Programming Homework 1

Matthis Brocheton

Problem 1

A)

I used the normal equation method to find the parameters of the polynomial W_1,W_2,W_3, for a polynomial of order $m = 2$, resulting in a 3×3 matrix.

normal equation:

$W = (X^T X)^{-1} X^T Y$

To find the inverse (A^{-1}) matrix I used the adjugate (or cofactor) method.

Step 1:

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

To find the determinant

$$\det(A) = a(ei - fh) - b(di - fg) + c(dh - eg)$$

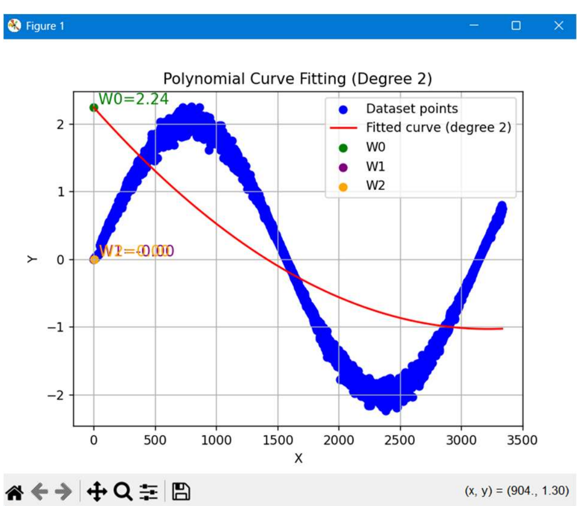
Step 2 :

$$C = \begin{bmatrix} e*i - f*h & -(b*i - c*h) & b*f - c*e \\ -(d*i - f*g) & a*i - c*g & -(a*f - c*d) \\ d*h - e*g & -(a*h - b*g) & a*e - b*d \end{bmatrix}$$

Step 3:

$$adj(A) = \mathrm{C}^{\mathrm{T}}$$

$$A^{-1} = \frac{1}{\det(\mathrm{A})} \times adj(A)$$



W1 = 2.2394943784624255

W2 = -0.0020304497052723164

W3 = 3.1524947624757616e-07

B)

I used the normal equation to find the parameters of the polynomial.

normal equation:

$$W = (X^T X)^{-1} X^T Y$$

To find the inverse (A^{-1}) matrix I used the gauss jordan inverse.

$$\mathrm{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

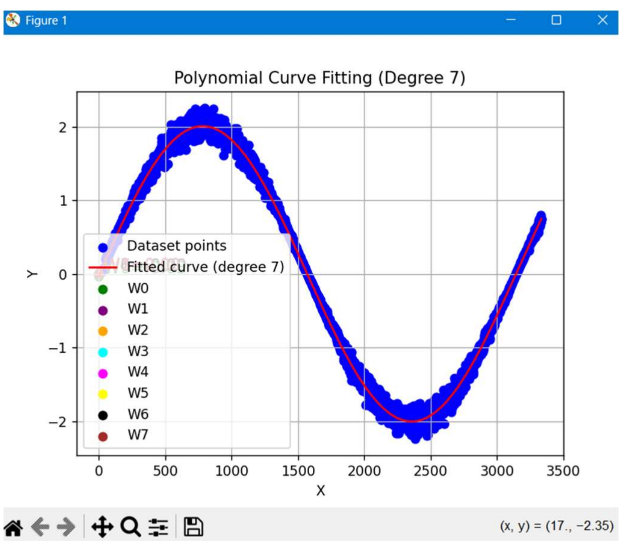
$$[\mathrm{A}|\mathrm{I}] = \left[\begin{array}{ccc|ccc} a_{11} & a_{12} & a_{13} & 1 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 1 & 0 \\ a_{31} & a_{32} & a_{33} & 0 & 0 & 1 \end{array}\right]$$

Transform the first element a_{11} into 1 by dividing the first row.

Transform the other elements in the first column into 0 by subtracting the first row multiplied by a suitable factor.

Repeat the process for the other columns until you obtain the identity matrix on the left.

Result: $[\, I \mid A^{-1} \,] \rightarrow A^{-1}$

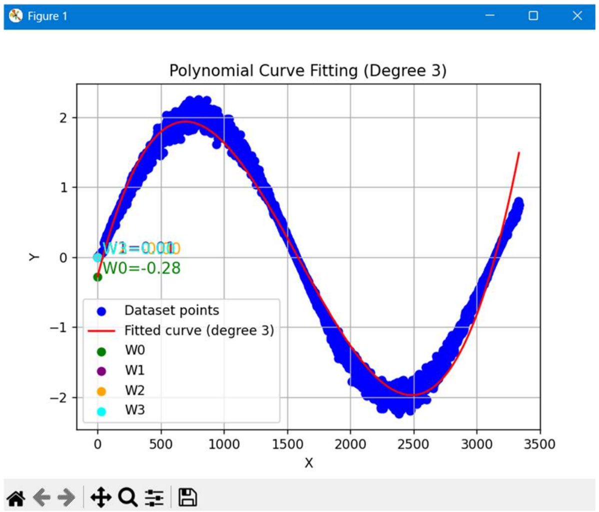


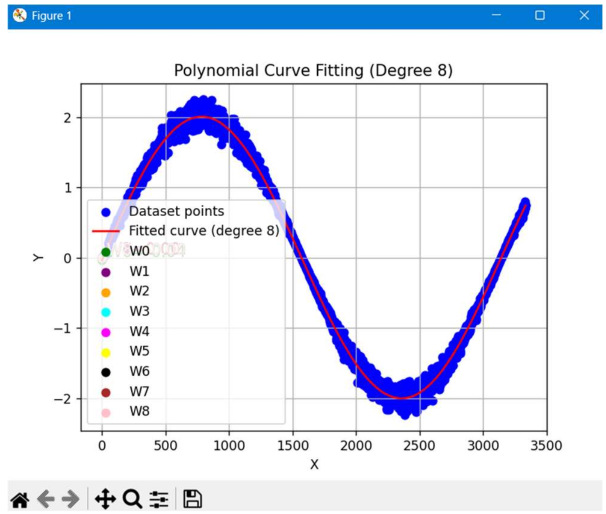
The appropriate order for this polynomial curve is $m = 7$.

C)

I use the formula $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_1 - \hat{y}_2)^2$

n → The total number of data points (samples).
y_1→ The actual (true) value of the dependent variable for the i-th data point.
$\hat{y}_2$→ The predicted value of the dependent variable for the i-th data point.





This is the average sum of squared errors:

MSE for $m = 3$: 0.03984

MSE for $m = 8$: 0.00443

Problem 2

A)

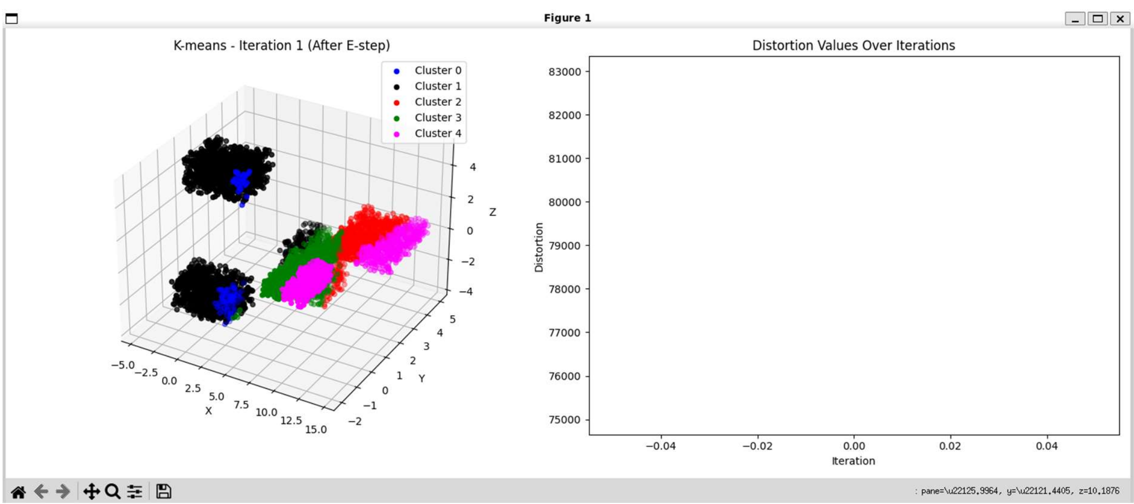
These are the centroids:

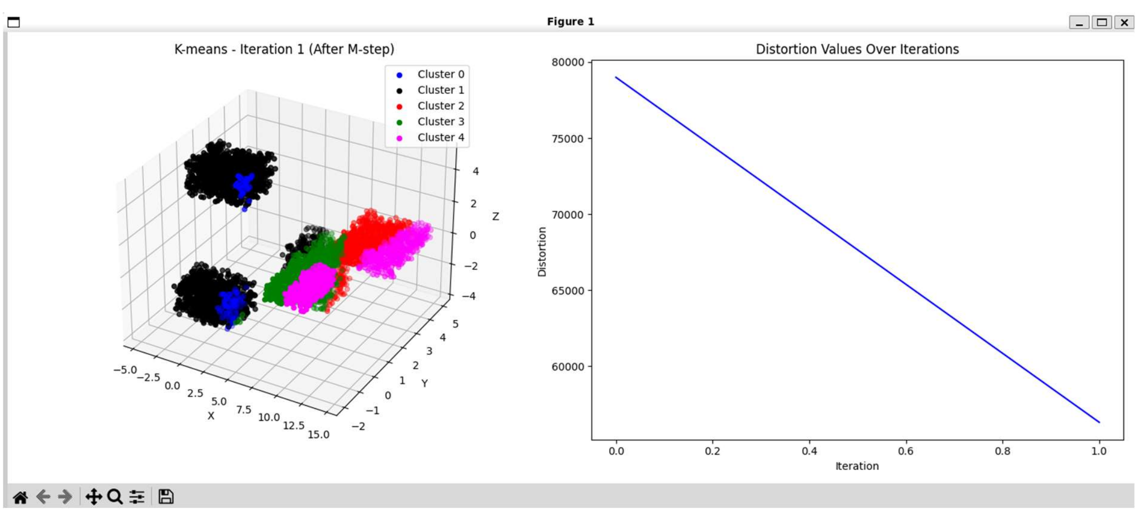
$$y_1 = m_1 = \begin{bmatrix} 4.0 \\ -0.5 \\ 2.0 \end{bmatrix}, \quad y_2 = m_2 = \begin{bmatrix} 2.0 \\ 2.5 \\ 1.0 \end{bmatrix}, \quad y_3 = m_3 = \begin{bmatrix} 10.0 \\ 2.0 \\ -1.0 \end{bmatrix}$$

$$y_4 = m_4 = \begin{bmatrix} 7.0 \\ 0.5 \\ -0.5 \end{bmatrix}, \quad y_5 = m_5 = \begin{bmatrix} 12.0 \\ 1.0 \\ -1.0 \end{bmatrix}$$

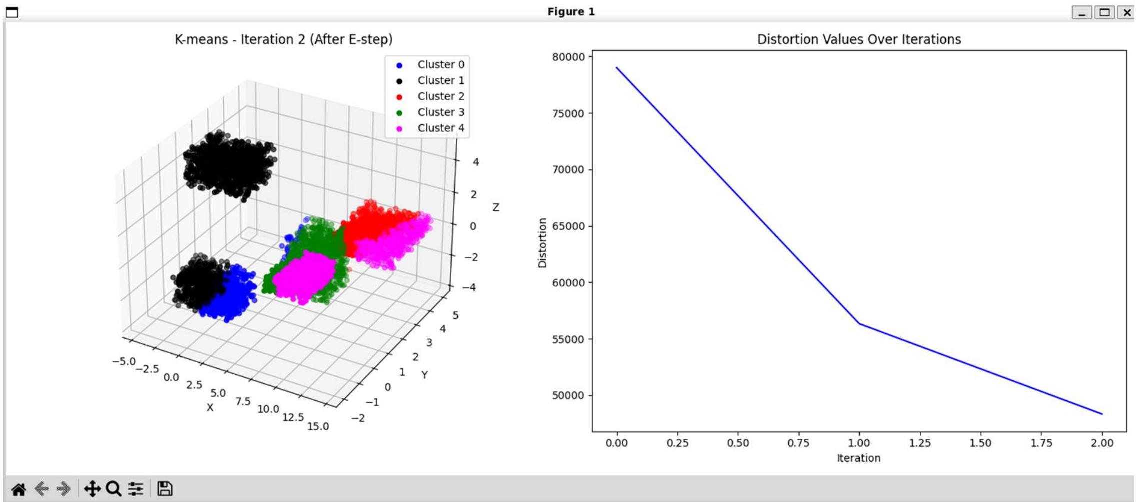
This is the result of the classification of the dataset points using the k-means = 5 algorithm, with the Euclidean distance as the distortion function.

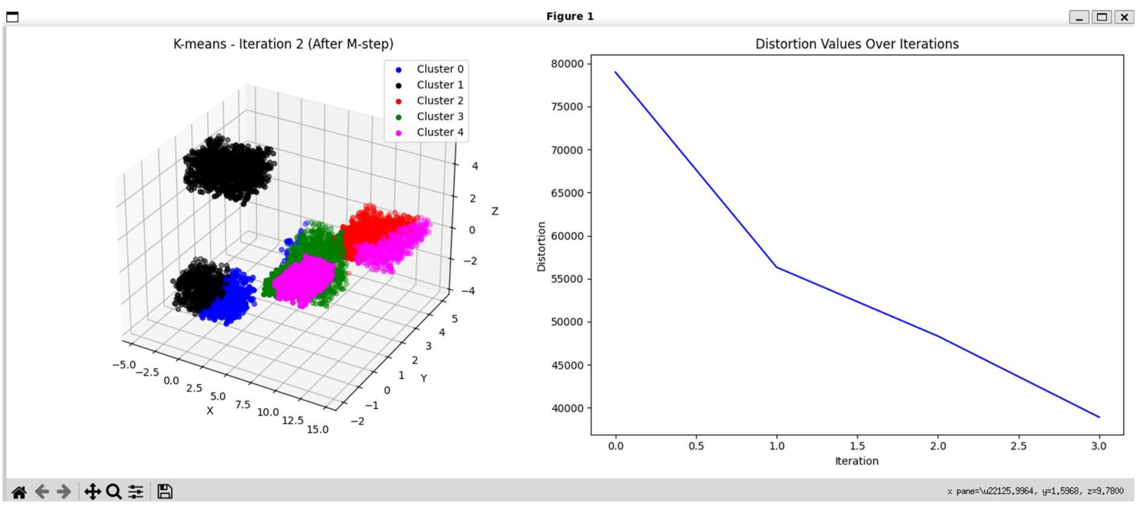
1er iteration:



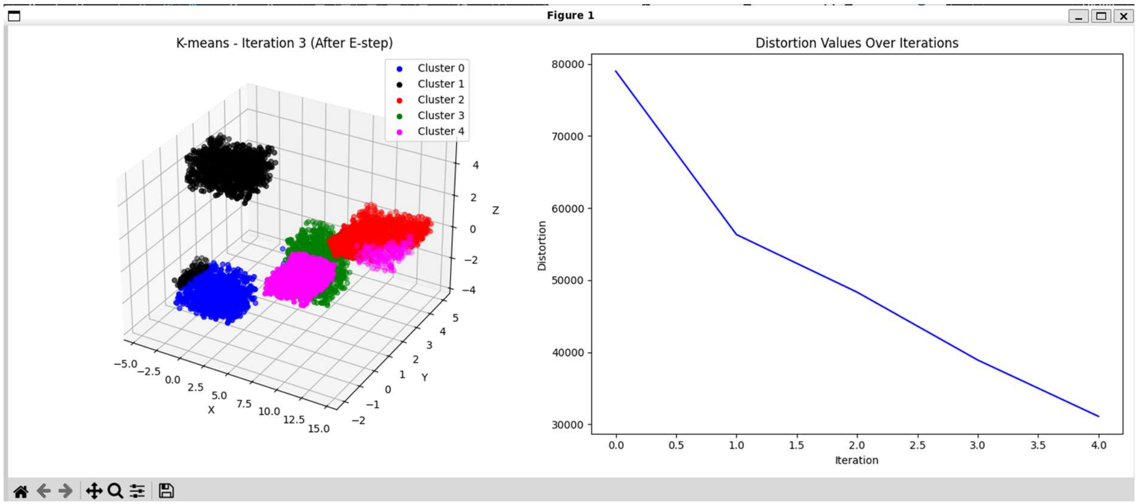


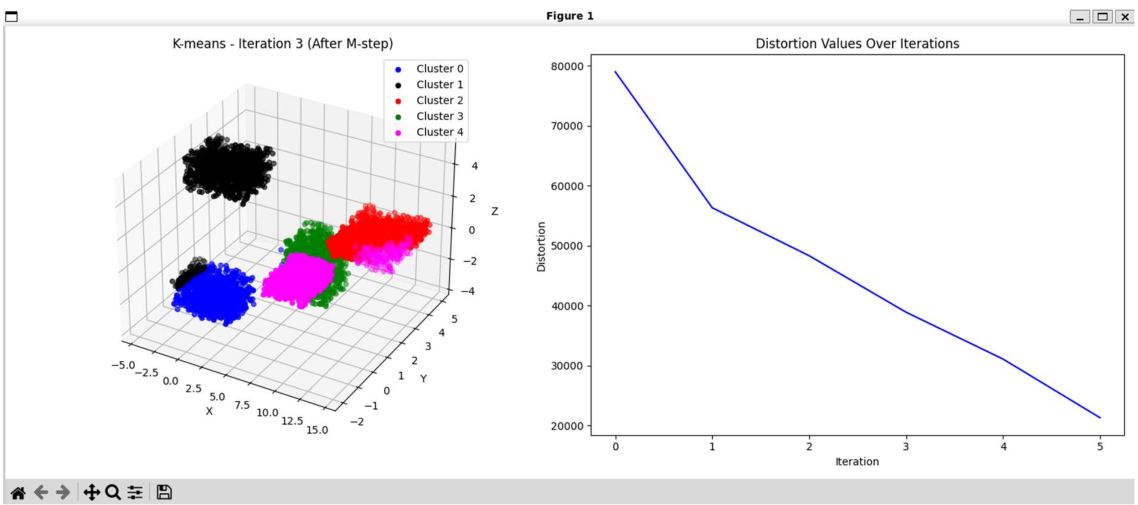
2eme iteration:



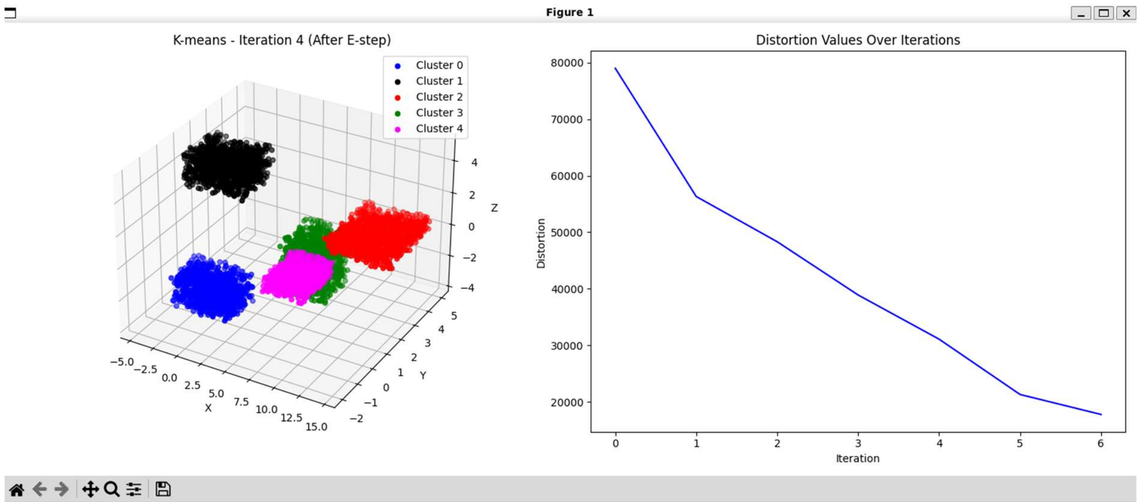


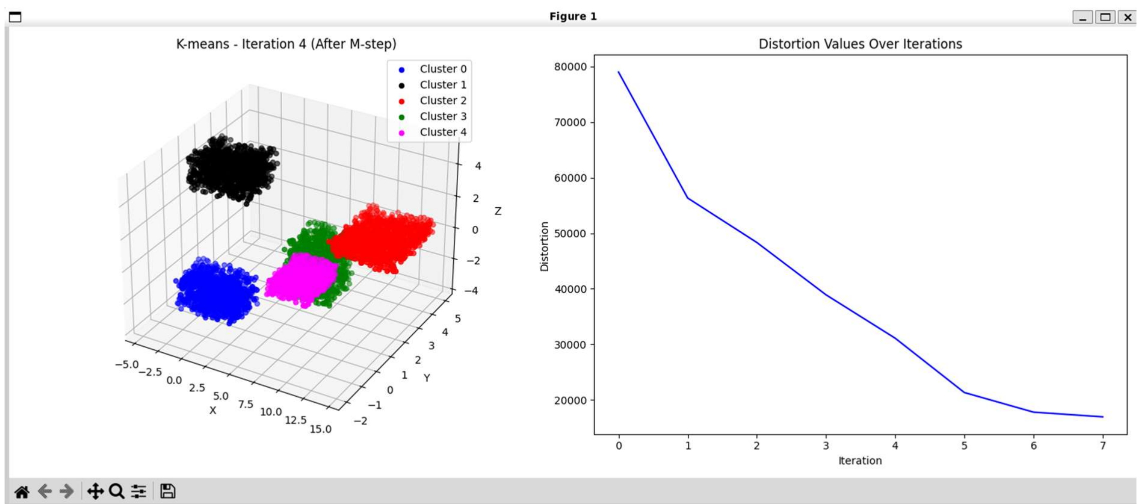
3eme iteration:



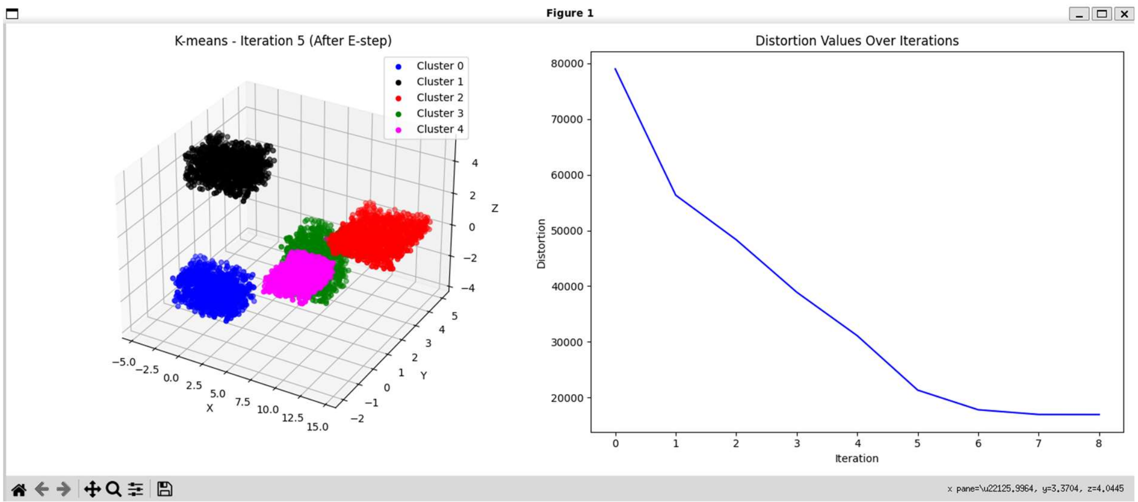


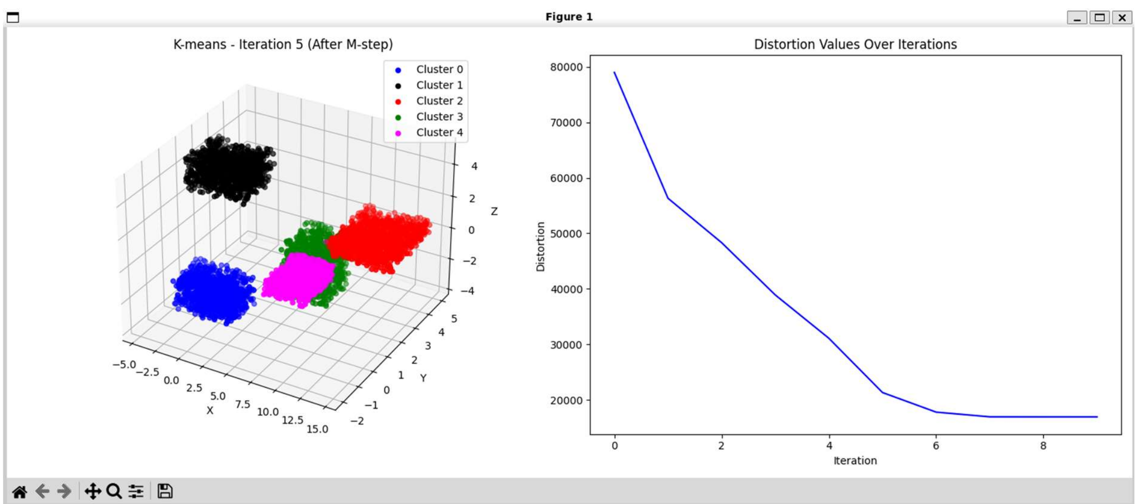
4eme iteration:



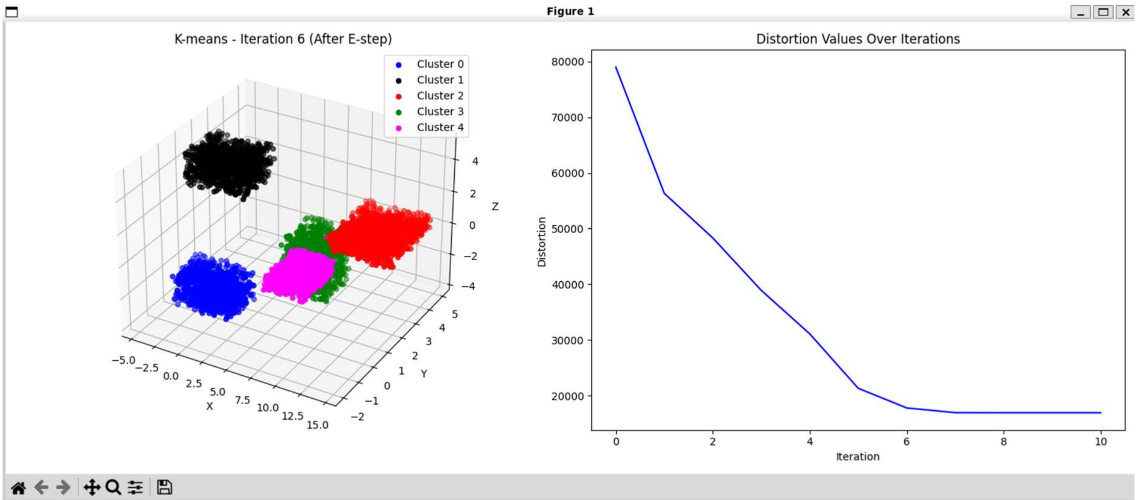


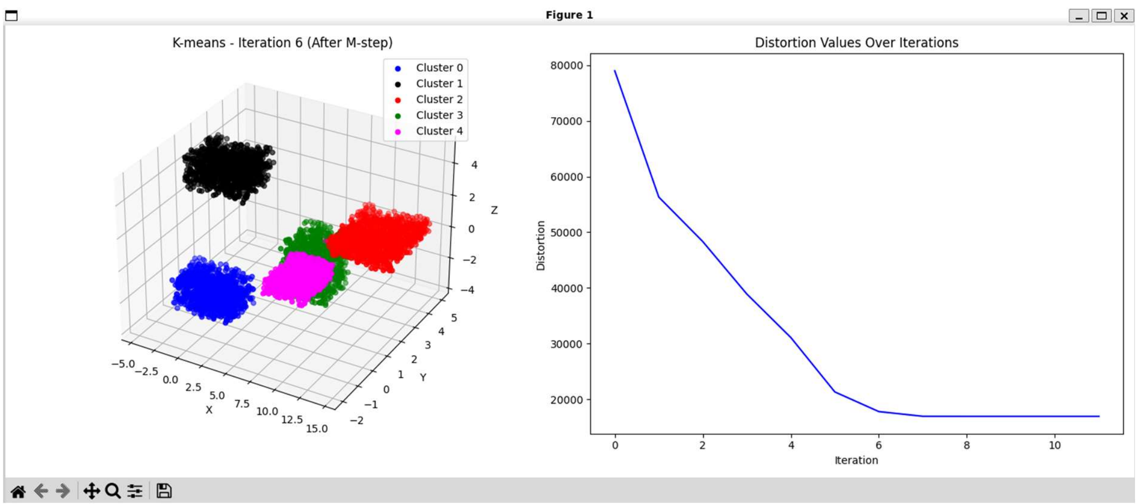
5eme iteration:



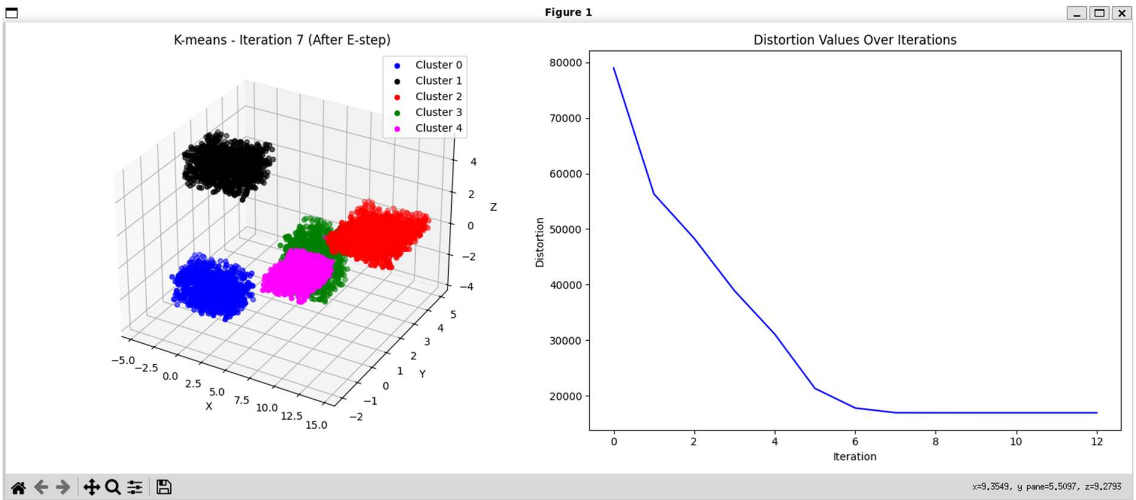


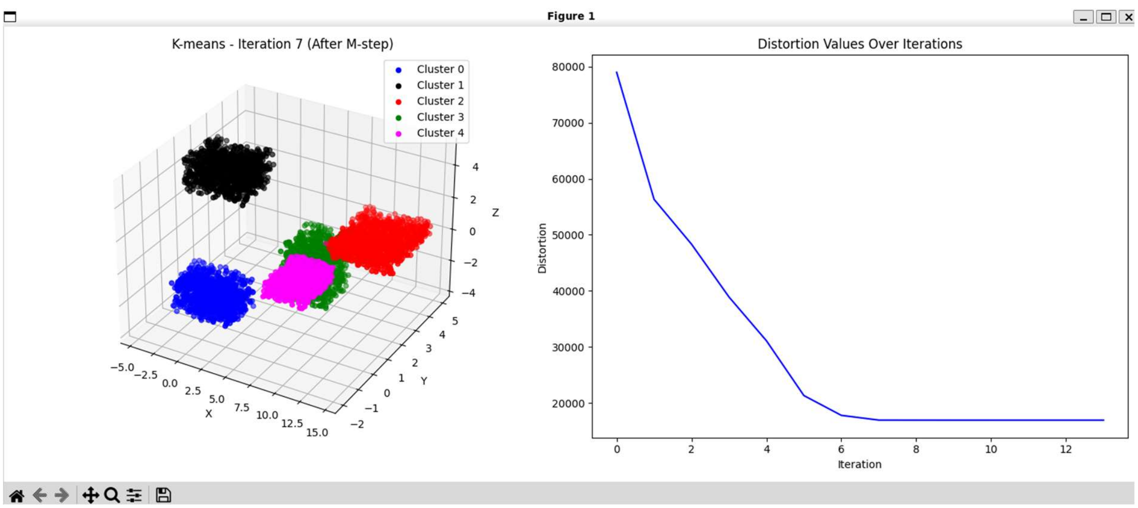
6er iteration:





7eme iteration:





B)

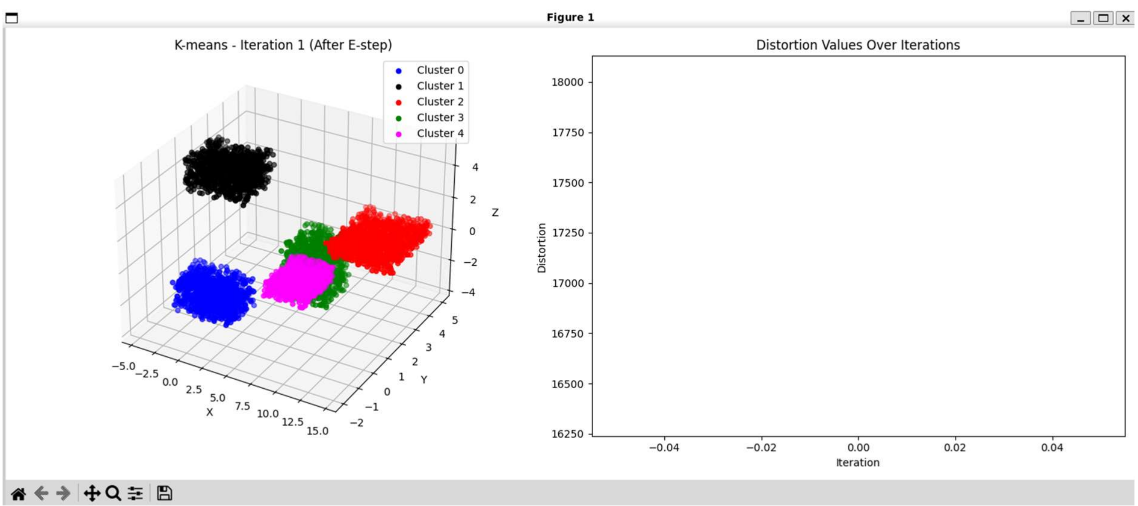
These are the centroids:

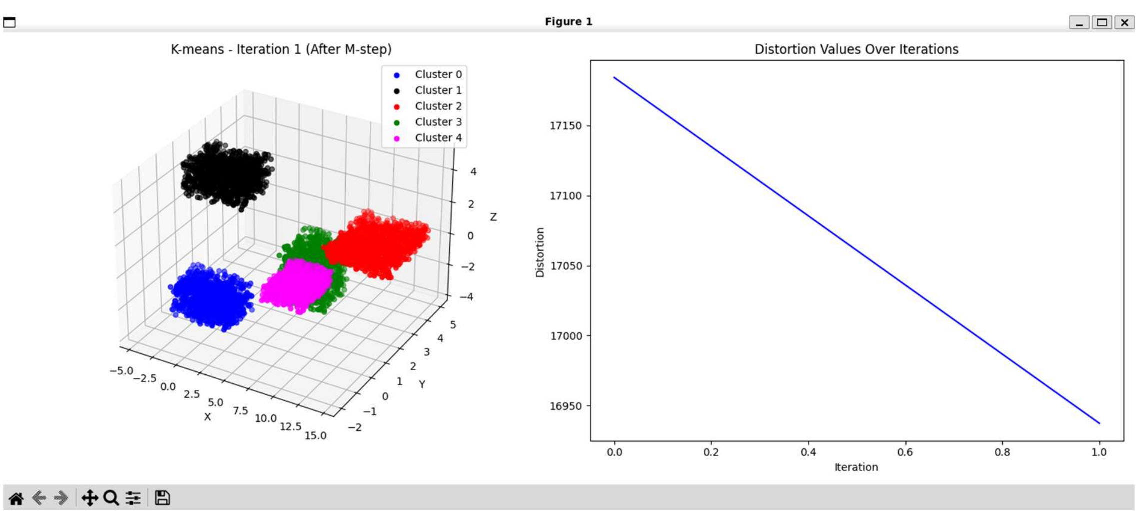
$$y_1 = m_1 = \begin{bmatrix} 0.0 \\ -0.3 \\ -2.0 \end{bmatrix}, \quad y_2 = m_2 = \begin{bmatrix} -1.3 \\ 1.5 \\ 4.0 \end{bmatrix}, \quad y_3 = m_3 = \begin{bmatrix} 11.3 \\ 3.0 \\ 0.2 \end{bmatrix}$$

$$y_4 = m_4 = \begin{bmatrix} 5.7 \\ 3.0 \\ -2.0 \end{bmatrix}, \quad y_5 = m_5 = \begin{bmatrix} 10.0 \\ -1.0 \\ 1.2 \end{bmatrix}$$

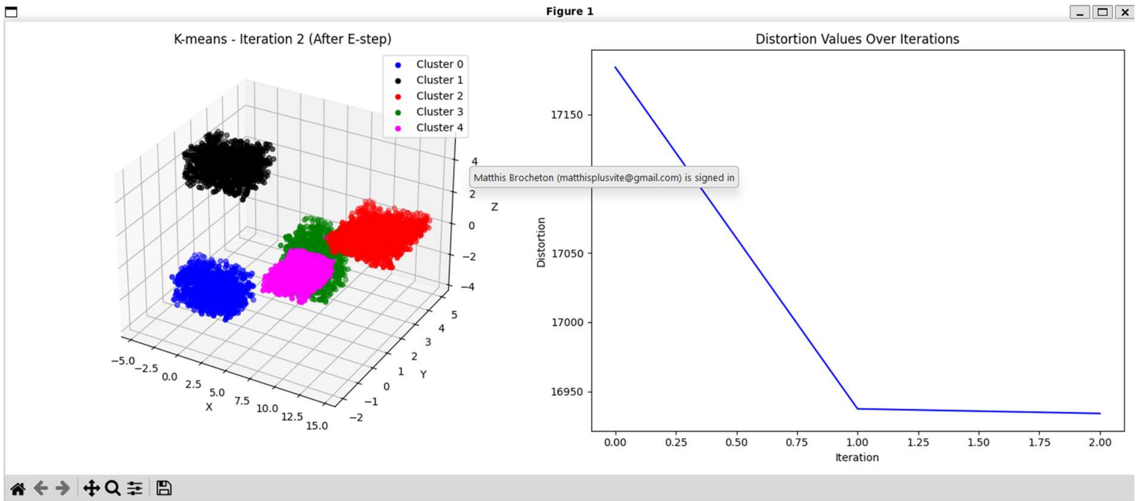
This is the behavior of the K-means (K = 5):

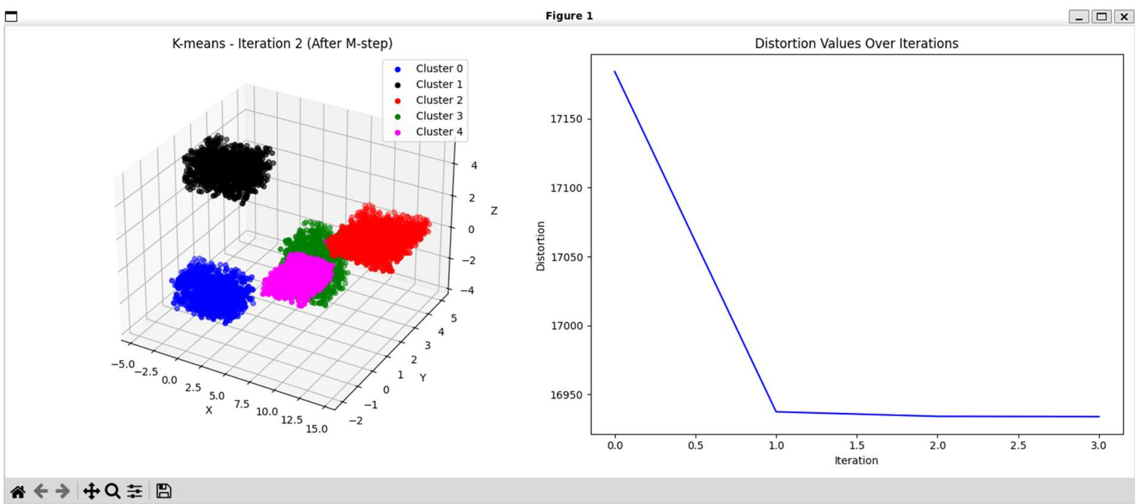
1er iteration:



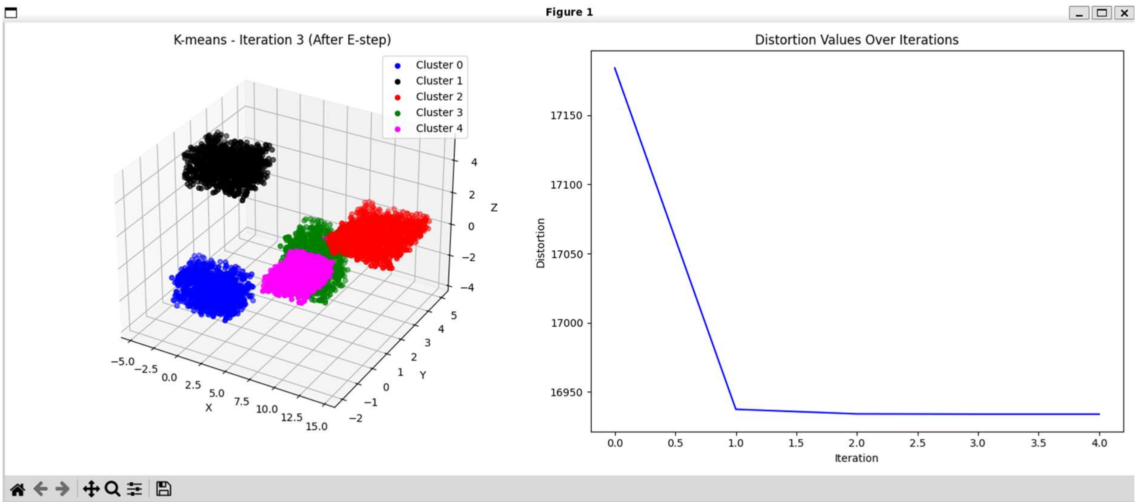


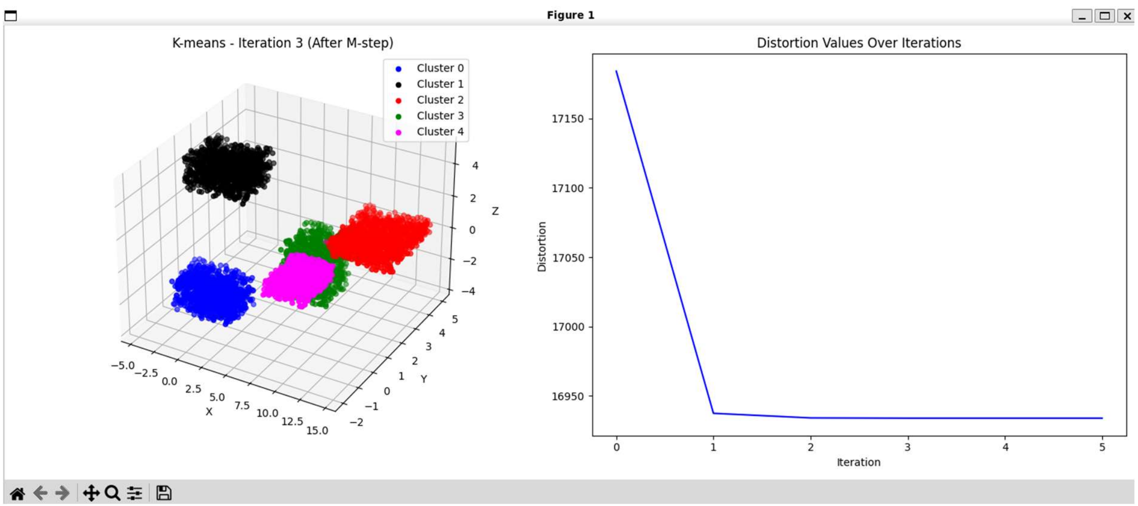
2eme iteration:





3eme iteration:





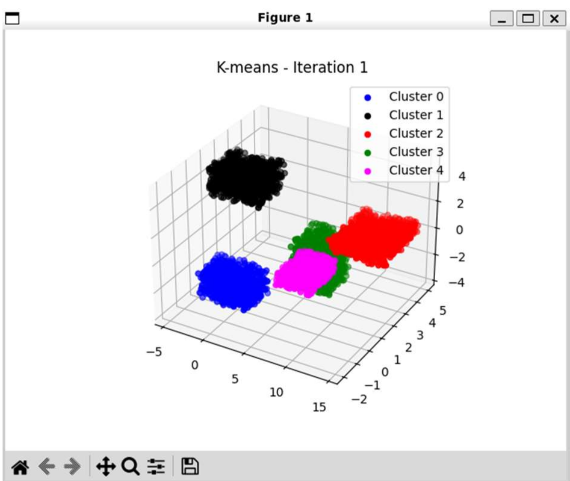
C)

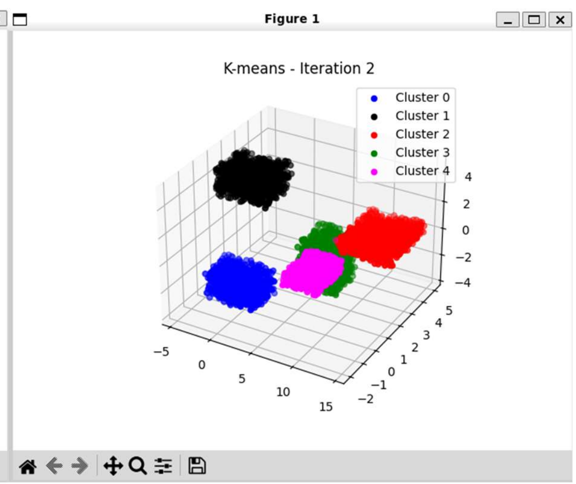
These are the centroids:

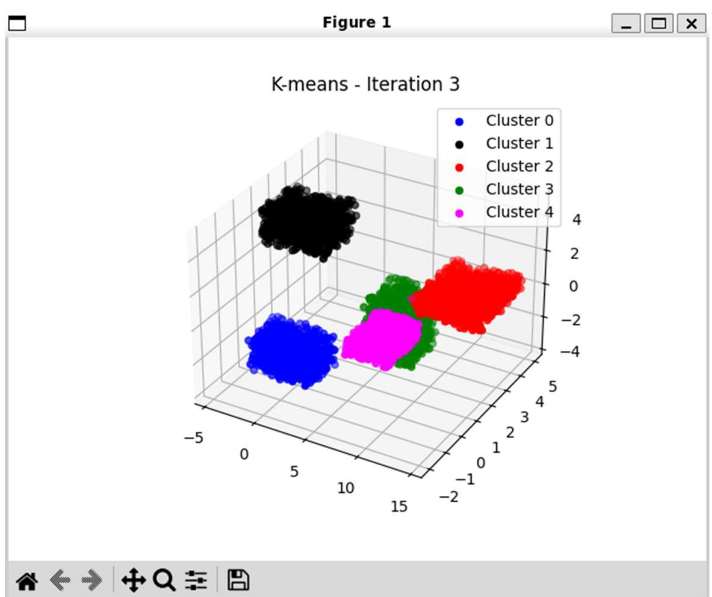
$$y_1 = m_1 = \begin{bmatrix} 0.0 \\ -0.3 \\ -2.0 \end{bmatrix}, \quad y_2 = m_2 = \begin{bmatrix} -1.3 \\ 1.5 \\ 4.0 \end{bmatrix}, \quad y_3 = m_3 = \begin{bmatrix} 11.3 \\ 3.0 \\ 0.2 \end{bmatrix}$$

$$y_4 = m_4 = \begin{bmatrix} 5.7 \\ 3.0 \\ -2.0 \end{bmatrix}, \quad y_5 = m_5 = \begin{bmatrix} 10.0 \\ -1.0 \\ 1.2 \end{bmatrix}$$

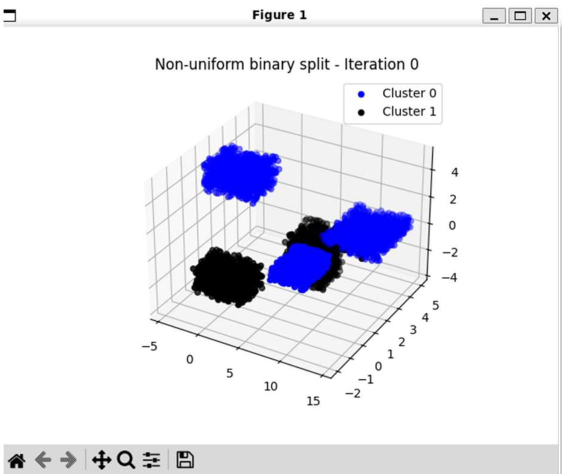
Whereas the **k-means algorithm** was able, from the very first iteration, to classify the dataset into clusters with a high degree of accuracy, convergence was reached quickly in only a few steps.

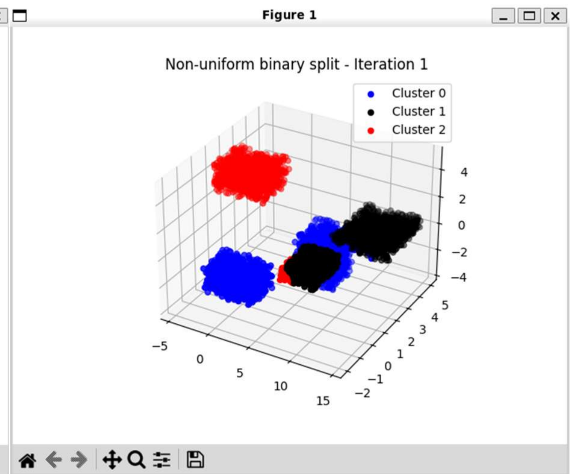


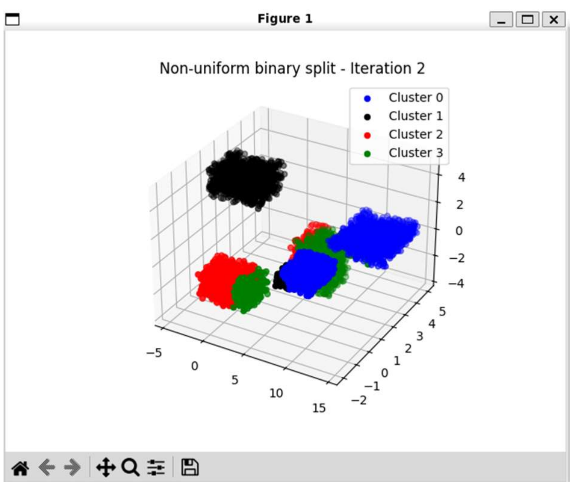


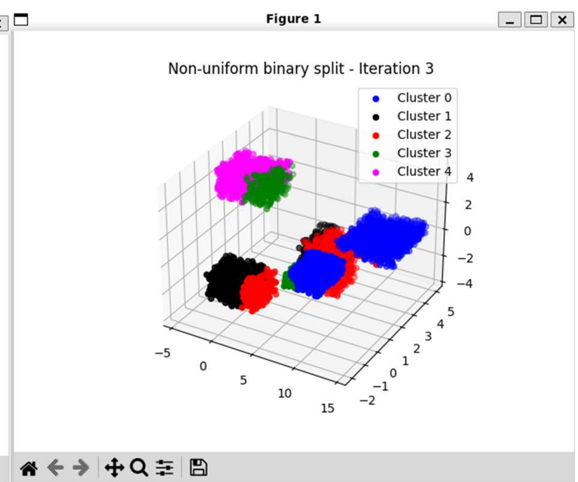


In contrast, the **non-uniform binary split algorithm** performs clustering in a **hierarchical and incremental** manner. As shown in the sequence of figures, the algorithm begins by splitting the dataset into two clusters (iteration 0), and then continues subdividing them in the subsequent iterations.









In conclusion, while **k-means** is fast and effective for well-separated and roughly spherical clusters, the **non-uniform binary split** method offers more flexibility at the cost of longer convergence times.