# Adaptive Control for Cellulo-Mori

Hadrien Sprumont

January 7, 2022

# Contents

# Chapter 1

# Introduction

As part of a collaborative project between the CHILI and the RRL laboratories of EPFL, the Cellulo robot [1] is tasked with carrying a varying number of elements of the Mori robot [2] on top of itself. The combination of the two platforms will provide new opportunities in interactive learning by conjugating the modularity of the Mori and the mobility and precision of the Cellulo.

This prospect comes with a variety of challenges, of which is the necessity for the Cellulo to overcome the added friction imposed by the changing mass coming from the additional elements. The current controllers, a proportional position controller and an open-loop velocity controller, cannot take this additional friction into account, and their velocity tracking performances degrade naturally with any mass increase.

This document will explore the different popular options available in the literature to deal with such adaptation needs and will provide a detailed explanation on the development of a new velocity controller that avoids some defects of the current Cellulo control scheme. The resulting controller will then be implemented in simulation and on a real Cellulo to assess its performance against the current controller.

# Chapter 2

# Adaptive controllers

An adaptive controller is a controller able to adjust its control scheme online when the system it oversees undergoes parameter changes or regime variations. It can do so using little or no prior knowledge of those parameters. This chapter makes a brief overview of some adaptive control methods to find a suitable controller for the Cellulo-Mori robot.

## 2.1 Gain Scheduling

[3, 390-418] Although it can also be classified as a robust controller instead of an adaptive controller due to its heavy emphasis on prior knowledge of a system, gain scheduling is one of the simplest approaches to non-linear control, hence its mention in this chapter. A gain scheduling controller is a linear controller, usually a PID controller, that uses a collection of linear control parameters all tuned offline for different operating points (OP) of a non-linear system. By monitoring operating conditions, generally by looking at one or multiple so-called scheduling variables, the controller can dynamically switch between each stored control set. We could say that a gain scheduling controller makes a piecewise-linear approximation of a non-linear system and optimises in advance a linear controller for each piece. The main problem of gain scheduling design is to find a suitable scheduling variable that is sufficiently correlated to the non-linearities of the system.

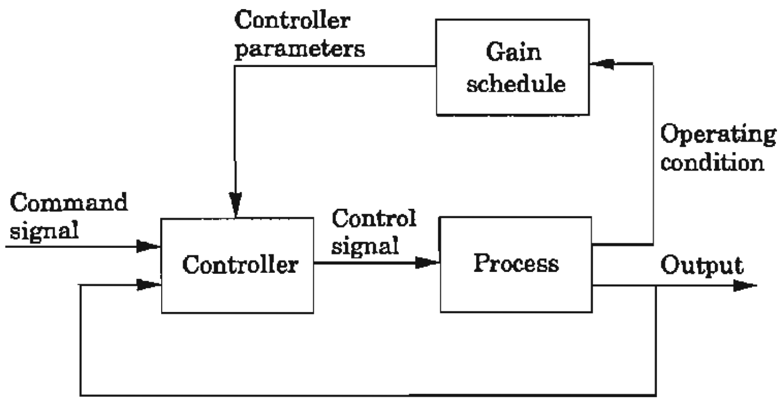


Figure 2.1: Block diagram of a gain scheduling controller. [3, 391]

### Advantages

A gain scheduling controller can respond quickly to process changes and is well suited for systems whose dynamics are well known, especially if they are close to piecewise linear or have discrete operating modes. Making a gain scheduling controller is a simple task that revolves around well-known linearisation and linear controller tuning techniques.

### Disadvantages

Because the method relies on prior knowledge of the system and there is no feedback to compensate a wrong scheduling, unknown disturbances and modelling errors can lead to poor performance or instability on the real system. Finding an appropriate and measurable scheduling variable can be time-consuming or even impossible, and its validation needs to be done by extensive simulation. For a system with numerous non-linearities or requiring very fine segmentation for a piecewise-linear approximation to be accurate, the number of operating points to tune individually grows rapidly, leading to a lengthy and tedious tuning process. The size of the lookup table in such cases can also lead to poor online performance.

## 2.2 Self-tuning regulator

[3, 90-184] [4] [5] [6] Self-tuning regulators (STR) are controllers that update their control parameters based on a recursive online estimation of some parameters of the system. The controller itself is generally a simple linear feedback controller. The mapping between process and controller parameters is developed assuming the estimation equal to the true value of the parameters. The self-tuning regulator can be seen as an attempt to automate the modelling and control design normally done by the control engineer. The estimation of parameters is generally done using recursive least squares, and the control scheme is computed with the minimum-degree pole placement method. In some cases, the parameters of the controller can be estimated directly to get a faster algorithm. In this configuration, the STR can be seen as a Model-reference controller (see section 2.3).

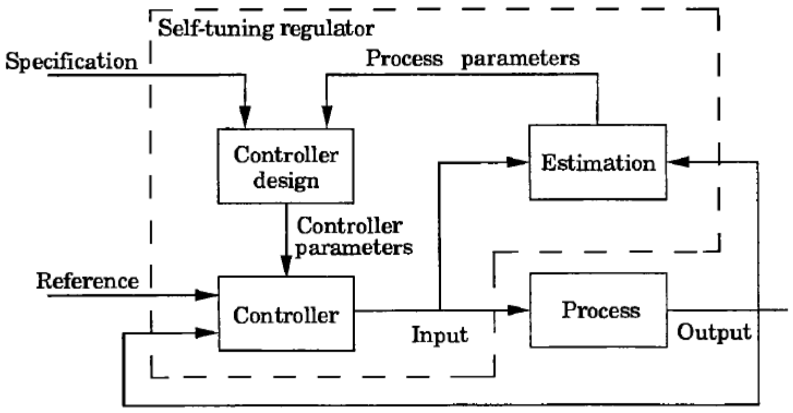


Figure 2.2: Block diagram of a self-tuning regulator [3, 91].

### Advantages

This approach involves standard development steps that are used regularly in non-adaptive control design. The estimation of system parameters can also serve a system identification purpose, as the estimation should converge to the true values. This also implies that STRs are by nature more robust to modelling errors.

### Disadvantages

Although we don't need as much knowledge of the system as for gain scheduling, STRs need a good a *priori* model structure to give satisfying performances and to guarantee the convergence of parameters to their correct values. The stability analysis of STRs can be complicated, due to the sometimes complex mapping between the regulator and estimated parameters, and the presence of singular points in this mapping is not impossible.

## 2.3 Model-reference adaptive control

[3, 185-262] [7] [8] [9] [10] [11] Model-reference adaptive control (MRAC) is a method that aims at adapting the controller parameters such that the closed-loop system behaviour matches the one of a chosen open-loop system of reference. MRAC uses a combination of the input and the error between the two system outputs to adjust the parameters. This can be done using either gradient-descent methods, like the MIT rule, or stability theory, like the Lyapunov rule. To compute these rules, some system parameter estimation may be needed. More broadly, a model-reference adaptive controller can learn unmodelled features, both linear and non-linear, while attempting to compensate them in the control loop as disturbances, which makes it a powerful adaptive control method.

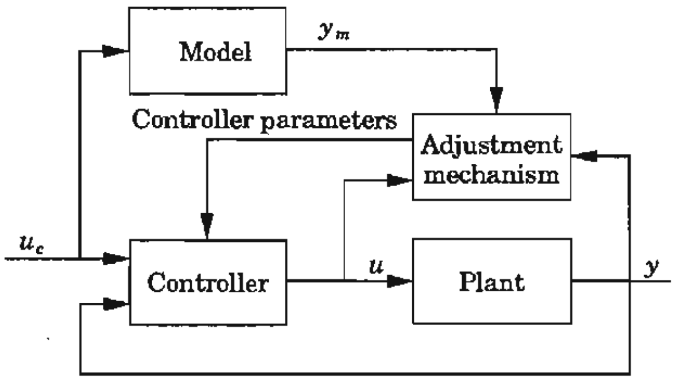


Figure 2.3: Block diagram of a model-reference adaptive system (MRAS) [3, 185].

### Advantages

As for the STRs, MRACs can estimate parameters in their control loop. With the additional linear and non-linear feature estimation, they are a useful tool for system identification. The structure of an MRAC controller can easily be adjusted to fit different disturbance models without reviewing its basic design. There exists multiple design methods, some based on the simple and flexible gradient method, others on the reliable stability theory.

### Disadvantages

Although simple in nature, the computation of parameters when using a large set of features can take some time, which can be a limiting factor. Because the controller only matches the outputs of the system and its reference, the output difference can tend to zero while the estimated parameters do not reach their true value.

## 2.4 Fuzzy control and fuzzy PID

[12][13] The fuzzy logic is a way of translating numerical states and parameters of a control system to non-numerical vague semantic states, from which control rules are written verbally, like: "*IF temperature IS warm THEN fan_speed IS moderate*". The fuzzy controller gains in adaptability due to the vagueness in its construction, enabling it to adapt better to modelling errors and non-linear systems than linear controllers can. A fuzzy version of the typical PID controller that demonstrates these properties is developed in [13]. The fuzzy logic is a tool that can more broadly be applied to a great variety of techniques, including the ones mentioned in the previous sections.

### Advantages

The structure of a fuzzy controller is simple and justifiable, with easy to derive rules from a general understanding of the system. Adding or modifying those rules while tuning or adapting the controller behaviour to specific situations, like sensor failure, for example, is also straightforward. As it does not rely on specific model parameter values, it is also a robust controller. More broadly, the fuzzy logic itself can be applied to a great variety of control schemes to increase their robustness at the cost of some performance loss.

### Disadvantages

Although standardised options are available, fuzzy controllers can be hard to develop due to the high number of design options for fuzzification and defuzzification. Fuzzy controllers can have worse performance than other controllers, taking more time to settle and behaving in an imprecise way. The fuzzy logic framework is not adapted to every system, where sometimes segmenting a variable in different semantic parts is not appropriate.

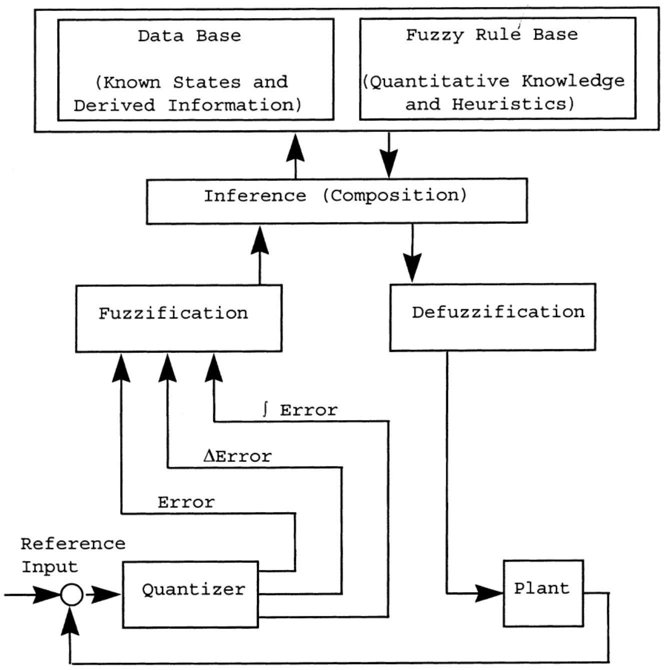


Figure 2.4: Structure of a typical fuzzy logic controller [13, 252].

## 2.5 Choice for the Cellulo-Mori

The main task of the adaptive controller for Cellulo-Mori will be to control its velocity while adapting quickly to both the mass changes created by additional Mori elements and the weight the user imposes on the system when manipulating it. We could expect the total mass of the Cellulo to change in a close to discrete way due to the Mori elements, but the unpredictability of the user impact invalidates this assumption.

In this context, a gain scheduling controller would require tuning of a large amount of OPs to accommodate a maximum of mass values. In addition, its sensitivity to modelling errors is very high in comparison to the other controllers. We will therefore avoid this controller in favour of other methods. A fuzzy controller could be used in combination with mass estimation to produce good results, but the fuzzification of the mass variable may pose a challenge and the controller stability is hard to assess. We will therefore avoid this controller as well in favour of an STR or an MRAC.

As stated previously, under certain circumstances, the two approaches are one and the same. Both those options could produce satisfying results, but the possibility to easily adjust the disturbance model of the system using the MRAC formulation makes it a useful tool when working with unknown user input. MRAC also comes with built-in stability analysis when basing its design on the Lyapunov rule. All these factors oriented our final choice to an MRAC controller.

# Chapter 3

# Controller design

In this chapter, we will develop an adaptive controller for the Cellulo-Mori. The objective is to control the velocity of the robot in the $x$, $y$, and $\theta$ dimensions, adapting to mass changes and other parameter variations. The controller of choice is a Model Reference Adaptive Controller. It will be used in conjunction with feedback linearisation to produce an adaptive and decoupled control scheme for each dimension.

## 3.1 Feedback linearisation

[14] [15] Feedback linearisation is a method to linearise and decouple non-linear MIMO systems using knowledge of the plant and its dynamics. It can only be used on invertible systems with the same number of inputs and outputs, and if key internal states can be measured or estimated. Thankfully, the Cellulo is a three input-three output plant and we can measure its absolute position by reading the Anoto pattern [16] below it, which gives us access to precise state measurement. Not only that, but its dynamics are well known and have been developed in Appendix B.

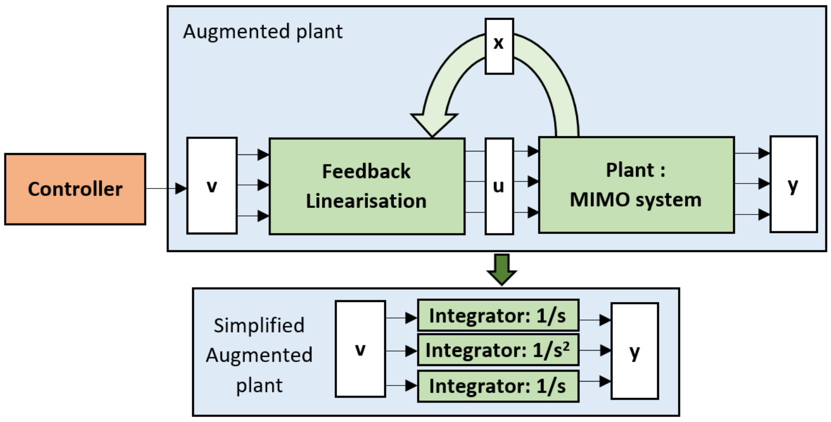


Figure 3.1: Illustration of the feedback linearisation architecture and its intended effect on an **unknown** MIMO plant with 3 inputs and 3 outputs (The feedback control loop is omitted here).

Feedback linearisation comes in the form of a virtual system block in front of the plant, as illustrated in Figure 3.1. The combined or augmented system is then reduced to a pure decoupled integrator of an order that depends on the system. To build the feedback linearisation system we need the Cellulo's corresponding differential equations, our MIMO plant (See section B.2 for a description of the matrices and vectors):

$$\begin{gathered}\ddot{q} = M^{-1}(-C' \cdot \dot{q} - D' \cdot \vec{c} + Q' \cdot u) \\ Y = q = \begin{pmatrix} x & y & \theta \end{pmatrix}^T\end{gathered} \tag{3.1}$$

Here $Y$, the output of the system, is the measured position and $u$ is the input vector of motor voltages. The feedback linearisation equations are basically the inverse of the plant equations. One way to calculate those inverse dynamics is to recursively take the time-derivative of the output $Y$

until the input $u$ appears in the expression:

$$\begin{aligned} Y &= q \\ \dot{Y} &= \dot{q} \\ \ddot{Y} = \ddot{q} &= M^{-1}(-C' \cdot \dot{q} - D' \cdot \vec{c} + Q' \cdot u) \end{aligned} \tag{3.2}$$

Choosing the new input of the augmented plant, $v$, as the second time-derivative of the output, we can now solve the expression to get $u$, the output of the feedback linearisation:

$$\ddot{Y} = v \quad \longrightarrow \quad u = (Q')^{-1}(M \cdot v + C' \cdot \dot{q} + D' \cdot c) \tag{3.3}$$

Assuming we have a good estimation of the plant parameters, the simplified augmented plant equation is now:

$$\ddot{q} = M^{-1}(Q'(Q')^{-1}(M \cdot v + C' \cdot \dot{q} + D' \cdot \vec{c}) - C' \cdot \dot{q} - D' \cdot \vec{c}) = v \tag{3.4}$$

Using inverse dynamics, we are hence able to make the plant behave as a pure decoupled double integrator. This is only true however if the real system is correctly modelled and the parameters are close to their true values. A typical approach would be to do a series of experiments to estimate those values and calibrate the controller, but the adaptive component of the controller we want to develop here gives us the opportunity to perform these steps automatically online.

To allow the controller to estimate the $C'$ and $D'$ parameters properly, it is better to implement these terms in the controller rather than the feedback linearisation. This can be done by revising the expression of the feedback linearisation output $u$ (3.3) and treating the $C'$ and $D'$ terms as a disturbance $f(X)$ to be estimated inside the controller. The new system of equations becomes:

$$\begin{aligned} u &= (\tilde{Q}')^{-1}\tilde{M} \cdot v \\ \ddot{q} = M^{-1}(Q'(\tilde{Q}')^{-1}\tilde{M} \cdot v - C' \cdot \dot{q} - D' \cdot \vec{c}) &= M^{-1}Q'(\tilde{Q}')^{-1}\tilde{M} \cdot (v - f(X)) \end{aligned} \tag{3.5}$$

Where the matrices marked by ˜ are the estimated values and $X$ is the system state. By looking at the forms of $M$ and $Q'$ in (B.26) and assuming the errors will only come from the parameters rather than the geometry we can state:

$$\tilde{M} = \begin{pmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{pmatrix} \cdot M \quad \textit{and} \quad \tilde{Q}' = \begin{pmatrix} q_1 & 0 & 0 \\ 0 & q_2 & 0 \\ 0 & 0 & q_3 \end{pmatrix} \cdot Q'$$
$$\Rightarrow M^{-1}Q'(\tilde{Q}')^{-1}\tilde{M} = \begin{pmatrix} \frac{m_1}{q_1} & 0 & 0 \\ 0 & \frac{m_2}{q_2} & 0 \\ 0 & 0 & \frac{m_3}{q_3} \end{pmatrix} = \begin{pmatrix} b_x & 0 & 0 \\ 0 & b_y & 0 \\ 0 & 0 & b_\theta \end{pmatrix} = B \tag{3.6}$$

Where $m_i$ and $q_i$ are unknown positive real constants close to 1. This allows us, using the unknown diagonal matrix $B$, to rewrite the final augmented system to control as:

$$\ddot{q} = B \cdot (v - f(X)) \tag{3.7}$$

## 3.2 MRAC controller

Now that the system has been simplified, we can develop a controller using the augmented plant. In this section, we will develop a MRAC controller using Lyapunov theory. The steps are as follows:

- Finding a stable reference model.
- Choosing a control law.
- Calculating the error dynamics between the reference and the plant models.
- Finding a suitable Lyapunov function.
- Calculating the adjustment mechanisms for the control parameters.

### 3.2.1 Reference model

The reference model is the ideal model that we would like the entire closed-loop system to behave like, including the controller. The MRAC controller will try to adjust its internal parameters to match this reference model's behaviour as close as possible. To find an appropriate reference model, we have to choose first a reference open-loop system, a reference ideal plant in other words. We pick for this the ideal state of the augmented system to control (3.7), that is with perfect modelisation and no disturbance:

$$\ddot{q} = B \cdot (v - f(X)) \quad \longrightarrow \quad \ddot{q} = v \tag{3.8}$$

We can now design the stable reference model by closing the loop on this virtual plant with state feedback and by adding a feed-forward term:

$$\ddot{q}_m = -K_{pm} \cdot \dot{q}_m + K_{rm} \cdot r \tag{3.9}$$

Where $K_{pm}$ and $K_{rm}$ are diagonal matrices and $r$ is the reference velocity we want the system to follow. Here and in the rest of this section, the index $_m$ designs the reference model states and variables.

### 3.2.2 Control law

To behave similarly as the reference model, the control law of the real system must have the same form as the reference plant has, that is with a proportional feedback term and a feed-forward term. In addition to that, the control law must have a term to estimate and eliminate the disturbance $f(X)$. Although non-linear, this disturbance is linearly separable for each dimension of $\dot{q}$, meaning it can be expressed as the product of a scalar gain vector $\hat{\theta}^T$ and a vector of non-linear terms $\Phi(X)$ that can be obtained accurately.

The expressions of $\Phi(X)$ must compensate the disturbance as expressed in (3.5), so we get the following:

$$\begin{pmatrix} \hat{\theta}_x^T \cdot \Phi_x(X) \\ \hat{\theta}_y^T \cdot \Phi_y(X) \\ \hat{\theta}_\theta^T \cdot \Phi_\theta(X) \end{pmatrix} = f(X) = (Q'(\tilde{Q}')^{-1}\tilde{M})^{-1}(C' \cdot \dot{q} + D' \cdot \vec{c}) \tag{3.10}$$

$$\Rightarrow \begin{cases} \Phi_x(X) = \begin{pmatrix} \dot{x} & -cos(\theta) \cdot c_1 & cos(\theta - \frac{\pi}{3}) \cdot c_2 & cos(\theta + \frac{\pi}{3}) \cdot c_3 \end{pmatrix}^T \\ \Phi_y(X) = \begin{pmatrix} \dot{y} & -sin(\theta) \cdot c_1 & -cos(\theta + \frac{\pi}{6}) \cdot c_2 & sin(\theta + \frac{\pi}{3}) \cdot c_3 \end{pmatrix}^T \\ \Phi_\theta(X) = \begin{pmatrix} \dot{\theta} & c_1 & c_2 & c_3 \end{pmatrix}^T \end{cases} \tag{3.11}$$

We can hence write the final control law with $k_{pi}$ and $k_{ri}$ ($i \in \{x, y, \theta\}$) as scalar feedback and feed-forward gains:

$$v = \begin{pmatrix} -\hat{k_{px}} \cdot \dot{x} + \hat{k_{rx}} \cdot r_x + \hat{\theta}_x^T \cdot \Phi_x(X) \\ -\hat{k_{py}} \cdot \dot{y} + \hat{k_{ry}} \cdot r_y + \hat{\theta}_y^T \cdot \Phi_y(X) \\ -\hat{k_{p\theta}} \cdot \dot{\theta} + \hat{k_{r\theta}} \cdot r_\theta + \hat{\theta}_\theta^T \cdot \Phi_\theta(X) \end{pmatrix} \tag{3.12}$$

Where all the terms marked with $\hat{\cdot}$ are the controller parameters that will be updated at run time. If the parameters of the MRAC perfectly represented the disturbance of the plant and compensated all the modelisation errors, the real closed-loop system and the reference model (3.9) would be exactly the same:

$$\ddot{q} = B \cdot (v_{perfect} - f(X)) \quad \longrightarrow \quad \ddot{q} = -K_{pm} \cdot \dot{q} + K_{rm} \cdot r \tag{3.13}$$

### 3.2.3 Error dynamics

Before deriving the error dynamics between the reference and the plant models, it is useful to express the matching conditions of the MRAC. Those are the expressions of the theoretical optimal values, expressed with $\bar{\cdot}$, of the estimated parameters $\hat{k_p}$, $\hat{k_r}$, and $\hat{\theta}^T$ for each dimension. Those are the values that would theoretically result in Equation 3.13:

$$b \cdot \bar{k_p} = k_{pm} \qquad b \cdot \bar{k_r} = k_{rm} \qquad \bar{\theta}^T \cdot \Phi(X) = f(X) \tag{3.14}$$

Where $b$ is the appropriate element of the diagonal of $B$. Using (3.7), (3.9), and (3.12), the general form for a single coordinate of the error dynamics can be written as:

$$e = s - s_m, \quad s \in \{\dot{x},\ \dot{y},\ \dot{\theta}\} \tag{3.15}$$

$$\begin{aligned}
\dot{e} &= \dot{s} - \dot{s_m} \\
&= b \cdot (-\hat{k_p} \cdot s + \hat{k_r} \cdot r + \hat{\theta}^T \cdot \Phi(X) - \bar{\theta}^T \cdot \Phi(X)) - (-k_{pm} \cdot s_m + k_{rm} \cdot r) \\
&= -k_{pm} \cdot (s - s_m) + (k_{pm} - b \cdot \hat{k_p}) \cdot s - (k_{rm} - b \cdot \hat{k_r}) \cdot r - b \cdot (\bar{\theta}^T - \hat{\theta}^T) \cdot \Phi(X) \\
&= -k_{pm} \cdot e + b \cdot \tilde{k_p} \cdot s - b \cdot \tilde{k_r} \cdot r - b \cdot \tilde{\theta}^T \cdot \Phi(X)
\end{aligned} \tag{3.16}$$

Where the $\tilde{\cdot}$ is the expression of the difference between the estimated and the optimal value of each parameter.

### 3.2.4 Lyapunov function

A Lyapunov function is a continuous, positive definite, scalar function that has a single equilibrium point where its gradient is null and is, informally, monotonically decreasing towards this point everywhere else. It can be used to prove a system's stability when its evolution over time, through the Lyapunov function, always converges to the function's equilibrium point. That is, the time-derivative of the Lyapunov function value linked to this system is always strictly negative. Here we want to design and use such a Lyapunov function to prove convergence over time between the real controlled system and the reference model.

There is no defined method to design a Lyapunov function, but knowing that we want the error dynamics to appear at some point we can pick a quadratic function with a similar structure. The quadratic nature of (3.17) guarantees the properties of a Lyapunov function and, should its value reach zero, the estimated parameters would have reached their true system values and perfect model tracking would be achieved. Note that we don't know most of the variables that appear in this function, including $b$. It is not needed as it is written only to design the parameter update laws and prove stability, and will never be explicitly computed.

$$V(e, \tilde{k_p}, \tilde{k_r}, \tilde{\theta}) = \frac{1}{2}e^2 + \frac{|b|}{2\gamma_p}\tilde{k_p}^2 + \frac{|b|}{2\gamma_r}\tilde{k_r}^2 + \frac{|b|}{2}\tilde{\theta}^T \cdot \Gamma_\theta^{-1} \cdot \tilde{\theta} \tag{3.17}$$

The parameters $\gamma_p$, $\gamma_r$, and $\Gamma_\theta$ are the learning rates of each parameters. Their presence will be explained once the updates laws are described (3.2.5). The derivative of the function is computed to examine its evolution over time. The ideal case would be that it always decreases, meaning that the controlled system is stable and converges to the reference model.

$$\begin{aligned}
\dot{V} &= e \cdot \dot{e} - \frac{|b|}{\gamma_p}\tilde{k_p} \cdot \dot{\hat{k_p}} - \frac{|b|}{\gamma_r}\tilde{k_r} \cdot \dot{\hat{k_r}} - |b|\tilde{\theta}^T \cdot \Gamma_\theta^{-1} \cdot \dot{\hat{\theta}} \\
&= -k_{pm} \cdot e^2 - \tilde{k_p}(\frac{|b|}{\gamma_p}\dot{\hat{k_p}} - b \cdot e \cdot s) - \tilde{k_r}(\frac{|b|}{\gamma_r}\dot{\hat{k_r}} + b \cdot e \cdot r) - \tilde{\theta}^T(|b| \cdot \Gamma_\theta^{-1} \cdot \dot{\hat{\theta}} + b \cdot e \cdot \Phi(X))
\end{aligned} \tag{3.18}$$

### 3.2.5 Parameter update

We still need to design the update laws for our parameter estimates. We would like that this update always makes the system closer to the reference model, meaning that our Lyapunov function always decreases over time. By looking at (3.18), we can find expressions for $\dot{\hat{k_p}}$, $\dot{\hat{k_r}}$, and $\dot{\hat{\theta}}$ that reduce the derivative of (3.17) to a negative function of the system error:

$$\left.\begin{aligned}
\dot{\hat{k_p}} &= sign(b) \cdot e \cdot \gamma_p \cdot s \\
\dot{\hat{k_r}} &= -sign(b) \cdot e \cdot \gamma_r \cdot r \\
\dot{\hat{\theta}} &= -sign(b) \cdot e \cdot \Gamma_\theta \cdot \Phi(X)
\end{aligned}\right\} \quad \Rightarrow \quad \dot{V} = -k_{pm} \cdot e^2 \tag{3.19}$$

Here we see that thanks to a careful choice of the update laws we now have a negative value for our Lyapunov function derivative. This means that over time the Lyapunov function value decreases towards the equilibrium which, as stated earlier, is the perfect model tracking point. We also see that the only a-priori information of the system we need to compute the parameter update is the sign of $b$, which as defined in (3.6) should be positive.

# Chapter 4

# Simulation results

To assess the performance of our controller, it is implemented on Simulink. There it is set to control an ideal Cellulo plant by receiving velocity commands. A first round of calibration is made to let the control parameters adapt to the system. This can be done simply by letting the system run for some time with a periodic pulse input for each controlled dimension. Because the MRAC controller adapts to the specific command it is given over time and not forcefully the system itself, it is best to make those different input signals out of phase and let the signal up time be long enough for the system to stabilise. The objective is to let the controller experience a maximum of transition cases. The inputs used for testing are described in Table 4.1.

| Coordinate | Amplitude | Offset | Period | Pulse width | Phase delay |
|---|---|---|---|---|---|
| x | 90 [mm/s] | 10 [mm/s] | 9 [s] | 50% | 0 [s] |
| y | 100 [mm/s] | 0 [mm/s] | 10 [s] | 50% | 2 [s] |
| $\theta$ | 1 [rad/s] | 0 [rad/s] | 8 [s] | 50% | 1 [s] |

Table 4.1: Pulse input signals for calibrating and testing.

The offset is placed to counter the fact that Simulink slows down considerably the simulation progress when their is no input at all. The repo containing the matlab and Simulink files can be found at `https://c4science.ch/diffusion/11873/` [17].

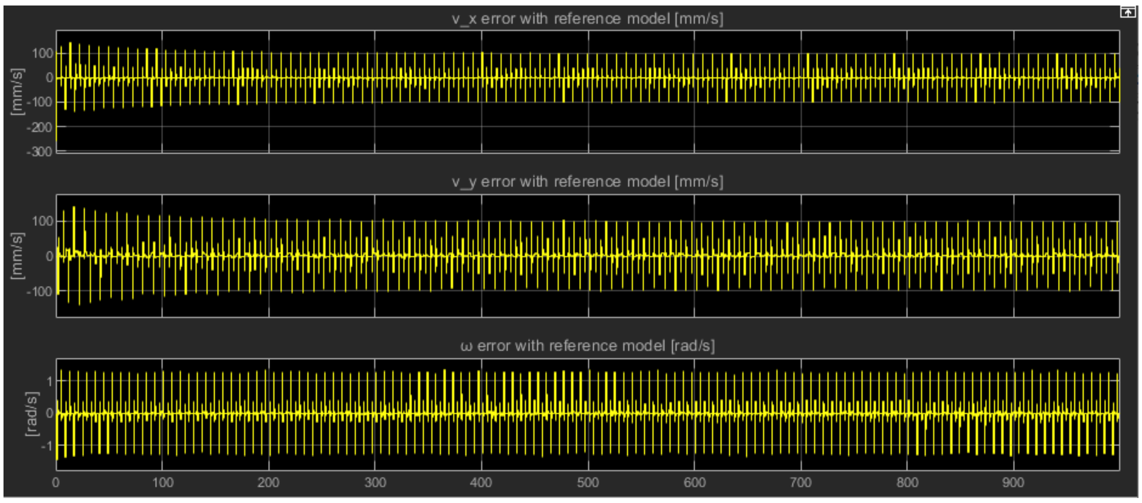
## 4.1 Discretisation

The stability criteria exposed in (3.19) are expressed in continuous time, but the system runs in the discrete world. The sensing and control time period of the Cellulo is around 0.02 seconds, and the controller performance changes drastically with this parameter. The control parameters have different optimal values depending on the update frequency, and experiments show that the greater the frequency the better the adaptive response gets, both in speed and in model tracking efficacy, and that a slow control loop may result in instability. Figure 4.1 illustrates those performance differences when using the calibration input from Table 4.1. Trying to decrease the update frequency further resulted in exploding parameters that stopped the simulation progress.

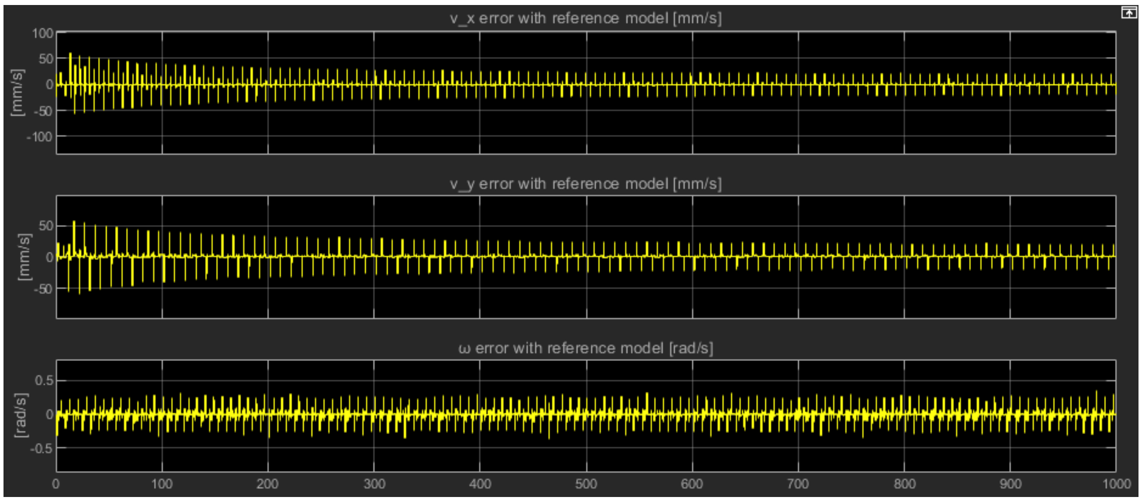
## 4.2 Adaptation

To see the adaptation capabilities of the new controller, we can take measures of the step response of the system with a plant whose parameters have been randomly altered by ±25% of their nominal values and whose mass has been increased to ten times the original value. Figure 4.2 shows the gradual improvement of the step response for a 100 [mm/s] tracking input in $x$ and $y$.

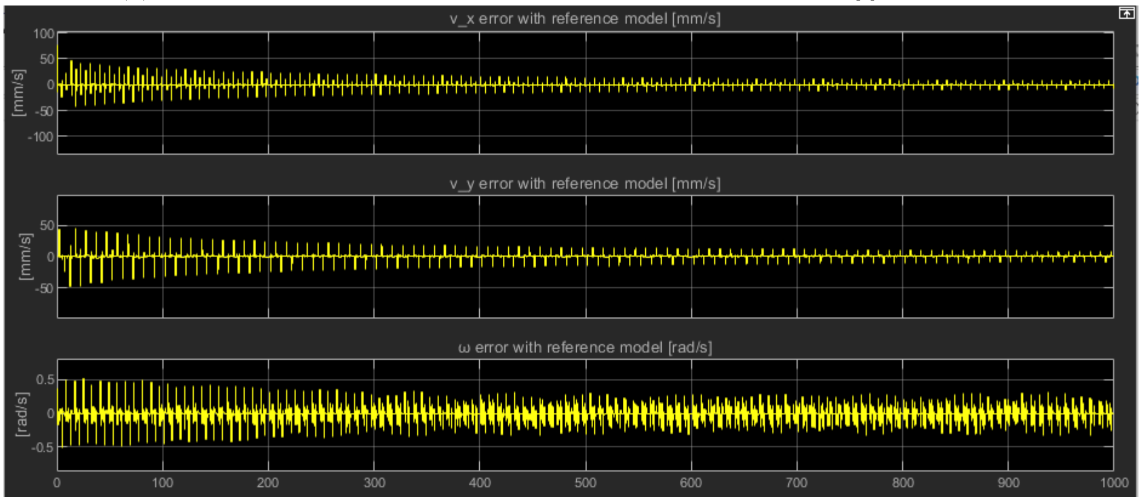
Due to the discretisation and delays, the controller is unable to reach exact model tracking even in this ideal environment. The model tracking error is expected to increase drastically when deployed on a real Cellulo. Despite this, the controller is able to track efficiently the desired velocity when the input is constant and demonstrates its ability to adapt over time to reduce this error.



(a) Error between reference model and system response with a 0.07 [s] update time.

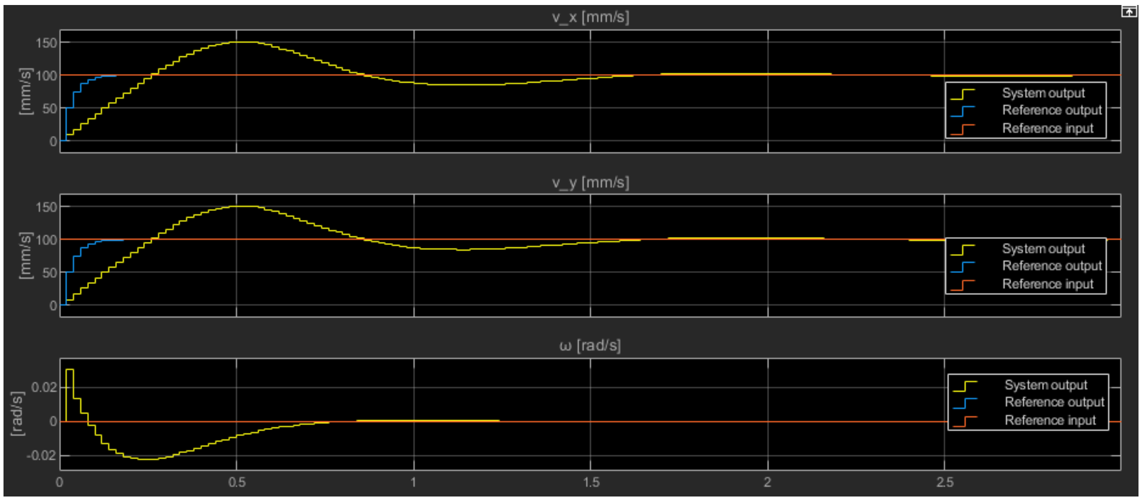


(b) Error between reference model and system response with a 0.02 [s] update time.

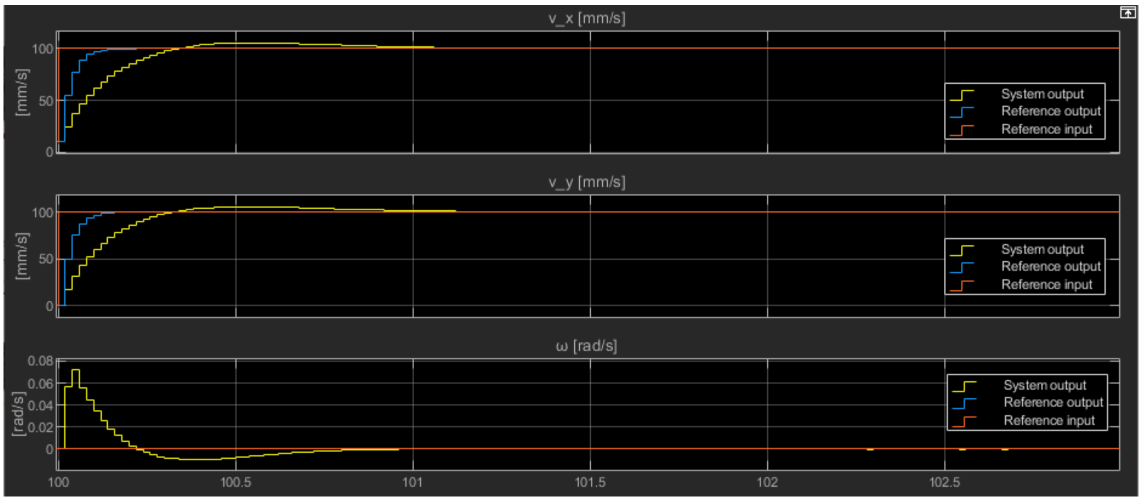


(c) Error between reference model and system response with a 0.002 [s] update time.

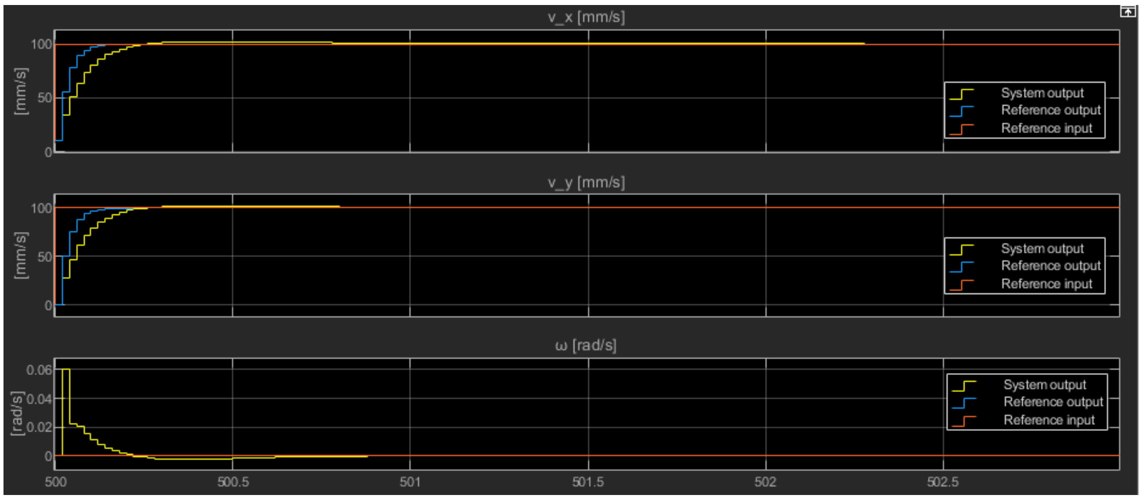
Figure 4.1: Model tracking error comparison at various update times of 0.07, 0.02, and 0.002 [s]. We see a better error reduction with higher update frequencies.



(a) Velocity tracking performance before adapting.



(b) Velocity tracking performance after adapting to oscillations for around 100 seconds.



(c) Velocity tracking performance after adapting to oscillations for around 500 seconds.

Figure 4.2: Adaptation over time to an alternate plant with random parameters.

Chapter 5

# Implementation on the real Cellulo

## 5.1 Deployment

The adaptive controller was exported on a real Cellulo unit using the C files available at `https://c4science.ch/diffusion/11873/` [17]. It had to undergo a new set of calibration to adapt to this new environment but the controller structure is similar to the Simulink model. Unfortunately, due to time constraints, proper calibration commands were not implemented and the procedure was carried out using the default calibration test of the Cellulo for all experiments. This test involves sequential rotations and $y$ movements only, calculated with position control. This results in a lack of calibration for the $x$ coordinate, or for crossed rotation and movements, and the velocity commands given to the controller are not fixed, which would be preferable for an ideal calibration.

Despite this drawback, the controller was successfully deployed and calibrated. It was able to track the velocity commands and follow the predetermined path. Qualitatively, the resulting motion is less precise than the original controller, with more lateral deviations, but this undesired behaviour reduced considerably in magnitude as time advanced so we can expect it to disappear with a proper calibration protocol and a long enough calibration time.

### 5.1.1 Adjustments

Here are some adjustments that were made to the controller after it was deployed. They were made to compensate some undesired effects observed while the robot was tracking velocity commands to reach specific positions on the playing field.

#### Oscillations

The rotations in particular were sometimes subject to unstable oscillations when close to the tracked position, as the controller was trying to compensate the swings of the body amplified by the gaps in the Cellulo motor assembly. This was easily fixed by smoothing the angular speed commands before passing them as the reference input:

$$\dot{\theta}_{r,in}[t] = 0.8 \cdot \dot{\theta}_{r,in}[t-1] + 0.2 \cdot \dot{\theta}_{cmd}[t] \tag{5.1}$$

The smoothed versions of the speed estimates were also used to attenuate large oscillations in the measures and thus the control signals:

$$\begin{aligned} \dot{x}_{in}[t] &= 0.5 \cdot \dot{x}_{in}[t-1] + 0.5 \cdot \dot{x}_{meas}[t] \\ \dot{y}_{in}[t] &= 0.5 \cdot \dot{y}_{in}[t-1] + 0.5 \cdot \dot{y}_{meas}[t] \\ \dot{\theta}_{in}[t] &= 0.8 \cdot \dot{\theta}_{in}[t-1] + 0.2 \cdot \dot{\theta}_{meas}[t] \end{aligned} \tag{5.2}$$

#### Saturation

One element that is not taken into account in the Simulink model is saturation. The motors of the Cellulo can take a voltage input in the $\pm 6$ [v] range [18]. To avoid integration windup on the controller parameters, all integration has to be stopped while the motors are close to the saturation limit. This condition is easy to monitor and the controller was adjusted accordingly.

## 5.2 Weighted comparison

To assess the performance of the adaptive controller compared to the original one, a series of tests should be made to highlight its adaptive capacity with increasing loads. The objective was to make several runs with both controllers, with different weights, and on different robots.

### 5.2.1 Cellulo design limitations

Unfortunately, the first weighted tests on the real Cellulo revealed less conclusive than expected. Indeed, the added weight caused friction to increase drastically in the transmission between the motors and the wheels, resulting in static friction barriers that occasionally prevented the motors from actuating the robot. Small bumps and ripples on the paper caused the robot to stop moving entirely at low speeds, which required manually giving the robot a little bump to get it moving again. This phenomenon was further amplified on robots with a different coating on the wheels, which made the use of multiple robots impossible as they comprised the large majority of the available robots at the time.

The minimum added weight at which this phenomenon developed was around 100 [gr] for the normal wheel coating and 50 [gr] for the alternative version. This meant that the tests had to be done on a single Cellulo and that the robot would need to be regularly manually interacted with, which makes for a poor representative value of the capabilities of the tested controllers.

This issue revealed however an unexpected quality of the adaptive controller, which will be discussed further in subsection 5.2.3.

### 5.2.2 Experiments

Despite this issue and the time lost trying to find a workaround, the comparison test was finally conducted. A set of labelled weights of 50, 100, and 150 [gr] was used for two trials with each controller and each weight. The 150 [gr] weight caused a lot of issues with motion, as discussed in the preceding subsection, and was thus tested only once per controller.

The control sequence used was once again the default calibration procedure as it was at the time the only working command that could log back the Cellulo's position. As it is the same sequence on which the adaptive controller was "trained", it is expected to behave optimally right from the start of the test.

### 5.2.3 Results

The average velocity of each experiment is shown in Table 5.1 below.

| Additional weight | 0 [gr] | 50 [gr] | 100 [gr] | 150 [gr] |
|---|---|---|---|---|
| Adaptive controller | 4.05 | 4.54 | 3.80 | 4.12 |
| Original controller | 3.29 | 3.25 | 3.78 | 2.95 |

(a) Average x-speed of both controllers (mm/s).

| Additional weight | 0 [gr] | 50 [gr] | 100 [gr] | 150 [gr] |
|---|---|---|---|---|
| Adaptive controller | 70.21 | 62.79 | 46.68 | 23.45 |
| Original controller | 66.69 | 52.14 | 37.55 | 28.30 |

(b) Average y-speed of both controllers (mm/s).

| Additional weight | 0 [gr] | 50 [gr] | 100 [gr] | 150 [gr] |
|---|---|---|---|---|
| Adaptive controller | 9.42 | 8.64 | 8.12 | 7.87 |
| Original controller | 8.33 | 8.07 | 7.99 | 8.89 |

(c) Average $\theta$-speed of both controllers (rad/s).

Table 5.1: Average velocity of both controllers. They were calculated on two trials per weight and per controller (except for the 150 [gr] weight which was tested once per controller).

We see an overall better performance from the adaptive controller with a faster mean speed in almost every situation and less speed reduction with the increasing weight. It reveals a better ability to follow hig-velocity commands and good adaptation. Qualitatively, the adaptive controller had a better response to the increasing weight than the default controller, which is expected. The interesting result is that not only it did react by scaling the motor commands to compensate for the overall friction, but it was also able to recover from some undesired stalling. Indeed, those stops are detected as model tracking errors and compensated by the controller, which responds by increasing the commands until the robot gets moving again.

These results are overall satisfying despite the non-ideal conditions of the experiments and show promising potential for deployment in a changing environment. The controller was able to adapt to increasing weights and overcome some drastic friction changes that stopped the default controller from moving completely.

A more thorough comparison done in the future could show the true magnitude of the advantages provided by this adaptive controller.

# Chapter 6

# Conclusion

The purpose of this project was to develop an adaptive controller able to overcome the changing dynamics of the Cellulo-Mori robot, and, more specifically, its weight changes as different Mori elements are joined together on top of the Cellulo. The controller development was aimed at a possible deployment on a real robot by the end of the project.

## Design

From a variety of controllers proposed in the literature, the Model Reference Adaptive Controller (MRAC) architecture was chosen as a suitable candidate for this task for its simple design, versatility, and expandability. Feedback Linearisation was used in conjunction with the Cellulo dynamics developed in parallel to decouple and linearise the system into a simplified plant, facilitating considerably the controller design without losing precision and generality, as is usually the case with simple linearisation. The controller itself was then designed using Lyapunov's stability theory, which helped guarantee stability and convergence to the desired reference model behaviour. The implemented disturbance model is simplistic, but due to the particular formulation of the controller, it can easily be expanded as needed without having to redesign the system.

The resulting adaptive controller can hence automatically update its internal parameters based on the observed robot response and can track a desired velocity even with modelling errors. It can thus adapt itself to external perturbations, be it additional weight or other friction variations.

## Simulation

The finished controller was tested in simulation to validate its stability and adaptive capabilities. The results revealed a strong dependency between the model tracking performance and the frequency of the update loop, with the faster control loops resulting in faster adaptation and more precise tracking. The results also showed an effective increase of the transient response performance to command steps as time passes, no matter the update frequency (as long as it is higher than 14 [Hz]), proving that the designed controller is indeed able to change its parameters to adapt to even large modelling errors.

## Implementation

The controller was finally deployed on a real Cellulo for a limited period of time. With minimal adjustments, it proved able to follow changing velocity commands to track specific positions in the playing area. This is encouraging as it is a task that the controller is normally not built to do, preferring constant inputs to update the parameters properly.

The objective was then to compare the adaptive controller to the default controller when an increasing amount of weight is added on top of the Cellulo. This task proved harder than expected, as it highlighted a limitation of the current Cellulo design, namely a largely increased friction in the wheel transmission when subject to a mass increase. This created static friction barriers that occasionally prevented the motors from actuating the robot, and resulted in small bumps and ripples on the paper causing the robot to stop moving entirely at low speeds. Manual help was required to make the robot move again.

Despite this issue, tests were conducted on limited runs with both controllers, revealing a good adaptation capability and faster movements compared to the original Cellulo controller. The experiment also revealed an unexpected advantage of the adaptive controller in regards to the unexpected stops. The controller was indeed able to scale the motor commands to respond to the mass increase, but it also increased them when a motor got stuck, which helped it occasionally get moving without external help.

The tests were insufficient to provide representative numerical results, but the overall qualitative performance is very satisfying. The controller showed good potential as it was able to adapt to increasing weights and overcame the non-ideal conditions of the experiments where the default controller failed.

A more thorough comparison done in the future could show the true magnitude of the advantages provided by this adaptive controller.

## 6.1 Future work

Here are some suggestions for ways to improve the controller in the future and potential approaches to compensate the limitations highlighted during the experiments:

**Proper calibration and testing:** Due to time constraints, a proper calibration procedure was not implemented. A procedure with a symmetric exploration of the $x$ and $y$ coordinates is important to guarantee proper calibration. Tests of the transient response to constant velocity commands were not made once deployed on a real robot either. They could reveal other adjustments that need to be made to the controller and provide a better comparison between controllers than the one made here.

**Discretisation:** One major change would be to adjust the controller formulation to take the discrete nature of the system into account. This could increase its response speed, stability, and robustness to a changing update frequency, as well as improve its model tracking performance.

**Disturbance model:** The current disturbance model is simplistic and makes assumptions regarding friction that do not hold well in reality. The controller is still able to adapt to those imperfections, but building a more complete disturbance model would result in cleaner motions and a better adaptation overall, with notably fewer parameter oscillations as the system could follow the reference model more closely. This could also be an opportunity to add specific compensations to user input in the disturbance model.

**Learning rates:** Learning rates do not have to be fixed. Finding a stable formulation that allows the learning rates to adapt to the situation may help with parameter oscillations and long calibration times. They may increase when the error does not change much and decrease when the error reduces considerably.

**Cellulo limitations:** The maximum mass of 100 [gr] is very limiting for the Cellulo-Mori robot, but adjusting the Cellulo design without a lot of changes is complicated due to its compact and optimised nature. As seen in the experiments, the adaptive controller can cope with some of the limitations due to friction. An increased maximum motor torque or dedicated software may be enough to overcome some issues, but any change to the design that reduces friction in the transmission would have a much better result on the weight robustness of the Cellulo. In the end, omniwheels may be considered a good option.

# Appendix A

# Linearisation

## A.1 Differential Equations

The following section makes extensive use of the notation and equations developed in the original Cellulo thesis by Ayberk Özgür [1]. They are used to establish the differential equations of the system, extract the state-space model and make a linearisation around a typical working point. We use the following generalised coordinates $(q)$, inputs $(u)$, and outputs $(Y)$ for our state space:

$$q = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \quad \dot{q} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} v_x^G \\ v_y^G \\ \omega \end{pmatrix} \quad u = \begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} \quad Y = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = q \tag{A.1}$$

With x and y being the Cellulo's position in the global frame and $\theta$ its rotation with respect to the x axis, as illustrated in figure A.1. $U_{1-3}$ are the voltage commands of the 3 driving motors of the robot. The Cellulo is using a localisation method based on an anoto pattern [1][16], it is therefore considered that the outputs and thus the states of the robot are known at any point.

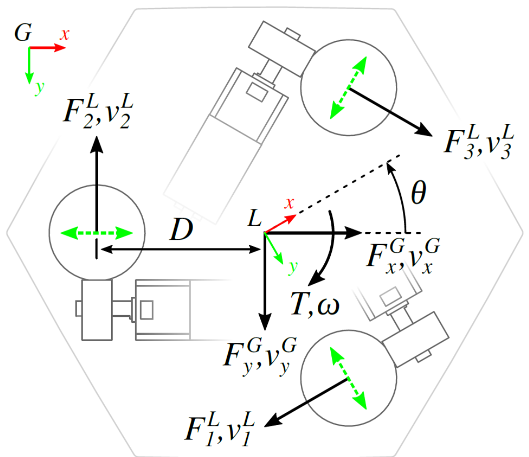


Figure A.1: [1, 71] Quantities of interest and reference frames of the Cellulo robot, top-down view. Variables are set either in the global or local reference frame, noted by G or L respectively. The green arrows mark the free DOF of each wheel. Vector quantities having same direction are overlaid on the same arrow. In the above state, the robot is rotated by about -30° with respect to the global frame.

Note that in this document the distance $D$ in figure A.1 is noted with $d$ to avoid confusion with other variables. From the geometry depicted in figure A.1, we can derive the following kinematic and dynamic relations between the robot's pose and the wheel ground speeds and ground forces [1, 71-72]:

$$\begin{pmatrix} v_1^L \\ v_2^L \\ v_3^L \end{pmatrix} = K^{-1} R(-\theta) \begin{pmatrix} v_x^G \\ v_y^G \\ \omega \end{pmatrix} \tag{A.2}$$

$$\begin{pmatrix} F_1^L \\ F_2^L \\ F_3^L \end{pmatrix} = C^{-1} R\left(-\theta\right) \begin{pmatrix} m\ddot{x} \\ m\ddot{y} \\ I_z\ddot{\theta} \end{pmatrix} \tag{A.3}$$

Where $m$ is the mass of the Cellulo, $I_z$ its inertia, and:

$$K^{-1} = \begin{pmatrix} -1 & 0 & d \\ 1/2 & -\sqrt{3}/2 & d \\ 1/2 & \sqrt{3}/2 & d \end{pmatrix}, \quad R(\theta) = \begin{pmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{A.4}$$

And:

$$C^{-1} = \begin{pmatrix} -2/3 & 0 & 1/(3d) \\ 1/3 & -\sqrt{3}/3 & 1/(3d) \\ 1/3 & \sqrt{3}/3 & 1/(3d) \end{pmatrix} \tag{A.5}$$

Notably we get the relation:

$$R(\theta)\ C\ K^{-1}\ R(-\theta) = \begin{pmatrix} 3/2 & 0 & 0 \\ 0 & 3/2 & 0 \\ 0 & 0 & 3d^2 \end{pmatrix} \tag{A.6}$$

The relation between each motor torque ($T_j$) and its corresponding ground force ($F_j^L$) was derived in [1, 70] as a discontinuous piece-wise linear function of the ground speed's sign. The discontinuity happens because the motor is not fixed to the robot frame and the number of ball transfers in contact with the wheel varies with the direction of motion.

$$T_j = c_F(sign(v_j^L))F_j^L + c(sign(v_j^L)) \tag{A.7}$$

Where $c_F(s)$ represents the force-torque coupling (always positive) and $c(s)$ represents friction (same sign as $s$). They are originally dual constants[1, 73] that take two different values depending on the sign of $s$, but to ease the calculations they have been simplified to have a single constant values, thus only the sign of $c$ changes:

$$T_j = c_F F_j^L + c\ sign(v_j^L) \tag{A.8}$$

Assuming no slip between the drive roller-wheel and wheel-ground contacts, the standard torque-voltage relation for DC motors can be rewritten as a function of the ground speed instead of the angular speed using the drive roller radius:

$$T_j = c_U U_j - c_\omega \omega_j \quad \longrightarrow \quad T_j = c_U U_j - \frac{c_\omega}{r_{drive}} v_j^L = c_U U_j - c_v v_j^L \tag{A.9}$$

Where $c_U$, $c_\omega$, and $c_v$ are three positive constants, and $\omega_j$ is the drive roller angular speed. We can now derive the differential equations by isolating the second order derivatives $\ddot{q}$ as functions of $q$, $\dot{q}$, and $u$. Using equations (A.8), (A.9), and the kinematics (A.2), we can express the external forces acting on the Cellulo motors as functions of $\dot{q}$, and $u$:

$$\begin{pmatrix} F_1^L \\ F_2^L \\ F_3^L \end{pmatrix} = \frac{c_U}{c_F} \begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} - \frac{1}{c_F} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} - \frac{c_v}{c_F} K^{-1} R\left(-\theta\right) \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} \tag{A.10}$$

Where the $c_{1-3}$ term represents the transfer friction and the $\dot{q}$ term the motor friction. $c_i$ is a shortened notation for $c\ sign(v_i^L)$. Using the inverse of the dynamic equations (A.3), we get the following differential equations for $\ddot{q}$:

$$\begin{pmatrix} m\ddot{x} \\ m\ddot{y} \\ I_z\ddot{\theta} \end{pmatrix} = \frac{1}{c_F} R\left(\theta\right) C \left( c_U \begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} - c_v K^{-1} R(-\theta) \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} - \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} \right) \tag{A.11}$$

Using the relation (A.6), we can simplify the result by extracting the velocities from the parenthesis to get a similar result as in [19, 51]:

$$\begin{pmatrix} m\ddot{x} \\ m\ddot{y} \\ I_z\ddot{\theta} \end{pmatrix} = \frac{c_U}{c_F} R\left(\theta\right) C \left( \begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} - \frac{1}{c_U} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} \right) - \frac{3}{2} \frac{c_v}{c_F} \begin{pmatrix} \dot{x} \\ \dot{y} \\ 2d^2\dot{\theta} \end{pmatrix} \tag{A.12}$$

## A.2 Linearisation

Now that we have the differential equations, we can linearise the system around an operating point to get an understanding of its behaviour in certain conditions. To do this we make a first order approximation of the state space equations (A.13) by evaluating its jacobian at the desired operating point $(\bar{X}, \bar{u})$ and extracting the $A$ and $B$ matrices:

$$X = \begin{pmatrix} x & y & \theta & \dot{x} & \dot{y} & \dot{\theta} \end{pmatrix}^T \quad \longrightarrow \quad \dot{X} = \begin{pmatrix} f_1(X,u) \\ \vdots \\ f_6(X,u) \end{pmatrix} \quad Y = \begin{pmatrix} x & y & \theta \end{pmatrix}^T \tag{A.13}$$

$$\begin{aligned} \dot{X} &= AX + Bu \\ Y &= CX + Du \end{aligned} \tag{A.14}$$

Where $X$ are the states of the system, $u$ the input, and $Y$ the output. An example is developed by linearising around the state $\bar{X}$ described in (A.15). It corresponds to the Cellulo moving at the constant velocity $v_x = 100$ mm/s. We can get the appropriate input vector $\bar{u}$ by replacing $X$ by $\bar{X}$ in equation (A.12), setting the second order derivatives to zero, and solving for $\bar{U}_1$, $\bar{U}_2$, and $\bar{U}_3$. In our example we get the following solution:

$$\begin{gathered} \bar{X} = \begin{pmatrix} 0 & 0 & 0 & 0.1 & 0 & 0 \end{pmatrix}^T \\ \bar{u} = \begin{pmatrix} -\frac{c}{c_U} - \frac{c_v}{10\ c_U} & \frac{c}{c_U} + \frac{c_v}{20\ c_U} & \frac{c}{c_U} + \frac{c_v}{20\ c_U} \end{pmatrix}^T \end{gathered} \tag{A.15}$$

The $A$, $B$, $C$, and $D$ matrices hence are:

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{3\ c_v}{2\ c_F\ m} & 0 & 0 \\ 0 & 0 & \frac{3\ c_v}{20\ c_F\ m} & 0 & -\frac{3\ c_v}{2\ c_F\ m} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{3\ c_v\ d^2}{I_z\ c_F} \end{pmatrix} \tag{A.16}$$

$$B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -\frac{c_U}{c_F\ m} & \frac{c_U}{2\ c_F\ m} & \frac{c_U}{2\ c_F\ m} \\ 0 & -\frac{\sqrt{3}\ c_U}{2\ c_F\ m} & -\frac{\sqrt{3}\ c_U}{2\ c_F\ m} \\ \frac{c_U\ d}{I_z\ c_F} & \frac{c_U\ d}{I_z\ c_F} & \frac{c_U\ d}{I_z\ c_F} \end{pmatrix} \tag{A.17}$$

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \quad D = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{A.18}$$

The C and D matrices simple forms come from the absolute observability of the output, thanks to the anoto pattern.

Using the numerical values we get the following state space matrices:

$$\bar{u} = \begin{pmatrix} -2.8037 & 1.4395 & 1.4395 \end{pmatrix}^T \tag{A.19}$$

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -119.9 & 0 & 0 \\ 0 & 0 & 11.99 & 0 & -119.9 & 0 \\ 0 & 0 & 0 & 0 & 0 & -417.7 \end{pmatrix} \quad B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -2.929 & 1.4645 & 1.4645 \\ 0 & -2.5366 & 2.5366 \\ 182.25 & 182.25 & 182.25 \end{pmatrix} \tag{A.20}$$

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \qquad D = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

From this we can get the poles of the system. The system has no zeroes:

$$\lambda = eig(A) = \begin{pmatrix} 0 & 0 & 0 & -119.9 & -119.9 & -417.7 \end{pmatrix}^T \tag{A.21}$$

# Appendix B

# Canonical Equations

## B.1 Lagrangian

To develop the canonical equations of motion of the Cellulo, we first need to write its Lagrangian. For this, we use the same generalised coordinates and notations seen in Appendix A, based on the original Cellulo thesis [1]:

$$q = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \quad \dot{q} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} v_x^G \\ v_y^G \\ \omega \end{pmatrix} \quad u = \begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} \quad Y = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = q \tag{B.1}$$

To derive the energy equations, we assume that the only moving parts of the robot are the frame, the ball wheels, and the magnetic drive rollers. As there is no notable change in height happening during the Cellulo movements, the potential energies can be chosen equal to 0. We will thus aim at writing the Lagrangian of the system as the sum of the kinetic and rotational energies of every moving part:

$$L = T_{total} = T_{body} + T_{drive} + T_{wheel} \tag{B.2}$$

### B.1.1 Body energy

The full body kinetic and rotational energy due to the Cellulo's speed.

$$T_{body} = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2) + \frac{1}{2}I_z\dot{\theta}^2 \tag{B.3}$$

Where $m$ is the mass of the whole robot and $I_z$ is its inertia along the vertical axis passing through its centre of mass. $I_z$ is calculated by approximating the Cellulo as a cylinder of radius r = 3 [cm].

### B.1.2 Drive rollers rotational energy

As the kinetic energy of the drive rollers is already contained in the body energy (B.3), the only energy left to account for them is their rotational energy:

$$T_{drive} = \frac{1}{2}I_{drive}(\omega_1^2 + \omega_2^2 + \omega_3^2) \tag{B.4}$$

Where $\omega_{1-3}$ are the angular speed of each drive roller. To relate the roller angular speeds to the generalised coordinates, we use the kinematic relation (A.2) to get the speed $v_i^L$ (the speed at the point of contact of the i-th ball wheel with the ground and in the direction of the drive roller's rotation) from the Cellulo velocity. Using the drive roller radius $r_{drive}$ we can then write:

$$\omega_{drive} = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} = \frac{1}{r_{drive}} \begin{pmatrix} v_1^L \\ v_2^L \\ v_3^L \end{pmatrix} = \frac{1}{r_{drive}} K^{-1} R(-\theta) \begin{pmatrix} v_x^G \\ v_y^G \\ \omega \end{pmatrix} \tag{B.5}$$

Neglecting the motor inertia, we calculate the drive roller inertia as if it was a full cylinder of height $h_{drive}$:

$$m_{drive} = \rho_{neodymium}\pi r_{drive}^2 h_{drive} \quad \longrightarrow \quad I_{drive} = \frac{1}{2}m_{drive}r_{drive}^2 \tag{B.6}$$

Using (B.5) and (B.6), we can simplify the drive energy (B.4) as:

$$T_{drive} = \frac{3}{4}\frac{I_{drive}}{r_{drive}^2}(2d^2\dot{\theta}^2 + \dot{x}^2 + \dot{y}^2) = \frac{3}{8}m_{drive}(2d^2\dot{\theta}^2 + \dot{x}^2 + \dot{y}^2) \tag{B.7}$$

### B.1.3 Wheels rotational energy

Similarly to the drive rollers, there is only the rotational energy of the wheels left to calculate. Due to their degree of freedom illustrated in figure B.1, the wheels rotate so that the ground speed of each wheel follows the frame $\dot{x}$ and $\dot{y}$ speeds, with an additional change due to the Cellulo's rotational speed $\dot{\theta}$. To get the correct angular speed for each wheel and extract the rotational energy, we need to calculate the sum of those interactions.

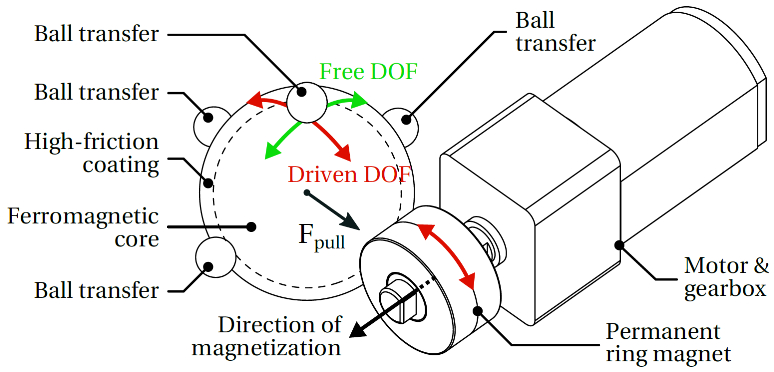


Figure B.1: [1, 64] Overview of the permanentmagnet-assisted ball wheel design of the Cellulo. The ball wheel with ferromagnetic core is driven by a permanent ringmagnet that acts as the drive roller. The magnet temporarily magnetizes the wheel, exerting a pull force and generating the necessary normal force, which in turn generates the friction force that drives the wheel. The wheel thus acquires one driven and one free DOF.

The direction of the speed component due to $\dot{\theta}$ can be expressed in the $x - y$ frame for each wheel as the corresponding unit vector $v_i$:

$$v_1 = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \quad v_2 = \begin{pmatrix} cos(\pi/3) \\ -sin(\pi/3) \end{pmatrix} \quad v_3 = \begin{pmatrix} cos(\pi/3) \\ sin(\pi/3) \end{pmatrix} \tag{B.8}$$

To account for the Cellulo's orientation, we must rotate them according to the angle $\theta$ and multiply them by the appropriate amplitude $||v_{\dot{\theta}}|| = D\dot{\theta}$. We then get the full rotational participation to the velocity of each wheel:

$$v_{\dot{\theta},i} = D\dot{\theta}\begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix} v_i \tag{B.9}$$

We can now sum all contributions and express the angular speed of each wheel:

$$v_{x,y} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \quad \longrightarrow \quad \omega_i = \frac{1}{r_{wheel}}\left|\left|v_{x,y} + v_{\dot{\theta},i}\right|\right| \tag{B.10}$$

The calculation of the wheel's inertia, $I_{wheel}$, is straight forward as the wheels are solid spheres of steel, coated with a fine layer of rubber (see figure B.1). We can finally write down the rotational energy for all the wheels:

$$T_{wheel} = \frac{1}{2}I_{wheel}(\omega_1^2 + \omega_2^2 + \omega_3^2) \tag{B.11}$$

Which can be simplified using (B.10) as:

$$T_{wheel} = \frac{3}{2}\frac{I_{wheel}}{r_{wheel}^2}(d^2\dot{\theta}^2 + \dot{x}^2 + \dot{y}^2) \approx \frac{3}{5}m_{steel}(d^2\dot{\theta}^2 + \dot{x}^2 + \dot{y}^2) \tag{B.12}$$

Where the approximation is made that the rubber inertia is negligible. The full inertia has been used for the numerical calculations, but the rubber accounts only for ∼2.4% of the wheel total inertia. To get the exact value, replace $m_{steel}$ in (B.12) by:

$$\frac{m_{rubber}(r_{wheel}^2 - r_{steel}^2) + m_{steel}r_{steel}^2}{r_{wheel}^2} \tag{B.13}$$

### B.1.4 Total energy

$$L = \left(\frac{m}{2} + \frac{3m_{steel}}{5} + \frac{3m_{drive}}{8}\right)(\dot{x}^2 + \dot{y}^2) + \left(\frac{I_z}{2} + \frac{3d^2 m_{steel}}{5} + \frac{3d^2 m_{drive}}{4}\right)\dot{\theta}^2 \tag{B.14}$$

## B.2 Equations of motion

The equations of motion are derived for each generalised coordinate using the Euler-Lagrange equation with the added input and friction forces [20, 37] [21, 150]:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right)^T - \left(\frac{\partial L}{\partial q}\right)^T + D = Q \tag{B.15}$$

Where $D$ are the friction forces and $Q$ the input on the system. We will rewrite the equation in the canonical form [20, 39]:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + D(q,\dot{q}) = Q(q,\dot{q},u) \tag{B.16}$$

Where $M$ is the mass/inertia matrix, $C$ is the Coriolis and centrifugal force matrix, and $G$ is the gravitational force matrix.

### B.2.1 Euler-Lagrange equation

Due to the simple form of the Lagrangian (B.14), we can easily evaluate the differential components of equation (B.15) and extract the $M$, $C$ and $G$ matrices:

$$M \approx \begin{pmatrix} m + \frac{3}{4}m_{drive} + \frac{6}{5}m_{steel} & 0 & 0 \\ 0 & m + \frac{3}{4}m_{drive} + \frac{6}{5}m_{steel} & 0 \\ 0 & 0 & I_z + \frac{3d^2}{2}m_{drive} + \frac{6d^2}{5}m_{steel} \end{pmatrix}$$
$$C = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad G = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{B.17}$$

To get the exact value for $M$, replace once again $m_{steel}$ in (B.17) by the expression (B.13). The $C$ and $G$ matrices are null because the Lagrangian (B.14) is a quadratic function of $\dot{q}$ only, and the potential energy of the system is null.

### B.2.2 Input and friction forces

Q is the vector of generalised forces applied to each generalised coordinate. This is to say, the sum of the direct impact every external force has on each coordinate. It can be expressed as a function of external forces directly using a Jacobian mapping:

$$Q = F_{generalised} = \left(\frac{\partial r}{\partial q}\right)^T F_{ext} \tag{B.18}$$

Where r is a displacement along an axis on which the external force or torque is actuating. It appears that this mapping ends up being equal to the same Jacobian using the derivatives [20, 45]:

$$\frac{\partial r}{\partial q} = \frac{\partial \dot{r}}{\partial \dot{q}} \tag{B.19}$$

To make use of the equations derived in Appendix A, we can choose external forces as the forces on the wheels $F_i^L$. Then the derivatives $\dot{r}$ are the corresponding velocities on the wheels $v_i^L$. The kinematic equation (A.2) can thus be used to get the Jacobian:

$$\frac{\partial \dot{r}}{\partial \dot{q}} = K^{-1}R(-\theta) = \begin{pmatrix} -cos(\theta) & -sin(\theta) & d \\ cos(\theta - \pi/3) & -cos(\theta + \pi/6) & d \\ cos(\theta + \pi/3) & sin(\theta + \pi/3) & d \end{pmatrix} \tag{B.20}$$

We recall the external forces applied on the Cellulo motors, as derived in equation (A.10):

$$\begin{pmatrix} F_1^L \\ F_2^L \\ F_3^L \end{pmatrix} = \frac{c_U}{c_F} \begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} - \frac{1}{c_F} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} - \frac{c_v}{c_F} K^{-1} R\left(-\theta\right) \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} \tag{B.21}$$

Where we find a term in $u$, the input to the system, and two terms in $c$ and $\dot{q}$, friction forces. From this equation, using (B.18), we can isolate the input and friction vectors $Q$ and $D$ described in the canonical equations (B.16):

$$Q = \frac{c_U}{c_F} \left(K^{-1} R\left(-\theta\right)\right)^T \begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} \tag{B.22}$$

$$D = \frac{3}{2} \frac{c_v}{c_F} \begin{pmatrix} \dot{x} \\ \dot{y} \\ 2d^2\dot{\theta} \end{pmatrix} + \frac{1}{c_F} \left(K^{-1} R\left(-\theta\right)\right)^T \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} \tag{B.23}$$

We will make a distinction between the two friction components by moving the speed-proportional element of $D$ to a separate term $C'$ similar to the Coriolis matrix $C$. This is purely to help with readability during the controller development. For the same reason, we will from now on express the input and friction vectors as a matrix-vector products:

$$Q = Q' \cdot u \quad \longrightarrow \quad Q' = \frac{c_U}{c_F} \begin{pmatrix} -cos(\theta) & cos(\theta - \frac{\pi}{3}) & cos(\theta + \frac{\pi}{3}) \\ -sin(\theta) & -cos(\theta + \frac{\pi}{6}) & sin(\theta + \frac{\pi}{3}) \\ d & d & d \end{pmatrix} \tag{B.24}$$

$$D = C' \cdot \dot{q} + D' \cdot \vec{c} \quad \longrightarrow \quad \begin{cases} C' = \dfrac{3}{2} \dfrac{c_v}{c_F} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2d^2 \end{pmatrix} \\ D' = \dfrac{1}{c_F} \begin{pmatrix} -cos(\theta) & cos(\theta - \frac{\pi}{3}) & cos(\theta + \frac{\pi}{3}) \\ -sin(\theta) & -cos(\theta + \frac{\pi}{6}) & sin(\theta + \frac{\pi}{3}) \\ d & d & d \end{pmatrix} \end{cases} \tag{B.25}$$

Where $\vec{c}$ is the vector of $c_i$.

### B.2.3 Numerical results

The numerical values obtained with the exact wheel inertia are as follows:

$$M = \begin{pmatrix} 0.2128 & 0 & 0 \\ 0 & 0.2128 & 0 \\ 0 & 0 & 1.018e-4 \end{pmatrix} \quad C = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad G = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$C' = \begin{pmatrix} 23.974 & 0 & 0 \\ 0 & 23.974 & 0 \\ 0 & 0 & 3.759e-2 \end{pmatrix}$$

$$D' = 128 \begin{pmatrix} -cos(\theta) & cos(\theta - \frac{\pi}{3}) & cos(\theta + \frac{\pi}{3}) \\ -sin(\theta) & -cos(\theta + \frac{\pi}{6}) & sin(\theta + \frac{\pi}{3}) \\ 0.028 & 0.028 & 0.028 \end{pmatrix} \tag{B.26}$$

$$Q' = 0.5858 \begin{pmatrix} -cos(\theta) & cos(\theta - \frac{\pi}{3}) & cos(\theta + \frac{\pi}{3}) \\ -sin(\theta) & -cos(\theta + \frac{\pi}{6}) & sin(\theta + \frac{\pi}{3}) \\ 0.028 & 0.028 & 0.028 \end{pmatrix}$$

Now if we want to get the evolution of the system given a state $(q, \dot{q})$ and an input $(u)$, we can do so by evaluating this expression:

$$\ddot{q} = M^{-1}(-C\dot{q} - G - D(q, \dot{q}) + Q(q, \dot{q}, u)) = M^{-1}(-C' \cdot \dot{q} - D' \cdot \vec{c} + Q' \cdot u) \tag{B.27}$$

### B.2.4 Simulation

As a small sanity check, we plug the model in Simulink to see if the full model gives similar results as the linearised model for similar inputs:

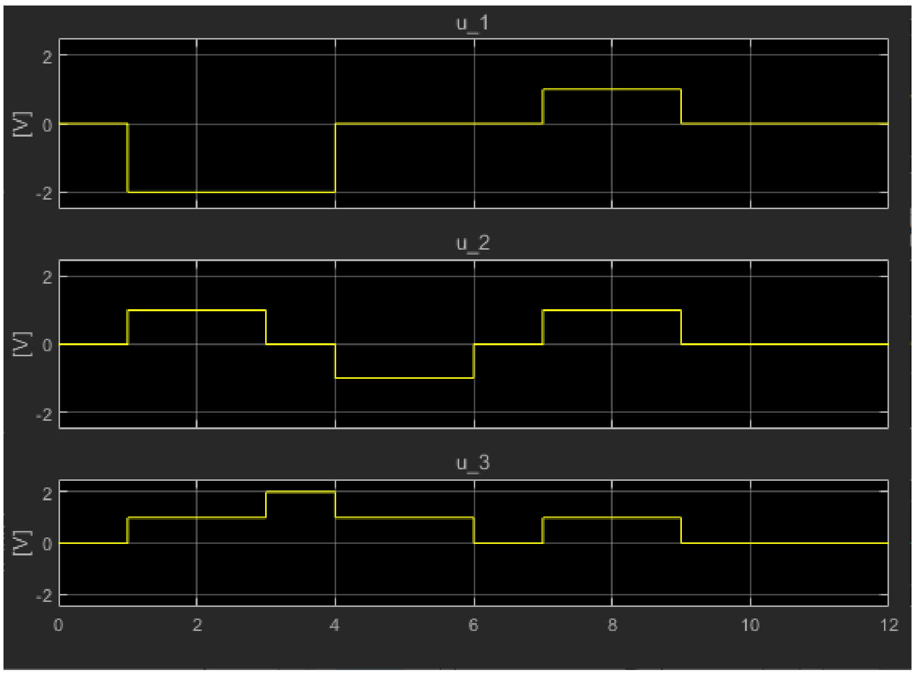


Figure B.2: Inputs given to both systems during a 10 seconds period. It does not include the operating point bias voltages.

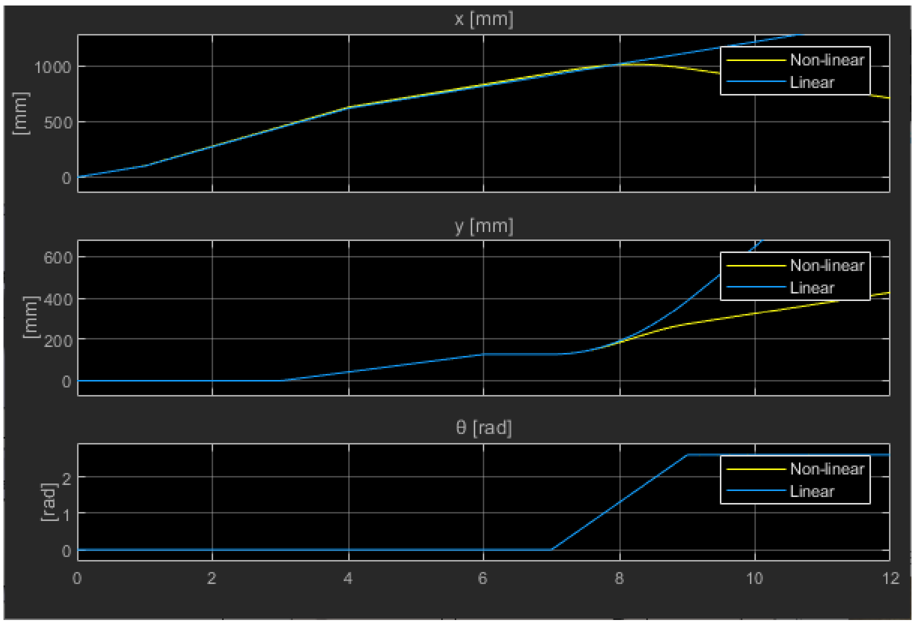


Figure B.3: $x$, $y$, and $\theta$ positions resulting from the simulation on a 10 seconds period.

The chosen inputs $u_i$ have been chosen to produce a dephased square perturbation on the resulting speeds, as illustrated in Figure B.4. As we can see, the two models have the same behaviour until there is an angle perturbation, which deviates the linear model from its operating point.

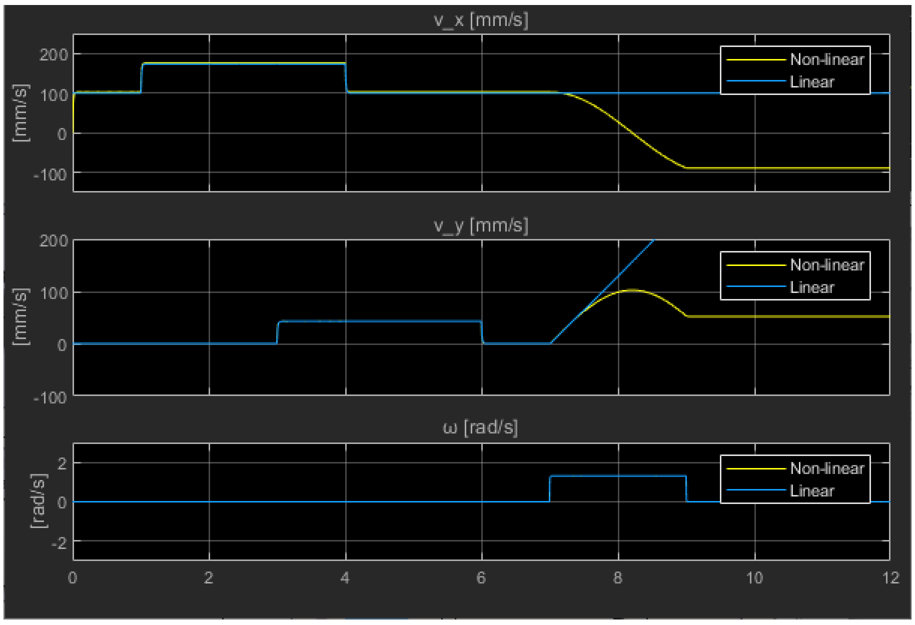


Figure B.4: $\dot{x}$, $\dot{y}$, and $\dot{\theta}$ speeds $(v_x, v_y, \omega)$ resulting from the simulation on a 10 seconds period.

# Bibliography

[1] Ayberk Özgür. *Cellulo: Tangible Haptic Swarm Robots for Learning.* PhD thesis, EPFL, 2018.

[2] Christoph H. Belke and Jamie Paik. Mori: A modular origami robot. *IEEE/ASME Transactions on Mechatronics*, 22(5):2153–2164, 2017.

[3] Karl Johan Åström and Björn Wittenmark. *Adaptive Control, Second Edition.* Dover Publications, Inc., Mineola, New York, 1995.

[4] Reshmi R., Gnanasoundharam J., and K. Kotteeswaran R. Design of self-tuning regulator for non-linear unstable system. *International Journal of Pure and Applied Mathematics*, 118(20):61–66, 2018.

[5] M.M. Bayoumi, K.Y. Wong, and M.A. El-Bagoury. A self-tuning regulator for multivariable systems. *Automatica*, 17(4):575–592, 1981.

[6] K.J. Åström and B. Wittenmark. On self tuning regulators. *Automatica*, 9(2):185–199, 1973.

[7] Ramon Costa, Liu Hsu, Alvaro Imai, and Petar Kokotović. Lyapunov-based adaptive control of mimo systems. *Automatica*, 39:1251–1257, 07 2003.

[8] Wafa Ghozlane and Jilani Knani. Model reference adaptive control design for nonlinear plants. *International Journal of Advanced Computer Science and Applications*, 10(3):116–124, 2019.

[9] Sumit Kumar Sar and Lillie Dewan. Mrac based pi controller for speed control of d.c. motor using lab view. *WSEAS TRANSACTIONS on SYSTEMS and CONTROL*, 9:10–15, 2014.

[10] The MathWorks Inc. Model reference adaptive control. `https://ch.mathworks.com/help/slcontrol/ug/model-reference-adaptive-control.html`, 2021. Accessed: 2021-12-11.

[11] IIT Delhi July 2018 [YouTube channel]. Nonlinear and adaptive control. Retrieved December 11, 2021, from `https://www.youtube.com/playlist?list=PLp6ek2hDcoNCjLvhsGNJnJi-UtYOCMBQR`, 2018.

[12] Kevin M. Passino and Stephen Yurkovich. *Fuzzy Control.* Addison Wesley Longman Inc., Menlo Park, California, 1998.

[13] James Carvajal, Guanrong Chen, and Haluk Ogmen. Fuzzy pid controller: Design, performance evaluation, and stability analysis. *Information Sciences*, 123:249–270, 2000.

[14] Wafa Ghozlane and Jilani Knani. Nonlinear control via input-output feedback linearization of a robot manipulator. *Advances in Science, Technology and Engineering Systems Journal*, 3(5):374–381, 2018.

[15] Denis Gillet. *Multivariable control and coordination systems [Course Notes].* EPFL, 2019.

[16] Mats Petter Pettersson. Method and device for decoding a position-coding pattern. *US Patent*, 7,145,556, 2006.

[17] Hadrien Sprumont. Cellulo adaptive controller [c4science repository]. `https://c4science.ch/diffusion/11873/`, 2021.

[18] Pololu Corporation. *Micro Metal Gearmotors: Pololu 2364*, 9 2021. Rev. 5.0.

[19] Tamás Kalmár-Nagy, Raffaello D'Andrea, and Pritam Ganguly. Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle. *Robotics and Autonomous Systems*, 46(1):47–64, 2004.

[20] Thomas Kølbæk Jespersen. Kugle - modelling and control of a ball-balancing robot. Master's thesis, Aalborg university, 2019.

[21] Cristina Roche and Victor Borja. Control of a ball-balancing robot. Master's thesis, Aalborg university, 2020.