TOBB ECENOMICS AND TECHNOLOGY UNIVERSITY

MAK 420
COMPUTER AIDED DESIGN (CAD)

HOMEWORK I
VACUUM CLEANER

Canberk URUŞ, 181501038

1. Contents

2. Problem Definition

In this assignment, the components of the vacuum shown in Figure 1(a) are modeled using primitive shapes (Unit Cube(Figure 3)) using the methods of scaling, translation, rotation and combinations of these transformations.

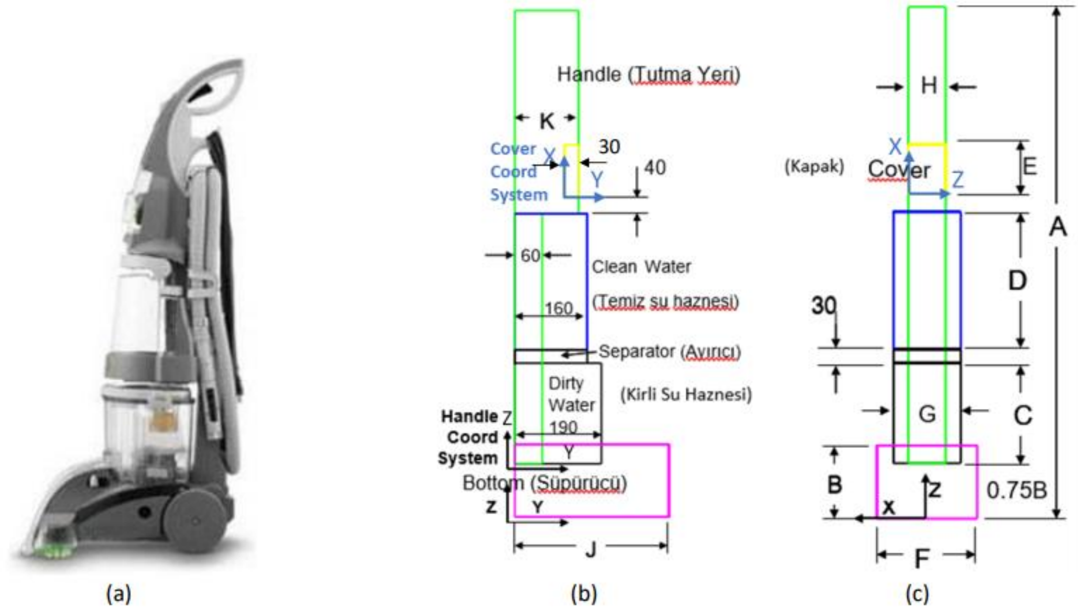


Figure 1. Vacuum Cleaner Model (a) Photo of the vacuum, Simple models (b) ZY (c) XZ view. [1]

A. Parts List

1. Bottom
2. Dirty Water Compartment
3. Separator
4. Clean Water Compartment
5. Handle
6. Cover

B. Parameters

Table 1. Design Parameters

	A	B	C	D	E	F	G	H	J	K	Θ_1	Θ_2
Design 1	1000	160	220	300	110	220	150	80	340	140	30	75
Design 2	1200	180	250	310	100	260	180	85	410	150	45	20
Design 3	800	150	200	250	100	210	150	75	320	120	60	30

C. Considerations

As shown in the Figure 2 (a), all components on top of the sweeper (bottom) can rotate with an angle of ϴ1 around the X axis of the coordinate system of the handle.

The cover piece can rotate around the local X axis at its lower front point, independently of other components, with an angle of ϴ2. As seen in Figure 2(a) and (b), it should turn outward.

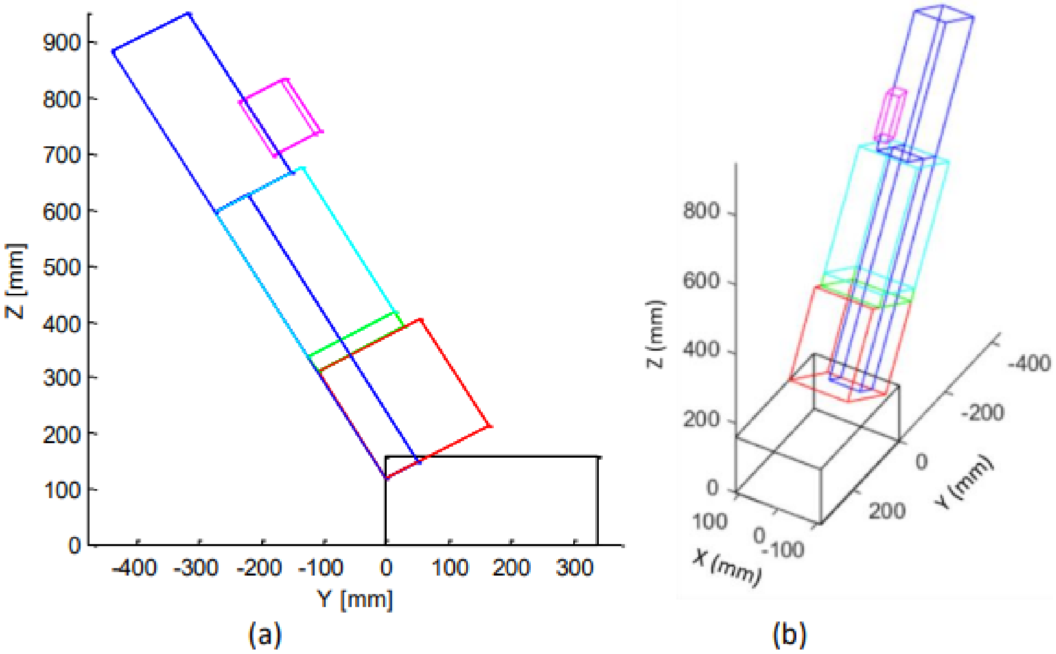


Figure 2. Vacuum Cleaner Model (a) Simple models (b) ZY (c) isometric view. [1]

3. Conceptual Design

Primitive Shapes

All the parts used are created by applying rigid body transformations on the Unit Cube.

The position of the unit cube is chosen in such a way that it makes the most sense for solving the given problem. (Figure 3)

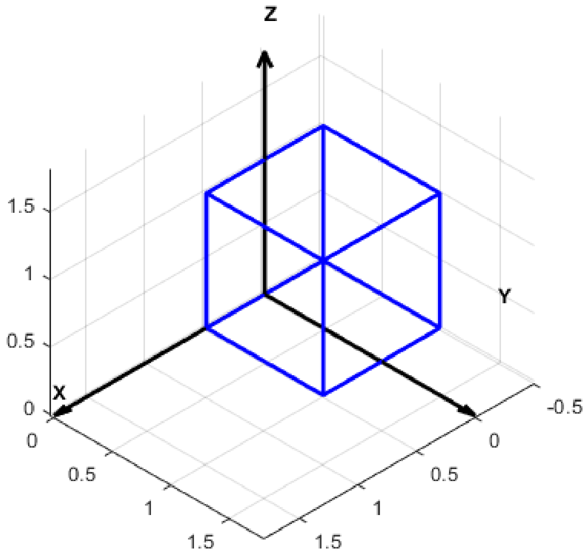


Figure 3. Unit Cube (Unit_Cube.m)

A. Bottom

i. Scale

Since the starting shape is a unit cube, in order to get the desired shape of the bottom, cube needs to be scaled in the x-axis with the magnitude equaling to the width of the bottom, and the same is repeated for the y axis with depth and z-axis with height.

For the rest of the parts, the same principle will be used. The magnitude of the variables are given in the Table 1 - Design Parameters.

$$Scale\ Matrix = S_b = \begin{bmatrix} F & 0 & 0 & 0 \\ 0 & J & 0 & 0 \\ 0 & 0 & B & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ii. Translate

As stated above, thanks to the starting position of the cube, translation with respect to Global Coordinate System (GCS) is not needed.

$$Translate\ Matrix = T_b = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

iii. Rotate

Since bottom piece is already in its place and rotation of this piece is not considered, there is no need for a rotation matrix to be created.

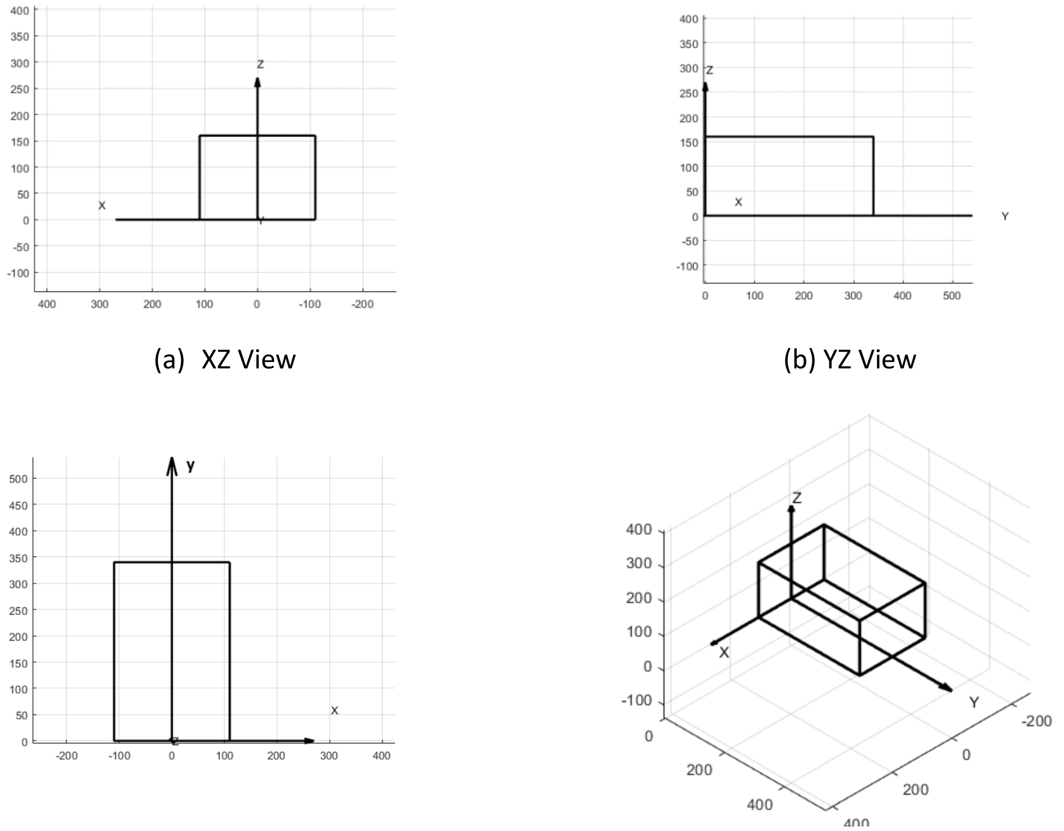
iv. Combined Transformations

$$b = T_b$$

v. Final Matrix

The final coordinates of the bottom piece is obtained by scaling the unit cube. Even though the translation matrix is created, translation operation is not performed here.

$$bottom = b * S_b * cube$$



(a) XZ View (b) YZ View

(c) XY View (d) Isometric View

Figure 4.Bottom Model Views

B. Dirty Water

i. Scale

$$Scale\ Matrix = S_{dw} = \begin{bmatrix} G & 0 & 0 & 0 \\ 0 & 190 & 0 & 0 \\ 0 & 0 & C & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ii. Translate (wrt. Bottom)

To position the dirty water compartment in its desired place the cube needs to be translated with respect to the bottom piece. In this case dirty water compartment is translated in the z-axis with the magnitude of 0.75*B units.

$$Translate\ Matrix = T_{dw} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.75 * B \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

iii. Rotate (x-axis)

Because the rotation (in the x-axis) of the dirty water compartment and the rest of the parts are desired, rotation matrix is created. P(x, Θ_1).

$$Rotate\ Matrix = R_{dw} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\Theta_1) & -\sin(\Theta_1) & 0 \\ 0 & \sin(\Theta_1) & \cos(\Theta_1) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

iv. Combined Transformations

The two transformations needs to be combined (rotation and translation). Here first translation then rotation is applied and the resulting “Combined Translation and Rotation Matrix” is created.

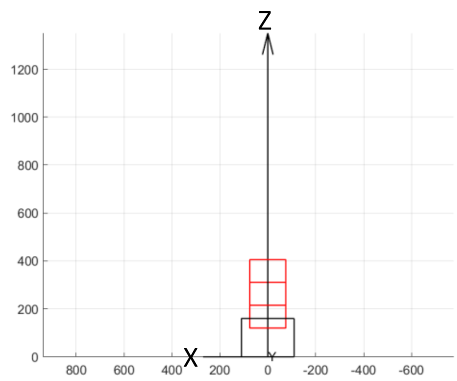
$$dw = T_{dw} * R_{dw}$$

Transformation order is from left to right (→).

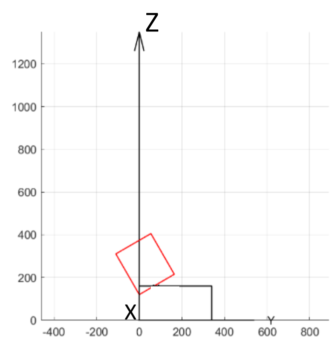
v. Final Matrix

The cube is first scaled then translated wrt. Bottom piece and rotated in its x-axis. Before these operations, part is positioned on the GCS by applying the previous parts combined transformations.

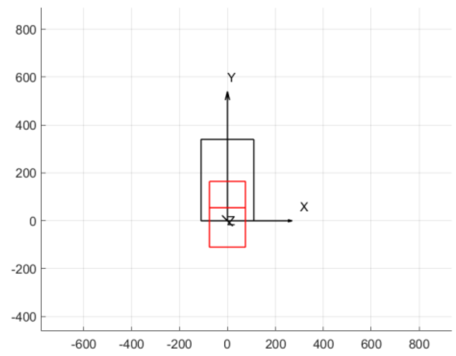
$$Dirty\ Water = b * dw * S_{dw} * cube$$



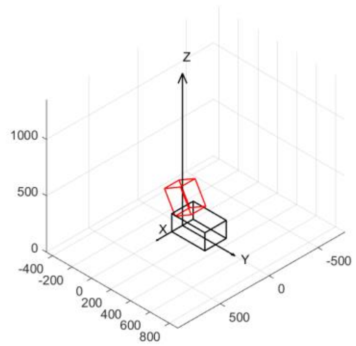
(b) XZ View



(b) YZ View



(c) XY View



(d) Isometric View

Figure 5. Dirty Water Compartment Model Views (No Rotation)

C. Seperator

i. Scale

$$Scale\ Matrix = S_s = \begin{bmatrix} G & 0 & 0 & 0 \\ 0 & 160 & 0 & 0 \\ 0 & 0 & 30 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ii. Translate (wrt. Dirty Water Compartment)

$$Translate\ Matrix = T_s = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & C \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

iii. Rotate

No Rotation (wrt. Dirty water compartment) is needed.

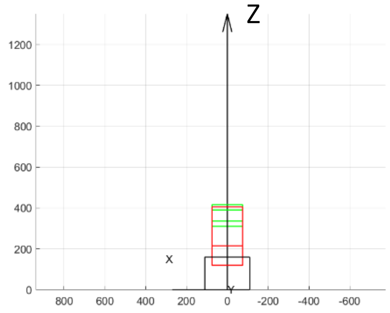
iv. Combined Transformations

$$s = T_s$$

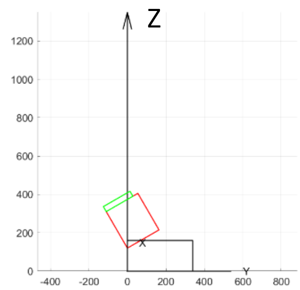
v. Final Matrix

Before applying transformations, the previous parts combined transformations are applied to the part to position it in the GCS (b*dw in this case).

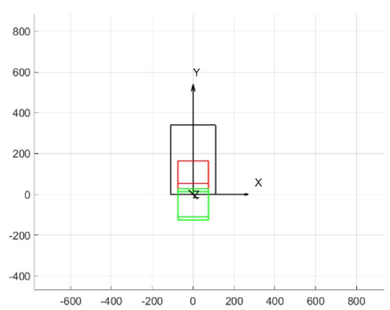
$$Seperator = b * dw * s * S_s * cube$$



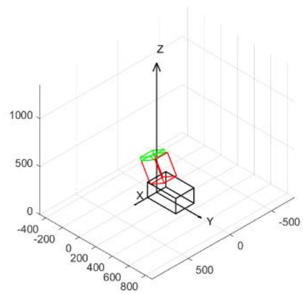
(c) XZ View



(b) YZ View



(c) XY View



(d) Isometric View

Figure 6. Seperator Model Views

D. Clean Water

i. Scale

$$Scale\ Matrix = S_{cw} = \begin{bmatrix} G & 0 & 0 & 0 \\ 0 & 160 & 0 & 0 \\ 0 & 0 & D & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ii. Translate (wrt. Seperator)

$$Translate\ Matrix = T_s = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 30 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

iii. Rotate

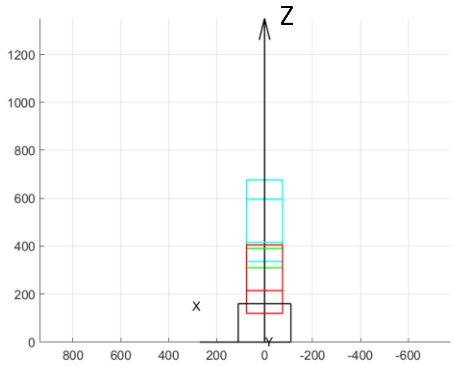
No Rotation (wrt. Seperator) is needed.

iv. Combined Transformations

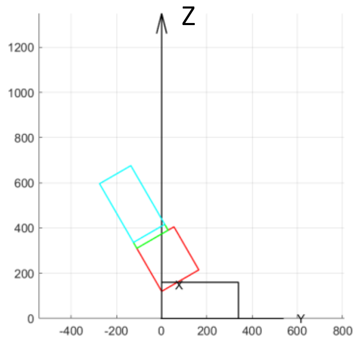
$$cw = T_{cw}$$

v. Final Matrix

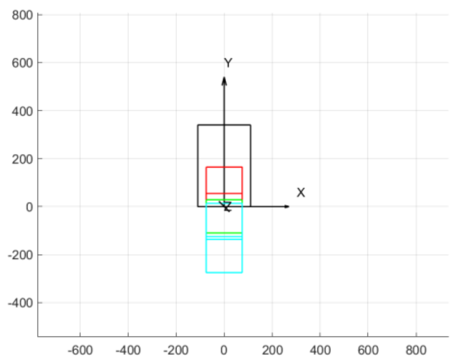
$$Clean\ Water = b * dw * s * cw * S_{cw} * cube$$



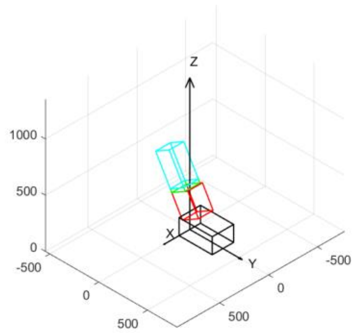
(d) XZ View



(b) YZ View



(c) XY View



(d) Isometric View

Figure 7. Clean Water Compartment Model Views

E. Handle (Lower)

i. Scale

$$Scale\ Matrix = S_{hl} = \begin{bmatrix} H & 0 & 0 & 0 \\ 0 & 60 & 0 & 0 \\ 0 & 0 & D + C + 30 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ii. Translate (wrt. Dirty Water Compartment)

Translation is not applied here since dirty water compartment and lower-handles positions are same wrt bottom.

$$Translate\ Matrix = T_{hl} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

iii. Rotate (x-axis)

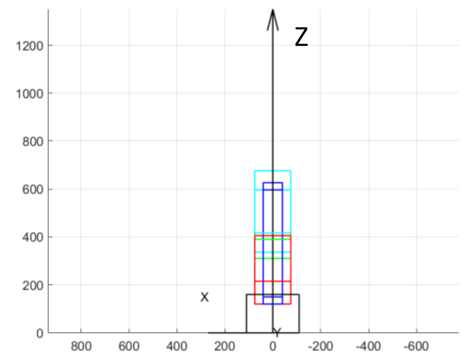
Handle doesn't have any rotation wrt. dirty water compartment. So, rotation is not applied either.

iv. Combined Transformations

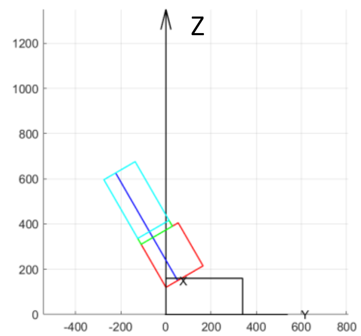
$$hl = T_{hl}$$

v. Final Matrix

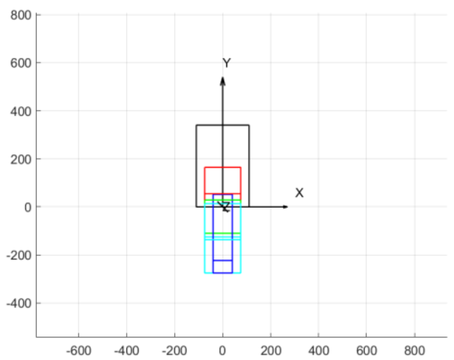
$$Handle\ Lower = b * dw * hl * S_{hl} * cube$$



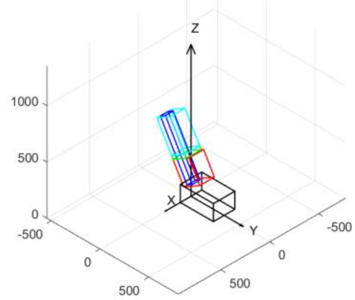
(e) XZ View



(b) YZ View



(c) XY View



(d) Isometric View

Figure 8. Lower Handle Model Views (No Rotation)

F. Handle Upper

i. Scale

$$Scale\ Matrix = S_{hu} = \begin{bmatrix} H & 0 & 0 & 0 \\ 0 & K & 0 & 0 \\ 0 & 0 & A - (0.75 * B + C + D + 30) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ii. Translate (wrt. Lower Handle)

$$Translate\ Matrix = T_{hu} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & C + D + 30 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

iii. Rotate

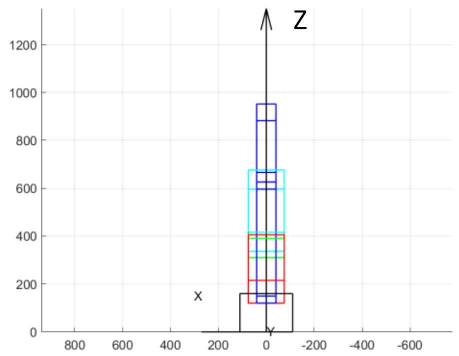
No Rotation (wrt. Lower Handle) needed.

iv. Combined Transformations

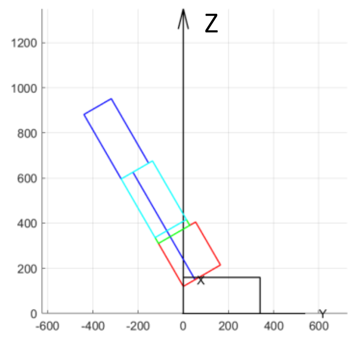
$$hu = T_{hu}$$

v. Final Matrix

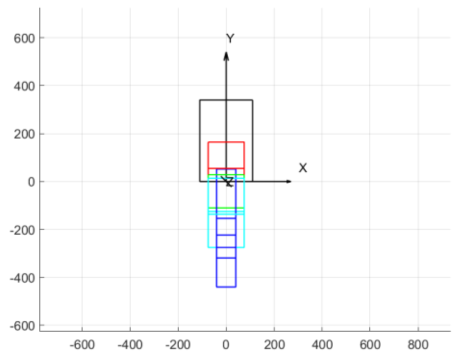
$$Handle\ Upper = b * dw * hl * hu * S_{hu} * cube$$

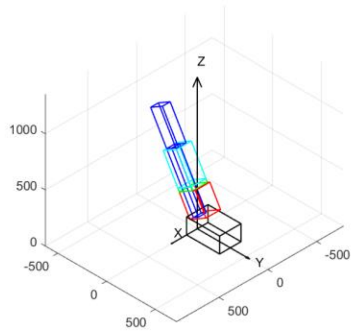


(f) XZ View



(b) YZ View





(c) XY View

(d) Isometric View

Figure 9.Upper Handle Model Views

G. Cover

i. Scale

$$Scale\ Matrix = S_c = \begin{bmatrix} H & 0 & 0 & 0 \\ 0 & 30 & 0 & 0 \\ 0 & 0 & E & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ii. Translate (wrt. Upper Handle)

The transformations of cover are more challenging than other, that it is because this part needs to rotate around its front-side line, for this we need to move our primitive shape 0.5 units in the x-axis and -1 units int y-axis. This operation effects our original translation matrix (wrt. Upper Handle) so counter that issue, we need to translate -H/2 in the x-axis.

$$Translate\ Matrix\ (Unit\_Cube) = T = \begin{bmatrix} 1 & 0 & 0 & 0.5 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Translate\ Matrix = T_c = \begin{bmatrix} 1 & 0 & 0 & -H/2 \\ 0 & 1 & 0 & K \\ 0 & 0 & 1 & 40 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

iii. Rotate (z-axis)

$$Rotate\ Matrix\ (Z - axis) = R_c = \begin{bmatrix} \cos(\Theta_2) & -\sin(\Theta_2) & 0 & 0 \\ \sin(\Theta_2) & \cos(\Theta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

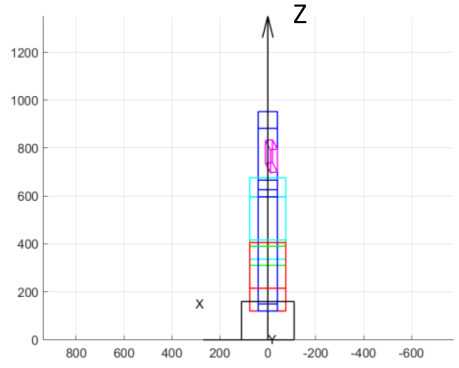
iv. Combined Transformations

$$c = T_c * R_c$$

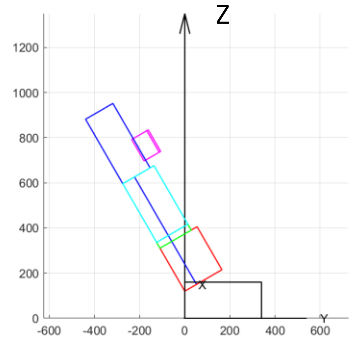
v. Final Matrix

Instead of using the T matrix, a different primitive shape (an unit cube where global origin point coincides with one of its corners) could have also been used.

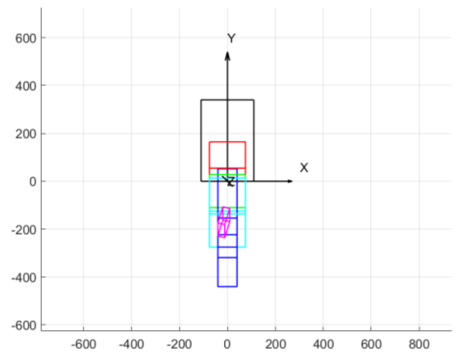
$$Cover = b * dw * hl * hu * c * S_c * (T * cube)$$



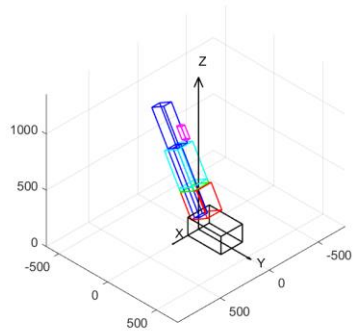
(g) XZ View



(b) YZ View



(c) XY View



(d) Isometric View

Figure 10.Cover Model Views (No Rotation)

4. Detail Design (with MATLAB)

After the preliminary design MATLAB Scripts were created,

These scripts plot the design according to the design parameters given in Table 1.

A. Explanation of Code Structure

There are 3 Scripts:

- *Unit_Cube.m*
  - Script
  - Displays the unit cube used in this assignment.
- *Plot3D_Vacuum.m*
  - Function
  - Starting with the bottom and by moving up plots the model part by part using the matrices and relations in the Chapter 3 – Conceptual Design.
- *MAK420_HW1_CANBERK_URUS.m*
  - Script
  - Starting (Main) Script !
  - Contains the parameters shown in the Table 1 in their respective structs.
  - Calls the Plot3D_Vacuum.m and passes the required parameters.

All the explanations, inputs, outputs, required functions and resources are given inside the code as a comment section. (Check Chapter 8 - Scripts)

B. Views for all 3 designs (No Rotation)

i. Design 1

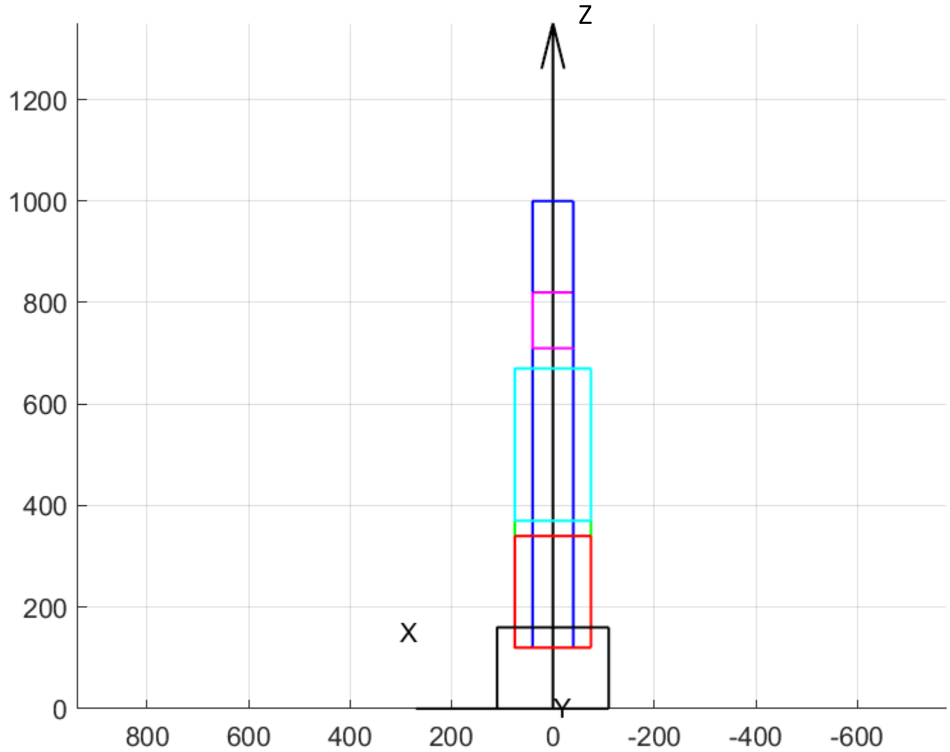


Figure 11.Vacuum Model XZ View – Without Rotation - Design 1

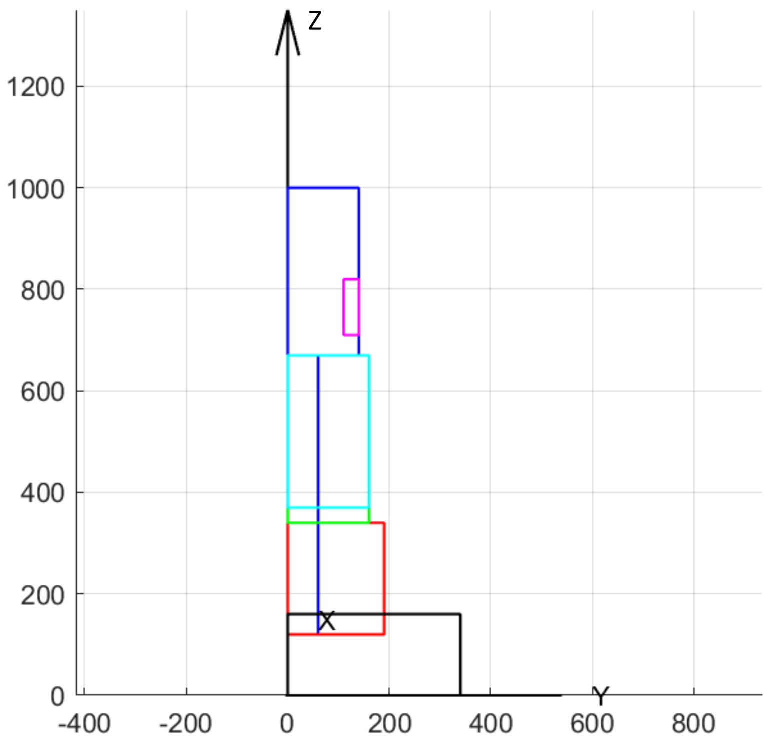


Figure 12.Vacuum Model YZ View – Without Rotation - Design 1

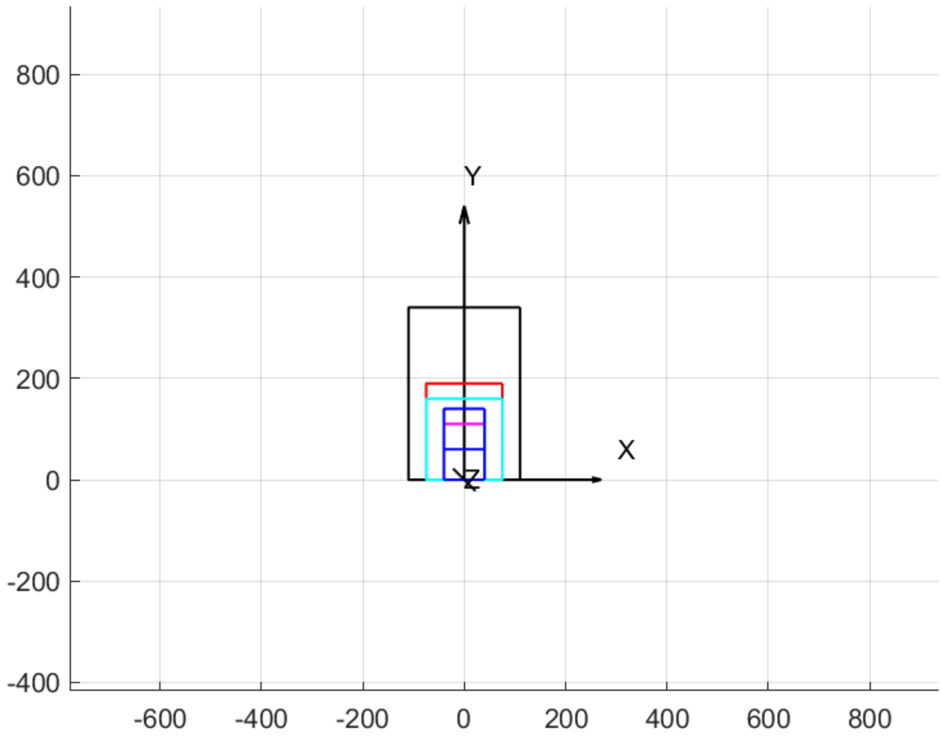


Figure 13 .Vacuum Model XY View – Without Rotation - Design 1

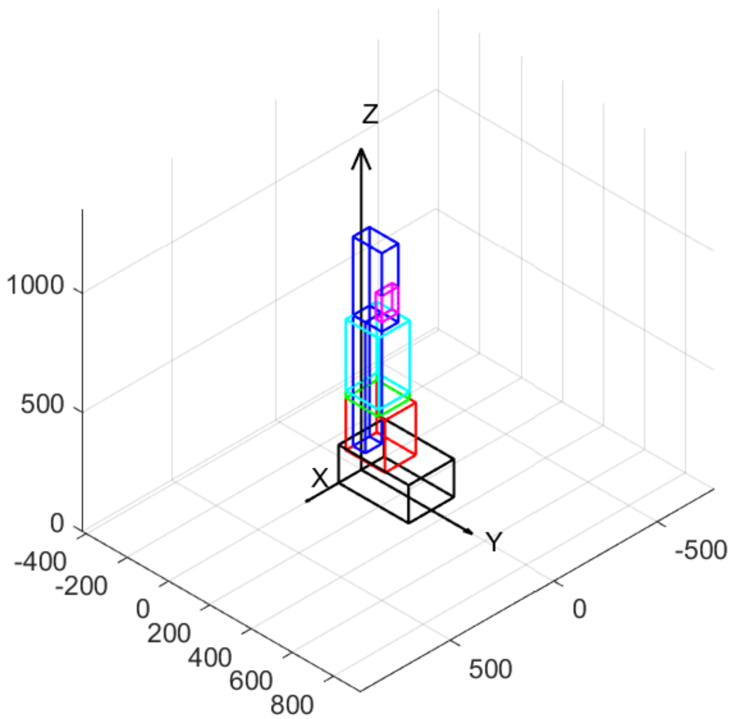


Figure 14. Vacuum Model Isometric View – Without Rotation - Design 1

ii. Design 2

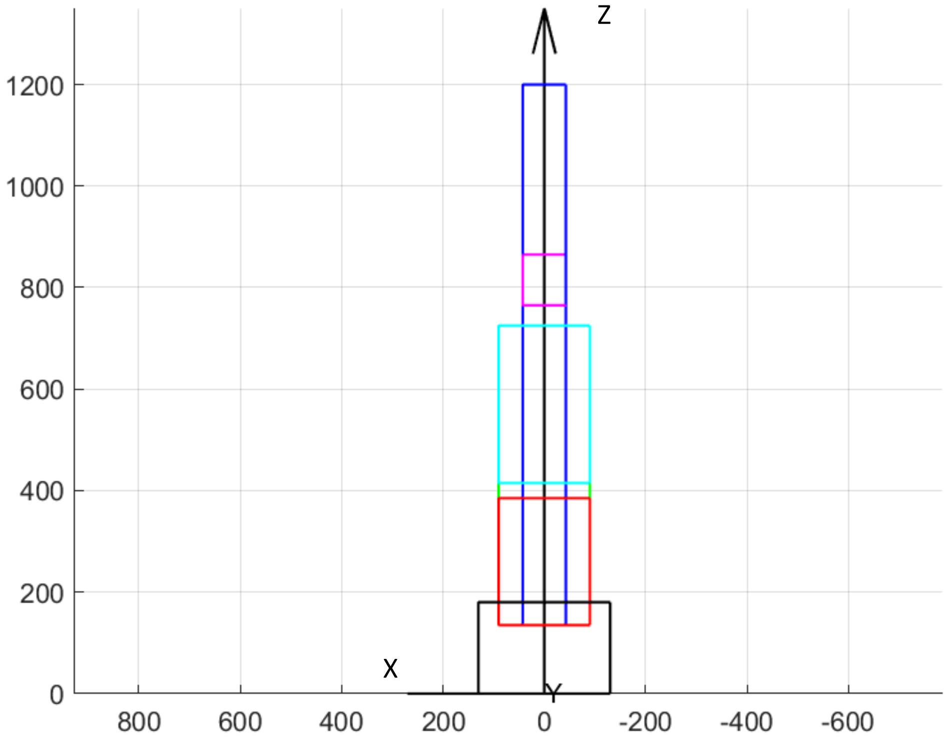


Figure 15. Vacuum Model XZ View – Without Rotation - Design 2

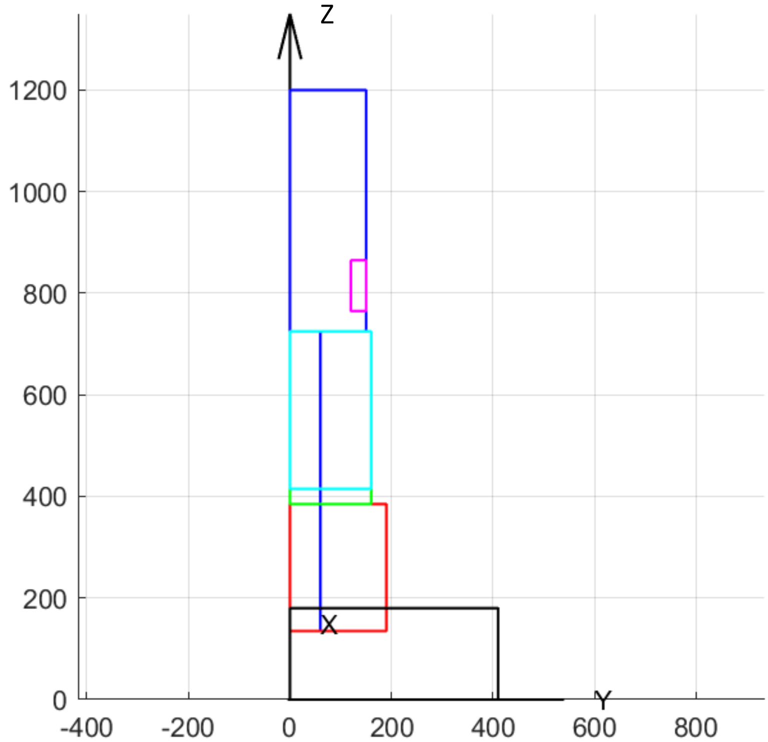


Figure 16. Vacuum Model YZ View – Without Rotation - Design 2

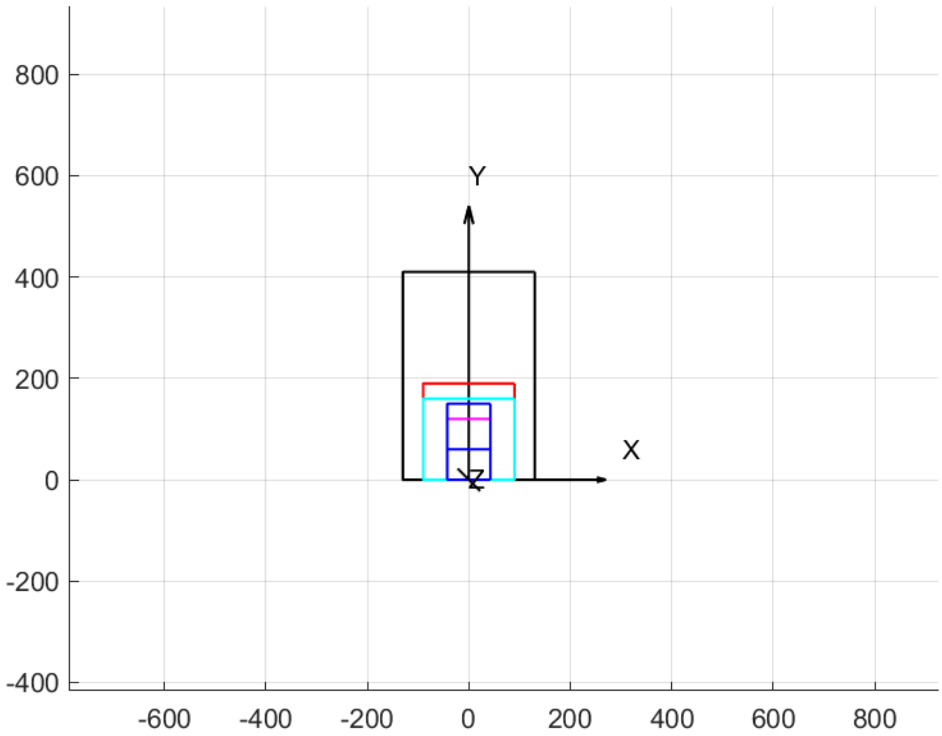


Figure 17 .Vacuum Model XY View – Without Rotation - Design 2

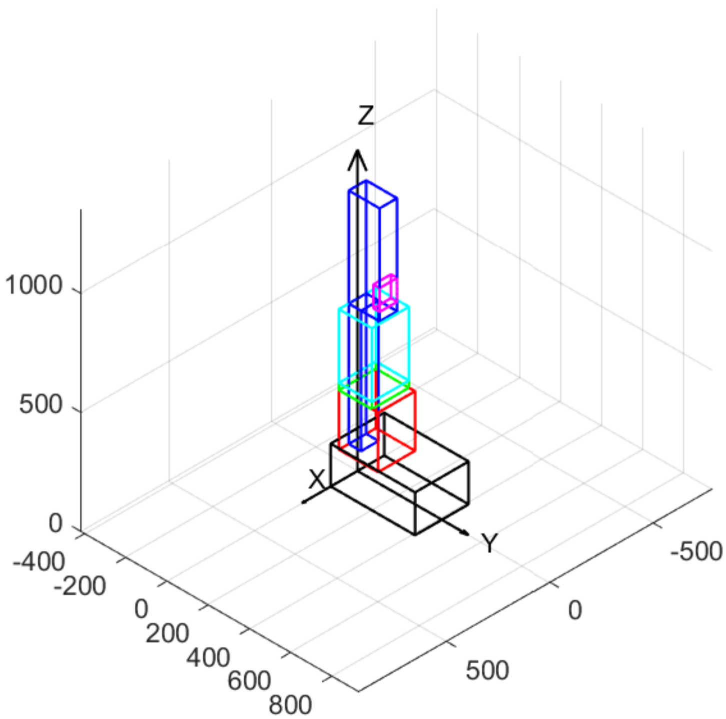


Figure 18. Vacuum Model Isometric View – Without Rotation - Design 2

iii. Design 3

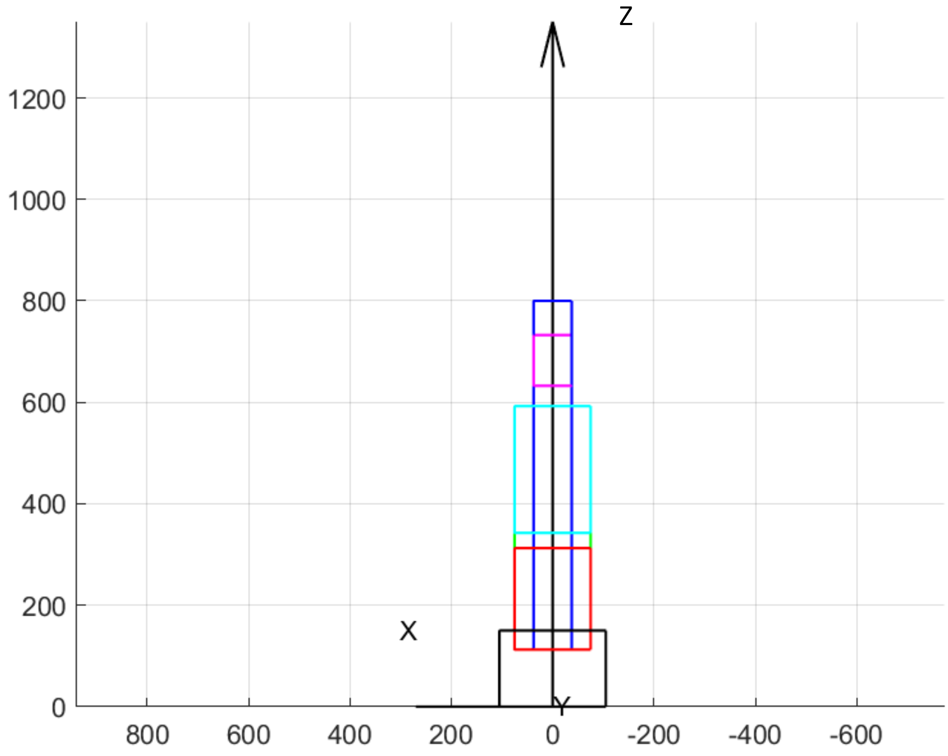


Figure 19.Vacuum Model XZ View – Without Rotation - Design 3

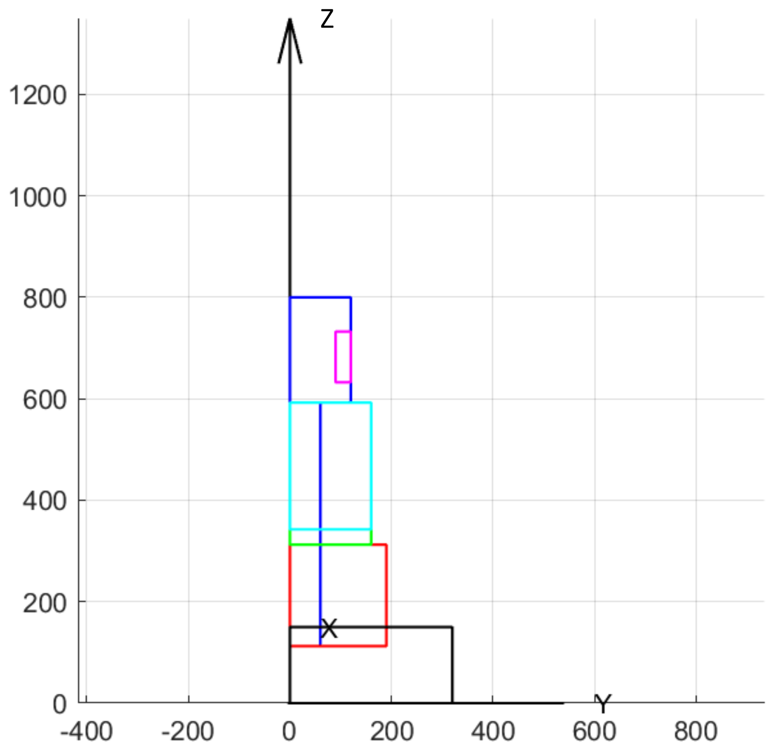


Figure 20.Vacuum Model YZ View – Without Rotation - Design 3

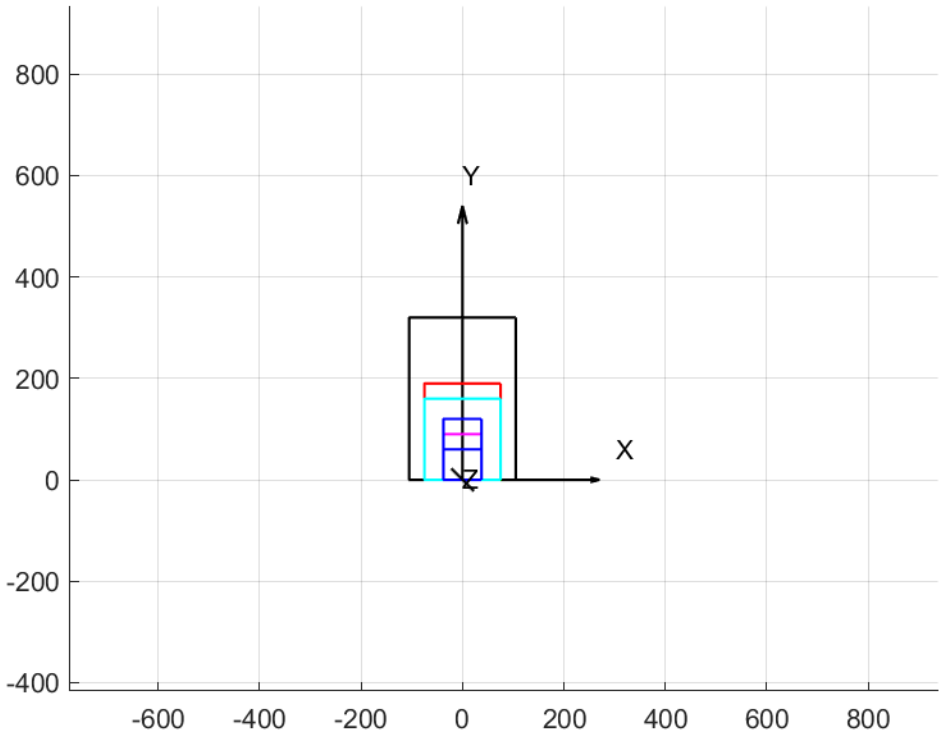


Figure 21 .Vacuum Model XY View – Without Rotation - Design 3

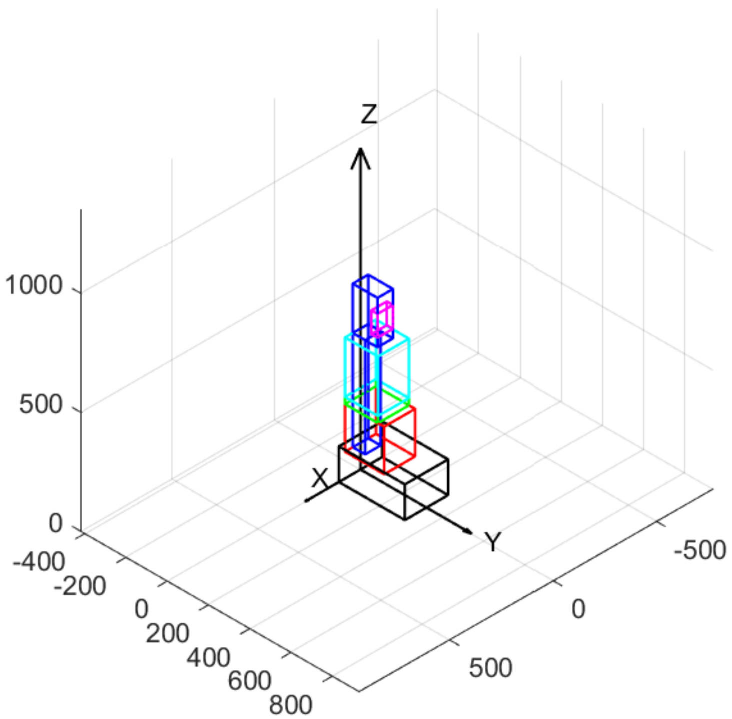


Figure 22. Vacuum Model Isometric View – Without Rotation - Design 3

C. Views for all 3 designs (With Rotation)

i. Design 1

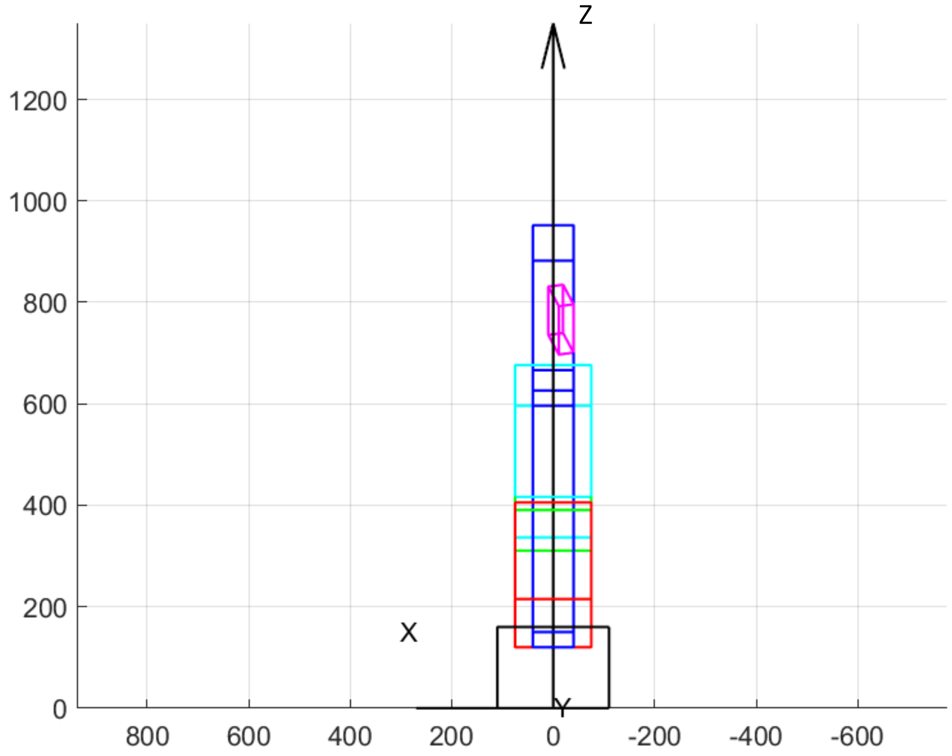


Figure 23. Vacuum Model XZ View – With Rotation - Design 1

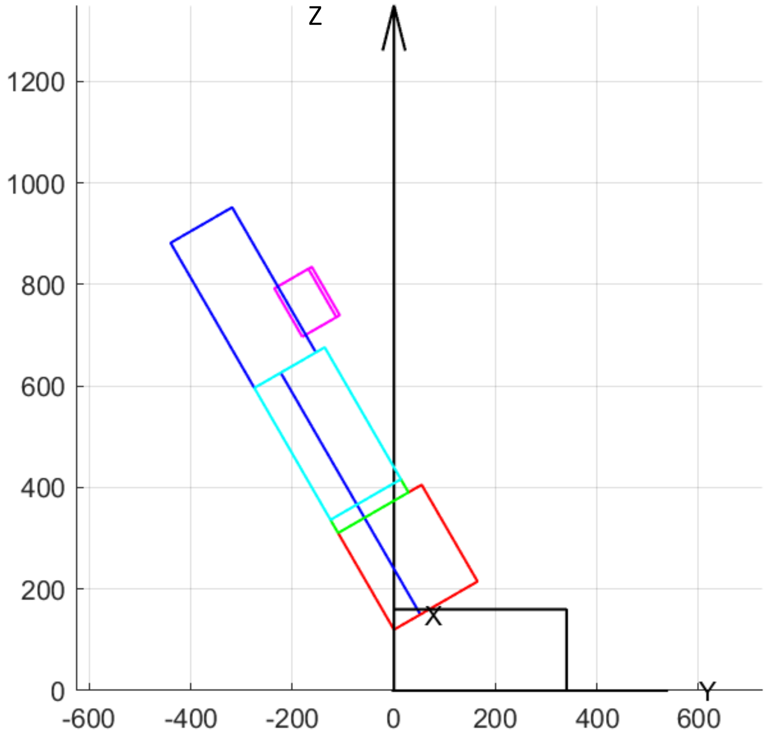


Figure 24.Vacuum Model YZ View – With Rotation - Design 1

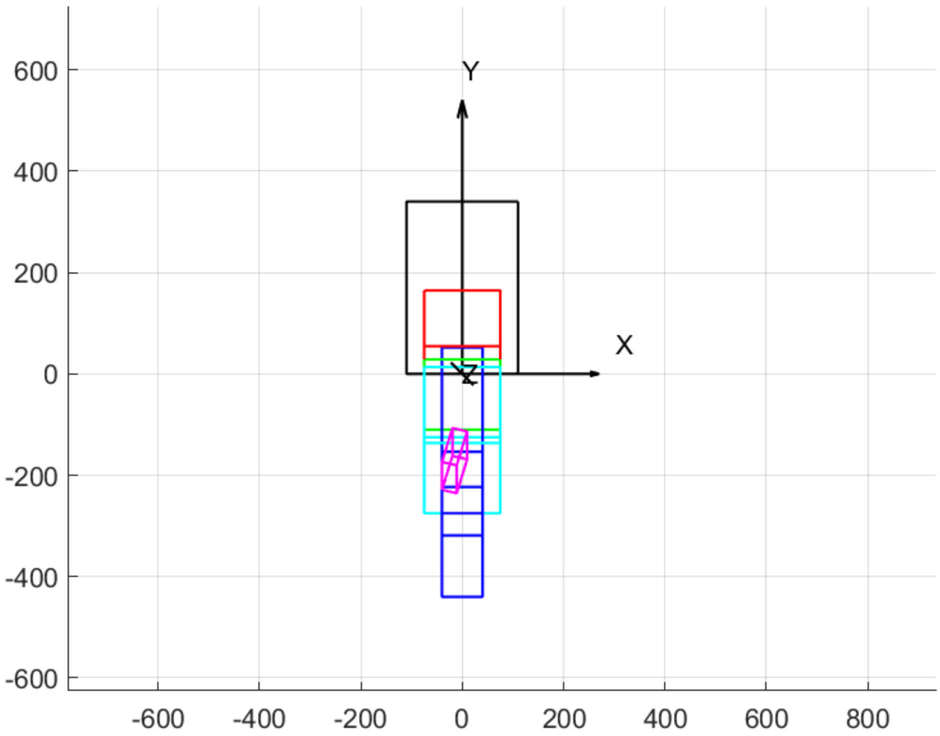


Figure 25 .Vacuum Model XY View – With Rotation - Design 1

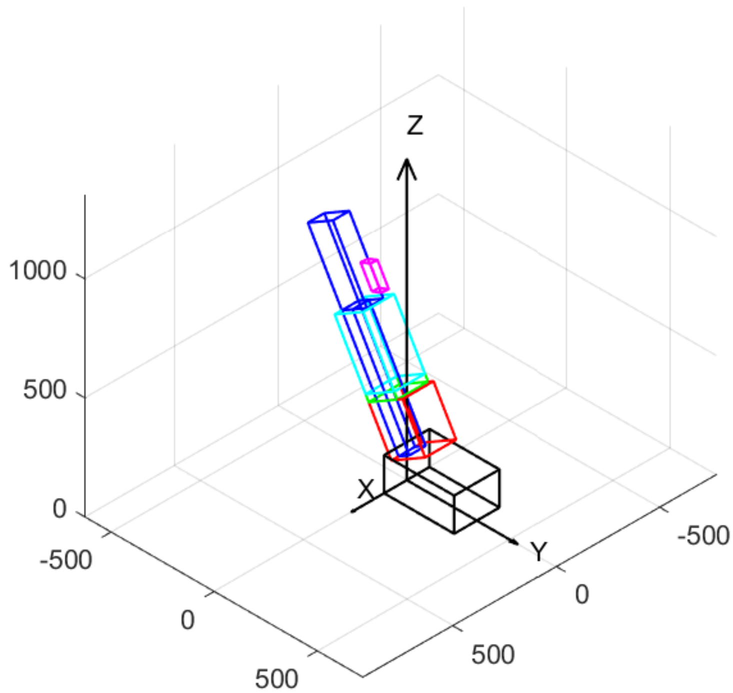


Figure 26. Vacuum Model Isometric View – With Rotation - Design 1

ii. Design 2

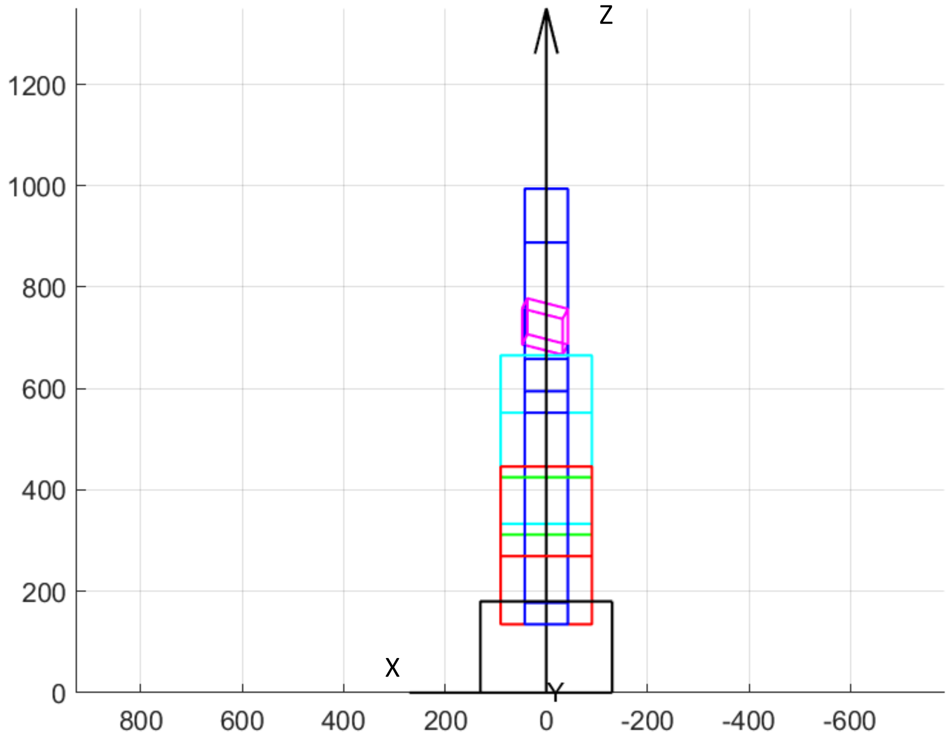


Figure 27.Vacuum Model XZ View – With Rotation - Design 2

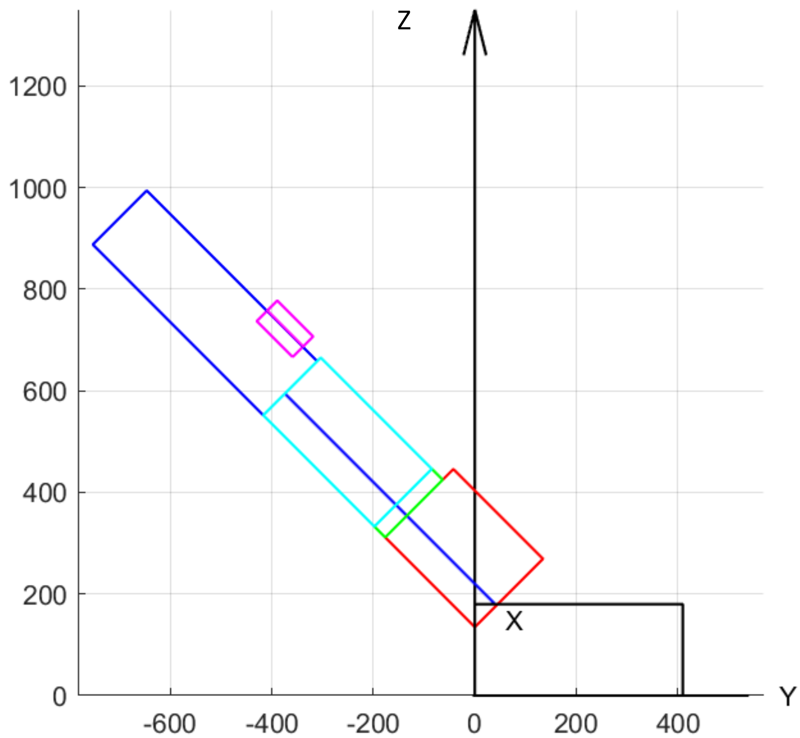


Figure 28.Vacuum Model YZ View – With Rotation - Design 2

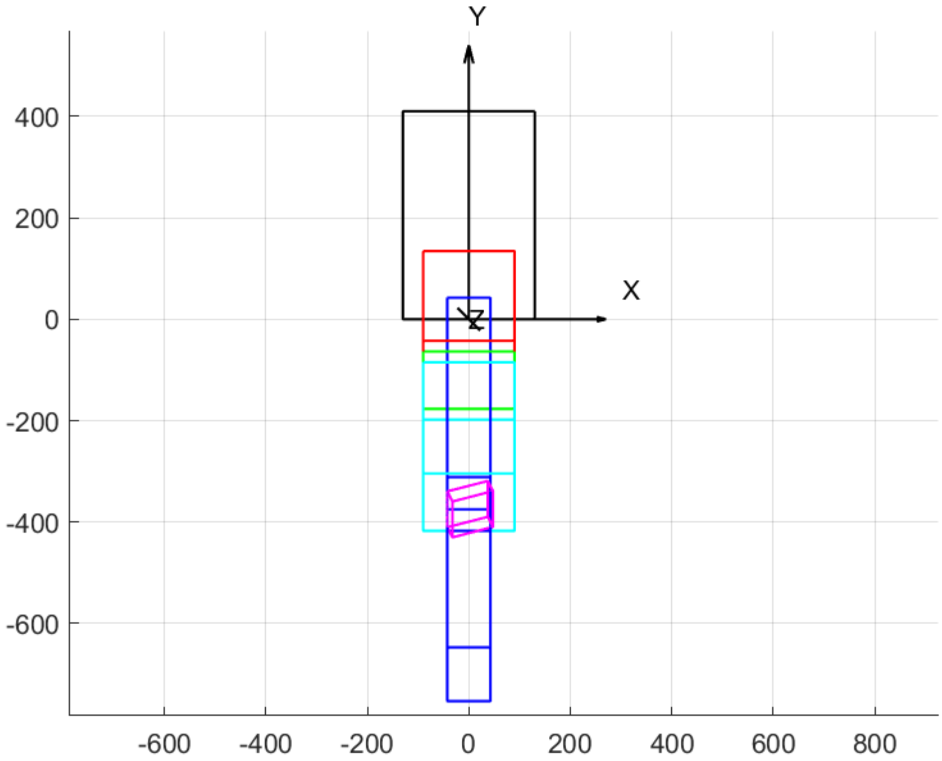


Figure 29 .Vacuum Model XY View – With Rotation - Design 2

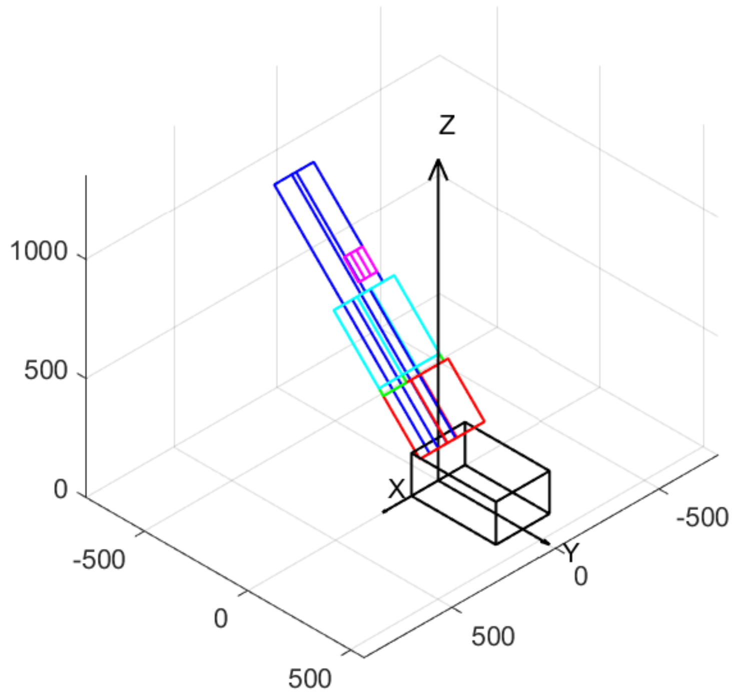


Figure 30. Vacuum Model Isometric View – With Rotation - Design 2

iii. Design 3

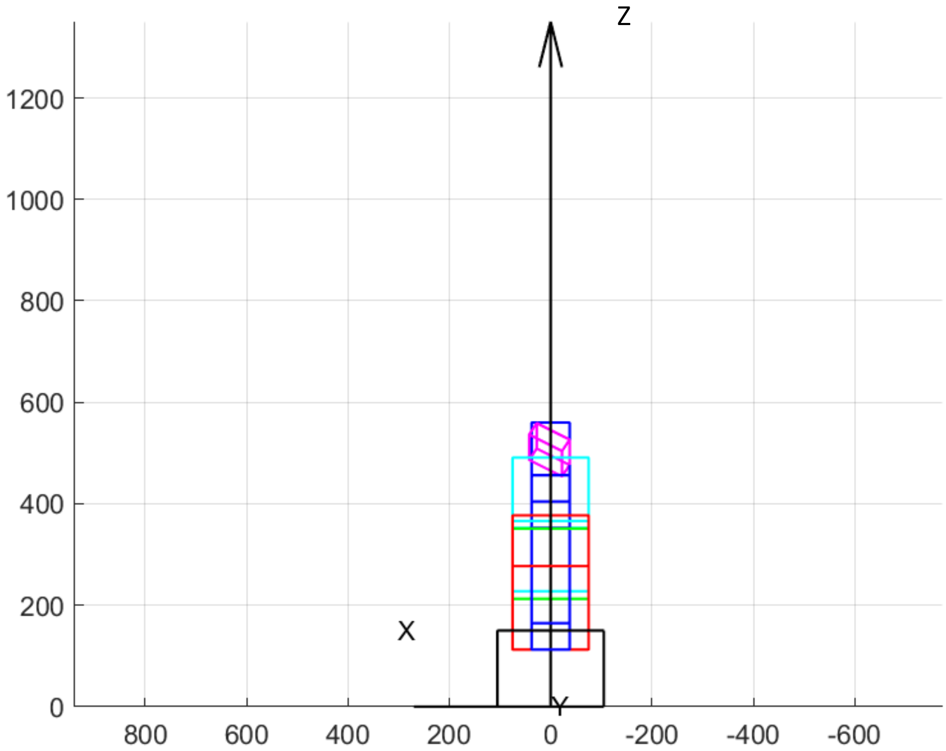


Figure 31.Vacuum Model XZ View – With Rotation - Design 3

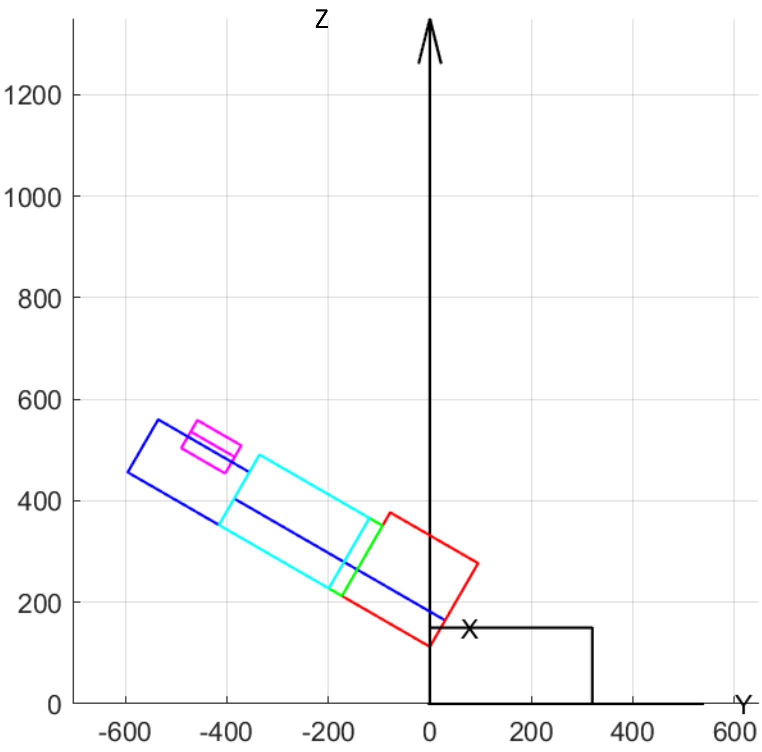


Figure 32.Vacuum Model YZ View – With Rotation - Design 3

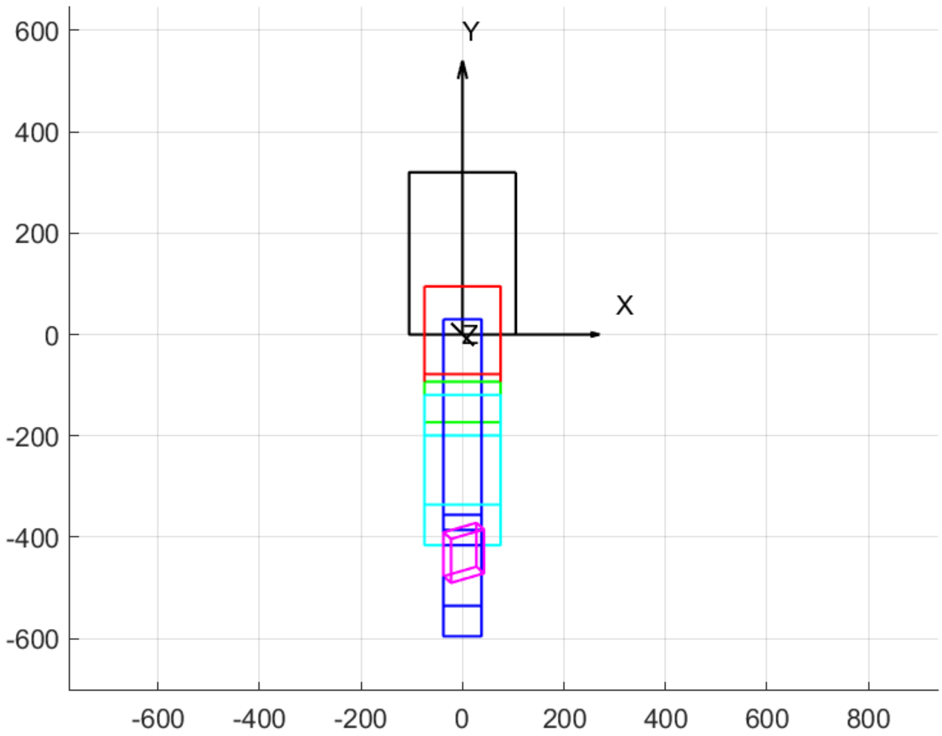


Figure 33 .Vacuum Model XY View – With Rotation - Design 3

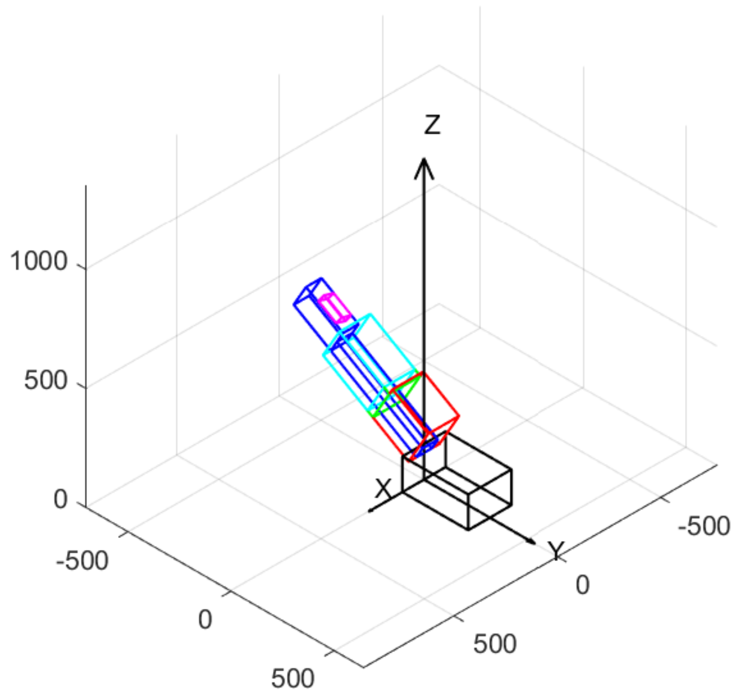


Figure 34. Vacuum Model Isometric View – With Rotation - Design 3

5. CAD Model

The Vacuum is also modeled with a commercial CAD Program.

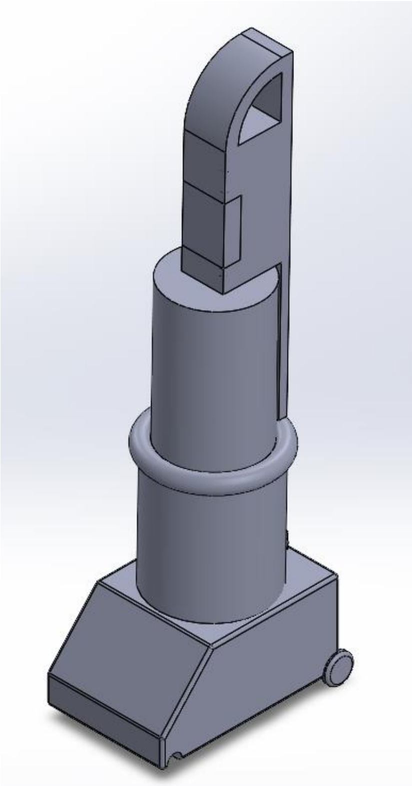


Figure 35. Vacuum Model CAD (No-Rotation)

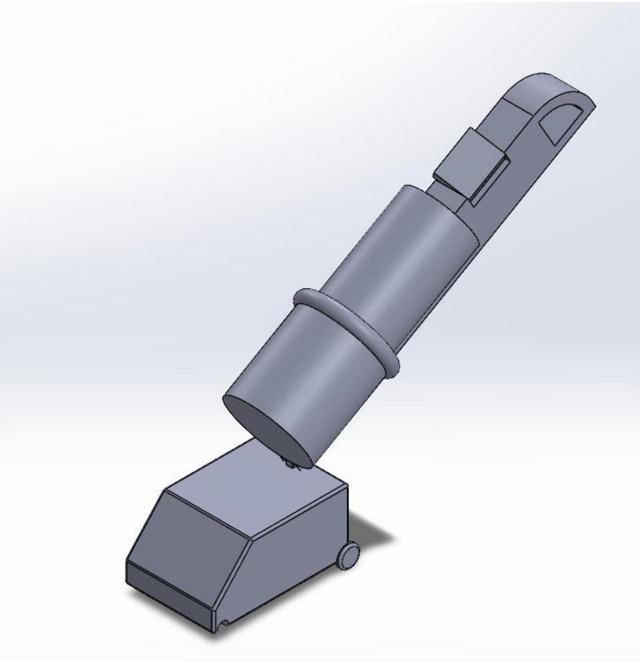


Figure 36. Vacuum Model CAD (With-Rotation)

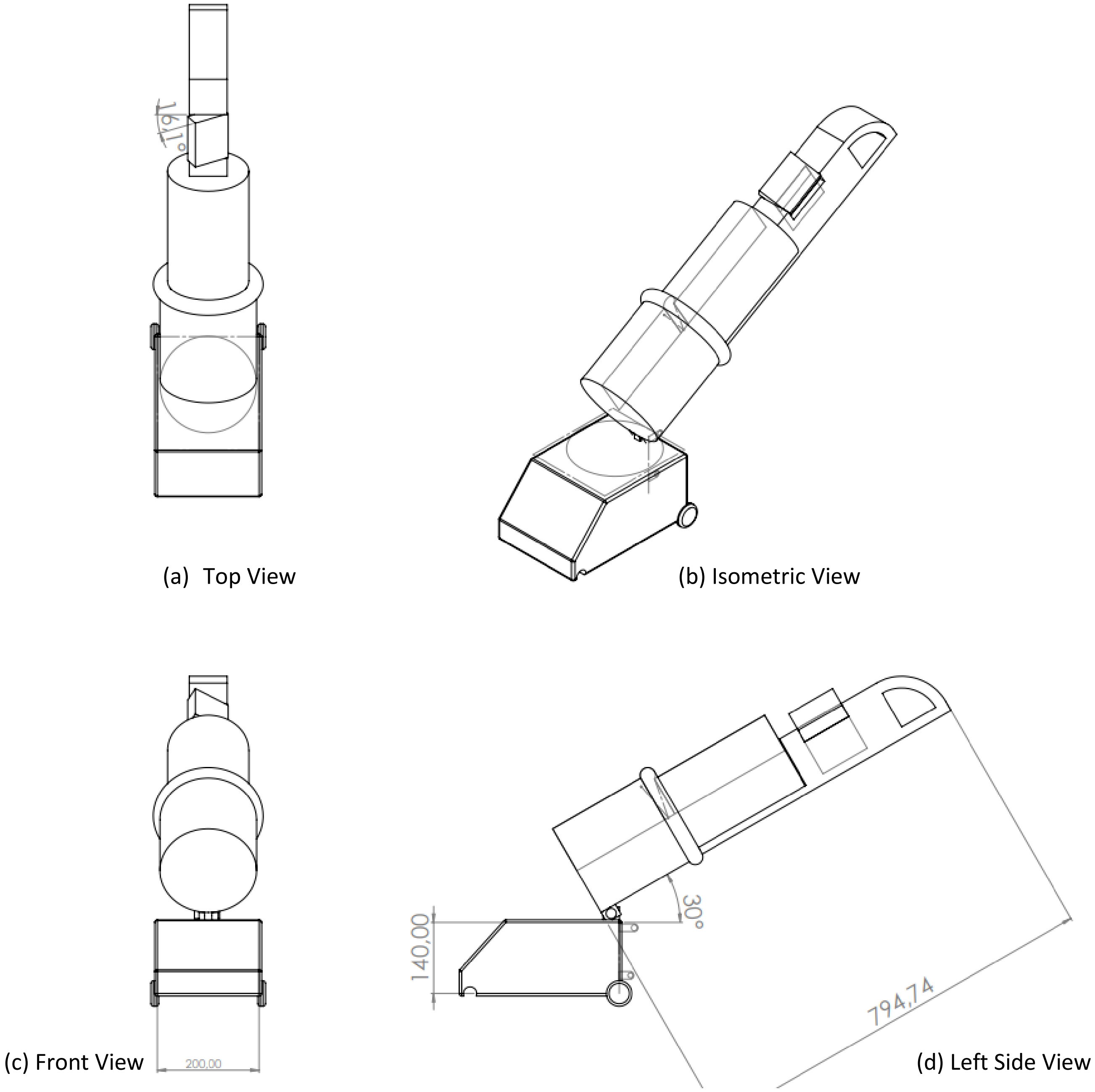


Figure 37. Vacuum Model CAD – 4 View

6. Note

Some shapes are plotted with thicker lines for better visuality.

- Figure 3,4 : LineWidth=2

7. Resources

1. MAK420_HW1_2021-22_Spring.pdf
2. Dr.Recep M.Gorguluarslan,TOBB ETU, MAK420 Windmill Example
3. Dr.Recep M.Gorguluarslan,TOBB ETU, MAK420 Course Notes

8. Scripts

MATLAB scripts created and used can be found in the following pages.

1) Unit_Cube.m

```
clear all
clc
close all
axis equal
hold on
%==========================================================================
%                               DEFINITIONS
%==========================================================================
% 1. PURPOSE: Shows the position of the unit cube

% 2. OUTPUT
% - Plot of the Unit Cube

% 3. Reference:
% Modifed Date: 25/01/2022
% By : Canberk URUŞ

%==========================================================================
%                             SCRIPT BODY
%==========================================================================

%  Plot 3D Isometric view of the Cube.
figure(1);
axis ('equal');
view([1,1,1])
grid on
hold on
val_xlim = 2; val_ylim = 2; val_zlim = 2;
% xlim([-val_xlim val_xlim]); ylim([-val_ylim val_ylim]) ; zlim([0 val_zlim])

% Plot global coordinate system
GCSx = [0 1; 0 0; 0 0];
GCSy = [0 0; 0 1; 0 0];
GCSz = [0 0; 0 0; 0 1];
GCS = [GCSx;GCSy;GCSz];
quiver3 (GCSx(1,1), GCSx(2,1), GCSx(3,1), val_xlim*GCSx(1,2), GCSx(2,2),
GCSx(3,2), 'k',LineWidth=2);
quiver3 (GCSy(1,1), GCSy(2,1), GCSy(3,1), GCSy(1,2), val_ylim*GCSy(2,2),
GCSy(3,2), 'k',LineWidth=2);
quiver3 (GCSz(1,1), GCSz(2,1), GCSz(3,1), GCSz(1,2), GCSz(2,2),
val_zlim*GCSz(3,2), 'k',LineWidth=2);
text(val_xlim*1.01,val_ylim*0.1,val_zlim*0.2,'X',FontWeight='bold')
text(0,val_ylim,1,'Y',FontWeight='bold')
text(0,0,val_zlim+0.1,'Z',FontWeight='bold')

windmill_cube = [-0.5,-0.5,-0.5,-0.5,-0.5,0.5,0.5,0.5,0.5,0.5,0.5,-0.5,-
0.5,0.5,0.5,-0.5;
           0,0,1, 1,0,0,0,1,1,0,0,0,1,1,1,1;
        -0.5,0.5,0.5,-0.5,-0.5,-0.5,0.5,0.5,-0.5,-0.5,0.5,0.5,0.5,0.5,-0.5,-0.5;
           1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1];
unit_cube = [-0.5,-0.5,-0.5,-0.5,-0.5,0.5,0.5,0.5,0.5,0.5,0.5,-0.5,-0.5,0.5,0.5,-
0.5
                0,0,1,1,0,0,0,1,1,0,0,0,1,1,1,1
                0,1,1,0,0,0,1,1,0,0,1,1,1,1,0,0
                1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
                ];

plot3(unit_cube(1,:),unit_cube(2,:),unit_cube(3,:),color='blue',LineWidth=2);
```

2) MAK420_HW1_CANBERK_URUS.m

```
clear all
clc
close all
axis equal
hold on
%==========================================================================
%                                 DEFINITIONS
%==========================================================================
% 1. PURPOSE: Primitave Instancing and Rigid-Body Transformation example --
% Vacuum Cleaner

% 2. INPUT PARAMETERS
% - Design parameters: required parameters in a struct (for 3 designs)

% 3. REQUIRED FUNCTION
% - Plot3D_Vacuum

% 4. OUTPUT
% - Plot of the Vacuum Cleaner

% 5. HELPFUL RESOURCES:
% [1]. Dr.Recep M.Gorguluarslan,TOBB ETU, MAK420 Course Notes
% [2]. Dr.Recep M.Gorguluarslan,TOBB ETU, MAK420 Windmill Example

% 6. Reference:
% Modifed Date: 25/01/2022
% By : Canberk URUŞ

%==========================================================================
%                               SCRIPT BODY
%==========================================================================


%Structs containing parameters for the vacuum design
design1 = struct('A',1000, 'B',160, 'C',220, 'D',300, 'E',110, 'F',220, 'G',150,
'H',80, 'J',340, 'K',140, 'theta1',30, 'theta2',75);
design2 = struct('A',1200, 'B',180, 'C',250, 'D',310, 'E',100, 'F',260, 'G',180,
'H',85, 'J',410, 'K',150, 'theta1',45, 'theta2',20);
design3 = struct('A',800 , 'B',150, 'C',200, 'D',250, 'E',100, 'F',210, 'G',150,
'H',75, 'J',320, 'K',120, 'theta1',60, 'theta2',30);
designs = [design1,design2,design3];

line_Width = 2; % Can be used to make thicker lines (For visuality)
rotation_flag = 'yes'; % 'yes' to rotate, 'no' to not.
%This for loop plots each design in different windows
for i = 1:length(designs)
    thisfig = figure(i);
    Plot3D_Vacuum(designs(i),rotation_flag,line_Width);
end
```

```
function Plot3D_Vacuum(design,rotation_flag,line_Width)
    %==========================================================================
    %                                  DEFINITIONS
    %==========================================================================
    % 1. PURPOSE: Primitave Instancing and Rigid-Body Transformation example --
    % Vacuum Cleaner

    % 2. INPUT
    % Cube              : coordinates of a unit cube
    % Design parameters: required parameters (A,B,C,D,E,F,G,H,J,K,theta1,theta2)
    % Line Width
    % Rotation Flag

    % 3. REQUIERED FUNCTIONS
    % - scale3D             [2]
    % - translate3D         [2]
    % - rotate3Dx           [2]
    % - rotate3Dz           [2]
    % - degree_to_radian    [2]

    % 4. OUTPUT
    % Plot of the Vacuum Cleaner

    % 5. HELPFUL RESOURCES:
    % [1]. Dr.Recep M.Gorguluarslan,TOBB ETU, MAK420 Course Notes
    % [2]. Dr.Recep M.Gorguluarslan,TOBB ETU, MAK420 Windmill Example

    % 6. Reference:
    % Modifed Date: 25/01/2022
    % By : Canberk URUŞ

    %==========================================================================
    %                                FUNCTION BODY
    %==========================================================================

    % Decide rotation
    if strcmpi(rotation_flag,'no')
        design.theta1 = 0;
        design.theta2 = 0;
    end

    %  Requiered primitive shape: cube
    cube = [-0.5,-0.5,-0.5,-0.5,-0.5,0.5,0.5,0.5,0.5,0.5,0.5,-0.5,-0.5,0.5,0.5,-
0.5
            0,0,1,1,0,0,0,1,1,0,0,0,1,1,1,1
            0,1,1,0,0,0,1,1,0,0,1,1,1,1,0,0
            1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
            ];

    %  Plot 3D Isometric view of the Vacuum.
    axis ('equal');
    view([1,1,1])
    grid on
    hold on
    %val_xlim = 1000; val_ylim = 1000; val_zlim = 1500;
    val_xlim = 300; val_ylim = 600; val_zlim = 1500;
    % xlim([-val_xlim val_xlim]); ylim([-val_ylim val_ylim]) ; zlim([0 val_zlim])

    % Plot global coordinate system
```

```
    GCSx = [0 1; 0 0; 0 0];
    GCSy = [0 0; 0 1; 0 0];
    GCSz = [0 0; 0 0; 0 1];
    %GCS = [GCSx;GCSy;GCSz];
    quiver3 (GCSx(1,1), GCSx(2,1), GCSx(3,1), val_xlim*GCSx(1,2), GCSx(2,2),
GCSx(3,2), 'k',LineWidth=line_Width);
    quiver3 (GCSy(1,1), GCSy(2,1), GCSy(3,1), GCSy(1,2), val_ylim*GCSy(2,2),
GCSy(3,2), 'k',LineWidth=line_Width);
    quiver3 (GCSz(1,1), GCSz(2,1), GCSz(3,1), GCSz(1,2), GCSz(2,2),
val_zlim*GCSz(3,2), 'k',LineWidth=line_Width);

    text(val_xlim*1.01,val_ylim*0.1,val_zlim*0.1,'X')
    text(0,val_ylim,1,'Y')
    text(0,2,val_zlim-2,'Z')
    %%% Uncomment to create -X -Y axes
    %     GCSx = [0 -1; 0 0; 0 0];
    %     GCSy = [0 0; 0 -1; 0 0];
    %     quiver3 (GCSx(1,1), GCSx(2,1), GCSx(3,1), val_xlim*GCSx(1,2), GCSx(2,2),
GCSx(3,2), 'k',LineWidth=line_Width);
    %     quiver3 (GCSy(1,1), GCSy(2,1), GCSy(3,1), GCSy(1,2), val_ylim*GCSy(2,2),
GCSy(3,2), 'k',LineWidth=line_Width);
    %     text(-val_xlim*1.01,-val_ylim*0.1,-val_zlim*0.1,'-X')
    %     text(0,-val_ylim,1,'-Y')

    %=======================================
    % PARTS LIST (RESPECTIVELY)
    % - Bottom
    % - Dirty Water (Can Rotate)
    % - Seperator
    % - Clean Water
    % - Handle (Constructed from 2 cubes)
    % - Cover (Can Rotate)
    %=======================================

    %=======
    % Bottom
    %=======
    S_b = scale3D(design.F,design.J,design.B);
    T_b = translate3D(0,0,0);
    b = T_b;
    bottom = b * S_b*cube;
    plot3(bottom(1,:),bottom(2,:),bottom(3,:),color='black',LineWidth=line_Width)


    %============
    % Dirty Water
    %============
    S_dw = scale3D(design.G,190,design.C);
    T_dw = translate3D(0,0,design.B*0.75);
    R_dw = rotate3Dx(degree_to_radian(design.theta1));
    dw = T_dw * R_dw;
    Dirty_Water = b * dw * S_dw * cube;

plot3(Dirty_Water(1,:),Dirty_Water(2,:),Dirty_Water(3,:),color='red',LineWidth=lin
e_Width)
```

```
    %==========
    % Seperator
    %==========
    S_s = scale3D(design.G,160,30);
    T_s = translate3D(0,0,design.C);
    %If rotation is needed,Rotation matrix should be added to the matrix below
    %R_s = rotate3Dx(degree_to_radian(0));
    s = T_s;
    Seperator = b * dw * s * S_s * cube;

plot3(Seperator(1,:),Seperator(2,:),Seperator(3,:),color='green',LineWidth=line_Wi
dth)


    %============
    % Clean Water
    %============
    S_cw = scale3D(design.G,160,design.D);
    T_cw = translate3D(0,0,30);
    %If rotation is needed,Rotation matrix should be added to the matrix below
    %R_cw = rotate3Dx(degree_to_radian(0));
    cw = T_cw;
    Clean_Water = b * dw * s * cw * S_cw * cube;

plot3(Clean_Water(1,:),Clean_Water(2,:),Clean_Water(3,:),color='cyan',LineWidth=li
ne_Width)


    %=====================
    % Handle (Lower-part)
    %=====================
    S_hl = scale3D(design.H,60,design.D+design.C+30);
    T_hl = translate3D(0,0,0); %Notice there is no actual translation applied.
    %R_hl = rotate3Dx(degree_to_radian(design.theta1));
    hl = T_hl;
    Handle_lower = b* dw * hl * S_hl * cube;

plot3(Handle_lower(1,:),Handle_lower(2,:),Handle_lower(3,:),color='blue',LineWidth
=line_Width)

    %====================
    % Handle (Upper-part)
    %====================
    Handle_upper_height = design.A-(design.D+design.C+design.B*0.75+30);
    S_hu = scale3D(design.H,design.K,Handle_upper_height);
    T_hu = translate3D(0,0,design.D+design.C+30);
    %If rotation is needed,Rotation matrix should be added to the matrix below
    %R_hu = rotate3Dx(degree_to_radian(0));
    hu = T_hu;
    Handle_upper = b * dw * hl * hu * S_hu * cube;

plot3(Handle_upper(1,:),Handle_upper(2,:),Handle_upper(3,:),color='blue',LineWidth
=line_Width)


    %=============
    % Cover
    %=============
    S_c = scale3D(design.H,30,design.E);
```

```
    T_c = translate3D(-design.H/2,design.K,40);
    R_c = rotate3Dz(degree_to_radian(design.theta2));
    T = translate3D(0.5,-1,0);
    c = T_c * R_c;
    Cover = b * dw * hl * hu * c * S_c * (T*cube);
    plot3(Cover(1,:),Cover(2,:),Cover(3,:),color='magenta',LineWidth=line_Width)

end
```