Project Link: http://hades.mech.northwestern.edu/index.php/Mobile_Manipulation_Capstone

# Introduction

In this project, I was able to simulate a youBot mobile manipulator ( a robot that has a mobile base with four mecanum wheels and a 5R robot arm) to move a block from one location to another. In order to do this, I planned out a trajectory for the end-effector of the robot. This dictated how the gripper of the youBot would move. I also performed odometry as the chassis of the robot moved, as well as perform feedback control to drive the robot. These things allowed me to make the robot move to the block, pick it up, carry it to the desired location, and put it down. The MATLAB functions generated csv files after running the scripts, which allowed me to simulate the results in CoppeliaSim.

During the project, 3 functions were created through three milestones

Milestone 1: NextState -

The function NextState is based on a simple first-order Euler step:

- new arm joint angles = (old arm joint angles) + (joint speeds) * Δt
- new wheel angles = (old wheel angles) + (wheel speeds) * Δt
- new chassis configuration is obtained from odometry, as described in Chapter 13.4 of the ME 449 textbook

Milestone 2: TrajectoryGenerator -

This function uses ScrewTrajectory function from the mr library to calculate the 8 reference trajectories for the gripper of the robot. The gripper has the following 8 trajectories.
1. A trajectory to move the gripper from its initial configuration to a "standoff" configuration a few cm above the block.
2. A trajectory to move the gripper down to the grasp position.
3. A trajectory to close the gripper.
4. A trajectory to move the gripper back up to the "standoff" configuration.
5. A trajectory to move the gripper to a "standoff" configuration above the final configuration.
6. A trajectory to move the gripper to the final configuration of the object.
7. A trajectory to open the gripper.
8. A trajectory to move the gripper back to the "standoff" configuration.

Milestone 3: FeedbackControl -

To calculate the control law FeedbackControl, we need the current actual end-effector configuration $X(q,\theta)$, a function of the chassis configuration q and the arm configuration θ. The values $(q,\theta)$ come directly from the simulation results (Milestone 1). In other words, assume perfect sensors.

The error twist Xerr that takes X to Xd in unit time is extracted from the 4x4 se(3) matrix:
[Xerr] = log(X^(-1) * Xd).
FeedbackControl also needs to maintain an estimate of the integral of the error, e.g., by adding Xerr * Δt to a running total at each timestep. The feedforward reference twist Vd that takes Xd to Xd,next in time Δt is extracted from [Vd] = (1/Δt) log(Xd^(-1)*Xd_next).

The output of FeedbackControl is the commanded end-effector twist V expressed in the end-effector frame {e}. To turn this into commanded wheel and arm joint speeds (u,thetadot), we use the pseudoinverse of the mobile manipulator Jacobian Je(θ),

# Operation

mr library Github: https://github.com/NxRLab/ModernRobotics

Before running the code, make sure that the mr library folder is downloaded and added to the path where you are running the script from. To run the code, open up a MATLAB terminal and run scripts to generate the csv files. Also make sure the mr library is added to the path before running the scripts

To see the best result, run `bestscript`

This generates:

- BestConfiguration.csv, which is a list of configurations for the robot
- bestfigure.pdf showing a plot of the six elements of Xerr as a function of time. The plot shows a convergence to zero.
- bestlogfile.txt showing how the program is called with the input
- BestXerr.csv showing all of the Xerr value of each iteration

To see the overshoot result, run `overshootscript`

This generates:

- OvershootConfiguration.csv, which is a list of configurations for the robot
- overshootfigure.pdf showing a plot of the six elements of Xerr as a function of time. The plot shows a convergence to zero.
- overshootlogfile.txt showing how the program is called with the input
- OvershootXerr.csv showing all of the Xerr value of each iteration

To see the newTask result, run `newTaskscript`

This generates:

- newTaskConfiguration.csv, which is a list of configurations for the robot
- newTaskfigure.pdf showing a plot of the six elements of Xerr as a function of time. The plot shows a convergence to zero.
- newTasklogfile.txt showing how the program is called with the input
- newTaskXerr.csv showing all of the Xerr value of each iteration

# Results

To see the animation of the results, open CoppeliaSim, and run Scene 6. Upload the generated csv files into the scene to watch what happens. In the animation, the robot should behave in such a way that it first drives up to the cube, picks it up, carries the cube to the desired destination, and puts it down. Corresponding videos of the animation can be found in the results folder.

# Error Plots

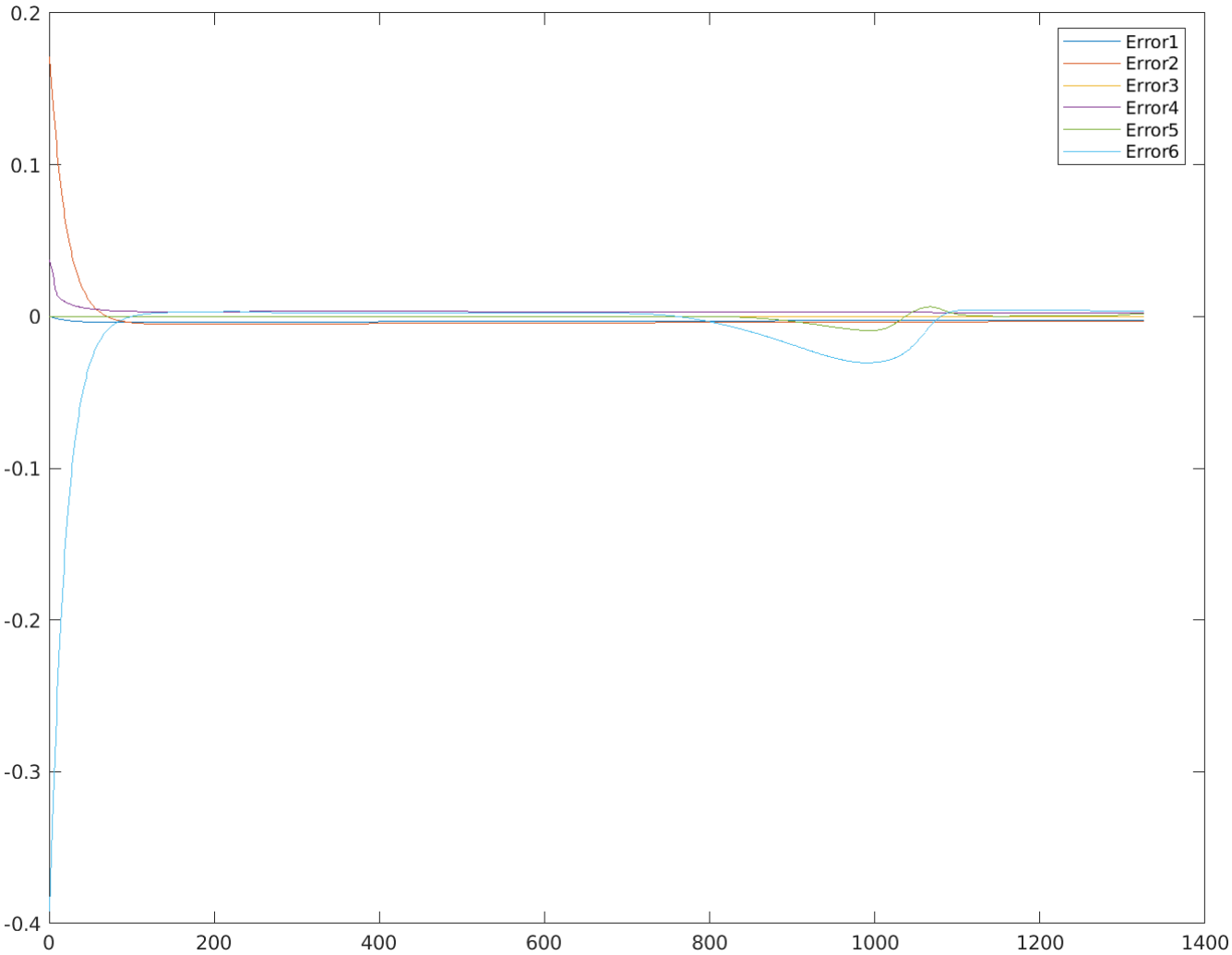


*Figure 1: Error Plot from bestscript.m*

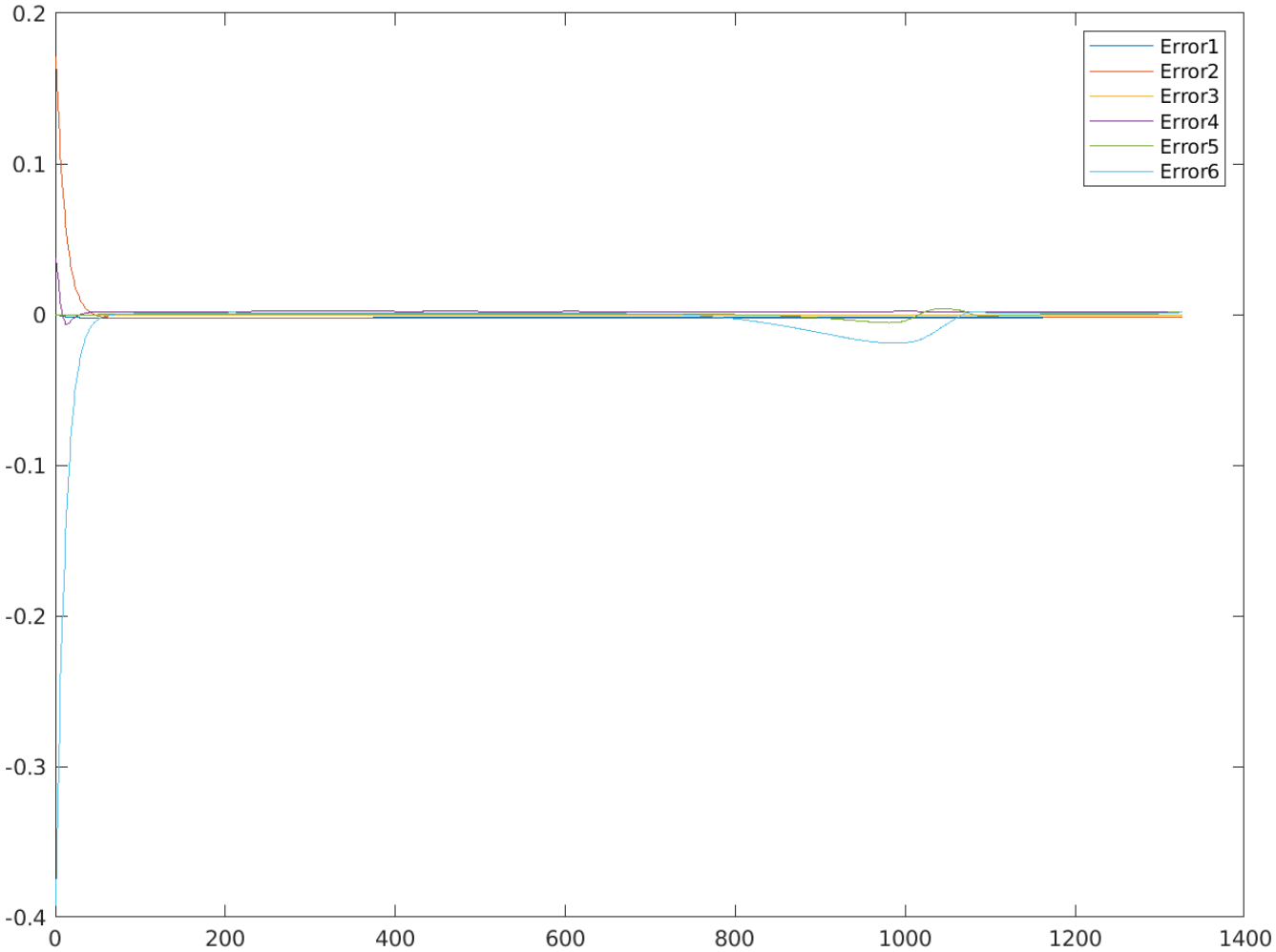


*Figure 2: Error plot from overshootscript.m*

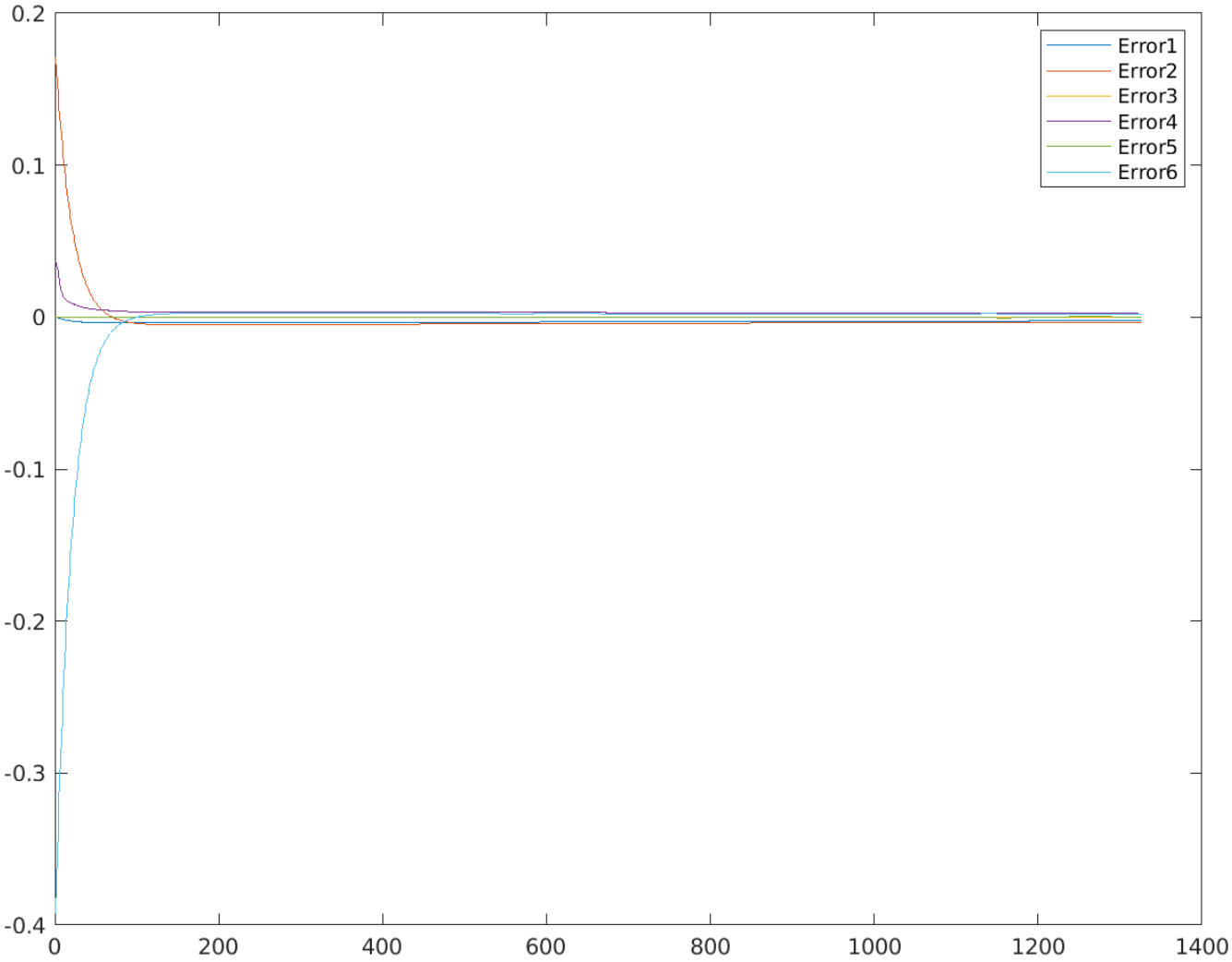


*Figure 3: Error plot from newTaskscript.m*