Control of a differential-wheeled robot

Ondřej Staněk

2013-07-17

www.ostan.cz

SRH Hochschule Heidelberg, Master IT, Advanced Control Engineering project

Abstract

This project for the Advanced Control Engineering class investigates a movement control system of a differential-wheeled robot. A model of the robot drive is designed in Matlab Simulink, and a control system is analyzed, implemented a tuned. The work is based on author's previous experience in robot control. The control system presented in this paper was already tested on two real robotic platforms. This time, the author investigates the theoretical background behind his particular control system, that was originally developed without any prior skill in Control Engineering theory.

Part I
Analysis

1 Differential Wheeled Robots

Differential drive is very popular in mobile robotics. Differential wheeled robots usually feature two individually propelled wheels[1] [2]. By adjusting the power applied to motors, the robot can be operated to go forward, rotate in place or perform movement on any arbitrary curve in plane. Such wheelframe is very flexible, yet still easy to control, so it is often prioritized over other steering systems, such as Ackerman drive. Thanks to high movement flexibility, the planning of robot trajectory is much easier compared to car-like steering system. If the robot is intended for operation on flat surface (indoor conditions), the differential drive is definitely good way to go.

[1] and one or more omni-directional supportive wheels that only ensure stability of the wheelframe

2 Goals and Objectives of the Control System

The control system ensures accurate movement of the robot. It uses feedback from wheel encoders that measure wheel rotation. Information from encoders is used to control speed and position of the robot. The control system calculates power applied to motor terminals to meet the speed and position requirements. The control system should compensate for any outer disturbances, caused by surface and tire friction, external forces to the robot and collisions with smaller objects that might temporary block robot's wheels.

Part II
Design

3 Robot Model

A model of differential drive is designed in Matlab Simulink. The model takes into account:

- outer disturbances caused by external forces (f.e. wind, collisions with smaller objects, blocking of a wheel..)
- tire slipping [2] (driving on wet surface/snow/ice/sand)
- slightly different transfer functions for left and right motor+gearbox+wheel subsystem is supported
- propagation delay of wheel encoders

4 Movement Controller

The original controller was designed to support various type of movements (straight movement, rotation in place, circular movement..). However, in this paper, the function of the controller was simplified as to only support straight movement and speed control. To keep things simple, the position control is not comprehended in the simulation. The position control would introduce another PID regulator that governs the whole speed/balance control system presented in this paper. That means the complete control system can be build in a modular manner. But we will only simulate the speed and wheel balance control for clarity.

[2] the controller cannot compensate for this because the only feedback is the information about rotation of the wheels, not the actual position of the robot in its environment

Part III
Implementation

5 Speed and Balance Controllers

The movement control system is designed as two chained PID controllers. There is a Speed PID Controller, which calculate the PWM[3] duty for motors according to the desired and actual speed of the robot. The calculated PWM duty is then proportionally distributed to left and right motor, which is done by the Wheel Balance PID Controller. This controller adjust the ratio of left and right motor power, so that a constant trajectory ratio between left and right wheel is maintained. Obviously, for a straight movement, the desired trajectory ratio is 1:1 (trajectory of left wheel : trajectory of right wheel). For a rotation movement, the ratio would be -1:1. And finally, any arbitrary ratio of wheel trajectories yields circular movement of the robot.

6 Robot Speed Feedback

The robot speed is calculated as an average of the speed of its two wheels. The speed of wheel is the derivative of the trajectory measured by the wheel encoder. It is not advised to use the derivative block in Simulink models as it may introduce extreme numerical noise, so instead of the derivative block we use the transfer function $G_d = \frac{s}{s+1}$.

7 Transfer Function of the Robot's Propeller System

The behavior of the motor, gearbox and wheel is modeled by a two-pole transfer function:

$$G_m(s) = \frac{1}{(s - p_1)(s - p_2)}$$

The poles of the left motor are chosen to be $p_1 = -2$, $p_2 = -3$ and for the right motor we assume a slight change in the transfer function due to inaccurate fabrication process, so that the second pole is slightly shifted to $p_2 = -3.1$. This simulates the variance in motor/gearbox characteristics. However, for the calculation of the PID control parameters we assume that the poles are at their expected value $p_1 = -2$, $p_2 = -3$.

[3] pulse width modulation

Part IV
PID Contrller Tuning

The PID regulator is commonly used in industry because it can be easily implemented and works reasonably well for many processes. However, choosing the proper values of the P, I and D parameters is often a challenge. There are several methods how to do this manually, using trial-and-error approach. The tuning process can be supported with some rule of the thumb methods[4].

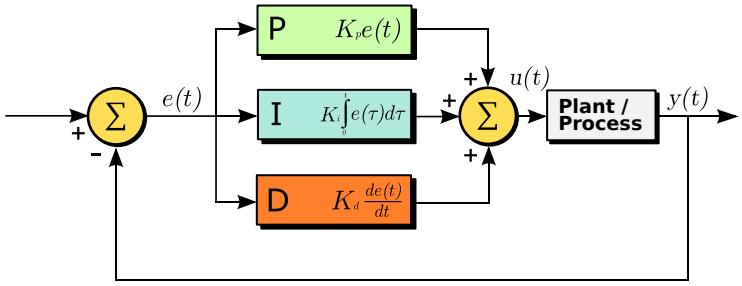

Figure 1: Block diagram of a PID regulator

8 Calculation of the PID Parameters for Speed Control

For our model, however, we assume that we already know the transfer function $G_m(s)$ of the robot propeller system. In reality, the transfer function has to be obtained from the real robot by performing a set of tests and measurements. I did not do this, I just chose appropriate[5] transfer function $G_m = \frac{1}{(s+2)(s+3)}$.

The PID control block can be expressed by its transfer function [1]:

$$G_c(s) = K_p + K_i \frac{1}{s} + K_d s = \frac{K_p s + K_i + K_d s^2}{s} = K \frac{(s+s_1)(s+s_2)}{s} \tag{1}$$

where $s \in \mathbb{C}$ and $K_p, K_i, K_d \in \mathbb{R}$ are the real parameters of the PID controller. These equations hold for the PID parameters:

$$K_p = K(s_1 + s_2) \tag{2}$$

$$K_i = K s_1 s_2 \tag{3}$$

[4] Ziegler–Nichols method is the popular one
[5] this two-pole transfer function is recognized as a motor transfer function

$$K_d = K \tag{4}$$

This can be easily verified substituting (2), (3), (4) to the equation (1).

For the closed-loop system we receive the transfer function[6]:

$$Y(s) = \frac{G_c G_m G_d}{1 + G_c G_m G_d} R(S) \tag{5}$$

Where G_d should be the derivative of the wheel trajectory, however, we chose $G_d = \frac{s}{s+1}$ to avoid numerical noise during simulation.

Once we have the motor transfer function $G_m = \frac{1}{(s-p_1)(s-p_2)}$, we choose the zeros of the PID transfer function to compensate for the poles of the motor transfer function G_m, so that $s_1 = -p_1 = 2$ and $s_2 = -p_2 = 3$:

$$G_c G_m = K \frac{(s+s_1)(s+s_2)}{s} \frac{1}{(s-p_1)(s-p_2)} = K \frac{(s-p_1)(s-p_2)}{s(s-p_1)(s-p_2)} = K \frac{1}{s}$$

Then we substitute $G_c G_m = K \frac{1}{s}$ to the transfer function of the system (5):

$$Y(s) = \frac{G_c G_m G_d}{1 + G_c G_m G_d} R(S) = \frac{K \frac{1}{s} G_d}{1 + K \frac{1}{s} G_d} R(s) = \frac{K G_d}{s + K G_d} R(s)$$

From this equation we can see that K is a free parameter that can be adjusted to move the pole of the complete feedback system on the real axis. By increasing parameter K, we move the pole away from the zero origin, making the system response faster.

Finally, we calculate the PID parameters given by equations (2), (3), (4):

$$K_p = K(s_1 + s_2) = K(-p_1 + (-p_2)) = K(2+3) = 5K$$

$$K_i = K s_1 s_2 = K(-p_1)(-p_2) = K \cdot 2 \cdot 3 = 6K$$

$$K_d = K$$

9 Calculation of the PID Parameters for Balance Controller

The speed and balance PID controller are dependent on each other and therefore it is not easy to calculate the PID parameters. However, we will make similar simplification as in previous section; for the balance regulator we assume that the power signal (output of the speed PID regulator) is constant. Then, the

[6] for simplification we assume that the speed PID controller controls only one wheel, and thus the balance PID regulator doesn't have to be considered

Balance PID regulator can be seen as a controller for the wheel path[7] and the transfer function of the system is:

$$Y(s) = \frac{G_c G_m}{1 + G_c G_m} R(S) = \frac{K\frac{1}{s}}{1 + K\frac{1}{s}} R(s) = \frac{K}{s + K} R(s)$$

This is almost the same as the transfer function for speed regulation loop (5), except there the G_d derivation term is missing. We can choose the Balance PID parameters exactly the same as for the speed PID regulator. The gain coefficient K has to be tuned separately for the balance regulator, of course.

Part V
Simulation

10 Simulink Model

The simulation is done in Matlab Simulink. Block diagram of the simulation settings is shown in Figure 2 on the following page. The "proportional power divider" is a custom made block that balances the power calculated by "Robot speed PID Controller" proportionaly to left and right motor according to the output of the "Wheel balance PID Controller". When the "balance adjustment" signal is zero, 50% of the power signal is sent to left motor and 50% to the right motor.

The "manual wheel blocking" block simulates a condition when the left wheel is blocked (due to external forces that effectively blocks the left wheel from rotation) for a few seconds. The effect of this heavy intervention to the system is then evaluated in the following sections.

The noise level of "disturbances" and "wheel slipping" blocks was set to `DisturbanceNoise = 0.5;` `WheelSlippingNoise = 0.1;` in the simulation.

11 Speed Control Simulation

The profile of the desired speed was chosen to simulate smooth take-off and smooth breaking of the robot. This is a good practice in robot control that prevents tire slipping and propeller system degradation. The speed setpoint is plotted in yellow and the actual robot speed is the pink line. There was noise added to the measurement to simulate real operation conditions. In Figure 3 on page 8 the gain of the PID controller was low ($K = 100$). The performance of the controller is not bad, but we can go even better if we increase the gain up to $K = 500$. Figure 4 shows very accurate speed control.

[7] again, I made another big simplification here

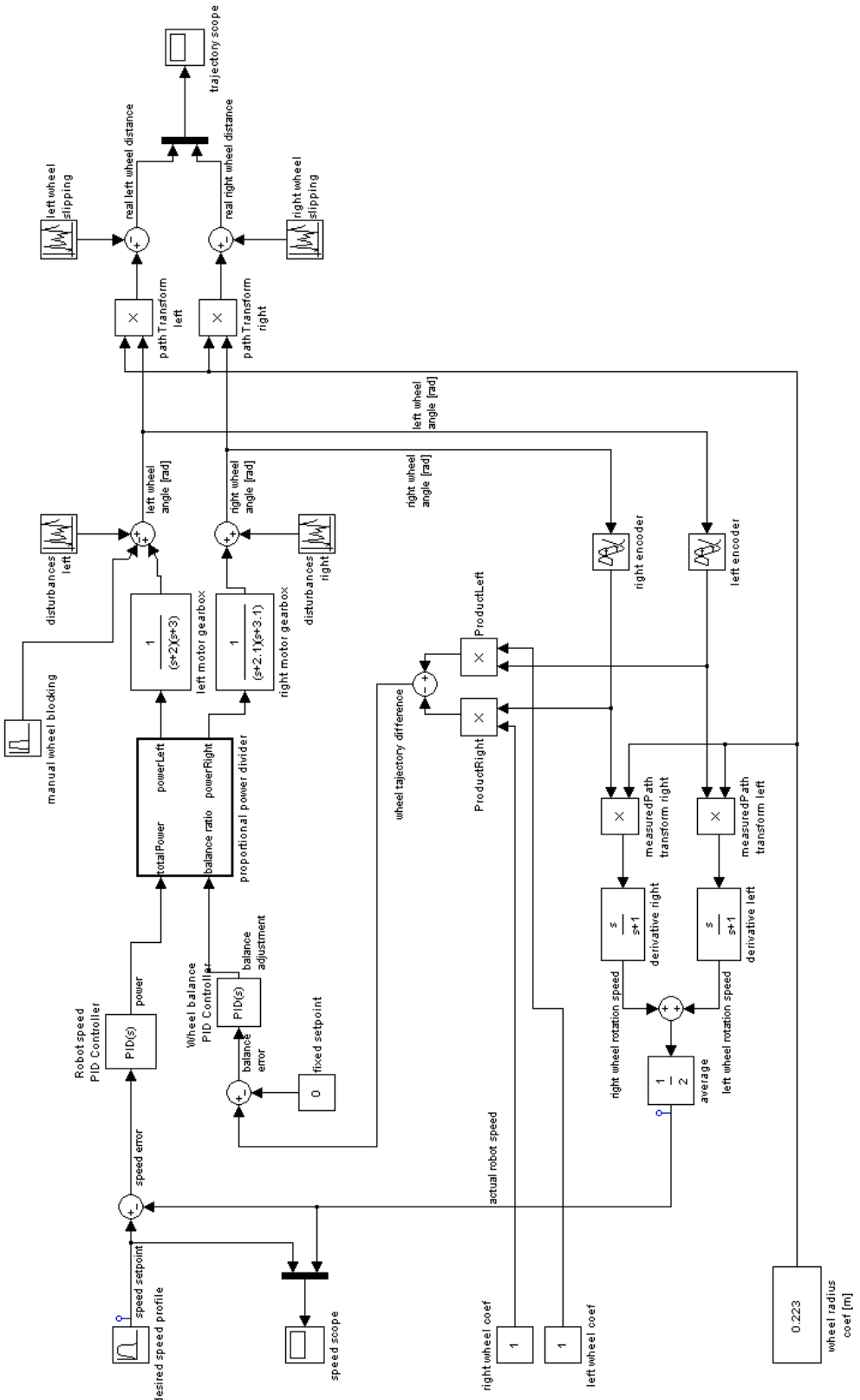

Figure 2: Simulink model of the complete control system

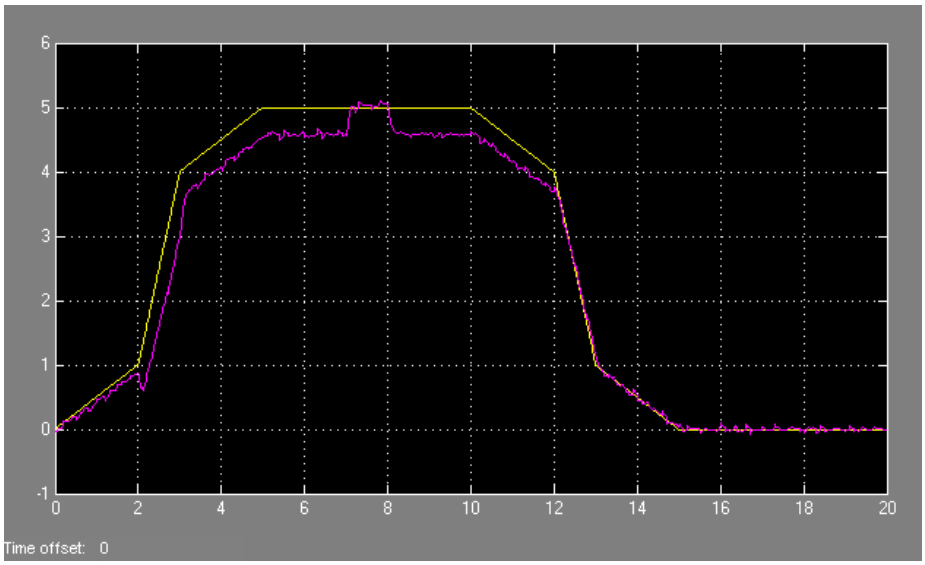

Figure 3: K_speed = 100 (suboptimal speed control)

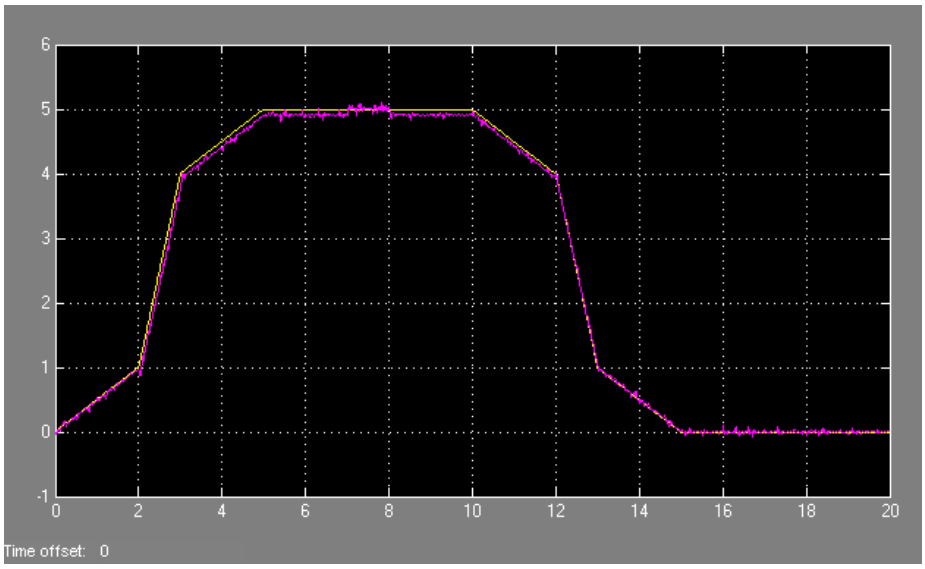

Figure 4: K_speed = 500 (accurate speed control)

12 Balance Control Simulation

The Balance Controller was evaluated on plots that show the trajectory traveled by the left and right wheel with respect to time. To recall, the purpose of the Balance Controller is to maintain the trajectories the same if possible, even if there are external forces that may block the wheel. In simulation, a resistance to the left wheel was applied for a period of time. If there was no Balance Regulator, the left wheel would stop for a while (yellow line in the plot), which would ultimately change the robot orientation, since the right wheel still rotates (pink line). After the intervention, the robot does not move in a line anymore. This scenario was plotted in Figure 5, the Balance PID regulator was effectively disabled by setting the gain $K = 0$.

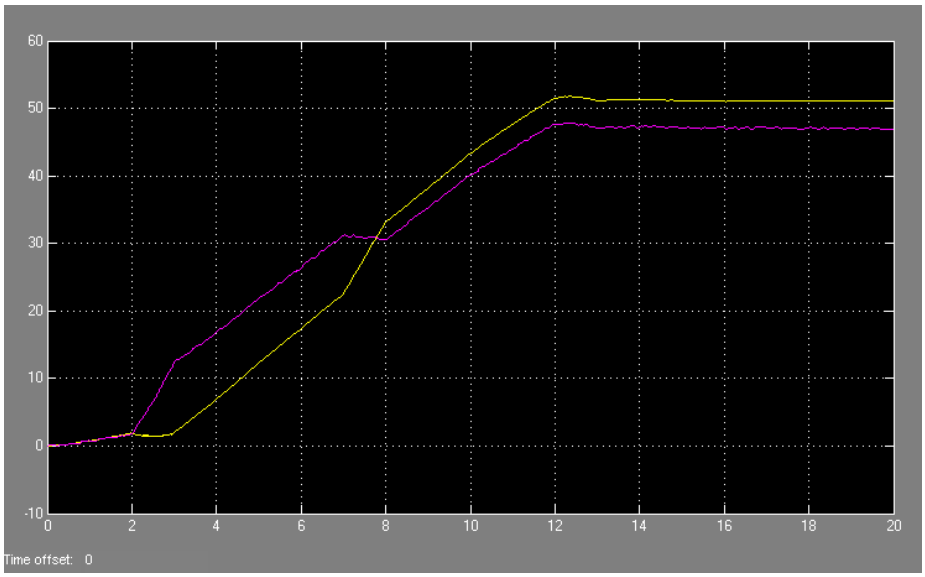

Figure 5: K_balance = 0 (without balance control)

Once the Balance Controller is enabled ($K = 0.001$), it compensates for the external forces and the robot still moves straight forward, with just a slight error in its trajectory. This is shown in Figure 6. Finally, we can increase the gain up to $K = 0.01$ to get perfect balance control, as shown in Figure 7.

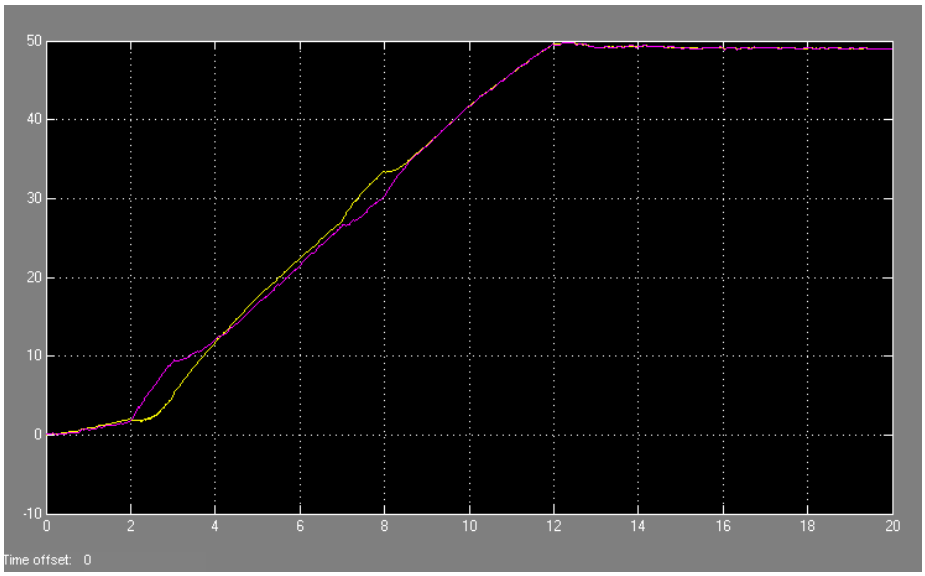

Figure 6: K_balance = 0.001 (moderate balance control)

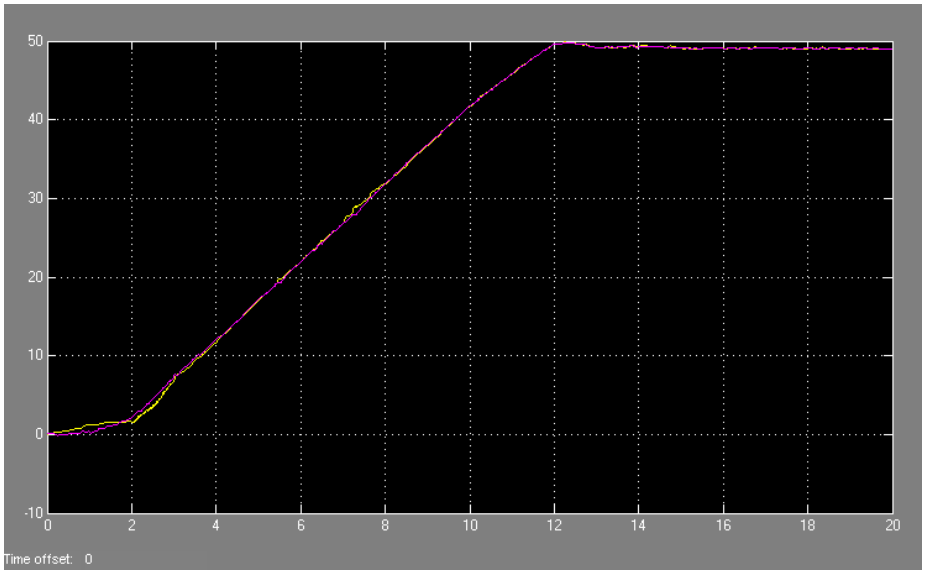

Figure 7: K_balance = 0.01 (accurate balance control)

Conclusion

A model of the movement control system for a differential driven robot was designed and verified in Matlab Simulink. When we are given the transfer function of the robot's motors, we can calculate the PID parameters for the Speed and Balance regulator. The simulations show very good results, even if there are extreme external interventions to the system, the controller still performs well.

Future works

The model can be further extended to support accurate positioning (distance control) and another types of movements, for example rotation in place, movement on a circular trajectory or a spline curve.

References

[1] Jan Christoph Schlake: Advanced Control Engineering, Block 5 – PID Control and Control Structures [lecture slides]

[2] https://en.wikipedia.org/wiki/Differential_wheeled_robot