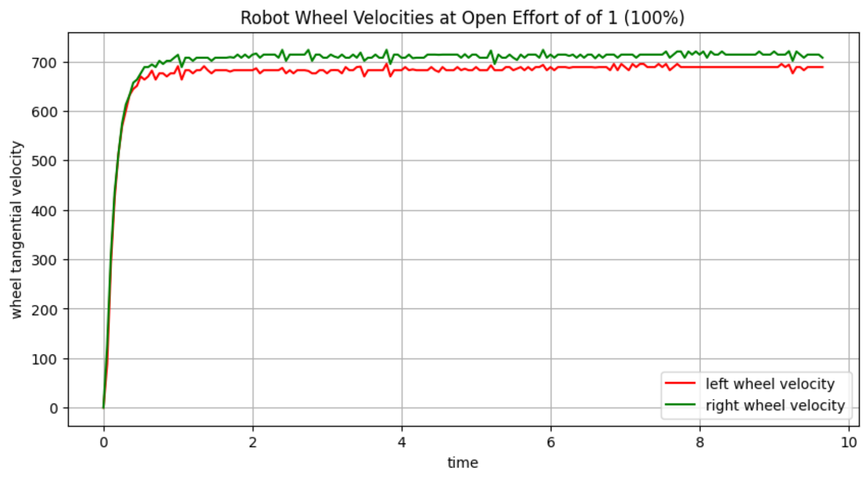
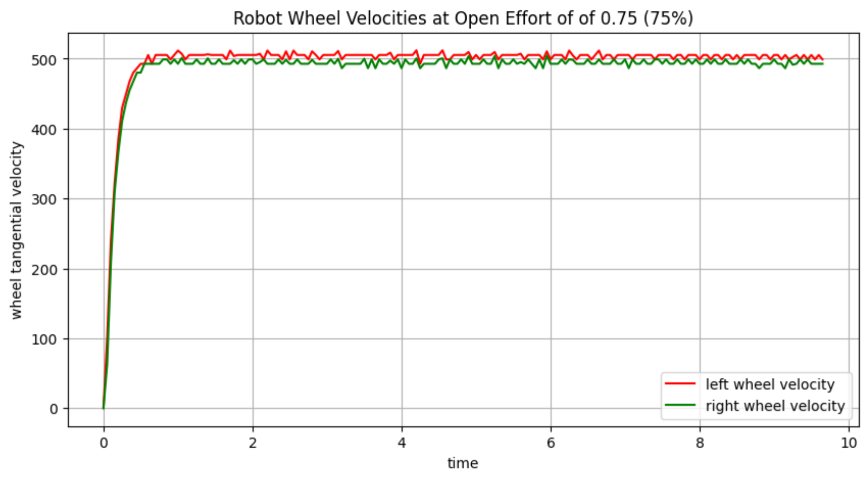
Aengus Kennedy
ME-0134 Robotics
Lab 1: Speed Controller
2025 06 02

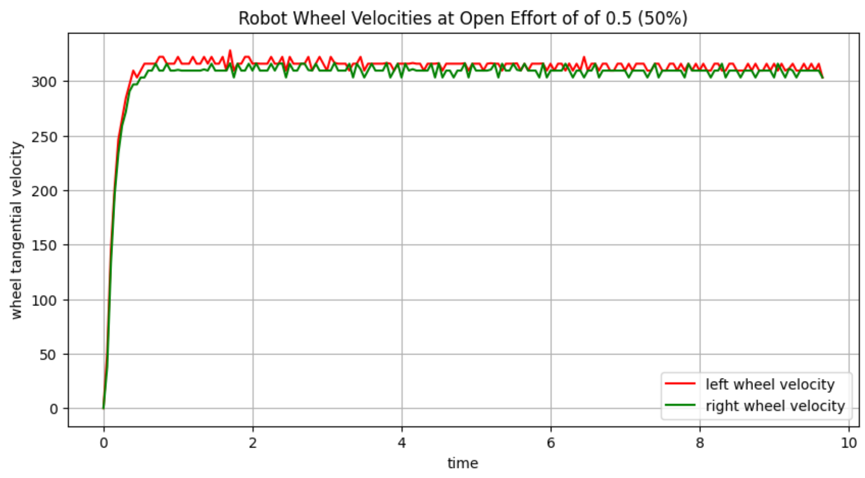
1: Wheel Velocities at Constant Effort Values, Suspended



$\mu_l = 685.8$ mm/s; $\sigma_l = 4.8\ mm/s$; $\mu_r = 712$ mm/s; $\sigma_r = 4.9\ mm/s$

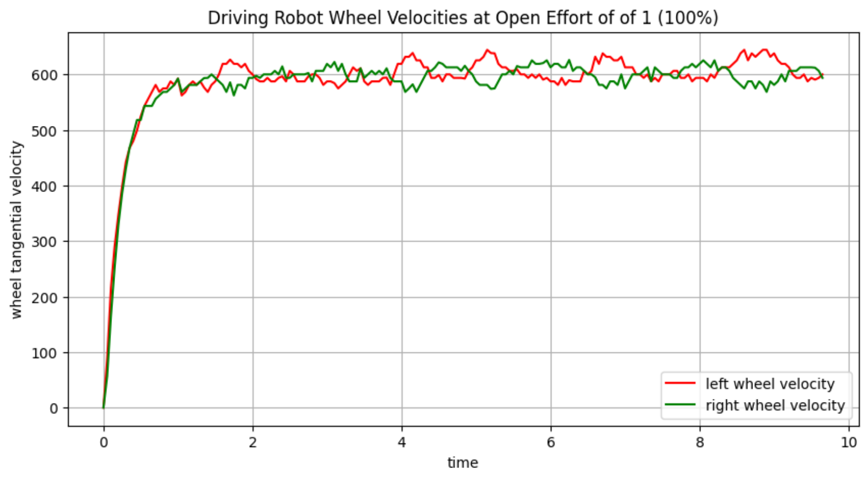


$\mu_l = 504.4$ mm/s; $\sigma_l = 3.4\ mm/s$; $\mu_r = 494.6$ mm/s; $\sigma_r = 3.9\ mm/s$

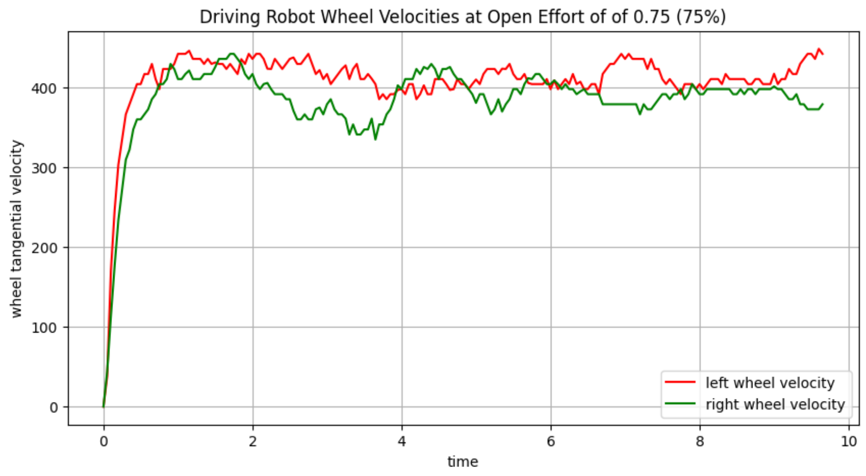


$\mu_l = 314.6$ mm/s; $\sigma_l = 3.9\ mm/s$; $\mu_r = 309.9$ mm/s; $\sigma_r = 3.8\ mm/s$

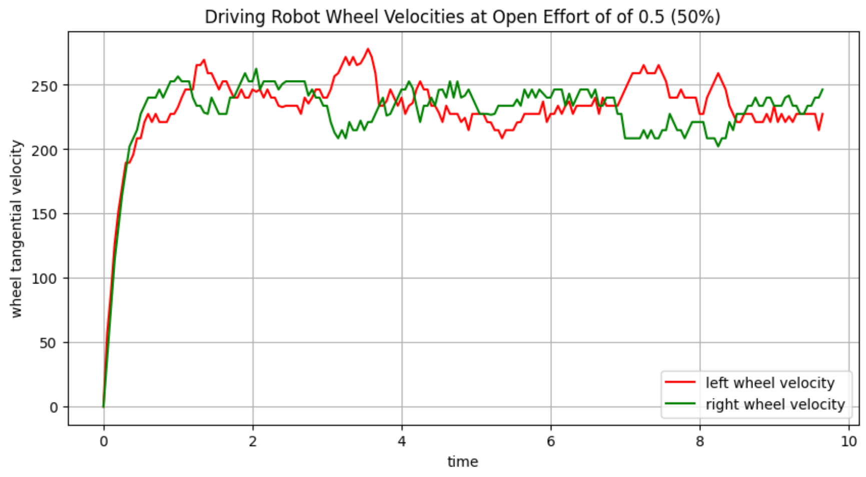
2: Wheel Velocities at Constant Effort Values, Driving



$\mu_l = 603.4$ mm/s; $\sigma_l = 17.8\ mm/s$; $\mu_r = 598.9$ mm/s; $\sigma_r = 14.6\ mm/s$



$\mu_l = 416.5$ mm/s; $\sigma_l = 15.2\ mm/s$; $\mu_r = 393.0$ mm/s; $\sigma_r = 21.3\ mm/s$



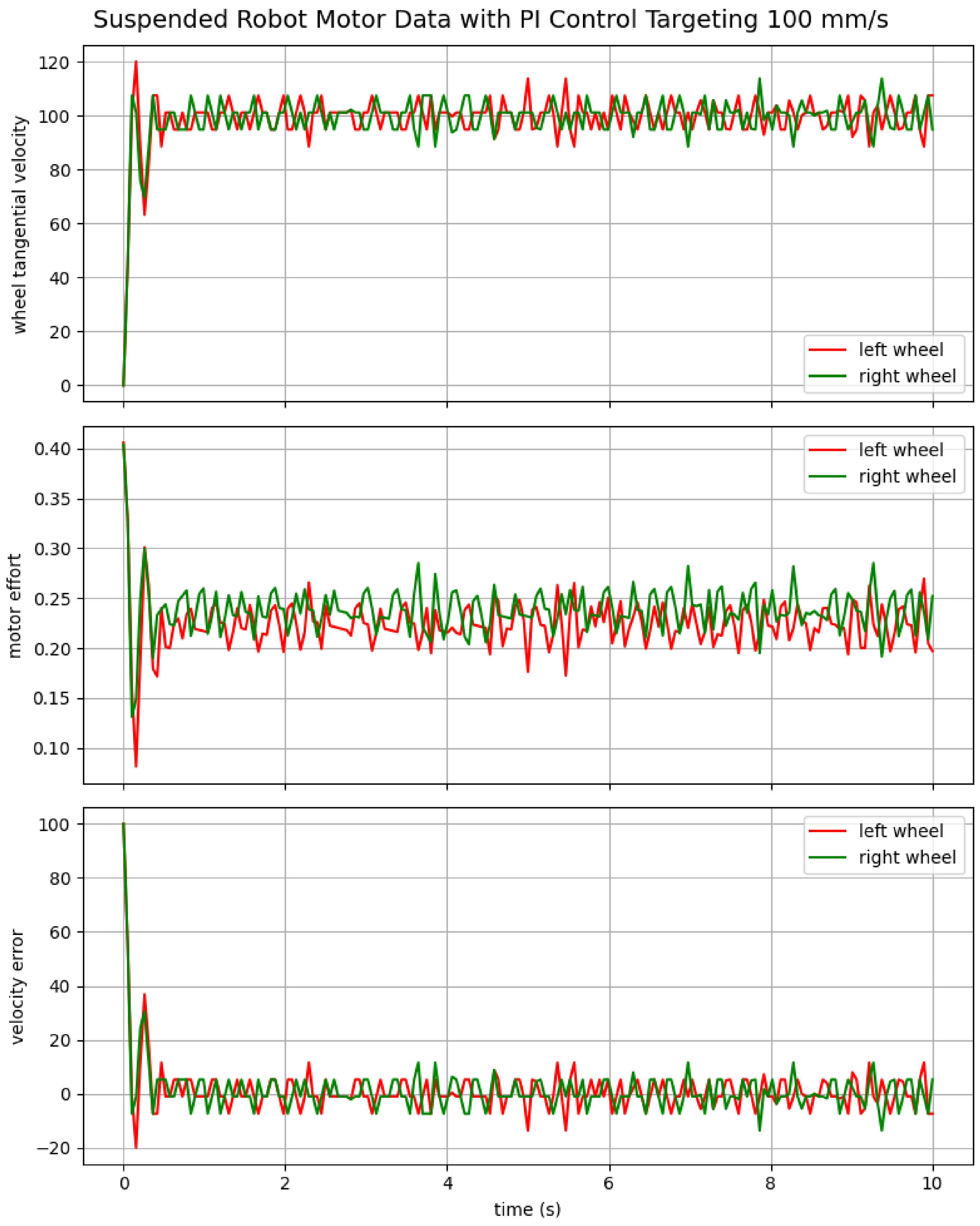
$\mu_l = 238.3$ mm/s; $\sigma_l = 14.4\ mm/s$; $\mu_r = 233.7$ mm/s; $\sigma_r = 13.8\ mm/s$

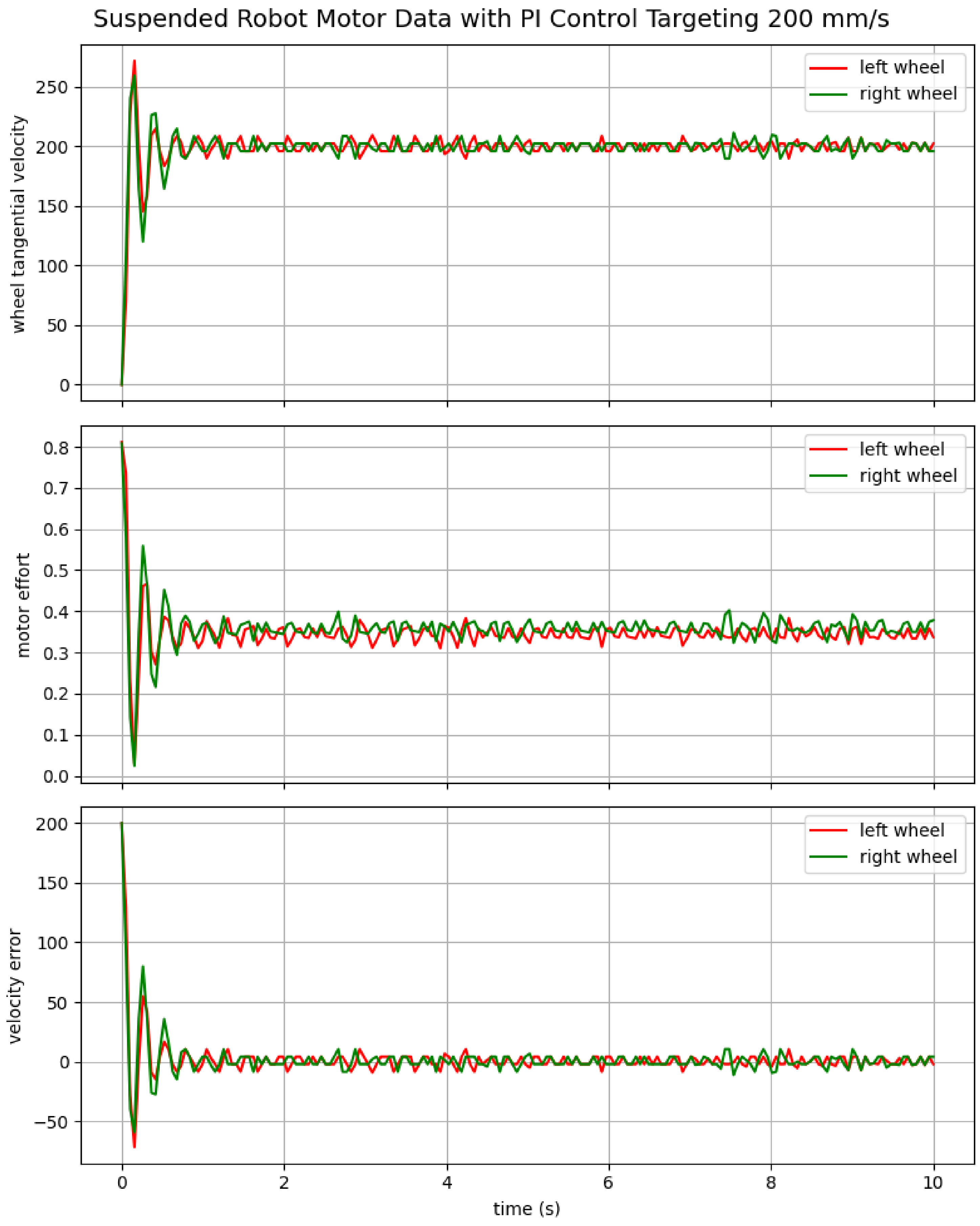
3: Tuned PID Controller Values

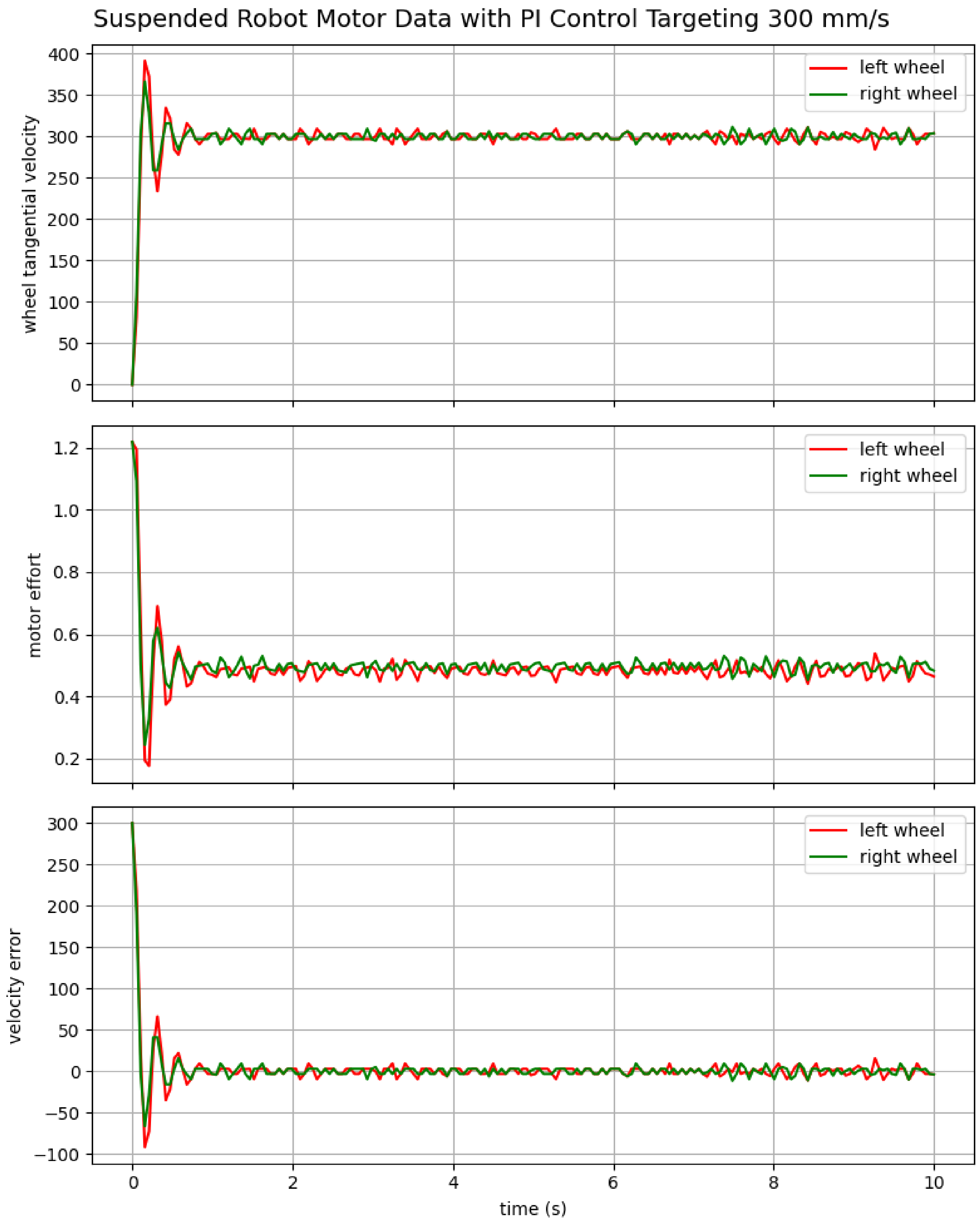
I tuned the proportional controller first, followed by the integral controller. Any nonzero value of the derivative term added to the oscillations in the motor's behavior, so I think a PI controller is optimal. My final values are:

```
Kp = 0.003
Ki = 0.02
Kd = 0
```

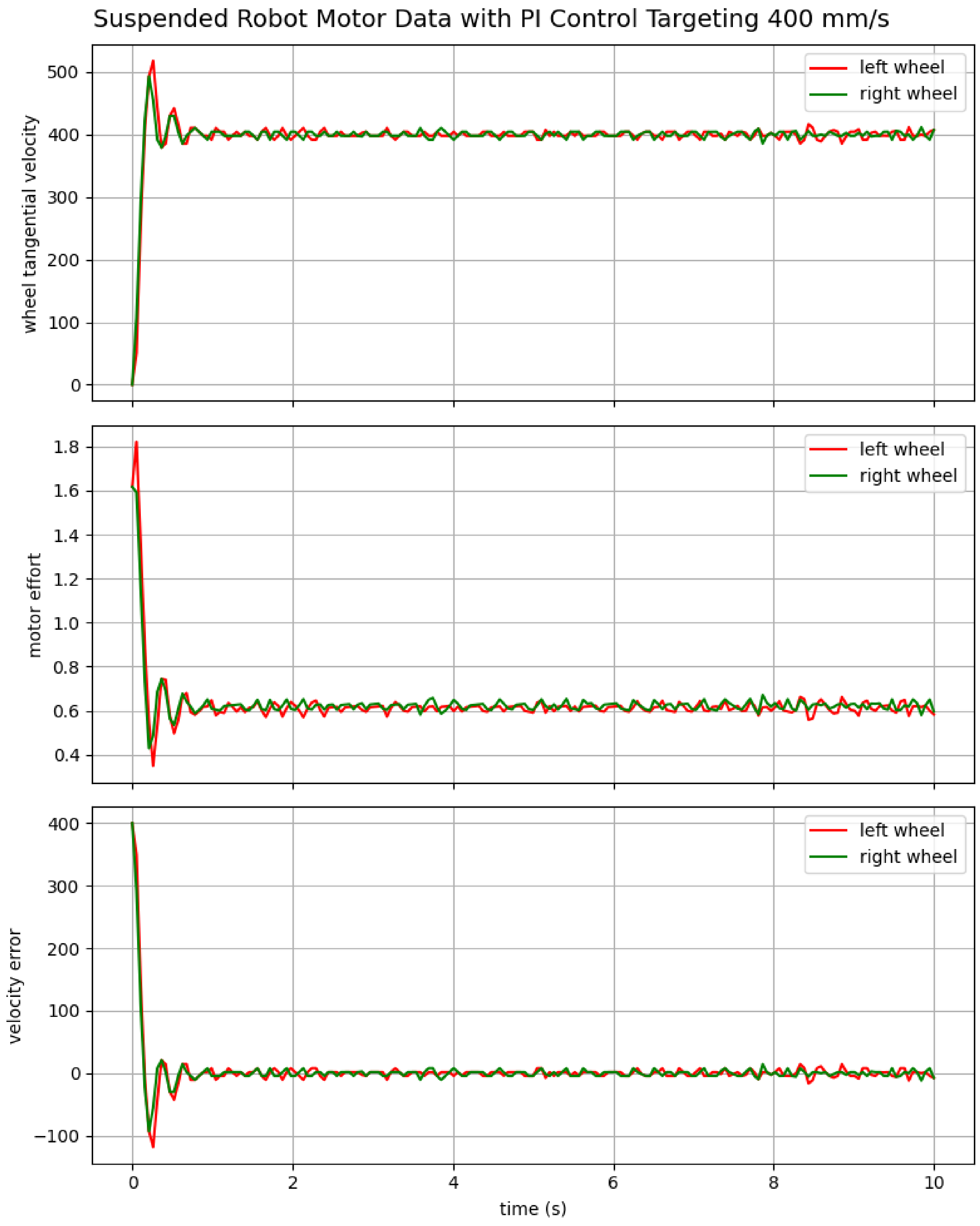
4: Motor Outputs Under PI Control, Suspended




Suspended Robot Motor Data with PI Control Targeting 200 mm/s
wheel tangential velocity
motor effort
velocity error
left wheel
right wheel
time (s)
0
2
4
6
8
10



Suspended Robot Motor Data with PI Control Targeting 300 mm/s
wheel tangential velocity
motor effort
velocity error
left wheel
right wheel
time (s)


Suspended Robot Motor Data with PI Control Targeting 400 mm/s
wheel tangential velocity
motor effort
velocity error
time (s)
left wheel
right wheel
0
100
200
300
400
500
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
−100
0
2
4
6
8
10

5: Distances Driven Under PI Control

time driven	velocity target	distance traveled
10 s	100 mm/s	98 cm
10 s	200 mm/s	198 cm
10 s	300 mm/s	298 cm
10 s	400 mm/s	392 cm

6: Analysis

To improve my controller further, the PID coefficients should be refined to address three significant flaws in the controller's current performance.

First, the motors exhibit velocity overshoot, which appears extreme in the graphs in part 4, but isn't very noticeable watching the robot's wheels run. A properly tuned derivative term should reduce overshoot and oscillations like this. In my current controller, any of the nonzero derivative terms that I tried systematically, across orders of magnitude, positive or negative, increased oscillations rather than decreasing them, but a more formal tuning process might be able to find an optimal derivative term that reduces oscillations.

Second, in part 5, the robot consistently drove 2 centimeters shorter than it should have driven given its target velocity and duration. This offset decreased as I increased the integral term up to a point, until any further increases in the integral term introduced oscillations. A well tuned combination of integral and derivative terms might be able to reduce this error without introducing oscillations.

Third, at target velocities higher than 350 mm/s, the motors' rise is unsteady and unpredictable, and the robot usually ends up turning unintentionally relative to its starting orientation. I'm confident that this problem could be fixed by a derivative term that prevents the motors from speeding up too quickly, but that derivative term would need to be found by a more formal process like the Ziegler-Nichols method.