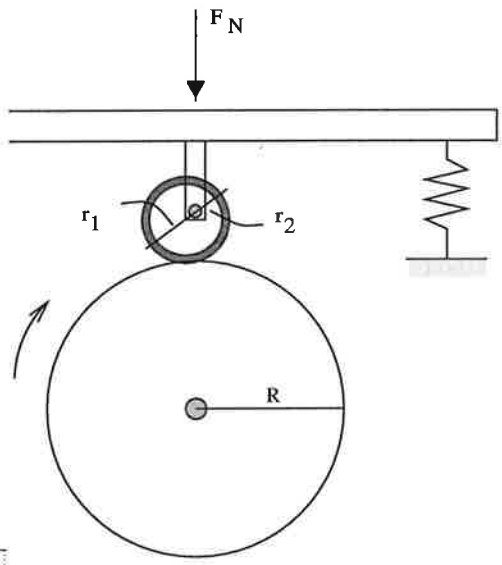
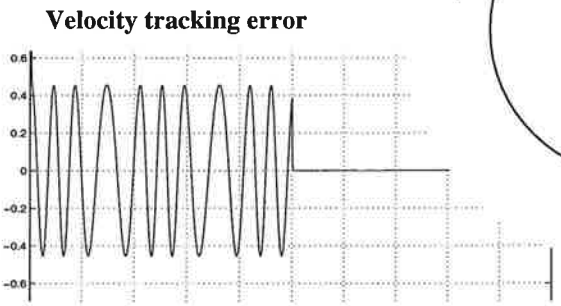
ISSN 0280–5316
ISRN LUTFD2/TFRT--5603--SE

Adaptive Eccentricity Compensation an Experimental Study

Holger Olofsson
Petra Posselius

ec · cen · tric (ik sen'trik, ek-), *adj.* **1.** [...] **2.** *Geom.* not having the same center; not concentric [...] **3.** not situated in the center, as an axis. **4.** *Mach.* having the axis or support away from the center, as a wheel. [...] **–*n*** **6.** a person who has an unusual, peculiar, or odd personality [...]





Department of Automatic Control
Lund Institute of Technology
October 1998

Department of Automatic Control
Lund Institute of Technology
Box 118
S-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
October 1998

Document Number
ISRN LUTFD2/TFRT--5603--SE

Author(s)
Holger Olofsson
Petra Posselius

Supervisor
Karl Johan Åström
Carlos Canudas de Wit (LAG)

Sponsoring organisation

Title and subtitle
Adaptive Eccentricity Compensation
– an Experimental Study –
(Adaptiv kompensation för excentricitet – en experimentell studie)

Abstract

This master thesis presents a validation of a new model for adaptive compensation of periodic disturbances with unknown amplitude and frequency. This compensation can be of interest in applications such as motor drives, gears, ball bearings, CD-players and rolling mills. The model, which in difference from most previous models is position-dependent, has been developed by Carlos Canudas de Wit and Laurent Praly. Results from tests performed in simulation for a general periodic perturbation are shown. A model error has been located for negative velocities and its origin has been described. The compensator has been applied to a system where the perturbation is due to friction between two rolls with eccentric axes. An experimental setup has been constructed for this purpose. An new observer is presented to improve the difficulties encountered for sign reversals in the reference signal. A comparative study with a conventional PI-controller is also presented. The experiments show that this adaptive observer performs very well for positive velocities and gives over all better results than linear control.

Key words
adaptive compensation, eccentricity, position-dependent

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
61

Recipient's notes

Security classification

Preface

Before leaving for a year as Erasmus students at Grenoble Institute for Electrical Engineering, we passed by Karl Johan Åström's office. He welcomed us in his typically friendly and enthusiastic way and proposed that if we wanted to do our masters thesis in France, we could talk to his French colleague Carlos Canudas de Wit at the Laboratoire d'Automatique de Grenoble.

After two trimesters of studying we finally decided to do our final project in France, and were offered several possible projects by Carlos. We decided on the eccentric proposal since it was a new idea requiring a lot of practical work, it was something fairly new and intriguing, and we could tell that Carlos was very interested in this work himself, partly because he was working on an article on the same topic. We would like to take the opportunity to thank Carlos for his tremendous support in the lab as well as outside of work. He sat down with us for hours to explain and to try to improve our results. We worked with this project during the spring and summer 1998.

When we finally had finished the practical part we went back to Sweden where we struggled with LaTeX to finish the report. One day Laurent Praly, Carlos' colleague behind the theory we've been working on, turned up in Lund. We worked in the same office for two weeks where he helped us formulating the theory. In particular he discovered an error in the model which we have been trying to take into account in this report despite the late notice.

To conclude we would like to wish you a pleasant reading and once again thank everybody involved who has helped us to accomplish this project.

October 1998
Lund, Sweden

Holger Olofsson

Petra Posselius

Contents

1. Introduction

In this master thesis we present an experimental study of a new model for adaptive compensation of periodic disturbances with unknown amplitude and frequency. This compensation can be of interest in applications such as motor drives, gears, ball bearings, CD-players and rolling mills. A typical situation is control of a rotating system with eccentric axis as in our experimental setup. Models of these disturbances have been made before but has been time-dependent. The model we are using is position-dependent which has the advantage of having the same frequency for all velocities.

The primary objective of our investigation is to validate a model developed by Carlos Canudas de Wit from Laboratoire d'Automatique de Grenoble, and Laurent Praly from Ecole des Mines de Paris. This is done in simulations as well as in practical experiments. The results show that the attenuation of the perturbation is very good, also when the velocity changes.

We have tested the results on a motor with a perturbation consisting of friction variations due to a wheel with an eccentric axis. In accordance with the perturbation caused by eccentricity we call the compensation *Adaptive Eccentricity Compensation*, later referred to as AEC. A series of experiments is presented in the later sections with very good results for constant as well as sinusoidal velocities.

The passage over zero velocity causes a problem in the velocity tracking for the compensator. Therefore, another compensation method, based on the same model, is presented in theory. Simulations as well as practical experiments show that the modified scheme works better.

The results are also compared with regular PI-control. This shows that the new adaptive system can operate over a much wider range of velocities with more or less the same error, but that the PI-controller performs well only for a specific velocity.

The report starts with the presentation of the model in chapter 2. Design of a compensator is presented in chapter 3 together with analysis. Chapter 4 covers a brief description of friction with emphasis on the LuGre model. Simulations are presented and analyzed in chapter 5. A new improved observer is introduced in chapter 6. The experimental setup is presented in chapter 7 and in chapter 8 we present the real-time experiment results. Finally conclusions of our work are given in chapter 9.

2. The Model

In this chapter we present the model for the perturbation that we want to compensate. The model is based on unpublished work of Carlos Canudas de Wit from Laboratoire d'Automatique de Grenoble, and Laurent Praly from Ecole des Mines de Paris.

We first present the system and the perturbation we want to compensate. Then we introduce a spatial operator ∇. This operator is used to convert the perturbation between the space- and the time-domain. Finally the internal system of the perturbation is presented in the space-domain.

An error in the model, which we discovered very late in the project is discussed together with its consequences.

2.1 Overview

The perturbation model we presented in this thesis is dependent of position, which differs it from most previous models. We have a system of the form

$$J\dot{v} = u + d(x); \quad v = \dot{x} = \frac{dx}{dt} \tag{2.1}$$

where x is the angular position of the system, J is the inertia, u is the control input and $d(x)$ the position-dependent periodic disturbance, defined as

$$d(x) = \Lambda \sin(\omega x + \varphi) \tag{2.2}$$

$$= a_1 \cos(\omega x) + a_2 \sin(\omega x) \tag{2.3}$$

In this thesis we design an internal model and an adaptive observer in the spatial domain. In most previous works d is considered as a function depending on time and not on space, of the form

$$d(x) = \Lambda \sin(\omega t + \varphi) \tag{2.4}$$

This is valid only if the velocity is constant so that $x(t)$ is proportional to time. As soon as the velocity changes the time-dependent model is no longer valid. As a basic example we can look at the case when velocity is zero, then the perturbation does not change while time is still running.

2.2 Internal Model

We start by presenting the definition of the ∇-operator.

Definition

First we introduce the spatial variable s defined as

$$s(t) = \int_0^t |v(\tau)| d\tau \tag{2.5}$$

s is thus the distance covered. We then define ∇ as

$$\nabla = \frac{d}{ds} = \frac{1}{|v|} D; \quad D = \frac{d}{dt} \tag{2.6}$$

We can now apply this definition to our internal model for the perturbation d, which is given from

$$y = J\dot{v} - u = d(x) \tag{2.7}$$

where $d(x)$ is defined as in (2.2).

Then y together with the calculation of ∇y gives

$$\begin{pmatrix} y \\ \nabla y/b \end{pmatrix} = \begin{pmatrix} \cos(\omega x) & \sin(\omega x) \\ -\sin(\omega x) & \cos(\omega x) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \tag{2.8}$$

and the internal model is defined as

$$\nabla^2 y = -\omega^2 y \tag{2.9}$$

Let $z_1 = y$, and $z_2 = \nabla y$, which gives a s-domain state space realization for (2.7) as

$$\boxed{\begin{aligned} \nabla z &= \begin{pmatrix} 0 & 1 \\ -\theta & 0 \end{pmatrix} z \\ y &= \begin{pmatrix} 1 & 0 \end{pmatrix} z \end{aligned}} \tag{2.10}$$

where $\theta = \omega^2$, and $z = [z_1\ z_2]^T$.

The model above is based on simple sinusoidal disturbances. We will later use perturbations, taking into account the non-linearities of friction, first by introducing the sign-function into the perturbation model

$$d(x) = F_0 \operatorname{sign}\big(1 + \Lambda \sin(\omega x + \varphi)\big) \tag{2.11}$$

and later by adding the entire LuGre friction model

$$d(x) = F_{LuGre} \operatorname{sign}\big(1 + \Lambda \sin(\omega x + \varphi)\big) \tag{2.12}$$

We tried to apply other internal models to these more accurate definitions of the perturbation. However, these calculations got very complex, and are maybe without solution, so for time efficient reasons, we decided not to continue this work. Therefore we use the same observer as defined above. This explains partly why the problems around zero velocity occur (see chapter 5 and 8), since the observer we use, does not compensate this phenomena properly.

2.3 There is Something Rotten in the State of Denmark – Part I

A late discovery during the thesis work is an error in the model, which can be seen when the velocity changes signs. The problem is due to the very basic definition described in equation (2.6). To explain why we start by the definition of x and $\hat{x}$

$$x = x_0 + \int_0^t v(\tau)\, d\tau$$

$$\hat{x} = x_0 + \int_0^t |v(\tau)|\, d\tau$$

They define the position, x, and the distance covered, $\hat{x}$, from a certain position, x_0, at time t_0. The perturbation is defined in 2.2. The difference between the true perturbation and our model is that the true perturbation is dependent on x while the model is dependent of $\hat{x}$. Let us divide the problem into two parts. The first is when the velocity is positive. Then we have:

$$\begin{aligned} x &= x_0 + \int_{t_0}^{t} v(\tau)\, d\tau \\ \hat{x} &= x_0 + \int_{t_0}^{t} |v(\tau)|\, d\tau = \\ &= x_0 + \int_{t_0}^{t} v(\tau)\, d\tau = x \end{aligned}$$

This will cause no problem since the two systems are equal. The second part is when the velocity is negative. The two definitions become:

$$\begin{aligned} x &= x_0 + \int_{t_0}^{t} v(\tau)\, d\tau \\ \hat{x} &= x_0 + \int_{t_0}^{t} |v(\tau)|\, d\tau = \\ &= x_0 - \int_{t_0}^{t} v(\tau)\, d\tau \end{aligned}$$

Here we can clearly see that the two definitions differ. The effect of this can be shown by putting x and $\hat{x}$ in the definition of the perturbation (2.2). We call the perturbation z_1 and the model of it $\hat{z}_1$.

$$\begin{aligned} z_1 &= \Lambda \cos(\omega x + \varphi) \\ &= \Lambda \cos\left(\omega(x_0 + \int_{t_0}^{t} v(\tau)\, d\tau) + \varphi\right) \\ \hat{z}_1 &= \Lambda \cos(\omega \hat{x} + \varphi) \\ &= \Lambda \cos\left(\omega(x_0 - \int_{t_0}^{t} v(\tau)\, d\tau) + \varphi\right) \end{aligned}$$

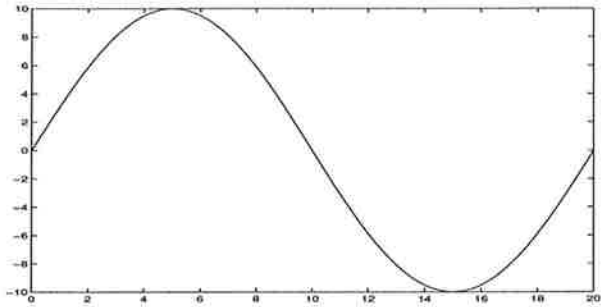
Using the relation $\cos(-\alpha) = \cos(\alpha)$ we get:

$$\begin{aligned} z_1 &= \Lambda \cos\left(\omega \int_{t_0}^{t} v(\tau)\, d\tau) + \omega x_0 + \varphi\right) \\ \hat{z}_1 &= \Lambda \cos\left(\omega \int_{t_0}^{t} v(\tau)\, d\tau) - \omega x_0 - \varphi\right) \end{aligned}$$

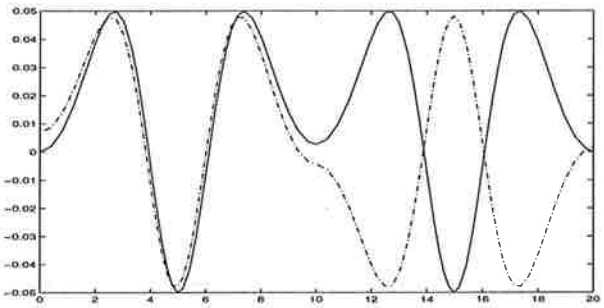
From these two equations it can be seen that z_1 and $\hat{z}_1$ have the same evolution but $\hat{z}_1$ has a phase shift of $2(\omega x_0 + \varphi)$.

How great this phase shift is can not be predicted. We have performed a 'worst case simulation' where the compensator is composed only of the primary system, to show how the model behaves for positive and negative values. The result is visualized in figure 2.1.

We clearly see that when the desired velocity is positive the estimated value of the perturbation follows closely to the real perturbation. But as soon



a) Desired velocity, $v_d(t)$



b) Real- and estimated perturbation, $z_1(t)(-)$ and $\hat{z}_1(t)(-\cdot)$

Figure 2.1 Worst case simulation with the real perturbation and and the calculated perturbation from the primary model.

as the sign changes to negative the observer gets a phase shift in difference from the accurate disturbance. This phase shift can be calculated. We have $v(t) = 10\sin(\frac{\pi}{10}t)$, $\omega = \frac{3\pi^2}{200}$ and $\varphi = -\frac{\pi}{2}$. x_0 can be calculated as

$$x_0 = \int_0^{t_0} v(\tau)\, d\tau = \frac{200}{\pi}$$

and thus the phase shift equals

$$2(\omega x_0 + \varphi) = 5\pi$$

A phase shift of 3π corresponds to an inversion of the disturbance as can be seen in figure 2.1. In this case the observer does not contain the correction terms which will be presented in the next chapter, which is why the error persists throughout the whole period when $v < 0$. If we would have run the real observer we would get a small peak at the sign change, but the observer would rapidly adjust to correct the error. This can be seen in chapter 5 in the simulations where the velocity crosses zero. This model error also explains partly the peaks in the real time experiments in similar situations.

The theory makers, Canudas de Wit and Praly, are presently working on a way to avoid this problem, which will most likely be presented in their report to be published later.

We would like to stress the fact that this error was not discovered until after we had completed all our experiments. Our work has thus been based on the model. Even if the model is not correct for negative values the compensation does reduce the error for all velocities.

3. Compensation Design

In this chapter we will derive an adaptive control for the model described in the previous chapter for the compensation of eccentricity. The idea is due to Carlos Canudas de Wit from the Laboratoire d'Automatique de Grenoble, and Laurent Praly from Ecole des Mines de Paris, whose work has not yet been published.

3.1 Observer Design

First we define the control, supposed to track the desired velocity v_d as

$$u = J\dot{v}_d - k_v(t)(v - v_d) - \hat{z}_1, \tag{3.1}$$

where $\hat{z}_1$ is the output of the following dynamic system

$$\nabla\hat{z}_1 = \hat{z}_2 + k_1(z_1 - \hat{z}_1) \tag{3.2}$$

$$\nabla\hat{z}_2 = -\hat{\theta}\hat{z}_1 + k_2(z_1 - \hat{z}_1) - \lambda\bar{z}_1\nabla\hat{\theta} \tag{3.3}$$

$$\nabla\bar{z}_1 = -\frac{1}{\lambda}(\mu\bar{z}_1 - \hat{z}_1) \tag{3.4}$$

$$\nabla\hat{\theta} = -\gamma\bar{z}_1(z_1 - \hat{z}_1) \tag{3.5}$$

with

$$z_1 = J\dot{v} - u = a_1\cos(\omega x) + a_2\sin(\omega x) \tag{3.6}$$

The parameters k_1, k_2, λ, μ and γ are positive constants, the property of $k_v(t)$ is presented later. The constants k_1 and k_2 are used to stabilize the observer if we set $\hat{\theta}$ to be constant. They move the poles from the imaginary axis to the negative half-plane. Note that with the assumption that $\dot{v}_d$ and v_d are bounded and continuous, with this controller we get a closed loop system whose dynamics are described by an ordinary differential equation of which the right hand side is continuous. To get the adaption law we observe the error equations in the closed loop system. We first introduce the error variables.

$$e = v - v_d \tag{3.7}$$

$$\tilde{z}_1 = z_1 - \hat{z}_1 \tag{3.8}$$

$$\tilde{\theta} = \theta - \hat{\theta} \tag{3.9}$$

The objective now is to find a transfer function between $\tilde{z}_1$ and $\bar{z}_1\tilde{\theta}$ of the form:

$$\tilde{z}_1 = G(\nabla)\left\{-\bar{z}_1\tilde{\theta}\right\}$$

where $\bar{z}_1$ is a filter, and $G(\nabla)$ is defined as:

$$G(\nabla) = \frac{\lambda\nabla + \mu}{\nabla^2 + k_1\nabla + (k_2 + \theta)}$$

The advantage of this function is that it can be written in the form:

$$\nabla\xi = A\xi - B\bar{z}_1\tilde{\theta} \tag{3.10}$$
$$\tilde{z}_1 = C\xi \tag{3.11}$$

where:

$$A = \begin{pmatrix} 0 & 1 \\ -(k_2+\theta) & -k_1 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad C^T = \begin{pmatrix} \lambda \\ \mu \end{pmatrix} \tag{3.12}$$

Since A in (3.12) is strictly Hurwitz, i.e. all its poles' real parts are negative, for any positive value of k_2 (note that $\theta > 0$) and k_1, it follows that we can find a symmetric positive definite matrix, P, such that:

$$A^T P + P A = -I$$

We then let λ and μ in C in (3.12) be defined as:

$$C = PB \ . \tag{3.13}$$

According to the Lyaponov theorem the system is stable if we can find a non negative function $V(\xi)$ so that its derivate $\nabla V(\xi)$ is non positive. if we choose

$$V = \xi^T P\xi + \gamma^{-1}\tilde{\theta}^2 \tag{3.14}$$

the derivate can then be calculated as

$$\begin{aligned} \nabla V &= -\xi^T\xi - 2(\xi^T PB)\bar{z}_1\tilde{\theta} + \gamma^{-1}(\nabla\tilde{\theta})\tilde{\theta} \\ &= -\xi^T\xi - 2\tilde{z}_1\bar{z}_1\tilde{\theta} - \gamma^{-1}(\nabla\hat{\theta})\tilde{\theta} \\ &= -\xi^T\xi - 2\left[\tilde{z}_1\bar{z}_1 + \gamma^{-1}\nabla\hat{\theta}\right]\tilde{\theta} \ . \end{aligned} \tag{3.15}$$

From equation (3.15) we can see that if we set $\nabla\hat{\theta} = -\gamma\bar{z}_1\tilde{z}_1$ we get a non positive function

$$\nabla V = -\xi^T\xi \tag{3.16}$$

which is what the Lyaponov theorem implied.

Now we have three error equations:

$$J\dot{e} = -k_v(t)e + \tilde{z}_1 \tag{3.17}$$
$$\tilde{z}_1 = G(\nabla)\left\{-\bar{z}_1\tilde{\theta}\right\} \tag{3.18}$$
$$\nabla\tilde{\theta} = \gamma\bar{z}_1(\tilde{z}_1) \tag{3.19}$$

In order to get our observer to correspond to the transfer function $G(\nabla)$ we get the term $-\lambda\bar{z}_1\nabla\hat{\theta}$ in equation (3.3).

We have now defined the observer system, but we see that this observer requires the acceleration to be measurable. To avoid this we first transform the system to the time domain, which gives

$$\begin{aligned}
\dot{\hat{z}}_1 &= |v|\left[\hat{z}_2 + k_1(J\dot{v} - u - \hat{z}_1)\right] \\
\dot{\hat{z}}_2 &= |v|\left[-\hat{\theta}\hat{z}_1 + k_2(J\dot{v} - u - \hat{z}_1) - \lambda\bar{z}_1\nabla\hat{\theta}\right] \\
\dot{\bar{z}}_1 &= |v|\left[-\frac{1}{\lambda}(\mu\bar{z}_1 - \hat{z}_1)\right] \\
\dot{\hat{\theta}} &= |v|\left[-\gamma\bar{z}_1(J\dot{v} - u - \hat{z}_1)\right]
\end{aligned} \tag{3.20}$$

Further to avoid $\dot{v}$ we note that

$$\frac{d}{dt}\left\{|v|\,v^i\right\} = (i+1)\,|v|v^{i-1}\dot{v}; \quad \forall i = 1, 2, 3, \ldots$$

We now introduce the new variables

$$\begin{aligned}
\hat{\zeta}_1 &= \hat{z}_1 - \frac{k_1 J}{2}|v|v \\
\hat{\zeta}_2 &= \hat{z}_2 - \frac{k_2 J}{2}|v|v - \frac{\gamma\lambda J}{2}|v|v\bar{z}_1^2 \\
\bar{\zeta}_1 &= \bar{z}_1 \\
\hat{\vartheta} &= \hat{\theta} + \frac{\gamma J}{2}\bar{z}_1|v|v
\end{aligned} \tag{3.21}$$

Deriving the equations above with the definition in (3.20) gives

$$\begin{aligned}
\dot{\hat{\zeta}}_1 =& |v|\left[\hat{z}_2 - k_1(u + \hat{z}_1)\right] \\
\dot{\hat{\zeta}}_2 =& |v|\Big[-(k_2 + \hat{\theta})\hat{z}_1 - k_2 u - \gamma\lambda\bar{z}_1^2(u + \hat{z}_1) + \\
&+ \gamma J|v|v\bar{z}_1(\mu\bar{z}_1 - \hat{z}_1)\Big] \\
\dot{\bar{\zeta}}_1 =& |v|\left[-\frac{1}{\lambda}(\mu\bar{z}_1 - \hat{z}_1)\right] \\
\dot{\hat{\vartheta}} =& |v|\left[\gamma\bar{z}_1(u + \hat{z}_1) - \frac{\gamma J}{2\lambda}|v|v(\mu\bar{z}_1 - \hat{z}_1)\right]
\end{aligned} \tag{3.22}$$

where, we can notice that the $\dot{v}$ dependence is no longer valid. With this controller we get a closed loop system whose dynamics are described by an ordinary differential equation whose right hand side is continuous.

3.2 Stability Analysis

In this section present the stability analysis for the system derived above. From the error equations (3.17 - 3.19) we can see that we have a coupled s-domain equation since equation (3.18) and (3.19) involves parameters in the t-domain and equation (3.17) has as an input a signal parameterized in the s-domain. Figure 3.2 is showing the inter-block connection between these two systems. We suppose in the analysis that v_d and $\dot{v}_d$ are bounded and continuous.

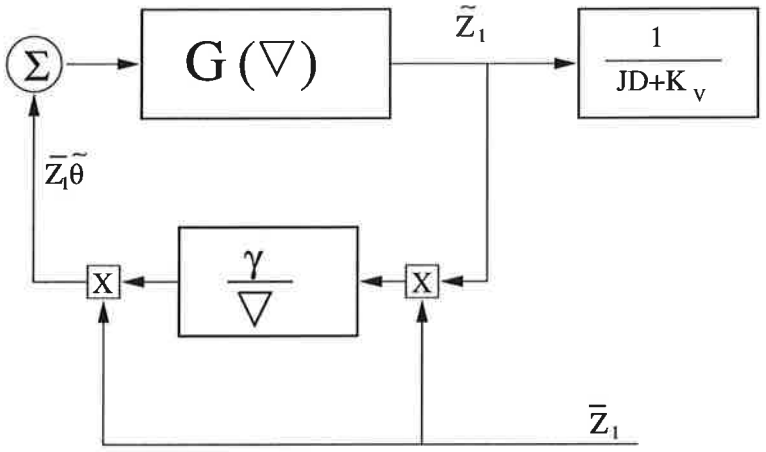


Figure 3.1 Block scheme of the coupled s-domain/t-domain system

The Lyaponov function that is presented in equation (3.14) does not show that the function is asymptotically stable since $\nabla V(\xi)$ is not strictly negative. We will here present a theorem with some constraints of v_d and $k_v(t)$.

Theorem:

Consider the system (2.1)-(2.2). Consider the dynamic feedback defined by the equation set (3.1- 3.6). Let the control gains $k_1 > 0, k_2 > 0$, and λ, μ be defined by (3.13). Then all the internal signal of the system are bounded and, in addition, the velocity tracking error e tends to zero, if the desired velocity v_d is such that:

(i) $J\dot{v}_d(t) + k_v(t)v_d(t)$ has no limit when t tends to infinity, and

(ii) $\dot{k}_v$ and $\ddot{v}_d$ are bounded.

Proof:

From what we have observed on the closed loop system, to any initial condition corresponds a unique solution. Let $[0, T)$ be its right maximal interval of definition in the t-domain. It corresponds functions in the s-domain defined on an interval $[0, S_{max})$ where:

$$S_{max} = \lim_{t \to T} \int_0^t |v(\tau)| d\tau \ (\leq +\infty) \tag{3.23}$$

Let us show that T must be infinite. Considering equations (3.14) - (3.16) we conclude that since ∇V is non positive, V is bounded on $[0, S_{max})$ and therefore on $[0, T)$ it follows that ξ, $\tilde{z}_1$ and $\hat{\theta}$ are bounded on $[0, T)$ and from (3.17), the same holds for e. Since the external signals $\dot{v}_d$, v_d and z_1 are bounded, we conclude that all the functions are bounded. So T must be infinity.

From (3.10) we get that $\nabla\xi$ is bounded on $[0, S_{max})$ and as a result ξ is uniformly continuous. From Barbalat's Lemma, this implies that, if $S_{max} = \infty$, i.e. $|v|$ is not summable in the time domain, then ξ and therefore $\tilde{z}_1$ converge to 0 as t goes to infinity. From (3.17) the same holds for e. So when $|v|$ is not summable in the time domain, we have:

$$\lim_{t \to +\infty} \{v(t) - v_d(t)\} = 0 \ . \tag{3.24}$$

It remains to show that $S_{max} = \infty$ when $J\dot{v}_d(t) + k_v(t)v_d(t)$ has no limit and $\dot{k}_v$ and $\ddot{v}_d$ are bounded. This can be proven by saying that if S_{max} is assumed to be bounded it implies that $u+k_v(t)v+\hat{z}_1$ and therefore $J\dot{v}_d + k_v(t)v_d$ has a limit (see (3.1)).

First, we observe from (3.1), (3.6) and (3.20) that, if $\dot{k}_v$ and $\ddot{v}_d$ are bounded, then $\dot{v}$ and $\ddot{v}$ are bounded. With Barbalat's Lemma, when v is summable, v and $\dot{v}$ tend to zero. Then, from (3.20), $\dot{\hat{z}}_1$ is summable. This results in that $\hat{z}_1$ converges. Finally, from (3.6), since $\dot{v}$ tends to zero and x converges, u converges. So we do get that $u + k_v v + \hat{z}_1$ converges.

Comments: There are two cases of practical interest where $J\dot{v}_d(t) + k_v(t)v_d(t)$ has no limit.

- *Constant velocity profiles.* In this case, it is mandatory to select a time-varying $k_v(t)$ such that $k_v(t)v_d$ has no limit (i.e. $k_v(t) = k_V + \sin\omega t$).
- *Constant velocity gain.* If k_v is constant, thus only time-varying velocity trajectories $v_d(t)$ can ensure that the tracking error goes to zero. Time-periodic trajectories may belong to this class.

We will consider velocities that vary and therefore we choose k_v to be constant.

3.3 There is Something Rotten in the State of Denmark – Part II

The observer that has been presented in this chapter has been proven to be globally stable with an error that tends towards zero. We would only like to point out that this is only true if the perturbation is of the form presented in section 2.2. Since the model is wrong if the velocity is negative, we will get an error each time the velocity changes signs.

4. Friction

To fully understand the disturbance rejection we are presenting here, we have to get a glimpse of its origin. In our case the disturbance is due to friction variations caused by the contact between a cylinder and an eccentric wheel.

Friction has been the subject for several research topics over the past years. A lot of new models have evolved which define friction and its non-linearities. Yet, friction's complexity is not entirely defined. We here present a very brief description of the LuGre model, which is what we have used to model friction in our experiments.

4.1 The LuGre Model

The LuGre (Lund/Grenoble) model was developed as a joint project between the Control Department at Lund Institute of Technology and Laboratoire d'Automatique de Grenoble at ENSIEG during the 1990s, by Karl Johan Åstrom and Henrik Olsson from the former, and Carlos Canudas de Wit and Pablo Lichinsky from the latter. The model has been presented in Canudas de Wit *et al.* (1995) and Olsson (1996)

The model starts from the description of the friction force caused by solid-to-solid contact. The surfaces' asperities are described by several elastic bristles with different lengths, visualized in figure 4.1.

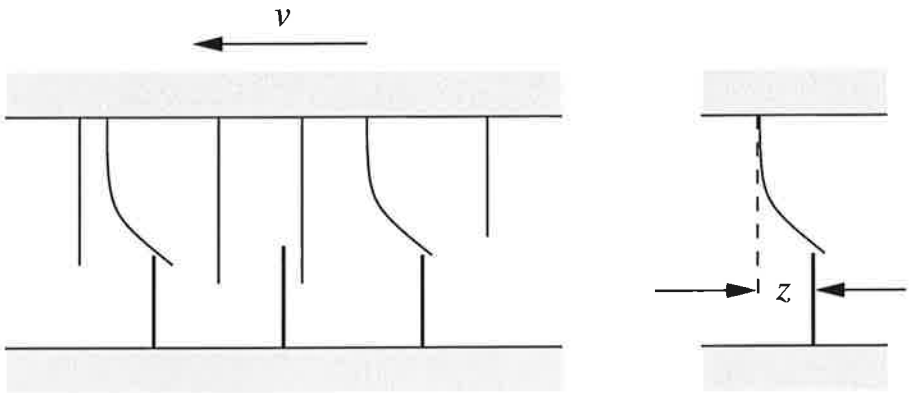


Figure 4.1 Left: The contact between two solid surfaces in relative movement can be visualized by bending bristles. Right: The average deflection of the bristles is denoted z. Source: Olsson (1996).

When the two surfaces are in contact with an applied tangential force the bristles bend, which provokes the friction force. If the applied force is big enough the bristles will slip off each other, and come in contact with new bristles.

The average deflection of the bristles is denoted by z, and modeled by

$$\frac{dz}{dt} = v - \frac{|v|}{g(v)}z \tag{4.1}$$

where v is the relative velocity between the two surfaces and $g(v)$ a function which is defined later. The first term gives a deflection that is proportional

to the integral of the relative velocity and the second term asserts that the deflection z approaches the value

$$z_{SS} = g(v)\frac{v}{|v|} = g(v)\text{sign}(v) \tag{4.2}$$

in steady state, i.e., when v is constant.

The friction force is generated by the bending of the bristles, it is proportional to the average deflection and the rate of change of deflection. The third factor that makes up the friction force is caused by the viscosity of the lubricant. The total friction force looks like

$$F = \sigma_0 z + \sigma_1(v)\frac{dz}{dt} + f(v) \tag{4.3}$$

where $\sigma_0 > 0$ is the stiffness and $\sigma_1(v) > 0$ a velocity dependent damping coefficient.

The damping coefficient σ_1 is essential to make the system well behaved in transitions between sticking and sliding. The simplest parameterization is for linear damping which may be limiting in some aspects, but in our experiments we use

$$\sigma_1(v) = \sigma_1 \tag{4.4}$$

The function f in (4.3) depends on the type of friction interface, but is in general simply linearly dependent on the relative velocity

$$f(v) = F_v v \tag{4.5}$$

where F_v is the coefficient of viscous friction.

The steady-state friction force is given when $dz/dt = 0$, and thus becomes

$$F_{SS}(v) = \sigma_0 g(v)\text{sign}(v) + f(v) \tag{4.6}$$

We here see the sign-function which in friction in general is referred to as the Coulomb effect, see figure 4.2, which we will also include in the simulations later on.

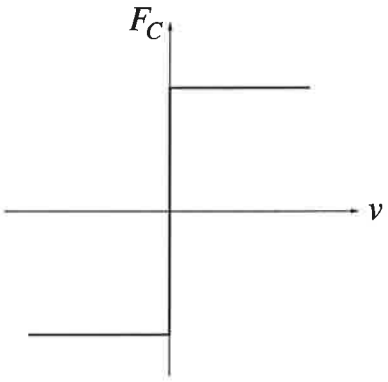


Figure 4.2 Friction as a function of velocity for Coulomb's model

The function $g(v)$ is always positive and includes the effect of the lubricant and other material properties. It is determined to adjust to the system.

Decreasing $g(v)$ implies that the dynamics are faster. But there are some restrictions in order to be coherent with the static models. The steady-state then determines $g(v)$ to

$$g(v) = \frac{F(v)}{\sigma_0} = \frac{1}{\sigma_0}\left(F_C + (F_S + F_C)e^{(v/v_S)^2}\right) \tag{4.7}$$

We have now given a simplified description of the LuGre model which in a complete form looks like

$$\boxed{\begin{aligned} \frac{dz}{dt} &= v - \frac{|v|}{g(v)}z \\ g(v) &= \frac{1}{\sigma_0}\left(F_C + (F_S + F_C)e^{-(v/v_S)^2}\right) \\ F_{LuGre} &= \sigma_0 z + \sigma_1 \frac{dz}{dt} + F_v v \end{aligned}} \tag{4.8}$$

5. Simulations

We have now completed the presentation of the new model and shown its theoretical stability analysis, we will further present the simulations performed to test the developed observer.

The compensator we evaluate is general and valid for many systems with periodic disturbances. It finds the period and amplitude of a system perturbation that is dependent of position and compensates it. Therefore simulations with such a perturbation were tested just to justify the theory. Then, because we are interested in compensating friction, a second set of simulations were made with a Coulomb friction model. The last simulations were made to be as close as possible to reality, with a LuGre friction model and a dynamic friction compensator added to our eccentricity compensator.

The scheme we used to simulate our system were in a block system made in SIMULINK^{TM} application in MATLAB^{TM}, see appendix A. For all simulations we use the following values for the compensator:

$$\begin{aligned} k_1 &= 1, \quad k_2 = 0.25, \\ \gamma &= 1, \quad \mu = 1, \\ \lambda &= 2. \end{aligned}$$

The compensation is added to a proportional controller with

$$k_v = 100,$$

and the system inertia is

$$J = 0.0022.$$

All values are chosen to be close to the real time experiments that will be done after the simulations. For all simulations we let the system work without compensation the first ten seconds. The proceeding ten we compensate.

5.1 Simulations with a Periodic Perturbation

The system, which the perturbation acts on, is described by

$$J\dot{v} = u - d(x)$$

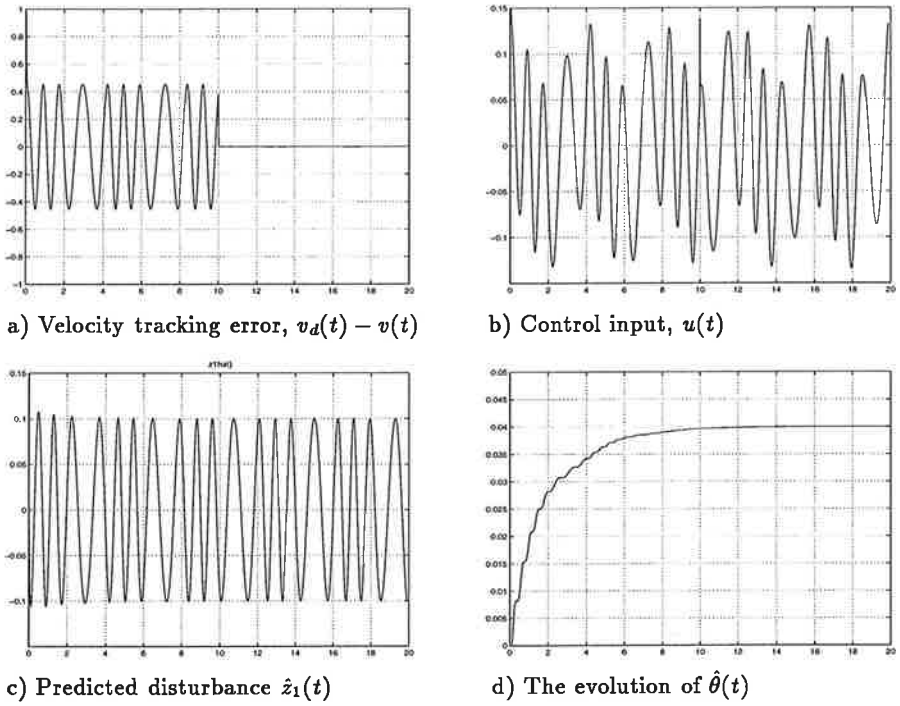
In our first simulation we use a simple sinusoidal perturbation which is described by

$$d(x) = \Lambda \cos(\omega x + \varphi) \tag{5.1}$$

where

$$\omega = 0.2, \quad \Lambda = 0.1, \quad \varphi = 3.$$

We start by testing the model for positive velocities. According to theory this perturbation will be compensated by the observer with an error that tends



a) Velocity tracking error, $v_d(t) - v(t)$

b) Control input, $u(t)$

c) Predicted disturbance $\hat{z}_1(t)$

d) The evolution of $\hat{\theta}(t)$

Figure 5.1 Simulation with a periodic disturbance. Reference $v_d = 30 + 10\sin(\frac{\pi}{2}t)$ rad/s.

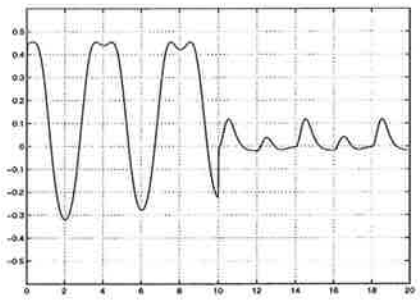
towards zero. A simulation for a sinusoidal positive reference velocity is shown in figure 5.1.

The result in figure 5.1 clearly shows that the error goes to zero and the compensator finds the frequency, $\hat{\theta}$ goes to 0.04 which is the square of the perturbation frequency, ω, which follows the theory as shown in chapter 2.

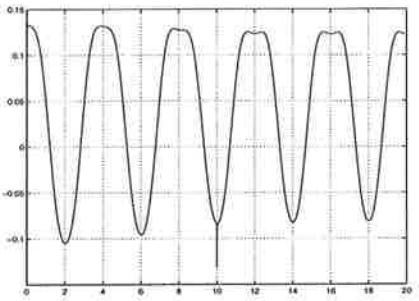
The time it takes to find the true value of $\hat{\theta}$ can be made faster or slower by changing the value of γ. A higher γ gives a better result but is more sensitive for noise and small system changes. We have performed simulations where γ has been several times bigger, which gives a much faster system, but in the real-time experiments the results got unstable for higher γ-values than are presented here.

Once we found that the observer was working as predicted for positive velocities, we tested it with a sinusoidal signal which crossed zero, see figure 5.2. The little peaks we can see during the last ten seconds in the velocity tracking error in subfigure a) are caused by the sign reversal in the reference signal. This error has its origin in the problem of the model for negative velocities as described in section 2.3. However we can still see that the error is a lot smaller during the last ten seconds when the compensation is on. This means that the observer is fast enough to make up for the problem and even though not entirely reject, still reduce the influence of the perturbation. The reasoning in section 2.3 means however that the size of the peaks may vary depending on the amplitude, the frequency and the phase lag. This means that the errors could have been bigger or smaller depending on how we choose these variables.

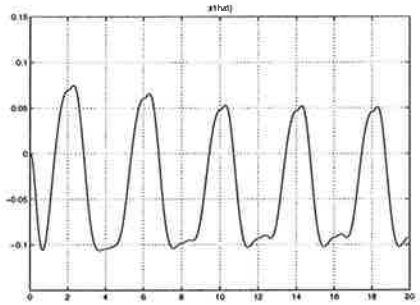
The parameters do not compose a worst case scenario, nor a best case. In our simulation $x_0 = 40/\pi$ for the passage from positive to negative, and $x_0 = 0$ for the passage from negative to positive, which together with $\omega = 0.2$



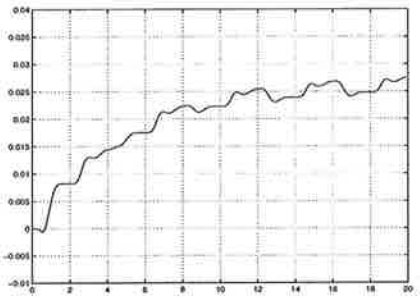
a) Velocity tracking error, $v_d(t) - v(t)$



b) Control input, $u(t)$



c) Predicted disturbance $\hat{z}_1(t)$



d) The evolution of $\hat{\theta}(t)$

Figure 5.2 Simulation with a periodic disturbance. Reference $v_d = 10\sin(\frac{\pi}{2}t)$ rad/s.

and $\varphi = 3$ corresponds to a phase shift of $\approx -\pi/2$ and ≈ 0 respectively. This is confirmed in figure 5.2 where can see that the peaks are relatively bigger for $t = 10$, 14 & 18 than for $t = 12$ & 16.

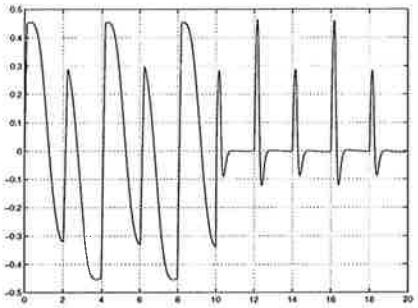
5.2 Simulations Based on Coulomb Friction

We continue by describing the perturbation more like a cause of friction by introducing the Coulomb effect described by the sign-function, see figure 4.2. The perturbation thus looks like

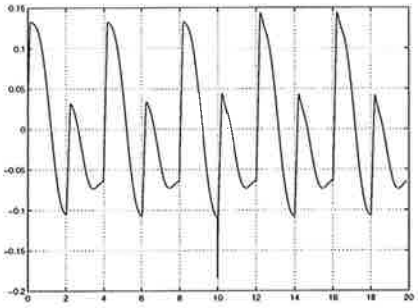
$$d(\boldsymbol{x}) = \Lambda \operatorname{sign}(v)\cos(\omega \boldsymbol{x} + \varphi). \tag{5.2}$$

where the oscillating perturbation is treated like a simple model of friction plus the periodicity introduced in equation (5.1). In the simulation we use a rate limiter after the sign-function. The purpose of this is to avoid that the velocity fluctuates between positive and negative values, since $\operatorname{sign}(0) = \varepsilon$, where $-1 < \varepsilon < 1$.

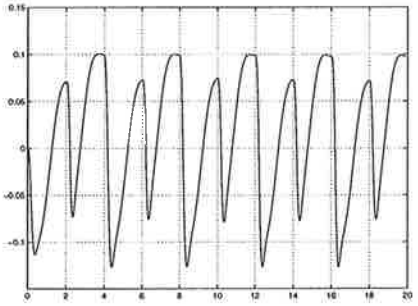
We expect the compensator to have certain difficulties with the fast changes introduced by the sign-function, and as we can see in figure 5.3, the output lags the reference in each such situation. The peaks in the velocity tracking error are, compared to figure 5.2, a lot more important. This is due to the fact that the observer misunderstands the change of signs and overcompensates it. The observer sees the change of signs as a high frequency perturbation and $\hat{\theta}$ tries to find this frequency. This introduces jumps in $\hat{\theta}$ at every sign change. Due to the correction terms in the observer the compensation soon finds its way back to its periodic behavior again. But we can observe the big jumps in the control input as well as in the predicted perturbation.



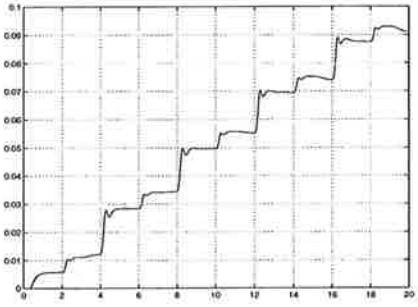
a) Velocity tracking error, $v_d(t) - v(t)$



b) Control input, $u(t)$



c) Predicted disturbance $\hat{z}_1(t)$



d) The evolution of $\hat{\theta}(t)$

Figure 5.3 Simulation based on Coulomb friction. Reference $v_d = 10\sin(\frac{\pi}{2}t)$ rad/s.

5.3 Simulations Based on the LuGre Model

The LuGre-friction model was used for the last simulation. This model is closer to reality than the Coulomb-model and has also the advantage that its first derivate is not singular. Our friction-model now has the form

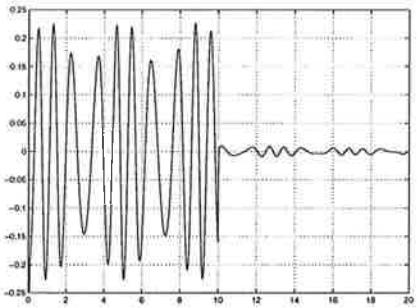
$$d(x) = F_{LuGre}\,(1 + \Lambda\cos(\omega x + \varphi)). \tag{5.3}$$

In difference from equation (5.2) the friction is not simplified to the Coulomb effect but takes into account all the parts of friction discussed in chapter 4. The friction can be separated into a fixed part, F_{LuGre}, and a variant part, $F_{LuGre}\,\Lambda\cos(\omega x + \varphi)$. The static part is compensated by the algorithm in equation (4.8) and the dynamic part is compensated by the AEC. The block-scheme is presented in appendix A.

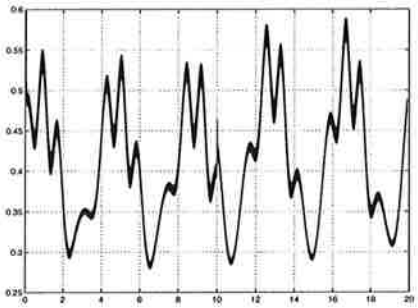
In figure 5.4 we show the simulation results for a reference velocity $v_d = 30 + 10\cos(\frac{\pi}{2}t)$. The effects of that the LuGre-model is richer than previous models that has been simulated can be observed; $\hat{\theta}$ has a slightly periodic evolution and we have a small velocity tracking error because of that the term F_{LuGre} is dependent of velocity. This explains the changes in amplitude in the predicted disturbance. The steady state function of the LuGre friction model is the Coulomb friction model.

When the velocity crosses zero, see figure 5.5, we notice the same lags as for the Coulomb friction. We can also remark that the jumps in $\hat{\theta}$ are less significant than in the simulations with the sign-function. This is due to that the change of signs is 'smoother'.

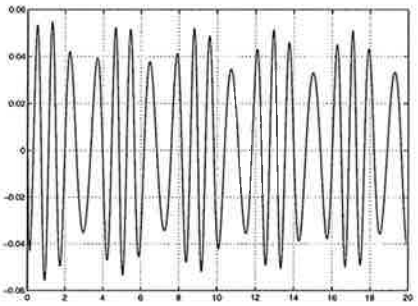
The problem associated with the velocity reversals caused us to try to find a possible way to get around this, the result of this research is presented next.



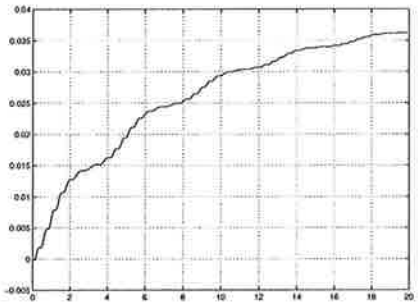
a) Velocity tracking error, $v_d(t) - v(t)$



b) Control input, $u(t)$

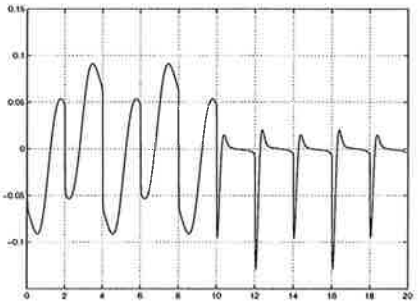


c) Predicted disturbance $\hat{z}_1(t)$

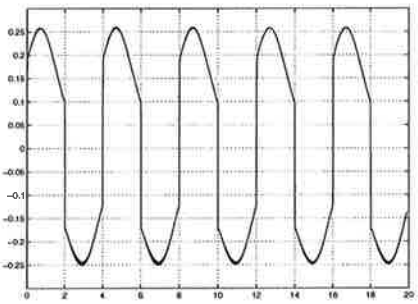


d) The evolution of $\hat{\theta}(t)$

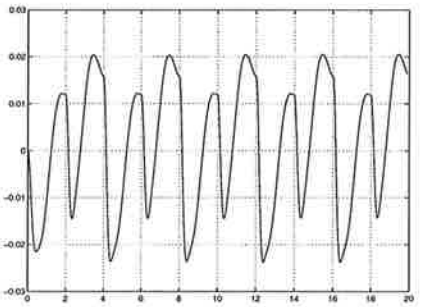
Figure 5.4 Simulation including F_{LuGre}. Reference $v_d = 30 + 10\sin(\frac{\pi}{2}t)$ rad/s.



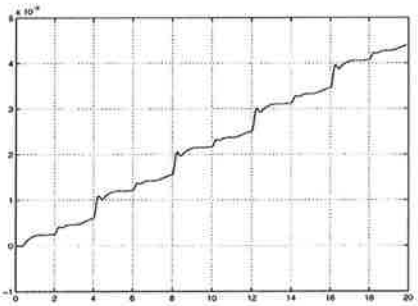
a) Velocity tracking error, $v_d(t) - v(t)$



b) Control input, $u(t)$



c) Predicted disturbance $\hat{z}_1(t)$

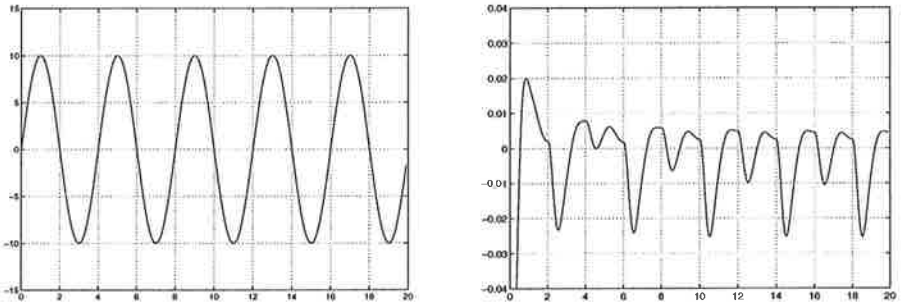


d) The evolution of $\hat{\theta}(t)$

Figure 5.5 Simulation including F_{LuGre}. Reference $v_d = 10\sin(\frac{\pi}{2}t)$ rad/s.

6. An Improved Observer

We have until now seen that the adaptive eccentricity compensator does not fully compensate the situations where the velocity changes signs. The estimated perturbation does not follow the change and it lags a bit before it corrects itself, which can be seen in the peaks in figure 6.1.



a) Desired velocity $v_d(t)$ b) Perturbation error, $d - \hat{z}_1$

Figure 6.1 Simulation with single observer with $d = \Lambda\cos(\omega x + \Phi)$. Reference $v_d = 10\cos(\frac{\pi}{2}t)$ rad/s.

The problem is due to several reasons

- the fact that the model is wrong for negative velocities
- because of the non-linearities in friction, also when $v = 0$

In this section we will present an alternative solution to this problem. The model error still remains a problem in the observer presented next, but we can nevertheless see major improvements.

6.1 The Parallel Observer

We use a heuristic idea based on the assumption that the prediction is linear in the velocity. Therefore we use two observers, one based on $v_+ = v + v_{constant}$ and the other based on $v_- = v - v_{constant}$, each working in the domains where we do not encounter the problems with velocity reversals. Then the mean value of these two observations is taken as the compensation. The new equations look like:

$$\begin{aligned}
\dot{\hat{z}}_{1+} &= |v_+|\,[\hat{z}_2 + k_1(J\dot{v}_+ - u - \hat{z}_{1+})] \\
\dot{\hat{z}}_{2+} &= |v_+|\left[-\hat{\theta}_+\hat{z}_{1+} + k_2(J\dot{v}_+ - u - \hat{z}_{1+}) - \lambda\bar{z}_{1+}\nabla\hat{\theta}_+\right] \\
\dot{\bar{z}}_{1+} &= |v_+|\left[-\frac{1}{\lambda}(\mu\bar{z}_{1+} - \hat{z}_{1+})\right] \\
\dot{\hat{\theta}}_+ &= |v_+|\,[-\gamma\bar{z}_{1+}(J\dot{v}_+ - u - \hat{z}_{1+})]
\end{aligned} \tag{6.1}$$

$$
\begin{aligned}
\dot{\hat{z}}_{1-} &= |v_-|\,[\hat{z}_{2-} + k_1(J\dot{v}_- - u - \hat{z}_{1-})] \\
\dot{\hat{z}}_{2-} &= |v_-|\left[-\hat{\theta}_-\hat{z}_{1-} + k_2(J\dot{v}_- - u - \hat{z}_{1-}) - \lambda\bar{z}_{1-}\nabla\hat{\theta}_-\right] \\
\dot{\bar{z}}_{1-} &= |v_-|\left[-\frac{1}{\lambda}(\mu\bar{z}_{1-} - \hat{z}_{1-})\right] \\
\dot{\hat{\theta}}_- &= |v_-|\,[-\gamma\bar{z}_{1-}(J\dot{v}_- - u - \hat{z}_{1-})]
\end{aligned}
\tag{6.2}
$$

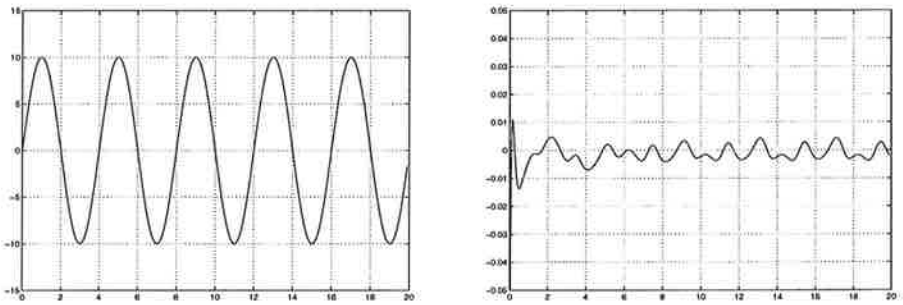
These calculations are executed in parallel with the previous observer, calculating the same equations but for the true v-value. We then use the following criteria to decide when to apply the new $\hat{z}_1$-values.

$$
\hat{z}_1 = \begin{cases} \hat{z}_1 & |v| > v_{limit} \\ (\hat{z}_{1+} + \hat{z}_{1-})/2 & |v| < v_{limit} \end{cases}
\tag{6.3}
$$

where v_{limit} is a constant about half of $v_{constant}$. In our simulations we have used

$$v_{constant} = 20 \quad \& \quad v_{limit} = 10$$

In figure 6.2 we observe how the new observer behaves to the perturbation described by equation 5.1.



a) Desired velocity, $v_d(t)$ b) Perturbation error, $d - \hat{z}_1$

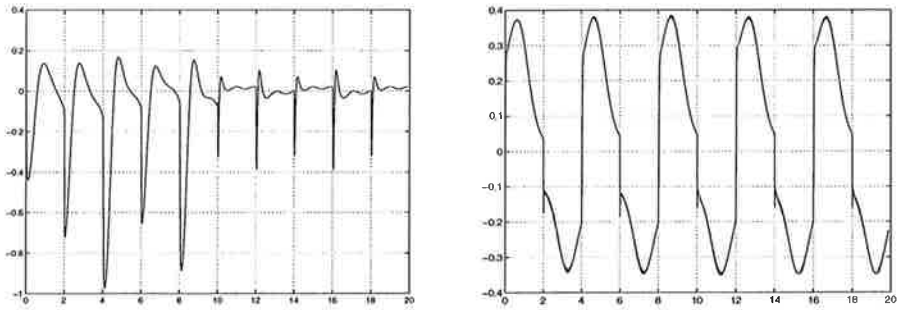
Figure 6.2 Simulation with parallel observer with $d = \Lambda\cos(\omega x + \Phi)$. Reference $v_d = 10\cos(\frac{\pi}{2}t)$ *rad/s*.

The error caused by the zero velocity crossing has been reduced substantially. We can still see small errors, these are a consequence of the model error. A reason why the parallel observer is still performing better is due to the fact that the correction terms followed by k_1 and k_2 above do keep their influence in the observer since they are not multiplied by 0 at the crossing of zero.

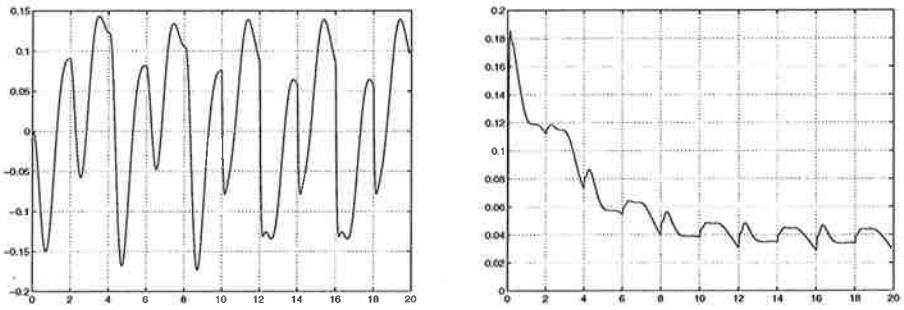
In figure 6.3 we compare the parallel observer with the single observer, where the single observer is running during the first 10 seconds, and the parallel observer is running for the last 10 seconds.

We see that this algorithm still lags but the lags are much more momentary. The fast change in the perturbation is caught up by the observer immediately. This can also be explained by the fact that k_1 and k_2 are multiplied by $v_{constant}$ instead of 0 at zero velocity.

We have studied each of $\hat{z}_{1+}, \hat{z}_{1-}, \hat{\theta}_+$ and $\hat{\theta}_-$ and seen their values are such that: $\hat{z}_{1+}$ has a smaller and $\hat{z}_{1-}$ has a bigger amplitude than $\hat{z}_1$, both with the



a) Velocity tracking error, $v_d(t) - v(t)$ b) Control input, $u(t)$



c) Predicted disturbance $\hat{z}_1(t)$ d) The evolution of $\hat{\theta}(t)$

Figure 6.3 Simulation comparing the single-, first 10 seconds, and the parallel observer with including F_{LuGre}. Reference $v_d = 10\cos(\frac{\pi}{2}t)$ rad/s .

same period. $\hat{\theta}_+$ and $\hat{\theta}_-$ find each a value around 0.02, added together we have as before $0.04 = \hat{\theta}$. The behavior of $\hat{\theta}_+ + \hat{\theta}_-$ differs a bit from $\hat{\theta}$, it tends first to high value and then it goes down 0.04 while $\hat{\theta}$ first go negative and then slowly goes up and approaches 0.04.

A statistical study of the values clearly shows the difference between the single AEC observer and the parallel one. We here present the velocity tracking error for the two observers in the histograms in figure 6.4. The concentration around zero error for the parallel observer can be seen very clearly.

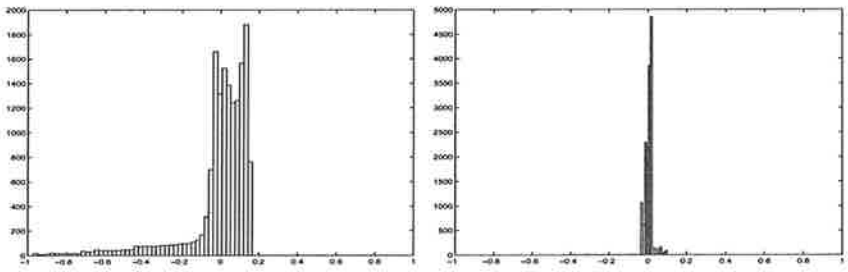


Figure 6.4 Histograms of the velocity tracking error for single (left) and parallel (right) AEC-observer.

7. Experimental Setup

We have now given a brief description of the theory which also has been tested in simulation. What has been presented until now is valid for all kinds of oscillating perturbations even though parameter values in the simulations have been adjusted to our specific experiment. The following part of our report should be seen as a general test of the model, even if it deals with our setup, where the periodic perturbation is due to friction, and our specific results.

This chapter presents the experimental setup itself and the interface for the execution of the experiments. The control law used, is also presented.

7.1 The Experimental Setup

To evaluate our results from theory and simulations we have performed a set of tests on an experimental system at the LAG (Laboratoire d'Automatique de Grenoble), see figure 7.1.

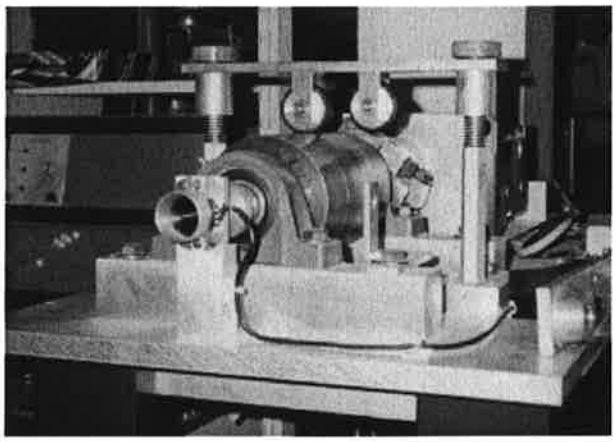


Figure 7.1 The experimental setup.

The setup consists of a motor taken from a robot that have worked for Renault in Paris. It is a highly linear motor and that way the experimental system is very close to the simulation model. The position of the rotor is measured by a reader with 120 000 values per round. The motor is turning a metal cylinder which on top has an eccentric wheel that has been visualized as the perturbation in the model in chapter 3. A simple scheme of the setup is presented in figure 7.2.

The top wheel is constrained by a constant normal force F_N. The total force is thus equal to F_N for the average radius of the wheel and is larger when the larger radius r_1 is in contact and smaller when the radius r_2 is in contact. The total load from the wheel onto the cylinder can be described as

$$F_N^{load} = F_N\left[1 + \frac{r_1 - r_2}{2}\cos(\bar{\omega}x_l + \varphi)\right] \tag{7.1}$$

where x_l is the load angle position (rad), φ is the phase shift (rad), and $\bar{\omega}$ is

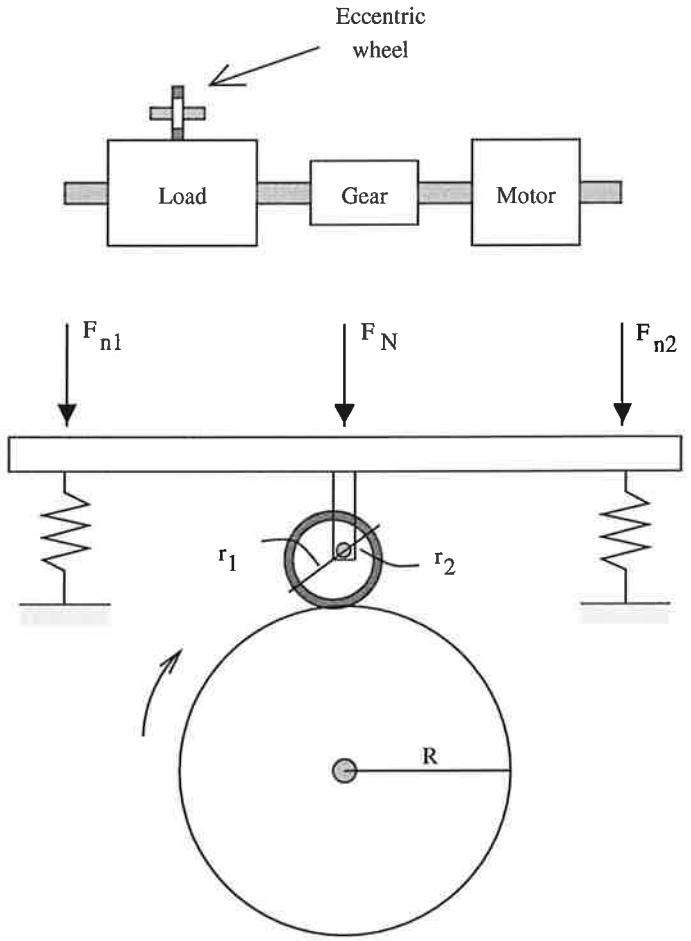


Figure 7.2 The experimental setup, top: front view, bottom: lateral view.

the dimensionless eccentricity frequency described by

$$\bar{\omega} = \frac{R}{r}; \quad r = \frac{r_1 + r_2}{2}$$

In real time experiments the frequency is however not considered as known. Between the rotor and the cylinder that is turning our eccentric wheel we have a gear-box that gives a scaling factor of 15.5. (15.5 rounds for the rotor equals one round for the cylinder).

The control law for the motor drive is given as

$$J\dot{v} = u - F_m : \qquad v = \dot{x} = \frac{dx}{dt} \tag{7.2}$$

where x is the angular position of the motor (rad), $J = J_m + J_l/n^2$ is the total system inertia (motor plus load) (Nm/s^2), u is the control torque input (Nm), and F_m is the motor friction torque at the ball bearings (Nm). F_m can be modeled as

$$F_m = \left[F_{Nm} + \frac{F_N^{load}}{n}\right] F_{LuGre}(z, \dot{z}, v) \tag{7.3}$$

where F_{Nm} is the average normal force on the ball bearings,F_{LuGre} and z are defined in equation (4.8). In equation (7.3) n is introduced which converts the forces to the side of the charge.

Model (7.2) can be rewritten as

$$J\dot{v} = u - F - \Lambda\cos(\omega x + \varphi) \tag{7.4}$$

with

$$F = \left[F_{Nm} + \frac{F_N}{n}\right](\bar{\sigma}_0 z + \bar{\sigma}_1 z + \bar{F}_v v) \tag{7.5}$$

$$= \sigma_0 z + \sigma_1 z + F_v v \tag{7.6}$$

$$\omega = \frac{\bar{\omega}}{n} = \frac{1}{n}\frac{R}{r} \tag{7.7}$$

$$\Lambda = \frac{F_N}{n}\frac{r_1 - r_2}{2}F_{LuGre}(z, \dot{z}, v) \tag{7.8}$$

where ω is the dimensionless frequency on the motor side, which is the same as the frequency on the load side but divided by the gear ratio n, and Λ is the amplitude of the perturbation.

For the model described in equation (2.2) the model is based on Λ, not as a variable depending on $F_{LuGre}(\cdot)$ but as a constant which we approximate to the steady state $F_{LuGre(SS)} = \mu_0\, sign(v)$.

$$\Lambda \approx \frac{r_1 - r_2}{2}\frac{F_N}{n}F_{LuGre(SS)} = \mu_0 \cdot sign(v) \tag{7.9}$$

where μ_0 replaces the function $g(v)$ in the LuGre friction model. For simplicity reasons we just use μ_0 which is the Coulomb friction normalized by the Normal force.

The parameters for all formulae are taken from the setup and are presented in table 7.1. The values have been calculated and presented in Canudas de Wit and Lischinsky (1997)

Friction parameters	Motor parameters	Wheel & cylinder characteristics
$F_C = 0.38\,[Nm]$	$J_m = 0.00196\,[Kg/m^2]$	$R = 6.0\,[cm]$
$F_S = 0.42\,[Nm]$	$J_l = 0.0125\,[Kg/m^2]$	$r_1 = 2.1\,[cm]$
$v_0 = 0.01\,[rad/s]$	$J = 0.0022\,[Kg/m^2]$	$r_2 = 1.9\,[cm]$
$\sigma_0 = 260.0\,[Nm\,s/rad]$	$K_t = 0.352\,[Nm/Amp]$	$r = 2.0\,[cm]$
$\sigma_1 = 0.6\,[Nm/rad]$	$n = 15.5$	
$\sigma_2 = 0.011\,[Nm\,s/rad]$	$P = 200\,[Watt]$	

Table 7.1 Friction, motor and load parameters.

The compensator we try to evaluate is done for perturbations which surround zero. This results in the fact that we have to compensate for all static friction before we start to use our eccentricity compensation. To do so we have used a dynamic compensator made in Lischinsky (1997) using the LuGre model, also using the values above.

7.2 The Interface

The control program has been written in C, see appendix B. Since the computer is not fast enough to do the control the compiled C-program has been transferred to a microcomputer called dSPACE^{TM}. To trace signals and command the system, we have used the softwares TRACE and COCKPIT respectively, that links the computer with dSPACE^{TM}. The cockpit interface allows us to start the experiment and to change variables in the program in dSPACE^{TM}. An image of the interface is shown in figure 7.3. TRACE plots all variables we want to analyze.

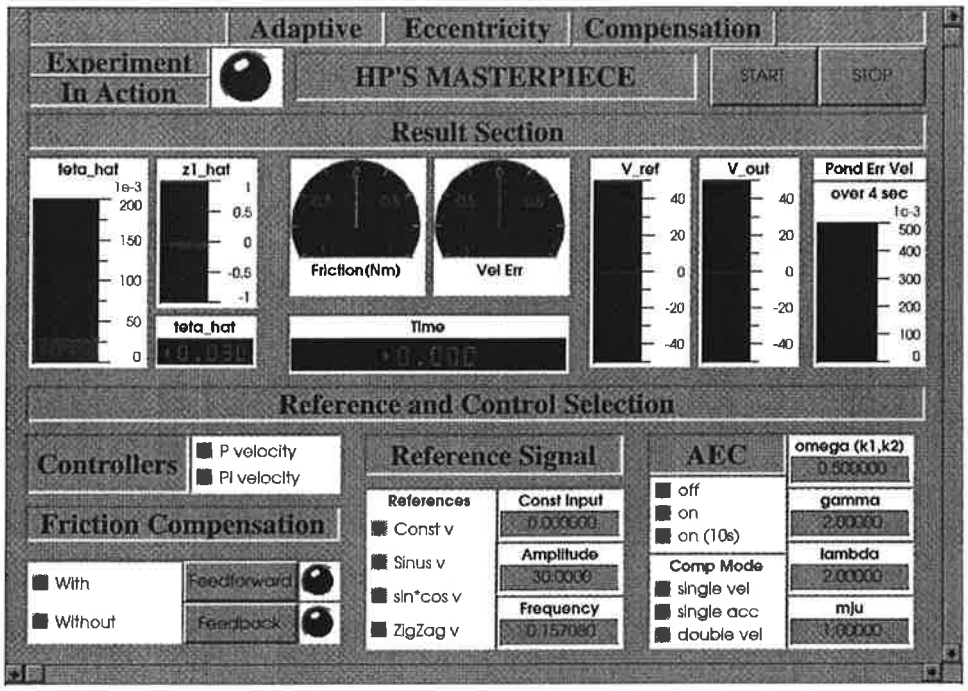


Figure 7.3 The cockpit interface, from where the program is launched.

8. Experimental Results

This chapter presents the experimental testing of the new AEC controller. We first present experimental results for different velocities showing the validation of the new model. This is done for constant and sinusoidal velocities and for sinusoidal velocities crossing zero. A comparative study with a regular PI-controller is also shown.

For the real time experiments, the sampling time could result in an unstable discrete time system when the motor is running for high velocities. The problem with reality is that there are other perturbations added to the one we try to compensate. In our case we found not only the eccentricity perturbation but also a high frequency perturbation with almost the same amplitude. Therefore all experimental results have been filtered by a first order Butterworth filter with a cut-off frequency at 25 Hz. The filter is chosen in order to remove only noise without influencing the major frequencies of the perturbation and resonance frequencies encountered by the system. This gives us a filter which looks like

$$H(s) = \frac{B(s)}{A(s)} = \frac{0.0592q + 0.0592}{q - 0.8816}$$

Another problem for the real time experiments which we did not encounter for the simulations is the scaling factor for the gear-box n, between the motor and the cylinder. This induced that the γ we used in simulations had to be substantially lower in the real time experiments. A higher γ implies that the AEC is faster but also that it is more sensitive. In the real time experiments we had to lower the γ when increasing the velocity, otherwise the results went unstable. It could have been a good idea to have a changing gamma dependent on the velocity. This idea was never implemented as an automatic change of γ but was done manually, where it varied between 1 and 5. Other than that, all observer values are the same as presented in chapter 5 about the simulations.

8.1 Validation of the New Model

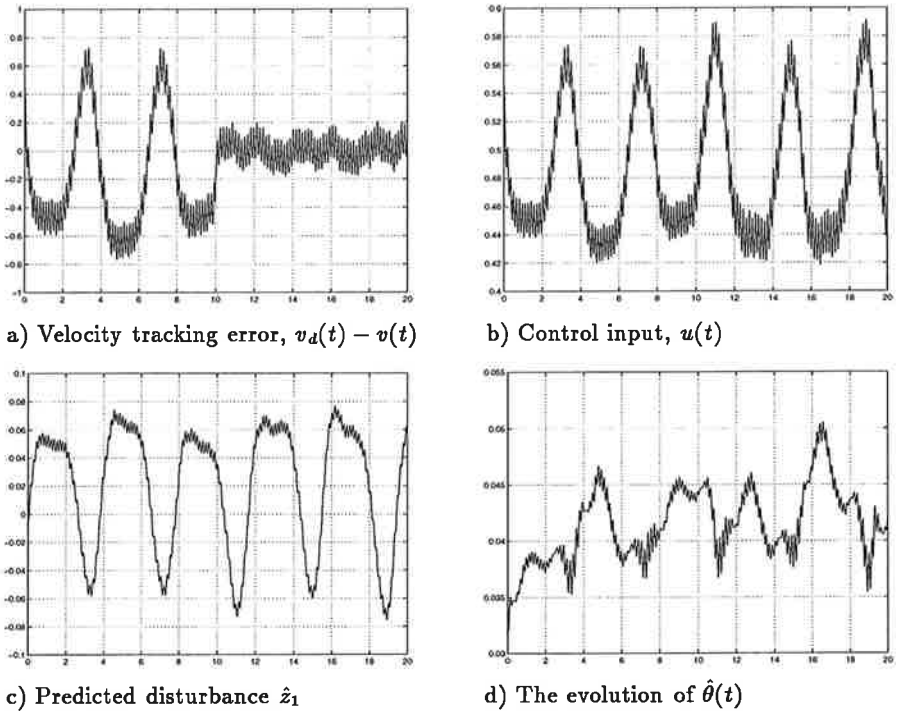
The first experiments were made to validate the principle of the new model. We ran the experimental setup for 20 seconds and started the compensation by the new observer from 10 seconds and on.

In order to get visible and good plots, we did not start the tracing of the curves from $t = 0$, but from $t = 0.5$, so the error should not start from v_d. However this means that we miss the beginning of the raise of $\hat{\theta}$.

Constant velocity

In figure 8.1 we can see the results for a constant velocity of 10 rad/sec seen from the rotor, (i.e. 10/15.5=0.6 rad/sec seen from the charge).

We can clearly see the perturbations effect on the output velocity when the system is running with only a P-controller. When the compensation is started, after 10 seconds, we see in figure 8.1a that the velocity tracking error, rapidly



a) Velocity tracking error, $v_d(t) - v(t)$

b) Control input, $u(t)$

c) Predicted disturbance $\hat{z}_1$

d) The evolution of $\hat{\theta}(t)$

Figure 8.1 Experimental result for $v_d = 10rad/s$.

converts towards zero. The small error we can still see is due to white noise from the system.

The control input in figure 8.1b is following the perturbation, from the plots for higher velocities below, one can better see the effect on the command when the observer in turned on, which increases the input.

In figure 8.1c the prediction of the perturbation is shown, which behaves periodically in order to counteract the perturbation. The reason why the perturbation is not more sinusoidal is due to the rubber tire on the eccentric wheel, which slips slightly. When the bigger radius is down, the high pressure on the tire squeezes the tire to an oval shape.

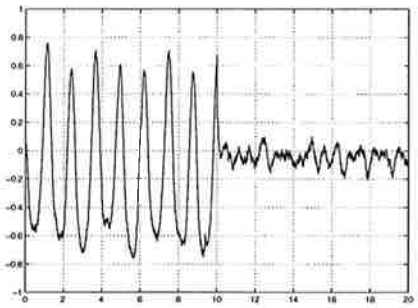
In figure 8.1d we see that the evolution of $\hat{\theta}$, which even if not as linearly as in the simulations reaches it predicted value around 0.04.

The same conclusions can be drawn for a higher velocity, 30 rad/sec (equivalent of 1.94 rad/sec for the charge), in figure 8.2. We still do not compensate until after 10 seconds.

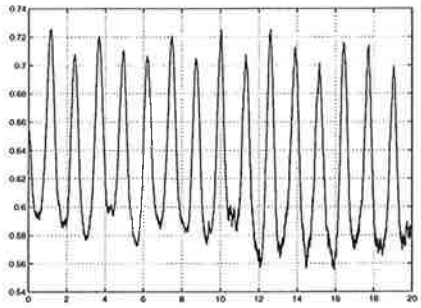
Also for relatively high velocities, we get good results with the new observer. In industry the general velocity of robot motions would be somewhere around 3-4 rad/sec. We next show the results from experiments at 50 rad/sec (corresponding to 3.2 rad/sec on the side of the charge), in figure 8.3.

The P-controller alone, can not handle these high velocities, the $\hat{\theta}$ rises rapidly from the beginning, but once its influence is included in the command when we start compensating, we can see that it soon decreases to its normal value.

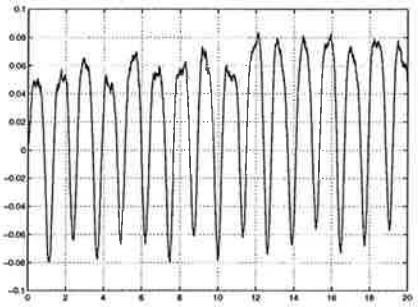
We never tried to run the setup for higher velocities than $50rad/sec$, due to its limits, but the trend shows that the observer would deal well also with higher velocities.



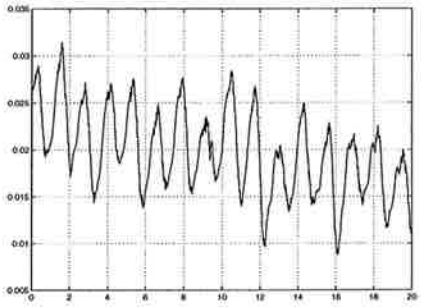
a) Velocity tracking error, $v_d(t) - v(t)$



b) Control input, $u(t)$

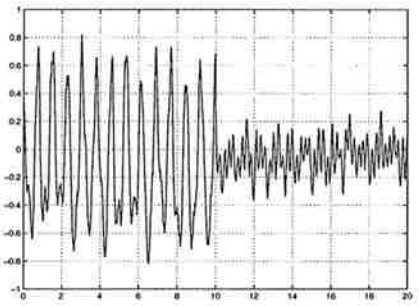


c) Predicted disturbance $\hat{z}_1$

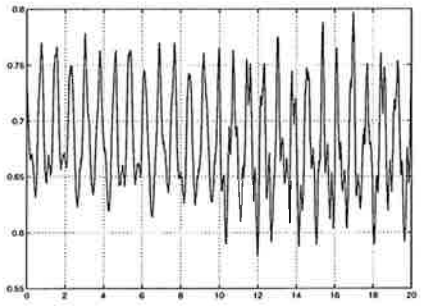


d) The evolution of $\hat{\theta}(t)$

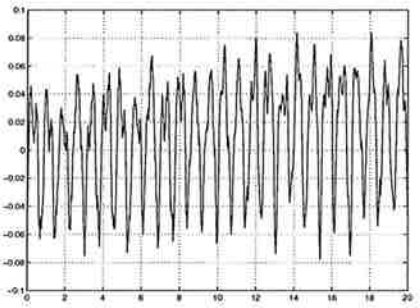
Figure 8.2 Experimental result for $v_d = 30 rad/s$.



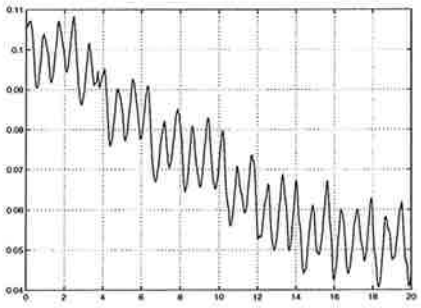
a) Velocity tracking error, $v_d(t) - v(t)$



b) Control input, $u(t)$



c) Predicted disturbance $\hat{z}_1$



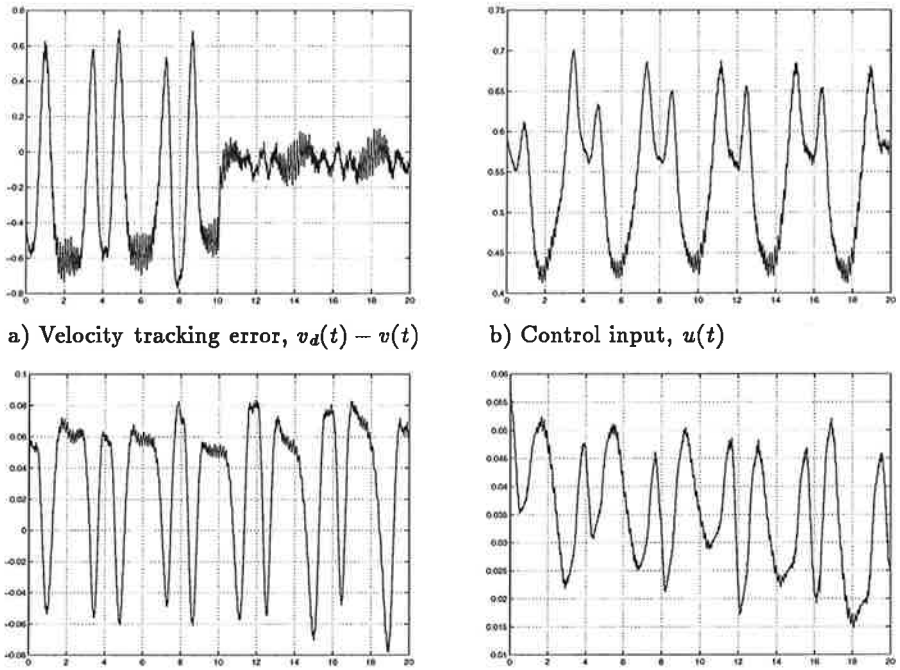
d) The evolution of $\hat{\theta}(t)$

Figure 8.3 Experimental result for $v_d = 50 rad/s$.

Sinusoidal velocity

The same experiments, where we start compensating after 10 seconds, were executed for sinusoidal velocities. The desired velocity is given by a sinus curve with an amplitude of 10 rad/sec added to a constant value of 20 rad/sec with a frequency of $\pi/2$. The same experiment is done where the constant value is

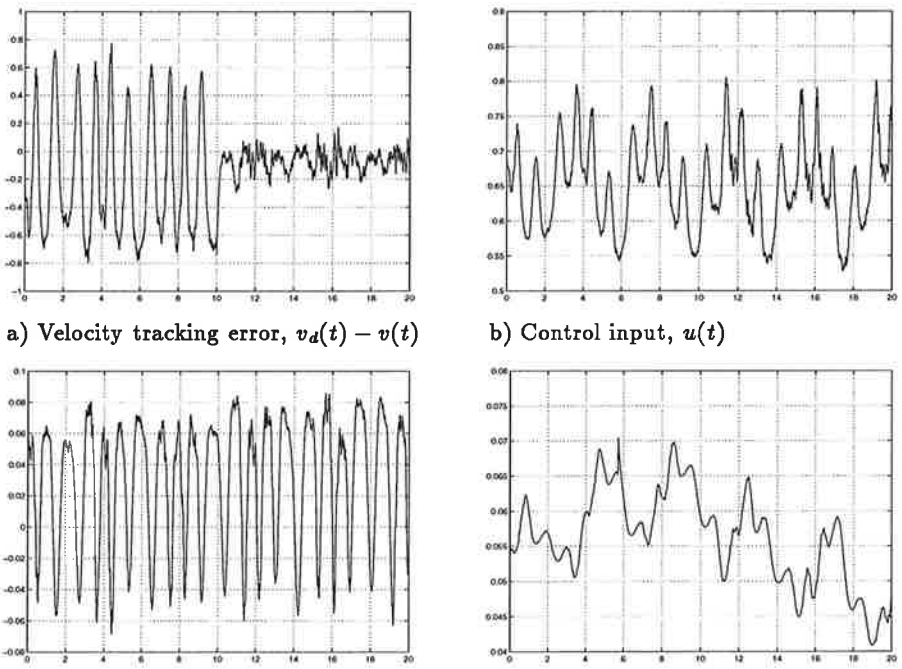
of 40 rad/sec. The results are depicted in figure 8.4 and figure 8.5 respectively.



a) Velocity tracking error, $v_d(t) - v(t)$ b) Control input, $u(t)$

c) Predicted disturbance $\hat{z}_1$ d) The evolution of $\hat{\theta}(t)$

Figure 8.4 Experimental result for $v_d = 20 + 10\sin(\pi/2\, t) rad/s$



a) Velocity tracking error, $v_d(t) - v(t)$ b) Control input, $u(t)$

c) Predicted disturbance $\hat{z}_1$ d) The evolution of $\hat{\theta}(t)$

Figure 8.5 Experimental result for $v_d = 40 + 10\sin(\pi/2\, t) rad/s$

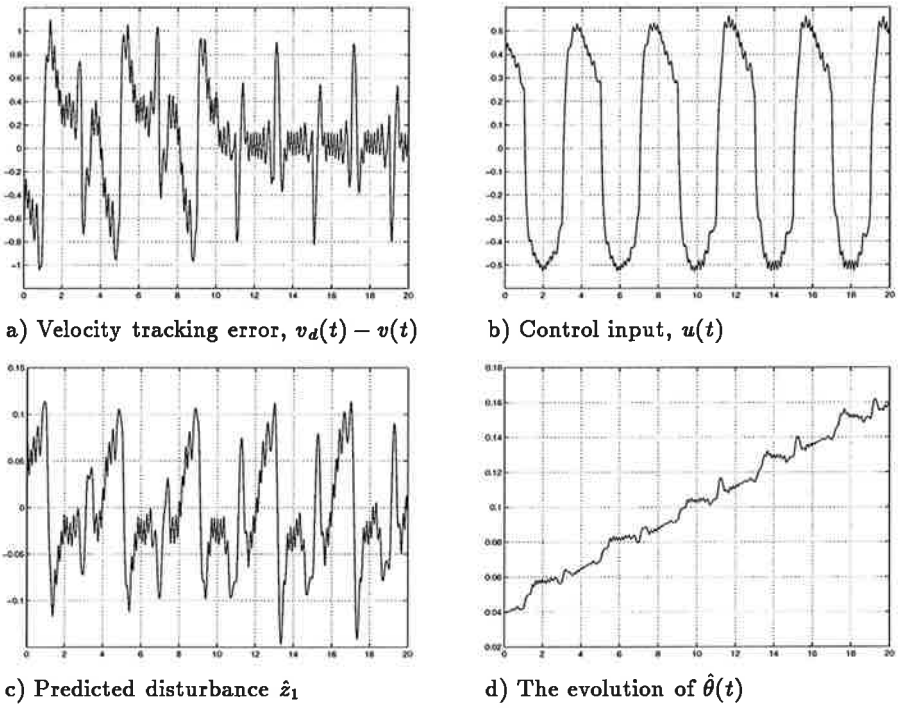
We can see from the curves that the increased perturbation frequency at the higher velocities is well followed by the observer, and the error during the last 10 seconds, with compensation, is basically still close to zero, see subfigure a.

The observer finds the frequencies of z_1 which is seen in the control input subfigure b, as well as in the perturbation prediction in subfigure c. The evolution of $\hat{\theta}$ in subfigure d, is however getting jumpier, but still remains around the predicted value.

The passage over zero velocity

We have now seen that for constant and varying velocities, but still with constant sign, the new observer eliminates the perturbation influence. The next step is to see how it behaves for the difficulties encountered at sign changes of the reference signal.

In figure 8.6 we can see the system response to a signal of amplitude 10 with a frequency of $\pi/2$.



a) Velocity tracking error, $v_d(t) - v(t)$ b) Control input, $u(t)$

c) Predicted disturbance $\hat{z}_1$ d) The evolution of $\hat{\theta}(t)$

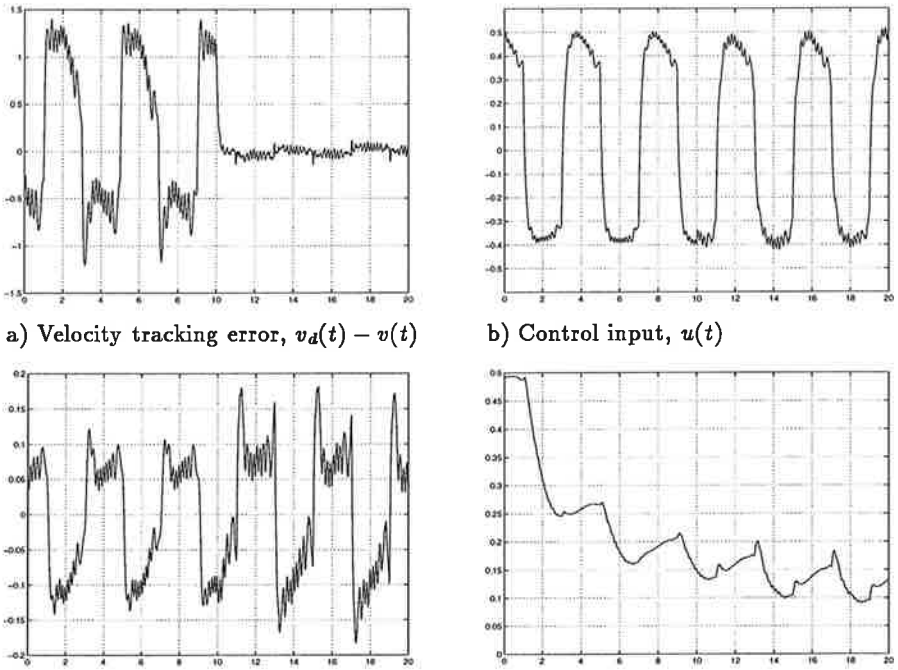
Figure 8.6 Experimental result for $v_d = 10\sin(\pi/2\,t) rad/s$

In figure 8.6a we clearly see the problem of the non-linearities around $v = 0$, and even though the result is still highly influenced by the passages over zero, we notice that the result is distinctively better when we start compensating. Part of the error is due to the error in the model described in section 2.3. We also distinguish distinctive jumps at each passage over zero, due to the non-linearities in friction discussed in section 5.2 and 5.3. The curves return however rapidly to their proper values and the results still stay stable.

Results with parallel observer

If we perform the same examples as in the section above, but compensate with the parallel AEC observer, we can improve the passage over zero velocity, see figure 8.7.

We clearly see that the peaks in the velocity tracking error in subfigure a, are a lot smaller than for the same signal for the single AEC in figure 8.6. The jumps at the crossing over zero velocity in the other subfigures are much cleaner.



a) Velocity tracking error, $v_d(t) - v(t)$ b) Control input, $u(t)$

c) Predicted disturbance $\hat{z}_1$ d) The evolution of $\hat{\theta}(t)$

Figure 8.7 Experimental result with parallel observer for $v_d = 10\sin(\pi/2\,t)rad/s$

8.2 Comparison with a PI-Controller

We have now verified that the new observer works well also in practice, and the next step is to see at what point our compensator is useful. To do this we compared it with an PI regulator chosen to have the same energy usage in the command as the AEC + P- controller. To chose a PI that is equivalent to our compensator is important. The PI must have gains that are in the same range as our P-controller. If we have a greater gain for the PI we will certainly see better results but this system will be more sensitive and less stable. The Bode diagram of the transfer function between the perturbation and the output for the controllers we have chosen are represented in figure 8.8. For low velocities we find that the PI works with about the same results as the AEC, but as the velocity gets higher the AEC is relatively better compared to the PI. This is not surprising. Our compensator should compensate the perturbation entirely no matter what speed. The PI on the other hand has a good perturbation correlation for low frequencies but as the frequency grows the correlation gets worse. This can theoretically be proved just by looking at the transfer function and the Bode-diagram for the perturbation.

$$G_P(s) = \frac{1}{Js + JK_v}$$

$$G_{PI}(s) = \frac{s}{Js^2 + JK_p s + JK_i}$$

where

$$J = 0.0022, \quad K_v = 40, \quad K_p = 50\,and \quad K_i = 400.$$

It can be seen in figure 8.9 that the perturbation after compensation is almost only white noise. The same for the PI regulation for low velocities but as they

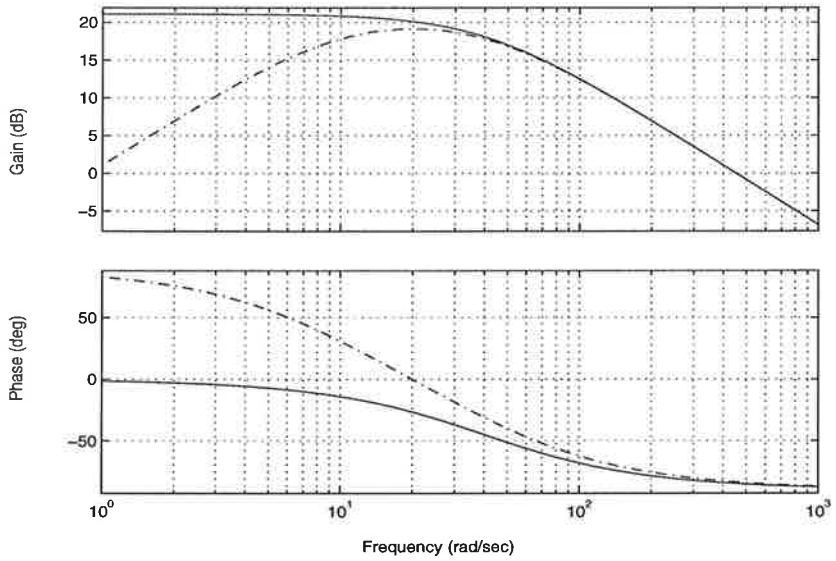


Figure 8.8 The Bode diagram for the P-controller (-) and the PI-controller (-·). The domain in which we work varies between $10/15.5 \approx 0.5 rad/sec$ and $50/15.5 \approx 3.5 rad/sec$.

get higher we find a frequency corresponding to our perturbation appearing in the error.

If we study the characteristics of the response to the input we observe

- that the PI with the same gain filters more high frequency signals.
- with the PI we have a second order that is more oscillating but that has the advantage of having an error that goes to zero.

If we compare the response to the perturbation we see that the P-controller works as a low-pass filter and the PI-controller works as a bandpass filter. So, the PI-controller works well for low and high perturbation frequencies. But there is a band of frequencies that can not be rejected by a single PI-controller. To cover desired frequencies, we can move the band to high frequencies by increasing the gains, but then as said before the system becomes more sensitive and less stable.

Statistics

From the comparison between the new AEC-controller and the regular PI-controller we have extracted certain statistics to better visualize the differences in control of the velocity tracking error $v_d - v$, as well as the cost in form of the control input, u.

For the statistics we have run the setup for 20 seconds with either AEC-compensation during the whole time or with a regular PI-controller. The values shown are - the maximum value, the square value, the mean value of the norm as well as of the absolute values, the variance and standard deviation of the velocity tracking error and the control input throughout the 20 seconds.

Table 8.1 shows these statistics for different constant velocities.

For $v_d = 10\ rad/sec$ we see that the PI-controller performs well even if the AEC-controller has a smaller error at a slightly lower cost. As we can see

	$v_d = 10\ rad/sec$		$v_d = 30\ rad/sec$		$v_d = 50\ rad/sec$	
	AEC	PI	AEC	PI	AEC	PI
$max\|e(t)\|$	0.433	0.283	0.145	0.403	0.408	0.842
$\sum e^2(t)$	17.830	27.150	5.048	95.519	71.036	167.042
$mean\|e(t)\|$	0.079	0.096	0.040	0.184	0.163	0.246
σ	0.094	0.116	0.042	0.219	0.100	0.289
$max\|u(t)\|$	0.598	0.619	0.716	0.761	0.787	0.836
$\sum u^2(t)$	469.343	515.921	765.964	850.220	891.413	1043.450
$mean\|u(t)\|$	0.482	0.505	0.617	0.650	0.666	0.720
σ	0.048	0.049	0.049	0.051	0.046	0.051

Table 8.1 Statistical comparison between the AEC-controller and a PI-controller for different constant velocities.

on the Bode-plot (figure 8.8), the frequency corresponding to 10 rad/sec = $10/15.5\ rad/sec \approx 0.5$ is in the domain where we get a good PI-control.

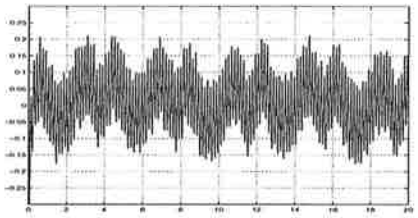
As for $v_d = 30\ rad/sec$ the velocity tracking error is smaller for the AEC-controller than for 10 rad/sec which is due to the relatively smaller influence by noise. The PI-controller gives now a distinctively worse result which is expected since the perturbation frequency is higher and the PI is designed for lower frequencies. The cost is still slightly higher for the PI-controller.

For $v_d = 50\ rad/sec$ the correction of the AEC-controller is getting a bit worse, and the control input increases as well. This is partly due to a bias introduced in the velocity error as can be seen in figure 8.11. The PI-controller however gives even worse results, concluding that also for high velocities the AEC-controller gives a much better result.

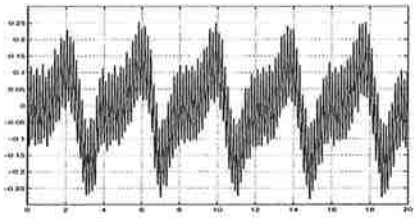
For sinusoidal velocities the same kind of comparison is also made, see table 8.2, and figures 8.12 and 8.13.

	$v_d = 20 \pm 10\ rad/sec$		$v_d = 40 \pm 10\ rad/sec$	
	AEC	PI	AEC	PI
$max\|e(t)\|$	0.193	0.425	0.503	0.5990
$\sum e^2(t)$	8.785	69.415	34.193	138.7120
$mean\|e(t)\|$	0.052	0.157	0.109	0.2250
σ	0.061	0.186	0.084	0.2620
$max\|u(t)\|$	0.714	0.754	0.814	0.868
$\sum u^2(t)$	618.616	665.270	843.921	992.354
$mean\|u(t)\|$	0.550	0.570	0.646	0.701
σ	0.082	0.088	0.064	0.069

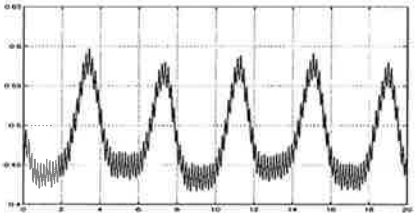
Table 8.2 Statistical comparison between the AEC-controller and a PI-controller for different sinusoidal velocities.



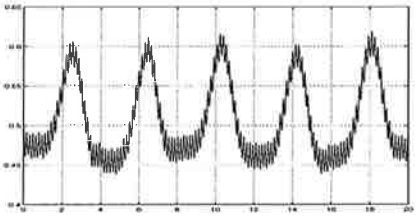
a) Velocity tracking error - AEC



b) Velocity tracking error - PI

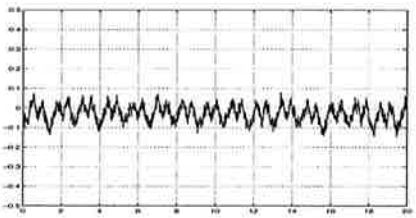


c) Control input - AEC

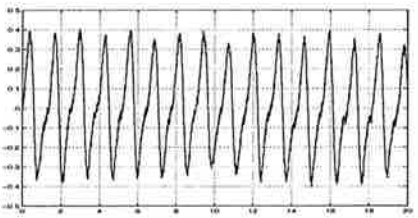


d) Control input - PI

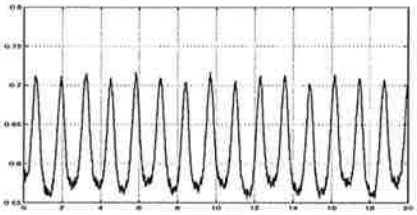
Figure 8.9 Comparison AEC- & PI-controller for $v_d = 10$ *rad/sec*



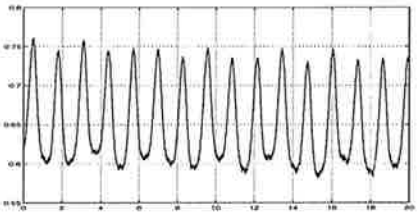
a) Velocity tracking error - AEC



b) Velocity tracking error - PI

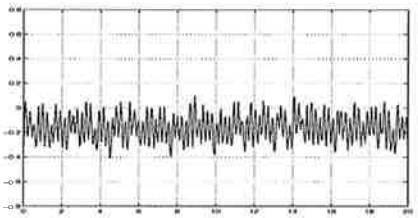


c) Control input - AEC

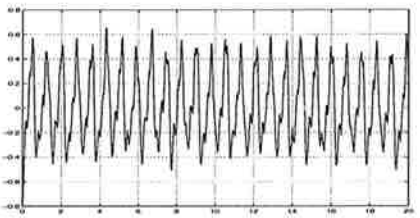


d) Control input - PI

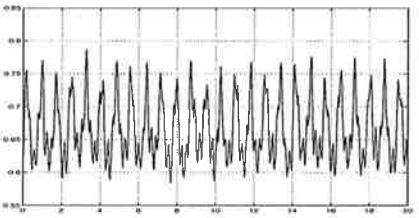
Figure 8.10 Comparison AEC- & PI-controller for $v_d = 30$ *rad/sec*



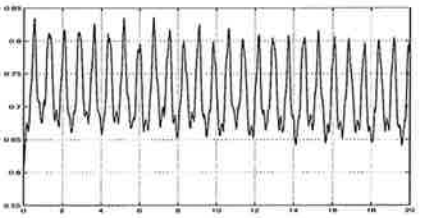
a) Velocity tracking error - AEC



b) Velocity tracking error - PI

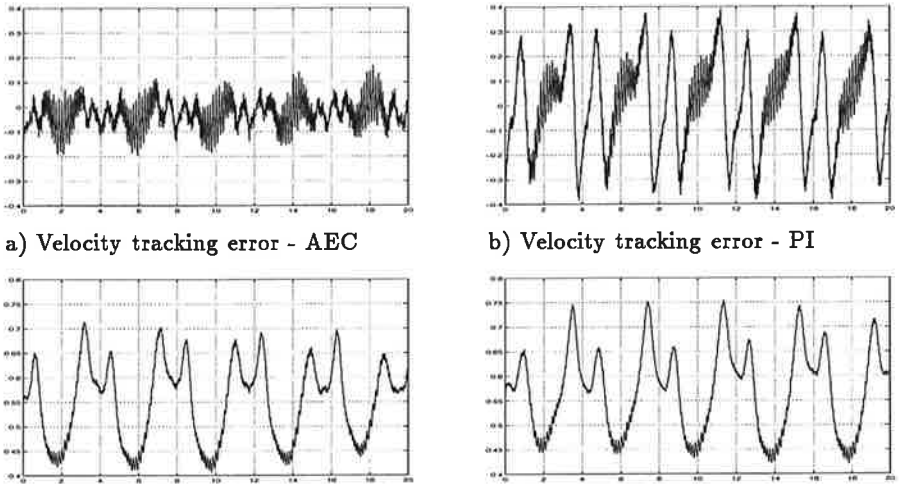


c) Control input - AEC



d) Control input - PI

Figure 8.11 Comparison AEC- & PI-controller for $v_d = 50$ *rad/sec*



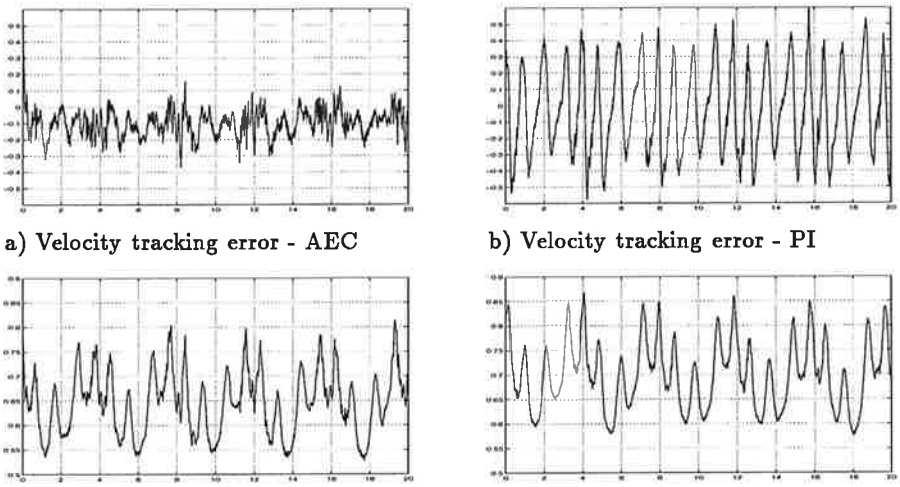
a) Velocity tracking error - AEC

b) Velocity tracking error - PI

c) Control input - AEC

d) Control input - PI

Figure 8.12 Comparison AEC- & PI-controller for $v_d = 20 + 10\sin(\pi/2\,t)\,rad/s$



a) Velocity tracking error - AEC

b) Velocity tracking error - PI

c) Control input - AEC

d) Control input - PI

Figure 8.13 Comparison AEC- & PI-controller for $v_d = 40 + 10\sin(\pi/2\,t)\,rad/s$

9. Conclusions

We have tested a control scheme called *Adaptive Eccentricity Compensation*, AEC, developed by Carlos Canudas de Wit and Laurent Praly. The AEC attempts to reject oscillating disturbances. There are many areas where this compensation is useful, e.g. drive systems, rolling mills and robotics. The basic idea is that it is position-dependent which differs it from previous time-dependent schemes.

An adaptive observer, which returns an estimate of the periodic disturbance has also been developed.

We have validated the system in simulation for a general sinusoidal perturbation. The results for velocity tracking have been very good when the velocity is positive. But an error in the model for negative velocities has been discovered. The observer does, however, correct this error quite fast.

Once this validation was performed we adjusted the perturbation model to deal with friction. This was based on the LuGre friction model. The strong non-linearity of the friction force at zero velocity causes problems but the method gives satisfactory results even in this case.

A new model with a parallel observer has been introduced and tested for the passage over zero velocity. The velocity tracking error has been reduced, even if the error in the initial model is still present.

The model has also been tested in practical experiments on a system consisting of a big cylinder whose velocity is perturbed by the friction from an eccentric wheel placed on the cylinder.

The velocity tracking error is very small for positive velocities, but with peaks in the output velocity for each sign change in the reference signal.

A comparative study has been made between the AEC-controller and a regular PI-controller. The results show that the PI-controller performs well only for certain velocities but the AEC-controller works well for all velocities in a range of $1 - 4\, rad/sec$. The errors are smaller for the AEC with the same control input.

Further investigation could be proposed in the evolution of the model to properly cover also negative velocities. The robustness of the model could also be studied. Comparisons with other models for the same rejection of periodic disturbances could be evaluated.

A. Simulink Models

General Simulation Interface

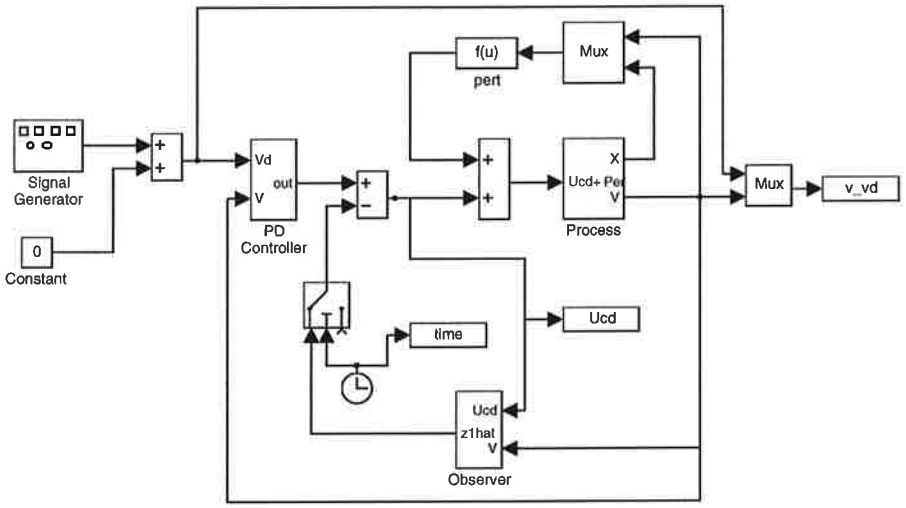


Figure A.1 The general scheme for the simulations in SIMULINK. f(u) is the function which calculates the simulated value of $d(x)$ in its different forms described in chapter 5.

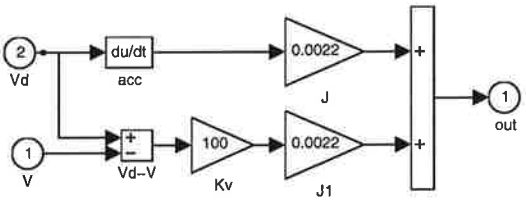


Figure A.2 Subblock: PD-controller.

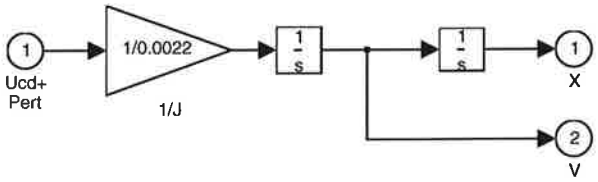


Figure A.3 Subblock: Process.

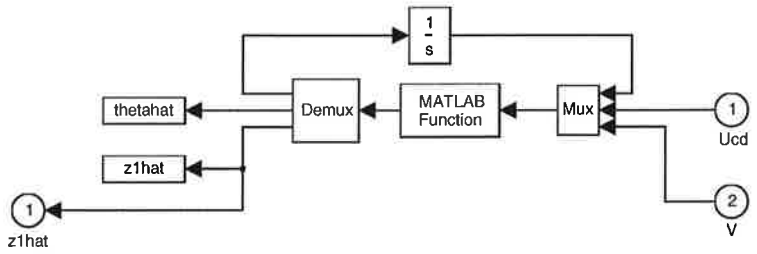


Figure A.4 Subblock: Observer. The subblock calculates and returns $\hat{z}_1$.

The MATLAB function in the Observer-block contains the following code:

```
% Function which calculates the derivates of the zeta-parameters

function [y1]=obs3(x)

% constant parameters ----------------------------------------
w = 0.5;
lambda = 2;
k1 = 2*w;
k2 = w^2;
gamma = 1;
mu = 1;
J = 0.0022;

% in parameters ----------------------------------------------
%x1 = zeta1hat;
%x2 = zeta2hat;
%x3 = zeta1bar;
%x4 = vetahat;
%x5 = u;
%x6 = v;

% calculations -----------------------------------------------
z1hat = x(1) + k1*J/2*abs(x(6))*x(6);
z1bar = x(3);
z2hat = x(2) + k2*J/2*abs(x(6))*x(6) +
        gamma*lambda*J/2*abs(x(6))*x(6)*z1bar*z1bar;
tetahat = x(4) - gamma*J/2*abs(x(6))*x(6)*z1bar;

dot_zeta1hat = abs(x(6))*(z2hat - k1*(x(5)+z1hat));
dot_zeta2hat = abs(x(6))*(-(k2+tetahat)*z1hat - k2*x(5) -
               gamma*lambda*z1bar*z1bar*(x(5)+z1hat) +
               gamma*J*abs(x(6))*x(6)*z1bar*(mu*z1bar-z1hat));
dot_zeta1bar = abs(x(6))*(-1/lambda*(mu*z1bar-z1hat));
dot_vetahat = abs(x(6))*(gamma*z1bar*(x(5)+z1hat) -
              gamma*J/2/lambda*abs(x(6))*x(6)*(mu*z1bar-z1hat));

z1hat = x(1) + k1*J/2*abs(x(6))*x(6);
tetahat = x(4) - gamma*J/2*x(3)*abs(x(6))*x(6);
```

```
%out parameters ------------------------------------------------
y1 = [0 0 0 0 0 0];
y1(1) = dot_zeta1hat;
y1(2) = dot_zeta2hat;
y1(3) = dot_zeta1bar;
y1(4) = dot_vetahat;
y1(5) = tetahat;
y1(6) = z1hat;
y1 = [y1(1) y1(2) y1(3) y1(4) y1(5) y1(6)];
```

Additions for Sign-Simulations

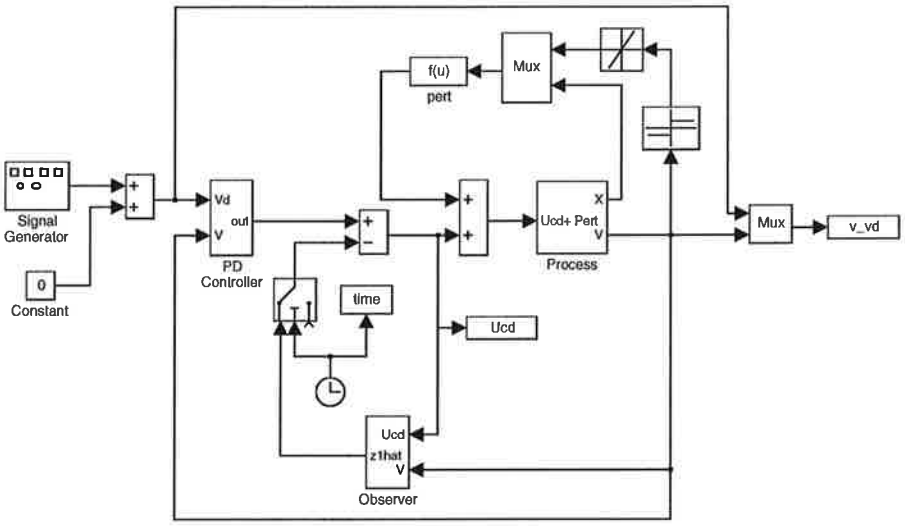


Figure A.5 Main blockscheme for the sign-function simulations.

Additions for the LuGre Simulations

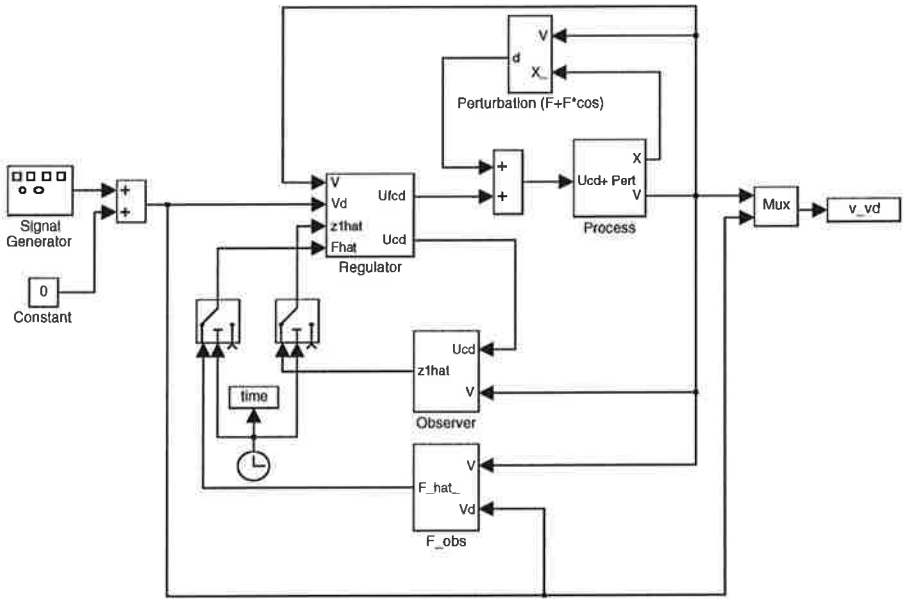


Figure A.6 Main blockscheme for the LuGre-simulations.

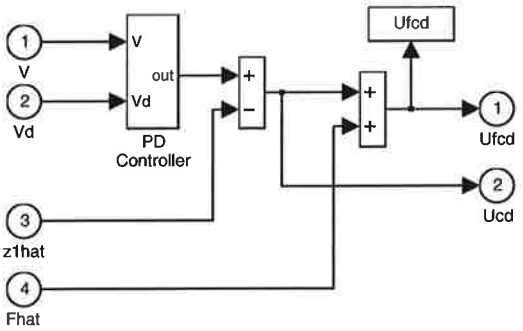


Figure A.7 Subblock: Regulator.

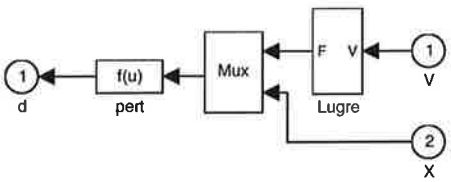


Figure A.8 Subblock: Perturbation (F + F*cos). The subblock calculates the simulated perturbation including the LuGre-friction and the formula for the periodic perturbation described in chapter 5

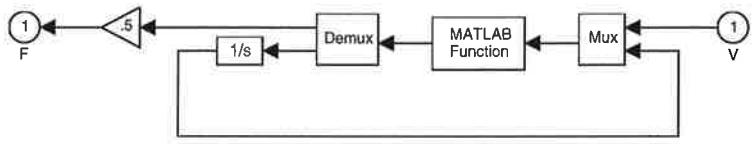


Figure A.9 Subblock: Lugre. The LuGre-friction force itself is calculated here.

The MATLAB function includes:

```
% Function which calculates the simulated value of the friction

function y=lugre(x)

% constant parameters ---------------------------------------
sigma0 = 260;
alfa0 = 0.285;
alfa1 = 0.05;
sigma2 = 0.018;
sigma1 = 0.6;
vs = 0.01;

% in parameters ---------------------------------------------
%x1 = v;
%x2 = z;

% calculations ----------------------------------------------
g = alfa0 + alfa1*(exp(-(x(1)/vs)^2));
z_dot = x(1) - (abs(x(1))/g)*x(2)*sigma0;
F = sigma0*x(2) + sigma1*z_dot + sigma2*x(1);

% out parameters --------------------------------------------
y = [0 0];
y(1) = F;
y(2) = z_dot;
y = [y(1) y(2)];
```

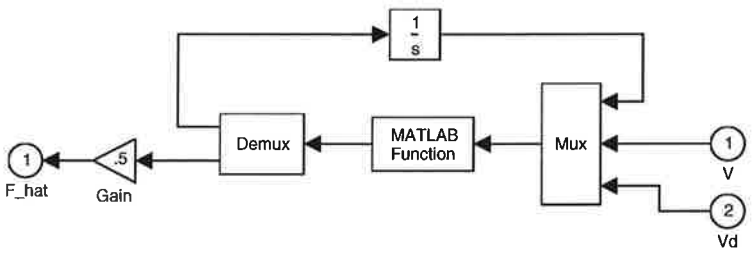


Figure A.10 Subblock: F_obs. The subblock calculates and returns $\hat{F}$, the fixed part of the perturbation.

The MATLAB function includes:

```
% Function which calculates the friction compensation

function y=lgfobs(x)

% constant parameters ----------------------------------------
sigma0 = 260;
alfa0 = 0.285;
alfa1 = 0.05;
sigma2 = 0.018;
sigma1 = 0.6;
vs = 0.01;
k = 0.01;

% in parameters ----------------------------------------------
%x1=zhat;
%x2=v;
%x3=vd;

% calculations -----------------------------------------------
g = alfa0 + alfa1*(exp(-(x(2)/vs)^2));
zhat_dot = x(2) - (abs(x(2))/g)*x(1)*sigma0 - k*(x(2)-x(3));
Fhat = sigma0*x(1) + sigma1*zhat_dot + sigma2*x(2);

% out parameters ---------------------------------------------
y = [0 0];
y(1) = zhat_dot;
y(2) = Fhat;
y = [y(1) y(2)];
```

Addition for Parallel Compensation

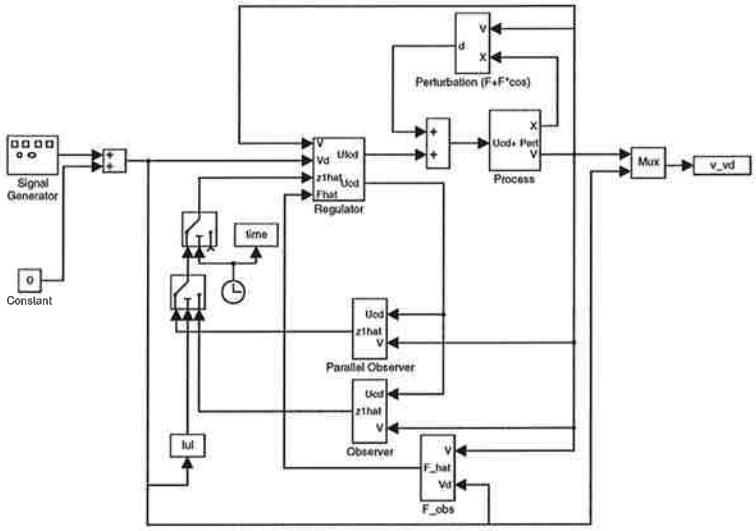


Figure A.11 General blockscheme for parallel compensation simulations.

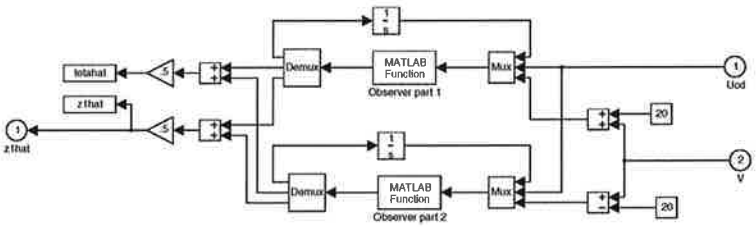


Figure A.12 Subblock: Parallel Observer.

B. Executable Program

```
#include "stdlib.h"
#include "math.h"
#include "impexc.h"   /* includes brtenv.h processor check code....
                         system_init ..->init() includes ds1002.h */


/*****************************   DECLARATIONS   *****************************/

float int_res(float);
float crenau(float,float,float);
float dead_zonep(float,float);
float sat(float,float);
int alter(int,int,float);
void lire(void);
void reference(void);
void signaux(void);
void commande(void);
void comp_fr(void);
void comp_hp_fr();
void sorties(void);
void saturation(void);
void envoie(void);
void config(void);
void service_interrupt();

void  ds2101(long base, long channel, float value);     /* D/A card */
float ds3001(long base, long channel);                  /* incr. enc. card*/
long  cda_[3] = {0x00000003,0x00000004,0x00000005},
                                  /* CDA of the  2101 channel 3,4,5 */
      cad_[3] = {0x00000011,0x00000012,0x00000013},
                           /* CAD vel. mot., pos. charge and current mot.*/
      cod = 0x00000004,            /*  new coder and the 3001 */
      ds2101_base = 0x00000080,    /*  DS2101 CDA card */
      ds3001_base = 0x00000040;    /* DS3001 codercard  */

#define  frq_e 1000.0            /* sampling frequency : Hz.  */
#define  per_e (1/frq_e)         /* sampling period : seconds  */
#define  te_2 (per_e/2)

#define  PI  3.1415926
#define  PIf2  (PI*2)
#define  np_cod  120000           /* number of points in coder  */
#define  rap_red  15.5            /* reduction q_moteur/q_charge */
#define  res_p  (PIf2/np_cod)     /* resolution in position  */
#define  res_ps2  (res_p/2)       /* resolution in position / 2 */
#define  res_v  (res_p*frq_e)     /* resolution in velocity */

#define  deg_a_rad  (PIf2/360.0)
#define  kq_cm  439.2264877774094  /* PIf2*pow(2,23)/np_cod resolution:
                                      d=2pi/120000 -> d* 2^23  =kq_cm*/
#define  VIT_MIN res_v
#define  MIN_DBL 1e-10

#define  DOWNSAMPL 20    /* parameters for the ponderating error*/
#define  FENETRE_POND 4  /* ponderation time in seconds */
#define  taille_vec_err (int)((FENETRE_POND*frq_e/DOWNSAMPL) - 1)

#define kca (3.0*0.353*10.0)  /* gain u */

float   c_tr = 0.0;

float   amp_dither = 0.0,
        fq_dither = 0.0,
        dither = 0.0;

float   temps = 0.0;        /* time variable */

float   sd_[3] = {0.0,0.0,0.0},  /* digital exits */
        ea_[3] = {0.0,0.0,0.0};  /* analog entries */
```

```
float   q_c = 0.0,      /* position counter on the motor axis */
        q_c_1 = 0.0;

int     compt_i = 0,    /* counter of complete tours on the motor axis */
        redo = 1,
        redo2 = 1;

float   qf = 0.0,       /* position counter on the motor axis (filtered) */
        qf_1 = 0.0,
        qtest = 0.0,    /* ea_[2] */

        A1_fq = 0.41421356,  /* butter(1,0.25) Fc = 0.25*0.5 Fs */
        B0_fq = 0.29289321,
        B1_fq = 0.29289321,  /* filter of estimated position */

        qp_est = 0.0,        /* estimed speed with q_c */
        qp_est_1 = 0.0,
        qpef = 0.0,          /* estimed speed (qp_est filtered) */
        qpef_1 = 0.0,
        acc = 0.0,
        acc_1 = 0.0,
        accf = 0.0,
        accf_1 = 0.0,
        A1_fv = 0.3249,      /* butter(1,0.25) Fc = 0.25*0.5 Fs */
        B0_fv = 0.3375,
        B1_fv = 0.3375;      /*  filter of estimated speed */

/* Controller */
float   J = 0.0025,    /* 0.0022 Kg*m^2  inertia for system motor axis */
        Kp = 0.0,      /* proportional gain */
        Ki = 0.0,      /* integral gain */
        Kd = 0.0,      /* derivative gain */
        Wo = 40.0,     /* eigenfrequency in closed loop PID */
        W1 = 10.0;

/* Configuration */
int     config_ref = 8,
        config_com = 3,
        comp_frott = 12,
        app_comp_frott = 1,
        comp_on = 0,
        comp_mode = 1,
        k_qp=1, i_qp=0;

/* Control Input */
float   u = 0.0,       /* command signal ... */
        ur = 0.0,      /* command signal ... */
        u_ord = 0.0;   /* command signal in computer units */

/* Errors */
float   erreur_q = 0.0,              /* position error */
        erreur_q_1 = 0.0,
        erreur_qp = 0.0,             /* velocity error */
        m_err_qp = 0.0,
        max_err_qp = 0.0,
        erreur_qpp = 0.0,            /* acceleration error*/
        err_pond_qp = 0.0,           /* ponderated velocity error (4 sec) */
        sum_err_qp = 0.0,
        vec_err_qp[taille_vec_err],  /* vector for calc of ponderated error */
        fsaturation = 0.0,           /* saturation flag */

float   amp = 0.21;


/* Adaptive Controller */
float   zp = 0.0,                /*  d/dt of internal state z   */
        z = 0.0,                 /*  internal state */
        gv = 0.0,                /*  function of static friction */
        v = 0.0,                 /*  velocity  */
        f_z = 0.0,
        f_zp = 0.0,
        f_v = 0.0,
        f_est = 0.0;             /* estimated friction */

float   ui = 0,
        up = 0,
        ud = 0,
```

```
        uacc = 0;

/* Friction model parametres */
float   ALF0 = 0.3728,
        ALF1 = 0.039,
        ALF2 = 0.0106,
        V0 = 0.01,
        SGM0 = 260.0,
        SGM1 = 0.0,
        zet_dyn = 1.0,
        k_obs = 1.5,     /*  tracking error obs. gain  */
        flag_vitf = 1.0; /* if flag_vitf>0.5 frict-precomp, else frict-comp */

long int  cont_ech = 0;       /*  counter of sampl. for velocity estimation */

/* AEC variables */
float   z1_m = 0.0,
        z1_hat = 0.0,
        z1_hat_1 = 0.0,
        z2_hat = 0.0,
        z1_bar = 0.0,
        teta_hat = 0.0,
        nabla_teta = 0.0,

        cc_z1_hat = 0.0,
        cc_z1_hat_1 = 0.0,
        cc_z2_hat = 0.0,
        cc_z1_bar = 0.0,
        cc_teta_hat = 0.0,
        e_z1 = 0.0,

        zg1_hat = 0.0,
        zg2_hat = 0.0,
        zg1_bar = 0.0,
        tetag_hat = 0.0,

        z1_view = 0.0,
        teta_view = 0.0,

        vtemp = 0.0,
        z1_hatp = 0.0,
        z1_hatp_1 = 0.0,
        teta_hatp = 0.0,
        zg1_hatp = 0.0,
        zg2_hatp = 0.0,
        zg1_barp = 0.0,
        tetag_hatp = 0.0,
        z1_hatn = 0.0,
        z1_hatn_1 = 0.0,
        teta_hatn = 0.0,
        zg1_hatn = 0.0,
        zg2_hatn = 0.0,
        zg1_barn = 0.0,
        tetag_hatn = 0.0,

        zg1_hat_p = 0.0,
        zg2_hat_p = 0.0,
        zg1_bar_p = 0.0,
        tetag_hat_p = 0.0,
        cc_z1_hat_p = 0.0,
        cc_z2_hat_p = 0.0,
        cc_z1_bar_p = 0.0,
        cc_teta_hat_p = 0.0,
        cc_z1_hat_p_1 = 0.0,
        cc_z2_hat_p_1 = 0.0,
        cc_z1_bar_p_1 = 0.0,
        cc_teta_hat_p_1 = 0.0,

        sigma_hp = 0.0,
        fabs_v = 0.0,
        omega = 0.5,
        lambda = 2.0,
        k1 = 2.0,                   /* 2*omega, */
        k2 = 1.0,                   /* omega*omega */
        gama = 2.0,
        old_gama = 2.0,
        my = 1.0,
```

```
        uhp_comp = 0.0,
        lim_on = 10.0,
        lim_calc = 30.0;
int     compt_j = 0,
        hplugre = 0;


/* Creneau de position */

float   refq = 0.0,
        refq_1 = 0.0,
        refqp = 0.0,
        refqp_1 = 0.0,
        refqpp = 0.0,
        consigne_ctte = 10.0,
        amp_hp = 10.0,
        freq_hp = 1.570796,
        tmp = 0.0,
        tmpn = 0.0,
        T_per = 2.0,
        refq_add = 0.0,
        refqp_add = 0.0,
        count_tmp = 0.0;


/******* SATURATION FUNCTION OF THE VARIABLE in WITH AMPLITUDE amp *******/

float sat(float in, float amp)
{
  float out;

  fsaturation = 0.0;
  if (in > amp)
  {
    out = amp;
    fsaturation = 1.0;
  }
  else if (in > -amp)
    out = in;
  else
  {
    out = -amp;
    fsaturation = -1.0;
  }

  return(out);
}


/******* FUNCTION FOR READING THE ANALOGUE POSITION OF THE COUNTER *******/

void lire(void)
{
  ea_[2] = ds3001(ds3001_base,cod);    /* position  motor counter */
  qtest = ea_[2];

  q_c_1 = q_c;

  if (qtest > 1 - 0.000004*refqp)   /* when the motor counter is saturated */
  {                                 /* we extrapolate the position, until */
    if (redo == 1)                  /* the counter has reached its new */
     {                              /* startvalue */
       redo = 0;
       redo2 = 0;
       compt_i = compt_i + 2;
     }
     q_c = q_c_1 + per_e*refqp;
  }
  else if (qtest < -1 + 0.000004*refqp)
  {
     redo2 = 1;
     q_c = q_c_1 + per_e*refqp;
  }
  else if (redo2 == 1)
  {
     q_c = kq_cm * (compt_i + ea_[2]);
     redo = 1;
```

```
  }

  qf_1 = qf;   /* filtered motor position y(k)=qf_1  x(k)=q_c  x(k-1)=q_c_1 */
  qf =  A1_fq*qf_1 + B0_fq*q_c + B1_fq*q_c_1;

  /* calcul de la vitesse */
  qp_est_1 = qp_est;
  qp_est = (qf - qf_1)*frq_e; /*  vitesse moteur estimee  */

  qpef_1 = qpef;   /* vitesse moteur estimee filtree */
  qpef =  A1_fv*qpef_1 + B0_fv*qp_est + B1_fv*qp_est_1;

  acc_1 = acc;
  acc = (qpef - qpef_1)*frq_e; /*  acceleration moteur estimee  */

  accf_1 = accf;
  accf = A1_fv*accf_1 + B0_fv*acc + B1_fv*acc_1;

/*  ea_[0] = ds2002(ds2002_base,cad_[0]);
   qp = kqpm * ea_[0];    vitesse moteur avec le tacho */
/*  ea_[1] = ds2002(ds2002_base,cad_[1]);  position potentiometre
   q_p = kq_p * ea_[1];   axe de la charge  */
}


/******************  CALCUATIONS OF REFERENCE SIGNAL  ******************/

void reference(void)
{
  if (config_ref == 8)                              /* constant velocity */
  {
    refq = consigne_ctte*temps;
    refqp = consigne_ctte;
    refqpp = 0;
  }

  else if (config_ref == 9)                         /* sinusoidal velocity */
  {
    refq = consigne_ctte*temps + amp_hp/freq_hp*(1-cos(freq_hp*temps));
    refqp = consigne_ctte + amp_hp*sin(freq_hp*temps);
    refqpp = amp_hp*freq_hp*cos(freq_hp*temps);
  }

  else if (config_ref == 11)                        /*  zig-zag-signal */
  {
    T_per = 1/freq_hp;
    tmp = fmod(temps,T_per);
    tmpn = temps/T_per - tmp;

    if (tmp < T_per*0.25)
    {
       refqp_add = tmp/T_per/0.25;
       refq_add = refqp_add*tmp/2;
    }
    else if (tmp < T_per*0.75)
    {
       refqp_add = 1 - (tmp-T_per*0.25)/T_per/0.25;
       refq_add = tmp - (tmp*tmp/2-T_per*0.25*tmp)/T_per/0.25 - T_per*0.25;
    }
    else
    {
       refqp_add = -1 + (tmp - T_per*0.75)/T_per/0.25;
       refq_add = -tmp + (tmp*tmp/2 - T_per*0.75*tmp)/T_per/0.25 +
                  T_per*0.75 + 10*T_per/8 ;
    }

    refq = consigne_ctte*temps + amp_hp*refq_add;
    refqp =  consigne_ctte + amp_hp*refqp_add;

  }

  else if (config_ref == 4)                         /* double frequency */
  {
    refq = consigne_ctte*temps + amp_hp/(1-10*10)*(1/freq_hp*
           (1-cos(freq_hp*temps)*cos(10*freq_hp*temps) -
           10*sin(freq_hp*temps)*sin(10*freq_hp*temps)));
    refqp = consigne_ctte + amp_hp*sin(freq_hp*temps)*cos(10*freq_hp*temps);
```

```
    refqpp = amp_hp*(freq_hp*cos(freq_hp*temps)*cos(10*freq_hp*temps) -
             10*freq_hp*sin(freq_hp*temps)*sin(10*freq_hp*temps));
  }
}

/********************  CALCULATION OF COMMAND SIGNALS  ********************/

void signaux()
{

  erreur_q_1 = erreur_q; /* valeurs anterieures pour filtrage  */
  erreur_q = refq - q_c;

  v = qpef;  /*qpef,  var. auxiliere de vitesse  */

  erreur_qp = refqp - v;
  m_err_qp = erreur_qp*1;          /*  erreur_qp*1000  */

  max_err_qp = fabs(m_err_qp);
  if (k_qp==DOWNSAMPL) {
    k_qp=1;
    sum_err_qp = sum_err_qp - vec_err_qp[0] + max_err_qp;
    for (i_qp=0;i_qp<taille_vec_err-1;i_qp++)
      vec_err_qp[i_qp]=vec_err_qp[i_qp+1];

    vec_err_qp[taille_vec_err-1] = max_err_qp;
    err_pond_qp = (float)(sum_err_qp/((FENETRE_POND*frq_e)/DOWNSAMPL));
  }
  else
    k_qp++;

  erreur_qpp = refqpp - accf;
}

/************************  CALCULATION OF CONTROL LAW ************************/

void commande()
{
  if (config_com == 3)                        /* P velocity */
  {
    Kp = Wo;
    uacc = J * refqpp;
    ud = 0;
    up = J *Kp * erreur_qp;
    ui = 0;
  }

  else if (config_com == 9)                   /* PI velocity */
  {
    Ki = Wo*W1;    /* =400  poles at 10 and 40 */
    Kp = Wo + W1;  /* = 50 */
    Kd = 0;
    uacc = 0;
    ud =  0;
    up =  J*Kp*erreur_qp;
    ui =  J*Ki*erreur_q;
  }

  ur = uacc+ud+up+ui;                         /* equation for motor torque */
  u_ord = ur/kca;
  u = ur;
}

/******************  ESTIMATION OF FRICTION COMPENSATION  ******************/

void comp_fr()
{
  if (flag_vitf < 0.5)      /* friction compensation */
    v = qpef;
  else                      /* friction precompensation */
    v = refqp;

  SGM1 = 2*zet_dyn*sqrt(SGM0*J) - ALF2;
```

```
  if (comp_frott == 12)   /* dynamic estimation of friction, velocity error */
  {
    if (v < 0)
    {
       ALF0 = 0.3040 - 0.002*(fabs(refqp)-10);
       ALF2 = 0.0111 - 0.00007*(fabs(refqp)-10);
    }
    else
    {
       ALF0 = 0.3940 - 0.002*(fabs(refqp)-10);
       ALF2 = 0.0111 - 0.00007*(fabs(refqp)-10);
    }

    gv = ALF0 + ALF1*exp(-((v*v)/(V0*V0)));

    if (fabs(v) <= VIT_MIN)
    {
      zp = 0.0;
      v = 0;
    }
    else {
      z = (z + te_2*(zp + v + 0.05*k_obs*erreur_qp))/
          (1 + te_2*SGM0*fabs(v)/gv);
      zp = (v - SGM0*fabs(v)*z/gv + 0.05*k_obs*erreur_qp);
    }

    f_z = SGM0*z;
    f_zp = SGM1*zp;
    f_v = ALF2*v;
    f_est = f_z + f_zp + f_v;           /*estimated friction */
  }

  f_est = sat(f_est,kca/5);

  if (app_comp_frott == 1)  /* if compensation on */
    {
      u_ord += f_est/kca;   /* the compensation */
      u += f_est;
    }
}


/*****************************  AEC COMPENSATION  ***************************/

void comp_hp_fr()
{
    k1 = 2*omega;
    k2 = omega*omega;

    v = qp_est;

    z1_hat_1 = z1_hat;
    z1_hatp_1 = z1_hatp;
    z1_hatn_1 = z1_hatn;
    cc_z1_hat_1 = cc_z1_hat;

    if ((comp_on == 1) && ((comp_mode ==1) || (comp_mode ==3)))   /* AEC on */
    {
       uhp_comp = ur - z1_hat;
    }
    else if ((comp_on == 1) && (comp_mode ==2))     /* Comp with acc-est on */
    {
       uhp_comp = ur - cc_z1_hat;
    }
    else if (((comp_on == 2) && ((comp_mode ==1) ||
              (comp_mode ==3)))  && (temps > 10.2))    /* Comp on after 10s */
    {
       uhp_comp = ur - z1_hat;
    }
    else if (((comp_on == 2) && (comp_mode ==2))  && (temps > 10.2))
                                      /* Comp with acc-est on after 10s plot */
    {
       uhp_comp = ur - cc_z1_hat;
    }
    else                                                  /* No compensation */
    {
```

```
    uhp_comp = ur;
}

acc=accf;
z1_m = J * acc - uhp_comp;                                    /* real z1 */

if (redo == 0)                    /* counter of the motor is saturated */
{
   if (compt_j == 0)
   {
      old_gama = gama;
      gama = 0;
      compt_j = 1;
   }
}
else
{
   gama = old_gama;
   compt_j = 0;
}

fabs_v = fabs(v);


/**  Compensation using acceleration  **/

e_z1 = z1_m - cc_z1_hat;

cc_z1_hat_p = fabs_v*(cc_z2_hat + k1*e_z1);
cc_z1_bar_p = fabs_v*(-1/lambda*(my*cc_z1_bar - cc_z1_hat));
nabla_teta = -gama*cc_z1_bar*e_z1;
cc_z2_hat_p = fabs_v*(-cc_teta_hat*cc_z1_hat + k2*e_z1 -
              lambda*cc_z1_bar*nabla_teta);
cc_teta_hat_p = -fabs_v*sigma_hp*cc_teta_hat +
                fabs_v*(-gama*cc_z1_bar*e_z1);

cc_z1_hat += per_e*cc_z1_hat_p;                /* integration by Euler */
cc_z2_hat +=per_e*cc_z2_hat_p ;
cc_z1_bar += per_e*cc_z1_bar_p;
cc_teta_hat += per_e*cc_teta_hat_p;

if (fabs(cc_z1_hat) > 2.0)                        /* security measure */
{
   cc_z1_hat = cc_z1_hat_1;
}


/**  Compensation not using acceleration  **/

z1_hat = zg1_hat + k1*J/2*fabs_v*v;
z1_bar = zg1_bar;
z2_hat = zg2_hat + k2*J/2*fabs_v*v +
         gama*lambda*J/2*fabs_v*v*z1_bar*z1_bar;
teta_hat = tetag_hat - gama*J/2*fabs_v*v*z1_bar;

zg1_hat_p = fabs_v*(z2_hat - k1*(uhp_comp+z1_hat));
zg2_hat_p = fabs_v*(-(k2+teta_hat)*z1_hat - k2*uhp_comp -
               gama*lambda*z1_bar*z1_bar*(uhp_comp+z1_hat) +
               gama*J*fabs_v*v*z1_bar*(my*z1_bar-z1_hat));
zg1_bar_p = fabs_v*(-1/lambda*(my*z1_bar - z1_hat ));
tetag_hat_p = fabs_v*(gama*z1_bar*(uhp_comp + z1_hat) -
                gama*J/2/lambda*fabs_v*v*(my*z1_bar - z1_hat));

zg1_hat = zg1_hat + per_e*zg1_hat_p;           /* integration by Euler */
zg2_hat = zg2_hat + per_e*zg2_hat_p;
zg1_bar = zg1_bar + per_e*zg1_bar_p;
tetag_hat = tetag_hat + per_e*tetag_hat_p;


/**  For the parallell observer: if abs(v) > 0.6 then use above values  **/
/**  if abs(v) < 0.6, use values below.                                 **/

vtemp=v;          /* store real v value */

/* Positive part of parallell observer */
v = vtemp + lim_calc;
fabs_v = fabs(v);
```

```
z1_hatp = zg1_hatp + k1*J/2*fabs_v*v;
z1_bar = zg1_barp;
z2_hat = zg2_hatp + k2*J/2*fabs_v*v +
         gama*lambda*J/2*fabs_v*v*z1_bar*z1_bar;
teta_hatp = tetag_hatp - gama*J/2*fabs_v*v*z1_bar;

zg1_hat_p = fabs_v*(z2_hat - k1*(uhp_comp+z1_hatp));
zg2_hat_p = fabs_v*(-(k2+teta_hatp)*z1_hatp - k2*uhp_comp -
            gama*lambda*z1_bar*z1_bar*(uhp_comp+z1_hatp) +
            gama*J*fabs_v*v*z1_bar*(my*z1_bar-z1_hatp));
zg1_bar_p = fabs_v*(-1/lambda*(my*z1_bar - z1_hatp ));
tetag_hat_p = fabs_v*(gama*z1_bar*(uhp_comp + z1_hatp) -
              gama*J/2/lambda*fabs_v*v*(my*z1_bar - z1_hatp));

zg1_hatp = zg1_hatp + per_e*zg1_hat_p;
zg2_hatp = zg2_hatp + per_e*zg2_hat_p;
zg1_barp = zg1_barp + per_e*zg1_bar_p;
tetag_hatp = tetag_hatp + per_e*tetag_hat_p;

/* Negative part of parallell observer */
v = vtemp-lim_calc;
fabs_v=fabs(v);

z1_hatn = zg1_hatn + k1*J/2*fabs_v*v;
z1_bar = zg1_barn;
z2_hat = zg2_hatn + k2*J/2*fabs_v*v +
         gama*lambda*J/2*fabs_v*v*z1_bar*z1_bar;
teta_hatn = tetag_hatn - gama*J/2*fabs_v*v*z1_bar;

zg1_hat_p = fabs_v*(z2_hat - k1*(uhp_comp+z1_hatn));
zg2_hat_p = fabs_v*(-(k2+teta_hatn)*z1_hatn - k2*uhp_comp -
            gama*lambda*z1_bar*z1_bar*(uhp_comp+z1_hatn) +
            gama*J*fabs_v*v*z1_bar*(my*z1_bar-z1_hatn));
zg1_bar_p = fabs_v*(-1/lambda*(my*z1_bar - z1_hatn ));
tetag_hat_p = fabs_v*(gama*z1_bar*(uhp_comp + z1_hatn) -
              gama*J/2/lambda*fabs_v*v*(my*z1_bar - z1_hatn));

zg1_hatn = zg1_hatn + per_e*zg1_hat_p;
zg2_hatn = zg2_hatn + per_e*zg2_hat_p;
zg1_barn = zg1_barn + per_e*zg1_bar_p;
tetag_hatn = tetag_hatn + per_e*tetag_hat_p;

v = vtemp;                                          /* return real v value*/

if (fabs(z1_hat) > 2.0)                               /* security measure */
{
   z1_hat = z1_hat_1;
}
if (fabs(z1_hatp) > 2.0)
{
   z1_hatp = z1_hatp_1;
}
if (fabs(z1_hatn) > 2.0)
{
   z1_hatn = z1_hatn_1;
}

if (fabs(refqp) < lim_on)    /* if velocity close to 0 use parallell obs*/
{
   if (comp_mode == 3)                  /* if parallell observer mode on */
   {
      z1_hat = (z1_hatp + z1_hatn)/2;
      teta_hat = (teta_hatp + teta_hatn)/2;
   }
}

if ((comp_mode == 1) || (comp_mode == 3))    /* normal or parallell obs */
{
   z1_view = z1_hat;
   teta_view = teta_hat;
}
else if (comp_mode == 2)                      /* compensation using acc */
{
   z1_view = cc_z1_hat;
   teta_view = cc_teta_hat;
}
```

```
    if (comp_on == 1)
    {
       u = u - z1_view;
       u_ord = u_ord - z1_view/kca;
    }
    else if ((comp_on == 2) && (temps > 10.2))
    {
       u = u - z1_view;
       u_ord = u_ord - z1_view/kca;
    }
}

/***************** ASIGNATION OF SIGNALS TO D/A CONVERTER *****************/

void sorties()
{
  dither = amp_dither*sin(fq_dither*temps);
  sd_[0] = u_ord + dither;            /* the command has to stay in sd_[0]  */
}


/***************** PROTECTION AGAINST TOO BIG EXIT SIGNALS *****************/

void saturation()
{
  u_ord = sat(u_ord,1.0);
  u = sat(u,kca);
}


/******************** SEND DIGITAL SIGNALS TO CDA-CARD ********************/

void envoie()
{
  int i;

  for (i=0;i<1;i++) ds2101(ds2101_base,cda_[i],sd_[i]);
}


/****************************** CONFIGURATION ******************************/

void config()
{
  for (i=0;i<taille_vec_err;i++)
  {
     vec_err_qp[i]=0.0;
  }
}


/********************** FUNCTION FOR SERVICE INTERRUPT **********************/

void service_interrupt()
{
  int i;

  if (c_tr > 0.4) {
  irs_initialize();
  temps += per_e;   /* updating var. temps */
  cont_ech++;
  lire();    /* read analogue entries, the counter and signals */
  reference();            /* calculate reference */
  signaux();          /* calculate command signals erreur_q, erreur_qp */
  commande();             /* calculate control law ui, ud, up */
  comp_fr();              /* friction and compensation estimation */
  comp_hp_fr();           /* eccentricity compensation */
  saturation();           /* command security, saturation */
  sorties();              /* exits for screen */
  envoie();               /* send exits to CDA */
  service_trace();        /* call for TRACE30 */
  irs_terminate();
  }
  else {
   temps = z = zp = 0.0;
```

```
    compt_i = 0;
    redo = redo2 = 1;
    old_gama = gama;
    err_pond_qp = sum_err = sum_err_qp = erreur_q = erreur_qp = 0.0;
    q_c = q_c_1 = qf = qf_1 = qp_est = qp_est_1 = qpef = qpef_1 = 0.0;
    refq = refq_1 = refqp = refqp_1 = refqpp  = 0.0;
    ur = 0.0;
    z1_hat = z2_hat = z1_bar = teta_hat = 0.0;
    zg1_hat = zg2_hat = zg1_bar = tetag_hat = 0.0;
    zg1_hat_p = zg2_hat_p = zg1_bar_p = tetag_hat_p = 0.0;
    cc_z1_hat = cc_z2_hat = cc_z1_bar = cc_teta_hat = 0.0;
    cc_z1_hat_p = cc_z2_hat_p = cc_z1_bar_p = cc_teta_hat_p = 0.0;
    zg1_hatp = zg2_hatp = zg1_barp = tetag_hatp = 0.0;
    zg1_hatn = zg2_hatn = zg1_barn = tetag_hatn = 0.0;

    for (i=0;i<taille_vec_err;i++)
    {
       vec_err_qp[i] = 0.0;
    }
    cont_ech = 0;
    ds3001_clear_counter(ds3001_base,cod);                /* init of encoder */
    for (i=0;i<3;i++) sd_[i] = 0.0;
    envoie();
  }                                                    /* stop if cockpit off */
}


/****************************** MAIN PROGRAM ******************************/

main()
{
  int i;

  *_error_flag = NO_ERROR;

  config();                                    /* take initial configuration */

  while (c_tr < 0.4)             /* wait for authorisation from cockpit (on) */
     service_cockpit();

  system_initialize();       /* initialisation from hardware and call init() */

  ds3001_clear_counter(ds3001_base,cod);                /* init of encoder */
  service_trace();

  enable_interrupts(per_e);              /* initialize sampling clock timer */

  while (c_tr >= 0.4)
    background();                       /* background process service_cockpit */

  system_terminate();

  for (i=0;i<3;i++) sd_[i] = 0.0;         /* reset of analogue exits to zero */
  envoie();
}
```

References

ARMSTRONG-HÉLOUVRY, B., P. D. and C. C. DE WIT (1994): "A survey of models, analysis tools and compensation methods for the control of machines with friction." *Automatica*, **30**, pp. 1083–1138.

ÅSTRÖM, K. J. and B. WITTENMARK (1995): *Adaptive Control*, second edition. Addison-Wesley, Reading, Massachusetts.

BODSON, M. and S. C. DOUGLAS (1997): "Adaptive algorithms for the rejection of sinusoidal disturbances with unknown frequency." *Automatica*, **12**, pp. 2213–2221.

CANUDAS DE WIT, C. and P. LISCHINSKY (1997): "Adaptive friction compensation with partially known dynamic friction model." *International Journal of Adaptive Control and Signal Processing*, **11**, pp. 65–80.

CANUDAS DE WIT, C., H. OLSSON, K. J. ÅSTRÖM, and P. LISCHINSKY (1995): "A new model for control of systems with friction." *IEEE Transactions on Automatic Control*, **40:3**.

CANUDAS DE WIT, C. and L. PRALY "Adaptive eccentricity compensation." *Submitted for publication.*

GARIMELLA, S. S. and K. SRINIVASAN (1996): "Application of repetitive control to eccentricity compensation rolling." *Journal of Dynamic Systems, Measurement, and Control*, **118**, pp. 2213–2221.

KHALIL, H. K. (1992): *Nonlinear Systems.* MacMillan, New York.

LISCHINSKY, P. A. (1997): *Compensation de Frottement et Commande en Position d'un Robot Hydraulique Industriel.* PhD Thesis, Laboratoire d'Automatique de Grenoble.

OLSSON, H. (1996): *Control Systems with Friction.* PhD thesis ISRN LUTFD2/TFRT--1045--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

The Authors

Holger Olofsson and Petra Posselius both started their engineering careers at Lund Institute of Technology in the faculties of Electronics and Physics respectively. They both studied their last year for a Masters Degree at Ecole Nationale Supérieure d'Ingénieurs Electriciens de Grenoble. This master thesis was developped at Laboratoire d'Automatique de Grenoble under the supervision of Carlos Canudas de Wit and Karl Johan Åström.