{vmatrix}$$