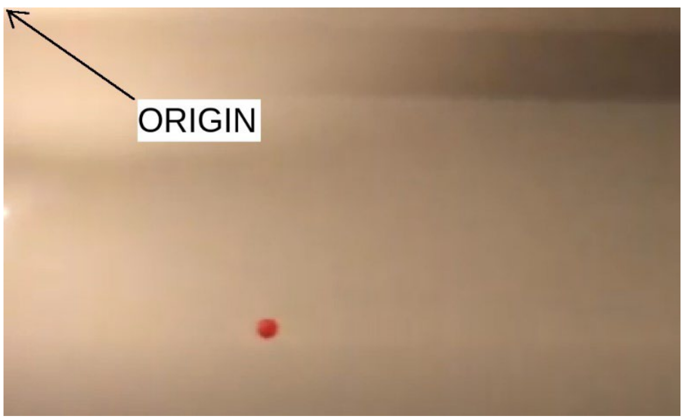
Problem #1:

In the given video, a red ball is thrown against a wall. Assuming that the trajectory of the all follows the equation of a parabola:

1. **Detect and plot the pixel coordinates of the center point of the ball in the video. [10]**
   (Hint: Read the video using OpenCV's inbuilt function. For each frame, filter the red channel)

2. **Use Standard Least Squares to fit a curve to the extracted coordinates. For the estimated parabola you must,**
   a. **Print the equation of the curve. [5]**
   b. **Plot the data with your best fit curve. [5]**

3. **Assuming that the origin of the video is at the top-left of the frame as shown below, compute the x-coordinate of the ball's landing spot in pixels, if the y-coordinate of the landing spot is defined as 300 pixels greater than its first detected location. [10]**



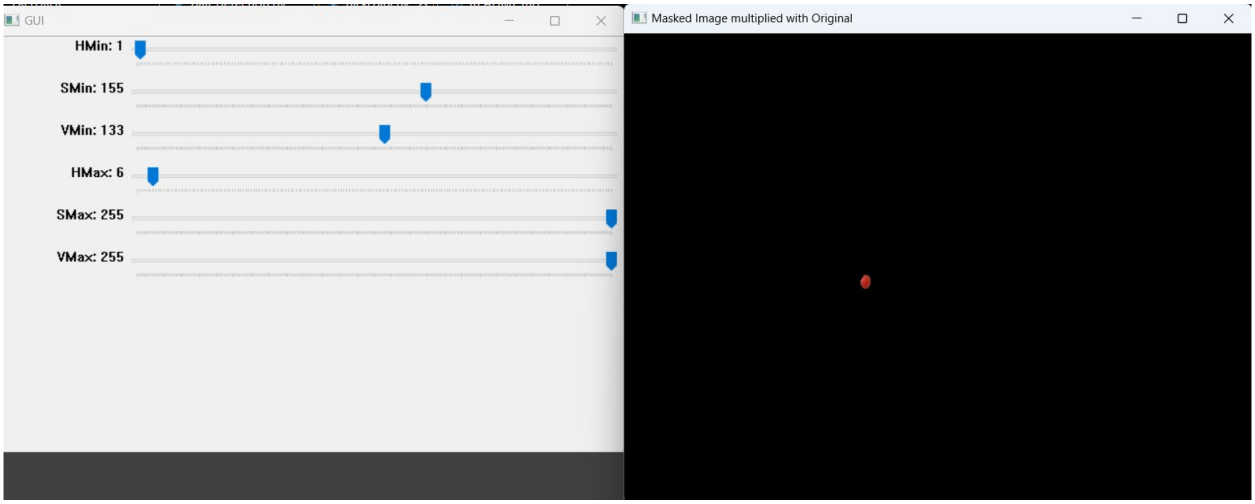
[Answer]:

1. Detect & Plot - Center-point of the ball:

The following steps are followed to detect the ball, from the given video:

(Note: All the computations related to Problem 1 – 1a,1b,1c – are done in the attached python program – "redball_trajectory.py")

1. For each frame:
   - convert to hsv image
   - find threshold values for hsv mask
   - detect the HSV of the red ball: (a python program – pickcolor – is used for this purpose, screenshot is shown below)



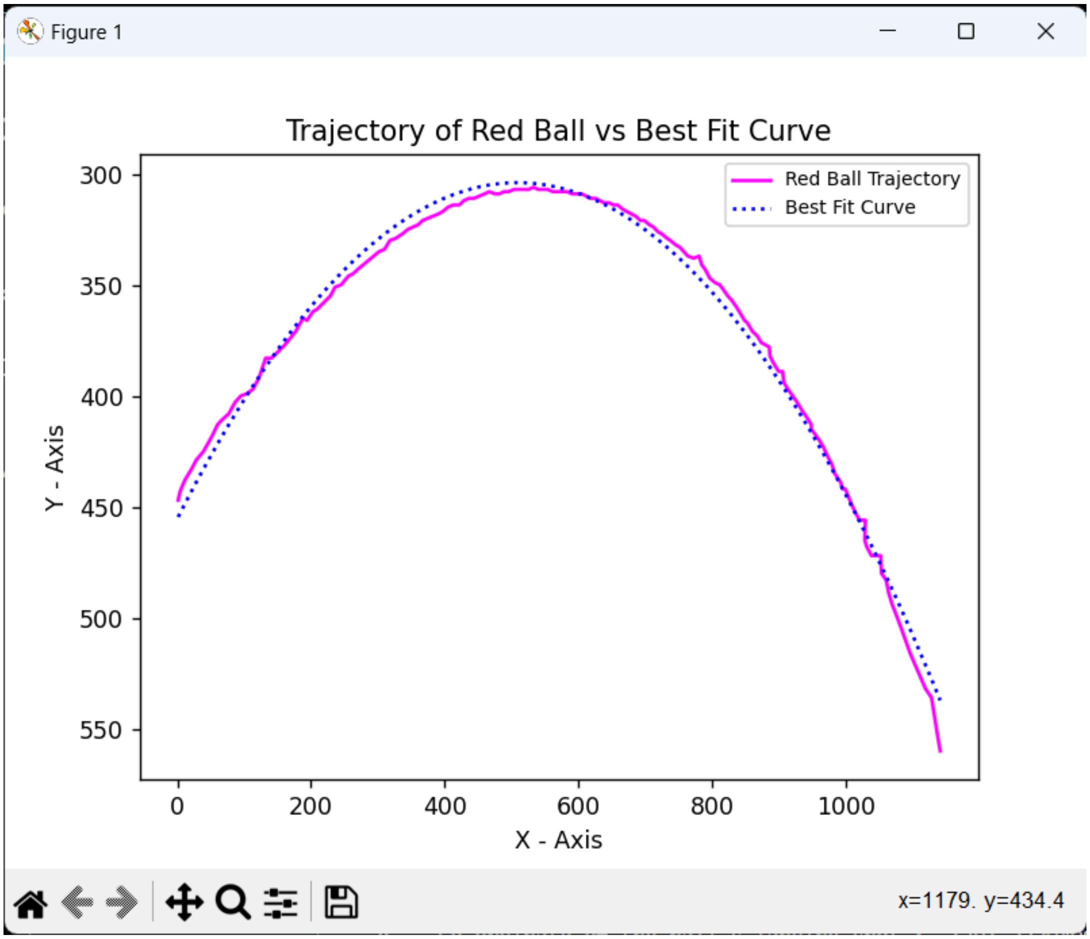
- using these threshold values, create a mask
- the output of masked image is the image with white and black
- the white color pixels are our area of interest
- find the co-ordinates of center of the red ball

-

2. Get a list of the x and y co-ordinates of the detected object in each frame
3. Plot the data
4. With x,y as data points, compute the best fit (least squares method) and plot on top of the ball trajectory.
5. Since this follows a parabolic path, which relates to standard form of quadratic equation,

$$y = ax^2 + bx + c$$

Plot:



2. Curve Fit for Trajectory

a. Equation of the Curve:

From the above same python program, the co-efficients for the following equation are computed

$$y = ax^2 + bx + c$$

and written in the equation format as below:

```
y = [0.00058286]x^2 + [-0.59380237]x + [455.16493066]
```

b. Plot – Best Curve Fit:

The best fit curve is achieved through the least square method and is shown in the above graph in the blue dotted line.

3. 'x' - co-ordinate of Ball's Landing:

From the program – redball_trajectory.py – the x co-ordinate distance is calculated as below:

```
'x' - Co-ordinate of the ball's landing spot :  1381.2729490408815
```

Problems encountered:

1. Narrowing down to the close hsv values
2. image inversion created miscalculations
3. The distances are subjective given the level of detection
4. erratic detections in the trajectory

Problem #2:

Given are two csv files, [pc1.csv]() and [pc2.csv](), which contain noisy LIDAR point cloud data in the form of (x, y, z) coordinates of the ground plane.

1. **Using pc1.csv:**
   a. **Compute the covariance matrix. [15]**
   b. **Assuming that the ground plane is flat, use the covariance matrix to compute the magnitude and direction of the surface normal. [15]**

2. **In this question, you will be required to implement various estimation algorithms such as Standard Least Squares, Total Least Squares and RANSAC.**
   a. **Using pc1.csv and pc2, fit a surface to the data using the standard least square method and the total least square method. Plot the results (the surface) for each method and explain your interpretation of the results. [20]**
   b. **Additionally, fit a surface to the data using RANSAC. You will need to write RANSAC code from scratch. Briefly explain all the steps of your solution, and the parameters used. Plot the output surface on the same graph as the data. Discuss which graph fitting method would be a better choice of outlier rejection. [20]**

[Answer]:

1. Covariance Matrix & Surface Normal

a. Covariance Matrix:

Using the attached python program – "Covariance.py" – the covariance matrix, for a given dataset – "pc1.csv" - is calculated as:

```
Covariance Matrix:
[[ 33.7500586   -0.82513692 -11.39434956]
 [ -0.82513692  35.19218154 -23.23572298]
 [-11.39434956 -23.23572298  20.62765365]]
```

b. Surface Normal – Magnitude & Direction:

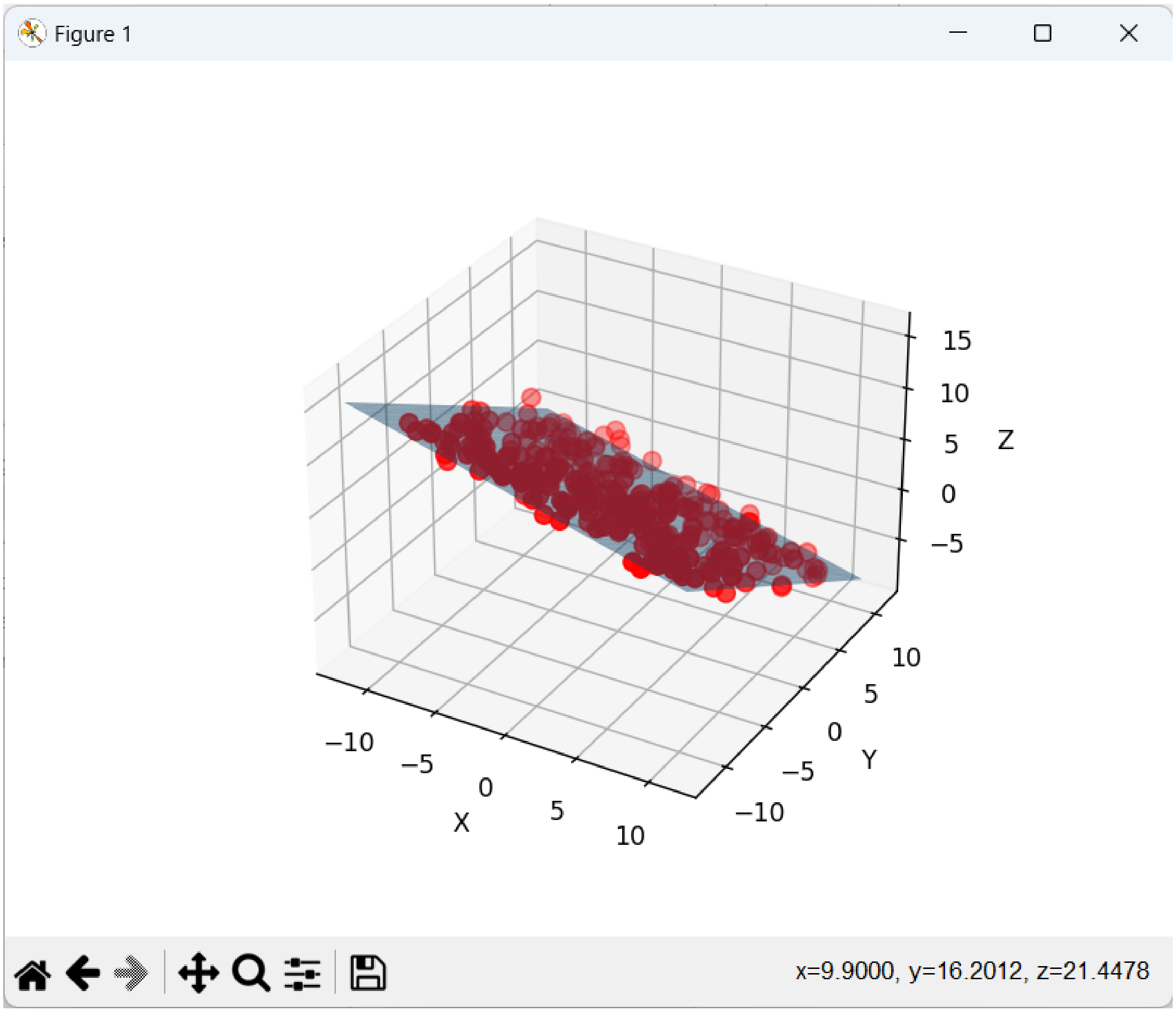
Using the same program as above, the magnitude and direction of the surface normal is calculated as below:

```
 Eigen Values:  [ 0.66950978 34.65757844 54.24280557]

 Eigen Vectors:  [[ 0.28616428  0.90682723 -0.30947435]
 [ 0.53971234 -0.41941949 -0.72993005]
 [ 0.79172003 -0.04185278  0.60944872]]

 Minimum eigen value: 0.6695097797333602        & its index: 0

Magnitude of Surface Normal:  1.0

Angle between projection of surface normal on xy plane & x-axis:  55 °

Angle between projection of surface normal on xy plane & surface normal:  49 °

Angle between surface normal & z- axis:  41 °
```

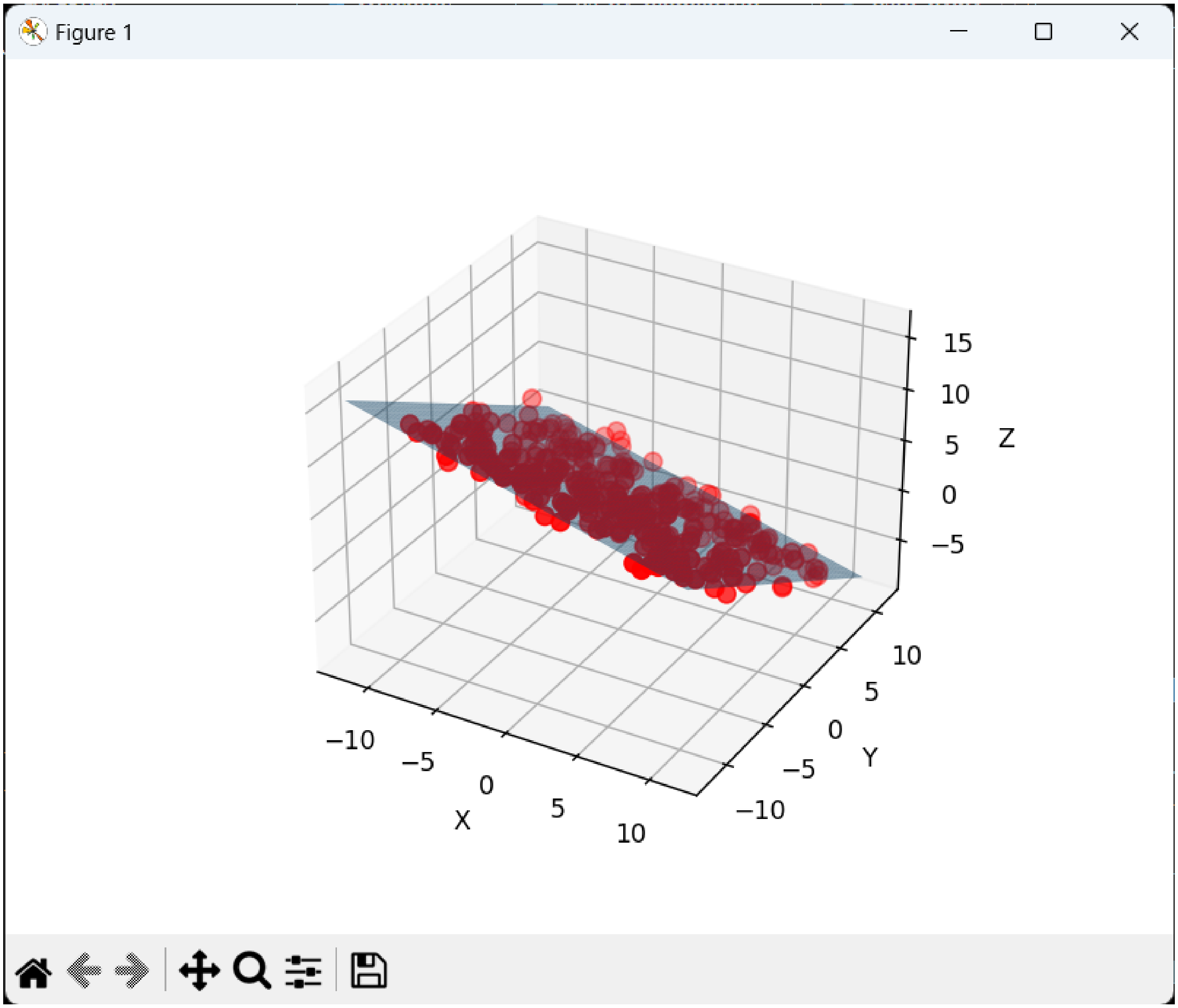
2. Estimation Algorithms

a. Surface Fit using LS and TLS methods:

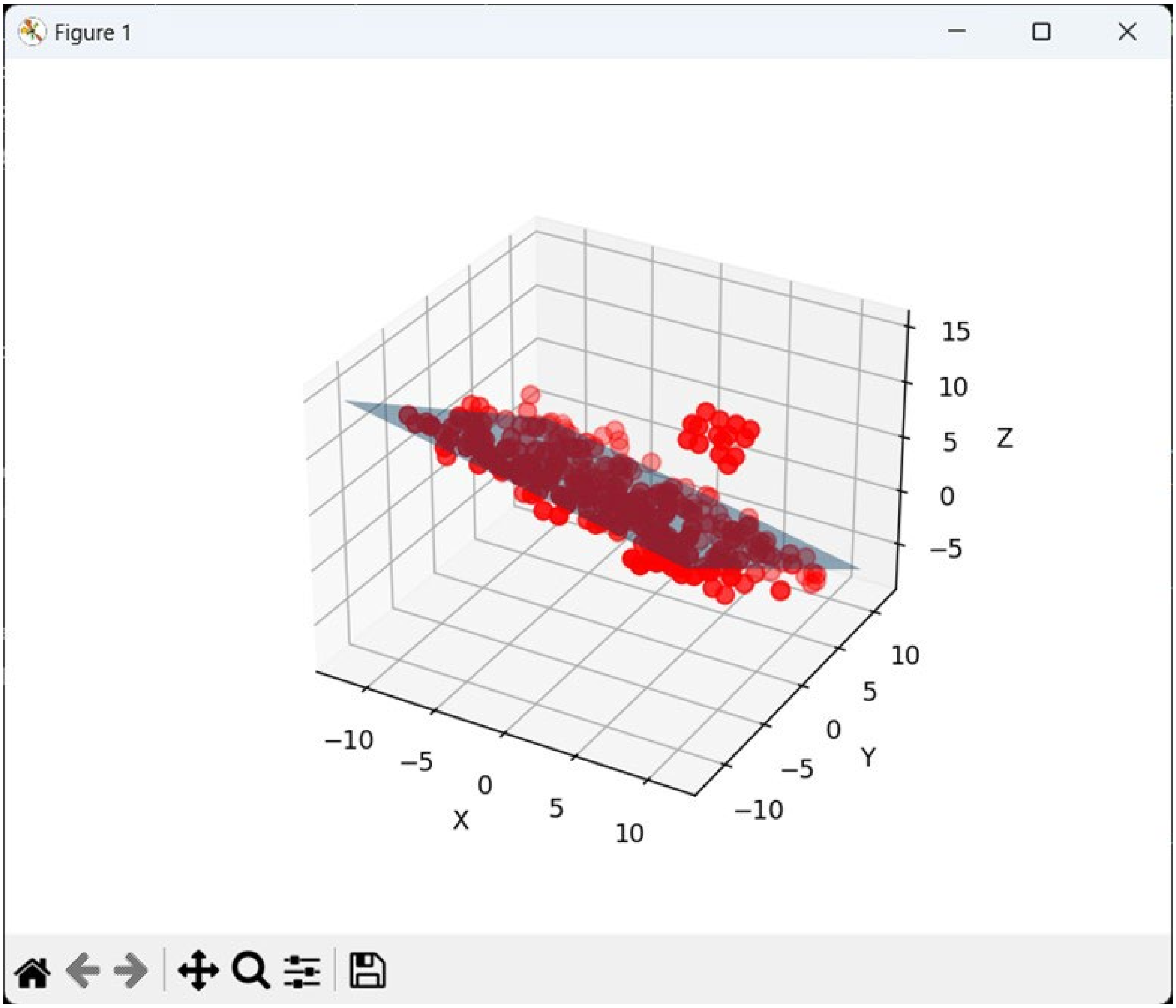
i. Plot: Least Squares method: Dataset pc1.csv



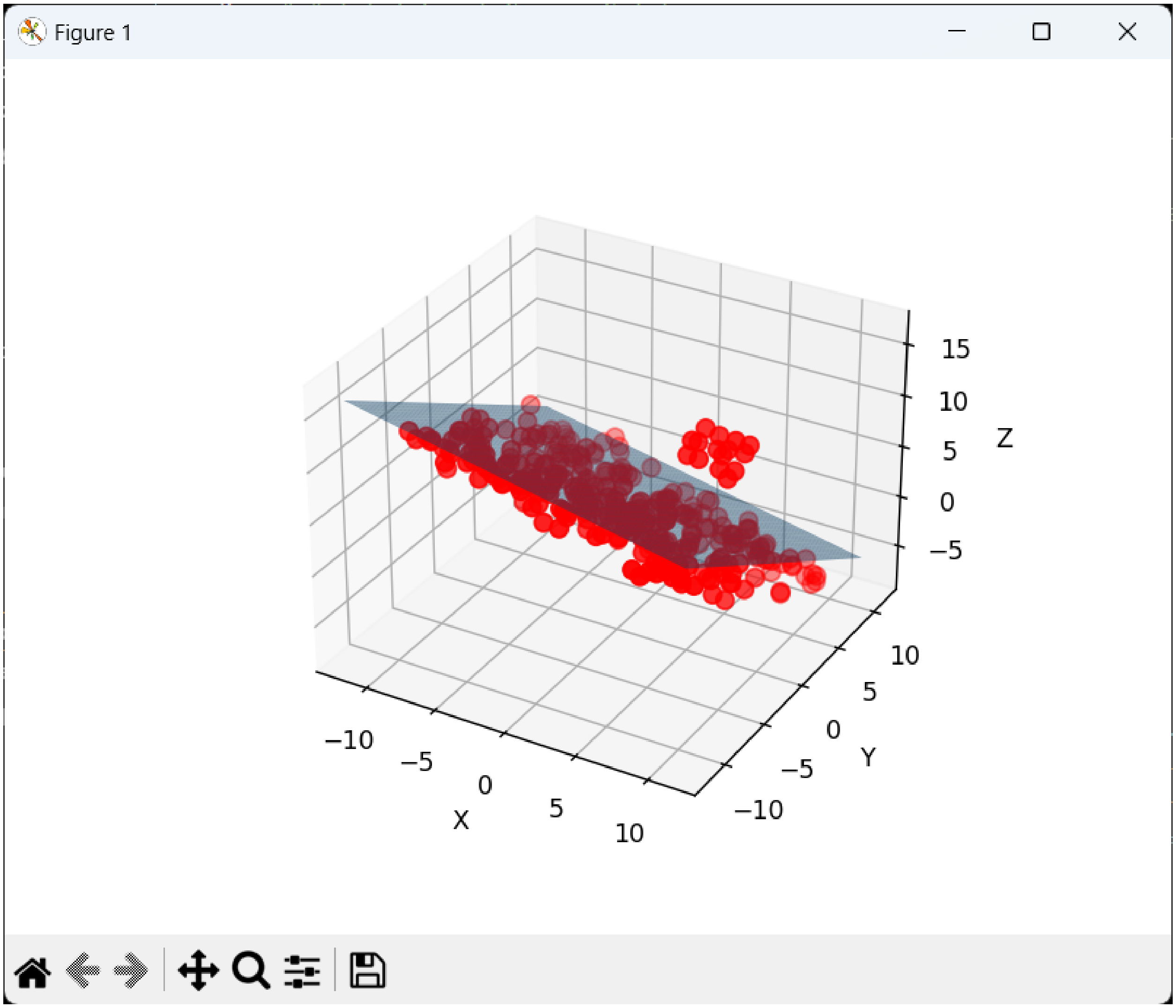
ii. Plot: Total Least square Method: Dataset pc1.csv



iii. Plot: Least square Method: Dataset pc2.csv



iv. Plot: Total Least square Method: Dataset pc2.csv



v. Interpretation:

As can be seen from the above graphs, the following are the observations -

- the plane surface pull relatively towards the outlier point cloud in dataset pc2.csv in relation to the dataset pc1.csv.
- The degree of pull is more in total least squares method in comparison to the least square method, compensating for its limitation to some extent.

b. Surface Fit using RANSAC Method:

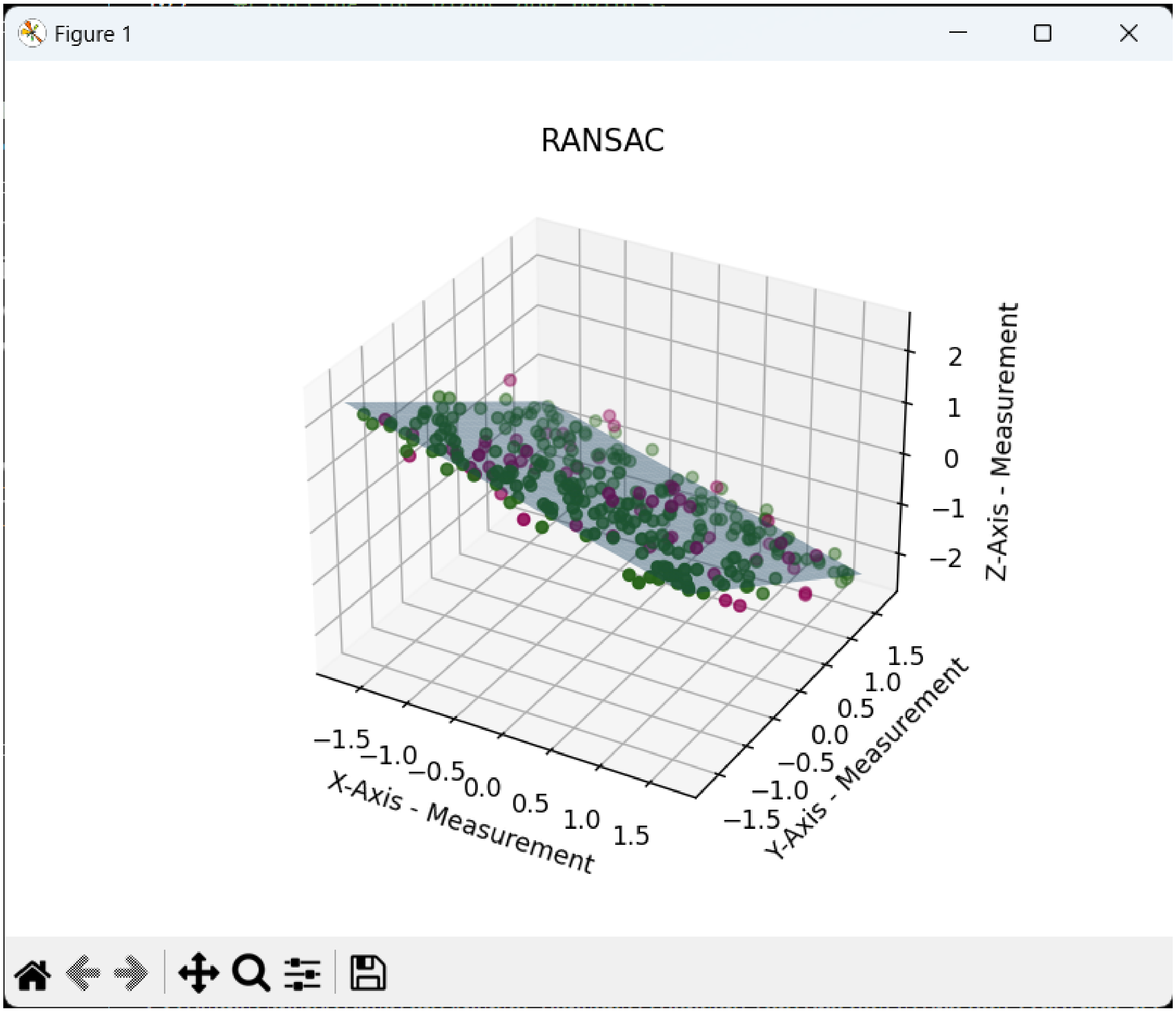
i. Steps of the solution:

The following steps are followed for computing the surface fit, for the given datasets, using the RANSAC method:

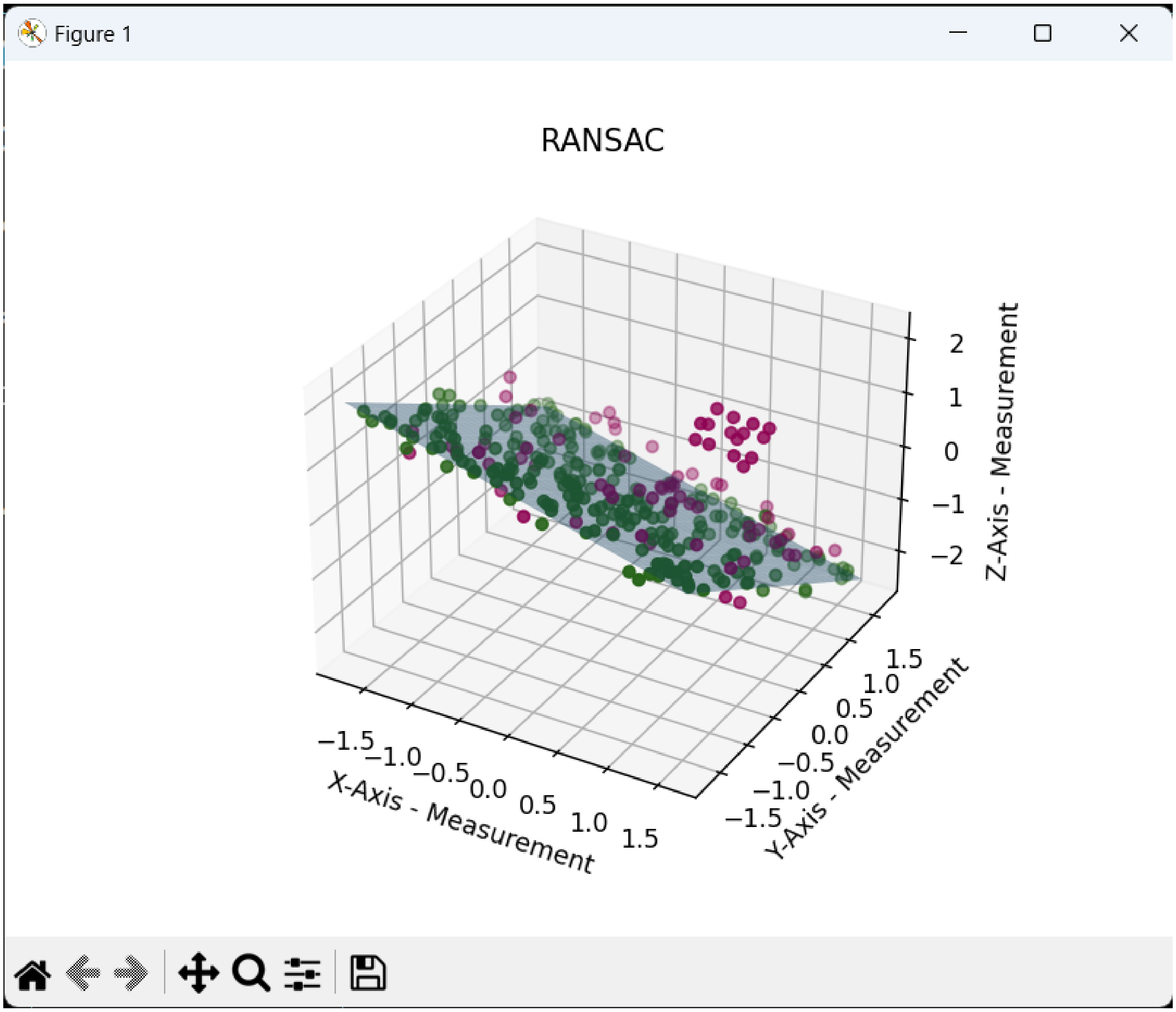
- Read the data from the 'csv' file
- Normalize data to avoid numerical errors and easier to fit a plane

- Fix the initial parameters
- Apply RANSAC algorithm
- Classify and store inliers and outliers
- Plot the data

ii. *Plot: RANSAC Method: Dataset pc1.csv*



iii. Plot: RANSAC Method: Dataset pc1.csv



iv. Summary:

As seen in the above graphs, the outliers are clearly rejected by the RANSAC algorithm, and the surface fit mostly aligns with the points which are inliers with a certain threshold level.

Problems encountered:

SVD calculations become frustrating

surface normal calculation is tedious.