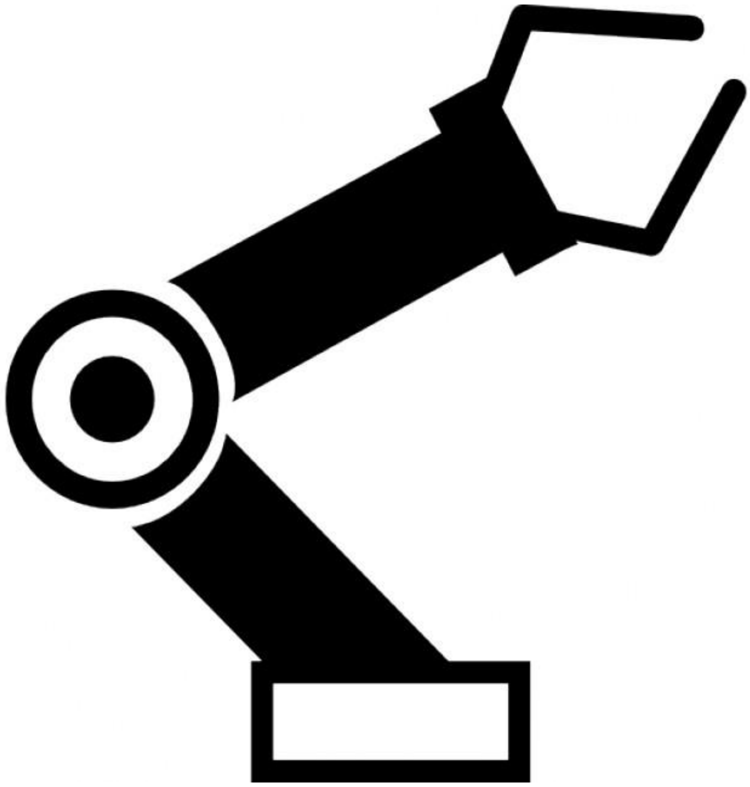
EE587

Introduction to Robotics

Term Project Report



Submitted to:

Prof. Dr. Aydan Erkmen

Student:

Raha Shabani

1993609

Contents

Project Definition

Consider 2 seahorses robot that carry a stick together stack to their wrapped tails.

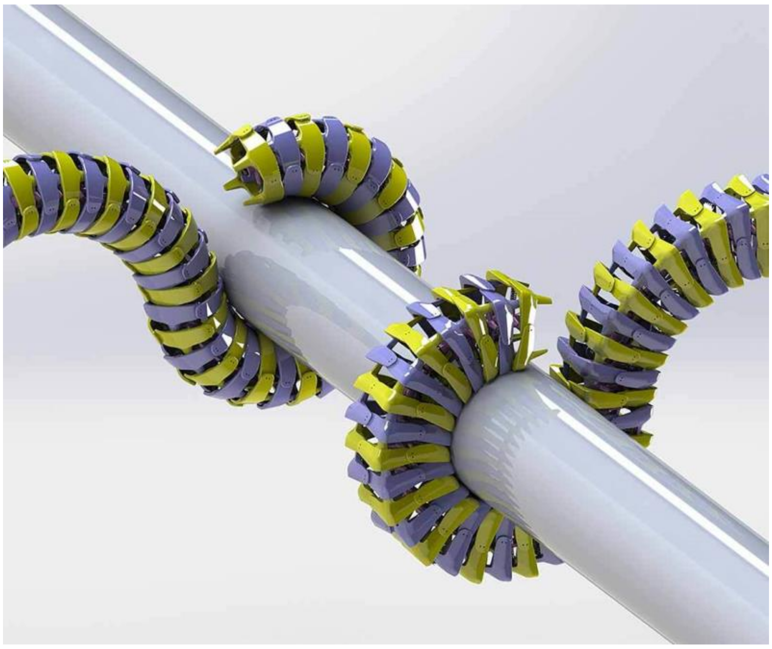


Figure 1 3D model of the tails of 2 seahorses

Assumptions

We Seahorses are of particular interest to robot researchers due to their unusual skeletal structure, which scientists believe could aid in the design of bots that are tough and strong while also being flexible enough to perform tasks in real-world settings.

For this issue there are 2 types based on searching through references and on the internet:

1. First one as shown in the figure 1 has too many joints and small links if we would like to simplify it and if we would like to analysis it on that format it will need expensive method and expensive experimental lab to work on those felexible small joints which has been simulated the body of real seahorse.
2. The second is based on the question' data combining with sea horse toys which is I think suit to this issue as explained in the question:
   - The second modelfor the first format of the assumption I assumed that the tail can be gripper which will be so easy therefore I changed to three joints which are twisted to the stick therefore both seahorses could carry it perfectly.
   - We can understand for doing a task "together" we need common links or joints in robots, on the otherhand it will help us to analysis one seahorse and then conclude the result for both of them.

- Fins of seahorses is shown with 2 other coupled revolute joints, beside this revolute joint other 2 joints one revolute and the other prismatic will shape the body and head of each seahorse.

Even though the water friction is a required for the thrust, the water friction due to other components are negligible. Also the water around is uniform and stationary, i.e no disturbing forces just fluidic force.

DH Parameters

In order to be able to generate DH parameters, we need to identify the end effector. In this case, seperating the system into two will be more appropriate because controlling one seperately cause the control of the other one as we know thses 2 seahorses carry one stick at the same time. In this system, there are 2 end-effectors if we start naming the seahorses from tail to head [1][2].

Since seahorses are identical and their centers overlaps in z direction intially, the DH parameters generated for one can be used in the other.

Since there are multiple end effectors, DH parameters will be generated for the frame which is just before the end effectors, namely the body frame for quadrotor design.

A system is a device which has 6-DOF. 1 of them are prismatic joints and 5 of them are revolute joints. Using Euler angles,

$$q = (X, \Theta)^T where\ X = (x, y, z)^T, \Theta = (\psi, \theta, \phi)^T$$

	α_i	a_i	d_i	θ_i
1	0	a_1	0	θ_1
2	90	a_2	0	θ_2
3	90	a_3	0	θ_3
4	90	a_4	0	θ_4
5	90	a_5	0	θ_5
6	90	0	d_6	0

Kinematics and Dynamics

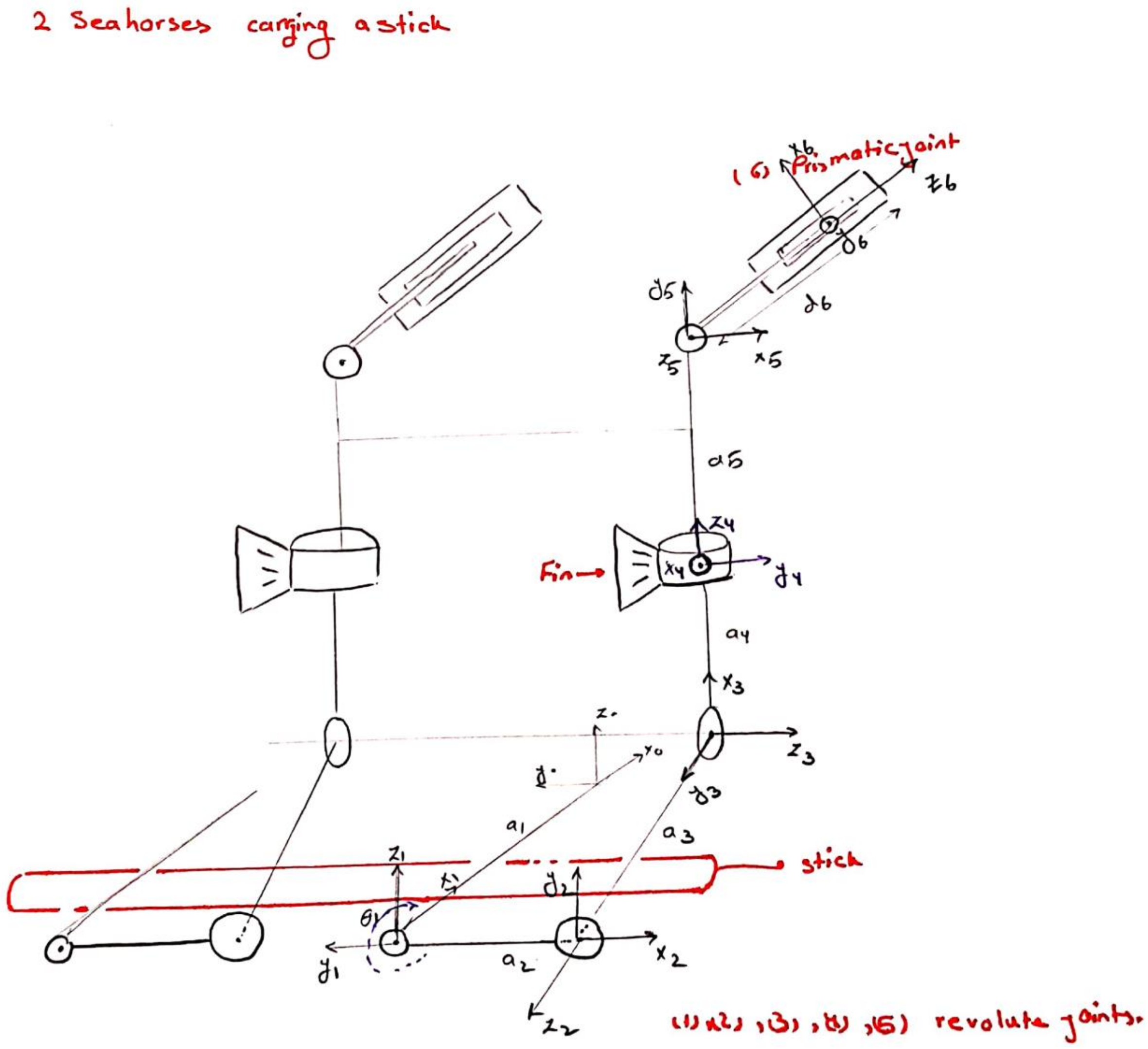


Figure 2: Planar viewof the seahorses system

A commonly used convention for selecting frames of reference in robotic applications is the Denavit-Hartenberg, or DH convention. In this convention, each homogeneous transformation Ai is represented as a product of four basic transformations:

$$A_i = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i}$$

$$A_1^0 = \begin{bmatrix} c\Theta_1 & -s\Theta_1 & 0 & a_1 c\Theta_1 \\ s\Theta_1 & c\Theta_1 & 0 & a_1 s\Theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad A_2^1 = \begin{bmatrix} c\Theta_2 & 0 & s\Theta_2 & a_2 c\Theta_2 \\ s\Theta_2 & 0 & -c\Theta_2 & a_2 s\Theta_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3^2 = \begin{bmatrix} c\Theta_3 & 0 & s\Theta_3 & a_3 c\Theta_3 \\ s\Theta_3 & 0 & -c\Theta_3 & a_3 s\Theta_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad A_4^3 = \begin{bmatrix} c\Theta_4 & 0 & s\Theta_4 & 0 \\ s\Theta_4 & 0 & -c\Theta_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5^0 = \begin{bmatrix} c\Theta_5 & 0 & s\Theta_5 & a_5 c\Theta_5 \\ s\Theta_5 & 0 & -c\Theta_5 & a_5 s\Theta_5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad A_6^5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

From now on although I try to calculate everything manually it will be critical because of the complexity of the system and on the other hand for having trustable evaluation tool I wrote all af these fromula as an mfile in MATLAB and will see the result by comparing to it.

```
% DENAVIT HARTENBERG Parameteres %
clear all
close all
clc
prompt = {'Enter how many robot arms?'};
dlg_title = 'Input';
num_lines = 1;
def = {'1'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
num = str2num(answer{:});
F = sym('A', [num 4]);
B=eye(4);
C = sym('C', [4 4]);
clc
for i=1:num
    prompt = {'Enter a:','Enter alfa:','Enter d:','Enter theta:'};
    dlg_title = sprintf('arm%d',i);
    num_lines = 1;
    def1 =
{sprintf('a%d',i),sprintf('alfa%d',i),sprintf('d%d',i),sprintf('t%d',i)};
    answer1 = inputdlg(prompt,dlg_title,num_lines,def1);
    F(i,1)=answer1(1,1);
    F(i,2)=answer1(2,1);
    F(i,3)=answer1(3,1);
    F(i,4)=answer1(4,1);
    C=simplify([cos(F(i,4)) -sin(F(i,4))*cos(F(i,2)) sin(F(i,4))*sin(F(i,2))
F(i,1)*cos(F(i,4));
             sin(F(i,4)) cos(F(i,4))*cos(F(i,2)) -cos(F(i,4))*sin(F(i,2))
F(i,1)*sin(F(i,4));
             0 sin(F(i,2)) cos(F(i,2)) F(i,3);
             0 0 0 1]);
    eval(sprintf('A%d = C;',i));
    B=B*C;
    eval(sprintf('A%d',i))
end
sprintf('T from Arm 0 to Arm %d is:',i)
pretty(simplify(B))
'R is Rotation Matrix'  , R=B(1:3,1:3);
pretty(R)
d=B(1:3,4);
'd is translation Matrix' , pretty(d)
```

For using this program I use an interface that I can use the data of the D-H table information and insert all the data from the table for each link from the table as shown below:

```
A1 =

[cos(t1), -sin(t1), 0, a1*cos(t1)]
[sin(t1),  cos(t1), 0, a1*sin(t1)]
[      0,        0, 1,          0]
[      0,        0, 0,          1]

A2 =

[cos(t2), -cos(90)*sin(t2),  sin(90)*sin(t2), a2*cos(t2)]
[sin(t2),  cos(90)*cos(t2), -sin(90)*cos(t2), a2*sin(t2)]
[      0,          sin(90),          cos(90),          0]
[      0,                0,                0,          1]

A3 =

[cos(t3), -cos(90)*sin(t3),  sin(90)*sin(t3), a3*cos(t3)]
[sin(t3),  cos(90)*cos(t3), -sin(90)*cos(t3), a3*sin(t3)]
[      0,          sin(90),          cos(90),          0]
[      0,                0,                0,          1]



A4 =

[cos(t4), -cos(90)*sin(t4),  sin(90)*sin(t4), a4*cos(t4)]
[sin(t4),  cos(90)*cos(t4), -sin(90)*cos(t4), a4*sin(t4)]
[      0,          sin(90),          cos(90),          0]
[      0,                0,                0,          1]



A5 =

[cos(t5), -cos(90)*sin(t5),  sin(90)*sin(t5), a5*cos(t5)]
```

```
[sin(t5),  cos(90)*cos(t5), -sin(90)*cos(t5), a5*sin(t5)]
[      0,          sin(90),          cos(90),          0]
[      0,                0,                0,          1]

A6 =

[1, 0, 0,  0]
[0, 1, 0,  0]
[0, 0, 1, d6]
[0, 0, 0,  1]

ans =

    'T from Arm 0 to Arm 6 is:'

-- --
| |  sin(t5) #1 - cos(t5) #4, sin(90) #6 + cos(90) sin(t5) #4 + cos(90) cos(t5) #1,
-- --

  cos(90) #6 - sin(90) sin(t5) #4 - sin(90) cos(t5) #1, a1 cos(t1) + a3 cos(t1 + t2) cos(t3) + a2 cos(t1)
cos(t2) - a2

                                                   2                        a4 #19 cos(t4) sin(t3)
   sin(t1) sin(t2) + a4 cos(t1 + t2) cos(t3) cos(t4) + a4 sin(90)  sin(t1 + t2) sin(t4) - ---------------------
-
                                                                                2

     a4 #20 cos(t4) sin(t3)             3                                                  3
   - ---------------------- + d6 cos(90)  sin(90) sin(t1 + t2) - a3 cos(90) sin(t1 + t2) sin(t3) - d6 sin(90)
               2

                                     2                                 2
   cos(t1 + t2) cos(t5) sin(t3) + a5 sin(90)  cos(t1 + t2) sin(t3) sin(t5) + a5 sin(90)

                              a5 #19 cos(t4) cos(t5) sin(t3)   a5 #20 cos(t4) cos(t5) sin(t3)
   sin(t1 + t2) cos(t5) sin(t4) - ------------------------------ - ------------------------------
```

```
                               2                          2


              3                                  2                                  3
   + d6 sin(90)  sin(t1 + t2) sin(t4) sin(t5) + d6 cos(90)  sin(90) cos(t1 + t2) sin(t3) + d6 cos(90)


                                       3                                  3
   sin(90) sin(t1 + t2) cos(t3) - d6 cos(90) sin(90)  sin(t1 + t2) cos(t4) - d6 cos(90) sin(90)  sin(t1 +
 t2) cos(t5) + a5


               2
   cos(90) sin(90)  sin(t1 + t2) sin(t5) - a4 cos(90) cos(t1 + t2) sin(t3) sin(t4) + a5


                                       2
   cos(t1 + t2) cos(t3) cos(t4) cos(t5) - a4 cos(90)  sin(t1 + t2) cos(t3) sin(t4) + d6


                                      d6 cos(90) sin(90) #19 sin(t3) sin(t4)
   cos(90) sin(90) cos(t1 + t2) cos(t3) sin(t4) - --------------------------------------
                                                          2


     d6 cos(90) sin(90) #20 sin(t3) sin(t4)             2                                          3
   - -------------------------------------- + d6 cos(90)  sin(90) cos(t1 + t2) cos(t4) sin(t3) + d6 cos(90)
                    2


                                             3                                          3
   sin(90) sin(t1 + t2) cos(t3) cos(t4) - d6 cos(90) sin(90)  sin(t1 + t2) cos(t3) cos(t5) - d6 cos(90)
 sin(90)


                                       2                                          2
   sin(t1 + t2) cos(t4) cos(t5) + a5 cos(90) sin(90)  sin(t1 + t2) cos(t3) sin(t5) + a5 cos(90) sin(90)


   sin(t1 + t2) cos(t4) sin(t5) - a5 cos(90) cos(t1 + t2) cos(t3) sin(t4) sin(t5) - a5


   cos(90) cos(t1 + t2) cos(t5) sin(t3) sin(t4) + d6 sin(90) cos(t1


                               a5 cos(90) #19 sin(t3) sin(t4) sin(t5)   a5 cos(90) #20 sin(t3) sin(t4) sin(t5)
   + t2) cos(t3) cos(t4) sin(t5) + -------------------------------------- + --------------------------------------
```

```
                              2                         2

   d6 sin(90) #19 cos(t4) sin(t3) sin(t5)   d6 sin(90) #20 cos(t4) sin(t3) sin(t5)
 - -------------------------------------- - --------------------------------------
                 2                         2


            2                              2                              3
 - a5 cos(90)  cos(t1 + t2) cos(t4) sin(t3) sin(t5) - a5 cos(90)  sin(t1 + t2) cos(t3) cos(t5) sin(t4) - a5
cos(90)


                             2                                   3
 sin(t1 + t2) cos(t3) cos(t4) sin(t5) + d6 cos(90)  sin(90) cos(t1 + t2) cos(t4) cos(t5) sin(t3) + d6
cos(90)


                                  2
 sin(90) sin(t1 + t2) cos(t3) cos(t4) cos(t5) - d6 cos(90)  sin(90) sin(t1 + t2) cos(t3) sin(t4) sin(t5) +
d6

 cos(90) sin(90) cos(t1 + t2) cos(t3) cos(t5) sin(t4) - d6 cos(90) sin(90) cos(t1

                        d6 cos(90) sin(90) #19 cos(t5) sin(t3) sin(t4)
 + t2) sin(t3) sin(t4) sin(t5) - ---------------------------------------------
                                 2


   d6 cos(90) sin(90) #20 cos(t5) sin(t3) sin(t4) --
 - ---------------------------------------------  |,
                   2                 --


 --
 |  cos(t5) #5 - sin(t5) #2, sin(90) #7 - cos(90) sin(t5) #5 - cos(90) cos(t5) #2,
 --

 cos(90) #7 + sin(90) cos(t5) #2 + sin(90) sin(t5) #5, a1 sin(t1) + a3 sin(t1 + t2) cos(t3) + a2 cos(t1)
sin(t2) + a2

                  2                                a4 #21 cos(t4) sin(t3)
```

```
  cos(t2) sin(t1) - a4 sin(90)  cos(t1 + t2) sin(t4) + a4 sin(t1 + t2) cos(t3) cos(t4) + ---------------------
-
                                                                                   2


     a4 #22 cos(t4) sin(t3)             3
  + ---------------------- - d6 cos(90)  sin(90) cos(t1 + t2) + a3 cos(90) cos(t1 + t2) sin(t3) + a5
              2


                                  a5 #21 cos(t4) cos(t5) sin(t3)   a5 #22 cos(t4) cos(t5) sin(t3)
  sin(t1 + t2) cos(t3) cos(t4) cos(t5) + ------------------------------ + ------------------------------
                                                2                                2


            3                              3                               2
  - d6 sin(90)  sin(t1 + t2) cos(t5) sin(t3) - d6 sin(90)  cos(t1 + t2) sin(t4) sin(t5) + a5 sin(90)


                                  3                                  3
  sin(t1 + t2) sin(t3) sin(t5) - d6 cos(90)  sin(90) cos(t1 + t2) cos(t3) + d6 cos(90) sin(90)  cos(t1 +
t2) cos(t4) + d6


              3                              2                              2
  cos(90) sin(90)  cos(t1 + t2) cos(t5) - a5 cos(90) sin(90)  cos(t1 + t2) sin(t5) + d6 cos(90)


                                                                      2
  sin(90) sin(t1 + t2) sin(t3) - a4 cos(90) sin(t1 + t2) sin(t3) sin(t4) + a4 cos(90)  cos(t1 + t2) cos(t3)
sin(t4) - a5


        2
  sin(90)  cos(t1 + t2) cos(t5) sin(t4) + d6 cos(90) sin(90) sin(t1


                        d6 cos(90) sin(90) #21 sin(t3) sin(t4)   d6 cos(90) sin(90) #22 sin(t3) sin(t4)
  + t2) cos(t3) sin(t4) + -------------------------------------- + --------------------------------------
                                    2                                  2


            3                                     3                                   3
  - d6 cos(90)  sin(90) cos(t1 + t2) cos(t3) cos(t4) + d6 cos(90) sin(90)  cos(t1 + t2) cos(t3) cos(t5)
+ d6 cos(90) sin(90)
```

```
                                      2                                    2
   cos(t1 + t2) cos(t4) cos(t5) - a5 cos(90) sin(90)  cos(t1 + t2) cos(t3) sin(t5) - a5 cos(90) sin(90)

                                 2
   cos(t1 + t2) cos(t4) sin(t5) + d6 cos(90)  sin(90) sin(t1 + t2) cos(t4) sin(t3) - a5

   cos(90) sin(t1 + t2) cos(t3) sin(t4) sin(t5) - a5 cos(90) sin(t1

                           a5 cos(90) #21 sin(t3) sin(t4) sin(t5)   a5 cos(90) #22 sin(t3) sin(t4) sin(t5)
   + t2) cos(t5) sin(t3) sin(t4) - -------------------------------------- - --------------------------------------
                                               2                                    2

                                            d6 sin(90) #21 cos(t4) sin(t3) sin(t5)
   + d6 sin(90) sin(t1 + t2) cos(t3) cos(t4) sin(t5) + --------------------------------------
                                                                  2

     d6 sin(90) #22 cos(t4) sin(t3) sin(t5)            2                                    3
   + -------------------------------------- + a5 cos(90)  cos(t1 + t2) cos(t3) cos(t5) sin(t4) + a5 cos(90)
                      2

                                      2
   cos(t1 + t2) cos(t3) cos(t4) sin(t5) - a5 cos(90)  sin(t1 + t2) cos(t4) sin(t3) sin(t5) - d6

                                                 3
   cos(90) sin(90) sin(t1 + t2) sin(t3) sin(t4) sin(t5) - d6 cos(90)  sin(90) cos(t1 + t2) cos(t3) cos(t4)
 cos(t5) + d6

          2                                        2
   cos(90)  sin(90) cos(t1 + t2) cos(t3) sin(t4) sin(t5) + d6 cos(90)  sin(90) sin(t1 + t2) cos(t4) cos(t5)
 sin(t3) + d6

                                              d6 cos(90) sin(90) #21 cos(t5) sin(t3) sin(t4)
   cos(90) sin(90) sin(t1 + t2) cos(t3) cos(t5) sin(t4) + ----------------------------------------------
                                                                          2
```

```
    d6 cos(90) sin(90) #22 cos(t5) sin(t3) sin(t4) --
  + ------------------------------------------  |,
                         2                    --

 [cos(t5) #9 - sin(t5) #8, - sin(90) #10 - cos(90) sin(t5) #9 - cos(90) cos(t5) #8,

 #3, d6 #3 + a3 sin(90) sin(t3) + a4 sin(t4) #17 + a5 cos(t5) #9 - a5 sin(t5) #8 + a4 sin(90) cos(t4)
sin(t3)],

            --
 [0, 0, 0, 1]  |
            --

where

             2
  #1 == sin(90)  #13 + cos(90) sin(t4) #12 - cos(90) cos(t4) #11

             2
  #2 == sin(90)  #16 + cos(90) sin(t4) #15 + cos(90) cos(t4) #14

  #3 == sin(90) sin(t5) #9 - cos(90) #10 + sin(90) cos(t5) #8

  #4 == cos(t4) #12 + sin(t4) #11

  #5 == cos(t4) #15 - sin(t4) #14

  #6 == sin(90) cos(t4) #11 + cos(90) sin(90) #13 - sin(90) sin(t4) #12

  #7 == sin(90) sin(t4) #15 - cos(90) sin(90) #16 + sin(90) cos(t4) #14

  #8 == sin(90) #18 - cos(90) cos(t4) #17 + cos(90) sin(90) sin(t3) sin(t4)

  #9 == sin(t4) #17 + sin(90) cos(t4) sin(t3)
                          2
  #10 == cos(90) #18 - sin(90)  sin(t3) sin(t4) + sin(90) cos(t4) #17
```

```
                                               2                       2
  #11 == cos(90) cos(t1 + t2) sin(t3) - sin(90)  sin(t1 + t2) + cos(90)  sin(t1 + t2) cos(t3)


         #19 sin(t3)   #20 sin(t3)
  #12 == ----------- + ----------- - cos(t1 + t2) cos(t3)
              2             2


  #13 == cos(90) sin(t1 + t2) + cos(t1 + t2) sin(t3) + cos(90) sin(t1 + t2) cos(t3)


                2                                                      2
  #14 == sin(90)  cos(t1 + t2) + cos(90) sin(t1 + t2) sin(t3) - cos(90)  cos(t1 + t2) cos(t3)


                                #21 sin(t3)   #22 sin(t3)
  #15 == sin(t1 + t2) cos(t3) + ----------- + -----------
                                     2             2


  #16 == cos(90) cos(t1 + t2) - sin(t1 + t2) sin(t3) + cos(90) cos(t1 + t2) cos(t3)


  #17 == cos(90) sin(90) + cos(90) sin(90) cos(t3)


                2                  2
  #18 == sin(90)  cos(t3) - cos(90)


  #19 == sin(t1 + t2 - 90)


  #20 == sin(t1 + t2 + 90)


  #21 == cos(t1 + t2 - 90)


  #22 == cos(t1 + t2 + 90)
ans =
```

```
'R is Rotation Matrix'

[[sin(t5) #1 - cos(t5) #4, sin(90) #6 + cos(90) sin(t5) #4 + cos(90) cos(t5) #1,

  cos(90) #6 - sin(90) sin(t5) #4 - sin(90) cos(t5) #1],

  [cos(t5) #3 - sin(t5) #2, sin(90) #5 - cos(90) sin(t5) #3 - cos(90) cos(t5) #2,

  cos(90) #5 + sin(90) sin(t5) #3 + sin(90) cos(t5) #2],

  [cos(t5) #8 - sin(t5) #7, - sin(90) #9 - cos(90) sin(t5) #8 - cos(90) cos(t5) #7,

  sin(90) sin(t5) #8 - cos(90) #9 + sin(90) cos(t5) #7]]

where

  #1 == sin(90) #15 + cos(90) sin(t4) #14 - cos(90) cos(t4) #13

  #2 == sin(90) #12 + cos(90) sin(t4) #11 + cos(90) cos(t4) #10

  #3 == cos(t4) #11 - sin(t4) #10

  #4 == cos(t4) #14 + sin(t4) #13

  #5 == sin(90) sin(t4) #11 - cos(90) #12 + sin(90) cos(t4) #10

  #6 == cos(90) #15 - sin(90) sin(t4) #14 + sin(90) cos(t4) #13

  #7 == sin(90) #17 - cos(90) cos(t4) #16 + cos(90) sin(90) sin(t3) sin(t4)

  #8 == sin(t4) #16 + sin(90) cos(t4) sin(t3)

  #9 == cos(90) #17 - sin(90)^2 sin(t3) sin(t4) + sin(90) cos(t4) #16
```

```
  #10 == sin(90) #20 + cos(90) sin(t3) #19 - cos(90) cos(t3) #18

  #11 == sin(t3) #18 + cos(t3) #19

  #12 == cos(90) #20 + sin(90) cos(t3) #18 - sin(90) sin(t3) #19

  #13 == cos(90) sin(t3) #22 - sin(90) #23 + cos(90) cos(t3) #21

  #14 == sin(t3) #21 - cos(t3) #22

  #15 == cos(90) #23 + sin(90) cos(t3) #21 + sin(90) sin(t3) #22

  #16 == cos(90) sin(90) + cos(90) sin(90) cos(t3)

               2              2
  #17 == sin(90)  cos(t3) - cos(90)

  #18 == cos(90) cos(t1) cos(t2) - cos(90) sin(t1) sin(t2)

  #19 == cos(t1) sin(t2) + cos(t2) sin(t1)

  #20 == sin(90) cos(t1) cos(t2) - sin(90) sin(t1) sin(t2)

  #21 == cos(90) cos(t1) sin(t2) + cos(90) cos(t2) sin(t1)

  #22 == cos(t1) cos(t2) - sin(t1) sin(t2)

  #23 == sin(90) cos(t1) sin(t2) + sin(90) cos(t2) sin(t1)

ans =

   'd is translation Matrix'

[[a1 cos(t1) - d6 (sin(90) sin(t5) #6 - cos(90) (cos(90) #7 - sin(90) sin(t4) #14 + sin(90) cos(t4) #13)
+ sin(90) cos(t5) #1)
```

- a3 sin(t3) #19 + a3 cos(t3) #20 - a5 cos(t5) #6 + a2 cos(t1) cos(t2) - a4 cos(t4) #14 - a2 sin(t1) sin(t2) - a4

sin(t4) #13 + a5 sin(t5) #1],

[a1 sin(t1) + d6 (cos(90) (sin(90) sin(t4) #12 - cos(90) #8 + sin(90) cos(t4) #11) + sin(90) sin(t5) #5 + sin(90) cos(t5) #2)

+ a3 sin(t3) #16 + a3 cos(t3) #17 + a5 cos(t5) #5 + a4 cos(t4) #12 + a2 cos(t1) sin(t2) + a2 cos(t2) sin(t1) - a4

sin(t4) #11 - a5 sin(t5) #2],

2

[d6 (sin(90) sin(t5) #4 - cos(90) (cos(90) #9 - sin(90) sin(t3) sin(t4) + sin(90) cos(t4) #10) + sin(90) cos(t5) #3)

+ a3 sin(90) sin(t3) + a4 sin(t4) #10 + a5 cos(t5) #4 - a5 sin(t5) #3 + a4 sin(90) cos(t4) sin(t3)]]

where

#1 == sin(90) #7 + cos(90) sin(t4) #14 - cos(90) cos(t4) #13

#2 == sin(90) #8 + cos(90) sin(t4) #12 + cos(90) cos(t4) #11

#3 == sin(90) #9 - cos(90) cos(t4) #10 + cos(90) sin(90) sin(t3) sin(t4)

#4 == sin(t4) #10 + sin(90) cos(t4) sin(t3)

#5 == cos(t4) #12 - sin(t4) #11

#6 == cos(t4) #14 + sin(t4) #13

#7 == cos(90) #18 + sin(90) cos(t3) #19 + sin(90) sin(t3) #20

#8 == cos(90) #15 + sin(90) cos(t3) #16 - sin(90) sin(t3) #17

```
         2                  2
#9 == sin(90)  cos(t3) - cos(90)

#10 == cos(90) sin(90) + cos(90) sin(90) cos(t3)

#11 == sin(90) #15 + cos(90) sin(t3) #17 - cos(90) cos(t3) #16

#12 == sin(t3) #16 + cos(t3) #17

#13 == cos(90) sin(t3) #20 - sin(90) #18 + cos(90) cos(t3) #19

#14 == sin(t3) #19 - cos(t3) #20

#15 == sin(90) cos(t1) cos(t2) - sin(90) sin(t1) sin(t2)

#16 == cos(90) cos(t1) cos(t2) - cos(90) sin(t1) sin(t2)

#17 == cos(t1) sin(t2) + cos(t2) sin(t1)

#18 == sin(90) cos(t1) sin(t2) + sin(90) cos(t2) sin(t1)

#19 == cos(90) cos(t1) sin(t2) + cos(90) cos(t2) sin(t1)

#20 == cos(t1) cos(t2) - sin(t1) sin(t2)
```

Z'Y'X' Euler Rotation matrix is:

$$R_{z'y'x'} = \begin{bmatrix} \mathrm{c}_\theta\ \mathrm{c}_\psi & c_\psi s_\theta\ s_\phi - c_\phi s_\psi & s_\phi s_\psi + c_\phi c_\psi s_\theta \\ c_\theta\ s_\psi & c_\phi c_\psi + s_\theta s_\phi s_\psi & c_\phi s_{\theta s_\psi} - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix}$$

And due to prismatic joints at the origin of the fixed frame.

The homogeneous transformation matrix of body frame(origin is at the center) to fixed frame will be:

$$T_B^0 = \begin{bmatrix} \mathrm{c}_\theta\ \mathrm{c}_\psi & c_\psi s_\theta\ s_\phi - c_\phi s_\psi & s_\phi s_\psi + c_\phi c_\psi s_\theta & d1 \\ c_\theta\ s_\psi & c_\phi c_\psi + s_\theta s_\phi s_\psi & c_\phi s_{\theta s_\psi} - c_\psi s_\phi & d2 \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi & d3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

However, this does not give us the pozition and orientation of the end effectors, this transformation gives us the body frame which has a origin at the center of the fixed frame. For the end effectors, we need to translate the frame in the ± x and ± y with respect to the body frame.

So by post-multiplying the T with Trans $_{\pm Xb}$ and Trans $_{\pm Yb}$, we can obtain the resulting end-effector frames.

$$\mathrm{Trans}_{\pm \mathrm{XB}} = \begin{bmatrix} 1 & 0 & 0 & \pm \mathrm{L_q} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{Trans}_{\pm \mathrm{YB}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & \pm \mathrm{L_q} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$X_B \ and\ Y_B\ are\ the\ axes\ with\ respect\ to\ the\ body\ frame.$$

$$T^0_{M_{1l}} = T^0_B * \mathrm{Trans}_{+\mathrm{XB}}$$

$$T^0_{M_{1u}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & +L_b \\ 0 & 0 & 0 & 1 \end{bmatrix} * T^0_B * \mathrm{Trans}_{+\mathrm{XB}}$$

Dynamics and the Jacobian

As stated above, since this is 6-DOF device its joint matrix will be :

$$\boldsymbol{q} = (X, \Theta)^T where\ X = (x, y, z)^T\ wrt\ fixed\ frame, \Theta = (\psi, \theta, \phi)^T$$

ψ is the yaw angle around the z-axis, θ is the pitch angle around the y-axis and φ is the roll angle around the x-axis.

Since controlling the seahorses is easier than directly trying to control the end effectors, I will generate the dynamics for the system. By doing so, only the potential energy term will differ from the dynamics of a single seahorse.

The Lagrangian in this system consists of 3 parameters:

$$L(\mathbf{q}, \boldsymbol{q}') = K_{\mathrm{trans}} + K_{\mathrm{rot}} - U$$

$$K_{\mathrm{trans}} = \frac{\mathrm{m}}{2} * X'^T * X'$$

$$K_{rot} = \frac{1}{2} * \Theta'^T * I * \Theta'$$

$$U = mgz + U_1 + U_2 + U_3 + U_4$$

where m is the mass of the body and I is the inertia matrix.

For the rotational kinetic energy:

$$\Theta = W(q) * q'\ where\ W\ is\ the\ rotational\ part\ of\ the\ jacobian.$$

This equation is valid since rotation on a body is independent from the X.

$$W(q) = \begin{bmatrix} -s_\theta & 0 & 1 \\ c_\theta s_\phi & c_\phi & 0 \\ c_\theta c_\phi & s_\phi & 0 \end{bmatrix}$$

Which results in

$$\Theta = \begin{bmatrix} \phi' - \psi' s_\theta \\ \theta' c_\phi + \psi' c_\theta s_\phi \\ \psi' c_\theta c_\phi - \theta' s_\phi \end{bmatrix}$$

And since the body is modeled as thin rods:

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

Substituting the relation above in the rotational kinetic energy formulation:

$$K_{rot} = \frac{1}{2} * q'^T * W^T * I * W * q'$$

Lagrange equation with respect to q:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial q'}\right) - \left(\frac{\partial L}{\partial q}\right) = \begin{pmatrix} F_X \\ \tau \end{pmatrix}$$

Where F_X is the translational forces on the seahorses caused by the thrust force of the fins and τ is the moments in the direction of yaw, pitch and roll with respect to the center of the seahorses frame.

Since initially heads of seahorses are horizontal and all the forces acting in the +z direction. Since there is 2 seahorses, net force generated by them become zero:

$$F_B = \begin{bmatrix} 0 \\ 0 \\ u \end{bmatrix} \; where\; u = \sum_{i=1}^{4} F_{Mi}$$

and

$$F_X^0 = R * F_B$$

Considering the torques:

$$\tau = \begin{bmatrix} \tau_{M1} + \tau_{M2} + \tau_{M3} + \tau_{M4} \\ (F_{M2} - F_2 . \cos\phi . \cos\theta - F_{M4} + F_4 \cos\phi \; \cos\theta) * L_q \\ (F_{M3} - F_3 \cos\phi \; \cos\theta - F_{M1} + F_1 \cos\phi \; \cos\theta) * L_q \end{bmatrix}$$

$$where\; \tau_{Mi}\; is\; the\; torques\; generated\; by\; the\; rotation\; of\; the\; motor\; system$$

$$F_i\; is\; the\; forces\; due\; to\; the\; fins\; connected\; to\; the\; seahorse\; and\; has\; only$$

$$+ z\; component\; due\; to\; 3rd\; part\; of\; assumption\; 2\; .$$

$$|F_i| = (M_i - M_i') * 2\; due\; to\; symmety$$

$$|F_1| = |F_3| = \left(\frac{L_b}{2} * \sin\theta_u + \frac{L_b}{2} * \sin\theta_l\right) * 2$$

$$|F_2| = |F_4| = \left(\frac{L_b}{2} * \sin\phi_u + \frac{L_b}{2} * \sin\phi_l\right) * 2$$

$$u\; and\; l\; represents, upper\; and\; lower\; part\; of\; these\; seahorses.$$

And also this causes change in the potential energy:

$$U_1 = U_3 = \frac{k}{2} * \left(\frac{L_b}{2} * \sin\theta_u + \frac{L_b}{2} * \sin\theta_l\right)^2$$

$$U_2 = U_4 = \frac{k}{2} * \left(\frac{L_b}{2} * \sin\phi_u + \frac{L_b}{2} * \sin\phi_l\right)^2$$

The lagrangian can be seperated in to two in terms of X and q since they contain no cross terms in the kinematic energy combining X and q [1].

$$\frac{d}{dt}\left(\frac{\partial L_{trans}}{\partial X'}\right) - \left(\frac{\partial L_{trans}}{\partial X}\right) = F_X$$

Then

$$mX'' + mgX(3) = F_x^0$$

$$mx'' = u(\sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta)$$

$$my'' = u(\cos\phi\sin\theta\sin\psi - \cos\psi\sin\phi)$$

$$mz'' = u\cos\theta\cos\phi - mg$$

In terms of q:

$$\frac{d}{dt}\left(\frac{\partial L_{rot}}{\partial q'}\right) - \left(\frac{\partial L_{rot}}{\partial q}\right) = \tau$$

$$\frac{d}{dt}\left(* q'^T * W^T * I * W * \frac{\partial q'}{\partial q}\right) - \frac{1}{2}\left(\frac{\partial q'^T * W^T * I * W * q'}{\partial q} + \frac{\partial(\sum_{i=1}^{4} U_i)}{\partial q}\right) = \tau$$

$$W^T * I * W * q'' + (W^T * I * W)' * q' - 1/2 * (\frac{\partial(q'^T * W^T * I * W * q')}{\partial q} + \frac{\partial\left(\sum_{i=1}^{4} U_i\right)}{\partial q}) = \tau$$

$$Def.\ C(q,q') = (W^T * I * W)' * q' - 1/2 * (\frac{\partial(q'^T * W^T * I * W * q')}{\partial q} + \frac{\partial\left(\sum_{i=1}^{4} U_i\right)}{\partial q})$$

$$(W^T * I * W)q'' + C(q,q') = \tau$$

$$\theta'' = W_\theta * \tau_\theta$$

$$\phi'' = W_\phi * \tau_\phi$$

In short, the overall dynamic model for the system:

$$\begin{bmatrix} mI_{3x3} & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} V' \\ w' \end{bmatrix} + \begin{bmatrix} w\ x\ mV \\ w\ x\ Iw \end{bmatrix} = \begin{bmatrix} F \\ \tau \end{bmatrix}$$

And force of gravity by two seahorses:

$$F_g = \left(mg + (Z_{seahorse1} - Z_{seahorse2})\right)\begin{bmatrix} -\sin\theta \\ \cos\theta sin\phi \\ cos\theta cos\phi \end{bmatrix}_{body}$$

The overall system mechanic is:

$$\begin{bmatrix} X'' \\ Y'' \\ Z'' \end{bmatrix} = -\frac{1}{m}\begin{bmatrix} 0 \\ 0 \\ F_{Thrust} \end{bmatrix} + \left(mg + 4 * (Z_{seahorse1} - Z_{seahorse2})\right)\begin{bmatrix} -\sin\theta \\ \cos\theta sin\phi \\ cos\theta cos\phi \end{bmatrix} - \begin{bmatrix} \theta' Z' - \psi' Y' \\ \psi' X' - \phi' Z' \\ \phi' Y' - \theta' X' \end{bmatrix}$$

$$\phi'' = \sum \frac{\tau_x}{I_{xx}} = [\theta'\phi'\left(I_{yy} - I_{zz}\right) + r(T_2 + T_4)\left(\frac{1}{I_{xx}}\right)$$

$$\theta'' = \sum \frac{\tau_y}{I_{yy}} = [\phi'\psi'(I_{zz} - I_{xx}) + r(T_1 + T_3)\left(\frac{1}{I_{yy}}\right)$$

$$\psi'' = \sum \frac{\tau_z}{I_{zz}} = [\phi'\theta'\left(I_{xx} - I_{yy}\right) + r(T_1 + T_3 - T_2 - T_4)\left(\frac{1}{I_{zz}}\right)$$

These quations is applicable for both seahorses and upper and lower parts of their body. Since we are calculating the spring force due to the world frame distances, it can be put into account directly.

Control

For the control of the system, I first considered implementing the dynamics in MATLAB then simulating inverse dynamics control on it. But then I decided to use professional robotic simulator for this purpose because my dynamic model has lots of flaws: for example I do no know the inertias I_{xx} , I_{yy} , I_{zz} or the mass of a links fins joints m_{total}. And in addition to them, inverse dynamics control would not be robust enough to compansate the actual water thrust applied to the upper part of seahorses in real world. Even though it works in matlab, it probably would not work in actual world. From this point on, the assumptions I made above will not be true because I implemented the control section directly in the simulation.

Because of the reasons I mentioned above, I searched for a robotic simulation which has decent quadrotor design and require less prior experience. I found V-REP(Virtual Robot Experimentation Platform) Robotic Simulation of Coppelia Robotics [3] whose educational version is free for the students. In addition to that, it can work on Windows, macOS and Ubuntu, it has both 32 and 64 bit versions. The most important part for me was the simulation environment can be controlled by python, Java, C, C#, C++, Matlab and Lua. It has remote api libraries for all the languages mentioned. There are 5 different types of physics engines used on it and it allows the user to choose one of them for the simulation. In the project I preffered using Lua scripting language which is the native script language supported by V-rep (no remote api required) and Bullet 2.78 pyhsics engine.

V-rep has a lot of predefined models based on commercial products and some some tools to creat your own if it is not available there.

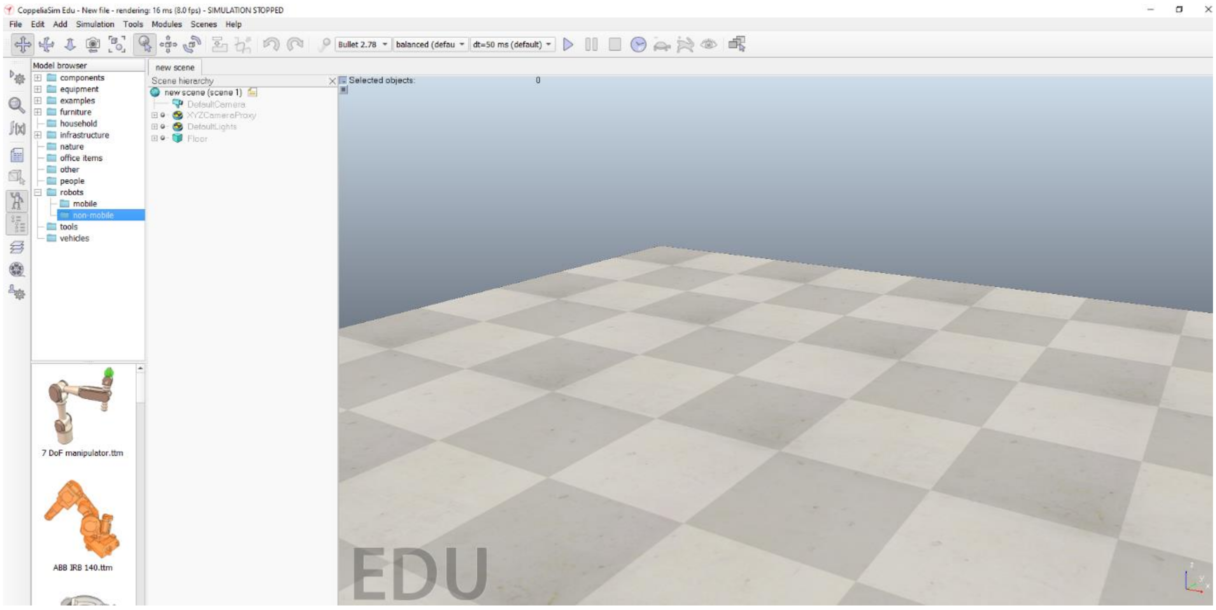


Figure 3: V-rep environment

The most problematic part for the simulation was implementing. There are scene models on the simulation but they were not suitable for the system because of their dimensions. So, I changed the dynamic model script of the prismatic joint like a pump to generate a force and add this force as thrust for the upper part of two seahorses. I left the spring constant as parameter so I can also observe the effect of the spring constant.

The location of the upper joints of two seahorses is not affected by the lower part joints because the lower part do the grabing stick throught the mission and upper part like a propler or a movement motor. Head propellers of the two seahorses rotate only in one direction, meaning that they can generate a force only in the +Y direction. But the lower joints cannot generate a force against the force pulling it up. When I increased the mass by the changing the stick size and angles of lower part joints, it changed all the dymamic model. If the propellers become reversible, since they have friction on them, they could not work properly at lower speeds because it can not carry and manage the movement and mission of seahorses.

The code generating dynamic behvious of the dynamic prismatic joint (like a spring) can be seen below:

```
-- Apply a reactive force onto the body:
totalExertedForce=particleCnt*particleDensity*particleVelocity*
                        math.pi*particleSize*particleSize*particleSize/(6*ts)
springForce=springDistance*springConstant
if(springDirection==0) then
    force={0,0,totalExertedForce-springForce}
else
    force={0,0,totalExertedForce+springForce}
end
```

Figure 4: Prismatic force dynamics applied, prismatic direction is 0 for the upper part of two seahorses system.

After implementing the prismatic joints as springs, I began to implement controllers. The model itself had primitive P controller for altitude(Z), X, Y and yaw. I developed my controller based on them.

I thought implementing force feedback(feedback from spring) and inverse dynamic controller based on my model but I could not do that because actual two seahorses model were different than mine and system become unstable with that controller due to uncertainity of trusters (fins), they have gyroscopic effect, generate water flow.

So I generated dicrete PID controller which is called everytime the simulation iterates. The code segments can be seen from the figure below. The particleTargetVelocities are the required propeller velocities to generate required torques and thrust forces. The other variables are the p,i,d parameters for the attitude control. ‘p’ prefix (as in pAlphaE) represents the previous iteration values. As a convention of the V-rep simulation, I used alpha for Euler angle around X-axis and Beta for the Euler angle around the Y-Axis.

```
particlesTargetVelocities={0,0,0,0}

pParam=2
iParam=0
dParam=0
vParam=-2

cumul=0
lastE=0
pAlphaE=0
pBetaE=0
psp2=0
psp1=0

prevEuler=0
```

Figure 5: Control constants and variables

The main purpose of the altitude (in our system depth) controller is making the error between the target object position’s altitude and altitude of itself zero. For this, error e is calculated for vertical control and fed back as thrust variable to the propellers. The parameters for this are p=2, i=0, d=0. The other constants like 5.335 or vParam were the constants given by the simulation itself.

For horizontal control, first I needed to get the rotation matrix because the direct position control should be done in X and Y directions. For this I first calculated the required roll(alpha) and pitch(beta) angles then using PID control I controled the velocities in the X and Y direction with respect to the two seahorses frame. I controlled the speeds because the kinematics allows motions only in the roll and pitch direction, the velocities in X and Y are linear functions of pitch and roll respectively. The PID parameters was p=0.25 and d=2.1.

```
    -- Vertical control:
    targetPos=simGetObjectPosition(targetObj,-1)
    pos=simGetObjectPosition(d,-1)

    posRel=simGetObjectPosition(d,d0)

    l=simGetVelocity(heli)
    e=(targetPos[3]-pos[3])
    cumul=cumul+e
    pv=pParam*e
    thrust=5.335+pv+iParam*cumul+dParam*(e-lastE)+l[3]*vParam
    lastE=e

    -- Horizontal control:
    sp=simGetObjectPosition(targetObj,d)
    m=simGetObjectMatrix(d,-1)

    vx={1,0,0}
    vx=simMultiplyVector(m,vx)
    vy={0,1,0}
    vy=simMultiplyVector(m,vy)

    alphaE=(vy[3]-m[12])
    alphaCorr=0.25*alphaE+2.1*(alphaE-pAlphaE)

    betaE=(vx[3]-m[12])
    betaCorr=-0.25*betaE-2.1*(betaE-pBetaE)

    pAlphaE=alphaE
    pBetaE=betaE

    alphaCorr=alphaCorr+sp[2]*0.005+1*(sp[2]-psp2)
    betaCorr=betaCorr-sp[1]*0.005-1*(sp[1]-psp1)

    psp2=sp[2]
    psp1=sp[1]
```

Lastly, there were a yaw controller implemented in the model. After calculating correction variables, the required propeller speeds are calculated. As can be seen from the code below, each propeller is affected from thrust differently because their effect on motion is unique.

```
-- Rotational control:
euler=simGetObjectOrientation(d,targetObj)
rotCorr=euler[3]*0.1+2*(euler[3]-prevEuler)
prevEuler=euler[3]

-- Decide of the motor velocities:
particlesTargetVelocities[1]=thrust*(1-alphaCorr+betaCorr+rotCorr)
particlesTargetVelocities[2]=thrust*(1-alphaCorr-betaCorr-rotCorr)
particlesTargetVelocities[3]=thrust*(1+alphaCorr-betaCorr+rotCorr)
particlesTargetVelocities[4]=thrust*(1+alphaCorr+betaCorr-rotCorr)
```

Figure 6: Yaw controller and the pump speed calculation. The effect of the pum positions are shown in red rectangles

After tuning the PID controller, the model follows a “dummy” (V-Rep object which represents a coordinate frame with its origin and rotations). The output of the altitude(Z), X and Y control responses can be seen from the graph below:

Figure 7: Altitude response

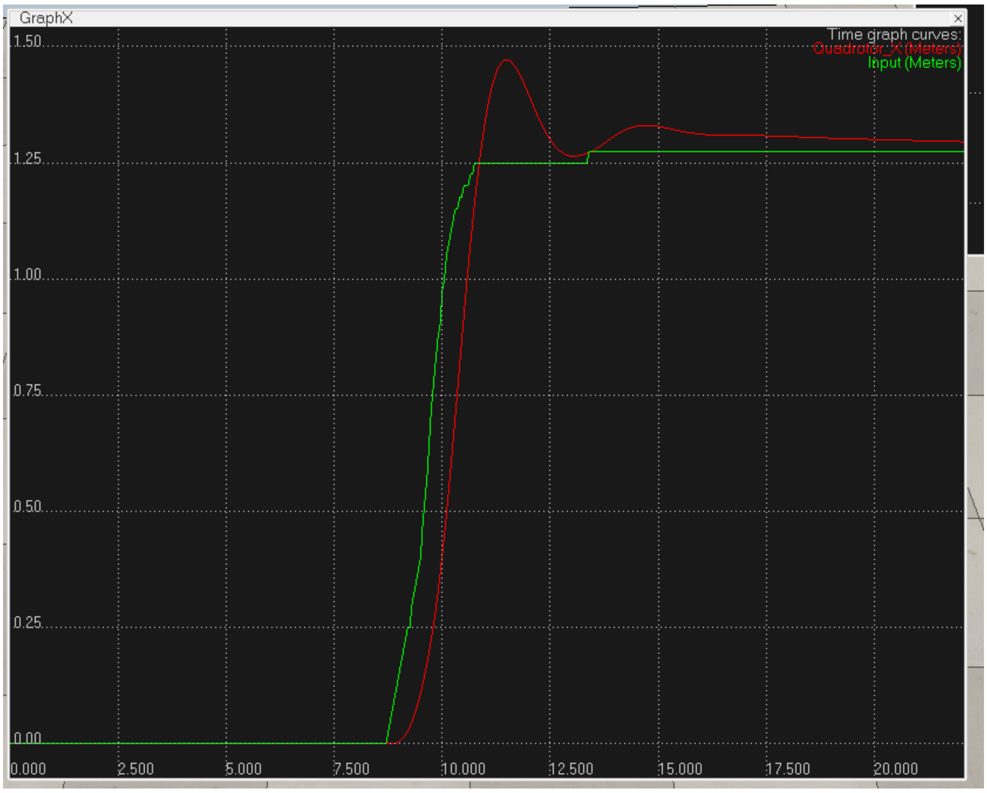


Figure 8: X control response of the two seahorses

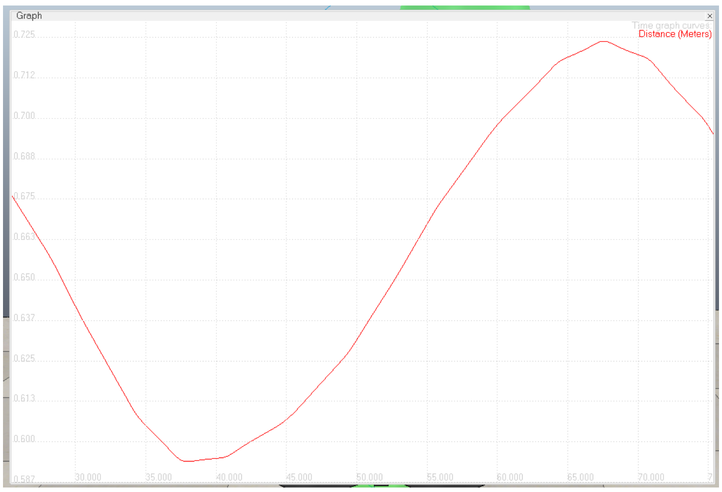
Figure 9: Y control reponse of the two seahorses

Steady-state error due to spring can be seen at the end of the graph. I tried to minimize it, but since the output of the system will be the difference of the positions of the two seahorses, the steady state term will cancel at the output since both seahorses behaves identical in X-Y control.

Since I could control a two seahorses in 3D(X,Y,Z), I implemented a path following algoritm. There was a path facility implemented in V-rep which generates intepolated points between the coordinates taken from the user.

Since the desired position is changing, the two seahorses controllers are following them, by doing so they also follows the path. I implemented this behaviour by changing the rotation speed i.e the period of the target position on the path.

I generated the output graph by measuring the distance with respect to the world frame since the velocity is actually the function of this distance because the base area is constant in this system.
Some of the output results can be seen below:



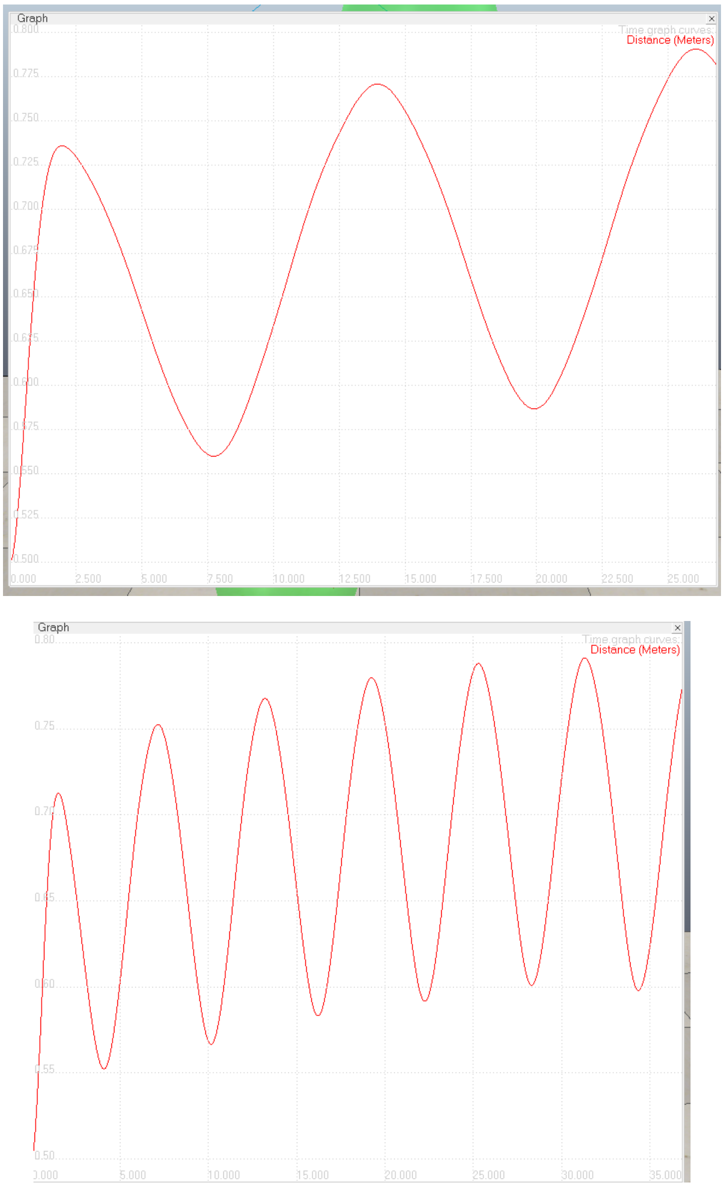


Figure 10: The output of the system with 3 different path following period(0.1,0.05,0.01 s), from highest to smallest

Conclusion

In this project, I created a dynamic model of a two-seahorse system and used a PID controller to control it so that it followed a closed-path with a variable period. This project enabled me to quickly learn how to generate D-H parameters and comprehend the dynamic behavior of a two-seahorse system. It also taught me how to use the LUA scripting language and how to utilize the V-Rep professional robot simulator. I had previously used discrete PID controllers to control position, but in this project I utilized a PID controller to manage velocity. I also obtain path planning and tracking experience. Overall, this project taught me a lot about robotic systems and helped me grasp the topics presented in class better.

All files related to this project can be found on:

https://github.com/users/RahaShabani/projects

References

[1] M. Spong, S. Hutchinson and M. Vidyasagar, *ROBOT MODELING AND CONTROL*, 1st ed. JOHNWILEY & Sons, 2012.

[2] Holt, Justin Dakota, "Design and Testing of a Biomimetic Pneumatic Actuated Seahorse Tail Inspired Robot" (2017). *All Theses*. 2637. https://tigerprints.clemson.edu/all_theses/2637

[3] C. Robotics, «VREP,» Coppelia Robotics Inc., 05 02 2022. Available: http://www.coppeliarobotics.com.