Module 3

Sunday, September 17, 2023 7:39 PM

In-Class Exercise 1: Pick three other applications and propose natural groupings of numbers for those applications.

1. Weather Forecasting

- Atmospheric Pressure Data: Readings at various altitudes
- Temperature Data: Readings at different times and locations
- Humidity Data: Humidity levels at different times and places
- Wind Speeds: Speed and direction at different altitudes

2. Natural Language Processing (NLP)

- Word Embeddings: Vectors that represent the semantic meaning of words.
- Document Vectors: Average or weighted sum of word embeddings in a document.
- TF-IDF Scores: Term Frequency-Inverse Document Frequency scores for words in a corpus

3. Time Series Analysis

- Historical Data: Past observations of a variable over time.
- Frequency Components: Fourier coefficients representing cyclical behavior.
- Predicted Values: Forecasted observations based on the model.

In-Class Exercise 3: Why is it true that the orientation (angle), except for flips, is unchanged by scalar multiplication?

Given a vector A and a scalar α, the scalar multiple αA can be represented as:

$$\alpha A = (\alpha x, \alpha y)$$

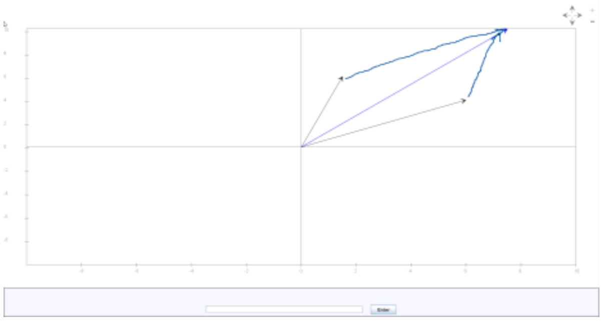
The orientation or angle θ of the vector A with respect to the x-axis is given by:

$\theta = \tan^{-1}(y/x)$ angle of A wrt x axis

$\Rightarrow \theta = \tan^{-1}(\alpha y/\alpha x)$ after scalar Mult.

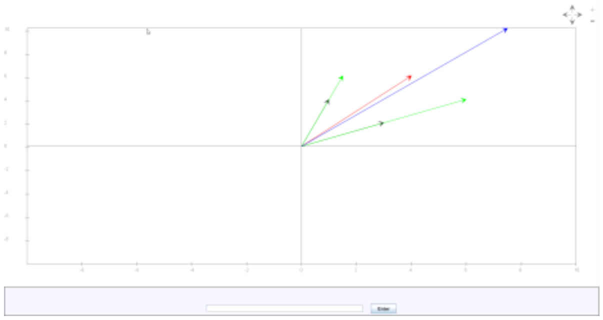
$$\Rightarrow \theta = \tan^{-1}(y/x) = \tan^{-1}(\alpha y/\alpha x)$$

In-Class Exercise 4: Download LinCombExample.java and DrawTool.java. Then, examine the code in LinCombExample to see how vectors and arrows are drawn. Then, draw the remaining arrows to complete the parallelogram.

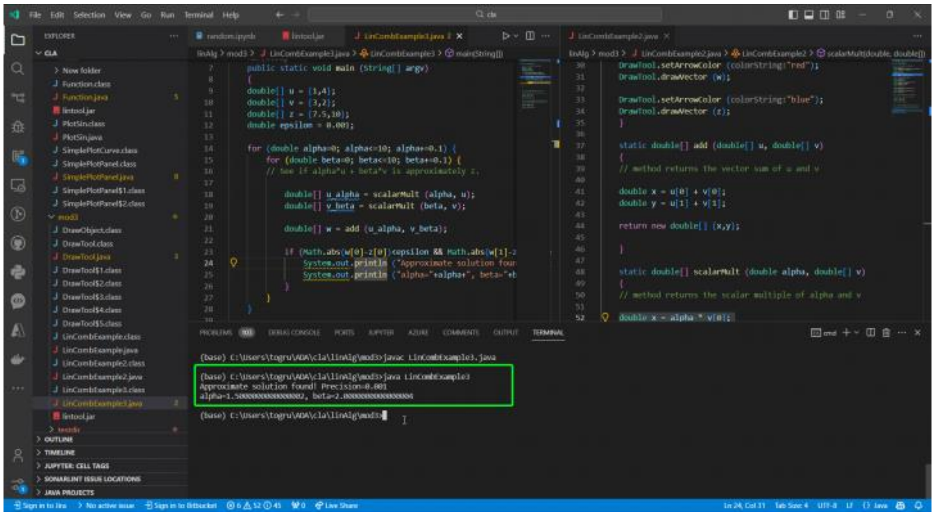


In-Class Exercise 5: Download LinCombExample2.java (you already have DrawTool). Write code to implement vector addition and scalar multiplication, and test with the example in main().

Here, red arrow indicates (u + v) while the green arrows indicate alpha * u and beta * v



In-Class Exercise 6: Download LinCombExample3.java. The double-for loop tries to systematically search over possible values of α,β to see if some combination will work. Write code to see if the linear combination αu+βv is approximately equal to z.



```java
public static void main (String[] argv)
{
double[] u = {1,4};
double[] v = {3,2};
double[] z = {7.5,10};
double epsilon = 0.001;

for (double alpha=0; alpha<=10; alpha+=0.1) {
    for (double beta=0; beta<=10; beta+=0.1) {
    // see if alpha*u + beta*v is approximately z.

        double[] u_alpha = scalarMult (alpha, u);
        double[] v_beta = scalarMult (beta, v);

        double[] w = add (u_alpha, v_beta);

        if (Math.abs(w[0]-z[0])<epsilon && Math.abs(w[1]-z
            System.out.println ("Approximate solution foun
            System.out.println ("alpha="+alpha+", beta="+b
        }
    }
}
```

```java
DrawTool.setArrowColor (colorString:"red");
DrawTool.drawVector (w);

DrawTool.setArrowColor (colorString:"blue");
DrawTool.drawVector (z);
}

static double[] add (double[] u, double[] v)
{
// method returns the vector sum of u and v

double x = u[0] + v[0];
double y = u[1] + v[1];

return new double[] {x,y};

}

static double[] scalarMult (double alpha, double[] v)
{
// method returns the scalar multiple of alpha and v

double x = alpha * v[0];
```

```
(base) C:\Users\togru\ADA\cla\linAlg\mod3>javac LinCombExample3.java

(base) C:\Users\togru\ADA\cla\linAlg\mod3>java LinCombExample3
Approximate solution found! Precision=0.001
alpha=1.5000000000000002, beta=2.0000000000000004

(base) C:\Users\togru\ADA\cla\linAlg\mod3>
```

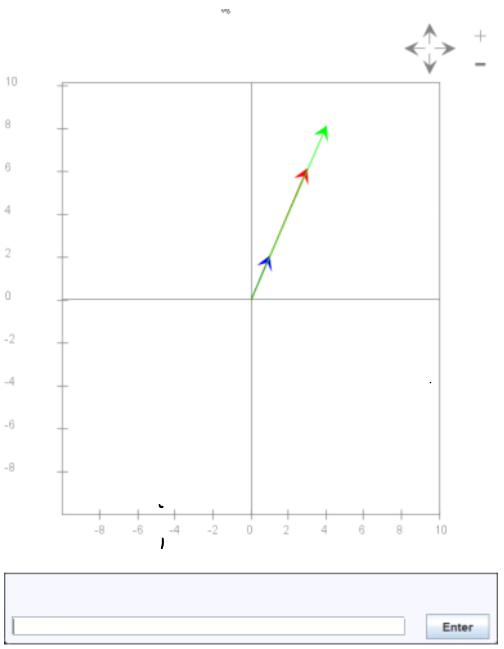
In-Class Exercise 9: Now consider **u**=(1,2),**v**=(3,6)and **z**=(4,8). Write code in LinCombExample5.java to draw these. Is there a unique solution? Explain both geometrically and algebraically.

As we can see below, all of the vectors are on the same line and are linearly dependent on each other.
Geometrically, the vectors u and v are collinear -> they lie along the same line. This means that their linear combinations will span only a one-dimensional subspace (a line in this case). The vector z also lies along this line. Therefore, there are infinitely many ways to represent z as a linear combination of u and v.

$$\begin{array}{c|cc|c} & \alpha & \beta & \\ R_1 & 1 & 3 & 4 \\ R_2 & 2 & 6 & 8 \end{array}$$

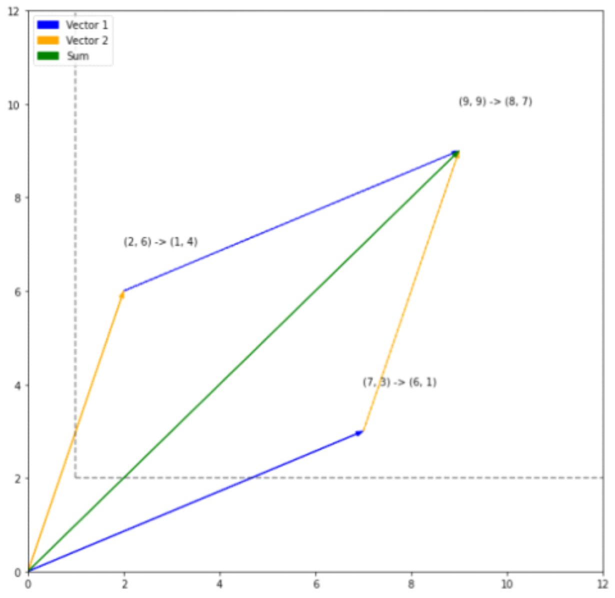
$2R_1 - R_2 \Rightarrow \boxed{0=0}$ tautology

this means we have infinitely many solutions



In-Class Exercise 2: Explain the following:

- Go back to the example with two coordinate systems, and verify by hand that addition produces the same result, relative to the new origin.
- What would be the case if a second set of coordinate axes had the same origin but was rotated slightly (say, by 60 degrees)?
- By the rules above, we aren't allowed to change the orientation of an arrow when it's moved for addition. Why is that? What would go wrong with the theory if we allowed the direction to change?



1. New Coordinate System at [1, 2]
   a. Vector 1: [7 3]
   b. Vector 2: [2 6]
   c. Sum: [9 9]
   d. Origin of new coordinate system: [1 2]
   e. Vector 1 in new coordinate system: [6 1]
   f. Vector 2 in new coordinate system: [1 4]
   g. Sum in new coordinate system: [8 7]
2. What would be the case if a second set of coordinate axes had the same origin but was rotated slightly (say, by 60 degrees)?
   a. the vectors themselves would not change, as vectors are geometric entities independent of any coordinate system. However, the coordinates of the vectors in this new rotated coordinate system would differ from those in the original system
   b. Let's say you have two vectors AA and BB in the original coordinate system, and A'A' and B'B' represent the coordinates of these vectors in the rotated system.
      In general, you can transform coordinates from one system to another using a rotation matrix RR. If θ is the angle of rotation, then the 2x2 rotation matrix R for a counter-clockwise rotation in 2D is given by:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Thus, the new coordinates would be:

$$A' = RA$$

$$B' = RB$$

   a. Interpretation:
      i. Direction: The vectors A and B will appear rotated by 60 degrees in this new system.
      ii. Magnitude: The lengths of the vectors will remain unchanged because a rotation doesn't affect vector magnitudes.
      iii. Vector Addition: The vector C=A+B will also rotate by the same angle when represented in the new coordinate system, and its coordinates would be C'=RC
1. By the rules above, we aren't allowed to change the orientation of an arrow when it's moved for addition. Why is that? What would go wrong with the theory if we allowed the direction to change?
   a. Since a vector also holds the concept of direction in addition to the concept of magnitude, changing its orientation would result in the terminal points of the vector being displaced, hence, creating a totally different vector
   b. Commutative and Associative Laws: In vector arithmetic, A+B=B+A and (A+B)+C=A+(B+C). Changing the orientation of a vector when adding would violate these laws, undermining the

mathematical structure we rely on

c. Also, when a vector is translated it is always linearly dependent on the original vector as the equation of the line on which it lies needs to transformed solely linearly

---

In-Class Exercise 7: Solve the above problem by hand: are there values of α, β such that $\alpha\mathbf{u} + \beta\mathbf{v} = \mathbf{z}$ when $\mathbf{u} = (1, 4)$, $\mathbf{v} = (3, 2)$ and $\mathbf{z} = (7.5, 10)$?

$$X: \alpha \cdot 1 + \beta \cdot 3 = 7.5$$

$$y: \alpha \cdot 4 + \beta \cdot 2 = 10$$

$$\Rightarrow \begin{array}{ccccc} \alpha & \beta & & & \\ 1 & 3 & | & 7.5 & R_1 \\ 4 & 2 & | & 10 & R_2 \end{array}$$

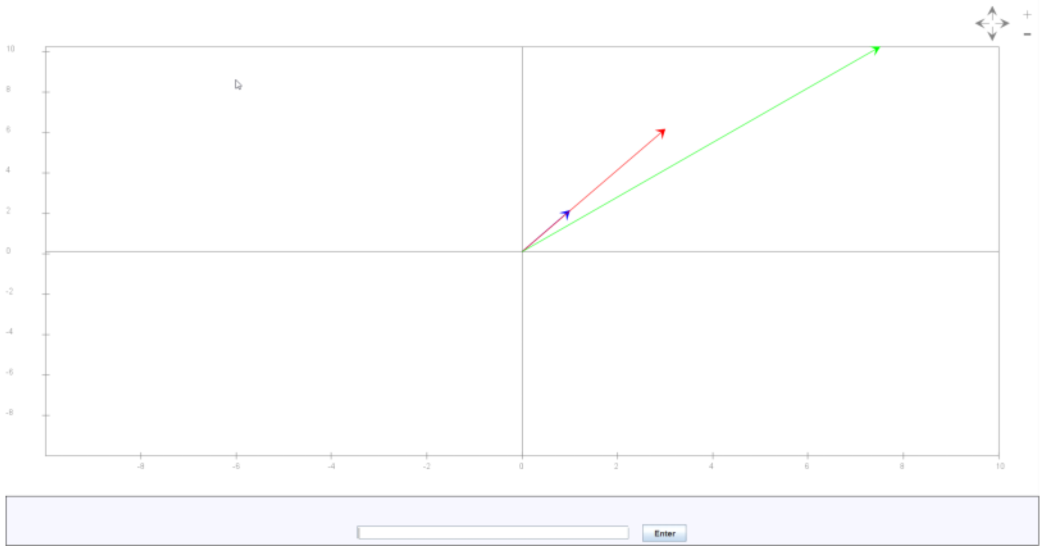
$$4R_1 - R_2 \Rightarrow 10\beta = 20$$

$$\boxed{\begin{array}{l} \beta = 2 \\ \alpha = 1.5 \end{array}}$$

In-Class Exercise 8: Suppose $\mathbf{u} = (1, 2)$, $\mathbf{v} = (3, 6)$ and $\mathbf{z} = (7.5, 10)$. Download LinCombExample4.java and draw all three vectors. Then:

- Use the drawing to explain why no linear combination of **u** and **v** will work.
- Solve by hand to confirm the explanation algebraically.

From the below image we can see that the vectors u and v (in blue and red respectively) simply linear transformations of each other. Since they are linearly dependent they are on the same line an can be defined by the same equation of the line. Thus, it is impossible to get to any other point on the plane XY aside from the points that fall onto the same line as u and v.



$v = 3 \cdot u$

$\Rightarrow$ we can use u for the comparison

For there to be a sol. the following system of eq. needs to have a sol.

[illegible]

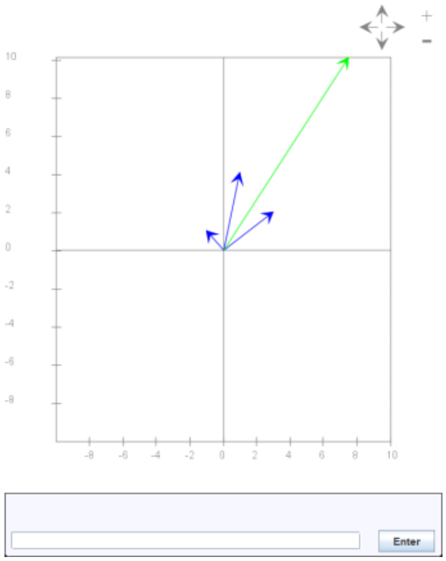
For there to be a sol. the following system of eq. needs to have a sol.

$$\begin{array}{c c} R_1 & \\ R_2 & \end{array}\left[\begin{array}{cc|c} 1 & 3 & 7.5 \\ 2 & 6 & 10 \end{array}\right]$$

$R_2 - 2R_1 \Rightarrow \quad 0 = -5 \rightarrow$ contradiction, so no solution,

In-Class Exercise 10: Finally, consider $\mathbf{u} = (1,4)$, $\mathbf{v} = (3,2)$, $\mathbf{w} = (-1,1)$ and $\mathbf{z} = (7.5, 10)$. We will ask whether there is a linear combination $\alpha\mathbf{u} + \beta\mathbf{v} + \gamma\mathbf{w}$ such that $\alpha\mathbf{u} + \beta\mathbf{v} + \gamma\mathbf{w} = \mathbf{z}$. Write out the equations and explain algebraically whether there is a solution. The code in LinCombExample6.java draws all four vectors. Can a linear combination of any two of **u, v, w** suffice to produce **z**?

Hypothesis: Considering that all 3 vectors are on 2D, and that they do not appear to have collinearity, there should be such a combination of 2 vectors that we can obtain the vector z. This is a logical conclusion from the fact that 2 equations (thus, 2 vectors) should generally be enough to prove existence or impossibility of a solution.



$$\begin{array}{c|ccc|c} & \alpha & \beta & \gamma & \\ R_1 & 1 & 3 & -1 & 7.5 \\ R_2 & 4 & 2 & 1 & 10 \end{array}$$

$$\underset{\downarrow}{R_1 + R_2} \Rightarrow \left[\begin{array}{cc|c} 5 & 5 & 17.5 \\ 4 & 2 & 10 \end{array}\right]$$
$$R_1$$

$$\Rightarrow \left[\begin{array}{cc|c} 1 & 1 & 3,5 \\ 2 & 1 & 5 \end{array}\right]$$

$$R_2 - R_1 \Rightarrow \begin{array}{l} \alpha = 1,5 \\ \beta = 2 \\ \gamma = 0 \end{array}$$

in fact, it's actually possible to get vect. z through combo of just u and v

$\gamma = 0$

$$\boxed{\text{in fact, it's actually possible to get vect. z through combo of just u and v}}$$

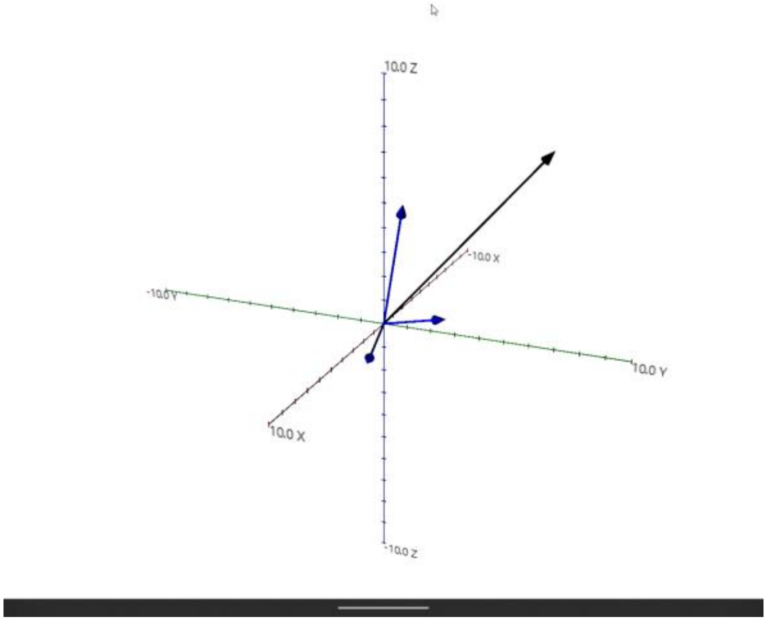
In-Class Exercise 11: The code in LinComb3DExample.java displays the vectors in the above example. (You need to have the Draw3D jar in your CLASSPATH). Write down the equations for α, β, γ and solve by hand.

$$\alpha(1,3,1) + \beta(4,1,0) + \gamma(3,2,6) = (1,7,8)$$

$$\begin{array}{c} R_1 \\ R_2 \\ R_3 \end{array} \left[\begin{array}{ccc|c} 1 & 4 & 3 & 1 \\ 3 & 1 & 2 & 7 \\ 1 & 0 & 6 & 8 \end{array}\right]$$

$$R_1 = 2R_1 - R_3 \Rightarrow \begin{array}{c} \begin{array}{ccc} \alpha & \beta & \gamma \end{array} \\ \left[\begin{array}{ccc|c} 1 & 8 & 0 & -6 \\ 3 & 1 & 2 & 7 \\ 1 & 0 & 6 & 8 \end{array}\right] \end{array}$$

$$R_2 = 3R_2 - R_3 = \left[\begin{array}{ccc|c} 1 & 8 & 0 & -6 \\ 8 & 3 & 0 & 13 \\ 1 & 0 & 6 & 8 \end{array}\right]$$



$$\begin{array}{c} R_1 \\ R_2 \end{array} \left[\begin{array}{cc|c} 1 & 8 & -6 \\ 8 & 3 & 13 \end{array}\right]$$

$$R_1 = 8R_1 - R_2 \Rightarrow \begin{array}{c} \begin{array}{cc} \alpha & \beta \end{array} \\ \left[\begin{array}{cc|c} 0 & 61 & -61 \\ 8 & 3 & 13 \end{array}\right] \end{array}$$

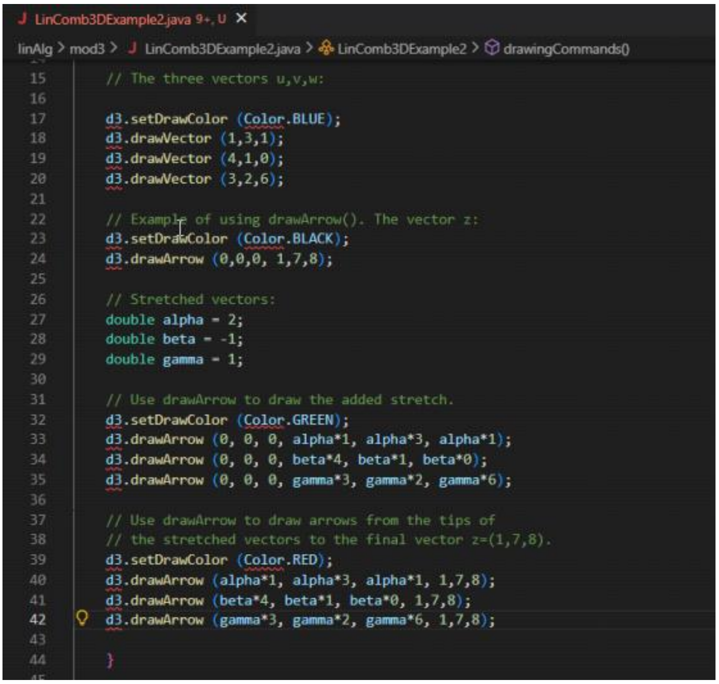
$$\beta = -1$$

$$8\alpha - 3 = 13 \Rightarrow \alpha = 2$$

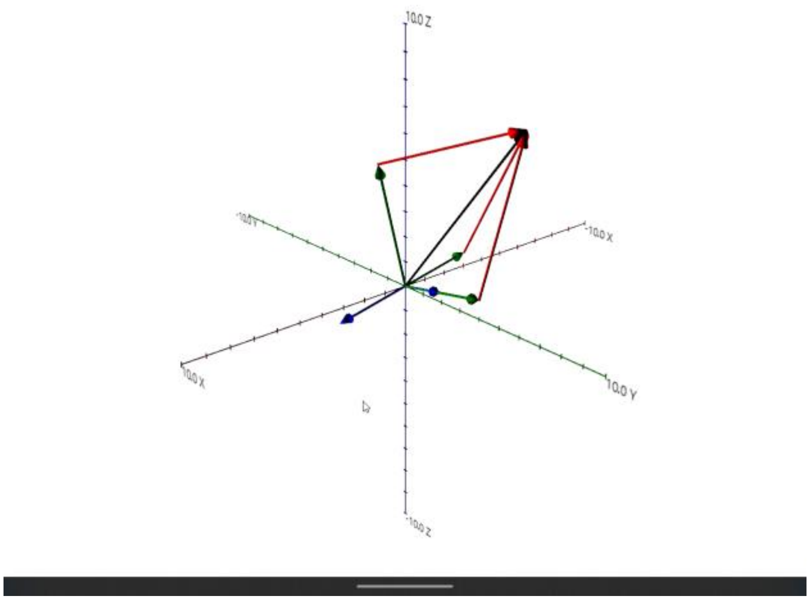
$$\boxed{\alpha = 2, \beta = -1, \gamma = 1}$$

$$1 \cdot \alpha + 6\gamma = 8$$
$$2 + 6\gamma = 8 \Rightarrow \gamma = 1$$

In-Class Exercise 12: Add code to LinComb3DExample2.java to display the 3D version of the 2D parallelogram. That is, draw arrows that show the stretched vectors when scaling by α, β, γ respectively. Then draw arrows from the tips of the stretched vectors to the resultant vector **z**.


LinComb3DExample2.java 9+, U
linAlg > mod3 > LinComb3DExample2.java > LinComb3DExample2 > drawingCommands()

```java
    // The three vectors u,v,w:

    d3.setDrawColor (Color.BLUE);
    d3.drawVector (1,3,1);
    d3.drawVector (4,1,0);
    d3.drawVector (3,2,6);

    // Example of using drawArrow(). The vector z:
    d3.setDrawColor (Color.BLACK);
    d3.drawArrow (0,0,0, 1,7,8);

    // Stretched vectors:
    double alpha = 2;
    double beta = -1;
    double gamma = 1;

    // Use drawArrow to draw the added stretch.
    d3.setDrawColor (Color.GREEN);
    d3.drawArrow (0, 0, 0, alpha*1, alpha*3, alpha*1);
    d3.drawArrow (0, 0, 0, beta*4, beta*1, beta*0);
    d3.drawArrow (0, 0, 0, gamma*3, gamma*2, gamma*6);

    // Use drawArrow to draw arrows from the tips of
    // the stretched vectors to the final vector z=(1,7,8).
    d3.setDrawColor (Color.RED);
    d3.drawArrow (alpha*1, alpha*3, alpha*1, 1,7,8);
    d3.drawArrow (beta*4, beta*1, beta*0, 1,7,8);
    d3.drawArrow (gamma*3, gamma*2, gamma*6, 1,7,8);

    }
```



In-Class Exercise 13: Suppose now that $\mathbf{u} = (1,3,1)$, $\mathbf{v} = (4,1,0)$, $\mathbf{w} = (9,5,1)$ and $\mathbf{z} = (1,7,8)$. We seek α, β, γ such that $\alpha\mathbf{u} + \beta\mathbf{v} + \gamma\mathbf{w} = \mathbf{z}$. Write down the equations for α, β, γ and solve by hand. The code in LinComb3DExample3.java displays the four vectors. Explain geometrically why this is consistent with your solution.

Keeping in mind that scalar multiplication serves solely to stretch a vector, since it is not possible to change the angle of the vector. We can somehow see that these three vectors may be on a single plane. We'll test this later

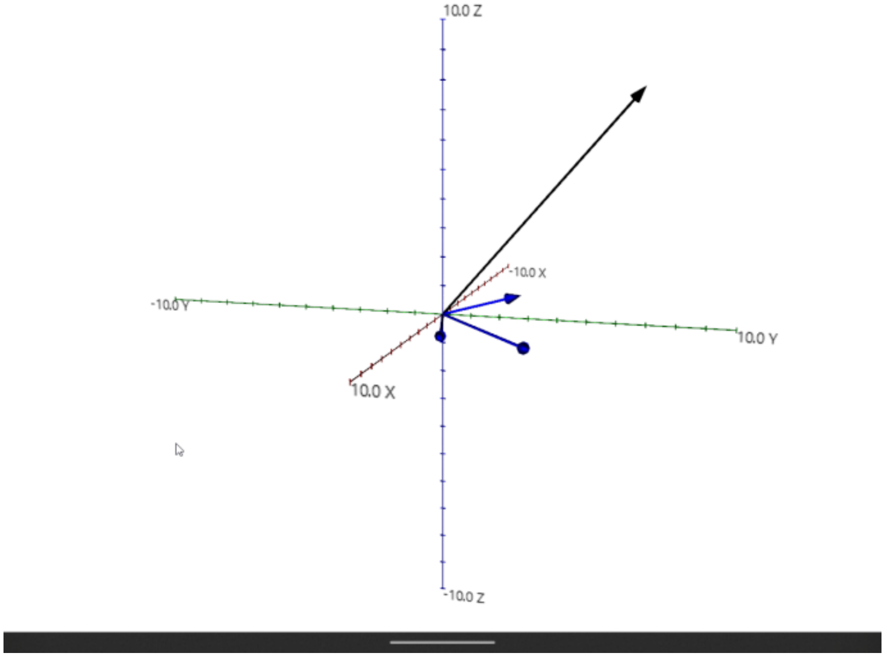
$$\begin{array}{c|ccc|c} & \alpha & \beta & \gamma & \\ \hline R_1 & 1 & 4 & 9 & 1 \\ R_2 & 3 & 1 & 5 & 7 \\ R_3 & 1 & 0 & 1 & 8 \end{array}$$

$$R_1 = R_1 - R_2 \Rightarrow \left[\begin{array}{ccc|c} 0 & 4 & 8 & -7 \\ 3 & 1 & 5 & 7 \\ 1 & 0 & 1 & 8 \end{array}\right]$$

$$R_2 = R_2 - 3R_1 \Rightarrow \left[\begin{array}{ccc|c} 0 & 4 & 8 & -7 \\ 0 & 1 & 2 & -17 \\ 1 & 0 & 1 & 8 \end{array}\right]$$

$$\begin{array}{cc|c} \beta & \gamma & \\ 4 & 8 & -7 \\ 1 & 2 & -17 \end{array}$$

$$R_1 = R_1 - 4R_2 \Rightarrow \left[\begin{array}{cc|c} 0 & 0 & 61 \\ 1 & 2 & 0 \end{array}\right]$$



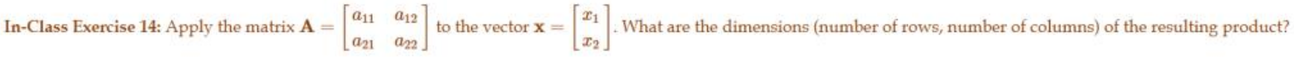
$$\boxed{0 \cdot \beta + 0 \cdot \gamma = 61 \rightarrow \text{No Solution}}$$

Let's see if these three vectors are on the same plane. To do this, we can check the determinant of the transformation matrix created by the 3 vectors. The determinant of a matrix formed by three vectors as its columns results in a scalar value that indicates the volume of the parallelepiped they can form in 3D. If the determinant is zero, the volume of the parallelepiped is zero, which means that the vectors are linearly dependent and lie on the same plane. Whilst a non-zero determinant means that the vectors

are linearly independent and span a three-dimensional space, not confined to a single plane.
In conclusion, if the Determinant will be 0, then we will have proved that it is indeed geometrically impossible to construct vector z out of the 3 vectors

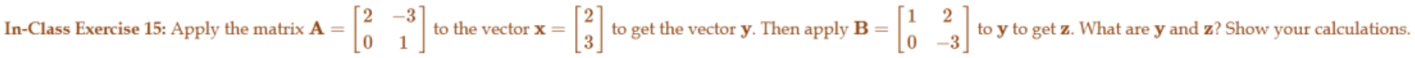
$$M = \begin{bmatrix} 1 & 4 & 9 \\ 3 & 1 & 5 \\ 1 & 0 & 1 \end{bmatrix}$$

$$\boxed{\begin{aligned} \mathrm{Det}(M) = {} & 1\cdot(1\cdot 1 - 0\cdot 5) - \\ & 4\cdot(3\cdot 1 - 5\cdot 1) + \\ & 9\cdot(3\cdot 0 - 1\cdot 1) = 1 + 8 - 9 = 0 \end{aligned}}$$



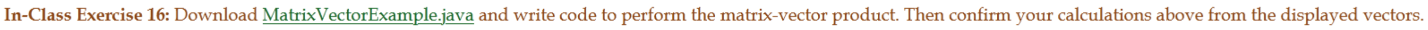
In-Class Exercise 14: Apply the matrix $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ to the vector $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. What are the dimensions (number of rows, number of columns) of the resulting product?

$$A_{2\times 2} \times X_{2\times 1} = M_{2\times 1} \qquad A \times X = \begin{bmatrix} a_{11}\cdot x_1 + a_{12}\cdot x_2 \\ a_{21}\cdot x_1 + a_{22}\cdot x_2 \end{bmatrix}$$

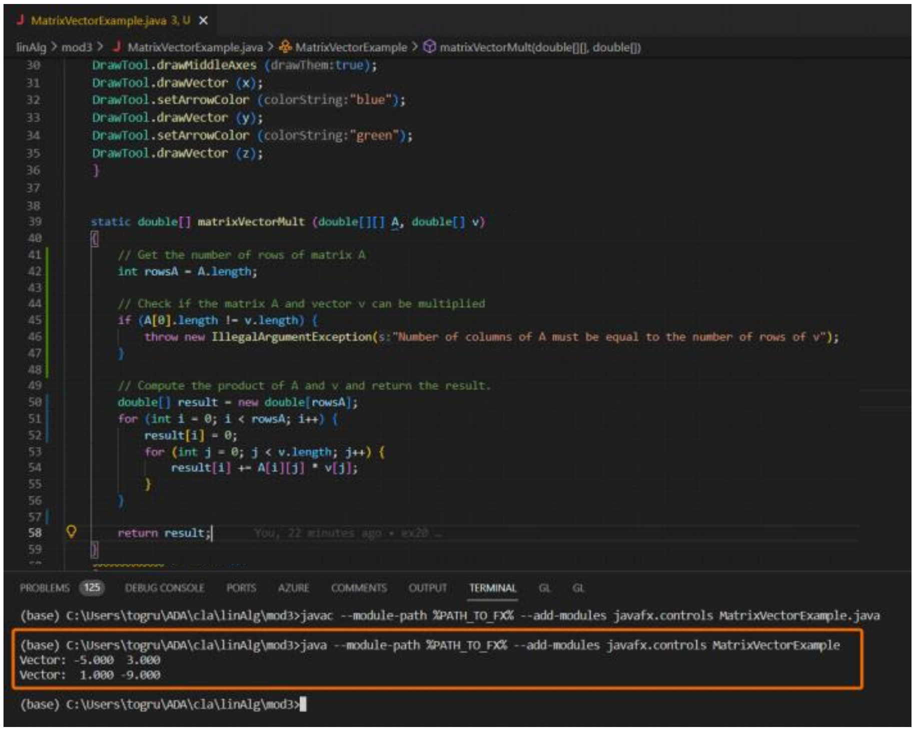


In-Class Exercise 15: Apply the matrix $\mathbf{A} = \begin{bmatrix} 2 & -3 \\ 0 & 1 \end{bmatrix}$ to the vector $\mathbf{x} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$ to get the vector $\mathbf{y}$. Then apply $\mathbf{B} = \begin{bmatrix} 1 & 2 \\ 0 & -3 \end{bmatrix}$ to $\mathbf{y}$ to get $\mathbf{z}$. What are $\mathbf{y}$ and $\mathbf{z}$? Show your calculations.

$$\begin{bmatrix} 2 & -3 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} -5 \\ 3 \end{bmatrix} = y \qquad B \times y = \begin{bmatrix} 1 & 2 \\ 0 & -3 \end{bmatrix} \times \begin{bmatrix} -5 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ -9 \end{bmatrix} = z.$$

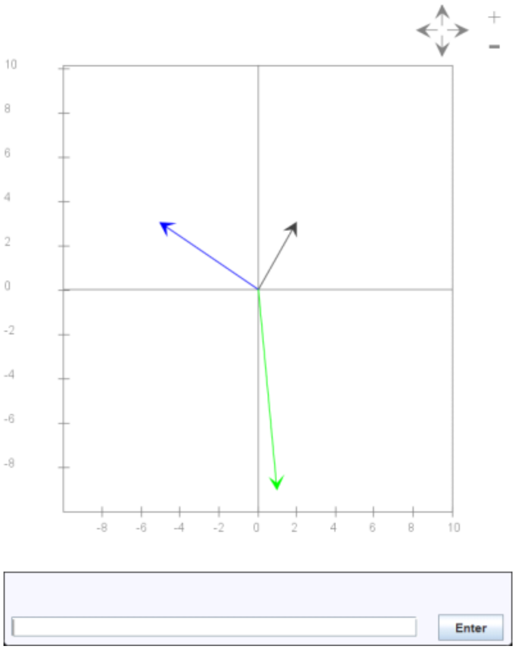


In-Class Exercise 16: Download MatrixVectorExample.java and write code to perform the matrix-vector product. Then confirm your calculations above from the displayed vectors.


MatrixVectorExample.java 3, U
linAlg > mod3 > MatrixVectorExample.java > MatrixVectorExample > matrixVectorMult(double[][], double[])
You, 22 minutes ago • ex20
PROBLEMS 125
DEBUG CONSOLE
PORTS
AZURE
COMMENTS
OUTPUT
TERMINAL
GL
GL

```java
        DrawTool.drawMiddleAxes (drawThem:true);
        DrawTool.drawVector (x);
        DrawTool.setArrowColor (colorString:"blue");
        DrawTool.drawVector (y);
        DrawTool.setArrowColor (colorString:"green");
        DrawTool.drawVector (z);
        }


        static double[] matrixVectorMult (double[][] A, double[] v)
        {
            // Get the number of rows of matrix A
            int rowsA = A.length;

            // Check if the matrix A and vector v can be multiplied
            if (A[0].length != v.length) {
                throw new IllegalArgumentException(s:"Number of columns of A must be equal to the number of rows of v");
            }

            // Compute the product of A and v and return the result.
            double[] result = new double[rowsA];
            for (int i = 0; i < rowsA; i++) {
                result[i] = 0;
                for (int j = 0; j < v.length; j++) {
                    result[i] += A[i][j] * v[j];
                }
            }

            return result;
        }
```

```
(base) C:\Users\togru\ADA\cla\linAlg\mod3>javac --module-path %PATH_TO_FX% --add-modules javafx.controls MatrixVectorExample.java

(base) C:\Users\togru\ADA\cla\linAlg\mod3>java --module-path %PATH_TO_FX% --add-modules javafx.controls MatrixVectorExample
Vector: -5.000  3.000
Vector:  1.000 -9.000

(base) C:\Users\togru\ADA\cla\linAlg\mod3>
```



In-Class Exercise 17: Compute the vector obtained by multiplying $\mathbf{x}$ above by the matrix $\mathbf{C} = \begin{bmatrix} 2 & -1 \\ 0 & -3 \end{bmatrix}$. Compare with the value of $\mathbf{z}$ calculated earlier.

$$C \times x = \begin{bmatrix} 2 & -1 \\ 0 & -3 \end{bmatrix} \times \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ -9 \end{bmatrix} = z$$

In-Class Exercise 18: Compute the vector obtained by multiplying $\mathbf{z}$ above by the matrix $\mathbf{C}_2 = \begin{bmatrix} 1.0/2 & -1.0/6 \\ 0 & -1.0/3 \end{bmatrix}$. Compare with the value of $\mathbf{x}$ calculated earlier.

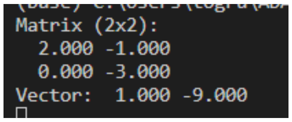
$$C_2 \times z = \begin{bmatrix} 1/2 & -1/6 \\ 0 & -1/3 \end{bmatrix} \times \begin{bmatrix} 1 \\ -9 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix} = X$$

In-Class Exercise 19: Earlier, you computed the product of $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ with the vector $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, which produces a result vector. Multiply this result vector by $\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$. (The result is admittedly not pretty.)

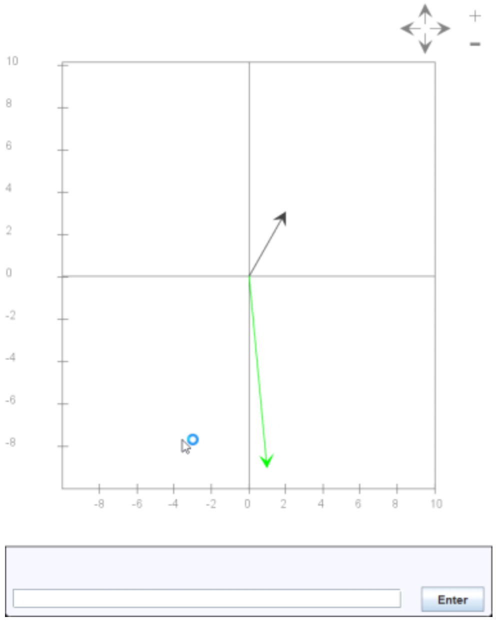
$$B \times (A \times X) = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \times \begin{bmatrix} a_{11} \cdot x_1 + a_{12} \cdot x_2 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 \end{bmatrix} = \begin{bmatrix} b_{11}(a_{11} x_1 + a_{12} x_2) + b_{12}(a_{21} \cdot x_1 + a_{22} x_2) \\ b_{21}(a_{11} x_1 + a_{12} x_2) + b_{22}(a_{21} \cdot x_1 + a_{22} x_2) \end{bmatrix}$$

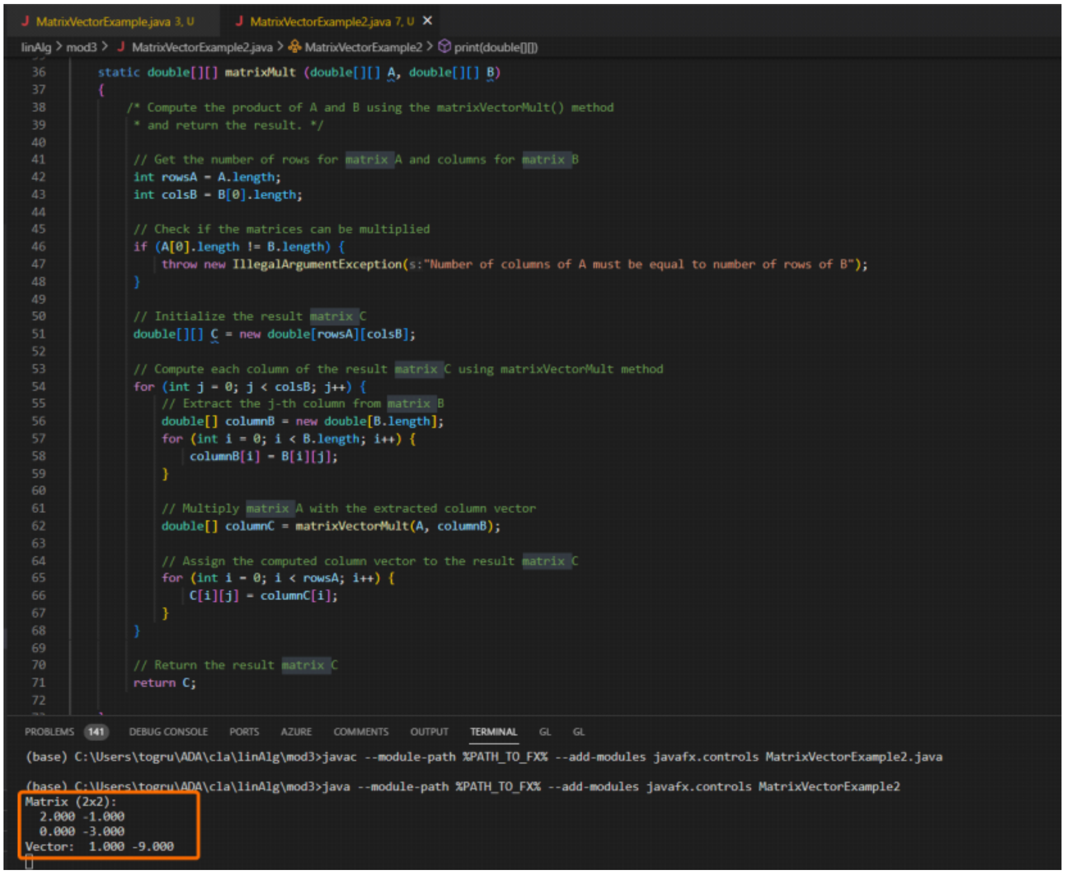
In-Class Exercise 20: Download MatrixVectorExample2.java and write matrix multiplication code to make the example work. Confirm that you obtained the same matrix in Exercise 17, and the same resulting vector.

The results fully align with ex17



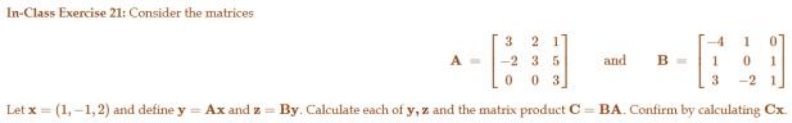
```
Matrix (2x2):
  2.000 -1.000
  0.000 -3.000
Vector:  1.000 -9.000
```




MatrixVectorExample.java 3, U
MatrixVectorExample2.java 7, U
linAlg > mod3 > MatrixVectorExample2.java > MatrixVectorExample2 > print(double[][])
PROBLEMS 141
DEBUG CONSOLE
PORTS
AZURE
COMMENTS
OUTPUT
TERMINAL
GL
GL

```java
static double[][] matrixMult (double[][] A, double[][] B)
{
    /* Compute the product of A and B using the matrixVectorMult() method
     * and return the result. */

     // Get the number of rows for matrix A and columns for matrix B
     int rowsA = A.length;
     int colsB = B[0].length;

     // Check if the matrices can be multiplied
     if (A[0].length != B.length) {
         throw new IllegalArgumentException(s:"Number of columns of A must be equal to number of rows of B");
     }

     // Initialize the result matrix C
     double[][] C = new double[rowsA][colsB];

     // Compute each column of the result matrix C using matrixVectorMult method
     for (int j = 0; j < colsB; j++) {
         // Extract the j-th column from matrix B
         double[] columnB = new double[B.length];
         for (int i = 0; i < B.length; i++) {
             columnB[i] = B[i][j];
         }

         // Multiply matrix A with the extracted column vector
         double[] columnC = matrixVectorMult(A, columnB);

         // Assign the computed column vector to the result matrix C
         for (int i = 0; i < rowsA; i++) {
             C[i][j] = columnC[i];
         }
     }

     // Return the result matrix C
     return C;
```

```
(base) C:\Users\togru\ADA\cla\linAlg\mod3>javac --module-path %PATH_TO_FX% --add-modules javafx.controls MatrixVectorExample2.java

(base) C:\Users\togru\ADA\cla\linAlg\mod3>java --module-path %PATH_TO_FX% --add-modules javafx.controls MatrixVectorExample2
Matrix (2x2):
  2.000 -1.000
  0.000 -3.000
Vector:  1.000 -9.000
```

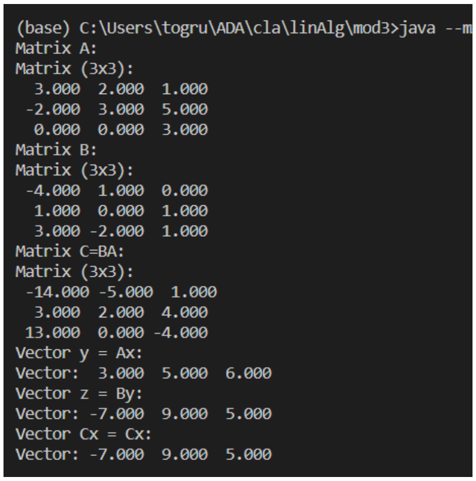


In-Class Exercise 21: Consider the matrices

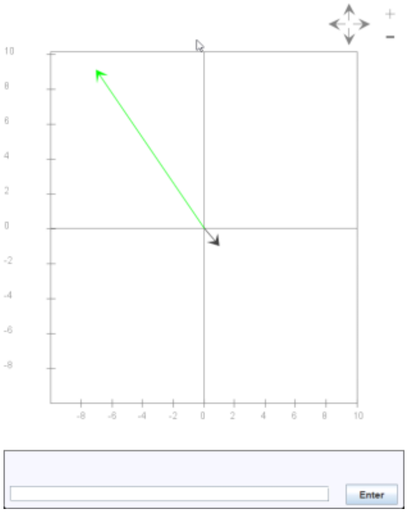
$$\mathbf{A} = \begin{bmatrix} 3 & 2 & 1 \\ -2 & 3 & 5 \\ 0 & 0 & 3 \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} -4 & 1 & 0 \\ 1 & 0 & 1 \\ 3 & -2 & 1 \end{bmatrix}$$

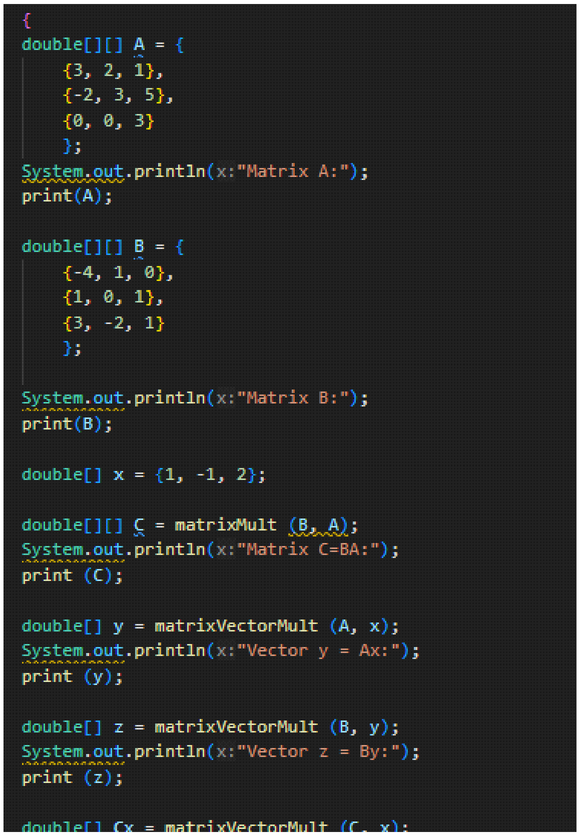
Let $\mathbf{x} = (1, -1, 2)$ and define $\mathbf{y} = \mathbf{Ax}$ and $\mathbf{z} = \mathbf{By}$. Calculate each of $\mathbf{y}, \mathbf{z}$ and the matrix product $\mathbf{C} = \mathbf{BA}$. Confirm by calculating $\mathbf{Cx}$.

For this I used the earlier script for the purposes of saving time. The modified code is available in the file MatrixVectorExample2ex21. The results are given below and are consistent

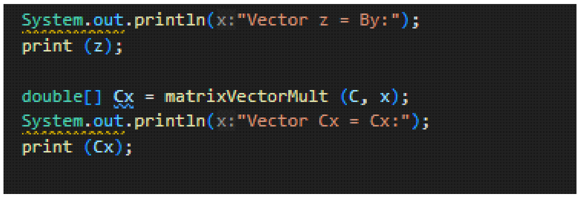


```
(base) C:\Users\togru\ADA\cla\linAlg\mod3>java --m
Matrix A:
Matrix (3x3):
  3.000  2.000  1.000
 -2.000  3.000  5.000
  0.000  0.000  3.000
Matrix B:
Matrix (3x3):
 -4.000  1.000  0.000
  1.000  0.000  1.000
  3.000 -2.000  1.000
Matrix C=BA:
Matrix (3x3):
 -14.000 -5.000  1.000
  3.000  2.000  4.000
 13.000  0.000 -4.000
Vector y = Ax:
Vector:  3.000  5.000  6.000
Vector z = By:
Vector: -7.000  9.000  5.000
Vector Cx = Cx:
Vector: -7.000  9.000  5.000
```





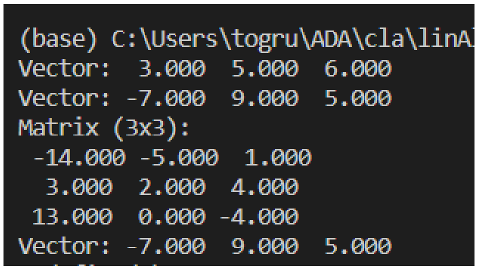
```java
{
double[][] A = {
    {3, 2, 1},
    {-2, 3, 5},
    {0, 0, 3}
    };
System.out.println(x:"Matrix A:");
print(A);

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

System.out.println(x:"Matrix B:");
print(B);

double[] x = {1, -1, 2};

double[][] C = matrixMult (B, A);
System.out.println(x:"Matrix C=BA:");
print (C);

double[] y = matrixVectorMult (A, x);
System.out.println(x:"Vector y = Ax:");
print (y);

double[] z = matrixVectorMult (B, y);
System.out.println(x:"Vector z = By:");
print (z);

double[] Cx = matrixVectorMult (C, x);
```



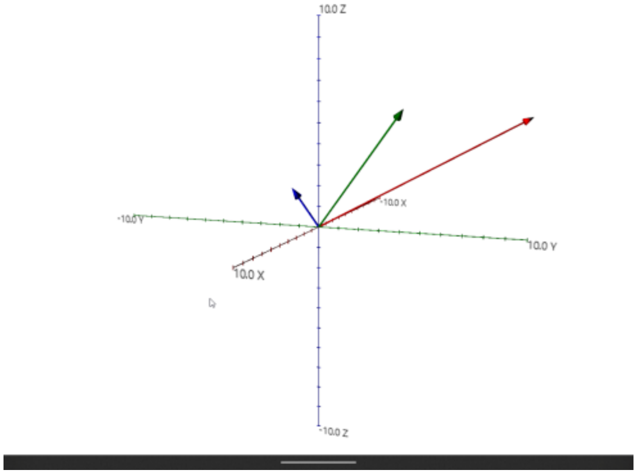
```
System.out.println(x:"Vector z = By:");
print (z);

double[] Cx = matrixVectorMult (C, x);
System.out.println(x:"Vector Cx = Cx:");
print (Cx);
```

In-Class Exercise 22: Download MatrixVector3DExample.java, then fill in the code (from previous exercises) to perform matrix multiplication and matrix-vector multiplication. Confirm that the results match the calculations in the previous exercise.

Matrix-matrix and matrix-vector multiplication methods were exactly the same as the above exercises. So I will not be attaching it in this section separately.

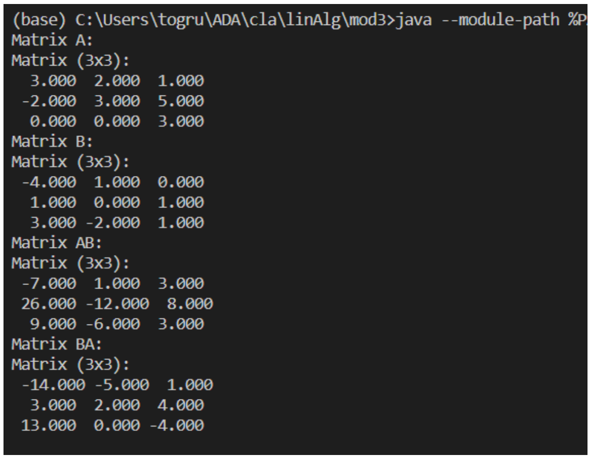


```
(base) C:\Users\togru\ADA\cla\linAl
Vector:  3.000  5.000  6.000
Vector: -7.000  9.000  5.000
Matrix (3x3):
 -14.000 -5.000  1.000
  3.000  2.000  4.000
 13.000  0.000 -4.000
Vector: -7.000  9.000  5.000
```



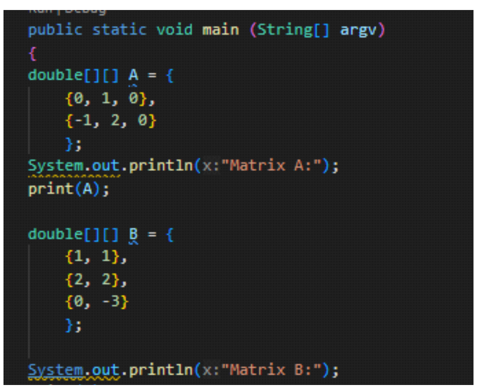
In-Class Exercise 23: Suppose $\mathbf{A}$ and $\mathbf{B}$ are two $n \times n$ matrices. Prove or disprove: $\mathbf{AB} = \mathbf{BA}$.

Using the coed from the previous exercise, we get the results illustrated on the right. This means that AB != BA

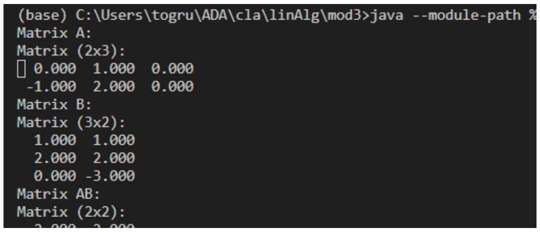


```
(base) C:\Users\togru\ADA\cla\linAlg\mod3>java --module-path %P
Matrix A:
Matrix (3x3):
  3.000  2.000  1.000
 -2.000  3.000  5.000
  0.000  0.000  3.000
Matrix B:
Matrix (3x3):
 -4.000  1.000  0.000
  1.000  0.000  1.000
  3.000 -2.000  1.000
Matrix AB:
Matrix (3x3):
 -7.000  1.000  3.000
 26.000 -12.000  8.000
  9.000 -6.000  3.000
Matrix BA:
Matrix (3x3):
 -14.000 -5.000  1.000
  3.000  2.000  4.000
 13.000  0.000 -4.000
```

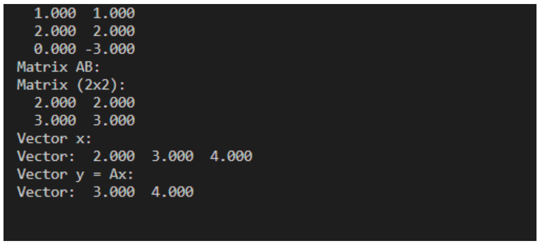
In-Class Exercise 24: What is the result of multiplying $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 2 & 0 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 0 & -3 \end{bmatrix}$? What is $\mathbf{Ax}$ when $\mathbf{x} = (2, 3, 4)$?



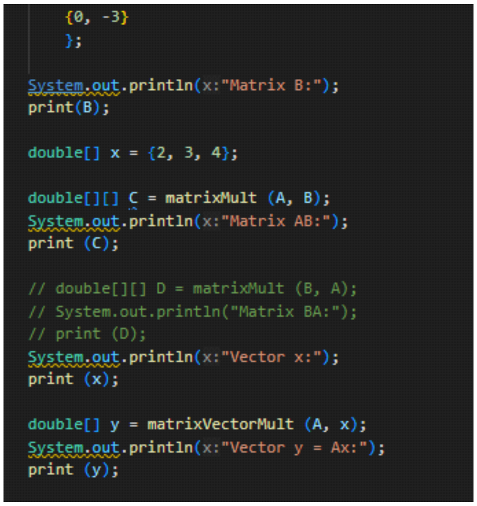
```
Run | Debug
public static void main (String[] argv)
{
double[][] A = {
    {0, 1, 0},
    {-1, 2, 0}
    };
System.out.println(x:"Matrix A:");
print(A);

double[][] B = {
    {1, 1},
    {2, 2},
    {0, -3}
    };

System.out.println(x:"Matrix B:");
```



```
(base) C:\Users\togru\ADA\cla\linAlg\mod3>java --module-path %
Matrix A:
Matrix (2x3):
□ 0.000  1.000  0.000
 -1.000  2.000  0.000
Matrix B:
Matrix (3x2):
  1.000  1.000
  2.000  2.000
  0.000 -3.000
Matrix AB:
Matrix (2x2):
```



```
  1.000  1.000
  2.000  2.000
  0.000 -3.000
Matrix AB:
Matrix (2x2):
  2.000  2.000
  3.000  3.000
Vector x:
Vector:  2.000  3.000  4.000
Vector y = Ax:
Vector:  3.000  4.000
```



```
    {0, -3}
    };

System.out.println(x:"Matrix B:");
print(B);

double[] x = {2, 3, 4};

double[][] C = matrixMult (A, B);
System.out.println(x:"Matrix AB:");
print (C);

// double[][] D = matrixMult (B, A);
// System.out.println("Matrix BA:");
// print (D);
System.out.println(x:"Vector x:");
print (x);

double[] y = matrixVectorMult (A, x);
System.out.println(x:"Vector y = Ax:");
print (y);
```