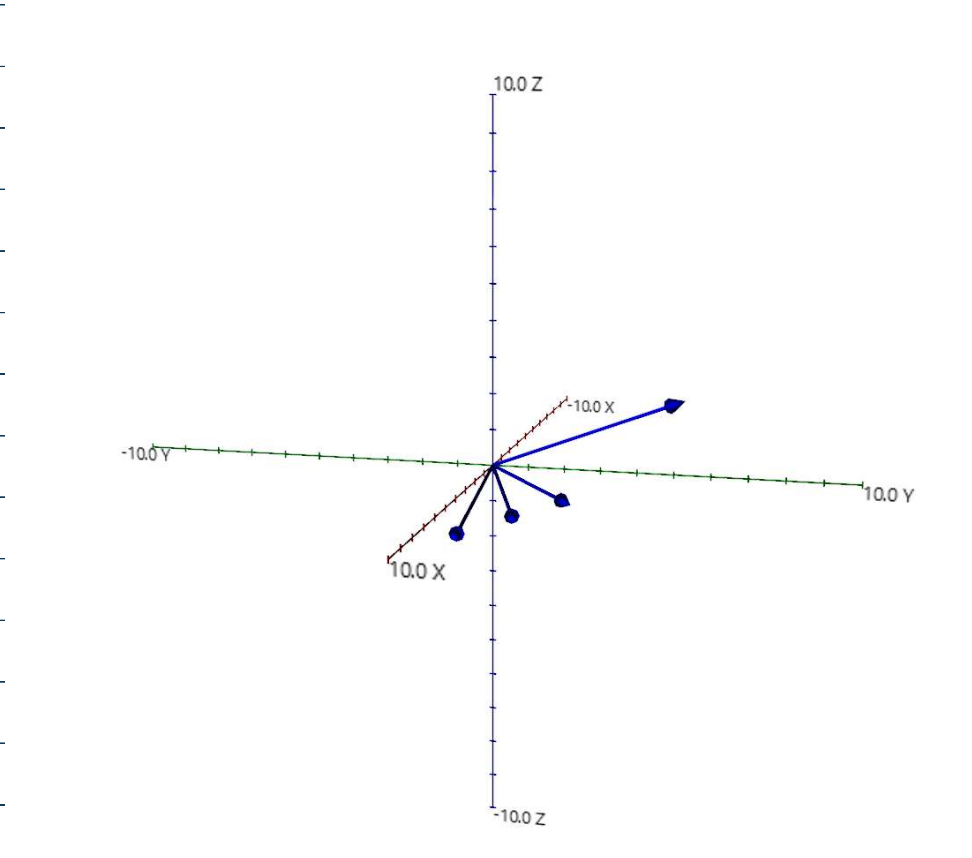
Module 7

Sunday, November 5, 2023 12:15 PM

In-Class Exercise 1: Download EquationExample3D.java and plot all three columns as vectors, and the vector $\mathbf{b} = (5, 6, 3)$.



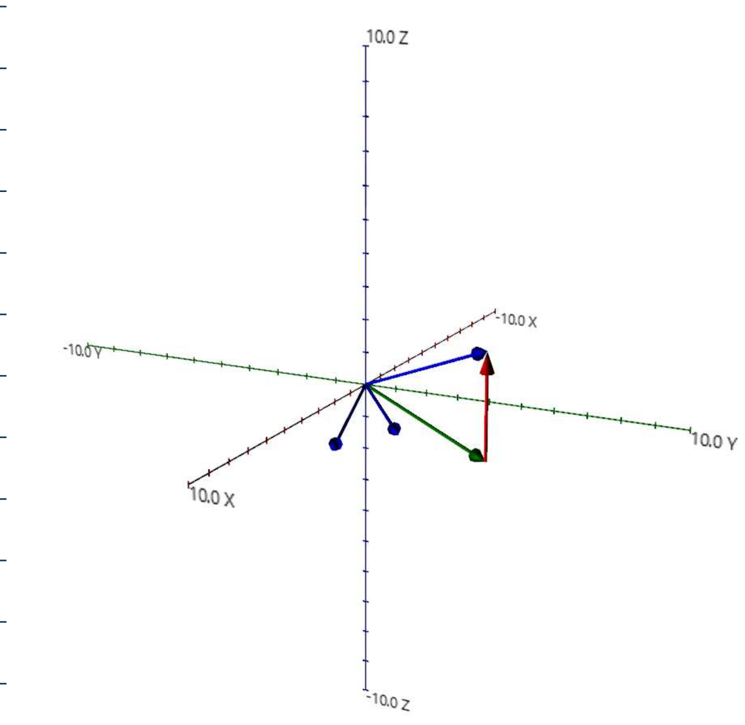
In-Class Exercise 2: Download SearchClosest3D.java which tries various linear combinations $\mathbf{y} = \alpha\mathbf{c}_1 + \beta\mathbf{c}_2$ to pick the one closest to $\mathbf{b}$.

- Why is $\mathbf{c}_3$ not included in the linear combination?
- Un-comment the code at the end to draw the vector from $\mathbf{y}$ to $\mathbf{b}$.
- Draw by hand the three axes, vectors $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{b}$ and your solution to the closest $\mathbf{y}$.
- Change the range of search to see if you can get a closer $\mathbf{y}$.

```java
    // Change the range to see if you can find a better y.
    double alphaLow=-100, alphaHigh=100, alphaStep=0.1;
    double betaLow=-100, betaHigh=100, betaStep=0.1;
    double min = Double.MAX_VALUE;
    double[] bestY = {0,0};

    for (double alpha=alphaLow; alpha<=alphaHigh; alpha+=alphaStep) {
        for (double beta=betaLow; beta<=betaHigh; beta+=betaStep) {
        double[] y = MatrixTool.add (
                MatrixTool.scalarMult (alpha, c1),
                MatrixTool.scalarMult (beta, c2)
                );
        // Find the difference vector and its length:
        double[] z = MatrixTool.sub (b, y);
        double zLength = MatrixTool.norm (z);
        if (zLength < min) {
            // Record best so far.
            min = zLength;
            bestY = y;
```

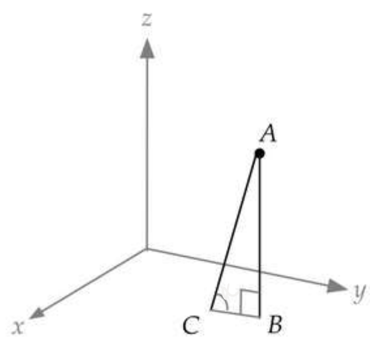
PROBLEMS 219 DEBUG CONSOLE PORTS AZURE COMMENTS OUTPUT TERMINAL GL GL

```
Undefined key
Undefined key
Undefined key
Undefined key

(base) C:\Users\togru\ADA\cla\linAlg\module7>javac --module-path %PATH_TO_FX% --add-modules javafx.controls SearchClosest3D.java

(base) C:\Users\togru\ADA\cla\linAlg\module7>java --module-path %PATH_TO_FX% --add-modules javafx.controls SearchClosest3D.java
best y = [4.9999999999859615, 5.9999999999929825, 0.0]
```



Let's now explore an idea:

- Consider a point above a plane.
- Claim: the shortest distance from the point to the plane is on the perpendicular to the plane.



In-Class Exercise 3: Prove that this is the case using the picture above.

Let's say C is a point on the plane while A is any point outside it. B is another point on the plane. We need to prove that any AC will be longer than AB provided that AB ⊥ plane.

$AB \perp$ plane $\Rightarrow AB = AC \cdot \sin\alpha$

$\Rightarrow AC = \dfrac{AB}{\sin\alpha}$, for this to hold $\alpha = 90°$

$\Rightarrow$ for $\alpha \in (0, 90)$ $\sin(\alpha) < 1$

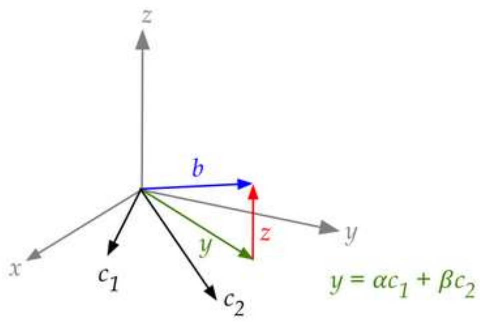
$\Rightarrow AC > AB$

$\Rightarrow$ the shortest point to plane distance is the ⊥ distance. Any other vector will be longer.

- Let

$$\mathbf{z} \triangleq \mathbf{b} - \mathbf{y}$$

be the "error" or distance between the closest linear combination $\mathbf{y}$ and $\mathbf{b}$:



In-Class Exercise 4: Prove that $\mathbf{z}$ is orthogonal to $\mathbf{c}_1$ and $\mathbf{c}_2$.

z is the closest distance between b and y. From earlier we know that $z \perp y$.

$\Rightarrow y = \alpha c_1 + \beta c_2$, linear combination

$\Rightarrow$ y is on the plane spanned by c_1 and c_2

$\Rightarrow z \perp c_1, c_2$, since $z \perp y$

In-Class Exercise 5: Use the data in the example to above to write down the equations for α, β. Then, solve the equations by hand or by using the demo equation solver from Module 1. After, enter the values of α, β in PlotClosest3D.java to see both $\mathbf{y}$ and $\mathbf{z}$.

$$\begin{bmatrix} 6 & 4 & 8 \\ 2 & 3 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 6 \\ 3 \end{bmatrix}$$

$$\begin{bmatrix} \vdots & \vdots & \vdots \\ \mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ 0 \end{bmatrix} \approx \mathbf{b}$$

- We will now solve for α, β.
- First, since $\mathbf{y} = \alpha\mathbf{c}_1 + \beta\mathbf{c}_2$

$$\begin{aligned} \mathbf{z} &= \mathbf{b} - \mathbf{y} \\ &= \mathbf{b} - \alpha\mathbf{c}_1 - \beta\mathbf{c}_2 \end{aligned}$$

- Next, since the vector $\mathbf{z}$ is orthogonal to $\mathbf{c}_1$ and $\mathbf{c}_2$, the dot products below must be zero:

$$\begin{aligned} \mathbf{c}_1 \cdot \mathbf{z} &= 0 \\ \mathbf{c}_2 \cdot \mathbf{z} &= 0 \end{aligned}$$

In other words

$$\begin{aligned} \mathbf{c}_1 \cdot (\mathbf{b} - \alpha\mathbf{c}_1 - \beta\mathbf{c}_2) &= 0 \\ \mathbf{c}_2 \cdot (\mathbf{b} - \alpha\mathbf{c}_1 - \beta\mathbf{c}_2) &= 0 \end{aligned}$$

Which gives us two equations for α, β.

$$c_1 = \begin{bmatrix} 6 \\ 2 \\ 0 \end{bmatrix} \quad c_2 = \begin{bmatrix} 4 \\ 3 \\ 0 \end{bmatrix} \quad b = \begin{bmatrix} 5 \\ 6 \\ 3 \end{bmatrix} \quad b - \alpha c_1 - \beta c_2 = \begin{bmatrix} 5 - 6\alpha - 4\beta \\ 6 - 2\alpha - 3\beta \\ 3 \end{bmatrix} = w$$

$$c_1 \cdot w = 0 \Rightarrow 30 - 36\alpha - 24\beta + 12 - 4\alpha - 6\beta = 0$$

$$c_2 \cdot w = 0 \Rightarrow 20 - 24\alpha - 16\beta + 18 - 6\alpha - 9\beta = 0$$

$$42 - 40\alpha - 30\beta = 0$$
$$38 - 30\alpha - 25\beta = 0$$

$$40\alpha + 30\beta = 42$$
$$30\alpha + 25\beta = 38$$

$$\left[\begin{array}{cc|c} 4 & 3 & 4.2 \\ 3 & 2.5 & 3.8 \end{array}\right] \Rightarrow \left[\begin{array}{cc|c} 0 & -1 & -2.6 \\ 3 & 2.5 & 3.8 \end{array}\right] \Rightarrow$$

$$\boxed{\begin{aligned} \beta &= 2.6 \\ \alpha &= -0.9 \end{aligned}}$$

$$\begin{bmatrix} 3 & 4.5 & | & 3.8 \end{bmatrix} \qquad \begin{bmatrix} 3 & 2.5 & | & 3.5 \end{bmatrix}$$

$$\boxed{\beta = 2.6 \qquad \alpha = -0.9}$$

- Interpretation:
  - Think of $\hat{\mathbf{x}}$ as the *approximation* solution to $\mathbf{Ax} = \mathbf{b}$.
  - The error is the difference vector

$$\mathbf{z} = \mathbf{b} - \mathbf{A}\hat{\mathbf{x}}$$

- Since the difference vector is orthogonal to every vector in the column space:

$$\begin{aligned} \mathbf{c}_1 \cdot \mathbf{z} &= 0 \\ \mathbf{c}_2 \cdot \mathbf{z} &= 0 \\ \mathbf{c}_3 \cdot \mathbf{z} &= 0 \end{aligned}$$

Which we'll now write as:

$$\begin{aligned} \mathbf{c}_1 \cdot (\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}) &= 0 \\ \mathbf{c}_2 \cdot (\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}) &= 0 \\ \mathbf{c}_3 \cdot (\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}) &= 0 \end{aligned}$$

- Suppose now that $\mathbf{B}$ is a matrix with the vectors $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3$ as *rows*.

- Then, we can write the above equations as

$$\mathbf{Bz} = \mathbf{0}$$

In-Class Exercise 6: Why is this true?

B is simply matrix rep. of column vect. c_1, c_2, c_3 as rows
since z is the shortest distance between y and Ax it'll be
$\perp$ to all mentioned vect.s.
Thus, the dot product of $c \cdot z = 0$

$$B = \begin{bmatrix} c_1^T \\ c_2^T \\ c_3^T \end{bmatrix} \cdot z = \begin{bmatrix} c_1 z \\ c_2 z \\ c_3 z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = 0$$

there $\vec{z}$ is the Null Space o B

- Next, because $\mathbf{z} = (\mathbf{b} - \mathbf{A}\hat{\mathbf{x}})$

$$\mathbf{B}(\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}) = \mathbf{0}$$

In-Class Exercise 7: What is the relationship between the matrices $\mathbf{A}$ and $\mathbf{B}$?

Here we can see that $B = A^T$
this could be instrumental when there is no exact solution for x in $b - A\hat{x} = 0$

$Ax = b$

$\vec{z}$
$b - Ax = 0 \rightarrow$ no exact solution when A is not a sq. matrix
thus, approx. $\rightarrow$ Least Squares.

Least Squares:

$A^T(b - Ax) = 0$, since $\vec{z}$ is orthogonal to all col. vect of A, it'll be orth. to all row. vect. of A^T

$A^T b - A^T A X = 0$
$A^T A X = A^T b$ suppose $A^T A$ has an inverse

$(A^T A)^{-1} A^T A X = (A^T A)^{-1} A^T b$

$I X = (A^T A)^{-1} A^T b$

$X = (A^T A)^{-1} A^T b$

- We now finally have a direct expression for the approximate solution $\hat{\mathbf{x}}$:

$$\hat{\mathbf{x}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$$

(Provided of course the inverse exists).

- This is called the *least-squares solution* to $\mathbf{Ax} = \mathbf{b}$.
- The method and terminology dates back to Gauss, who started by minimizing $(\mathbf{Ax} - \mathbf{b})^2$.
  ▷ Which results in the same solution.

In-Class Exercise 8: Why does this result in the same solution? That is, why is our solution the solution to minimizing $(\mathbf{Ax} - \mathbf{b})^2$? (Hint: make a simple geometric argument)

Minimizing $y = |Ax - b|^2$ is equivalent to

$$\Rightarrow \frac{dy}{dx} = 2(Ax - b) = 0$$

$\Rightarrow$ We are still trying to find the matrix that'll project A onto b $\Rightarrow$ orthogonality

Also, $(Ax-b)^2$ is the Euc. dist. $\rightarrow$ len of the line
Line segment connecting Ax to b

Minimizing this means finding the shortest length vector. Which is exactly what we get using the Least Squares Approximation

In-Class Exercise 9: Suppose $\mathbf{A}_{m\times n}$ is a matrix with m rows and n columns. What are the dimensions of $\mathbf{A}^T\mathbf{A}$? Verify that the dimensions all work out correctly in the expression $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$.

$A^T A = M^{n\times n}$

$(A^T A)^{-1} = M_1^{n\times n}$

$M_1^{n\times n} \; A^{T\, n\times m} \; b \Rightarrow M_2^{n\times m} \cdot b$

- **Theorem 7.1**: If $\mathbf{A}^{-1}$ exists, then $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} = \mathbf{A}^{-1}\mathbf{b}$.
- This is good news: the least-squares solution *is* the exact solution if the exact solution exists.
- **Theorem 7.2**: If the columns of $\mathbf{A}_{m\times n}$ are linearly independent, then $(\mathbf{A}^T\mathbf{A})^{-1}$ exists.
- Let's understand what 7.2 is saying:
  - If $m < n$ then the column rank (which is the row rank) is less than n.
    - ▷ The columns cannot be linearly independent.
  - If $m \geq n$ then it's possible that the columns are independent.
    - ▷ In which case, least-squares will work.
- Unfortunately, neither result is easy to prove. We'll need to roll up our sleeves.

t's start with the first result, which is the easier of the two:

- We will do this in steps.
- Suppose we could show that

$$(\mathbf{A}^T\mathbf{A})^{-1} \;=\; \mathbf{A}^{-1}(\mathbf{A}^T)^{-1}$$

In-Class Exercise 10: Use the above result to complete the proof of Theorem 7.1.

Let's say $A^{-1}b = X$ which is true when A is invertible

$(A^TA)^{-1}A^T b = X = A^{-1}b$

- We really need to show two things here:
  1. If $\mathbf{A}^{-1}$ exists, so does $(\mathbf{A}^T)^{-1}$.
  2. And that the inverse-transpose multiplication can be switched as above.

In-Class Exercise 11: If $\mathbf{A}^{-1}$ exists, what does this tell us about the dimensions and column rank of $\mathbf{A}$? And what does *that* tell us about the row rank of $\mathbf{A}$, and therefore the column rank of $\mathbf{A}^T$? What can we conclude about whether $(\mathbf{A}^T)^{-1}$ exists?

A must be a square matrix.
The column rank of A must be full, meaning it must equal the number of columns n. For a square matrix, this also means its row rank must be full. The rank of a matrix is the dimension of the vector space spanned by its columns (or rows), and for an invertible matrix, this must be the full dimension of the space

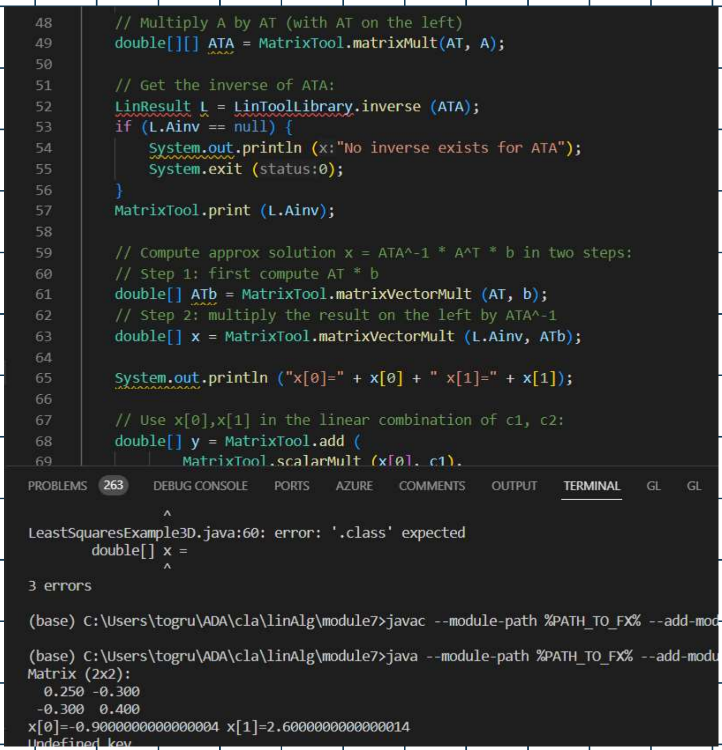
det(A)!=0

The row rank of A is the same as the column rank of AT due to the rank's invariance under transposition. This implies that if A is invertible, then AT also has full rank.

From these properties, we can conclude that if A−1A−1 exists, then:

ATAT must also be a square matrix.
ATAT must have full rank.
ATAT is non-singular.

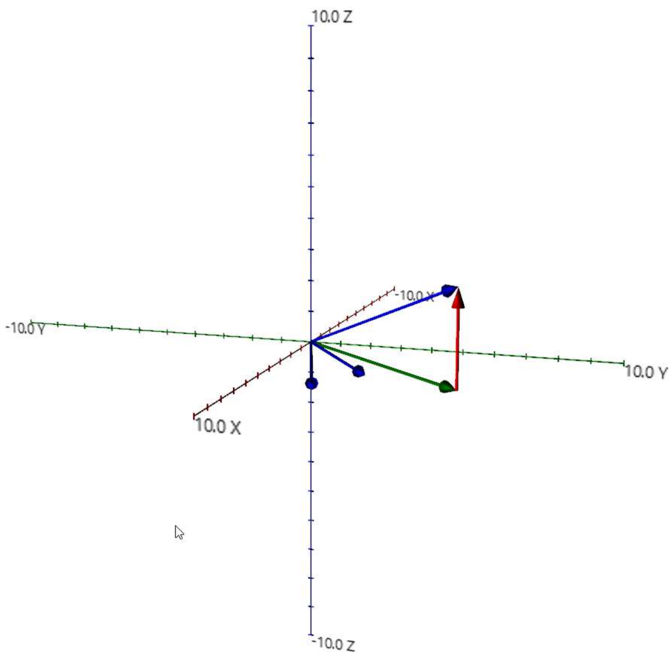
In-Class Exercise 13: Download LeastSquaresExample3D.java to implement the above solution and confirm. You will also need MatrixTool.class (or your own `MatrixTool.java`). Note: this will work only if `lintool.jar` is in your CLASSPATH.



```
// Multiply A by AT (with AT on the left)
double[][] ATA = MatrixTool.matrixMult(AT, A);

// Get the inverse of ATA:
LinResult L = LinToolLibrary.inverse (ATA);
if (L.Ainv == null) {
    System.out.println (x:"No inverse exists for ATA");
    System.exit (status:0);
}
MatrixTool.print (L.Ainv);

// Compute approx solution x = ATA^-1 * A^T * b in two steps:
// Step 1: first compute AT * b
double[] ATb = MatrixTool.matrixVectorMult (AT, b);
// Step 2: multiply the result on the left by ATA^-1
double[] x = MatrixTool.matrixVectorMult (L.Ainv, ATb);

System.out.println ("x[0]=" + x[0] + " x[1]=" + x[1]);

// Use x[0],x[1] in the linear combination of c1, c2:
double[] y = MatrixTool.add (
        MatrixTool.scalarMult (x[0], c1),
```

PROBLEMS 263 DEBUG CONSOLE PORTS AZURE COMMENTS OUTPUT TERMINAL GL GL

```
LeastSquaresExample3D.java:60: error: '.class' expected
        double[] x =
        ^
3 errors

(base) C:\Users\togru\ADA\cla\linAlg\module7>javac --module-path %PATH_TO_FX% --add-mod

(base) C:\Users\togru\ADA\cla\linAlg\module7>java --module-path %PATH_TO_FX% --add-modu
Matrix (2x2):
  0.250 -0.300
 -0.300  0.400
x[0]=-0.9000000000000004 x[1]=2.6000000000000014
Undefined key
```



In-Class Exercise 14: Add the column $\mathbf{c}_3 = (8, 1, 0)$ to the matrix $\mathbf{A}$ in the above code.

1. Explain why the columns are not linearly independent.
2. What happens as a result?

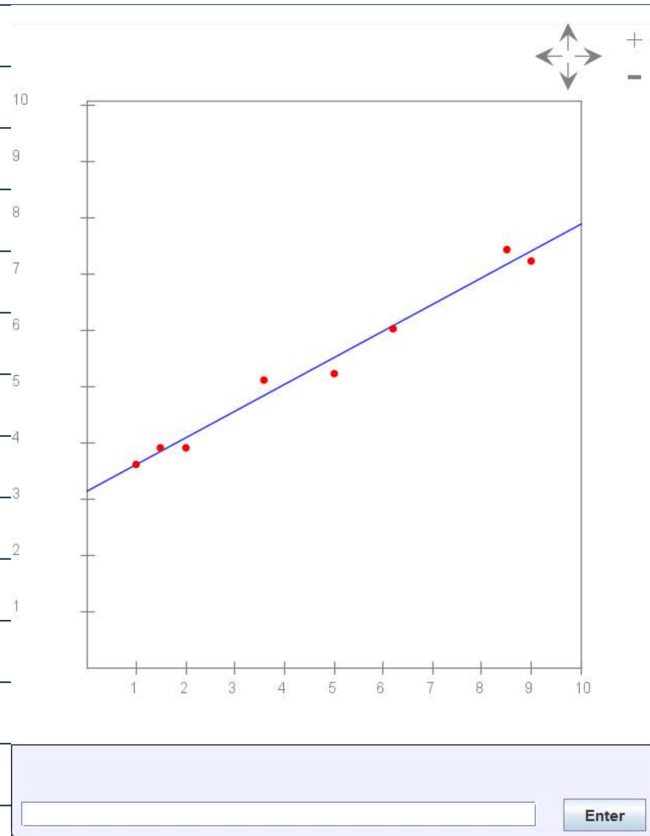
```java
    // Get the inverse of ATA:
    LinResult L = LinToolLibrary.inverse (ATA);
    if (L.Ainv == null) {
        System.out.println (x:"No inverse exists for ATA");
        System.exit (status:0);
    }
    MatrixTool.print (L.Ainv);

    // Compute approx solution x = ATA^-1 * A^T * b in two steps:
    // Step 1: first compute AT * b
    double[] ATb = MatrixTool.matrixVectorMult (AT, b);
    // Step 2: multiply the result on the left by ATA^-1
    double[] x = MatrixTool.matrixVectorMult (L.Ainv, ATb);

    System.out.println ("x[0]=" + x[0] + " x[1]=" + x[1]);

    // Use x[0],x[1] in the linear combination of c1, c2:
    double[] y = MatrixTool.add (
            MatrixTool.scalarMult (x[0], c1),
            MatrixTool.scalarMult (x[1], c2)
            );

    // The difference vector z:
    double[] z = MatrixTool.sub (b, y);

    // Draw y and z:
```

PROBLEMS 264 DEBUG CONSOLE PORTS AZURE COMMENTS OUTPUT TERMINAL GL GL

```
LinTool.LU.solve(): Matrix is singular
```

In-Class Exercise 15: Download RegressionExample.java, compile and execute to view the data. Then un-comment the least-squares part of the code, add code to complete the least-squares calculation. What are the values of $\hat{m}$ and $\hat{c}$?

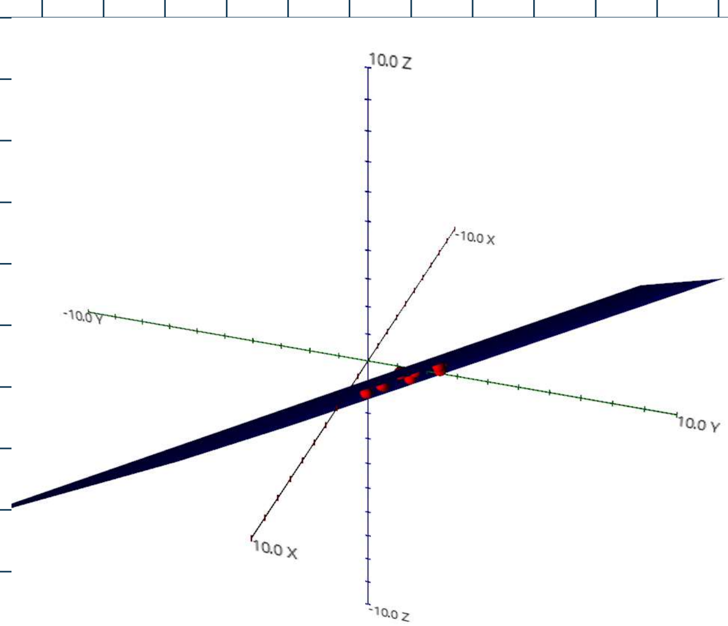
```java
    // INSERT YOUR CODE HERE for least squares
    // Compute the transpose of A:
    double[][] AT  = MatrixTool.transpose (A);

    // Multiply A by AT (with AT on the left)
    double[][] ATA = MatrixTool.matrixMult(AT, A);

    // Get the inverse of ATA:
    LinResult L = LinToolLibrary.inverse (ATA);
    if (L.Ainv == null) {
        System.out.println (x:"No inverse exists for ATA");
        System.exit (status:0);
    }

    // Compute approx solution x = ATA^-1 * A^T * b in two steps:
    // Step 1: first compute AT * b
    double[] ATb = MatrixTool.matrixVectorMult (AT, b);
    // Step 2: multiply the result on the left by ATA^-1
    double[] xhat = MatrixTool.matrixVectorMult (L.Ainv, ATb);

    // xhat[0] = m (slope)
    // xhat[1] = c (y-intercept)
    System.out.println ("xhat[0]=" + xhat[0] + " xhat[1]=" + xhat[1]);

    // Draw the line. Here drawLineFromEquation() expects the
    // line specified in ax+by+c format. To convert from y=mx+c,
    // observe that a=m, b=-1, c=c.
    DrawTool.setEquationLineColor (colorString:"blue");
    DrawTool.drawLineFromEquation (xhat[0], -1, xhat[1]);

    }
}
```



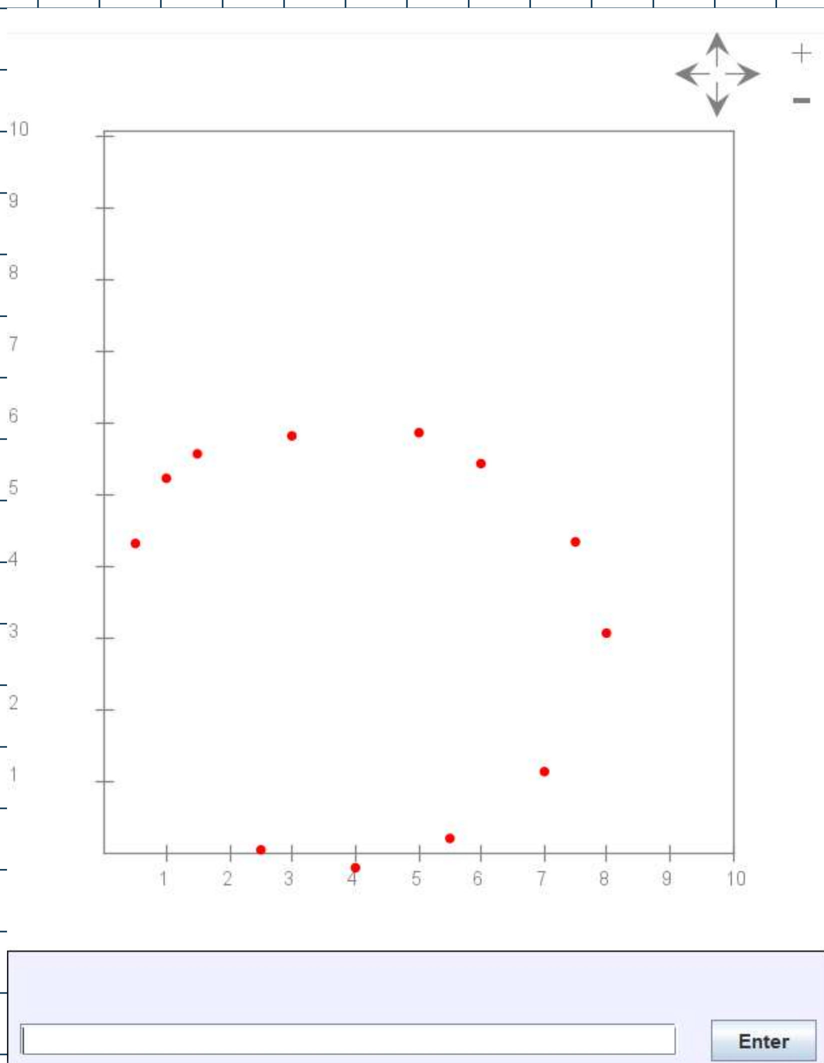

```
(base) C:\Users\togru\ADA\cla\linAlg\module7>java --module-path %PATH_TO_FX% --add-modules javafx.controls RegressionExample.java
xhat[0]=0.4721975746820464 xhat[1]=3.1153911564625876
```

In-Class Exercise 16: Download RegressionExample3D.java, compile and execute to view the data. Then un-comment the least-squares part of the code, add code to complete the least-squares calculation and plot a plane.

```java
// INSERT YOUR CODE HERE for least squares
// Compute the transpose of A:
double[][] AT = MatrixTool.transpose (A);

// Multiply A by AT (with AT on the left)
double[][] ATA = MatrixTool.matrixMult(AT, A);

// Get the inverse of ATA:
LinResult L = LinToolLibrary.inverse (ATA);
if (L.Ainv == null) {
    System.out.println (x:"No inverse exists for ATA");
    System.exit (status:0);
}

// Compute approx solution x = ATA^-1 * A^T * b in two steps:
// Step 1: first compute AT * b
double[] ATb = MatrixTool.matrixVectorMult (AT, b);
// Step 2: multiply the result on the left by ATA^-1
double[] xhat = MatrixTool.matrixVectorMult (L.Ainv, ATb);

MatrixTool.print (xhat);
// Should print a'=-0.499, b'=-0.583, d'=-1.179

// Now we need to convert the least-squares solution to the
// parameters of the plane: a',b',c=1,d'
d3.setDrawColor (Color.BLUE);
d3.drawPlane (xhat[0], xhat[1], 1, xhat[2]);
```



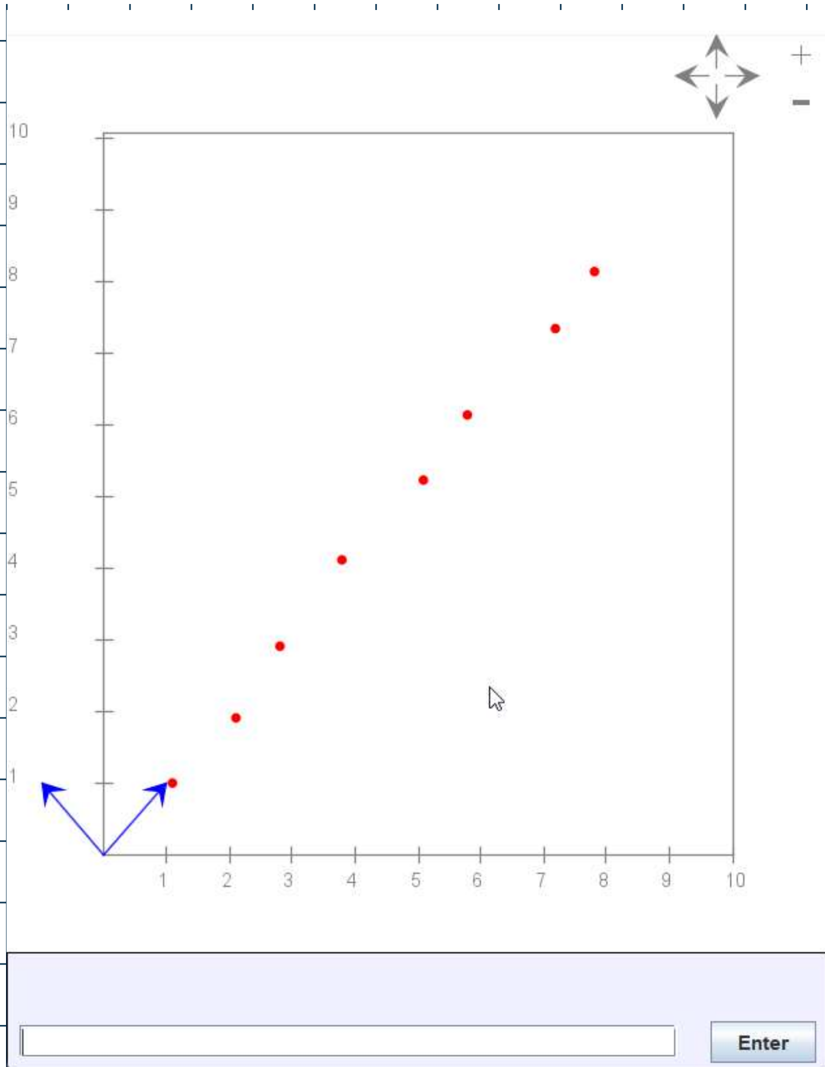
In-Class Exercise 17: Download Ellipse.java, compile and execute to view the data. You'll also need DrawTool.java. (You already have MatrixTool.class). Then un-comment the least-squares part of the code, and see the resulting ellipse.



In-Class Exercise 18: Download ChangeOfBasisExample.java, compile and execute to view the statistics before and after a change-of-basis.

```
(base) C:\Users\togru\ADA\cla\linAlg\module7>java --module-path %PATH_TO_FX% --add-modules javafx.controls ChangeOfBasisExample.java
Before change of basis:
  meanX= 4.462, meanY= 4.575 varX= 5.846, varY= 6.505 covariance=49.260
```

In-Class Exercise 19: Download DrawBasisExample.java, compile and execute to view the points along with the new basis vectors. Notice that the second coordinate is nearly zero.



- Define the vector

$$\mathbf{z} \stackrel{\triangle}{=} \mathbf{w} - \alpha \mathbf{v}$$

- Since $\mathbf{z}$ is orthogonal to $\mathbf{v}$, the dot product is zero:

$$\mathbf{z} \cdot \mathbf{v} \ = \ 0$$

Or

$$(\mathbf{w} - \alpha \mathbf{v}) \cdot \mathbf{v} \ = \ 0$$

Which means

$$\alpha \ = \ \frac{\mathbf{w} \cdot \mathbf{v}}{\mathbf{v} \cdot \mathbf{v}}$$

- Note: α is a *scalar*.

In-Class Exercise 20: Calculate α by hand for the example above. Then, calculate $\mathbf{z}$ and verify that $\mathbf{z} \cdot \mathbf{v} = 0$.

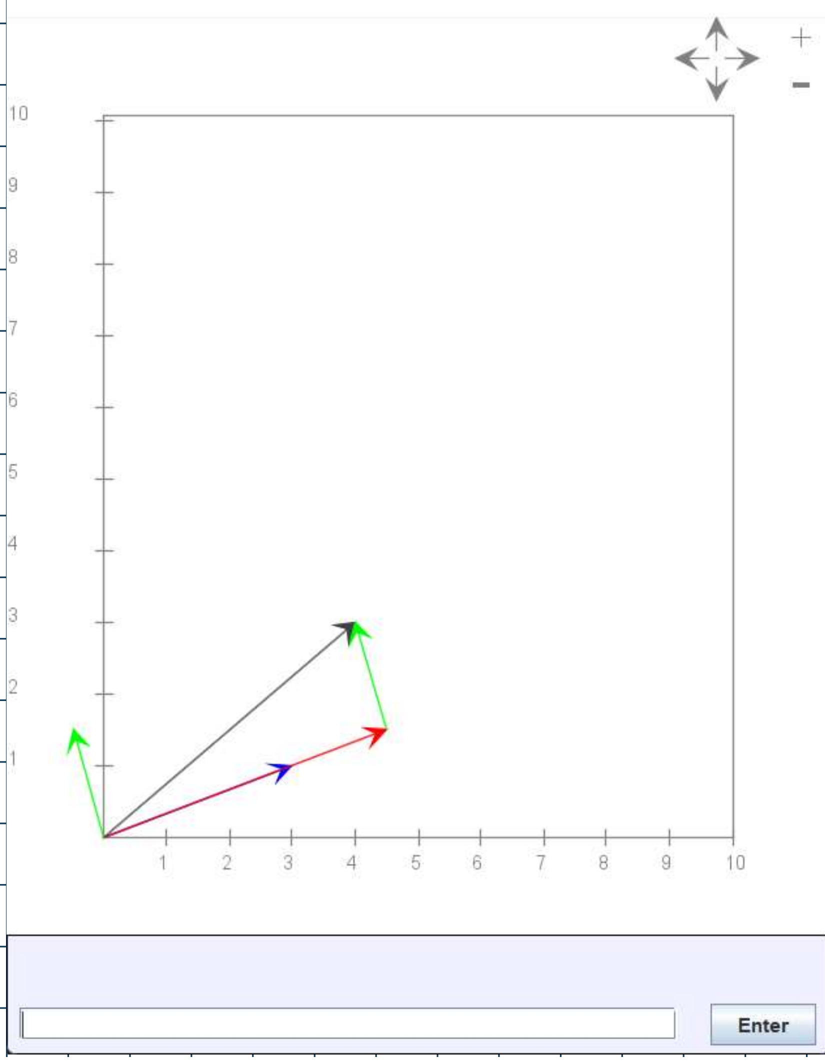
$$\begin{aligned} \mathbf{w} &= (4,3) \\ \mathbf{v} &= (6,2) \end{aligned}$$

$$\alpha = \frac{24+6}{40} = 0.75$$

$$z = w - \alpha v = \begin{bmatrix} 4 \\ 3 \end{bmatrix} - 0.75 \begin{bmatrix} 6 \\ 2 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 1.5 \end{bmatrix}$$

$$\boxed{z \cdot v = -3 + 3 = 0}$$

In-Class Exercise 21: For the same $\mathbf{w}$ as above but with $\mathbf{v} = (3,1)$ use the code in ProjectionExample.java to compute $\mathbf{y} = \text{proj}_{\mathbf{v}}(\mathbf{w})$. What is the additional arrow (starting at the origin)?



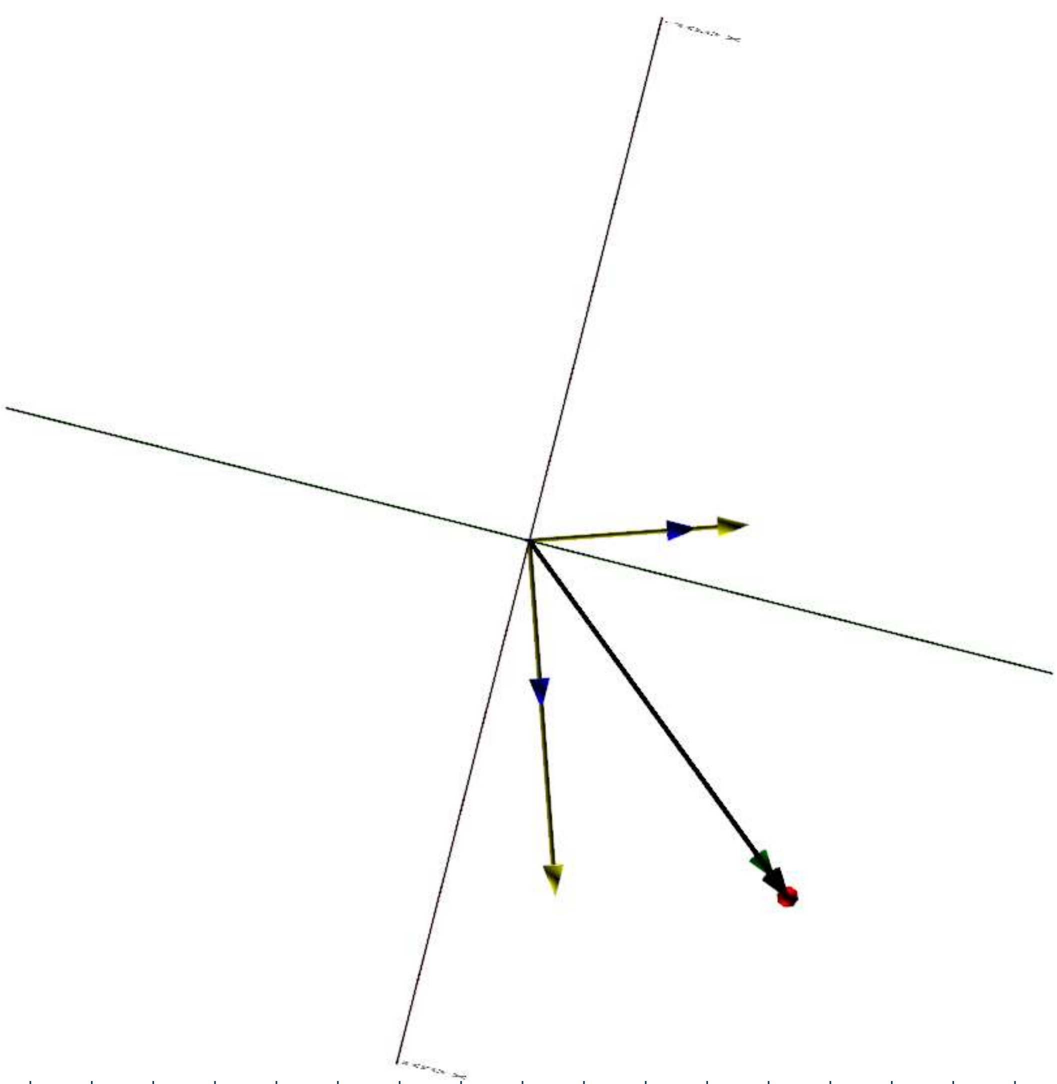
In-Class Exercise 22: Fill in the key step in going from $(\mathbf{w} - \alpha_1 \mathbf{v}_1 - \alpha_2 \mathbf{v}_2) \cdot \mathbf{v}_1 = 0$ to $\alpha_1 = \frac{\mathbf{w} \cdot \mathbf{v}_1}{\mathbf{v}_1 \cdot \mathbf{v}_1}$, and show why $\mathbf{v}_2$ disappears altogether in the expression for α_1. Download and execute Projection3D.java to confirm that the code contains the above calculations. Change the view to look down the z-axis and examine the stretched versions of $\mathbf{v}_1$ and $\mathbf{v}_2$. What do these stretched versions add up to?

$$w - \alpha_1 v_1 - \alpha_2 v_2 = 0$$

$$w = \alpha_1 v_1 + \alpha_2 v_2$$

$$\alpha_1 = \frac{w - \alpha_2 v_2}{v_1} \cdot \frac{v_1}{v_1} = \frac{w v_1 - \alpha_2 v_2 v_1}{v_1 v_1}$$

$$\Rightarrow \frac{w v_1}{v_1 v_1}, \quad \text{since} \quad v_1 \cdot v_2 = 0$$



In-Class Exercise 23: Suppose $\mathbf{w} = (4, 3)$, $\mathbf{v}_1 = (6, 2)$ and $\mathbf{v}_2 = (-1, 3)$. Do the following by hand:

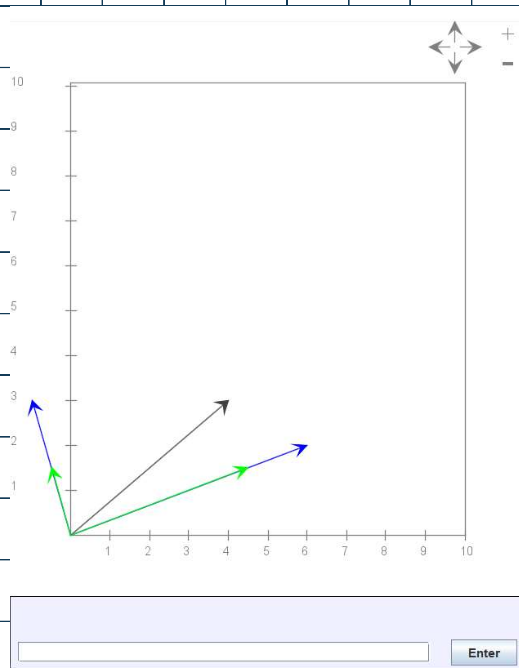
1. Show $\mathbf{v}_1$ and $\mathbf{v}_2$ form an orthogonal basis.
2. Find the coordinates of $\mathbf{w}$ in the basis.
3. Find the projection vectors.
4. Add the projection vectors to get $\mathbf{w}$.

Confirm your calculations by adding code to ProjectionExample2.java.

```
        // INSERT YOUR CODE to compute the coordinates alpha1,alpha2
        // You should get alpha1=0.75, alpha2=0.5

        // Coordinates:
        double alpha1 = MatrixTool.dotProduct(w,v1) / MatrixTool.dotProduct(v1,v1);
        double alpha2 = MatrixTool.dotProduct(w,v2) / MatrixTool.dotProduct(v2,v2);
        // Projections:
        double[] y1 = MatrixTool.scalarMult (alpha1, v1);
        double[] y2 = MatrixTool.scalarMult (alpha2, v2);

        System.out.println ("alpha1=" + alpha1 + "  alpha2=" + alpha2);

        DrawTool.setArrowColor (colorString:"green");
        DrawTool.drawVector (y1);
        DrawTool.drawVector (y2);

        // Confirm:
        double v1Dotv2 = MatrixTool.dotProduct (v1,v2);
        System.out.println ("v1 dot v2 = " + v1Dotv2);

        double[] w2 = MatrixTool.add (y1,y2);
        System.out.println ("w2: (" + w2[0] + "," + w2[1] + ")");
        }

}
```

PROBLEMS 339 DEBUG CONSOLE PORTS AZURE COMMENTS OUTPUT TERMINAL

```
(base) C:\Users\togru\ADA\cla\linAlg\module7>javac --module-path %PATH_TO_FX%

(base) C:\Users\togru\ADA\cla\linAlg\module7>java --module-path %PATH_TO_FX%
alpha1=0.75  alpha2=0.5
v1 dot v2 = 0.0
w2: (4.0,3.0)
```



The G-S algorithm tries to solve the following problem:

- We are given a collection of linearly independent vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n$.
- We'll call the space spanned by these vectors

$$\mathbf{W} \;=\; \text{span}(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n)$$

- Note: if they are not linearly independent, we can easily find an equivalent collection of linearly independent vectors to span $\mathbf{W}$ and work with those.

In-Class Exercise 24: How do we do this?

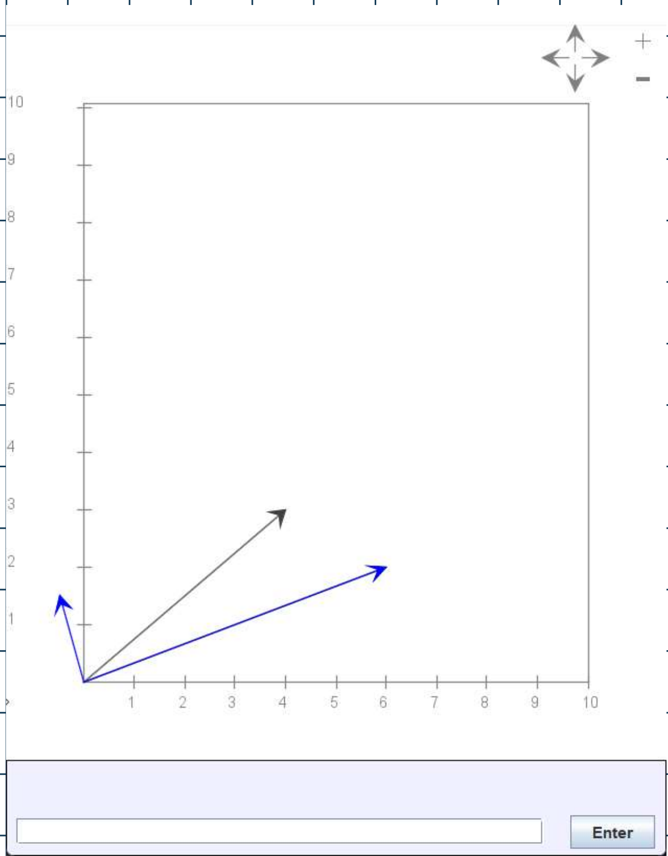
Take the RREF of the matrix where w_i are the column vectors. The pivot columns in this case will be linearly independent

- The goal: find an *orthogonal* set of vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ to span $\mathbf{W}$.

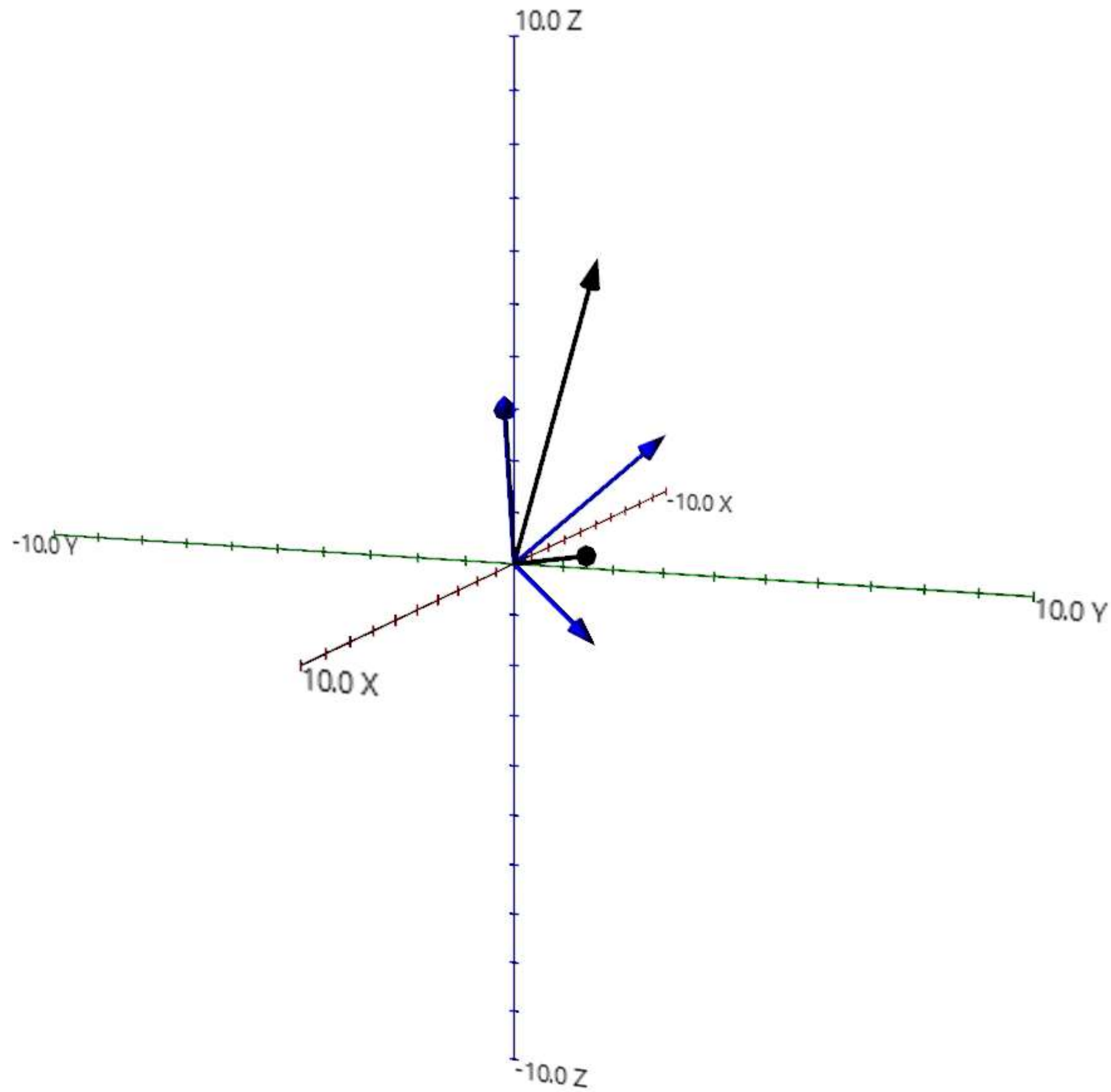
In-Class Exercise 25: How do we know that exactly n $\mathbf{v}_i$'s are needed?

By definition, the dimension of a vector space is the number of vectors in a basis for that space. Since a basis is a set of linearly independent vectors that spans the space, any set of vectors that spans W and has the same number of vectors as the dimension of W must also be a basis for W

In-Class Exercise 26: Download, compile and execute GramSchmidt.java to first draw the vectors $\mathbf{w}_1 = (6,2)$, $\mathbf{w}_2 = (4,3)$. Then, examine the section that performs the projection-subtraction to compute $\mathbf{v}_2$.



In-Class Exercise 27: Download, compile and execute GramSchmidt3D.java to first draw three vectors $\mathbf{w}_1 = (6,2,4)$, $\mathbf{w}_2 = (4,3,1)$, $\mathbf{w}_3 = (1,2,6)$. Then, un-comment the section that performs the sequence of projection-subtractions to compute orthogonal vectors $\mathbf{v}_2, \mathbf{v}_3$.



In-Class Exercise 31: Prove the following:

1. $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$
2. If if $\mathbf{A}^{-1}$ exists then $(\mathbf{A}^T)^{-1}$ exists and $(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T$

$A = \mathbb{R}^{m\times n}$, $B = \mathbb{R}^{n\times p}$ $AB = \mathbb{R}^{m\times p}$

$$(AB)_{ji} = \sum_{k=1}^{n} A_{jk} \, B_{ki}$$

$$\left((AB)^T\right)_{ij} = (AB)_{ji} = \sum_{k=1}^{n} A_{jk} \, B_{ki}$$

$$\left((BA)^T\right)_{ij} = \sum_{k=1}^{n} B^T_{ik} \, A^T_{kj} = \boxed{\sum_{k=1}^{n} B_{ki} \cdot A_{jk}}$$