ELEC-E8101 Group project: Final report Group 14

LOSI Jacopo, MAJOR Gábor, SALJOUGHI Nicola

December 15, 2019

Contents

1 Lab A

1.1 Task 4.1

We want to express the linearized EOM in the form of a general linear time-invariant (LTI) SS system:

$$\begin{cases} \dot{\mathbf{x}} = A\mathbf{x} + Bu \\ y = C\mathbf{x} + Du \end{cases}$$

The input is the voltage applied to the motors and the measurement is the angular deviation of the balancing robot from the vertical upright position:

$$u = v_m$$
$$y = \theta_b$$

We choose state $\mathbf{x}$ as:

$$\begin{bmatrix} x_w \\ \dot{x}_w \\ \theta_b \\ \dot{\theta}_b \end{bmatrix}$$

The obtained A, B, C and D matrices are here presented in parametric form:

$$A =$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -2\frac{(l_b\,(l_b+l_w)\,m_b+I_b)\,(K_e\,K_t+1/2\,b_f\,R_m)}{((I_b\,{l_w}^2+{l_b}^2\,({l_w}^2m_w+I_w))\,m_b+I_b\,({l_w}^2m_w+I_w))\,R_m} & -\frac{{m_b}^2{l_b}^2{l_w}^2g}{((m_w+m_b)\,{l_w}^2+I_w)\,I_b+{l_b}^2m_b\,({l_w}^2m_w+I_w)} & 2\frac{l_w\,(m_b\,{l_b}^2+m_b\,l_b\,l_w+I_b)\,(K_e\,K_t+1/2\,b_f\,R_m)}{(((l_b{}^2m_w+I_b)\,m_b+I_b\,m_w)\,{l_w}^2+I_w\,(m_b\,{l_b}^2+I_b))\,R_m} \\ 0 & 0 & 0 & 1 \\ 0 & 2\frac{((m_w+m_b)\,{l_w}^2+m_b\,l_b\,l_w+I_w)\,(K_e\,K_t+1/2\,b_f\,R_m)}{l_w\,(((l_b{}^2m_w+I_b)\,m_b+I_b\,m_w)\,{l_w}^2+I_w\,(m_b\,{l_b}^2+I_b))\,R_m} & \frac{m_b\,((m_w+m_b)\,{l_w}^2+I_w)\,gl_b}{((l_b{}^2m_w+I_b)\,m_b+I_b\,m_w)\,{l_w}^2+I_w\,(m_b\,{l_b}^2+I_b)} & -2\frac{((m_w+m_b)\,{l_w}^2+m_b\,l_b\,l_w+I_w)\,(K_e\,K_t+1/2\,b_f\,R_m)}{(((l_b{}^2m_w+I_b)\,m_b+I_b\,m_w)\,{l_w}^2+I_w\,(m_b\,{l_b}^2+I_b))\,R_m} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 2\dfrac{l_w\,K_t\,\left(m_b\,{l_b}^2+m_b\,l_b\,l_w+I_b\right)}{\left(\left(\left(m_w+m_b\right){l_w}^2+I_w\right)I_b+{l_b}^2m_b\,\left({l_w}^2m_w+I_w\right)\right)R_m} \\ 0 \\ -2\dfrac{K_t\,\left(\left(m_w+m_b\right){l_w}^2+m_b\,l_b\,l_w+I_w\right)}{\left(\left(\left({l_b}^2m_w+I_b\right)m_b+I_b\,m_w\right){l_w}^2+I_w\,\left({l_b}^2m_b+I_b\right)\right)R_m} \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \end{bmatrix}$$

Here we report matrices A, B, C and D in numeric form:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -773.7853734 & -6.573516819 & 16.24949284 \\ 0 & 0 & 0 & 1 \\ 0 & 3313.238430 & 63.07193800 & -69.57800702 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 36.59795686 \\ 0 \\ -156.7072230 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \end{bmatrix}$$

1.2 Task 4.2

The transfer function of an LTI SS system is given by the formula:

$$G(s) = C(sI - A)^{-1}B + D$$

The result reported has been obtained by computing an analytical expression of the transfer function and then substituting the data.

$$G(s) = -156.70 \frac{s}{(s + 843.40)(s + 5.64)(s - 5.67)}$$

At first we used the numeric forms of the matrices A, B, C and D to define the SS system in `MATLAB` and then to compute the transfer function. This however resulted in strange results, giving us additional poles and complex conjugate zeros (with real part = 0 and imaginary part = $\pm i$). We then tried to go fully analytical and we got much more reasonable results; in this phase we also compared the results obtained using `Maple` and `MATLAB`, concluding that they are quite different since these softwares run in completely different manners. From this point onwards we only used `MATLAB`.

1.3 Task 4.3

We would like to design a PID controller using the pole placement method. We have a transfer function of the form:

$$G(s) = K \frac{s}{(s - p_1)(s - p_2)(s - p_3)}$$

One of the poles has positive real part, making the system unstable (i.e. $p_3 = 5.67$). We want to change this pole by using the PID controller; a good candidate would be $p_3' = -3$, along the real axis.
Following the pole placement method the following values for the controller parameteres were obtained:

- $K_P = -46.976$ for the proportional gain;
- $K_I = -263.186$ for the integral gain;
- $K_D = -0.055$ for the derivative gain.

The resulting closed loop function in its numerical form is here reported:

$$H(s) = \frac{8.62\,s^2 + 7361.48\,s + 41243.15}{(s + 843.35)\,(s + 5.65)\,(s + 2.98)}$$

The PID controller was implemented in `SIMULINK` through the `PID Controller` block. This block requires the `Filter coefficient (N)` to be set. At first, it was used the default value, i.e. $N = 100$, and subsequently some of other values were tried. By doing this, it was observed that the output of the system does not change setting both positive values not extremely far from the standard one or small positive values. Moreover, the input motor voltage does not change neither by increasing the value of the Filter coefficient to a huge value and this outcome is probably due to some errors done during some of the calculations. As a matter of fact, the input motor voltage should slightly change, because, apply a very high Filter coefficient leads to approach a pure derivative controller. At the end, the outputs obtained and shown in the figure (3a), (3b), (17) below were obtained using the default value.

1.4 Task 4.4

The new EOM obtained by including the effect of disturbance are:

$$\begin{cases} \left({l_b}^2 m_b + I_b\right)\ddot{\theta}_b = m_b\, l_b\, g sin(\theta_b) - m_b\, l_b\, \ddot{x}_w cos(\theta_b) - 2\,\dfrac{K_t\, v_m}{R_m} + \left(2\,\dfrac{K_e\, K_t}{R_m} + b_f\right)\left(\dfrac{\dot{x}_w}{l_w} - \dot{\theta}_b\right) + l_w d \\ \\ \left(\dfrac{I_w}{l_w} + l_w\, m_b + l_w\, m_w\right)\ddot{x}_w = -m_b\, l_b\, l_w\, \ddot{\theta}_b cos(\theta_b) + m_b\, l_b\, l_w\, \dot{\theta}_b^2 sin(\theta_b) + 2\,\dfrac{K_t\, v_m}{R_m} - \left(2\,\dfrac{K_e\, K_t}{R_m} + b_f\right)\left(\dfrac{\dot{x}_w}{l_w} - \dot{\theta}_b\right) + l_b d\cos\left(\theta_b\right) \end{cases}$$

Here we report the new linearized EOM in numerical state space form:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -773.7853734 & -6.573516819 & 16.24949284 \\ 0 & 0 & 0 & 1 \\ 0 & 3313.238430 & 63.07193800 & -69.57800702 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 36.59795684 & 17.64000724 \\ 0 & 0 \\ -156.7072230 & -66.40438947 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

Now our input vector is:

$$u = \begin{bmatrix} v_m \\ d \end{bmatrix}$$

while state $\mathbf{x}$ and y remain unchanged.

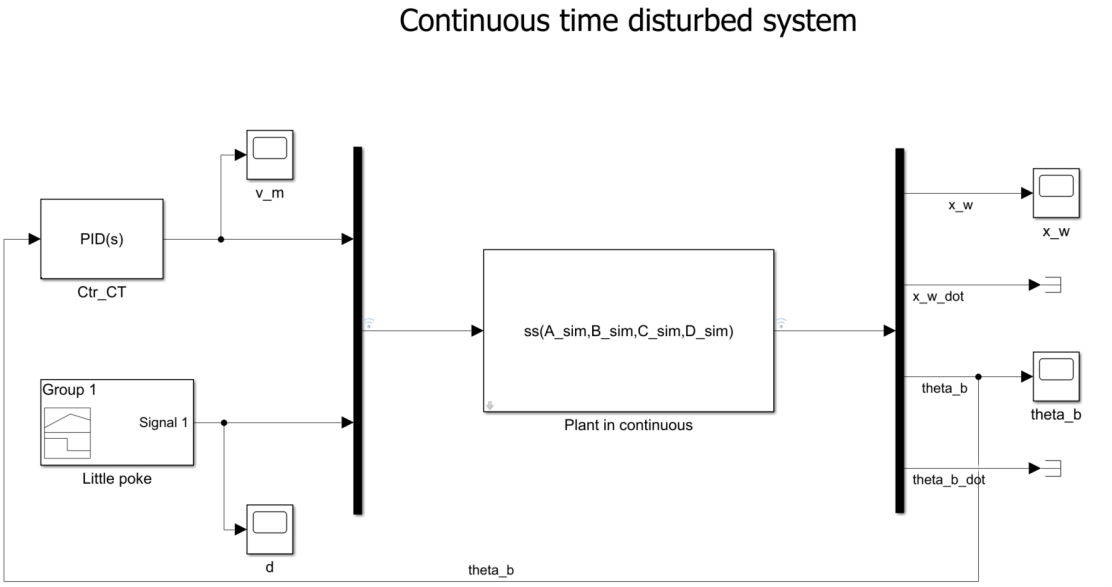
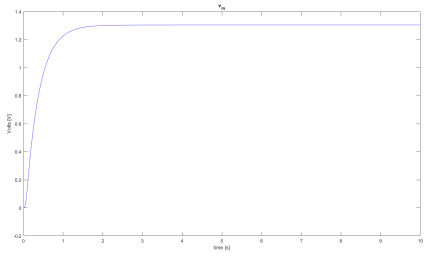
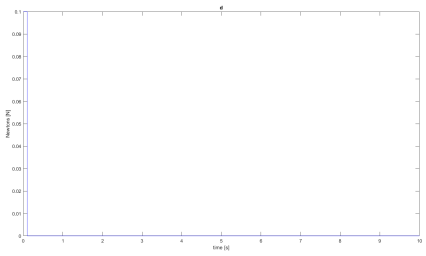


Figure 1: `Simulink` solution for task 4.5



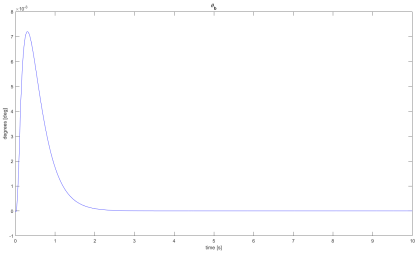
(a) Voltage of the motors v_m



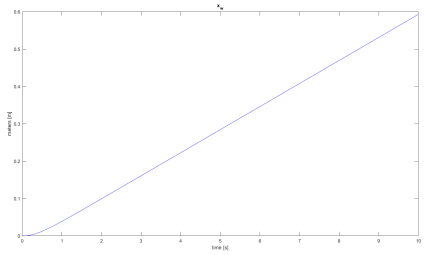
(b) Disturbance applied d

1.5 Task 4.5

Figure 1 shows the obtained `SIMULINK` scheme while Figure 2, Figure 3, Figure 4 and Figure 5 show the realisations of $\theta_b(t)$, $x_w(t)$, $v_m(t)$ and $d(t)$.
We can observe that the voltage required to use such a controller as the one we have designed is too high given the batteries we have. Further improvement is required to make the controller feasible.



(a) Inclination of the robot θ_b



(b) Position of the wheels x_w

1.6 Task 4.6

In order to discretize the controller transfer function $C(s)$, it is necessary to choose the sampling time correctly. As a matter of fact, if a too large value of the sampling interval is used in the Analog-Digital Conversion, it could lead the robot system to be unstable. Therefore, in order to find the correct sampling rate, the bandwidth of the transfer function of the system was set to the frequency where three decibels (dB) are lost from the peak magnitude of the plot of the transfer function. Thus, using the `MATLAB`'s Data Tips instrument the peak magnitude was found and then the frequency correspondent to the $3dB$ loss was identified. Regarding this, as shown in figure(**??**) the peak magnitude is at a frequency of about $5.38 rad/s$, hence the cut-off frequency, i.e. the bandwidth, is around $13.6 rad/s$, that is almost $-3dB$ with respect to the peak magnitude.
Then, in order to apply the discretization procedure correctly, it is necessary to transform the frequency from rad/s to $Hertz$, that means use the following equation:

$$1\frac{rad}{s} = \frac{1}{2\pi} Hertz$$

Thus, the bandwidth becomes:

$$13.6\ dB = 2.16\ Hertz$$

At this point it is necessary to define the sampling period T, that is given through the sampling frequency and it is used to convert the transfer function from the continuous (analog) to the discrete (digital) domain. Regarding the sampling frequency, it is conventionally taken as 25 times the cut-off frequency. Thus, the result will be:

$$S_{freq} = 25\omega_0 = 25 \cdot 2.16 = 54.11\ Hertz$$

Therefore, the sampling period becomes:

$$T = \frac{1}{54.11} = 0.018 \approx 0.02$$

The discrete-time transfer function C(z) of our controller in its parametric form is:

$$C(z) = K_p + K_I \frac{T}{z-1} + K_D \frac{z-1}{z}$$

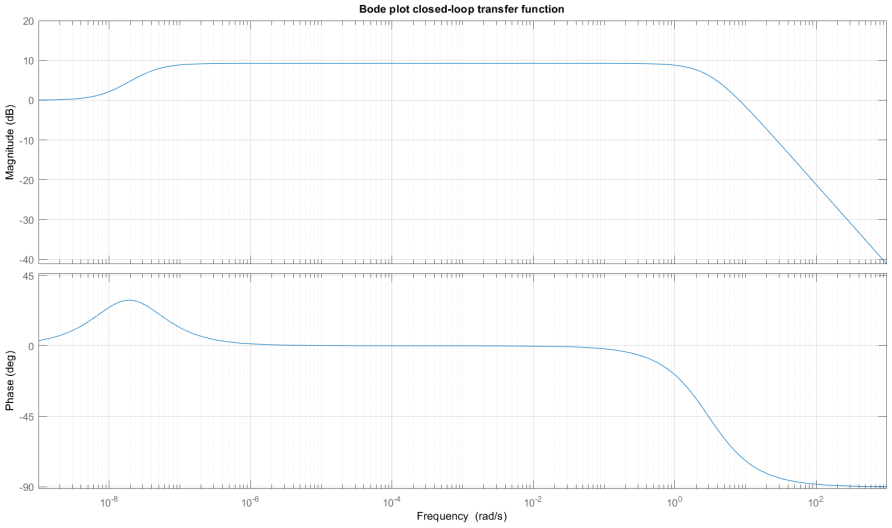
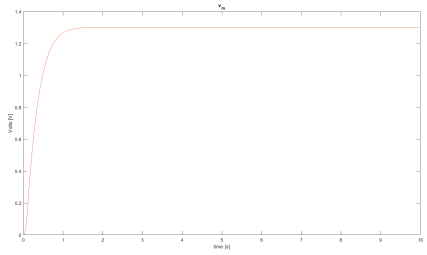
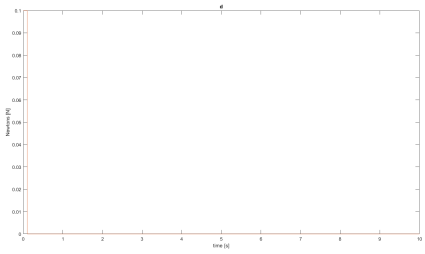


Figure 4: Magnitude and Phase of the Bode plot of the system transfer function



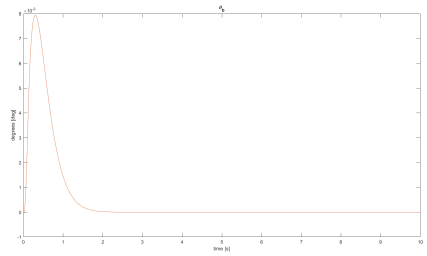
(a) Discretized voltage of the motors v_m



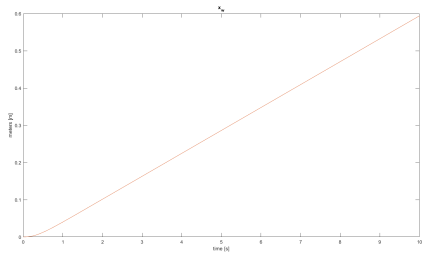
(b) Discretized disturbance applied d

1.7 Task 4.7

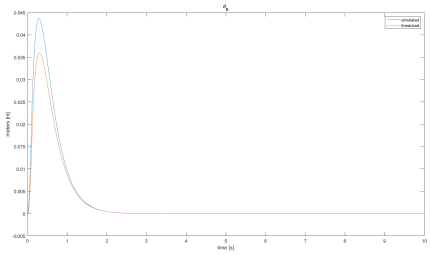
Continuous Time These are the results for a disturbance of 0.1. The robot fall if the disturbance is bigger than 0.59.



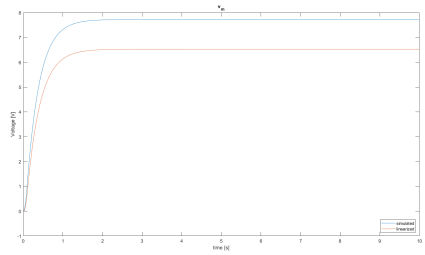
(a) Discretized inclination of the robot θ_b



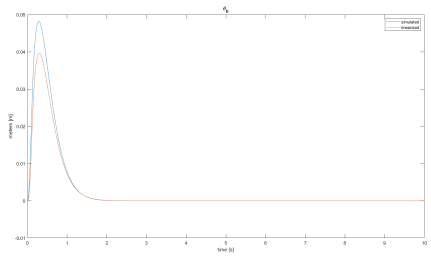
(b) Discretized position of the wheels x_w



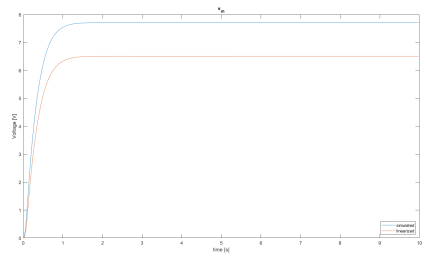
(a) Linearized and Simulated plots for θ_b in continuous case



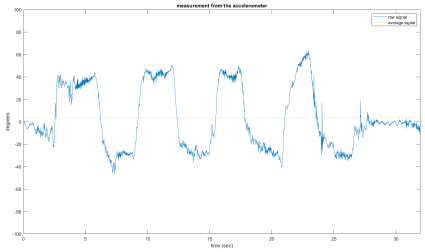
(b) Linearized and Simulated plots for v_m in continuous case



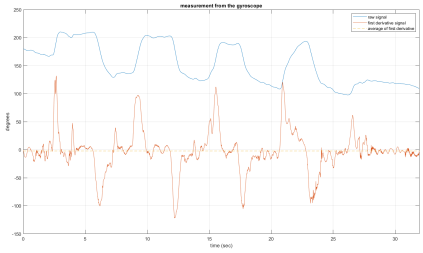
(a) Linearized and Simulated plots for θ_b in discrete case



(b) Linearized and Simulated plots for v_m in discrete case



(a) Plot of the Accelerometer signal



(b) Plot of the Gyroscope signal

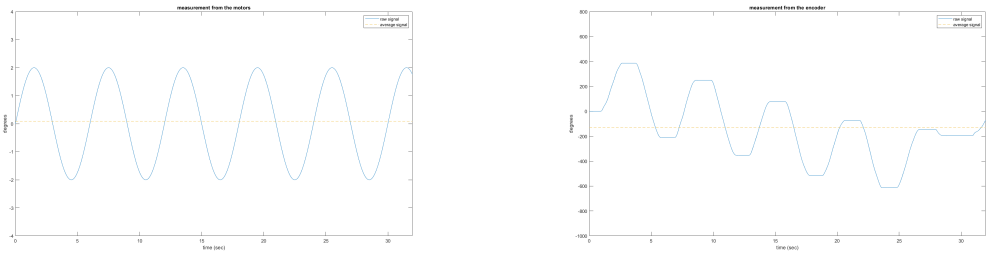
Figure 9: Accelerometer and Gyroscope signals

2 Lab B

The main target of this section of the project is to start designing a controller using continuous time (CT) concepts. Basically, we will start dealing with a CT approach and then we will discretize the results in order to have a more efficient implementation with the Arduino 2560 Board.
As a preparatory task for the this laboratory, we have checked that the communication between the computer and the board performed correctly. Thus, the Arduino 2560 was connected to the computer through a USB cable, and running the provided SIMULINK file `LabB_CheckCommunications.slx`, it was checked that the communication worked correctly. Therefore, the data regarding the signals of interest were gathered to have a response of the effective communication and to understand if the sensors worked properly. The outcomes are shown in Figure (9) and (10), where the signals of the different sensors are plotted. Moreover, interesting is in particular Figure (9b), where it is possible to notice that the raw signal has values around 180°, but its first derivative, thus its variation is stable around zero. This bias of the gyroscope is due to the fact that in the SIMULINK block no bias is added to the sensor's measurement. As a matter of fact, for the following implementations the gyroscope will be calibrated and a correct bias will be added to the signal to have a precise result.
The stable behaviour around zero is highlighted by the plots of the other sensors too. Therefore, making reference to Figure(9) and (10), the correctness of the communication can be considered valid.
The only things that should be underlined regards the accelerometer. As matter of fact, its signal presents lots of small oscillations, that are probably due to some high frequency disturbances, both in the positive and in the negative peaks. This behaviour suggests that some filtering has to be applied to that sensor in order to have a smoother signal. Moreover, also for the accelerometer a value of the bias equal to zero is added to the block. Therefore, also this sensor will be correctly calibrated.

2.1 Task 5.1

As aforementioned, the gyroscope was calibrated adding a bias to the output of the sensor. The calibration was performed exploiting the given MATLAB file `LabB_TuneTheGyro_SaveParameters.m` that basically tunes the bias of the sensor computing a moving average over the received signal.
After this first step of the calibration, the PID controller defined in the previous laboratory was tested on the robot. Specifically, multiple sets of values for the gains of the controller were set. This was done in order to analyse the response of the system controllers with



(a) Plot of the Motor signal (b) Plot of the Encoder signal

Figure 10: Encoder and Motors signals

Number of Test	K_P	K_I	K_D
1	-47.02	-263.54	-0.055
2	-80.00	-300.00	-0.060
3	-150.00	-500.00	-0.10
4	-200.00	-500.00	-0.10
5	-200.00	-300.00	-0.055

Table 1: Testing of the PID controller on the robot

different characteristics. In Table (1) the tested values are reported and some discussion about the outcomes registered are presented below.

The first test was completed using the values of the gains computed in the previous laboratory section, which were obtained applying the poles placement method on the system.

Then, regarding the other cases described in Table (1), it was though to try to increase the values of all the constants, in order to have a stronger controller. As a matter of fact, the overall performance of the system has progressively improved. This was an expected result, being known that if the proportional and the integral gains grow the system becomes more stable. This is due to the fact that, with higher gains, the controller *knows* more about the system. Specifically, the proportional gain is responsible for measuring *how far* the process variable is away from the set point, whereas the integral term sums over the error to determine *how long* the process value has been away from the target. Then, the derivative coefficient assesses *how fast* error in the process is changing. As the rate of error either increases or decreases so in the same way does the magnitude of the derivative response. Therefore, in order to have a small error, it is a good practice to have a small value of the derivative controller, as it was done for all the tests. Moreover, regarding the proportional and the integral gains, increasing their values the response of the system improves. This is a clear result, because of with an higher value of the proportional term the response of the system to an increase of the error is faster, thus increasing K_P the system recover the error in less time. Furthermore, considering the integral term, increasing its size the system *knows* more about its past. In a nutshell, it is like having a system with a bigger memory.

Therefore, developing the presented tests, it was clear that increasing (in magnitude) the values of the coefficients of the PID, the system becomes more stable. Considering this, it is also necessary to evaluate the results obtained in terms of feasibility. In fact, a trade-off between usable values of the PID and the performance should be taken into account. This

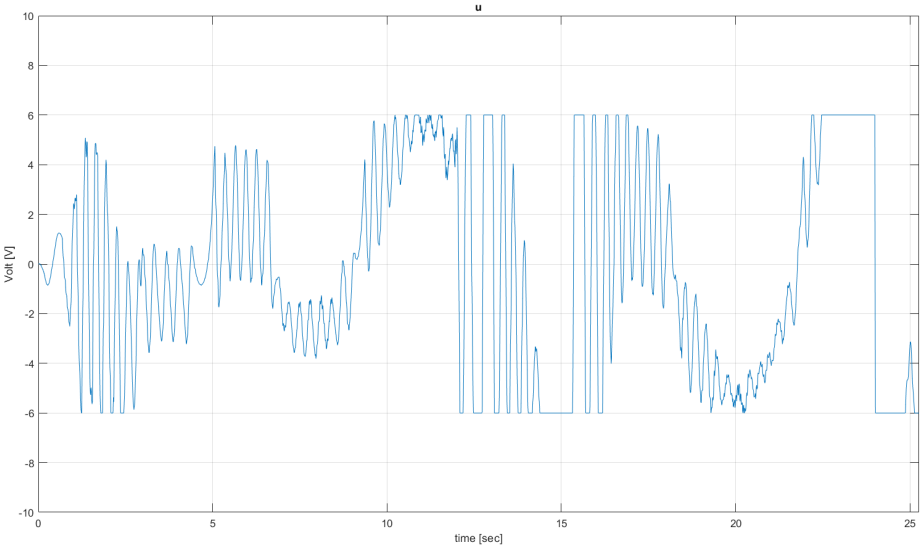


Figure 11: Input to the system

means that, an increasing in the values of K_P and K_I will affect both the saturation of the batteries and the position of the poles of the system.
Thus, it is not always possible to set optimal values for the PID gains but it is also necessary to design a stronger system with different techniques.
Besides this, the best performance at this point of the analysis was obtained using the values in the fifth case where the values of K_I and K_D are similar to the one obtained in the previous part of the project. Instead of, the value of K_P is highly different from the one in the first case and this allows to have a fast response of the system to changes. In Figure (11), (12) and (13) the results obtained with the PID described above, respectively for the input, the position of the wheel and the angle of the robot are shown.

2.2 Task 5.2

The *controllability matrix* of a system is given by

$$\mathcal{C} := \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix}$$

while the *observability matrix* of a system is given by

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

where n is the number of states (i.e. dimension of the state vector).
The obtained controllability and observability matrices for our case are here reported in

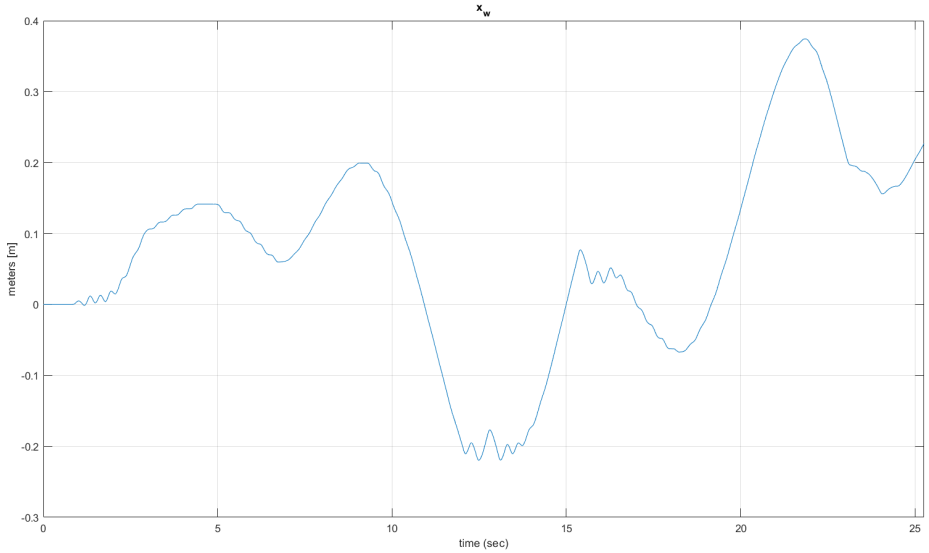


Figure 12: Position of the wheels

numerical form:

$$\mathcal{C} = \begin{bmatrix} 0 & 36.59 & -30865.38 & 26031758.49 \\ 36.59 & -30865.38 & 26031758.49 & -21955189530 \\ 0 & -156.71 & 132161.13 & -111469744.1 \\ -156.71 & 132161.13 & -111469744.1 & 94013600850 \end{bmatrix}$$

$$\mathcal{O} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 3313.24 & 63.07 & -69.58 \\ 0 & 3313.24 & 63.07 & -69.58 \end{bmatrix}$$

The results show that the system is not completely observable, due to the fact that the observability matrix is not full-rank. The controllability matrix is instead full-rank and therefore the system is controllable. Let's try to understand what this means in practice by looking at how we modelled our system.
The system in SS form is represented by

$$\begin{cases} \begin{bmatrix} \dot{x}_w \\ \ddot{x}_w \\ \dot{\theta}_b \\ \ddot{\theta}_b \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & 0 & 1 \\ 0 & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_w \\ \dot{x}_w \\ \theta_b \\ \dot{\theta}_b \end{bmatrix} + \begin{bmatrix} 0 \\ b_{21} \\ 0 \\ b_{31} \end{bmatrix} u \\ y = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_w \\ \dot{x}_w \\ \theta_b \\ \dot{\theta}_b \end{bmatrix} + 0\, u \end{cases}$$

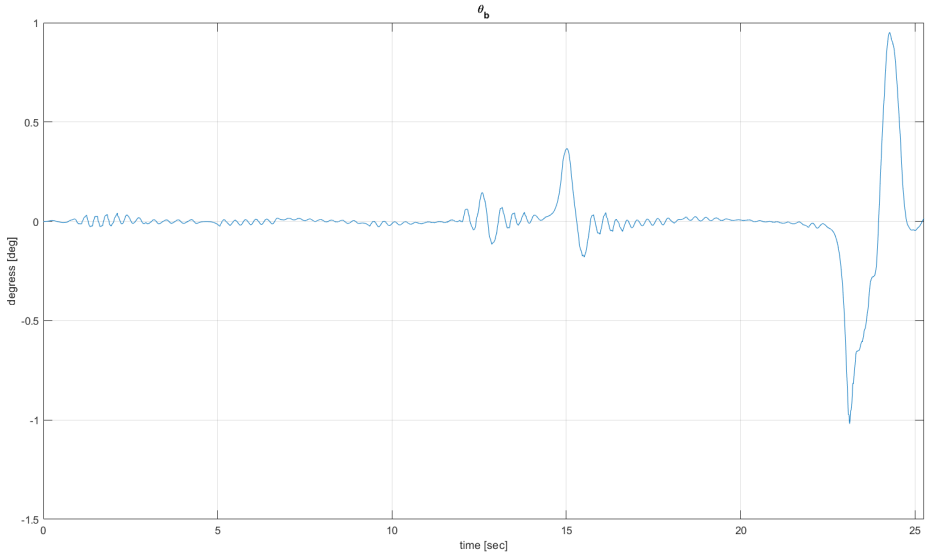


Figure 13: Angle of the robot wrt the vertical

We can not compute the canonical observable decomposition, therefore we use a scheme to analyse the practical implications of the characteristics of the system. Figure 11 shows the scheme of the system:

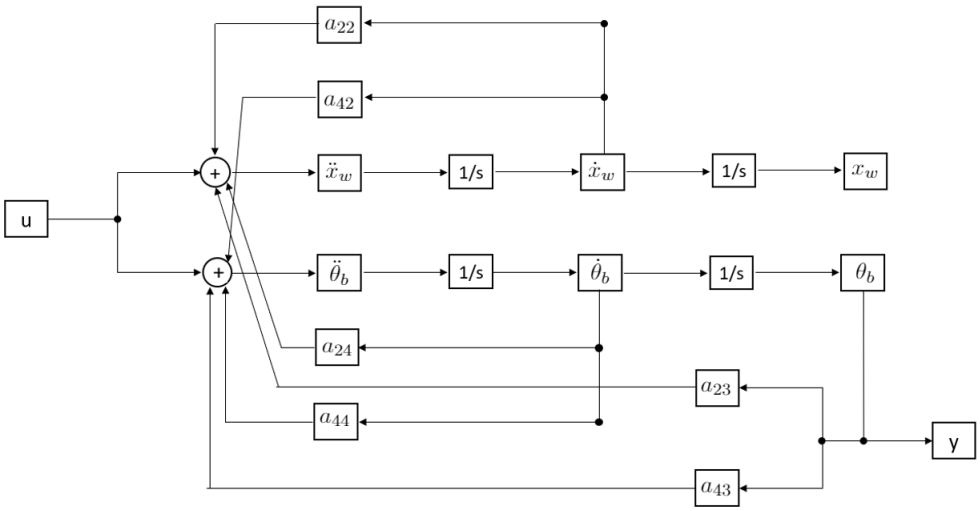


Figure 14: System block-scheme

As we can see the input influences all the states of the system; regarding the output, only θ_b is measured but it is itself dependent on all the other states but the position x_w. This means that this is the only non-observable state, while all the states are controllable. In order to fix the problem we could measure the position x_w as well, and by doing this the system would become observable.
The lack of observability of the system causes zero-pole cancellations in the transfer function.

2.3 Task 5.3

Considering the model developed in Lab A, the input that can be actually commanded is the voltage to the motors. Therefore, we should consider the first linearized equations, i.e., the ones without the disturbance. As a matter of fact, this external force, that we have modelled as a poke applied to the robot cannot be precisely controlled. Basically, the disturbance d was used in Lab A to analyse the effects of common disturbances which can affect the system. Thus, in order to model them in our equations, it was useful to design them as an external horizontal force applied to the centre of mass of the robot.
Therefore, we built a controller through a second order approximation of the plant using the `MATLAB` built-in function `acker`, which can be used only for SISO (single input single output) systems. Basically, this function gives as output a gain matrix, generally called **K**, that has the function of a controller for the Plant. As inputs to the function, it is needed to put the matrices **A** and **B** of the state-space form, that in our case are the matrices calculated for the first linearized equations. Then, the other function's input is the position of the poles that we want to obtain for the closed loop system. Considering this, it was chosen to operate in the following way:

- the two stable poles of the open loop transfer function was kept;
- the unstable pole at $p = 5.67$ was shifted to $p = -3$
- then the integrator, that is the pole at zero, was shifted to $p = -5$

Therefore, the final poles were put at: $p1 = -843.40$, that is the fast pole correlated to the motor, $p2 = -5.64$, which are the two stable poles, then $p3 = -3$ and $p4 = -5$, that is the added quite slower pole.
Applying the `acker` function, the gain matrix **K** obtaines is:

$$\mathbf{K} = [-55.8226 \ -60.8787 \ -87.2338 \ -14.3051]$$

The results for the angle θ_b and for the voltage in input v_m are presented in Figure (15) and (16).

2.4 Task 5.4

The full order observer structure is given by

$$\dot{\hat{\mathbf{x}}} = A\hat{\mathbf{x}} + Bu + L(\mathbf{y} - C\hat{\mathbf{x}}),$$

and L can be designed through `L = ( place( A', C', afPoles ) )';`, where `afPoles` are the poles of the closed-loop system. In the previous phases we designed a PID controller using the pole placement method to obtain three poles in our closed-loop transfer function; the poles are

$$p_1 = -843.4002$$
$$p_2 = -5.6430$$
$$p_3 = -2.9992$$

Now, for the full order observer structure matrices A is

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -773.7853734 & -6.573516819 & 16.24949284 \\ 0 & 0 & 0 & 1 \\ 0 & 3313.238430 & 63.07193800 & -69.57800702 \end{bmatrix}$$

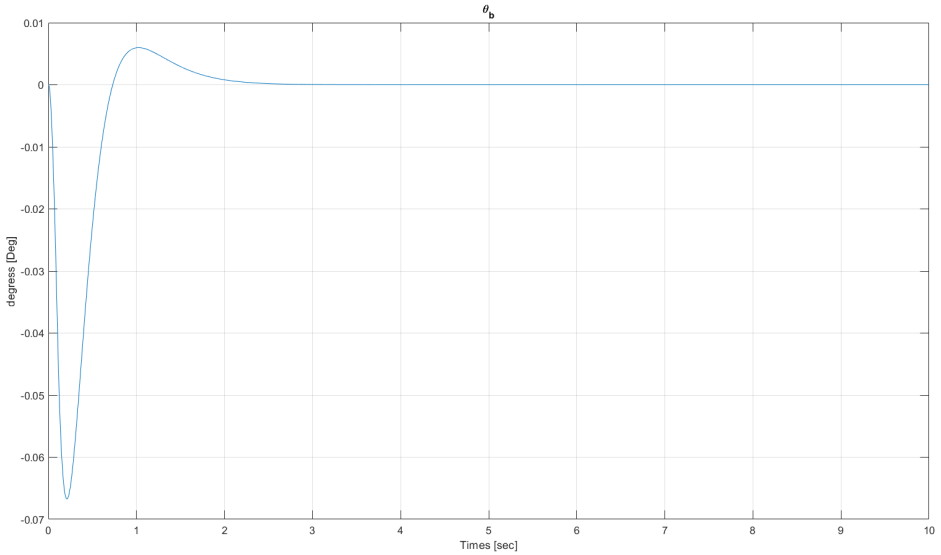


Figure 15: Angle of the robot using **K** in continuous time

and C is

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The poles need to be of number equal to the number of rows of A^T and C^T, i.e. 4. This implies that we need to add another pole to the closed-loop transfer function, and we chose it to be

$$p_4 = -5.$$

The obtained gain of the full order estimator, L, is

$$L = \begin{bmatrix} 5.4660 & 0.1405 \\ -0.6181 & 0.7596 \\ 1.5540 & 8.2130 \\ 9.2225 & 48.6239 \end{bmatrix}.$$

For the reduced order system, the obtained gains of the estimator $M1$, $M2$, $M3$, $M4$, $M5$, $M6$, $M7$ are

$$M1 = \begin{bmatrix} -9.6581 & -61.2665 & 1 \\ -58.8137 & -772.8063 & 16.2495 \\ 3123.9529 & 3313.2 & -69.5780 \end{bmatrix}, \; M2 = \begin{bmatrix} 0 \\ 36.5980 \\ -156.7072 \end{bmatrix}, \; M3 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$M4 = \begin{bmatrix} 9.6581 \\ 52.2402 \\ -171.3052 \end{bmatrix}, \; M5 = \begin{bmatrix} 61.2665 \\ -0.9790 \\ 189.2855 \end{bmatrix}, \; M6 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \; M7 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The calculated errors are the following:

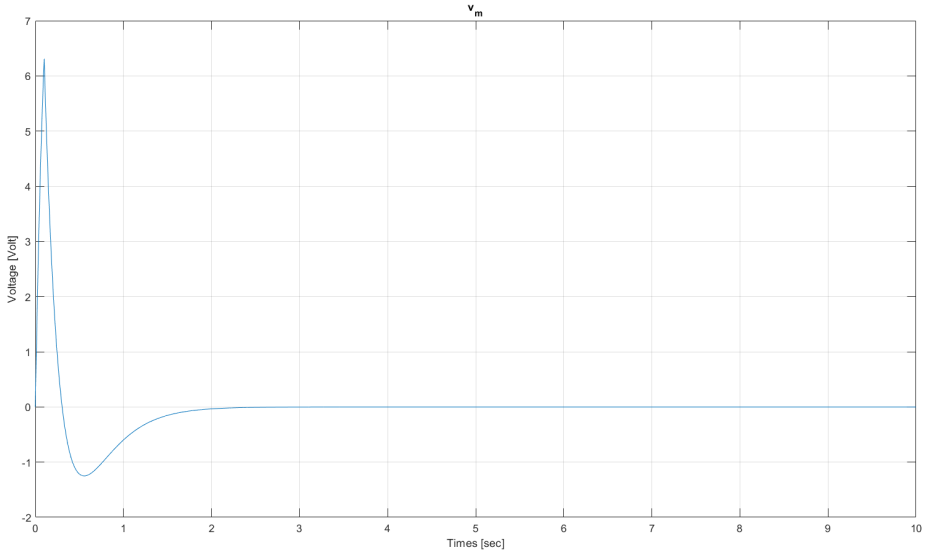


Figure 16: Input to the system using **K** in continuous time

- max error normal-full for x_w : $|x_w(t) - \hat{x}_w^{full}(t)| = 2.381510^{-04}$
- max error normal-reduced for x_w : $|x_w(t) - \hat{x}_w^{reduced}(t)| = 0$
- max error normal-full for θ : $|\theta_b(t) - \hat{\theta}_b^{full}(t)| = 0.0179$
- max error normal-full for θ : $|\theta_b(t) - \hat{\theta}_b^{reduced}(t)| = 0.0052$

2.5 Task 5.5

To obtain matrices `Ad`, `Bd`, `Cd` and `Dd` we have used `MATLAB` and we followed this procedure:

- define original system in SS form using the syntax `sys = ss(A, B, C, D);`
- discretize the system using the syntax `sys_disc = c2d(sys, fSamplingPeriod, 'zoh');`
- extract the matrices `Ad`, `Bd`, `Cd` and `Dd` from the discrete system by using the syntax `[Ad, Bd, Cd, Dd] = ssdata(sys_discr);`

The obtained matrices are

$$Ad = \begin{bmatrix} 0 & 0.0019 & 0 & 0.0002 \\ 0 & 0.0825 & -0.0019 & 0.0192 \\ 0 & 0.0346 & 1.0019 & 0.0093 \\ 0 & 3.9338 & 0.3575 & 0.9193 \end{bmatrix}$$

$$Bd = \begin{bmatrix} 0.0004 \\ 0.0434 \\ -0.0016 \\ -0.1861 \end{bmatrix}$$

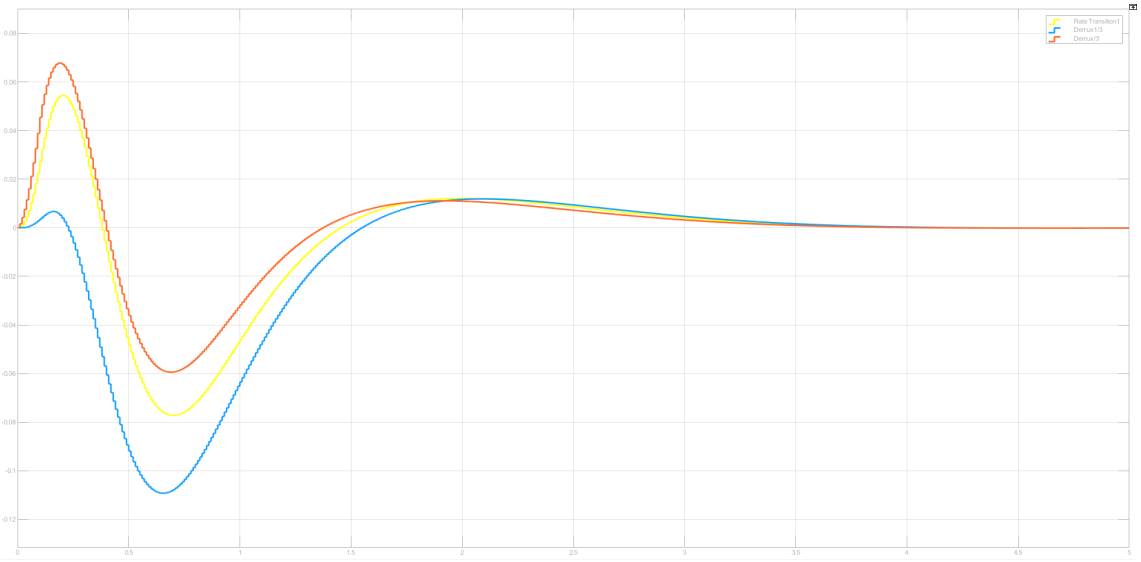


Figure 17: θ_b

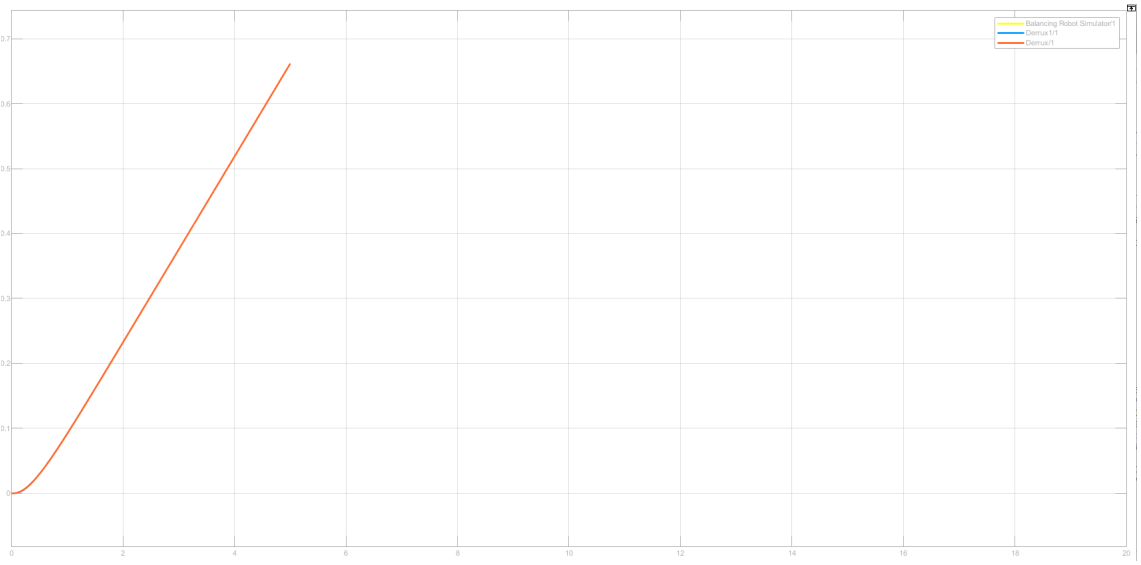


Figure 18: x_w

$$Cd = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$Dd = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

To compute matrices `Kd`, `Ld`, `Md1`, `Md2`, `Md3`, `Md4`, `Md5`, `Md6` and `Md7` we followed the same procedure as in the continuous case, but first we mapped the poles from the continuous to the discrete domain. The poles in the discrete domain are

$$p_1 = 0.002$$
$$p_2 = 0.9451$$
$$p_3 = 0.9705$$

Also in this case to compute `Kd` and `Ld` we introduced a fourth pole, and since in the continuous case it was $p_4 = -5$ in the discrete domain it maps to

$$p_4 = 0.9512.$$

The obtained matrices are:

$$Kd = \begin{bmatrix} -52.1469 & -58.5285 & -83.9040 & -13.7533 \end{bmatrix}$$

$$Ld = \begin{bmatrix} 0.0530 & 0.0014 \\ 0.0076 & 0.0109 \\ 0.0159 & 0.0837 \\ 0.0362 & 0.4802 \end{bmatrix}$$

$$Md1 = \begin{bmatrix} 0.9454 & -0.0845 & -0.0013 \\ -0.1028 & 0.0881 & 0.0197 \\ 0.3638 & 3.5157 & 0.8822 \end{bmatrix}, \; Md2 = \begin{bmatrix} -0.0255 \\ 0.0445 \\ -0.2697 \end{bmatrix}, \; Md3 = \begin{bmatrix} -62.3405 \\ 2.9244 \\ -218.6869 \end{bmatrix}$$

$$Md4 = \begin{bmatrix} 0.0591 \\ 0.1008 \\ 0.0029 \end{bmatrix}, \; Md5 = \begin{bmatrix} 62.3405 \\ -2.9244 \\ 218.6869 \end{bmatrix}, \; Md6 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \; Md7 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The following figures show the plots obtained from the simulation for θ_b, x_w and u respectively.

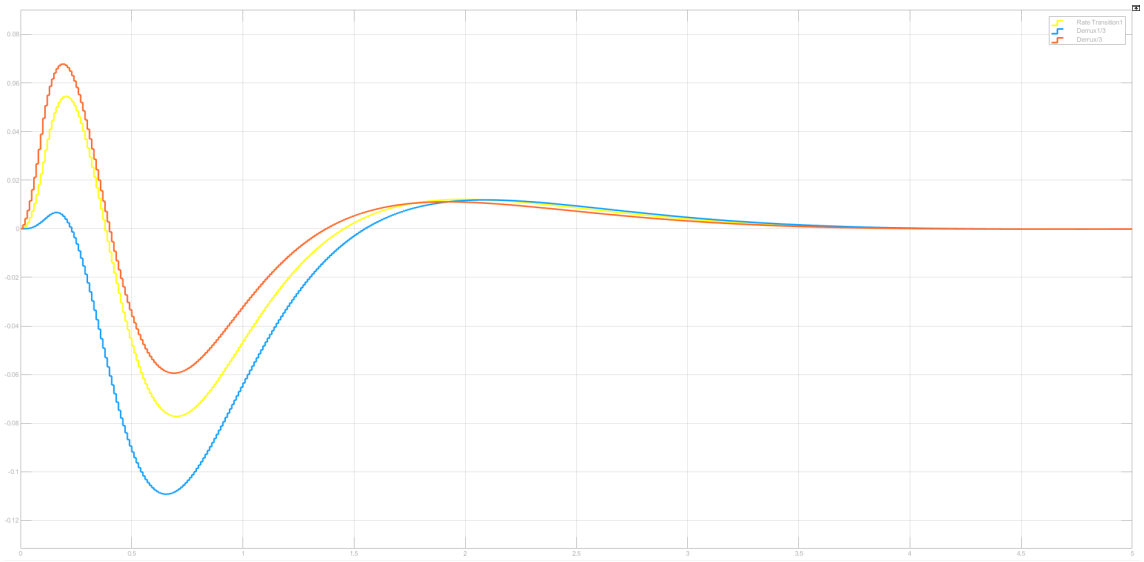


Figure 19: θ_b

Figure 20: x_w

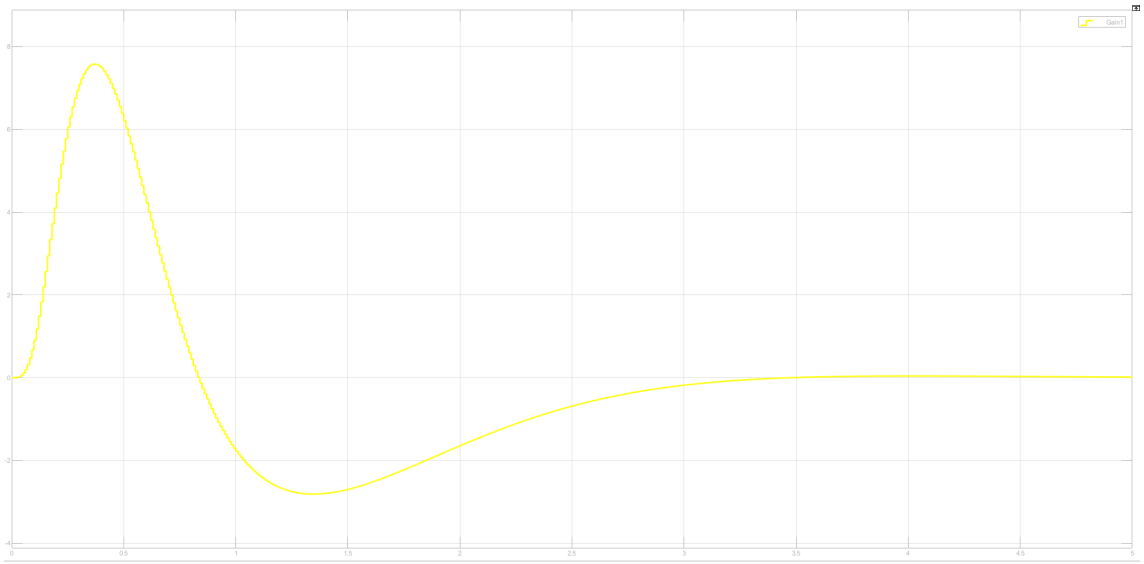


Figure 21: u

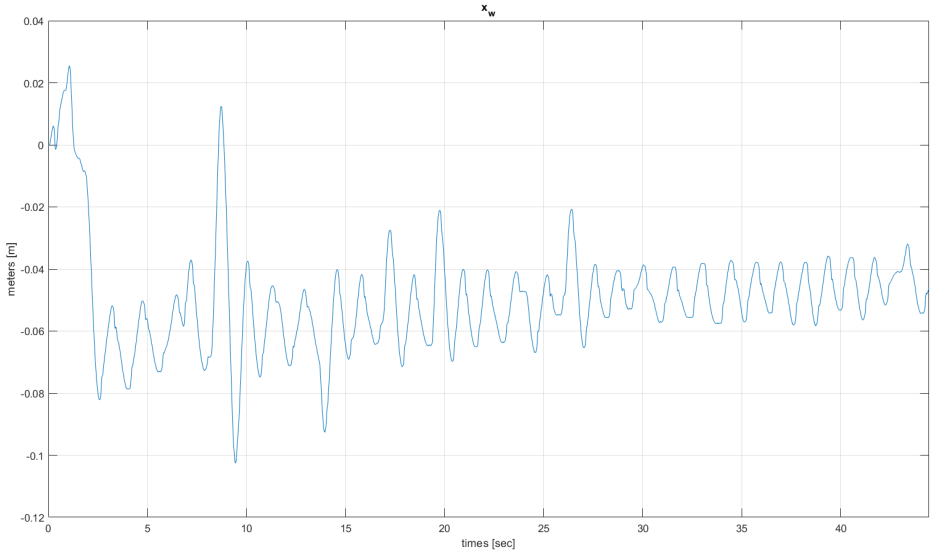


Figure 22: Measured relative position of the wheels using numerical observer

2.6 Task 5.6

With the first implementation of the Luenberger estimator (both full and reduced order) we were not able to obtain satisfactory results in the testing phase. We tried to obtain better results by changing the poles of the closed loop system in order to obtain a better estimator that converges rapidly to the real behaviour of the system and therefore is able to properly estimate its states; the poles taken by the function `L = ( place( A', C', poles )` determine the convergence properties of the observer; based on this considerations we used the following poles to compute the gain of the Luenberger estimator

$$
\begin{aligned}
p_1 &= -843.35 \\
p_2 &= -5.65 \\
p_3 &= -15.00 \\
p_4 &= -20.00
\end{aligned}
$$

that we then mapped in the discrete domain and used to design our discrete-time Luenberger observer.
Even though the simulation seemed to be quite satisfactory using these parameters, in the testing phase the robot could not manage to balance itself neither with the full nor with the reduced order estimator; we have not been able to identify the reason.
We report the results obtained by using the numerical observer, the only one that made the robot capable of balancing itself quite well, keeping it balanced for more than 40 seconds. The robot also managed to keep itself balanced when subjected to small external disturbances (i.e. small pokes).

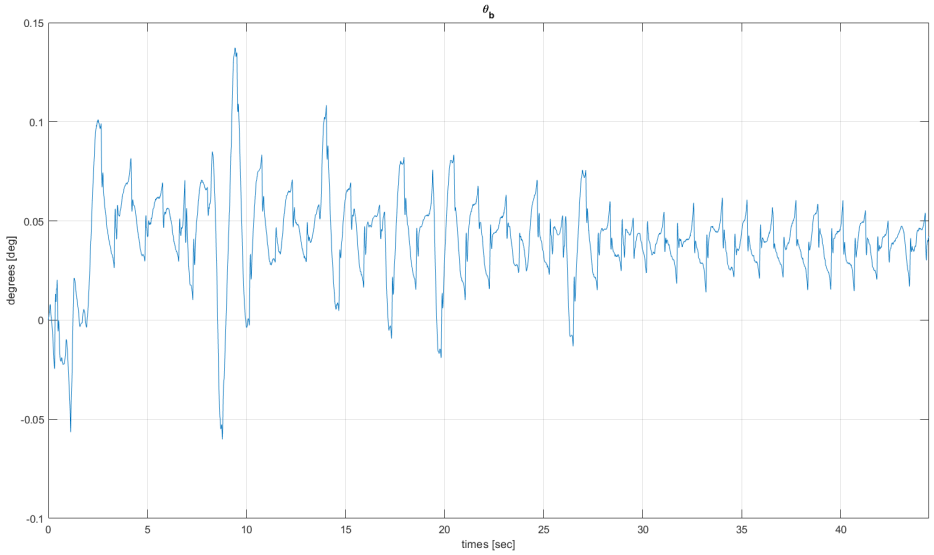


Figure 23: Measured relative angle θ_b using numerical observer

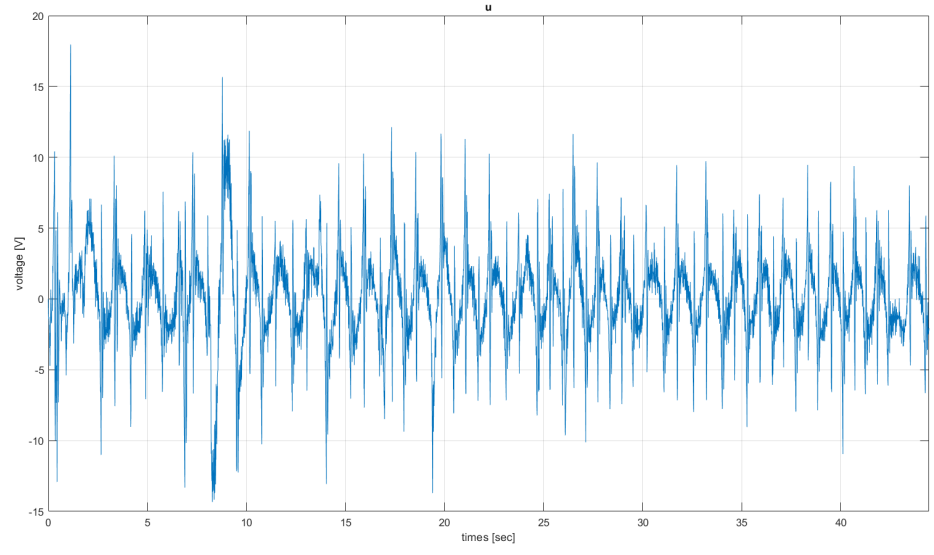


Figure 24: Measured input voltage using numerical observer

3 Lab C

NOTE: Since we changed the batteries in our robot, the voltage supply was changed from 7.5 V to 9 V.

3.1 Task 6.1

The best sampling frequency that we obtained at the end of the lab is

$$f_s = 160\ Hz$$

and the numerical values for A_d, B_d, C_d and D_d relative to this sampling frequency are

$$Ad = \begin{bmatrix} 1 & 0.0016 & 0 & 0.0001 \\ 0 & 0.0882 & -0.0045 & 0.0191 \\ 0 & 0.0189 & 1.0008 & 0.0056 \\ 0 & 3.9058 & 0.2289 & 0.9187 \end{bmatrix}$$

$$Bd = \begin{bmatrix} 0.0002 \\ 0.0431 \\ -0.0009 \\ -0.1847 \end{bmatrix}$$

$$Cd = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$$

$$Dd = \begin{bmatrix} 0 \end{bmatrix}.$$

3.2 Task 6.2

Our choice for matrix $\bar{C}$ and for ρ are

$$\bar{C} = \begin{bmatrix} 5 & 1 & 10 & 2 \end{bmatrix}$$

$$\rho = 0.1\ .$$

To choose $\bar{C}$ and ρ we have adopted the definition of continuous time performance index as given by the expression

$$\mathcal{J} = \int_0^\infty \Big(\begin{bmatrix} x_w & \dot{x}_w & \theta_b & \dot{\theta}_b \end{bmatrix} \bar{C}^T \bar{C} \begin{bmatrix} x_w \\ \dot{x}_w \\ \theta_b \\ \dot{\theta}_b \end{bmatrix} + \rho\, u^2 \Big)\ dt$$

It is basically the same expression found in the `LabBook` except from the fact that ρ multiplies the second factor instead of the first one; it is though equivalent and we used this definition also in the next steps.
This choice resulted in a small value or ρ in order to prioritise error minimization and therefore control performance against control effort. Large values of ρ make sense when the control effort is expensive and we therefore are interested in minimizing it.
Matrix $\bar{C}$ is used to design the controller in such a way to take into account a certain weighting attitude towards different states. We thought that it made sense to keep the matrix as it was provided in the task since no real improvement was obtained by changing it and its values seemed reasonable.

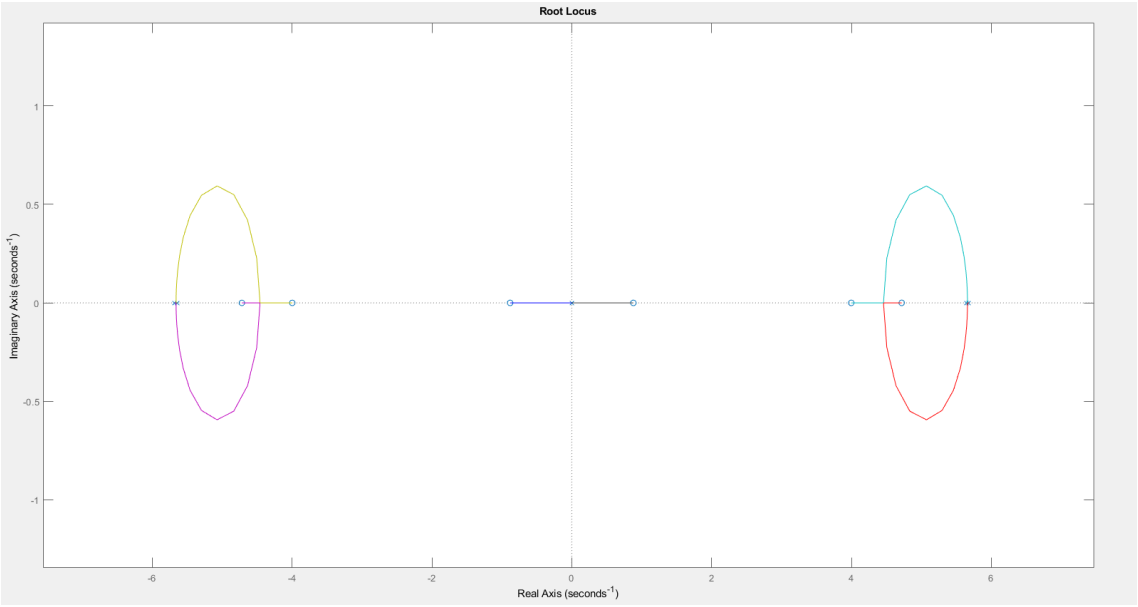


Figure 25: Symmetric Root Locus

Symmetric Root Locus We want to plot the root locus

$$1 + \frac{1}{\rho} G^T(-s)G(s) = 0$$

To use the command `rlocus()` in `MATLAB` we need to represent $G^T(-s)G(s)$ as a state-space system; we do it in the following manner

$$G(s) = C(sI - A)^{-1}B + D$$
$$G^T(-s) = B^T(-sI - A)^{-1}C^T + D^T$$

The overall system in state-space form is described by the following matrices

$$A_{LQR} = \begin{bmatrix} A & 0 \\ -C^TC & -A^T \end{bmatrix}, \; B_{LQR} = \begin{bmatrix} B \\ -C^TD \end{bmatrix}$$

$$C_{LQR} = \begin{bmatrix} D^TC & B^T \end{bmatrix}, \; D_{LQR} = D^TD$$

By using these matrices we can construct a `ss` object in `MATLAB` and then by applying function `rlocus(ss)` we obtained the graph in Figure 25.

LQR Gain The gain matrix K has been obtained by using the command `K = lqr(A, B, Q, rho)` in `MATLAB` where `Q = (C'*C)`. The obtained value is

$$K = [-15.8114 \; -46.4023 \; -78.7156 \; -13.2734]$$

Chosen controller Figure (26) and Figure (27) shows the plots of θ_b and v_m respectively obtained with the chosen controller.

In the upcoming tasks we will use the LQR technique as it is the one that gives the best results while being really control-efficient.

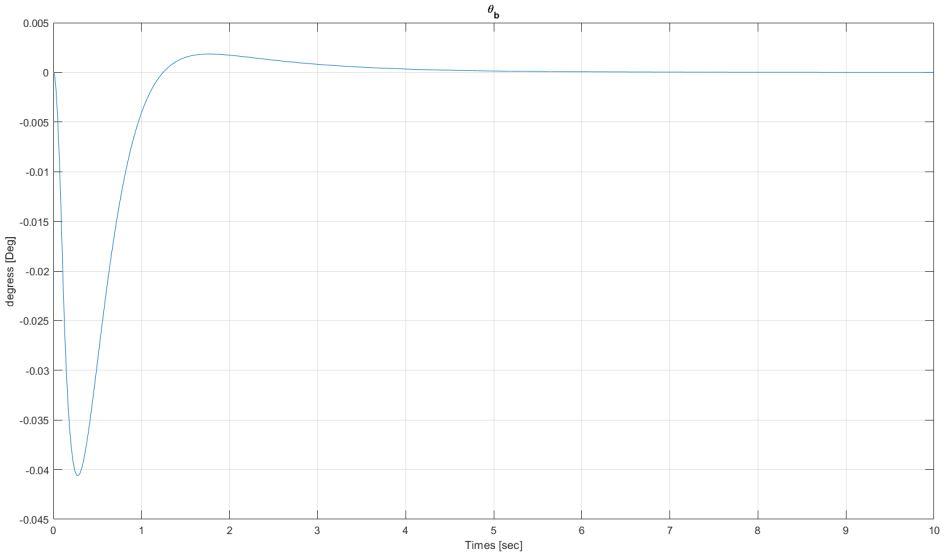


Figure 26: θ_{lqr}

3.3 Task 6.3

Stability is lost for a frequency of

$$f_s \approx 4Hz$$

and the corresponding values of K_d and A_d, B_d, C_d, D_d are

$$Kd = [-2.4116 \ -25.3312 \ -36.7153 \ -5.9585]$$

$$Ad = \begin{bmatrix} 1.0000 & 0.0280 & 0.0219 & 0.0063 \\ 0 & 0.1677 & 0.2126 & 0.0394 \\ 0 & 1.3404 & 2.1963 & 0.3156 \\ 0 & 8.5770 & 11.0961 & 2.'162 \end{bmatrix}$$

$$Bd = \begin{bmatrix} 0.0105 \\ 0.0394 \\ -0.0634 \\ -0.4057 \end{bmatrix}$$

$$Cd = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$

$$Dd = \begin{bmatrix} 0 \end{bmatrix}.$$

3.4 Task 6.4

The sampling frequency at which we lose stability is

$$f_s = 10Hz$$

The corresponding values for matrices K_d, L_d, M_{1d}, ..., M_{7d} and A_d, B_d, C_d and D_d are

$$Kd = \begin{bmatrix} -5.9653 & -30.9321 & -44.4190 & -7.2747 \end{bmatrix}$$

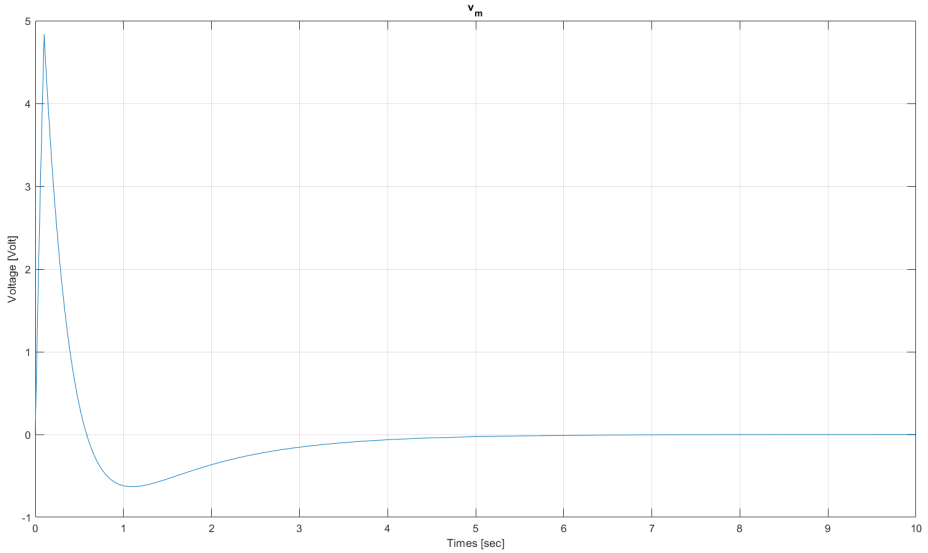


Figure 27: v_m

$$Ld = \begin{bmatrix} 0.8978 & 0.0237 \\ 0.0251 & 0.2701 \\ 0.0791 & 2.0924 \\ 0.7450 & 13.7824 \end{bmatrix}$$

$$Md1 = \begin{bmatrix} 0.0849 & -0.0478 & 0.0033 \\ -0.0364 & 0.0122 & 0.0052 \\ -0.0448 & -0.1158 & 0.1108 \end{bmatrix}, \; Md2 = \begin{bmatrix} -0.2229 \\ 0.0073 \\ -2.3044 \end{bmatrix}, \; Md3 = \begin{bmatrix} -47.6083 \\ -8.3388 \\ -488.3775 \end{bmatrix}$$

$$Md4 = \begin{bmatrix} 0.9616 \\ 0.0761 \\ 2.2198 \end{bmatrix}, \; Md5 = \begin{bmatrix} 47.6083 \\ 8.3388 \\ 488.3775 \end{bmatrix}, \; Md6 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \; Md7 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

$$Ad = \begin{bmatrix} 1.0000 & 0.0096 & 0.0026 & 0.0020 \\ 0 & 0.0923 & 0.0611 & 0.0216 \\ 0 & 0.4095 & 1.1686 & 0.0970 \\ 0 & 4.5751 & 3.4272 & 1.0725 \end{bmatrix}$$

$$Bd = \begin{bmatrix} 0.0043 \\ 0.0429 \\ -0.0194 \\ -0.2164 \end{bmatrix}$$

$$Cd = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$

$$Dd = \begin{bmatrix} 0 \end{bmatrix}.$$

3.5 Task 6.5

The first sampling frequency for which the robot balances is

$$f_s = 50\,Hz$$

Figure (28) shows a graph reporting the L_2 norms of θ_b and x_w against sampling frequency starting from 50 Hz up to 200 Hz by increasing it of 20 Hz each time.

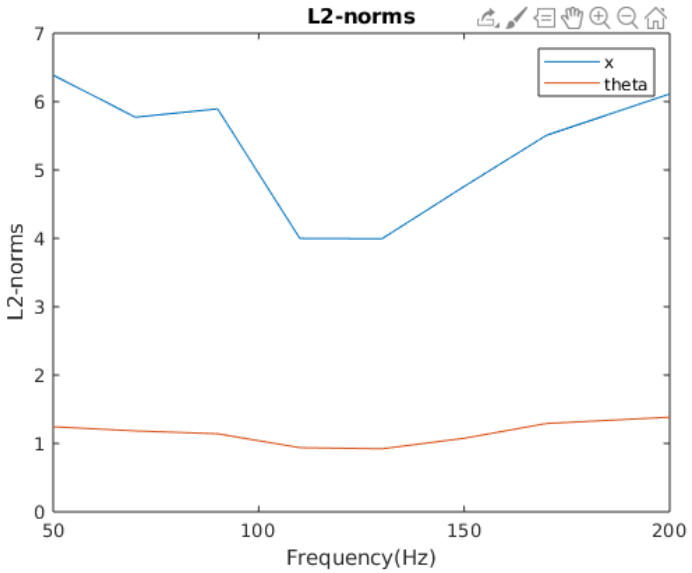


Figure 28: L_2 norms of θ_b and x_w

3.6 Task 6.6

N computation To find the static gain we know that

$$x_\infty = (I - Ad + Bd\,Kd)^{-1} Bd\,N\,y_{ref}$$
$$y_\infty = C(I - Ad + Bd\,Kd)^{-1} Bd\,N\,y_{ref}$$

and since we want to have a unit static gain we obtain the expression

$$C(I - Ad + Bd\,Kd)^{-1} Bd\,N = 1$$

that yields

$$N = -9.7357$$

Reference signals We used three different reference signals to test our robot:

- `signal_1`: up-ramp, stable and down-ramp;
- `signal_2`: up-ramp, stable, up-ramp, stable and down-ramp;
- `signal_3`: up-ramp and stable.

Figures (29), (30) and (31) show the plots of r, x_w, θ_b and v_m for the three reference signals we have defined.

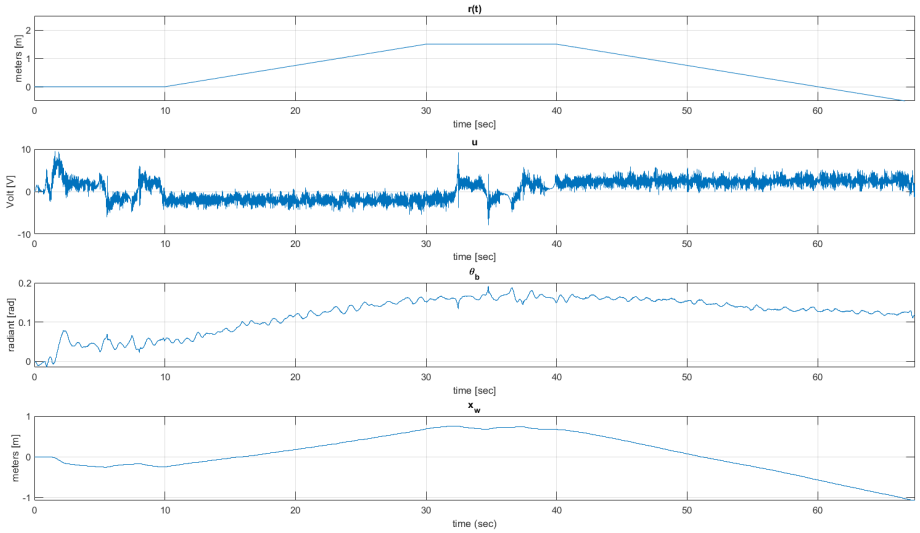


Figure 29: `signal_1`

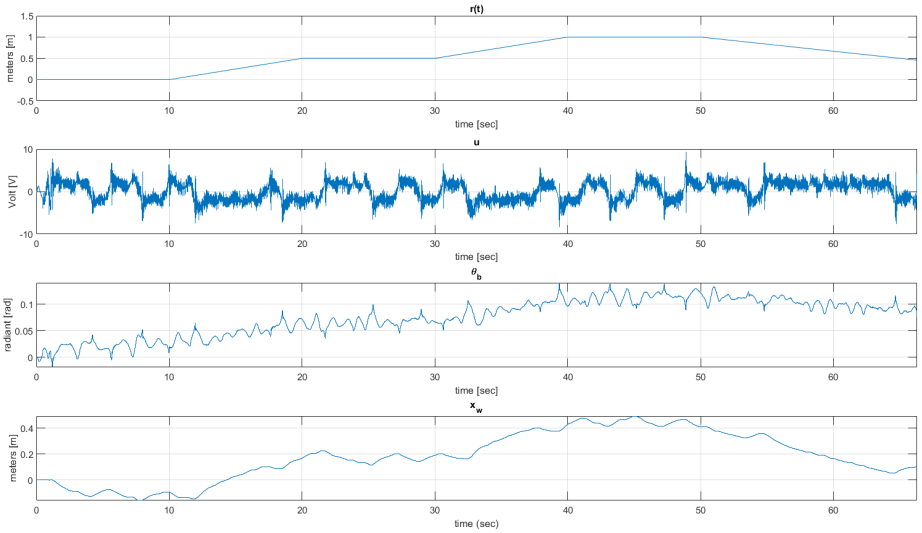


Figure 30: `signal_2`

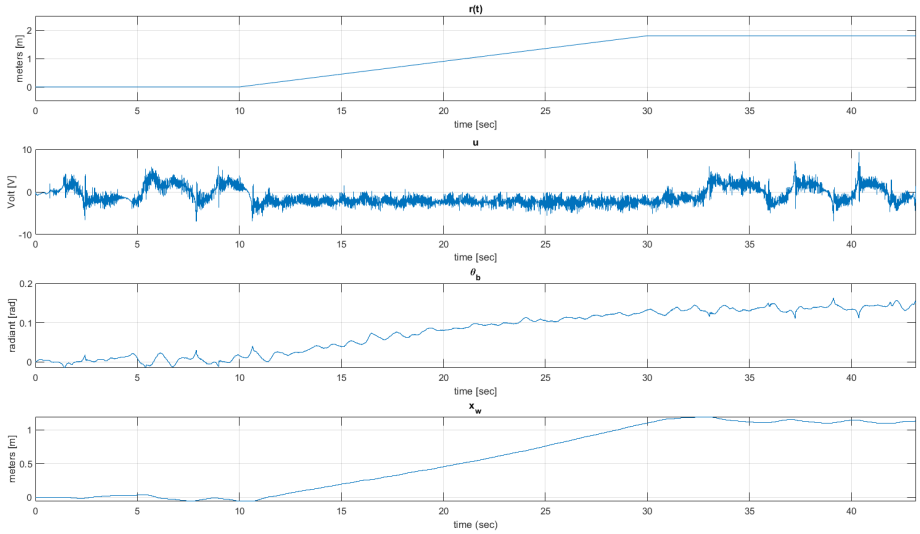


Figure 31: `signal_3`

The robot manages to follow the reference signal quite well, even though increasing the sampling frequency results in better results. Given the definition of best sampling frequency in the LabBook however we decided to keep this one, corresponding to a frequency of

$$f_s = 160\ Hz$$

3.7 Task 6.7

Figure (32) shows the results obtained with the first signal while Figure (33), (34), (35) and (36) report the results obtained applying the second reference signal with different values of r_{max}; the maximum value of r_{max} (i.e. the highest value of r_max that the robot can manage without falling) we have obtained is $r_{max} = 0.23 \frac{m}{s}$.

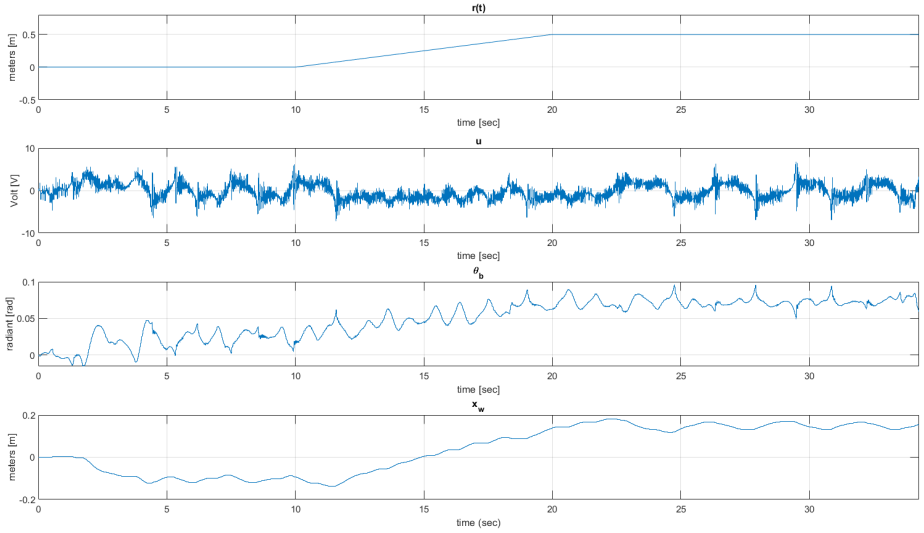


Figure 32: $r_{max} = 0.05 \frac{m}{s}$

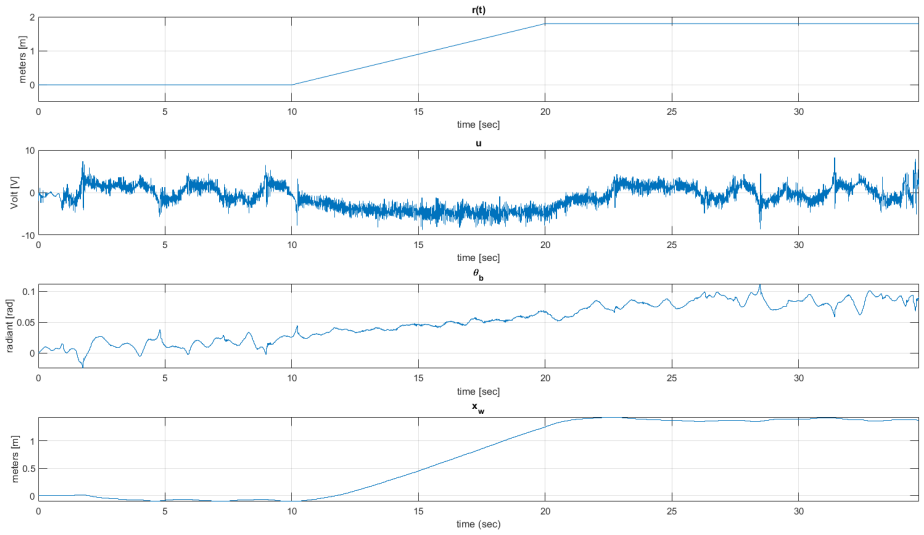


Figure 33: $r_{max} = 0.17 \frac{m}{s}$

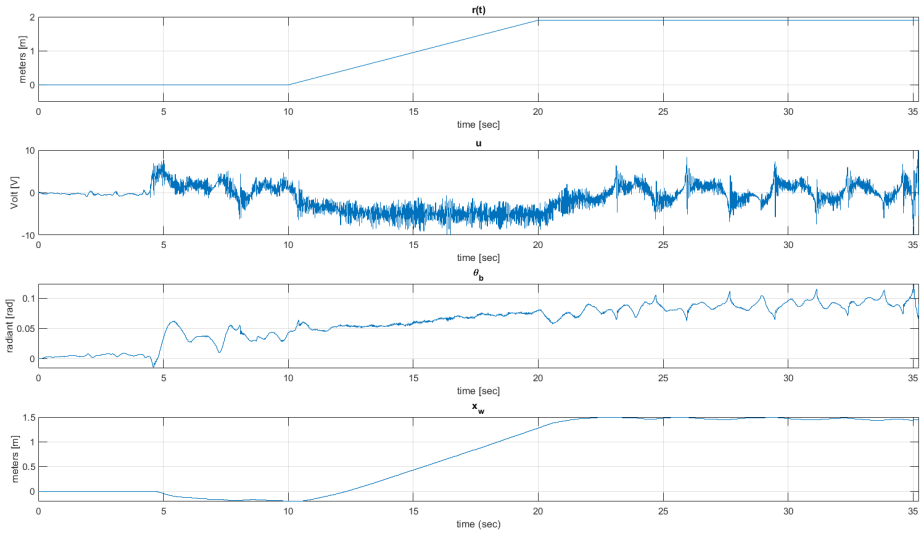


Figure 34: $r_{max} = 0.18\,\frac{m}{s}$

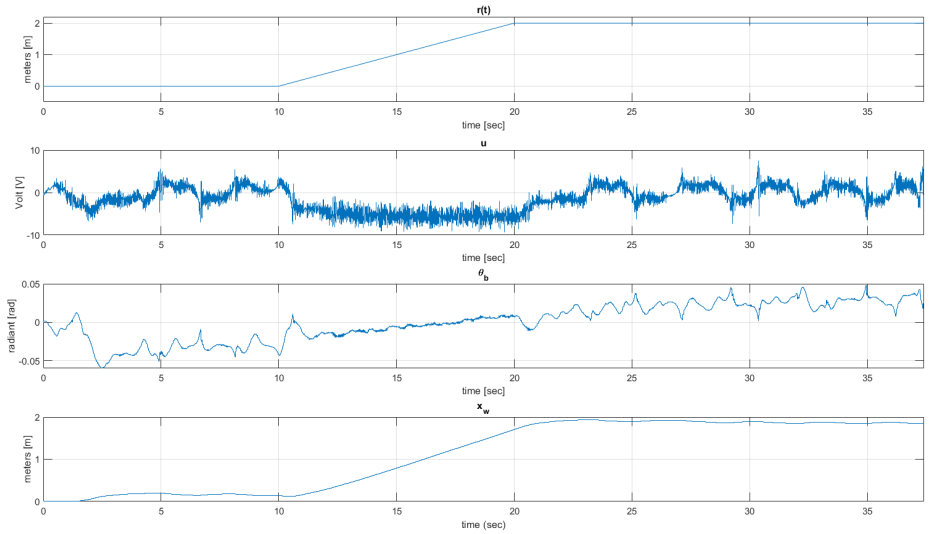


Figure 35: $r_{max} = 0.20\,\frac{m}{s}$

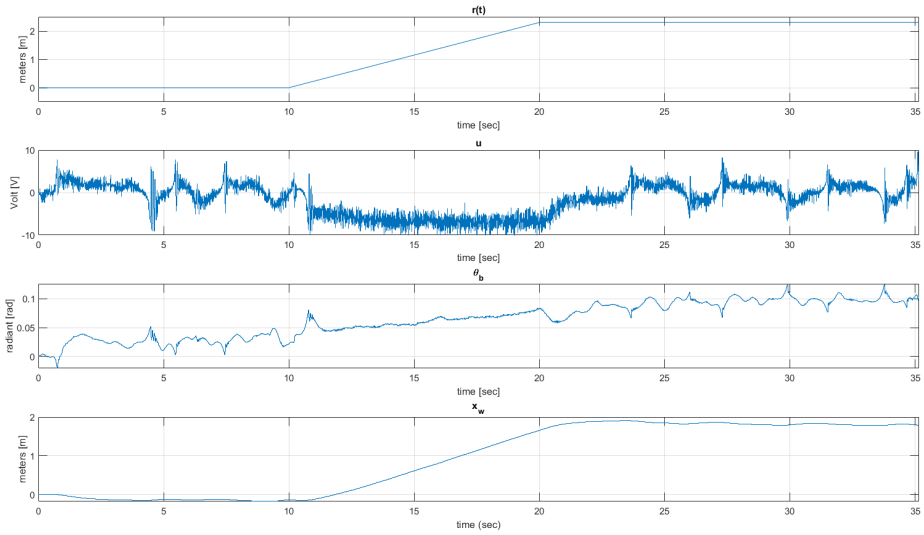


Figure 36: $r_{max} = 0.23 \, \frac{m}{s}$