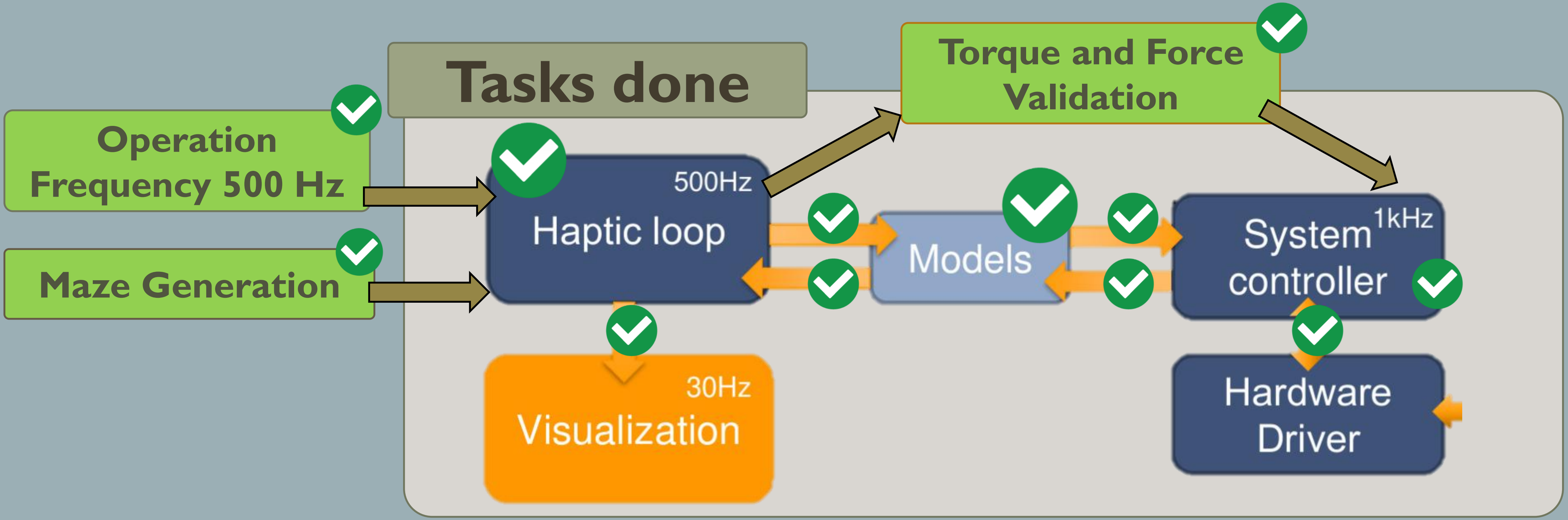
# HAPTIC LOOP

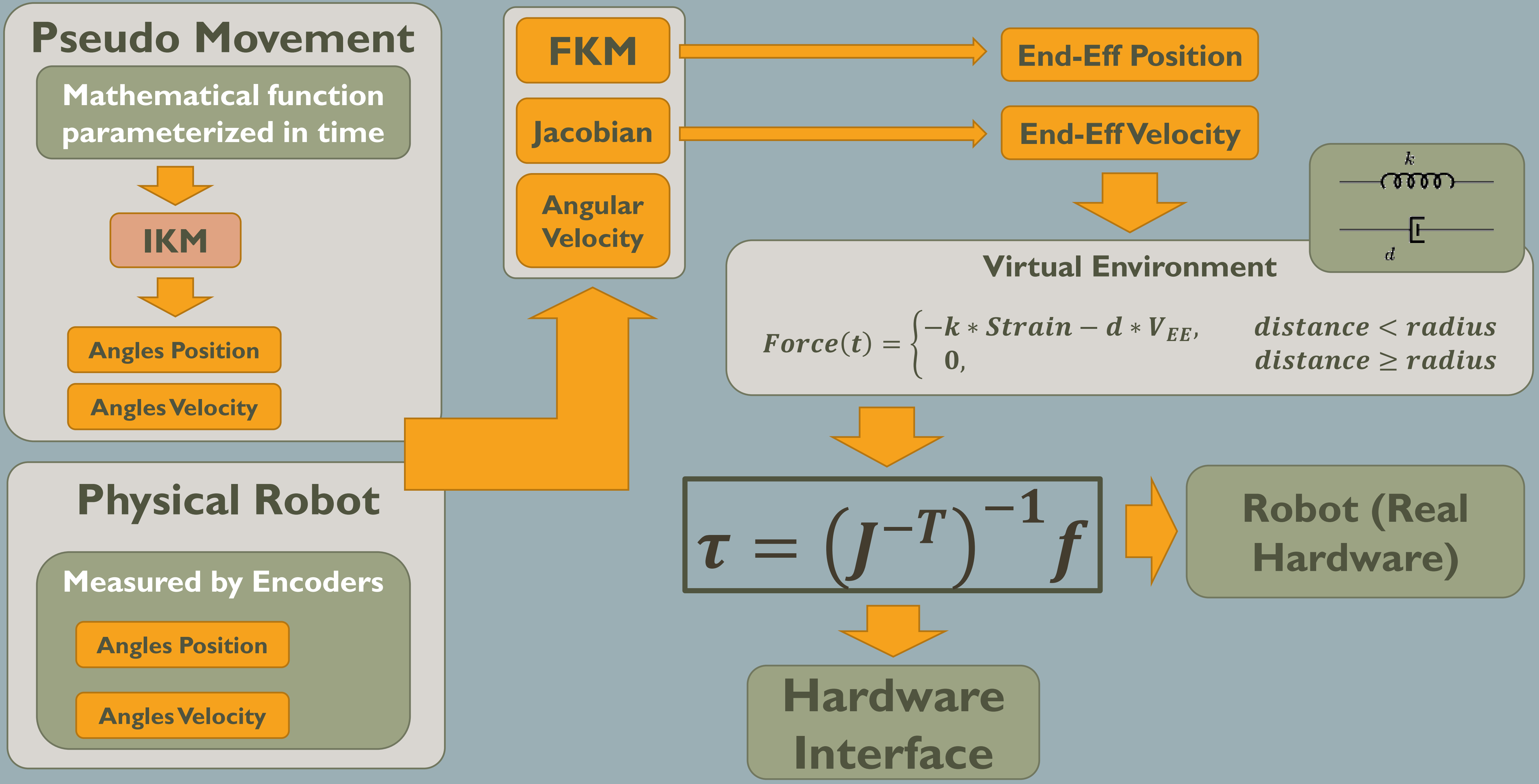
**Student:** Jesse Alves.

**Advisor:** Maciej Bednarczyk.



# INTRODUCTION

# The Haptic Loop is a Collision Model



## Pseudo Movement

Mathematical function parameterized in time

IKM

Angles Position

Angles Velocity

## Physical Robot

Measured by Encoders

Angles Position

Angles Velocity

FKM

Jacobian

Angular Velocity

End-Eff Position

End-Eff Velocity

Virtual Environment

$$Force(t) = \begin{cases} -k * Strain - d * V_{EE}, & distance < radius \\ 0, & distance \geq radius \end{cases}$$

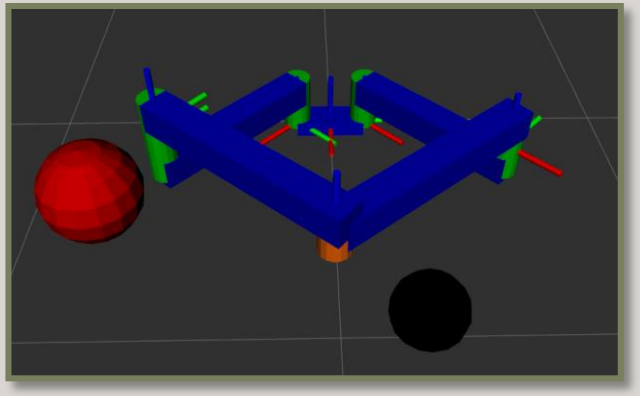
$$\boxed{\tau = \left(J^{-T}\right)^{-1} f}$$

Robot (Real Hardware)

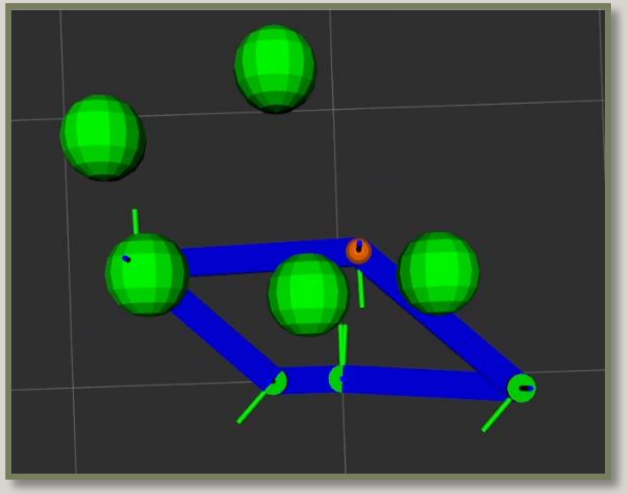
Hardware Interface

# The Model Evolution

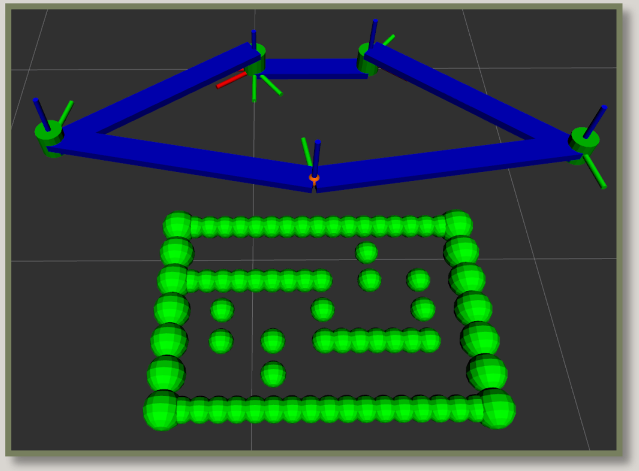
## First Model



## Second Model



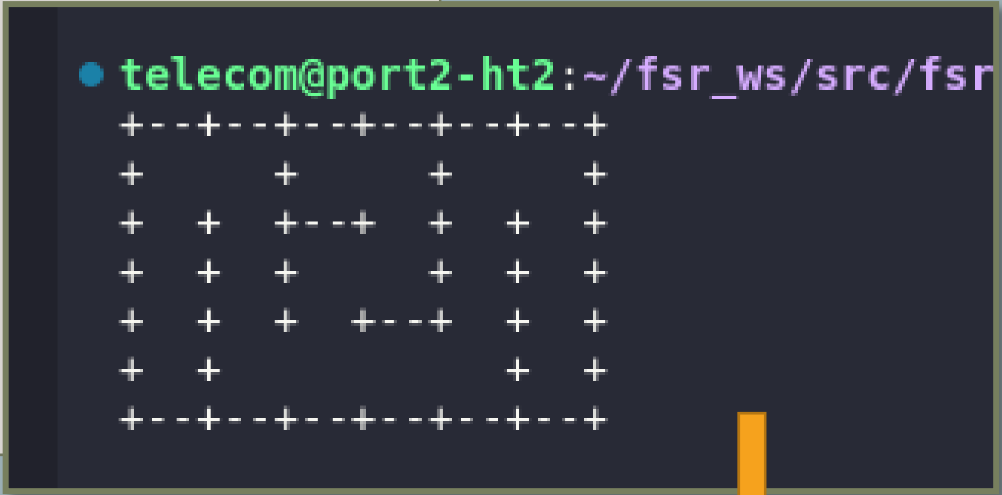
## Last Model



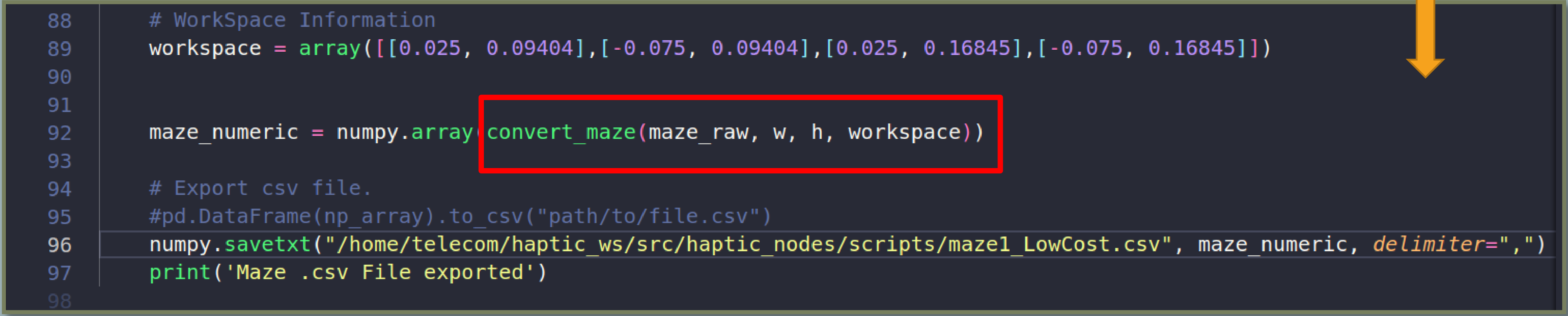
# MAZE GENERATION

# Maze Generation Code

- A new code to generate random **mazes in String** format.
- A new function to **convert** this String Maze:
  - **Center** of Spheres.
  - **Radius** of each sphere.
  - Inside of a **workspace given.**
- Export through a matrix in **.csv file**



```
telecom@port2-ht2:~/fsr_ws/src/fsr
+--+--+--+--+--+--+
+     +     +     +
+  +  +--+  +  +  +
+  +  +     +  +  +
+  +  +  +--+  +  +
+  +           +  +
+--+--+--+--+--+--+
```



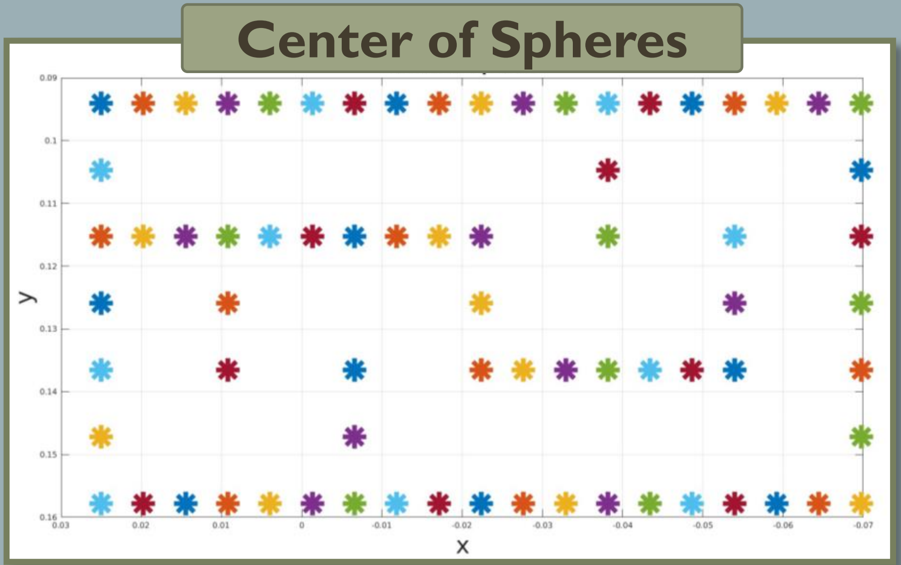
```
    # WorkSpace Information
    workspace = array([[0.025, 0.09404],[-0.075, 0.09404],[0.025, 0.16845],[-0.075, 0.16845]])


    maze_numeric = numpy.array(convert_maze(maze_raw, w, h, workspace))

    # Export csv file.
    #pd.DataFrame(np_array).to_csv("path/to/file.csv")
    numpy.savetxt("/home/telecom/haptic_ws/src/haptic_nodes/scripts/maze1_LowCost.csv", maze_numeric, delimiter=",")
    print('Maze .csv File exported')

```

# The .csv file

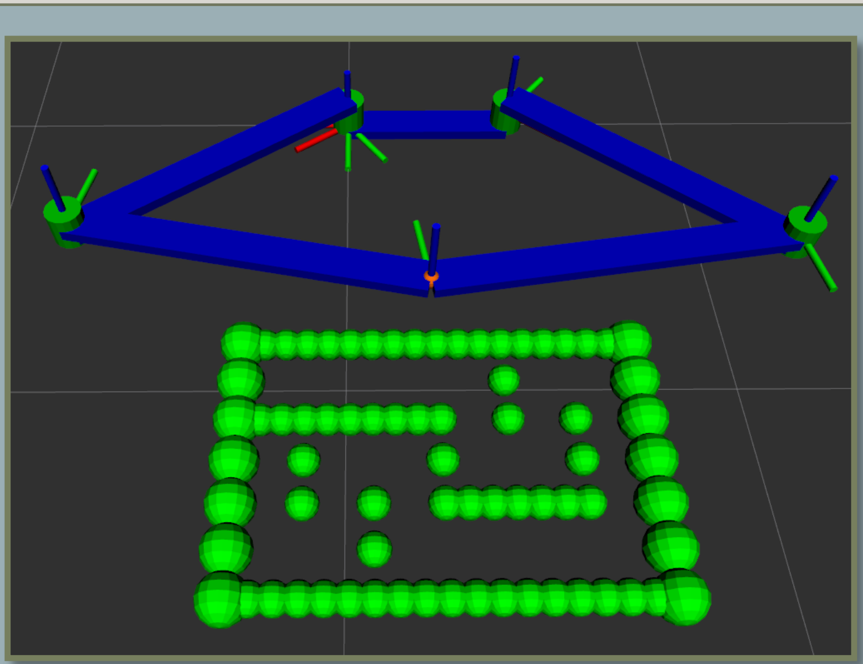
| Radius | X | Y |
|---|---|---|
| $radius_1$ | $x_1$ | $y_1$ |
| $radius_2$ | $x_2$ | $y_2$ |
| $\dots$ | $\dots$ | $\dots$ |
| $\dots$ | $\dots$ | $\dots$ |
| $radius_N$ | $x_N$ | $y_N$ |

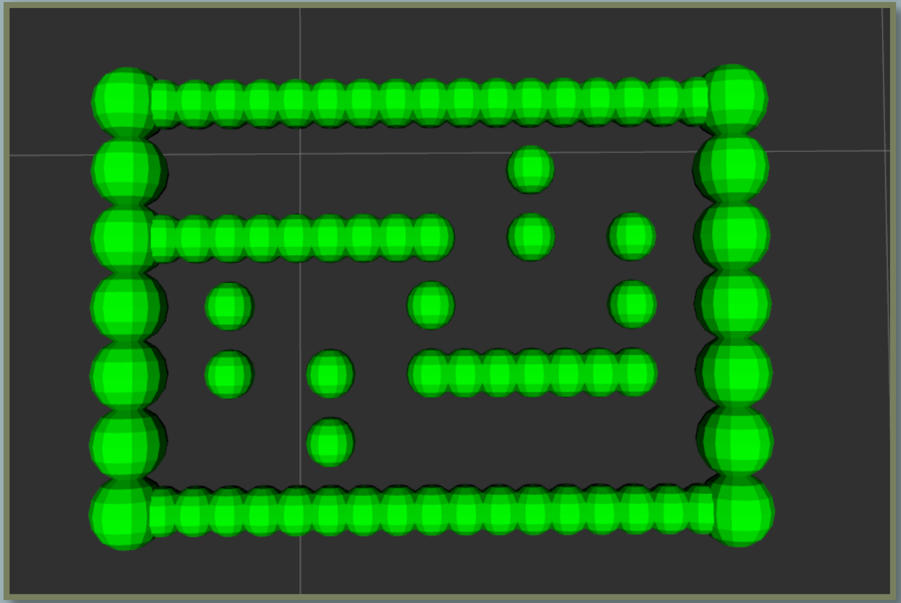
# Maze Generation Code

- This .csv file can be imported by:
  - Haptic Loop Node in ROS.
  - RViz (To be tested).
  - Visualization Part in Unity.



Center of Spheres



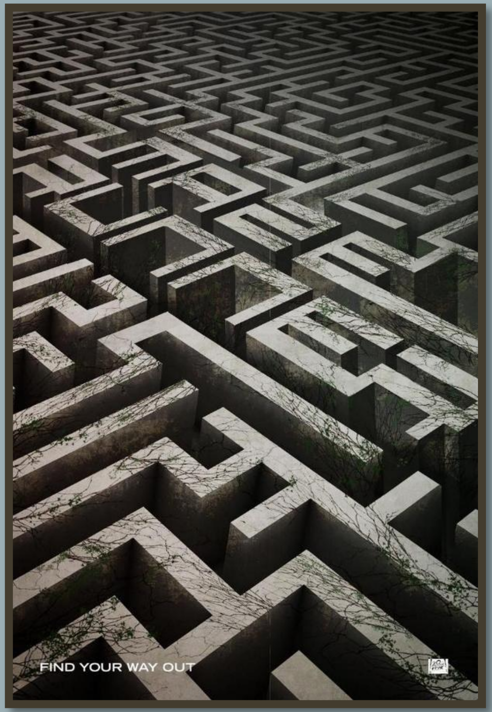


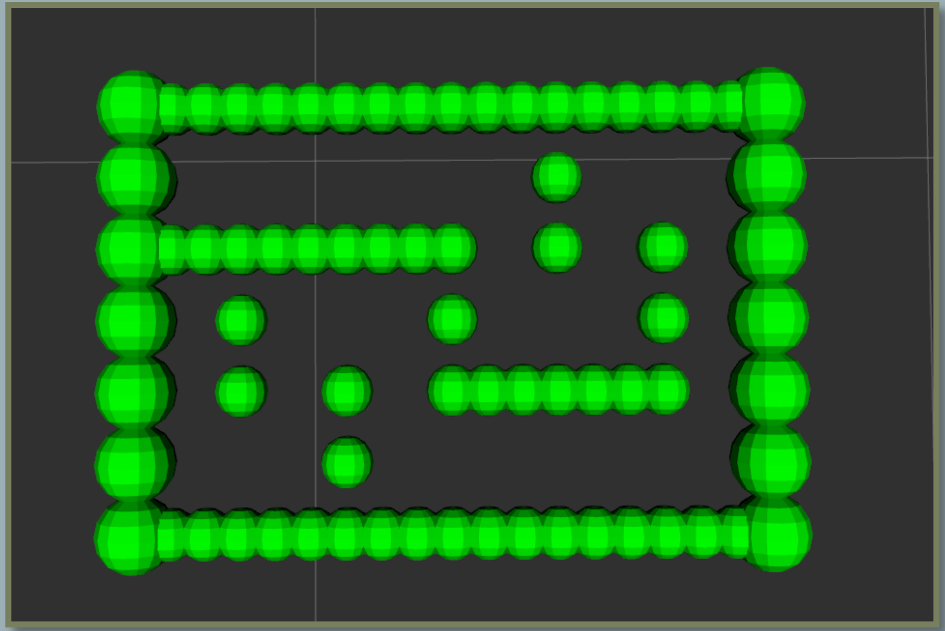
# Maze Generation Code

- It is a Collision Model.
- Complete Model = Visual Model + Collision Model (Hidden).
- Change the Spring and Damper Model.

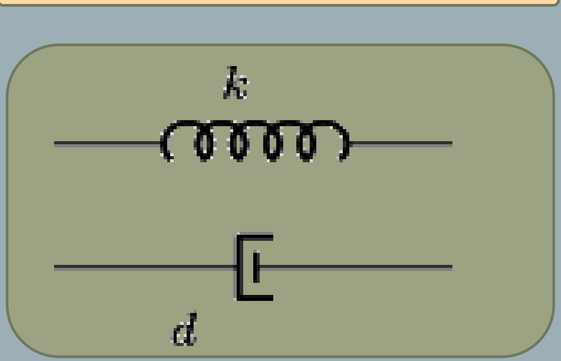
Collision Model
Visual Model

Font: Courtecuisse Hadrien Slides





Soft or Stiff Walls

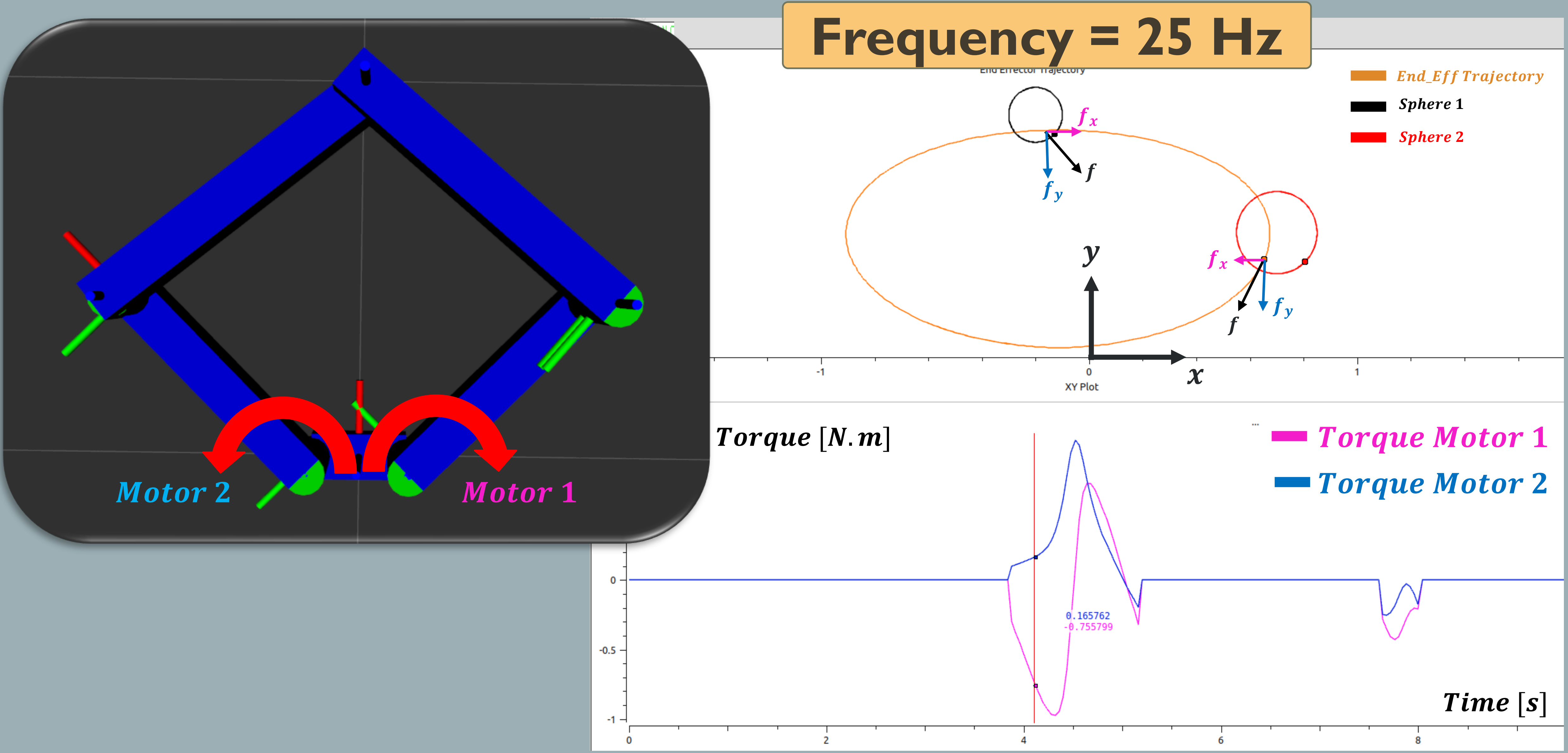


[Maze Simulation]

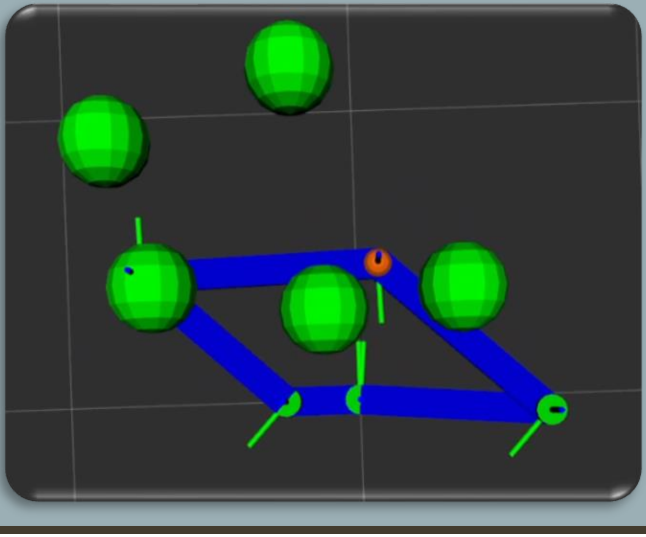
# OPERATION FREQUENCY

# First Simulation

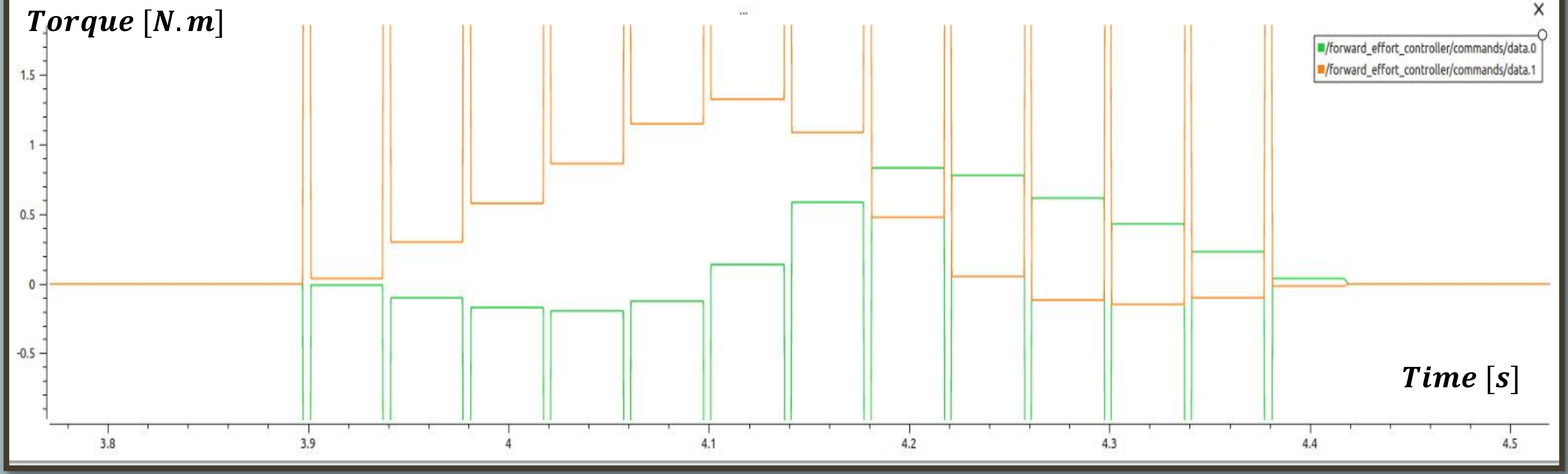
Frequency = 25 Hz



# Second Simulation



**Frequency = 500 Hz**



# Third and Last Simulation

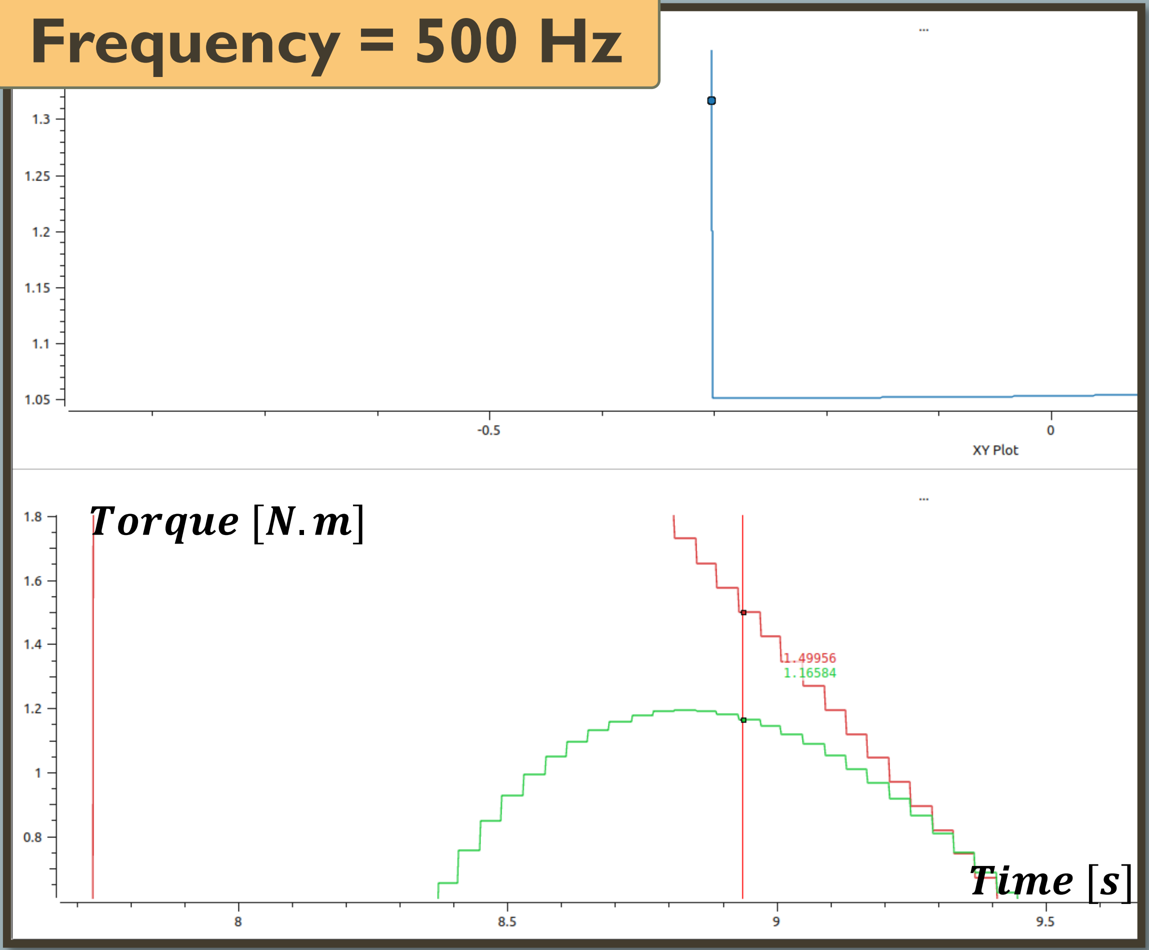
Kalman Filter ✗

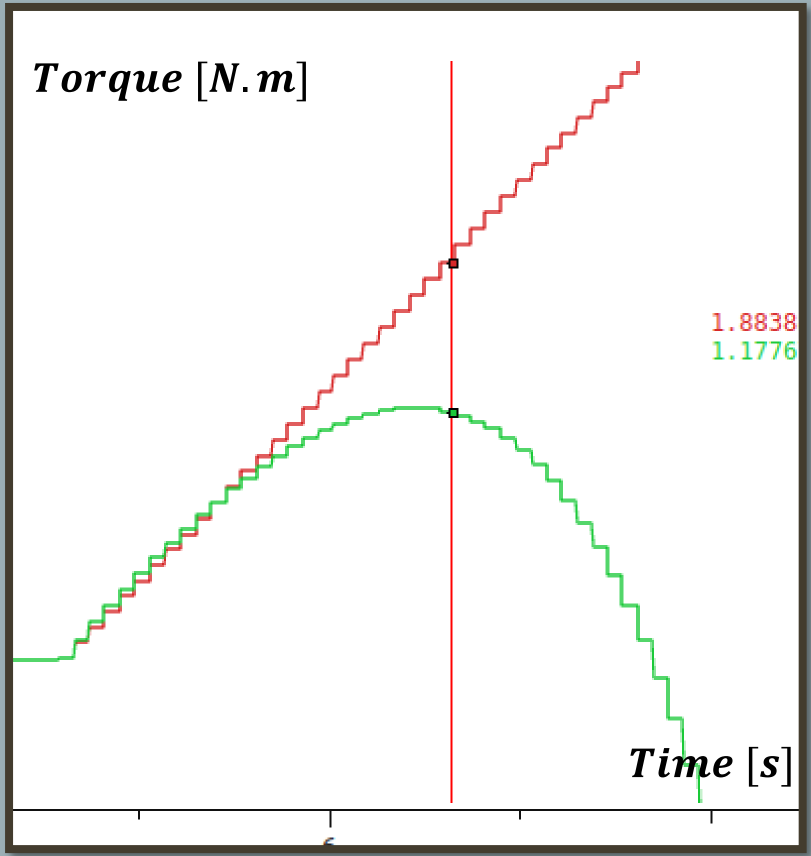
Moving Average Filter ✗

Jesse Filter ✓

[Torque Simulation]

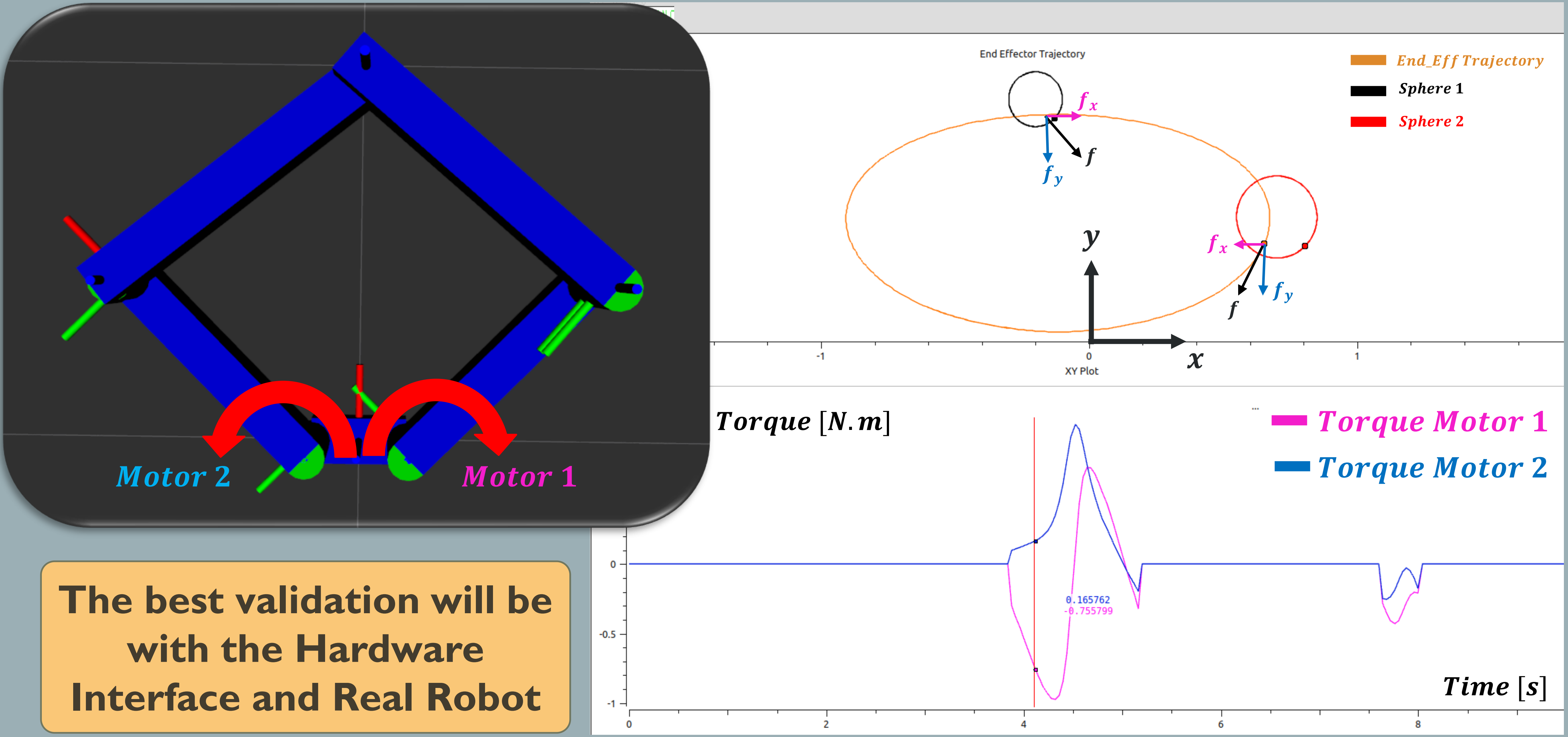
Frequency = 500 Hz





# TORQUE AND FORCE VALIDATION

# The Validation in the First Simulation



**The best validation will be with the Hardware Interface and Real Robot**

# A New Validation

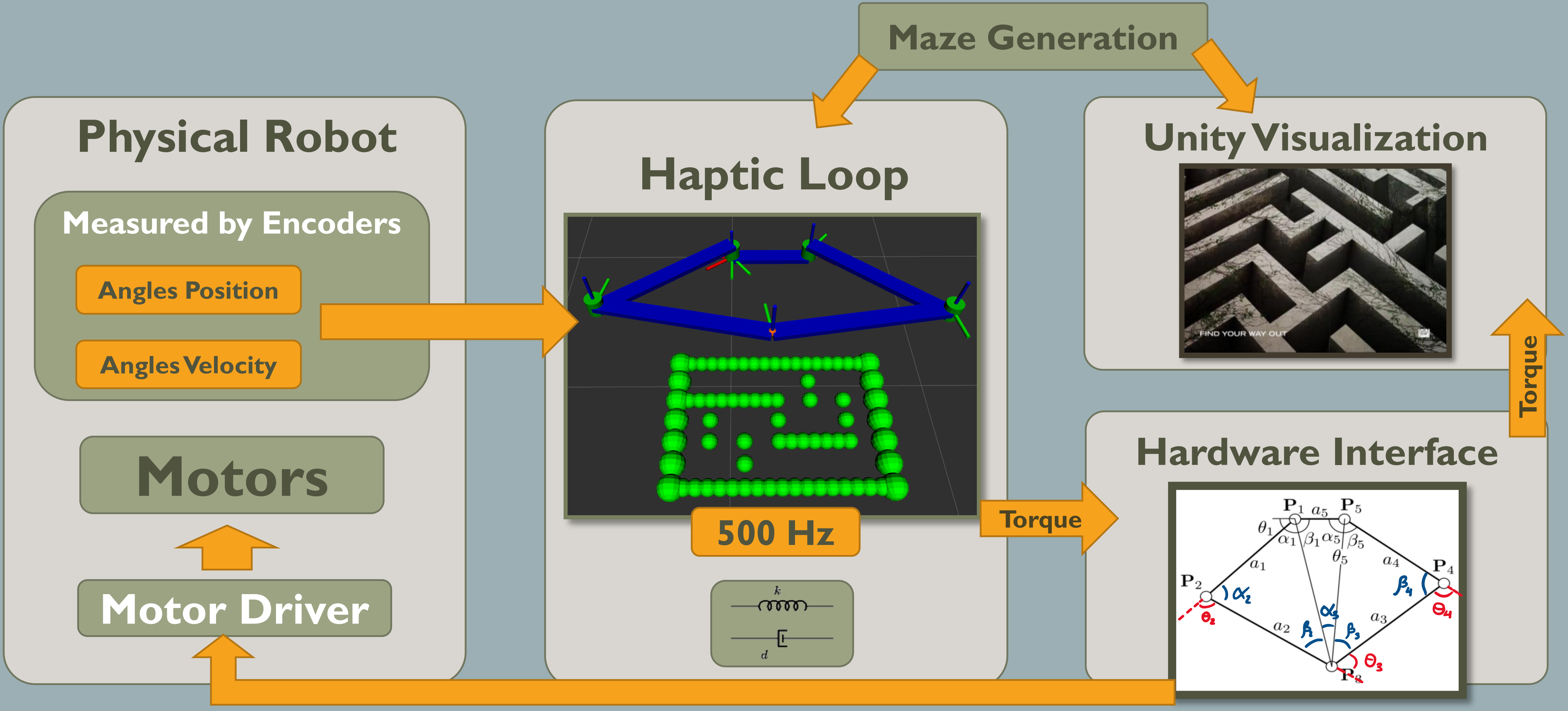
## Algorithm

1. Compute the Force Impact Collision
2. Compute the Torque using the Jacobian.
3. Compute Force using Jacobian and Torque.
4. Plot the Force Vector in RVIZ.

Simulation to Validate the Torque and Force.

# CONCLUSION

# The Paper of Haptic Loop



Maze Generation

Physical Robot

Measured by Encoders

Angles Position

Angles Velocity

Motors

Motor Driver

Haptic Loop

500 Hz

Torque

Unity Visualization

Hardware Interface

Torque

# Task 3 – Visualization Part

| Tasks | Done | Deliverable | Pending |
|---|---|---|---|
| Models in ROS | ✓ | | |
| Haptic Loop in ROS Validated | ✓ | ✓ | |
| Effort Controller | ✓ | ✓ | |
| Communication in ROS via topics | ✓ | ✓ | |
| Connection with Hardware Interface | | | ⚠ |
| Connection with different types of Visualization | | | ⚠ |

# THANKS FOR YOUR ATTENTION !