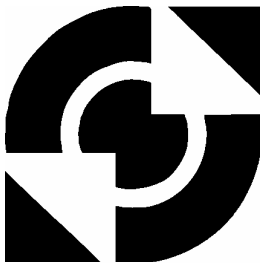
University of Twente

EEMCS / Electrical Engineering
Control Engineering



Demonstrator of advanced controllers

Hans Dirne

Masters Thesis

Supervisors
prof.dr.ir. J. van Amerongen
dr.ir. J.F. Broenink
dr.ir. T.J.A. de Vries
ir. P.B.T. Weustink

May 2005

Report nr. 013CE2005
Control Engineering
EE-Math-CS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Summary

The aim of this Masters of Science assignment was to build a demonstration setup on which various control systems can be tested. This 'Mechatronic Demonstrator' is supposed to be used at lectures in control engineering in the Electrical Engineering and Mechatronics curriculum of the University of Twente. The objective is to support the theory by examples in practice on a mechatronic device. Besides making the theory more insightful, real limitations in practical setups can be shown easily.

Several options for demonstrators were investigated, of which the demonstration setup of Controllab Products B.V. (CLP), the developing company of the software tool '20-sim' appeared to be the most promising. The design of the setup is originated from a printer – it has a slider moving back and forth, with as special construction a flexible frame. Due to this, the demonstrator is assumed to behave like a fourth order model which makes it interesting for educational purposes.

Simulations were performed with PID- and LQG control algorithms in combination with a mathematical model of the CLP-setup. The resulting 20-sim models were tested on the demonstration setup of CLP. Differences between the control systems were shown on this physical setup, after which the decision was made to build our own demonstration setup, based on the design of CLP. The objective was to make the new demonstrator a compact device by integrating all parts in one device. The demonstrator turned out to be an integrated machine, small, though a bit heavy. Only a power cable and an Ethernet cable are required for normal operation, the system is plug-and-play and it has extended remote capabilities such as parameter variation and real-time variable monitoring. An especially interesting feature is the animation capability, which shows a 3D-representation of the setup moving along with the demonstrator itself in real time.

The demonstrator consists of the mechatronic parts copied from the CLP setup, embedded 600MHz PC, a PC104 I/O card, storage device with a real-time Linux operating system and peripheral electronics such as power supplies and a motor amplifier. Improvements that were implemented compared to the CLP setup involve integration of all system parts, a homing action based on index pulses of the linear measurement strips and a hardware security system.

PID- and LQG control algorithms have been tested on the mechatronic demonstrator. After being optimized in simulation according to a quadratic criterion, it appeared that the results in practice were quite different. Especially the friction elements seemed to be larger in practice than in the model. An addition to the LQG controller, in the form of an integration of the difference between process and Kalman filter model has been implemented. This so called LQG+ control algorithm showed substantial improvements compared to the PID and LQG controller.

Recommendations concerning the software and control systems mainly focus on implementing an extensive safety system. Furthermore, it is advised to deduct a better mathematical model of the plant. More experiments can be performed with other control systems such as MRAS, (L)FF, ILC, etc. Suggested improvements on the hardware of the mechatronic system are aimed at weight reduction, for instance by using another motor amplifier and only one (smaller) power supply. Furthermore, parallel processing will be an interesting research area that can be addressed by the demonstrator.

Key part in this assignment was the system integration. Several parts of the project were developed separately already. This assignment focused on combining knowledge to design, build and test an integrated mechatronic demonstration setup.

Samenvatting

Het doel van deze afstudeeropdracht was het bouwen van een demonstratieopstelling waar diverse regelalgoritmen op getest kunnen worden. Op den duur dient deze 'Mechatronic Demonstrator' gebruikt te gaan worden bij colleges in de regeltechniek zoals die gegeven worden in de curricula van Elektrotechniek en Mechatronica aan de Universiteit Twente. Het doel is het demonstreren van regeltechnische theorie op een fysieke opstelling. Naast het kweken van meer inzicht in regelaar theorie kunnen tevens de beperkingen van een praktijk opstelling gedemonstreerd worden.

Diverse opties voor demonstratieopstellingen zijn overwogen. De opstelling van Controllab Products B.V., ontwikkelaar van onder andere het modelvormings- en simulatiepakket '20-sim', bleek het meest veelbelovend. Het principe van deze opstelling is gebaseerd op een printer – er is een slider die heen en weer kan bewegen, met als speciale constructie een bewegend frame. Hierdoor kan de opstelling gerepresenteerd worden door een vierde orde model, hetgeen interessant is uit educatief oogpunt.

Talrijke simulaties zijn uitgevoerd met PID en LQG systemen in combinatie met een wiskundig model van de CLP-opstelling. Deze 20-sim modellen zijn vervolgens getest op het apparaat. Verschillen tussen deze regelstrategieën zijn aangetoond bij deze praktijktesten, waarna de beslissing is genomen een eigen versie van deze opstelling te realiseren. Het doel was de nieuwe demonstratieopstelling compact vorm te geven, door alle systeemonderdelen te integreren. Het resultaat werd een geïntegreerd apparaat, klein maar toch redelijk zwaar. Slechts een voedingkabel en een Ethernetaansluiting zijn toereikend voor normaal gebruik, het systeem is snel te installeren en het heeft zeer uitgebreide remote mogelijkheden zoals het aanpassen van parameters en het real-time monitoren van variabelen. Noemenswaardig aspect van de software is de 3D-animatie mogelijkheid, die een (mee-)bewegende representatie van het proces demonstreert in real time.

De nieuwe demonstratie opstelling bestaat uit de mechatronica van de CLP-opstelling, een embedded 600MHz PC, een PC104 I/O kaart, opslagmedium met een real-time Linux besturingssysteem en randapparatuur zoals voedingen en een motorversterker. Verbeteringen die zijn doorgevoerd ten opzichte van de CLP-opstelling omvatten de integratie van alle systeemonderdelen, een 'homing'-actie gebaseerd op index pulsen van de meetlinten en een hardware beveiligingssysteem.

PID- en LQG regel algoritmes zijn getest op de mechatronische demonstratieopstelling. Na optimalisatie in een softwareomgeving volgens een bepaald criterium bleek dat de resultaten in de praktijksituatie sterk verschilden van de theorie. Met name de wrijvingselementen zijn naar verwachting in de praktijk groter dan in het model. Er is een module toegevoegd aan de LQG controller, in de vorm van een integrator van de error tussen proces en Kalman filter model. Met dit zogenaamde LQG+ regelalgoritme zijn aanzienlijke verschillen aangetoond ten aanzien van de PID en LQG controllers.

Aanbevelingen met betrekking tot de software en regelsystemen zijn met name gericht op het implementeren van een gedegen veiligheidssysteem. Daarnaast is het verstandig een beter wiskundig model van de mechatronische demonstratieopstelling af te leiden. Meerdere experimenten kunnen uitgevoerd worden met andere regelsystemen als MRAS, (L)FF, ILC etc.
Verbeteringen wat betreft de hardware zijn geconcentreerd op gewichtsbesparing, bijvoorbeeld door een andere motorversterker toe te passen en slechts één (lichtere) voeding. Tot slot zijn gedistribueerde rekensystemen een interessant aandachtsgebied om te testen op de demonstratieopstelling.

Kerngebied van deze afstudeeropdracht was systeemintegratie. Diverse onderdelen van het project waren reeds afzonderlijk ontwikkeld. Deze opdracht was gericht op het combineren van kennis en kunde om uiteindelijk tot het ontwerp, de bouw en het testen van een geïntegreerde mechatronische demonstratieopstelling te komen.

Preface

With this report, I finish my study Electrical Engineering at the University.

I would like to thank the following people for making this thesis possible:

My supervisors Jan Broenink, Paul Weustink, Theo de Vries and especially Job van Amerongen for providing me with such an interesting assignment and the various insightful discussions about this topic.

The team of Controllab Products B.V., Christian Kleijn, Frank Groen, Johan Hemssems and again Paul Weustink for the good times at their office, the opportunity of working with their demonstration setup and all other support with my project.

The technical support staff, Marcel Schwirtz and Gerben te Riet o/g Scholten for the help with constructing the mechatronic demonstrator. Also Peter Scheeren and Gert Niers of TCO are thanked.

Marcel Groothuis – his contribution to the RT-Linux environment has been tremendous. Without his help, the project could not have been realized!

The rest of the people at the Control Engineering department, for the good discussions and lots of laughs.

Furthermore, special thanks go out to my best friends: Job Kamphuis, Arthur Baas, Erik Verdam, Manon Wevers, Matthijs Hettema and Hans Nijhuis.

My family, my father and sister for their continuing support. My mother deserves special attention – without her stimuli I would not have made it this far…!

Hans Dirne
Enschede, May, 2005

Table of Contents

1 Introduction

The aim of this Master of Science assignment is to build a demonstration setup on which various control systems are tested. The idea originates from the fact that the educational means of tutoring Control Engineering at the University of Twente are mostly theoretically oriented. In some cases the theory is supported by simulation lab works. However, using a real demonstration setup will give an extra dimension to the theory. Actually seeing a device putting control engineering in practice will make it easier for students to understand control engineering. Secondly, specific problems can be addressed that are inherent for practical situations, such as friction and non-linearities.

At first, the demonstration setup will be put into action at lectures in Control Engineering, such as 'Digital Control Engineering', 'Intelligent Control' and 'Mechatronics'. In a later stage, it can also be used in lab works where students will work with the setup themselves. During the design process of the device, a broad application area was kept in mind, so that the setup can be useful in more areas than control engineering. Examples can be thought of in: 'Embedded System Design', 'Real-Time Software Development' or 'Parallel Processing'.

1.1 Objectives

The objective of this Msc project is to design, build and test a mechatronic demonstration setup on which several control algorithms can be demonstrated. The working setup should be able to clearly demonstrate differences in performance between control algorithms of varying complexity in practice.

1.2 Scope

The design of a mechatronic demonstrator involves following a number of steps, as depicted in Figure 1.

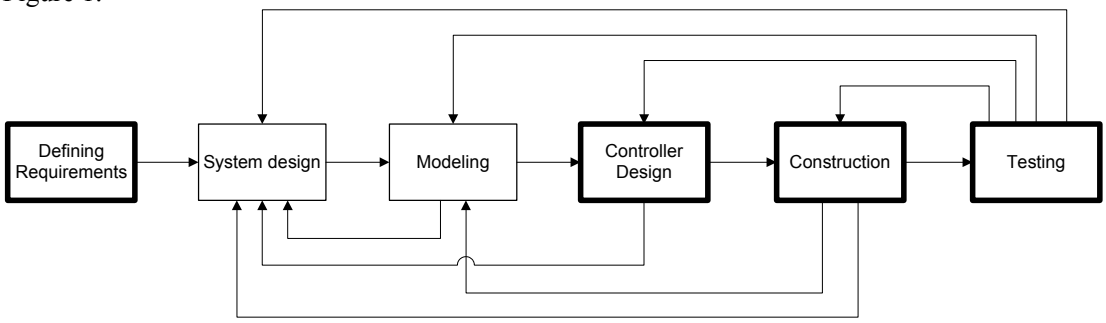


Figure 1: Design process of demonstration setup

The highlighted blocks indicate the parts that are investigated in this assignment. Each of these parts is treated extensively in this thesis. The requirements have been set up in collaboration with the primary users of the future demonstration setup: the professors that will use the setup during lectures. One of the demands on the new system is being able to run control systems in real time. A real-time system is an information system whereby correct functioning not only depends on the output of an algorithm but also on the time of delivery of the answer (Groothuis, 2004). A real-time system can be defined as a "system capable of guaranteeing timing requirements of the processes under its control" (Kopetz, 1997; Groothuis, 2004).

Proper system design and a competent mathematical model of the system are essential meeting the defined requirements. This project however will be based on an existing demonstration setup, which decreases the direct need for extended system design and mathematical models. Systems integration is the key part in this assignment. The goal to provide the 'Control Engineering' group of the University of Twente with a versatile and reliable demonstration setup at the end of the graduation project requires a view strongly aimed at results.

Clear demonstration of several control algorithms in practice is the purpose of the demonstration setup. In this project, the start of this is made by delivering a device on which these algorithms can be tested. First, controller design takes place by means of simulation with a model of the plant. Several iteration steps may have to take place in order to design a controlled system that meets the requirements. In future work, all mathematical control algorithms that are treated in the control engineering courses may be tested on the demonstration setup. One could think of a broad range of controllers that more or less require a proper mathematical model: a PID or a learning feed forward system that require little model knowledge versus for instance an LQG or a model reference adaptive controller. This assignment focuses on two different types of control systems, namely PID and LQG. Simulations have been performed in the software package '20-sim' (Controllab Products B.V., 2005). With 20-sim, the behavior of dynamic systems can be modeled and simulated. Interesting feature of this package is the extensive 3D-reprensentation possibilities, which make the results of simulations clear in a quick and insightful manner.

The construction part of the design process of a mechatronic demonstrator involves not only the actual realization of the physical device, but it incorporates the integration of all parts of the demonstrator: mechanics, real-time computer hardware and software packages.

Testing a newly built device will lead to many iteration steps as shown in Figure 1. It may even lead to redoing the system design if the criteria are not met. In this report the testing phase is concentrated on presenting the results of PID and LQG control systems. Testing the machine itself is not extensively treated since at the end of this assignment, the setup is presented as being functional and reliable.

1.3 Outline

Various physical setups are suitable in teaching control engineering in practice, but design choices have to be formulated in order to build a device that optimally suits the needs; chapter 2 describes the process of defining design choices and criteria. Once a design had been proposed, several control systems have been tested on a mathematical model (chapter 3). After showing that the suggested physical setup was able to demonstrate performance differences of several control algorithms (chapter 4), the construction of the demonstrator was initiated. An elaboration about the new mechatronic demonstrator is presented in chapter 5. Chapter 6 describes the results of tests performed on the demonstration setup. The last chapter of this report, chapter 7, discusses the conclusions and recommendations for this thesis.

2 Introduction to the demonstration setup

The new demonstration setup should be able to run control systems as developed in a software modeling environment. First, models will be generated and tested on a PC by means of simulation. At satisfying results, these models (for instance a PD control algorithm) can be put to the test on a real physical setup; see Figure 2.

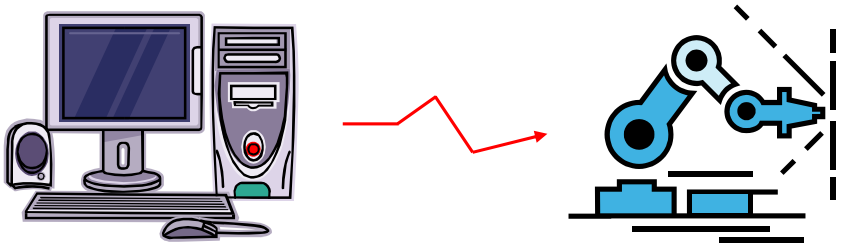


Figure 2: PC and demonstration setup picture

The following sections elaborate on several setups that might be used for demonstration purposes. Both possible demonstrators themselves as well as interfacing hardware will be treated. At first, demands on the setup will be presented in section 2.1.

2.1 Demands on the setup

A number of demands have been formulated which a new demonstration setup must fulfill. Since the lectures of the control engineering group mainly concentrate on *mechatronic systems*, the demonstrator should have a mechatronic nature. The study major that this assignment is part of is also called 'Mechatronics', which can be described as follows (Van Amerongen, 2005):

"Mechatronics is a synergistic approach to the integrated and optimal design of a mechanical system and its embedded control system, where solutions are sought that cross the borders of the different domains"

Many mechatronic systems in practical situations can be represented by a fourth order mathematical model. These systems have been treated extensively in the courses 'Mechatronics' and 'Control Engineering'. An example of a fourth order system is shown in iconic diagram representation in Figure 3.

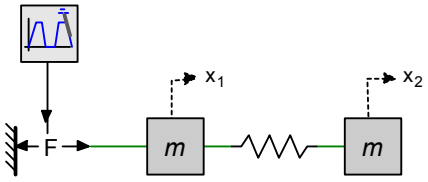


Figure 3: Basic fourth order mass-spring-mass system

A demonstration setup that can be described by a competent linear 4^{th} order model is ideal for educational purposes. The lowest eigenfrequency of the 4^{th} (or higher) order dynamic system has to be in a visible range, so smaller than about 15 Hertz.
Besides being used by professors in lectures, the setup might be used for future student practicals and projects. The design should therefore be robust, both in hardware durability as in overall fail safety. A side aspect of a robust design is easy replacing of possible broken parts of the setup. Furthermore, effort must be spent to ensure proper emergency handling, such as automatic shutdown. The system should preferable be portable, easy to connect and straight-forward in getting the demonstrator up-and-running.
For further educational purposes, a high level of observability is also desired. Both collocated and non-collocated control can then be demonstrated. Additionaly, high observability enables

comparison of actual measurements with the results of various state estimators. Therefore it is advised to implement sensors such that many states in the system can easily be obtained.
A clear link with a well known practical device is also advised: for the imagination of a control engineering task in practice, it is best if students can imagine the task to be resolved in a real situation.
Finally, the demonstration setup will be used to show various aspects of control engineering theory, such as:

1. System Identification e.g. frequency sweeps, STR, MRAS
2. Estimation e.g. LQE, STR, MRAS
3. Control Systems e.g. PID, (L)FF, MRAS
4. Practical Limitations e.g. noise, discretization, saturation
5. Parallel processing e.g. distributed control

The setup must be suitable for performing these tasks well and, be able to show the differences between control algorithms in practice; each controller (PID, LQG, MRAS etc) has its own specific strengths and weaknesses.

Summarizing, the following demands have been formulated for the new demonstration setup:

1. Mechatronic system
2. Representable by linear 4^{th} order model, with sufficiently low eigenfrequency
3. Portable and easy to set up
4. Robust, safe and failsafe design
5. High level of observability
6. Clear link with well known practical device
7. Clear demonstration of (differences between) various control algorithms

2.2 Demonstrator options

The mechatronic demonstrator to be built, can be based on one of the current setups at the Control Laboratory, but designing a totally new setup is also an option. Three possible setups are examined in more detail:

1. New build
2. ‘Linix’ laboratory setup
3. ‘Controllab Products’-setup

2.2.1 New build

The advantage of a new setup designed totally ‘from scratch’ is that there is complete freedom about the design and properties of the system. A device that can be thought of might be a ‘Pick-and-Place’-machine. This will require a lot of time, however. Furthermore, proceeding with this option will leave little time for control system research, which is also an important aspect of this assignment. Therefore, investigating other options is preferred.

2.2.2 Linix Laboratory Setup

The Linix demonstration setup is an insightful fourth order system equipped with two rotational inertias, a (variable) flexible transmission, a motor and two sensors (on both motor and load), as shown in Figure 4.

Figure 4: Linix laboratory setup

A mathematical model of the Linix demonstration setup is depicted in Figure 5.

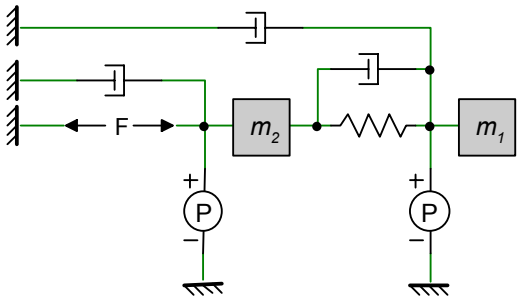


Figure 5: Model of Linix system

The Linix system has been extensively tested with PID control algorithms (Coelingh, 1997). The main disadvantage of the Linix system is the slip that occurs between the belt and the inertias, which introduces undesired non linearities in the system. This limits the bandwidth of the system. Performance increase by for instance an LQG-controller is therefore minimal. Looking back at the design criteria of the new demonstration setup, it can be concluded that this setup is robust, 4th order, observable and portable but, most importantly, makes a clear comparison between several control systems difficult due to the limited bandwith.

2.2.3 Controllab Products setup

CLP is a spin-off company (Controllab Products B.V., 2005) of the University of Twente. It is known as the developing company of the modeling- and simulation package '20-sim'. For promotional purposes, CLP designed and built their own setup for demonstrations at conferences and workshops (Kleijn, 2003). The setup originates from the principle of a printing device: a slider moving back and forth over a rail guide. The frame of the system is flexible, which introduces vibrations in the setup when the slider accelerates. See Figure 6 for a picture.

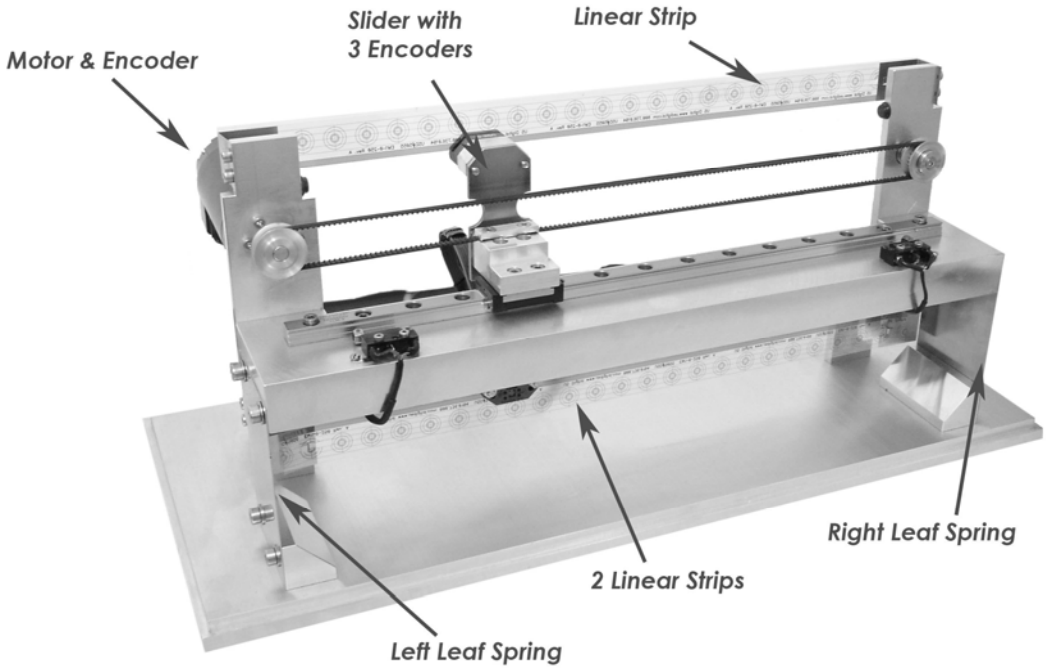


Figure 6: Demonstration setup of 'Controllab Products B.V.'

The educational aspect of this device lies in the vibrations in the frame: it should be able to show that 4th or 6th order controllers perform better (in amount of vibration / consumed power) than a lower order 2nd order (PD-like) control system. The resonance frequency can be chosen (e.g. by choosing different leaf springs) such that it lies in the visible range.

Comparing the three options described in the previous pages, the following overview can be made:

	New build	**Linix**	**CLP-setup**
Mechatronic system	yes	yes	yes
Linear 4th order model	yes	limited linear	to be determined
Portable, easy to set up	yes	yes	not in current form
Robust, safe, failsafe	yes	yes	feasible
Observability	yes	yes, 2 position sensors	yes, 4 position sensors
Link with practical device	yes	yes, a transmission	yes, a printing device
Shows controller differences	yes	no, due to nonlinearities	to be determined
Realizable	no	yes	yes

The CLP-setup is most promising of all three suggestions so we concluded that this will be investigated in more detail.

2.3 I/O options

The software tool ’20-sim’ not only provides a modeling and simulation environment for domain independent systems. It is also provided with a C-code generator, which allows easy integration in embedded applications. Figure 7 depicts the steps that should be undertaken to get a model up and running on a physical setup.

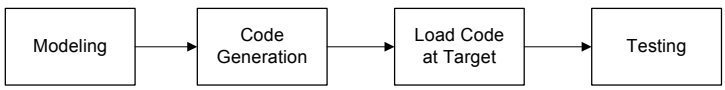


Figure 7: Procedure for testing a model in practice

In which the target consists of three main parts: CPU, communication module and I/O interface, as depicted in Figure 8.

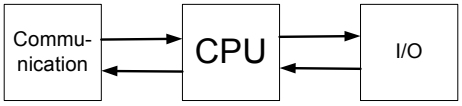


Figure 8: graphical representation of target

Two of the currently used systems at the Control Engineering department are presented here: the mixed signal DSP board by Analog Devices (ADSP-21992) and a real-time Linux operating system provided with I/O hardware. The following sections briefly present both systems and there advantages and weaknesses.

2.3.1 DSP board

The ADSP-21992 board can be connected to a PC by a USB-port, and has the following specifications:

- 160MHz mixed-signal DSP
- 32K program memory RAM
- 8 analog input signals (14 bit ADC)
- 8 analog output signals (12 bit DAC)
- 3 PWM output signals
- 2 auxiliary PWM output signals
- 3 encoder interfaces
- SPORT, SPI, UART and CAN interface

A software development environment named VisualDSP++ (ADSP, 2002), gives users the ability to compile, assemble, link and upload code to the DSP board, after this C-code has been generated by 20-sim. It is a compact solution, which contains all necessary functionality onboard: processing and I/O are combined. See Figure 9 for a photo of the DSP-board.

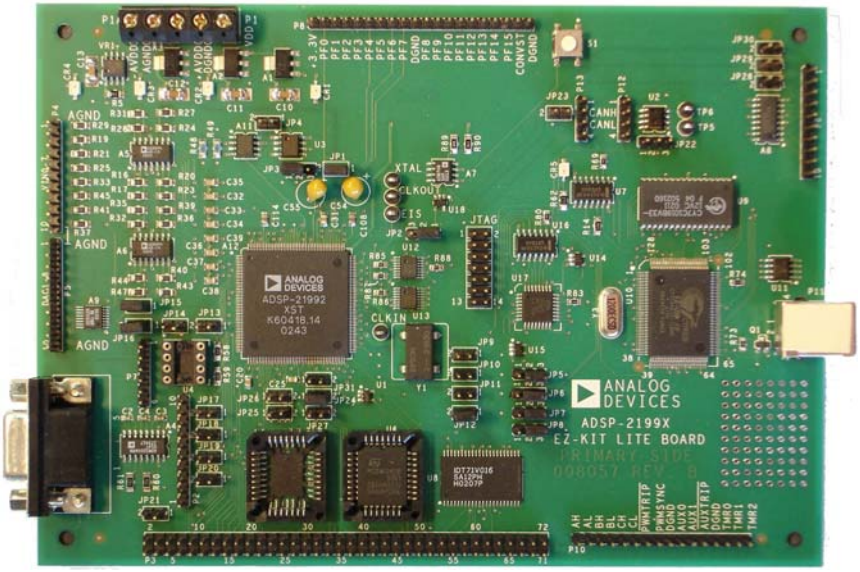


Figure 9: Analog Devices 21992 DSP board

During tests with the DSP-board it was noticed that uploading the models to the board requires several steps to be undertaken. Furthermore, at the time it was not possible to view variables of the model real-time; the only way to view the results of the test was to download the measurement data afterwards through the serial port. The two most important disadvantages of the DSP-board are the fixed point calculations (the system gets slow due to floating point emulations), and the current impossibility of the ADSP board to work with models that contain matrices and multibonds.

2.3.2 Real-time Linux operating system

Another option is a real-time Linux operating system in combination with I/O-interfaces. CLP uses an off-the-shelf product made by CosaTeq (CosaTeq, 2005). The software consists of client and server software. The Linux server module adds a real-time component to the kernel, which enables it to run certain programs at a specified sample rate. A (windows) client program can connect to the server and upload and start/stop models. Furthermore, parameters can be adjusted easily and variables can be monitored real-time. This system offers all functionality that is desired. The user interface is not optimal; for instance there are two separate programs for uploading and starting/stopping the models. The programmers at CLP however are working on their own windows client, which offers higher user friendliness. The CosaTeq server/client packages will be treated in more detail in sections 4.2 and 5.1.1.
Even though the user interface of the real-time Linux is not optimal, this option is preferred over a system based on a DSP-board due to its extended capabilities.

2.4 More detailed description of CLP-demo setup

Based on the defined criteria and the research for possible mechatronic setups and I/O-interfacing, the Controllab Products setup in combination with real-time Linux is suggested as the best option for a demonstration setup. Currently, the CLP-setup consists of several parts which all together make the setup not easily portable. Here is a list of the current parts needed for the setup:

1. PC running Windows (connected to Ethernet)
    a. 20-sim installed – models can be generated here
    b. Client software for uploading and starting/stopping models installed
2. PC running Linux (connected to demonstration setup and Ethernet)
    a. Real-time component and server software installed
    b. PCI I/O boards present
3. External box with motor controller
4. Demonstration setup

See Figure 10 for an overview of the setup.

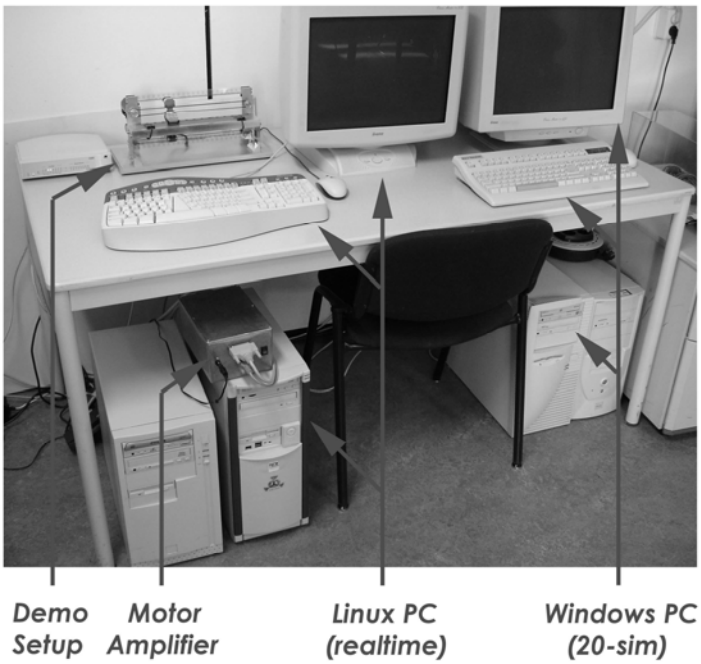


Figure 10: Overview of current CLP-setup

Concentrating on the mechatronic demonstration setup itself, four encoders (sensors) are attached, which measure the following variables:

1. Motor position (rotary encoder)
2. Position of slider with respect to the fixed world (linear encoder)
3. Position of slider with respect to the frame, upper sensor (linear encoder)
4. Position of slider with respect to the frame, lower sensor (linear encoder)

See Figure 11 for a rear view of the setup including sensor denotation.

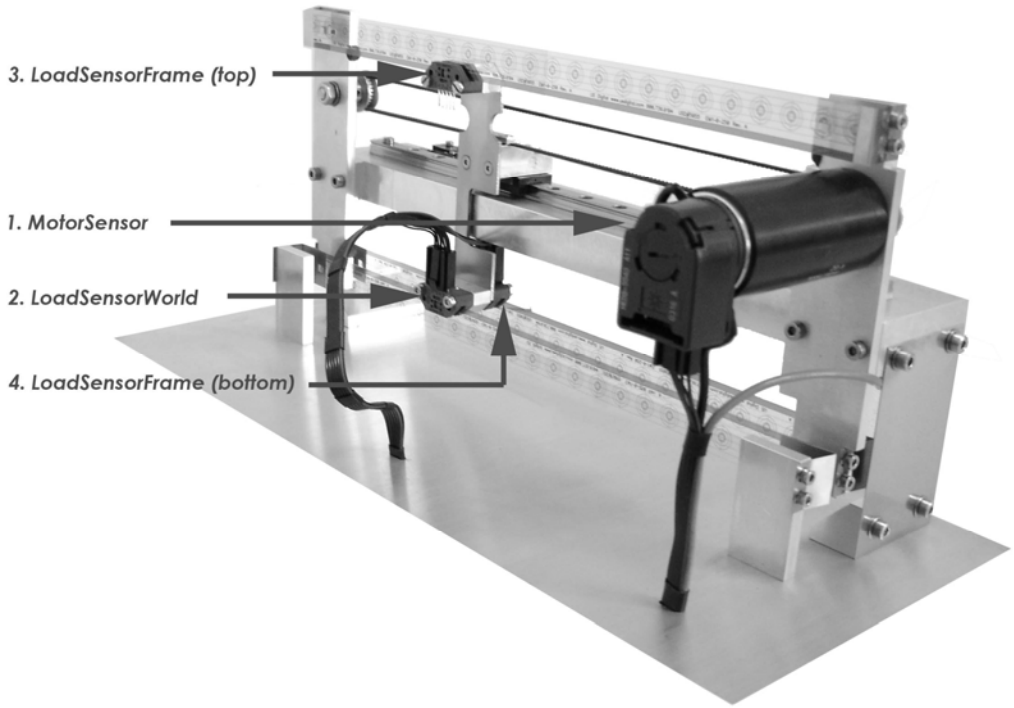


Figure 11: Four encoders of CLP-setup (rear view)

Using two sensors that measure the position of the slider with respect to the frame was originally intended for measuring the small rotation of the slider during movement. In practice however, the rotation appeared to be so small that the linear strips had too low resolution for measuring it.

The motor sensor offers possibilities to test collocated control. The combination of frame- and fixed-world sensors makes it possible that control systems with an order higher than two can be tested.

2.5 Model

2.5.1 6^{th} order non linear model

A mathematical model has been made by the designers of the setup (Figure 12). It is a 6^{th} order model, since both the flexibility of the frame and the flexible toothed belt are included.

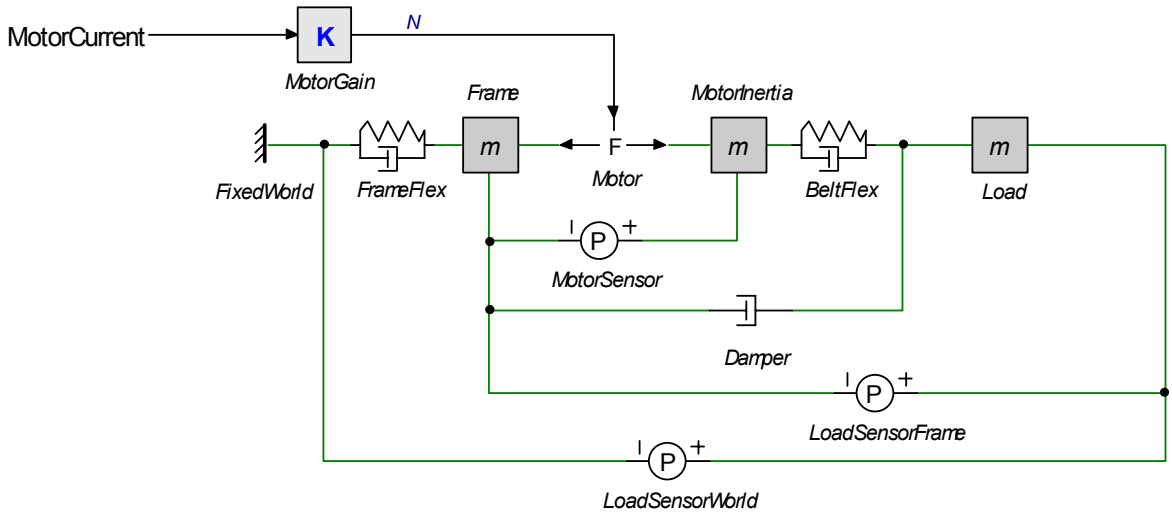


Figure 12: 6th order model of CLP-setup

The parameters are as follows:

Element	Parameter	Value
Motorgain	Motor constant	5.7
Frame	Mass of frame	0.8 kg
MotorInertia	Inertia of motor	1e-5 kg
Load	Mass of end effector (slider)	0.3 kg
FrameFlex	Spring constant	6 KN/m
	Damping in frame	6 Ns/m
BeltFlex	Spring constant	800 N/m
	Damping in belt	1 Ns/m
Damper	Viscous friction	3 Ns/m
	Coulomb friction	0.5 N

These parameter values have been calculated during the design process, based on construction drawings. They have not been extensively verified by an identification process. For future experiments in 20-sim, we assume these values to be correct. Note that the 'Damper'-element contains a coulomb friction part, which makes the model non-linear. In practice also stiction will be present in the system, but this is not taken into account in this model.

2.5.2 4th order linear model

For educational purposes a competent linear 4th order model is desired (see section 2.1). Downsizing the system order as well as linearization is therefore required.
Linearizing a model means making a linear representative of a non-linear model at a certain point in time or certain state of the system. A linear system description is required for instance when linear state estimators are to be designed (which is one of the objectives). In the case of the mechatronic demonstrator the non-linearities mainly consist of:

1. Coulomb friction
2. Stiction
3. Actuator saturation

Actuator saturation is not an issue in this stage since only the mechatronic demonstrator itself is investigated, without examining any I/O interfacing that will limit the range. Care must be taken when these parts of the system are included in the model.

The force-velocity relation of for instance the slider however consists for a great part of nonlinearities – see Figure 13 for an overview of the elements of a standard force-velocity relation.

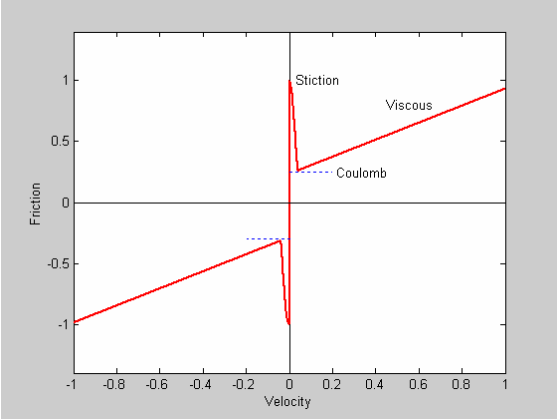


Figure 13: Force-Velocity diagram showing stiction and viscous&coulomb friction

Three different sections can be noticed: stiction, coulomb friction and viscous friction. The viscous part is the only linear element in the diagram. Stiction only occurs at the transition phase from a non-moving to a moving state. Coulomb friction is an offset in the diagram, depending on the direction of motion. In the previously presented model of the system (Figure 12), only coulomb- and viscous friction are taken into account, by the following force-velocity relation in the element 'Damper',

$$\boldsymbol{F} = \mathrm{d}\,\boldsymbol{v} + \mathrm{d}_c \tanh(1000\,\boldsymbol{v})$$

In which 'd' represents the viscous friction and 'd_c' the coulomb friction. Figure 14 depicts the relation in a graph.

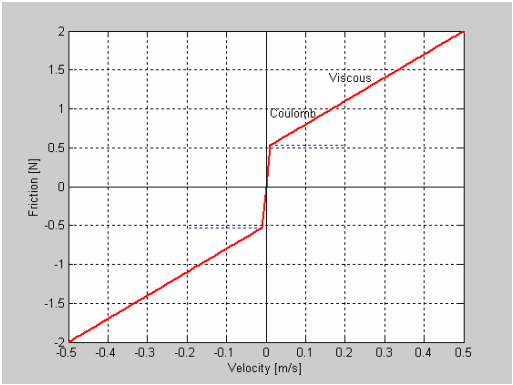


Figure 14: Friction in model of CLP setup

The friction model of the CLP-setup can be linearized by eliminating the coulomb part, resulting into only a (linear) viscous friction. If the coulomb friction is still included when the linearization tool in 20-sim is called, the friction-velocity relation will have a steep gradient due to the *tanh*(.) function.

By examining the pole-zero plot of the linear system, decisions can be made about the model reduction. Intuitively, the belt flexibility (non dominant stiffness) and the motor inertia should be removed from the 6th order model, since the (transformed) mass "motor inertia" can be approximated by zero. Concentrating on the springs in the system, the flexibility due to the belt should be removed, in order to remain only one spring; the frame flexibility. Checking the pole-zero plot is advised, to examine the effect of these propositions; only the non-dominant poles and zeros should be removed in order not to change the dynamical system too much.

Checking the locations of the poles and zero's of the 6th order linear system leads to the following overview (system description from "MotorCurrent\input" to "LoadSensorWorld\position" of Figure 12).

$z_1 = -3.75 + 86.52i$	(dominant)
$z_2 = -3.75 - 86.52i$	(dominant)
$z_3 = -800$	(zero far away – *not* dominant)
$p_1 = -9.92e+4$	pole far away – *not* dominant)
$p_2 = 6.075e-13$	(dominant)
$p_3 = -10.05$	(dominant)
$p_4 = -5.6 + 86.2i$	(dominant)
$p_5 = -5.6 - 86.2i$	(dominant)
$p_6 = -806.5$	(pole far away – *not* dominant)

Eliminating the mentioned spring and mass from the 6th order system leads to the pole-zero plot in Figure 15. Now, only the dominant zero's and poles from the 6th order system remain.

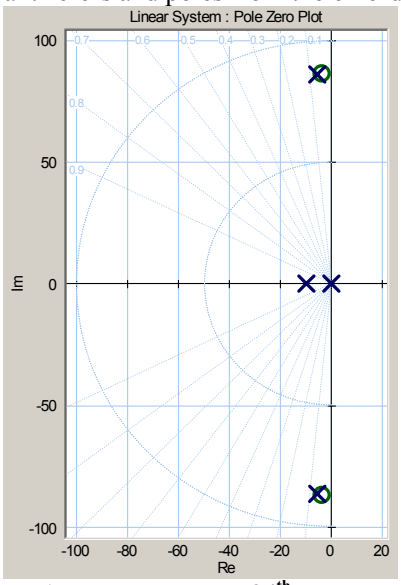


Figure 15: Pole-zero plot of 4th order system

A comparison in the time domain of the 4th and 6th order system to an input pulse has been made, according to the model in Figure 16.

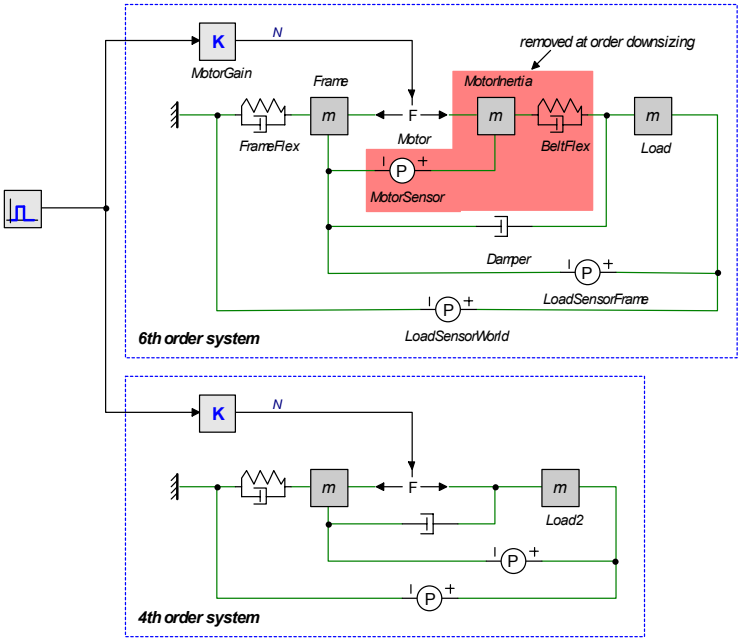


Figure 16: Comparison of 6th- and 4th order models

The results in the simulator are comparable - see Figure 17.

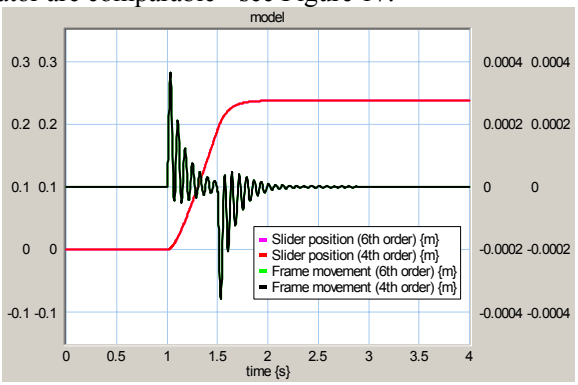


Figure 17: 6th and 4th order systems are comparable

A difference between the two systems exists, due to the extra phase lag in the 6th order system, caused by the elements removed in the 4th order system. A small delay is visible when the error is plotted (LoadSensorWorld 6th order minus 4th order) – see Figure 18: the 6th order system lags behind the 4th order system.

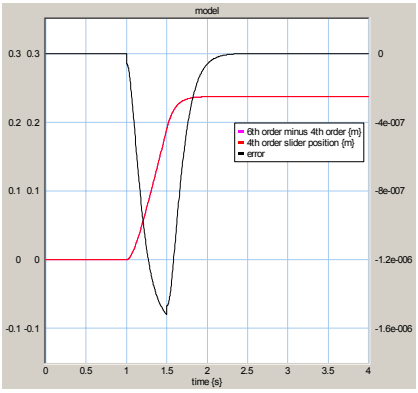


Figure 18: Phase difference due to model reduction

Since the poles and zeros that were removed by the model reduction were non-dominant and the simulation showed little effect of the model reduction, the reduced 4th order model is assumed to be competent.

The following linear 4th order system is the result of the linearization and order-reduction process:

$$\dot{\boldsymbol{x}} = \mathbf{A}\,\boldsymbol{x} + \mathbf{B}\,\boldsymbol{u}$$
$$\boldsymbol{y} = \mathbf{C}\,\boldsymbol{x} + \mathbf{D}\,\boldsymbol{u}$$

In which:

x_1 = velocity end effector
x_2 = position end effector with respect to fixed world
x_3 = velocity of the frame
x_4 = stretching of frame leaf springs
u = current through motor,
y = position end effector with respect to fixed world (= x_2)

and

$$\mathbf{A} = \begin{bmatrix} -10 & 0 & 10 & 0 \\ 1 & 0 & 0 & 0 \\ 3.75 & 0 & -11.25 & 7500 \\ 0 & 0 & -1 & 0 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 19 \\ 0 \\ -7.125 \\ 0 \end{bmatrix}$$

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

This system has been implemented in various configurations (iconic diagram, state space, bond graph, linear system) - see Figure 19. Other types of representations may be required for controllers that need a reference model. Figure 20 shows that these models are really equal.

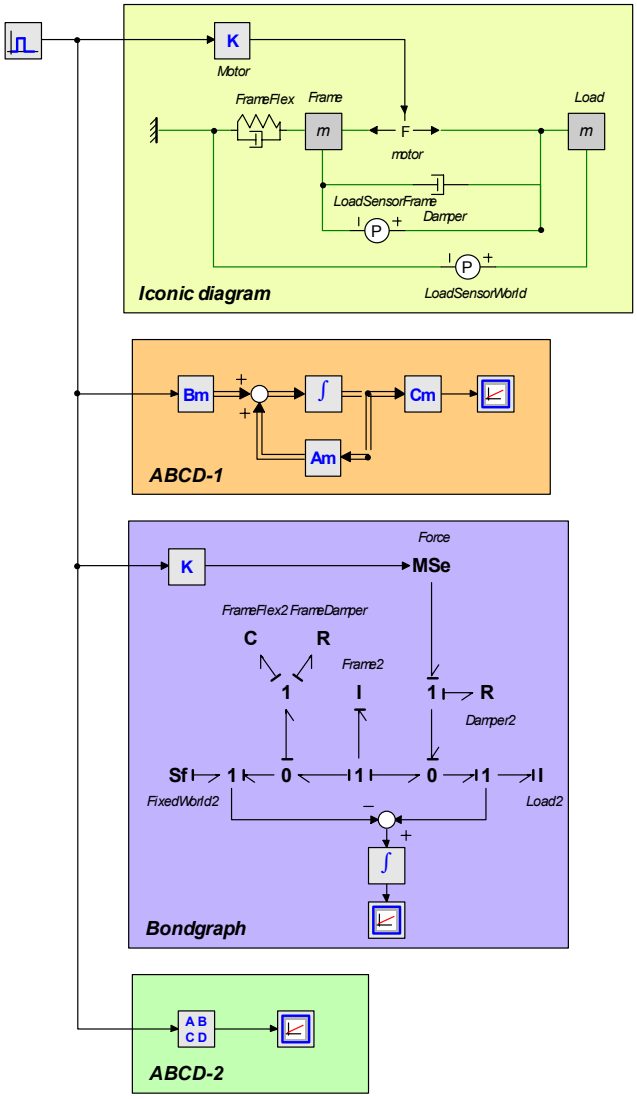


Figure 19: Various implementations of 4th order linear systems

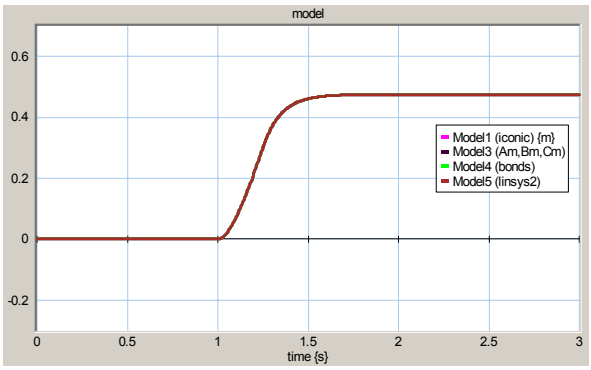


Figure 20: The results are equal

2.5.3 Interfacing

The mechatronic demonstrator will be a system controlled by a computer, which requires interfacing elements to connect the (analog) demonstrator to the (discrete) computer. The CLP setup is provided with a digital-to-analog converter (DAC). Digital encoders are attached to the

sensors that measure the positions of the slider. Figure 21 depicts the 6th order model, expanded with interfacing. The shaded area is the analog part of the model, to which encoders and a D/A convertor have been added. This D/A convertor is connected to blocks that represent the interfacing electronics, such as the power amplifier. Furthermore, counters are added to convert the encoder signals ('ticks') to 'position' expressed in meters.

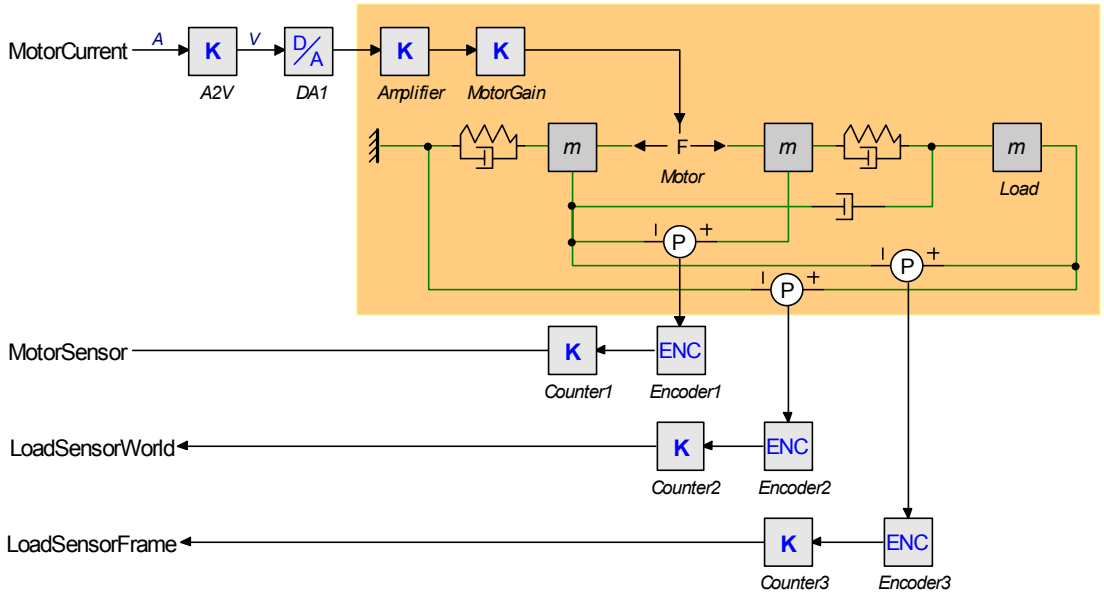


Figure 21: CLP setup model expanded with interfacing

The parameters of these components are as follows:

Element	Parameter	Value
A2V	Current to voltage conversion	10 V/A
DA1	Min, Max	+/- 10V
	Resolution	12 bits
Amplifier	Voltage to current conversion	0.1 A/V
Motorgain	Current to force (incl. transmissions)	5.7 N/A
Encoder 1,2,3	Resolution	24 bits
	Counts / rev	40.000 m^{-1}
Counter 1,2,3	Gain, distance per count	$2.5*10^{-5}$ m

3 Control Systems

The model presented in chapter two can be controlled by various control algorithms. Concentrating on the educational aspect of the mechatronic demonstrator, the following control algorithms can be thought of:

- Proportional, differential, integral (PID)
- Model reference adaptive systems (MRAS)
- Feed forward (FF)
- Linear quadratic regulators (LQR)
- Linear quadratic estimators (LGE)
- Linear quadratic gaussian (LGQ)
- Self tuning regulators (STR)
- Iterative learning control (ILC)

This assignment focuses on PID- and Linear Quadratic systems. The objection is to show the difference in performance between various control algorithms. In future work, more control algorithms can be tested.

3.1 PID

A proportional-integral-differential (PID, see Figure 22) controller is one of the most popular control algorithms in industry. In fact a PD algorithm can be considered as a state-feedback controller for a second-order system. The integral term ensures that the average error is driven to zero, under certain conditions (Van Amerongen, 2001). The PID structure can be simplified by setting one or two of the gains to zero, which will result in for instance a PI or PD control algorithm.

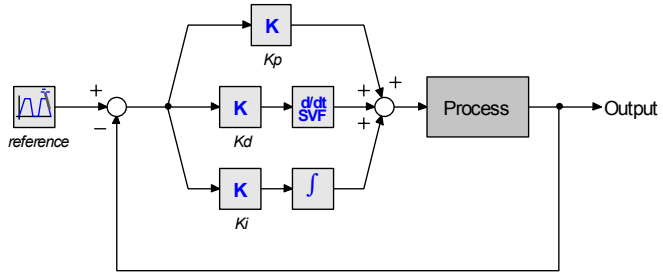


Figure 22: A PID controlled system

The mechatronic demonstrator will be a system controlled by a computer, which requires a discrete control algorithm and model – see Figure 23.

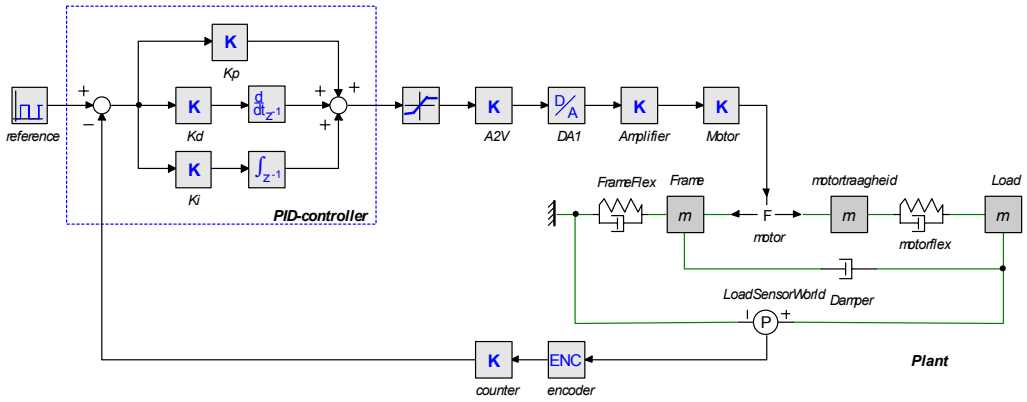


Figure 23: PID Controlled System

To prevent continuous integration of the control error in the case of actuator saturation, an anti-windup scheme is required. Then, the integrator is reset when the actuator saturates – see Figure 24.

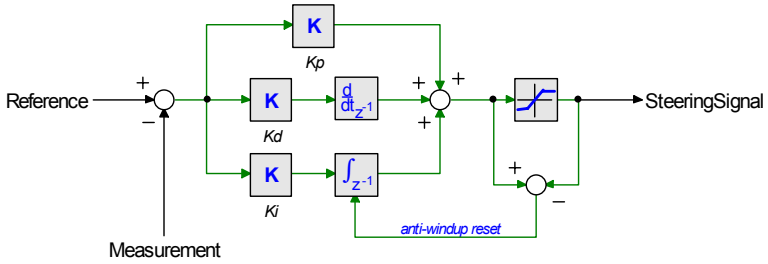


Figure 24: PID control algorithm with anti-windup

As a tuning criterion, the actuator is allowed to clip (only) at the acceleration phase. Control parameters are chosen as follows.

$K_p = 60$
$K_d = 1$
$K_i = 25$

See Figure 25 for the results of the simulation.

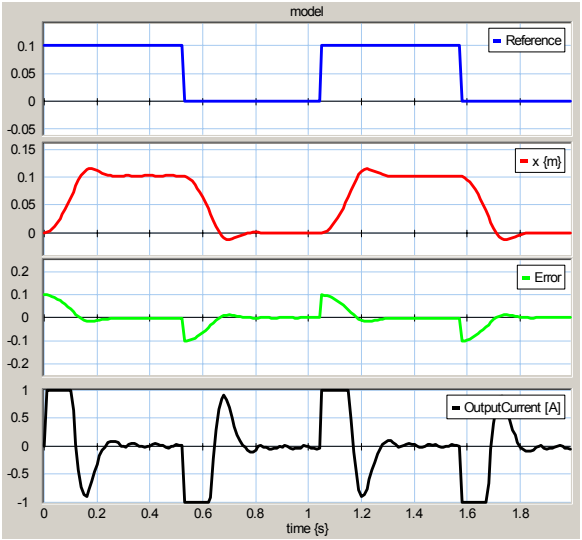


Figure 25: Results of second PID simulation

The current shows a fluctuation around zero of 11Hz, which is caused by controller counteracting the resonance of the frame. Due to the fact that the PID controller is tightly tuned, the resonances of the structure are excited. The current PID control algorithm assumes a second order model, whereas the mechatronic demonstrator can be represented competently by a fourth order model. Higher-order effects therefore can hardly be controlled by the PID algorithm. This can be overcome by reducing the control parameters of the PID system or by choosing a higher order control algorithm. Furthermore, filtering the reference signal to eliminate frequencies higher than the resonance of the system will minimize these effects. The maximum acceleration of the reference generator (currently: 15 m/s^2) is lower than the maximum achievable acceleration of the system (17.8 m/s^2).

3.2 LQR

A Linear Quadratic Regulator (LQR) is a control algorithm based on the principle of state feedback. A state feedback system is depicted in Figure 26.

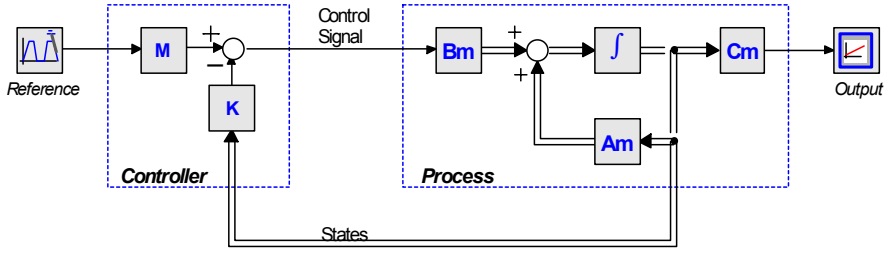


Figure 26: Principle of State Feedback

The internal states of the system are fed back to the controller which converts it to a steering signal for the process. Three parts are required for this type of feedback controller (Van Amerongen and The Vries, 2001):

- A mathematical model of the plant to be controlled
- A criterion for optimization of the state feedback
- Availability of the internal states of the system (either by measuring or by reconstruction)

3.2.1 Mathematical model of the system

The derivation of a model of the system has been described in section 2.5. A competent fourth order was deducted.

3.2.2 Criterion for optimization

The criterion of a state feedback system determines which optimization of the control algorithm is used in order to calculate a steering signal. Any optimization criterion can be chosen, depending on the application of the control algorithm. For the linear quadratic regulator (LQR) described here, the following criterion is used

$$J = \int \left(\boldsymbol{e}^T \mathbf{Q}\, \boldsymbol{e} + u^T \mathbf{R} u \right) dt$$

It punishes error as well as steering energy. By changing parameters Q and R, the emphasis can be placed on controlling steering energy or speed of convergence of the output. The elements of matrix Q determine which states are most important to control.

3.2.3 Internal states of the system

The mechatronic demonstrator is a SIMO system, consisting of one input (motor current) and three outputs (position of: 1.motor 2.slider with respect to fixed world 3. slider with respect to frame). Downsizing the model to a 4th order system (see section 2.5.2) 'eliminates' the motor sensor, leaving only the two sensors mounted at the slider as output signals. Taking into account that these are the only measured states in the system, the controller needs to be made such that these states can be used as an input for the controller. The states of the system as described in section 2.5.2 are not corresponding to this, since they were defined as follows

x_1 = *velocity end effector*
x_2 = *position end effector with respect to fixed world* *(measurement available)*
x_3 = *velocity of the frame*
x_4 = *stretching of frame leaf springs* *(measurement not available)*

The current CLP-setup does not supply direct measurements of the stretching of the frame leaf springs. A state transformation is therefore desired to obtain a state description based on the following states

x^*_1 = *velocity end effector with respect to fixed world*
x^*_2 = *position end effector with respect to fixed world*
x^*_3 = *velocity end effector with respect to the frame*
x^*_4 = *position end effector with respect to the frame*

This state description matches with the position measurements available at the CLP-setup. Velocities are not measured at the setup, but these can be reconstructed by for instance 'state-variable filters'. The motorsensor is not used since this only provides information about the non-dominant behavior of the geared belt.

The states can be transformed by a matrix H as follows

$$\boldsymbol{x}^* = \mathbf{H}\,\boldsymbol{x} \rightarrow \dot{\boldsymbol{x}}^* = \dot{\mathbf{H}}\,\boldsymbol{x} + \mathbf{H}\,\dot{\boldsymbol{x}} = \mathbf{H}\,\dot{\boldsymbol{x}}$$
$$\dot{\boldsymbol{x}}^* = \mathbf{H}\left(\mathbf{A}\,\boldsymbol{x} + \mathbf{B}u\right) = \mathbf{HA}\,\boldsymbol{x} + \mathbf{HB}u$$
$$\left[and : \boldsymbol{x}^* = \mathbf{H}\,\boldsymbol{x} \rightarrow \boldsymbol{x} = \mathbf{H}^{-1}\,\boldsymbol{x}^*\right]$$
$$\dot{\boldsymbol{x}}^* = \mathbf{HAH}^{-1}\,\boldsymbol{x}^* + \mathbf{HB}u \xrightarrow{\mathbf{HAH}^{-1}=\mathbf{A}^* \text{ and } \mathbf{HB}=\mathbf{B}^*}$$
$$\dot{\boldsymbol{x}}^* = \mathbf{A}^*\,\boldsymbol{x}^* + \mathbf{B}^*u$$

Furthermore

$$\boldsymbol{y} = \mathbf{C}\,\boldsymbol{x} + \mathbf{D}u$$
$$\left[again : \boldsymbol{x}^* = \mathbf{H}\,\boldsymbol{x} \rightarrow \boldsymbol{x} = \mathbf{H}^{-1}\,\boldsymbol{x}^*\right]$$
$$\boldsymbol{y} = \mathbf{CH}^{-1}\,\boldsymbol{x}^* + \mathbf{D}u$$

Now the dynamics of the system have been expressed in the new state vector x^*. Matrix H describes the relation between x and x^*. This relation can be deducted by knowledge of the plant (for instance, it is known that the position of the slider with respect to the frame is the position of the slider with respect to the fixed world plus the stretching of the leaf springs). It is advised that these results are verified by simulation, to exclude errors for instance in the signs. In the case of the mechatronic demonstrator, the relations are defined as follows

$x1^* = x1$
$x2^* = x2$
$x3^* = x1 - x3$
$x4^* = x2 + x4$

which leads to the following H-matrix

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Care must be taken in choosing the states with which the system will be described. As a rule of thumb, for an n-th order mechatronic system (n is even), $n/2$ velocities and $n/2$ positions should be chosen. In correspondence to the derivations presented earlier, the new system matrices can be calculated as follows

$$\mathbf{A}^* = \mathbf{HAH}^{-1}$$
$$\mathbf{B}^* = \mathbf{HB}$$
$$\mathbf{C}^* = \mathbf{CH}^{-1}$$
$$\mathbf{D}^* = \mathbf{D}$$

In combination with the original state matrices described in section 2.5.2, this leads to the following state space description

$$\mathbf{A}^* = \begin{bmatrix} 0 & 0 & -10 & 0 \\ 1 & 0 & 0 & 0 \\ 7.5 & 7500 & -21.3 & -7500 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \mathbf{B}^* = \begin{bmatrix} 19 \\ 0 \\ 26.125 \\ 0 \end{bmatrix}$$

$$\mathbf{C}^* = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \qquad \mathbf{D}^* = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$$

Since a discrete LQR-controller will be implemented in the system, discretization of these state matrices is required. This has been done by the C2DM-command in Matlab, assuming a zero-order-hold discretization process (corresponding to the mechatronic demonstrator) and a sample frequency of 1KHz. The results.

$$\mathbf{A}^*_{discrete} = \begin{bmatrix} 1.0000 & -0.0372 & -0.0099 & 0.0372 \\ 0.0010 & 1.0000 & -0.0000 & 0.0000 \\ 0.0111 & 7.4115 & 0.9752 & -7.4115 \\ 0.0000 & 0.0037 & 0.0010 & 0.9963 \end{bmatrix} \qquad \mathbf{B}^*_{discrete} = \begin{bmatrix} 0.0189 \\ 0.0000 \\ 0.0259 \\ 0.0000 \end{bmatrix}$$

$$\mathbf{C}^*_{discrete} = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \qquad \mathbf{D}^*_{discrete} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$$

With this discrete state space description of the mechatronic demonstrator, the last part of the three requirements for a state feedback system has been supplied.

3.2.4 Optimal state feedback

The output of the state feedback controller is.

$$u = -\mathbf{K}\boldsymbol{x}$$

In which x represents the state of the system and K is the gain vector based on the optimization criterion and the system model. The 'Ricatti-equation' determines the solution of the optimization problem (Van Amerongen, 2001) and therefore also the optimal gain vector K. The continuous time solution is.

$$\mathbf{A}^T\mathbf{P} + \mathbf{PA} + \mathbf{Q} - \mathbf{PBR}^{-1}\mathbf{B}^T\mathbf{P} = \mathbf{0}$$
$$\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}$$

In which

A, **B** = state matrices of plant to be controlled
Q, **R** = matrices in the optimization criterion (section 3.2.2)
P = solution to Ricatti equation
K = state feedback vector

The solution of the optimization problem in discrete time is the following (Van Amerongen, 2001)

$$\mathbf{A}^T\mathbf{PA} - \mathbf{P} - \mathbf{A}^T\mathbf{PB}\left(\mathbf{B}^T\mathbf{PB} + \mathbf{R}\right)^{-1}\mathbf{B}^T\mathbf{PA} + \mathbf{Q} + \mathbf{P} = \mathbf{0}$$

$$\mathbf{K} = \left(\mathbf{B}^T\mathbf{PB} + \mathbf{R}\right)^{-1}\mathbf{B}^T\mathbf{PA}$$

Implementation of the discrete LQR controlled system in 20-sim results in the model of Figure 27.

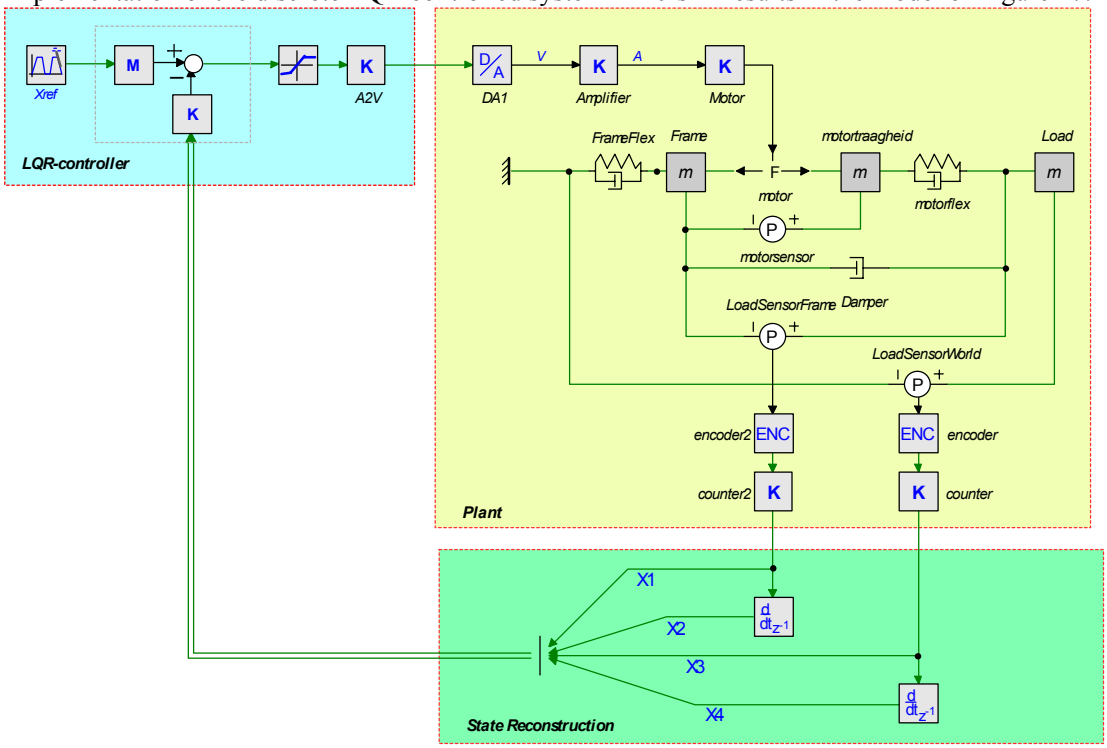


Figure 27: Mechatronic Demonstrator with LQR controller

The controller itself contains code to calculate the solution to the ricatti equation and to determine the optimal output signal. The output signal is in this case

$$Output = \left(\mathbf{K}[2] + \mathbf{K}[4]\right)REF - \mathbf{K}\,\boldsymbol{x}$$

The results of the simulation are depicted in Figure 28.

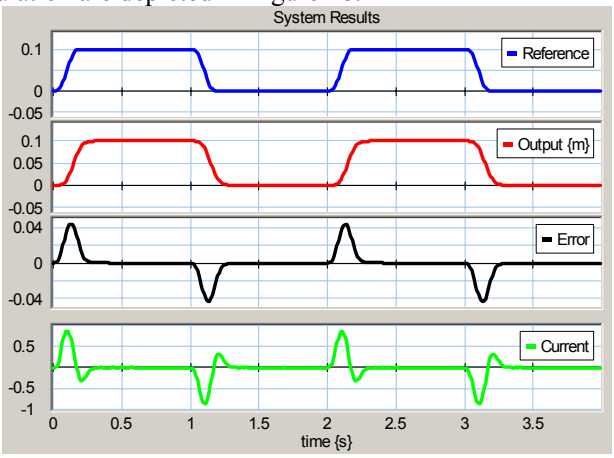


Figure 28: Simulation results of LQR controlled system

This simulation shows hardly any static error, a quick response and low energy use. The parameters of the controller are set to

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix} \quad \mathrm{R} = 0.1$$

This leads to the following controller gains: $\mathbf{K}_{LQR} = \begin{bmatrix} 1.7 & 27.7 & 0.008 & 16.3 \end{bmatrix}$

These values make sure the controller takes both the slider position and the frame vibration into account. Furthermore, the output signal is not clipping at these settings. Further tuning will take place in section 3.5, where several control systems will be compared to each other in practice.

3.3 LQE

Some control algorithms require information about the internal states of the system, such as the linear quadratic regulator treated in the previous section. Some of the states of the system can be measured (positions of sliders with respect to fixed world and frame) and others (velocities) have to be deducted for instance by applying state-variable filters. An other way to estimate the internal states of the system (without introducing phase lags) is by using 'observers'. An observer is based on a mathematical model of a process (Åstrom and Hägglund, 1995). It is driven by the control signals to the process and the measured variables. Its output is an estimate of the state of the system. An observer offers the possibility of combining mathematical models with measurements to obtain signals that can not be measured directly. The basic principle of operation is depicted in Figure 29.

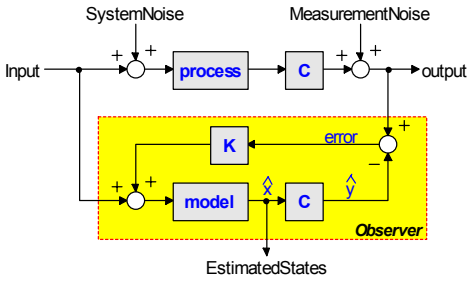


Figure 29: Principle of an observer

One type of observer is extensively treated in control theory: a Linear Quadratic Estimator, known as a Kalman filter. This observer provides an optimal estimate of the states of the system in the presence of system noise and measurement noise. The internal states are an approximation of the real states of the system. The algorithm works similarly to a LQR: solving the ricatti equation and calculating a gain matrix, which in the case of the LQE feeds the error between process and model (formed by disturbances and modeling errors) back to the states of the model. The equations for both a continuous and a discrete LQE are as follows

Continuous case

$$\mathbf{AP} + \mathbf{PA}^{\mathrm{T}} + \mathbf{Q} - \mathbf{PC}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{CP} = \mathbf{0}$$
$$\mathbf{L} = \mathbf{CPR}^{-1}$$

Discrete case

$$next(\mathbf{P}) = \mathbf{A}(\mathbf{I} - \mathbf{LC})\mathbf{PA}^{\mathrm{T}} + \mathbf{Q}$$
$$\mathbf{L} = \mathbf{PC}^{\mathrm{T}}\left(\mathbf{CPC}^{\mathrm{T}} + \mathbf{RI}\right)^{-1}$$

Within the framework of this assignment propositions were made for a wizard that automatically generates a Kalman filter. This wizard should guide the user to generate code and an icon in 20-sim by the following steps

Step 1 - Presentation of system matrices, indication of time domain (continuous or discrete) and optionally the sample time, based on Linear System Editor
Step 2 - Requesting estimator parameters (KF matrices 'Q' and 'R'). Dimensions of matrices are preset
Step 3 - Asking for dynamic or static solution of the 'Ricatti equation'
Step 4 - Summary of results – overview of KF properties
Step 5 - Kalman filter is added to 20-sim model with proper icon

Appendix II elaborates more on the propositions for the Kalman filter wizard.
The following section puts a Kalman filter in practice, in the function of a state estimator.

3.4 LQG

A Linear Quadratic Gaussian controller is a combination of a Kalman filter (LQE) and a Linear Quadratic Regulator (LQR) – see Figure 30. The LQE performs the state-estimation of the process, in order to calculate an optimal state feedback by the LQR.

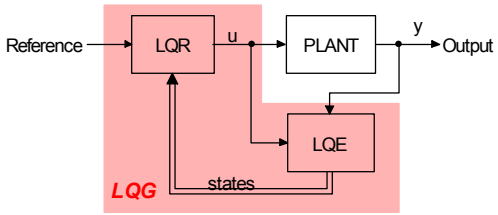


Figure 30: LQG explanation

The steps undertaken to obtain a 4th order discrete LQG controller are the following

1. Design of a competent 4th order model of the plant
2. Linearization of this model
3. Transformation of states (matching with measurements)
4. Discretization of the result to the desired sample frequency
5. Implementation of these discrete state space represtation in both LQR and LQE
6. Set LQG-parameters Q and R:
    - LQR: state feedback
    - LGE: state estimation

The resulting model of process and LQG-controller in the case of the mechatronic demonstrator is depicted in Figure 31.

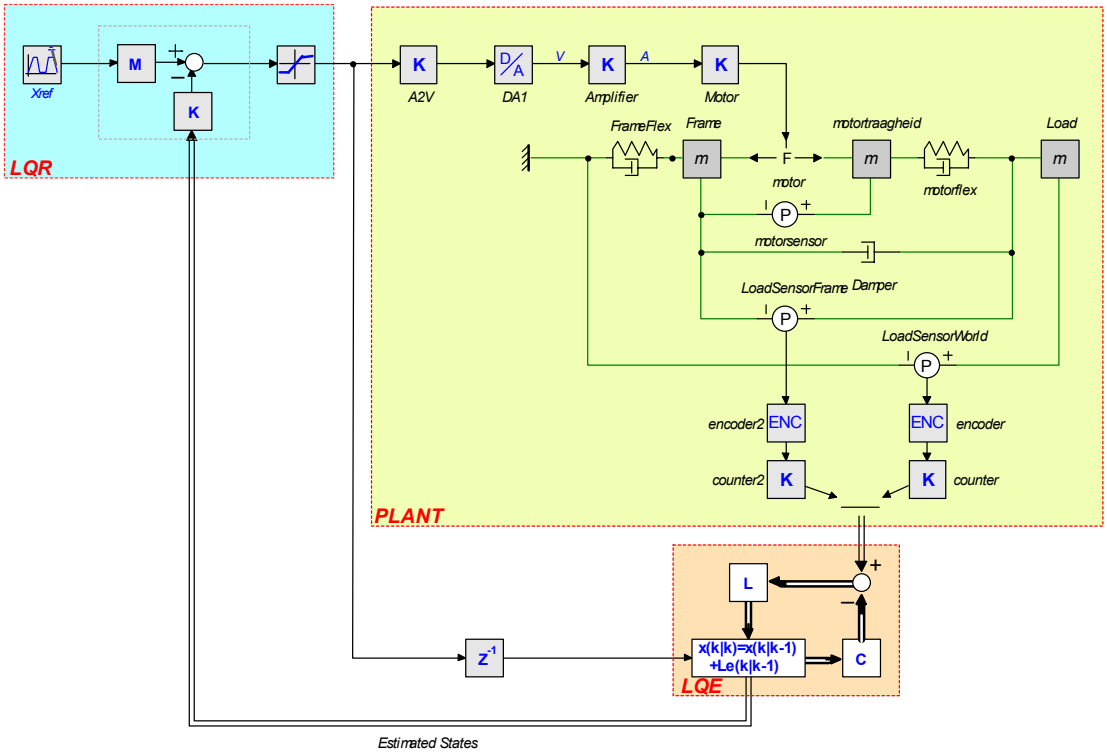


Figure 31: Linear Quadratic Gaussion (LQG) controlled system

A delay in the input signal of the LQE is required to avoid algebraic loops. Furthermore, a signal limiter in the LQR (set to a slightly smaller range than the D/A converter) is required to send the same signal to the LQE as to the demonstrator itself (where the signal is automatically limited by the D/A-converter).

The following settings were used to test the LQG-controller

		Subsystem	
		LQR	**LQE**
Parameter	**Q**	diag(0,100,0,0)	diag(0,100,0,100)
	R	0.001	0.001

These settings lead to the following gains.

$$\mathbf{K}_{LQR} = \begin{bmatrix} 5.7 & 267 & -0.37 & 34 \end{bmatrix} \qquad \mathbf{L}_{LQE} = \begin{bmatrix} 2.0e^{-6} & 2.8e^{-8} \\ 1.0 & 3.2e^{-15} \\ 7.4e^{-6} & -5.2e^{-6} \\ 3.7e^{-15} & 1.0 \end{bmatrix}$$

The results of the simulation are shown in Figure 32.

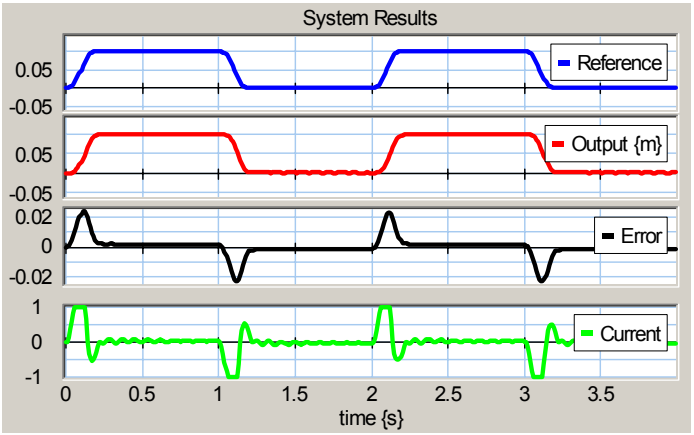


Figure 32: Results of LQG controlled system

This simulation result shows that the error approaches zero and that the system hardly vibrates at the eigenfrequency of the flexible frame. The states are estimated by the Kalman filter (see Figure 33). In this case, the state estimator has an error (0.14mm at loadsensorworld measurement) that is comparable to the resolution of the linear strip that the measurement is based on (250LPI → 0.1mm resolution), which makes this state estimation performing well.

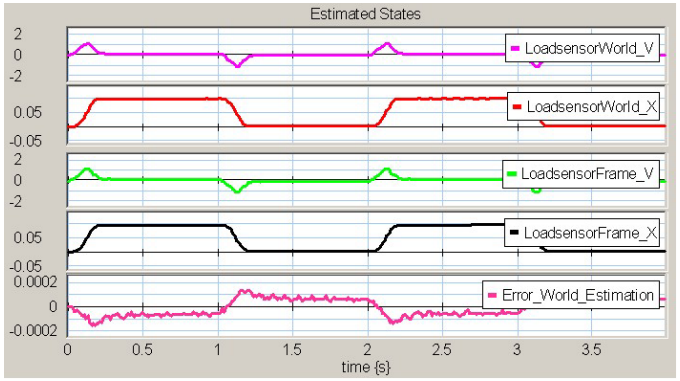


Figure 33: Results of Kalman filter state estimation

Most problems arise when the model that the Kalman filter is based on, differs from the process it wants to estimate the states of. Non-linearities, structural differences and large parameter variations disturb the state estimation. The only non-linearity in the currently used process is the coulomb friction in the element 'Damper'. Figure 34 shows the state estimation results when the coulomb friction is set to 0.5, 3 and 5N.

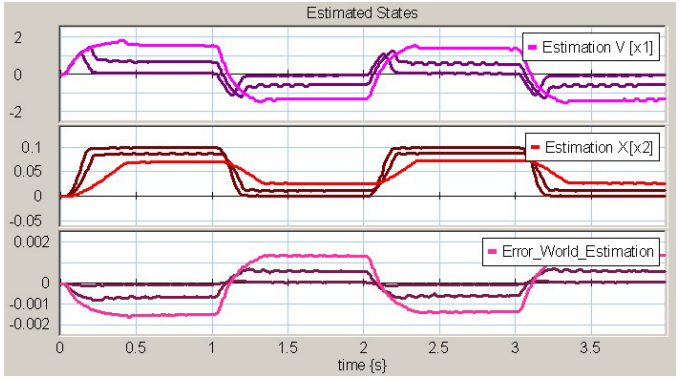


Figure 34: Results with applied non-linearities in process

Two observations can be made

1. The system will not reach its reference value anymore (0.1 and 0) due to extra coulomb friction.
2. The Kalman filter estimates the velocity wrong: even though the position is estimated constant, the estimated velocity becomes non-zero, which is physically impossible. This may also result in extra control errors.

In the process model, only coulomb friction is taken into account – stiction is not modeled. Tests in practice should show whether the coulomb friction and stiction are small enough for a LQG control algorithm to perform well or that modifications to the controller are needed.

The LQG and PID control systems will be tested on the CLP-setup, to compare their performances in practice. The stiction and real coulomb friction will then be taken into account.

3.5 Comparison PID and LQG control algorithms

3.5.1 Introduction

This section describes the comparison between a PID and an LQG control algorithm in simulation. Both controllers should be tuned with the same criterion. The algorithms will be compared at the following points

1. frame vibration $\left(X_{slider\_world} - X_{slider\_frame}\right)$
2. energy consumption of the system $\left(\int u^2 dt\right)$
3. position error $\left(X_{ref} - X_{slider\_world}\right)$

The function chosen as criterion is the same as the criterion in the Linear Quadratic Regulator

$$J = \int \left(\boldsymbol{e}^T \mathbf{Q}\, \boldsymbol{e} + u^T \mathbf{R} u\right) dt$$

In which the elements are defined as follows

Q importance of error reduction by controller
R importance of controller output signal
u output signal of controller
e error between reference and states, defined as: [0; ref; 0; ref] – *statevector*, in order to steer both second state (slider w.r.t. world) and fourth state (slider w.r.t. frame) to the reference

Parameter Q is set to diag[0,100,0,0] in order to control only the position of the slider with respect to the fixed world (second state of the system). R should be tuned such that the controller does not saturate. If the results of the optimization procedures (J) for both the PID and LQG system are equal, the two systems are assumed to control according to the same criterion, which allows proper comparison of the measurement data.

Optimization can be done by means of solving the Ricatti equation and by means of the multiple run facility of 20-sim. Both approaches appeared to give the same controller gain at the LQR system of Figure 27, at a reference of zero and a certain offset of the slider. To avoid the need for an analytical solution to the optimization of the PID-controlled system, the multiple run toolbox was used. This toolbox adjusts certain parameters such that a criterion is minimized or maximized. The criterion is chosen as *J* defined above, and the adjusted parameters are the controller parameters (LQG: K1, K2, K3, K4; PID: K_p, K_i, K_d). At first, the optimization toolbox roughly defines an *area* where the global minimum is expected to be found in a predefined number of steps. After this procedure, a (probably local) minimum will be found which is a solution to optimizing the criterion.

3.5.2 Optimization of LQG control parameters

Figure 35 depicts the mathematical model for determining the optimal 4th order state feedback.

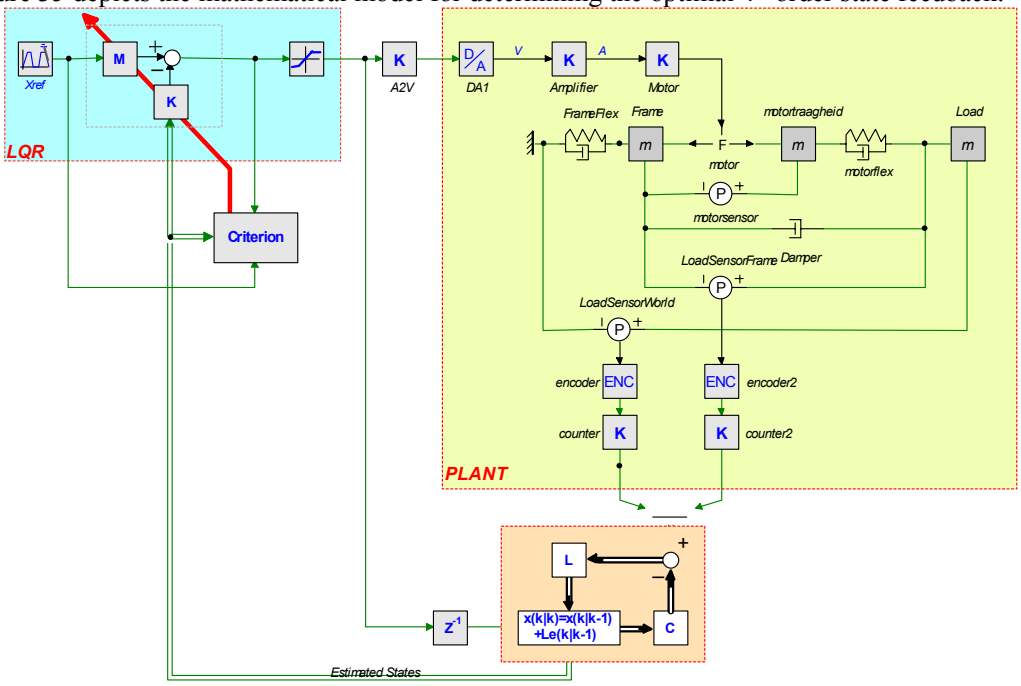


Figure 35: Tuning of LQR controlled system

Only the controller itself (LQR) will be tuned in the LQG algorithm – the estimator (LQE) is, just as in previous sections, set as a state variable filter. Controller parameter R is tuned by an iterative process aimed at maximizing the controllers responsiveness without saturating the actuator (+/- 1A). This simulation led to the following parameters

$R = 0.5$
$K_1 = 5.5$
$K_2 = 197$
$K_3 = 9.9$
$K_4 = 149$

The results of these feedback parameters in a simulation of the LQR controller are depicted in Figure 36.

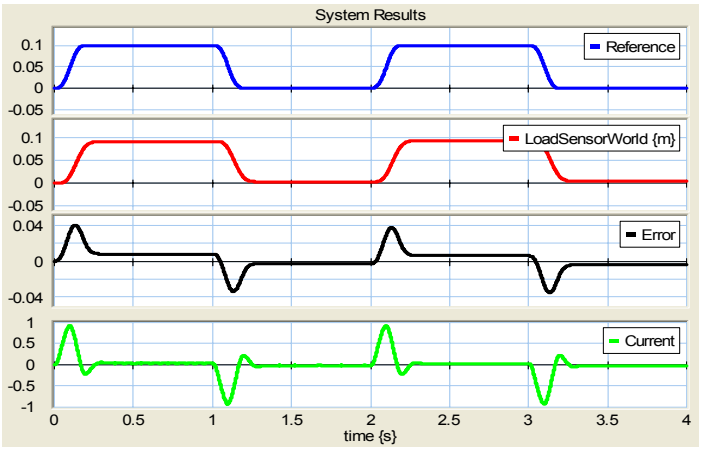


Figure 36: Results of first LQR tuning

The maximum current is 0.93A, which is within the boundaries of the actuator. A static error still remains in the system of 0.005m, which is due to the simulated nonlinear coulomb friction in the model. A higher proportional gain or an integral term will decrease this error. A higher proportional gain however is not desired since this will affect the maximum current of the system. Addition of an integral term is an interesting aspect which will be treated later in this report (section 3.5.4).
The minimization of the criterion led to a value for *J* of 0.17.

3.5.3 Optimization of PID control parameters

The PID control algorithm can be tuned in a similar way. The criterion is identical and the maximum current output will be determined by the parameter R in the optimization process. See Figure 37 for the mathematical model by which the optimization has been performed.

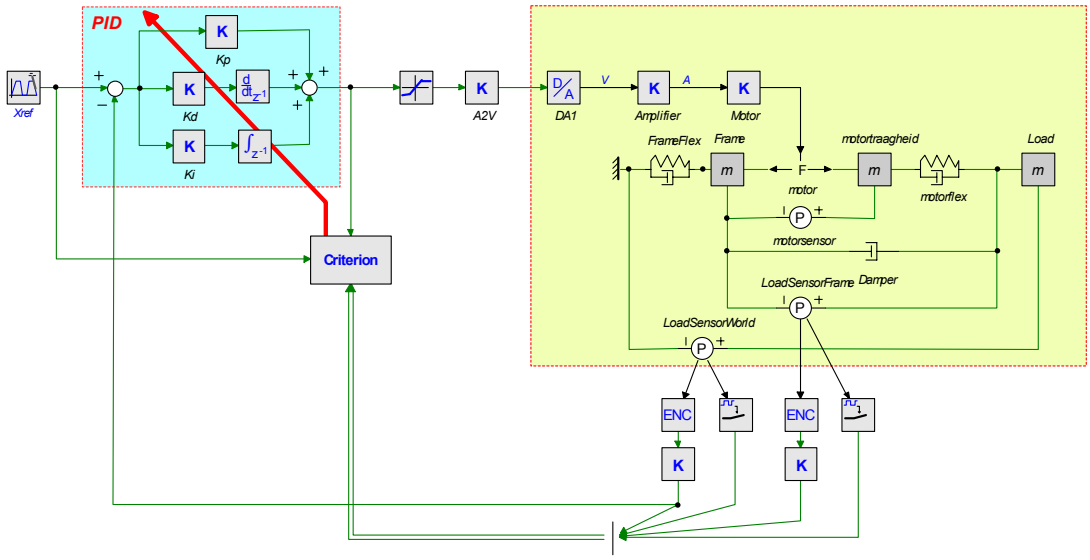


Figure 37: Tuning of PID controlled system

Iteration of the optimization routine, led to a minimum value of J = 0.37 (at R = 1), corresponding to the following controller parameters

K_p = 15.7
K_i = 42
K_d = 1.6

The corresponding response of the system is shown in Figure 38.

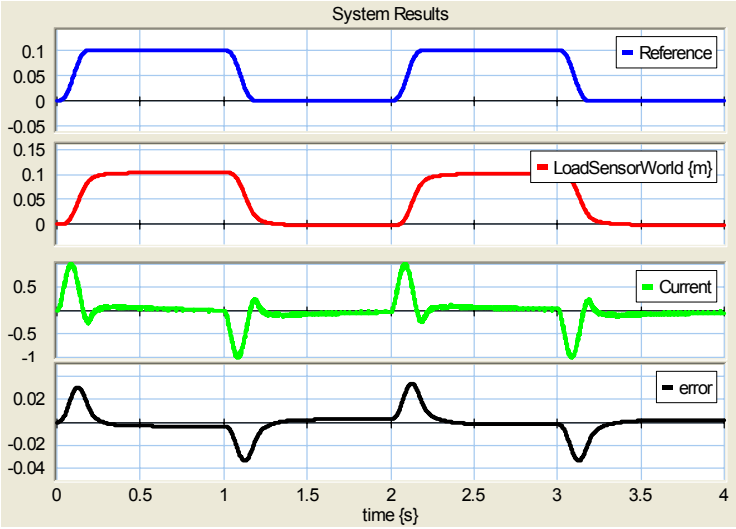


Figure 38: Tuning results of PID controlled system

Tuning the PID controlled system leads to larger values of J, therefore it is harder to build a system corresponding to a certain low result of the minimization criterion $J = \int \left(e^T Q e + u R u \right) dt$ with a PID controlled system than with an LQR controlled system. This can be explained by the advantages of full state feedback that is performed with the LQR algorithm.

3.5.4 Comparison of LQG and PID

The procedure for comparing two control systems is to make the optimization parameters Q and R equal at both systems to ensure the systems are compared while being optimized with the same criterion. Q and R must be set such that neither of the two systems exceeds the maximum current (1A). Intuitively, the PID control algorithm is expected to consume most power, so the R parameter of this system that results in maximum 1A is also set for the LQR control algorithm. Concluding, the settings for the optimization procedure of both systems are

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix} \quad \mathrm{R} = 1$$

The results were already presented in the previous section, but will be repeated here for completeness (see also Figure 38)

J = 0.37
K_p = 15.7
K_i = 42
K_d = 1.6

In which case, as stated before, the maximum current was 1A.

Applying the same optimization settings to the LQG control algorithm, the following results are obtained

J = 0.29
K_1 = 3.74
K_2 = 74
K_3 = 8.2
K_4 = 69.5

The maximum current in this situation is bounded to 0.6A. These parameter values result in a response of the linear system as depicted in Figure 39. NB: tuning is performed on the linear model. Actual comparison of PID and LQG control algorithms take place with the nonlinear model.

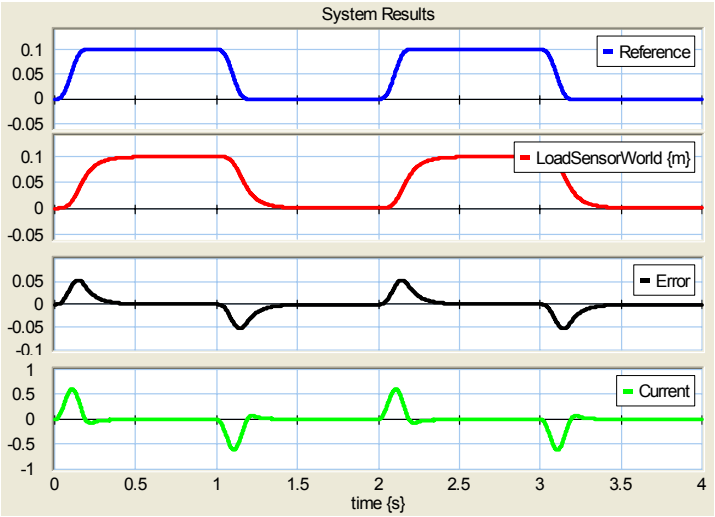


Figure 39: Results of second tuning of LQG controlled system

Both systems have now been optimized with the same criterion such that their performances can be compared. The model is Figure 40 is used for comparison.

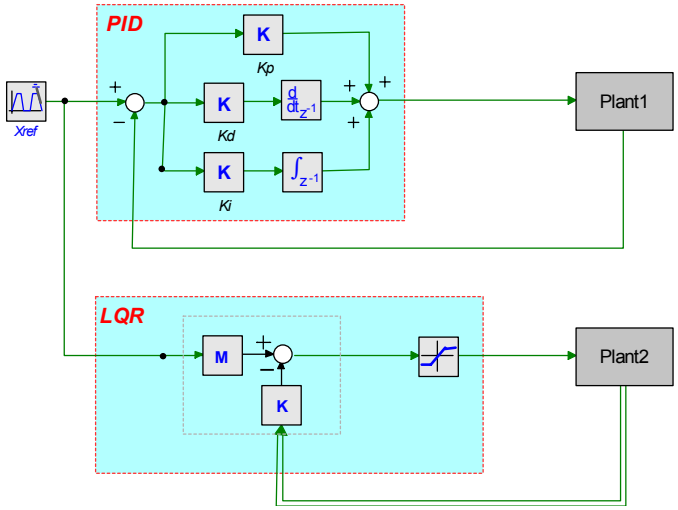


Figure 40: Comparing the tuned PID and LQG controlled systems

The following conclusions can be drawn based on the results

1. Maximum frame movement of PID controlled system is 80% more than the frame movement of the LQG controlled system
2. The PID controlled system consumes about twice the power that the LQG controlled system consumes
3. The LQG controlled system performs 3.5x less in resolving a static error

See the following figures for the results of the simulation on the nonlinear plant (Figure 41).

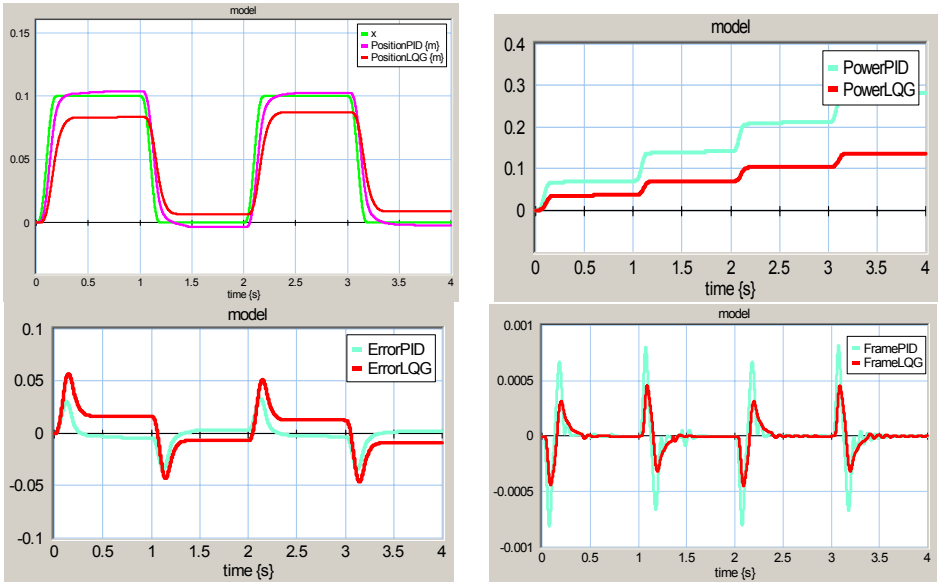


Figure 41: Results of comparison between tuned PID and LQG controlled systems

The remaining static error in the LQG system is expected to be decreased by adding an integral term to the LQG control algorithm.
The disturbances that is responsible for the static error in the LQG-controlled system is the element 'damper', that represents a (non linear) coulomb friction. In a real situation, also stiction will be present and most probably more disturbances that have not been taken into account in the current model. These disturbances can be modeled as an extra source at the input of the plant, as depicted in Figure 42.

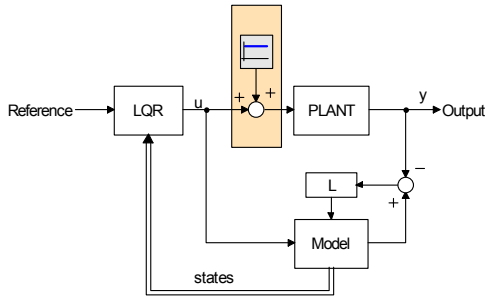


Figure 42: Disturbances modeled as extra input of plant

This extra input (just as various disturbances) causes differences between process and Kalman filter, by which the performance of the LQG algorithm is reduced. However, the error can be compensated for. Integration of the error between measurement (output of process) and estimation (output of reference model) can approach these disturbances. See Figure 43.

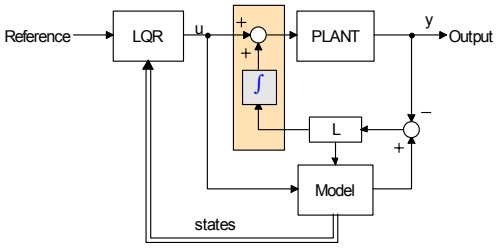


Figure 43: Addition of integrator to LQG

The gain of the integrator can be tuned manually, or it can be included in the solution of the ricatti equation. Because of time constraints, the integral gain was tuned manually. Adding the integrator to the designed LQG-control algorithm leads to the model of Figure 44.

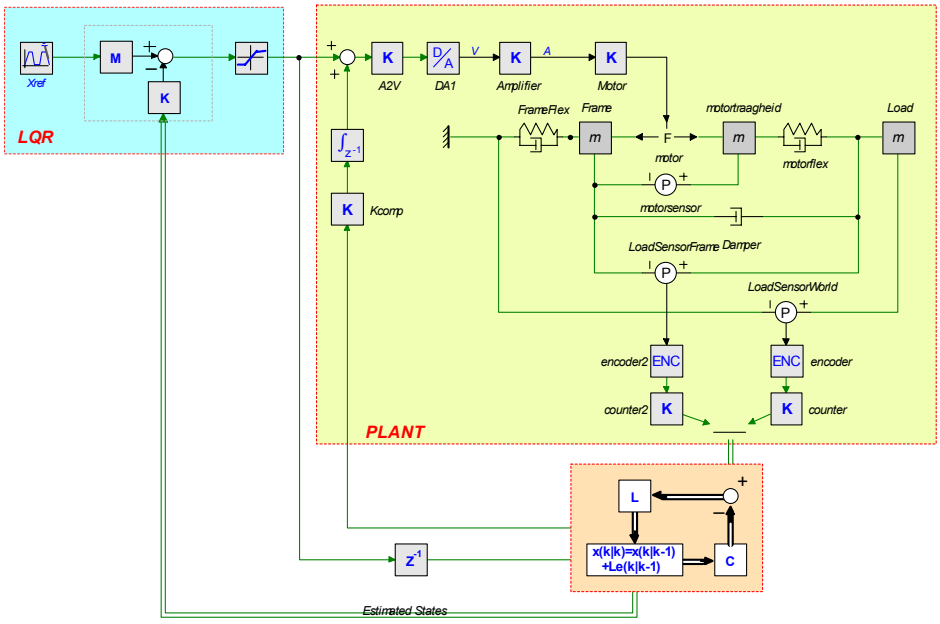


Figure 44: LQG I-addition in simulation

The error between measurement and estimation of the position of the slider with respect to the fixed world was taken as input of the compensator. An example of the results of this added integral action to the LQG control algorithm is depicted in Figure 45, where K_{comp} = *-5000* was used. The graph depicts the results without integral action (dark lines) and the results after adding the integral action (light lines).

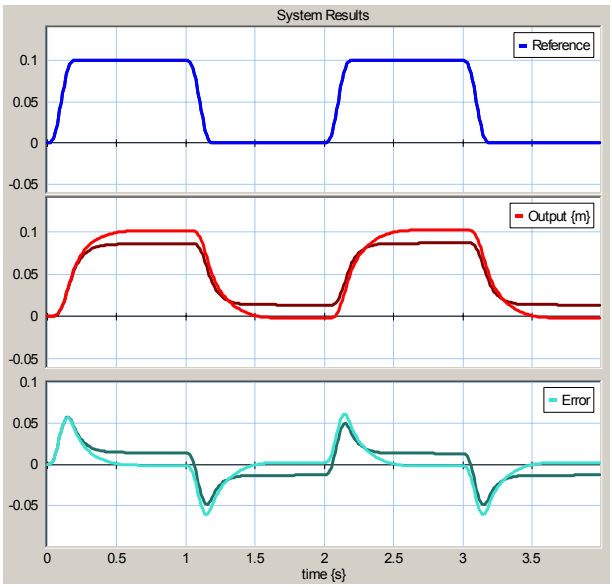


Figure 45: Results of added integral to LQG control algorithm

The simulation shows that considerable performance increase can be achieved by adding the integrating action to the LQG control algorithm. Later on in this report, the same system will be used in practice.

4 Experiments on CLP setup

This chapter describes tests performed on the demonstration setup of CLP. The aim is to check whether the CLP-setup will be a good starting-point for the new mechatronic demonstrator. This will be decided on the basis of ability to show performance differences between PID and LQG control algorithms in practice. First, the procedure to run a model in practice will be treated, after which several client tools are presented. The last section of this chapter elaborates on the actual tests on the system.

4.1 Procedure

To test a control algorithm in practice on the CLP-setup, the following procedure needs to be undertaken

1. Build a control algorithm in 20-sim
2. Add input/output elements for the sensors and actuator
    a. These elements describe the hardware in the system and provide the interface between software and hardware
3. Start the simulator
    a. Adjust system parameters
    b. Set desired discrete sample frequency
    c. Set certain parameters and variables as ‘Favorite’
        i. These parameters and variables can be edited or viewed on the client machine
4. Start the ‘C-code generator’ (part of the ‘Real-time Toolbox’)
5. Choose the ‘Prosys RT-Linux Environment’ as target
6. C-code will now be generated and compiled, according to the template files
    a. These files describe the conversion of 20-sim code to compileable c-code
    b. They also contain the way the specific hardware should be controlled (in the case of the CLP-setup: Kolter counter card and Meilhaus I/O card)
7. The compiled control algorithm can be uploaded and started by the client software.

4.2 Client Software

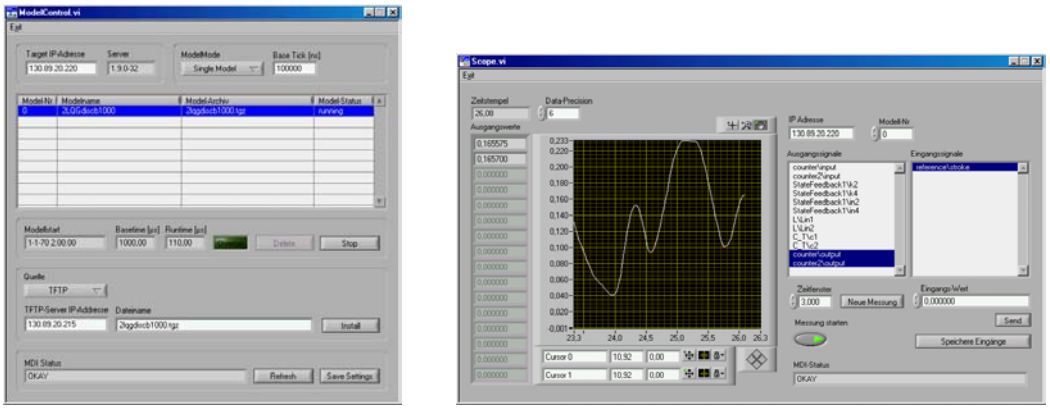
Several client tools are available for the communication with the real-time server: a tool made by Controllab Products (which is not yet commercially available), a webclient and client tools provided by CosaTeq. Each tool has its own disadvantages and advantages - the tools will be described in the following sections. Installation of a TFTP-server at the client is required to send the compiled models to the RT-Linux machine (see also appendix IV). A TFTP server (“Trivial File Transfer Protocol”) is basically a simplified version of an FTP server, with fewer demands on the hardware (TCPIPguide, 2005).

Cosateq client tools

Three separate tools are available

- ‘Modelcontrol’	Provides uploading, deleting and starting/stopping models (Figure 46a)
- ‘Scope’	Variables set as ‘Favorites’ can be monitored by numbers and in a graph. Furthermore, this tool enables changing ‘favorite’ parameters (Figure 46b)
- Measurement’	Logs variables to a file

A major disadvantage of these tools is that the functionality is scattered over three programs.



(a) (b)

Figure 46: Screenshots of Cosateq client tools

Cosateq webclient

The CosaTeq software includes a web server that allows uploading, deleting and starting/stopping models. Furthermore, all parameters and variables set as 'favorite' can be changed and monitored. This interface is slow, but it does not require installation of software (other than a TFTP-server).

CLP-tools

Controllab Products designed a client tool specifically for communication with the real-time CosaTeq server. This tool combines all desired functionality (model control, parameter/variables control and logging) in one program. Major advantage of this system is the built in animator, which is linked to the variables of the system – see Figure 47 for a screenshot.

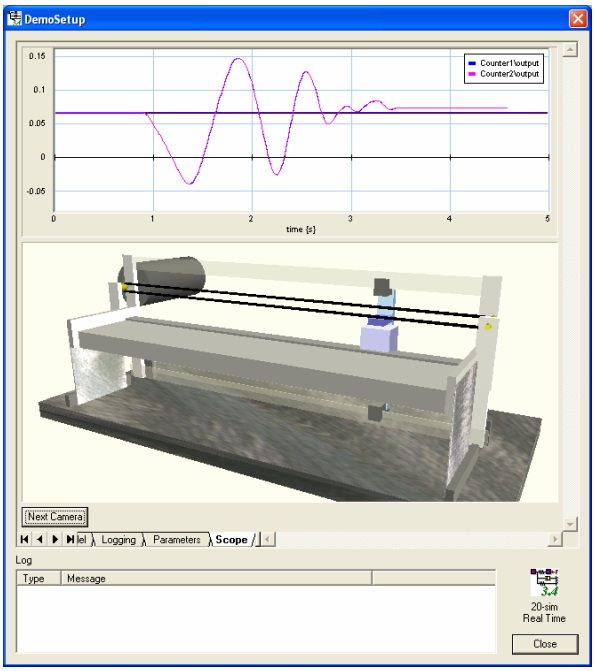


Figure 47: Screenshot of CLP client tool "DemoSetup.exe"

4.3 LQG versus PID

Before a decision was made to actually build the mechatronic demonstrator, it had to be shown that educational goals can be met by the demonstrator

1. Being able to run several control systems on the device
2. Being able to show functional differences between control systems

This section describes the testing of PID- (Figure 48) and LQG- (Figure 49) control algorithms on the demonstration setup of Controllab Products. The PID and LQG controllers are copied from the theoretical sections of previous chapters. Changes made to these systems concern the simulated plant being replaced by I/O hardware blocks. These blocks represent the hardware in the real plant and take care of the interface between the embedded controller and the plant. Furthermore, the controller parameters are changed – the model used in the theory has not been verified with the actual system. Friction elements are most likely to differ between theory and practice, especially since stiction is not taken into account in the mathematical model. The control parameters are therefore expected to be changed in order to obtain the same result in practice as in theory.

The PID control algorithm for testing in practice is depicted in Figure 48.

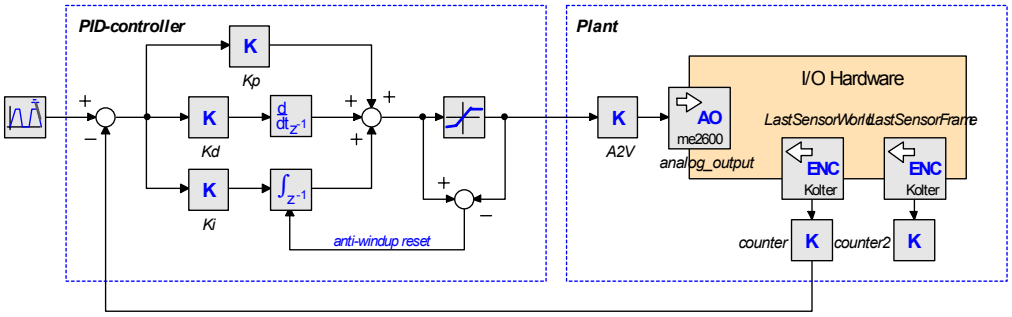


Figure 48: PID controller for testing in practice

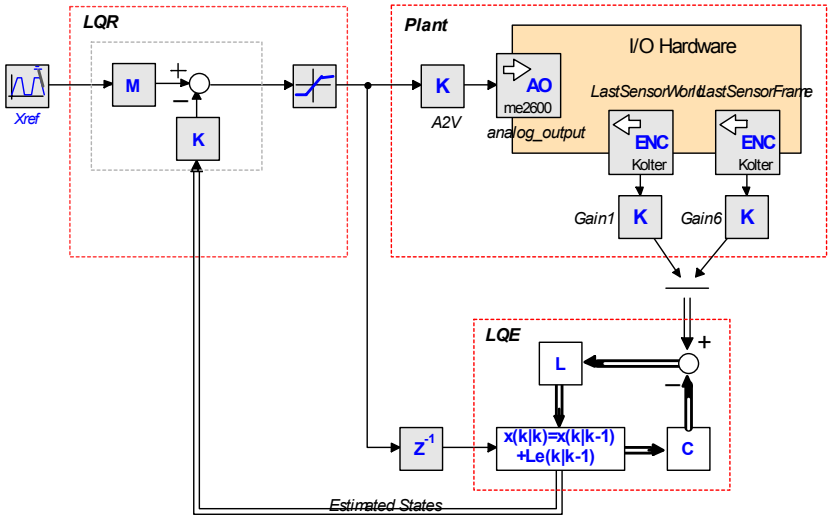


Figure 49: LQG controller for testing in practice

The controller parameters have not been tuned according to the mathematical criteria, such as presented in the previous chapter since these were investigated in a later stage of the project. The criteria for determining the control parameters for these first tests were set as follows

- tracking error of maximum 10% of setpoint
- no clipping of the controller output signal at the decelerating phase of the slider movement

The results of the control systems are compared, based on the following criteria

- Error between reference and measurement of the position of the slider with respect to the fixed world (both static and dynamic) $\left(X_{ref} - X_{slider\_world}\right)$
- Vibration of the frame $\left(X_{slider\_world} - X_{slider\_frame}\right)$
- Power consumption $\left(\int u^2 dt\right)$

Since the mathematical model of the plant does not anticipate to non-linear coulomb friction and stiction, the Kalman filter parameters are set to mainly rely on the measurements – it is set as a state-variable filter. The LQR controller is tuned to maximum steering signal without causing oscillations in the system (standing wave in rubber belt). After obtaining the measurement data of the LQG controlled system, the PID control parameters were set roughly according to the same criteria. The following controller parameters were used

PID	**LQG**
$K_p = 150$	$R_{Kalman} = 1e^{-4}$
$K_d = 5$	$Q_{Kalman}(2,2),(4,4) = 100$
$K_i = 10$	$R_{LQR} = 1e^{-4}$
	$Q_{LQR}(2,2) = 100$

The LQG controller parameters led to the following gains in controller and estimator

$$\mathbf{K}_{LQR} = \begin{bmatrix} 10.9 & 894 & -0.6 & 18.4 \end{bmatrix} \quad \mathbf{L}_{LQE} = \begin{bmatrix} 2.0e^{-6} & 2.8e^{-8} \\ 1.0 & 3.2e^{-15} \\ 7.4e^{-6} & -5.2e^{-6} \\ 3.7e^{-15} & 1.0 \end{bmatrix}$$

Measurements were performed during the tests with both control systems. The results are depicted in Figure 50.

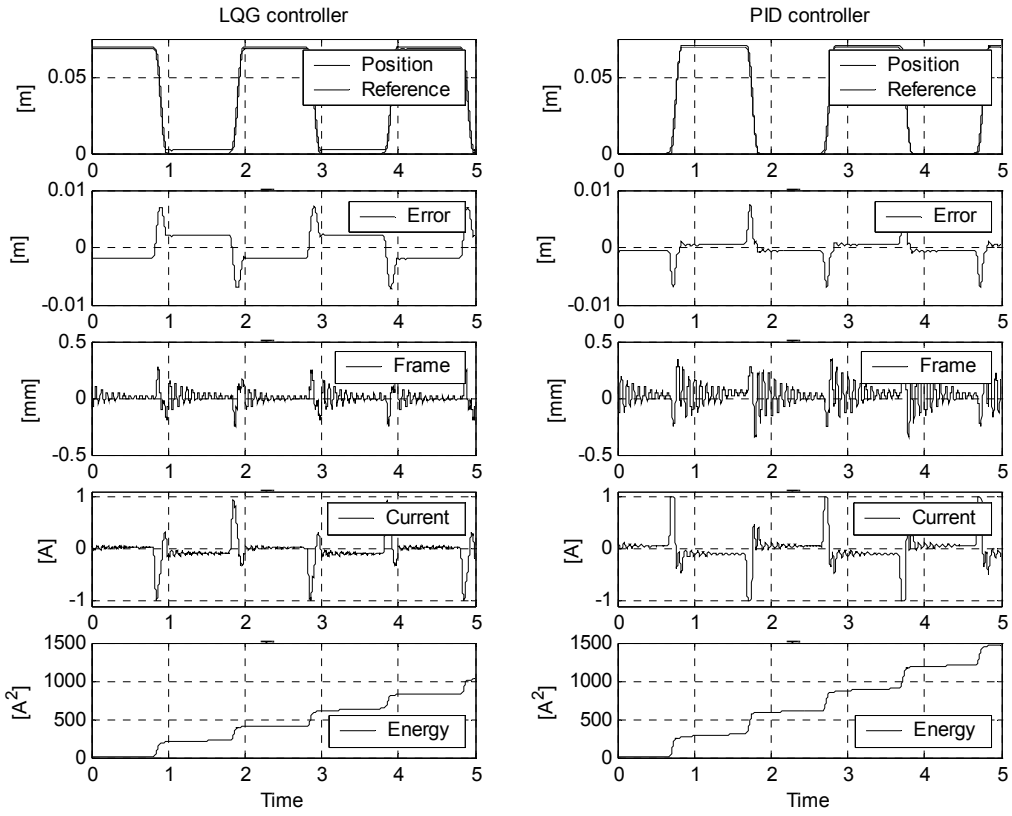


Figure 50: Results of first experiment on CLP-setup

As can be seen, the resulting position error of the two control algorithms is comparable, except for a static error that remains in the LQG-controlled system, due to the absence of an integrator in the controller. Furthermore, the LQG-controlled system results in less vibration of the frame. This can be explained by the fact that the fourth order LQG-system takes the resonant behavior of the structure into account – it can compensate the frame vibration, whereas the (second order) PID-controlled system will not. Last comparison of the two control systems is that the PID system consumes about 50% more energy, whereas this is not rewarded by better performance.
The static error in the LQG controlled system is assumed to be caused by differences in friction between process and mathematical model. The implemented LQG control algorithm is linear, which enables only a viscous friction part to be modeled. Both stiction and coulomb friction are not taken into account. Errors might be made in the estimation of states which results in velocity estimations unequal to zero, whereas in reality the slider is not moving. The LQR controller does not give any output to compensate for a position error, since it gets data that the slider is still moving in the right direction.
Tests on the amount of stiction indicated that even 0.35A of current (limitation: +/-1A) is not enough to overcome the stiction (force on slider: 2N). Further experiments on the amount of stiction, coulomb and viscous friction are advised. Attempts have been made to minimize the total amount of friction on the CLP-setup by performing maintenance

1. Putting oil on the slider
2. Loosening the geared rubber belt

Tests showed that the maintenance has led to improved performance of the LQG-control algorithm. Simulation showed that this can be caused by reduced friction. See Figure 51 for the comparison of LQG results before and after maintenance.

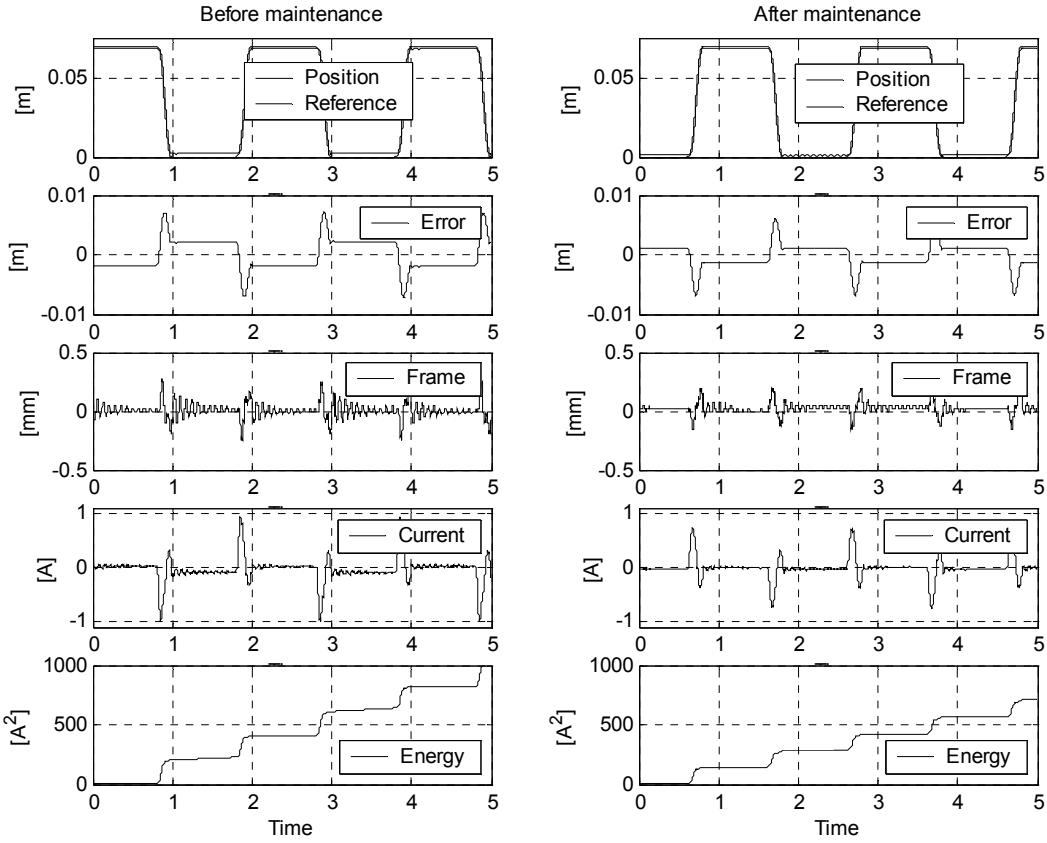


Figure 51: Effects of maintenance to performance of LQG-controller

The measurements show that considerable improvements have been made. The static error is still present, but it is decreased by 28%. The maximum current is decreased by 22%, and the total energy consumption is reduced by 14%. Considering the frame movement, it can be seen that the maximum movement is decreased (20%), but more importantly, the damping of the vibration is increased. On the whole, a significant increase in performance of the LQG-controlled system has been achieved by performing some maintenance.
Due to reduced friction in the demonstrator, the controller parameters can be tweaked even more. By decreasing the R_{LQR} parameter by a factor 10 (into 10^{-5} in stead of 10^{-4}), the control signal becomes more aggressive due to for instance a higher proportional gain. This was not possible before the maintenance since vibrations in the geared belt occurred (standing wave). See Figure 52 for the performance increase of the LQG controlled system at the start, after maintenance and after decreasing R_{LQR}.

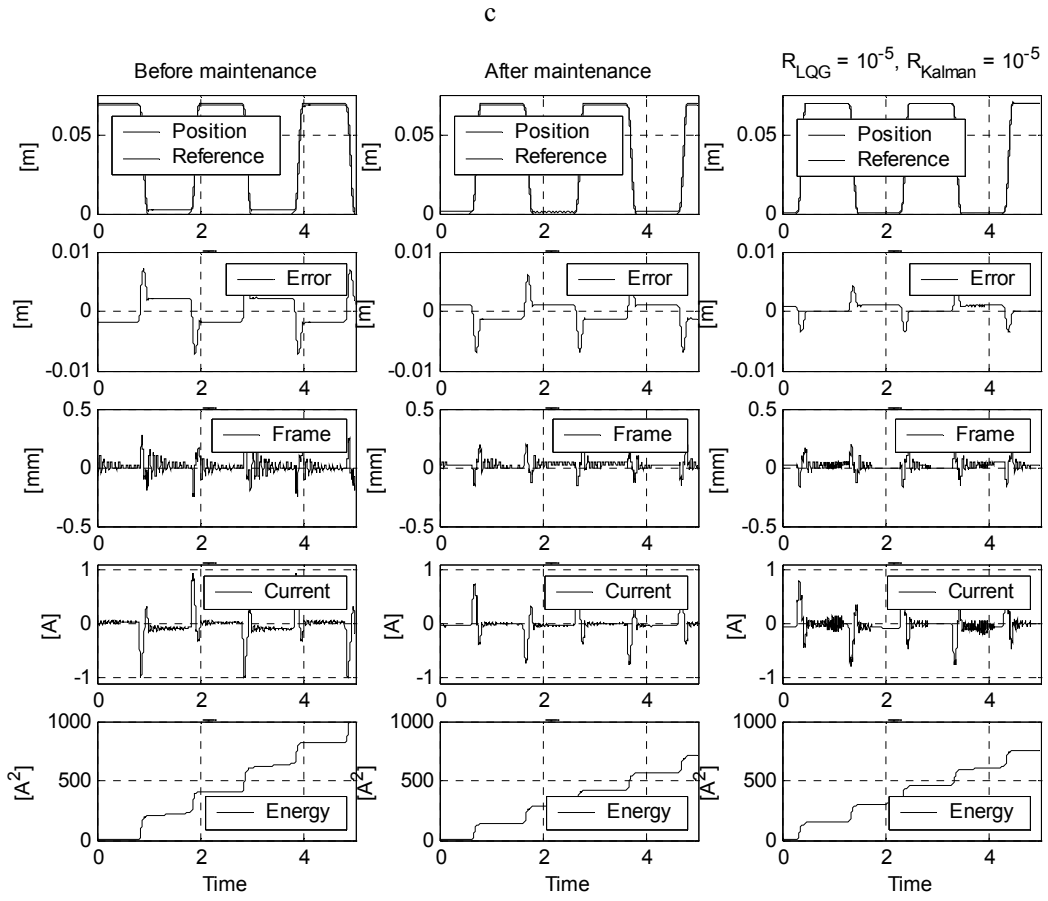


Figure 52: Increased performance of LQG controlled system

Clearly, the changes made to the demonstrator and the controller parameters result in better performance in (static) error, frame vibration reduction and power usage. Comparing the LQG controlled system with the PID system results in the graphs depicted in Figure 52.

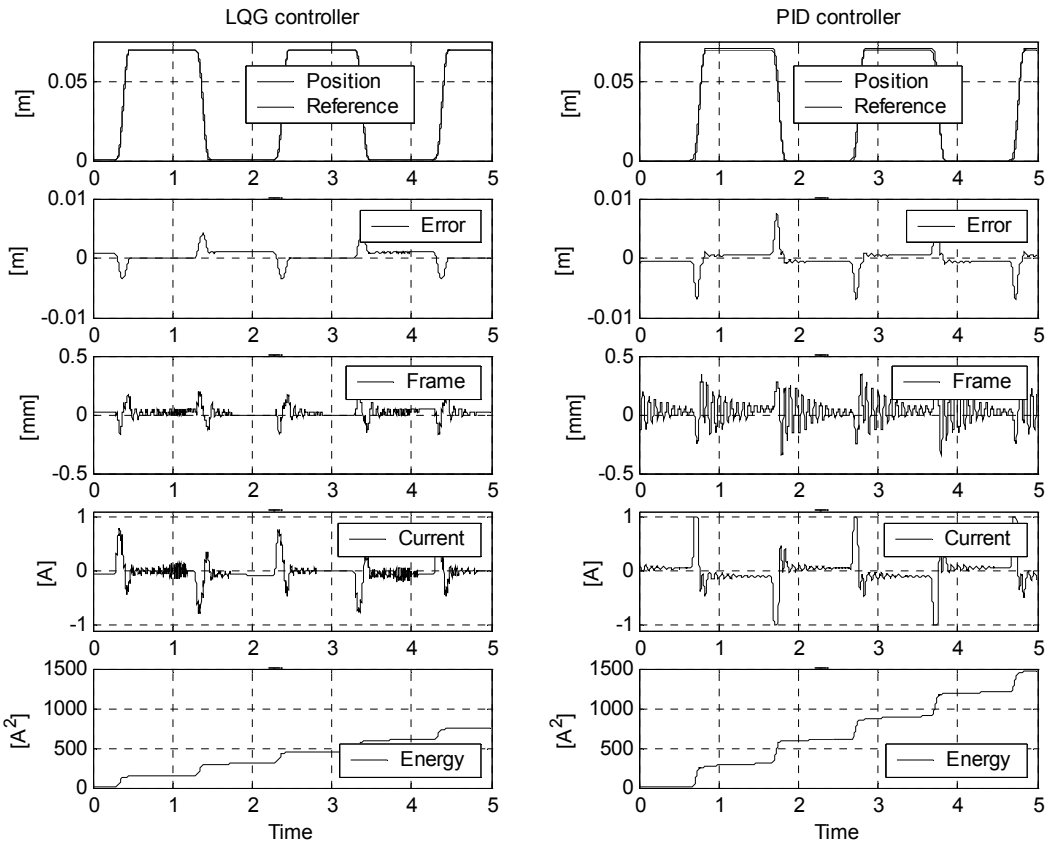


Figure 53: Comparison of improved LQG- with PID controlled system

The conclusions about this final experiment are that the LQG-system outperforms the PID controlled system at most criteria defined at the beginning of this chapter, at half of the energy consumption. Especially the suppression of frame vibration is interesting to be shown in practice. Due to the higher-order control algorithm, the LQG system performs differently than the PID system. The PID system remains better in solving the static error. It must be stated that the controllers have not been both tuned to a certain mathematical criterion, which makes precise comparison in this stage of the project difficult.

It is recommended to investigate 'add-on' systems, such as inner control loops, that decrease the static error in the LQG system by addition of an integrating term. Furthermore, a nonlinear Kalman filter could be tested on the system to cope better with nonlinear friction. Finally, tests with (learning) feed forward systems are advised, specifically to overcome stiction.

With the tests described in this chapter, two types of control systems were tested on the demonstrator of CLP. Differences between the control algorithms can be shown in practice, which makes it interesting to build such a demonstration setup.

5 Design and realization

The design of the new mechatronic demonstrator will be based on the demonstration setup made by Controllab Products B.V. This chapter first gives a description of both hardware and software of the existing CLP-setup, after which design choices for the new mechatronic demonstrator are presented. Secondly, the system parts of the new demonstrator are treated. A few improvements and possible expansions of the new demonstrator, that have been formulated in the process of building and testing are presented in the last section of this chapter.

5.1 Current configuration

This setup was presented in sections 2.2.3 and 2.4 and will be treated in more detail in this section. Basically the system consists of two personal computers and the demonstrator itself, as depicted in Figure 54.

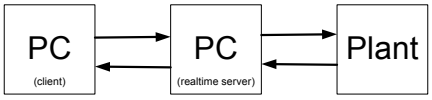


Figure 54: Schematic description of demonstration setup

5.1.1 Software

The PC connected to the plant is provided with a real-time Linux operating system that runs the control software. The real-time module that is added to the operating system is a software package called Prosys-RT made by Cosateq (Cosateq, 2005). It is a prototyping environment to connect simulation models to physical systems and execute them in real time. It consists of a customized real-time Linux kernel, with which model sample frequencies up to 100KHz can be achieved (Cosateq, 2005). Furthermore, the Prosys-RT package consists of a MDI-server (Model Data Interface), which forms the interface between kernel-space and user-space. The server enables starting/stopping of models, changing parameters and viewing variables real-time on a client PC. The server can be accessed via a standard internet browser, Cosateq client tools and a tool built by Controllab Products. The latter tool provides a real-time 3D-representation of the process that is controlled (see Figure 47). A link with 20-sim has been made to generate code for the Cosateq environment. Templates have to be built for the specific I/O interface that is connected to the hardware. The 20-sim code generator generates C-code (according to the templates), which is compiled by a cygwin Linux C compiler. The Cosateq software communicates with the hardware through a Comedi interface (‘control and measurement device interface’). Comedi is a collection of hardware drivers, a common kernel interface and a support library. Comedi standardizes communication with general I/O at a higher level than driver level, to uniform the commands in user-space. Comedi device drivers are generally available on the internet for the hardware in the CLP-setup.
Prosys-RT (version 2.1) requires a SuSE LINUX operating system.

5.1.2 Hardware

As depicted in Figure 10, the hardware of the CLP setup consists of a personal computer, an external amplifier and a power adapter. The PC is supplied with I/O interfaces, both digital and analog

1. Digital Counter card : Kolter PCI ‘Counter-2’
    a. 3x 24 bit encoder input channel
2. Analog and Digital I/O card : Meilhaus ME-2600 PCI
    a. 16x 12bit A/D channel
    b. 4x 12bit D/A channel
    c. 32x digital I/O channel

The digital counter card is used to measure the position of three encoders (motorsensor, loadsensorworld and one of the sensors that measure the position of the slider with respect to the frame). The Meilhaus card only supplies an analog output for the motor amplifier.

More information about the CLP-setup can be found in (Kleijn, 2003).

5.1.3 Experiences

Extensive testing and working with the CLP-setup led to a good impression of the system in terms of robustness and capabilities. Some bugs had to be resolved in the code generation of 20-sim. On the whole however, experiences with the setup were satisfying. The system runs the tested control systems real-time and without glitches. Most interesting part of the Prosys-RT software is the ability to adjust system parameters and to view variables real-time. The coupling with 20-sim works fine, except for the fact that the user interface at the windows-side of the system is either laborious (CosaTeq tools) or not optimal in functionality (CLP tool). Recommendations will be presented later in this report. Main disadvantage of the CLP setup is the size and amount of all parts – the setup in its current form is not really suitable as a portable demonstration setup. Besides this, only three sensors can be measured at the same time, whereas four sensors are mounted.

5.2 New configuration

5.2.1 Goals

The following goals have been determined for the to be built demonstration setup

1. Mechatronic part is based on the CLP-setup
2. System should be 'plug-and-play', with a minimal amount of cables and external components
3. Mostly use of off-the-shelve components
4. Robust and reliable system
5. 'Open' construction for demonstration purposes
6. Portable system

5.2.2 Hardware

The largest part in the current CLP-setup is the personal computer that runs the operating- and control system. This system will be replaced by an embedded PC. Within the CE-group there is a lot of experience with PC104 CPU boards (Groothuis, 2004), which is a decisive reason of acquiring one of those for the new setup. This type of board comes in a 600MHz (fanless) and 800&1000MHz (with fan) model. A fanless system is preferred (less noise, less dust attraction, less danger) so tests were performed to check whether a 600MHz system would be fast enough to run the most complex LQG control system sofar. It appeared that with a 600MHz *Via Eden* processor, the LQG controller as depicted in Figure 49, runs with a loop time of 72µs. This measurement assumes 10µs time duration for copying variables – a number which is assumed to be comparable in the Prosys-RT environment. I/O operations were *not* taken into account in the tests. Based on these results, the LQG system is expected to run without problems at maximum 10KHz on a Via Eden 600MHz CPU. In tests sofar a samplefrequency of only 1KHz was used.
Since the processing unit of the mechatronic demonstrator will be PC104-based, so will the I/O interface. Research on the internet for manufacturers of these type of boards with analog and digital I/O including encoder interfaces showed that the multifunction I/O boards made by Sensoray (Model 526) does include all functionality required

- 4x 24 bit quadrature encoder
- 4x 16 bit analog output
- 8x 16 bit analog input
- 8x digital I/O
- Single 5 volts power supply
- PC104 bus

Another I/O option is the PC 104 'Anything I/O' board, based on a Xilinx Spartan FPGA chip (Mesa, 2000 via Groothuis, 2004). It can be programmed to support PWM-outputs and quadrature encoder inputs. According to the goal to use as much standard of the shelve components led to the decision to choose the Sensoray board as I/O interface for the mechatronic demonstrator. Furthermore, this board gives versatility to the existing embedded systems equipment.

Similarly to the CLP-setup, a standard motor amplifier (Analog Devices ADS 50/5) was chosen to provide current to the motors. This device is set up as a current source (max 1A) modulated by an input voltage, delivered by the I/O interface.
The connection with a client PC that the user will operate will be established through a router. The advantage of this above a direct connection is that standard network settings (DHCP) can remain unchanged. Standard Ethernet cables can be used (in stead of crosslink cables) and a router offers functionality for internet access for both demonstrator embedded PC as well asclient PC.
Finally, two power supplies provide power to the system, one for the computer boards and one for the motor. Data (operating system, Prosys-RT, models) is stored at a hard disk drive. In the future, the hard disk can be replaced by a Compact Flash Card.

Figure 55 shows an overview of the hardware in the mechatronic demonstrator.

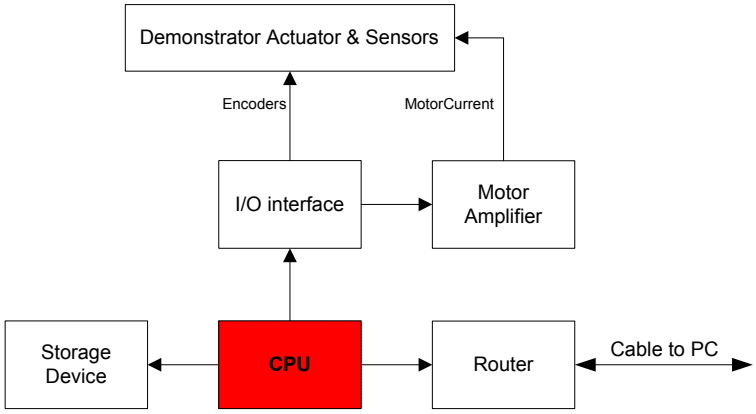


Figure 55: Hardware in Mechatronic Demonstrator

For normal demonstration purposes, only the connection of a mains- and Ethernet cable are sufficient, without the need of external components. More information about the parts in the mechatronic demonstrator can be found in appendix I. An extensive wiring diagram is presented in appendix III.

5.2.3 Software

The ProSys-RT software has briefly been described in section 5.1.1. Installations must be performed on both server (PC104) and client computer. Installation of SuSE and ProSys-RT on the embedded PC required numerous manual interventions due to the use of obsolete kernel versions (CosaTeq) since not all required modules were available. Appendix IV elaborates on installation of the ProSys-RT software on the server in more detail.
Besides running the server software on the PC104 CPU board, the hardware should be supported by Comedi device drivers, as explained in section 5.1.1). Comedi device drivers for the Sensoray 526 were not available so they were developed within the framework of this project. Documentation of the I/O board, available windows drivers and comedi manuals formed the base for this. Encoder inputs, DA-conversion and digital outputs are now supported by the comedi driver. AD-conversion should still be implemented, if desired in the future.
Disadvantages that became clear after installation and testing of the SuSE Linux distribution on the embedded PC appeared to be the long boot and shutdown procedures. A changeover to for instance a Debian Linux distribution, which is extensively used within the CE-group, is recommended. This system also makes it easy to shutdown the entire PC without losing data by mounting the storage device
Several programs need to be installed on the client PC to be able to get access to full functionality of the real-time environment. Appendix IV provides more information about software installation on the client PC.

5.2.4 Construction

The mechatronic demonstrator has been built by the TCO-department of the University of Twente. This is the same organization that built the CLP-setup in 2003. Major change to that system is the inclusion of all parts of the demonstrator into one device. The top plate was be expanded to the left for mounting the PC104 stack. Below the CLP-design, a casing will be placed, in which the other electronics (motor amplifier, power supplies) can be mounted (see Figure 56).

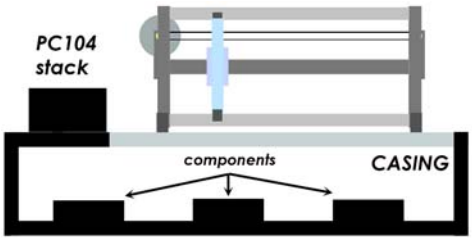


Figure 56: Sketch of demonstrator expansions

The frame has been built of lightweight aluminum tubes, of which the 45mm version is assumed to form a frame stiff enough to suppress vibrations. All around the frame, perspex plating is applied to make a clear view of the inside of the demonstrator possible. One of the side-panels of the casing (Figure 57) holds connectors for the peripheral devices, such as keyboard, monitor etc. For normal operation, these devices will not been needed. See appendix III for the wiring of the demonstrator.

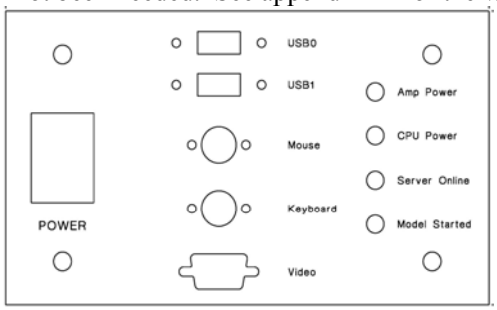


Figure 57: Side panel of mechatronic demonstrator

The design of the demonstrator itself is not changed. A number of changes were considered, but rejected:

1. Adding bars for possible extra weights to demonstrator to change its dynamical behavior. Parameter adaptation can be tested by assuming the process parameters unknown.
2. Remove bottom (frame-) linear strip, since its measurement is the same as the top linear strip (the slider does not rotate). In the future, a disturbed bottom strip might be used as non ideal measurement which might be interesting for educational purposes.
3. Use a different pulley to prevent (small) unexpected friction caused by the belt hitting the sides of the pulley. A pulley with a round surface would solve these problems, but these are not available. The induced friction is expected to be minimal compared to other elements of friction however.

A picture of the final mechatronic demonstrator is depicted in Figure 58.

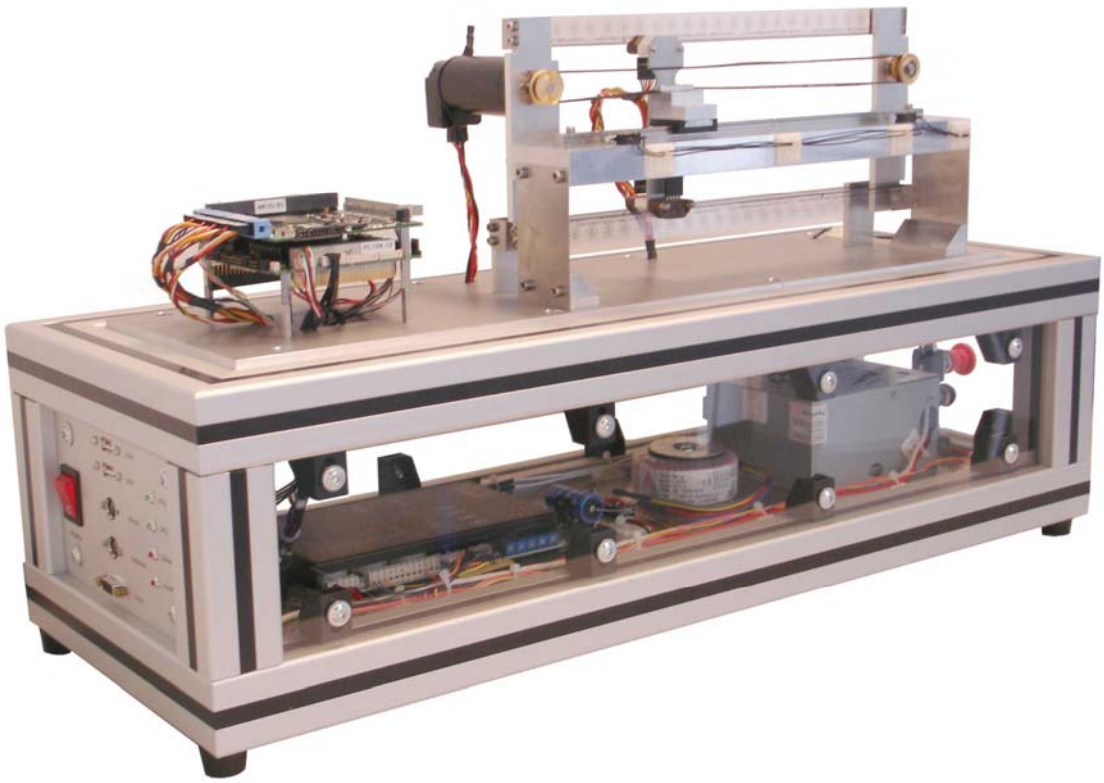


Figure 58: Picture of mechatronic demonstrator

5.2.5 Improvements and expansions

After finishing all parts of the demonstrator, some points for improvement have been observed.

1. Use of smaller aluminum tubes for the frame will lead to less weight. Currently, the demonstrator is quite heavy and can hardly be called 'portable'
2. Implementation of a hardware safety system, that actively decelerates the slider before it reaches the end-stop-switches. Especially when the setup is used at practicals or projects, this is strongly advised to reduce risk of damage.
3. In the future, the setup can be used to demonstrate parallel processing. PC 104 CPU boards with different tasks could be placed on the stack (e.g. one for measurements, one for control).
4. If the use of off-the-shelf components is no longer demanded.
    a. Build own current amplifier
    b. Use own real-time software in stead of CosaTeq software
    c. Possibly use only one power supply

6 Testing

This chapter describes tests that have been performed on the new mechatronic demonstrator. The procedure for getting mathematical models to run at the setup is the same as with the CLP-setup (described in section 4.1). Objective is to make a clear comparison in performance of the LQG and PID control algorithms, using the settings that resulted from the optimization routines from section 3.5.

6.1 Comparison of LQG with PID

The controller gains as derived in the optimization routine in section 3.5 are repeated here for completeness.

$$\mathbf{K}_{LQR} = \begin{bmatrix} 3.74 \\ 74 \\ 8.2 \\ 69.5 \end{bmatrix}^T$$

$K_p = 15.7$
$K_i = 42$
$K_d = 1.6$

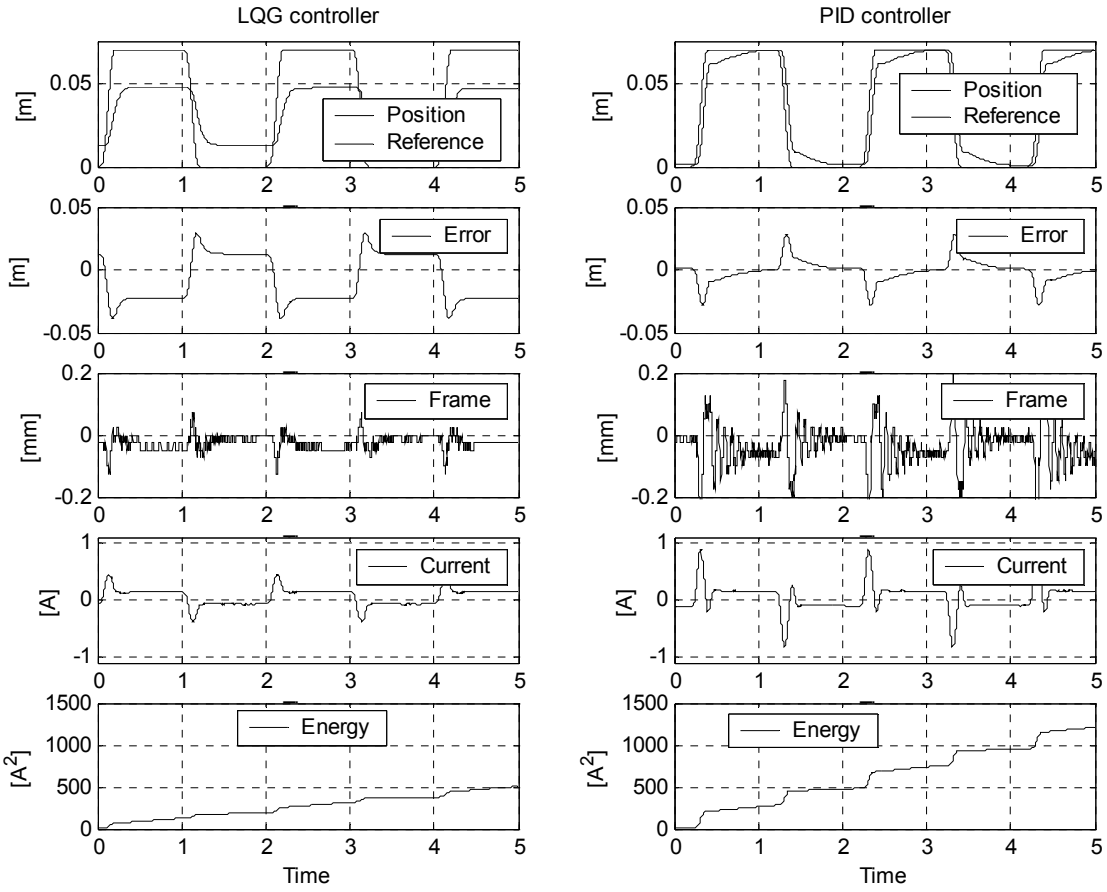


Figure 59: Comparison of tuned LQG and PID control algorithms at mechatronic demonstrator

The following observations can be made.

1. The LQG controlled system does not reach its setpoint of 0.07m – a considerable static error remains in the system. The PID controlled system eventually does reach its setpoint
2. The maximum frame movement in the PID controlled system is 2.5 times the maximum frame movement of the LQG controlled system

3. The maximum current of the PID controlled system is twice the maximum current of the LQG controlled system
4. The PID controlled system consumes about 2.5 times the power of the LQG controlled system

Basing the comparison in performance of the two control systems mainly on position error (difference between slider (with respect to fixed world) and reference) the PID controlled system considerably outperforms the LQG controlled system. This is expected to be the result of the LQG controller being based on an incompetent mathematical model of the plant. The model, originally corresponding to the CLP demonstrator has not been verified. Differences in amount of friction (viscous, but most likely non linear coulomb friction and stiction) are expected to be the cause of this. Comparing the results of the LQG controlled systems in theory and in practice (Figure 59 with Figure 41) it can be seen that the static error is twice as big in the real situation which indicates strongly that the mathematical model is not corresponding competently to the real process.
Also the PID controller could have better performance if the tuning process (in simulation, section 3.5) according the defined quadratic criterion, would have been performed with a competent model of the system.
The optimal solution to the observation that the current mathematical model is incompetent is to perform identification on the plant. In accordance with the objective to have the new mechatronic demonstrator finished at the end of this project, the decision was made to set other priorities than identifying the plant. Assuming that differences between process and reference model cause the position errors in the LQG system also other, less time consuming methods can be put into practice: expanding the LQG system with an integrating element that reduces the static error is the solution tested on the mechatronic system.

6.2 Addition of integral term

The method of adding an integrator to an LQG control algorithm to compensate for differences between process and reference model has been tested in section 3.5.4 by means of simulation. This section treats the tests on the physical setup. The model used for these tests is depicted in Figure 60.

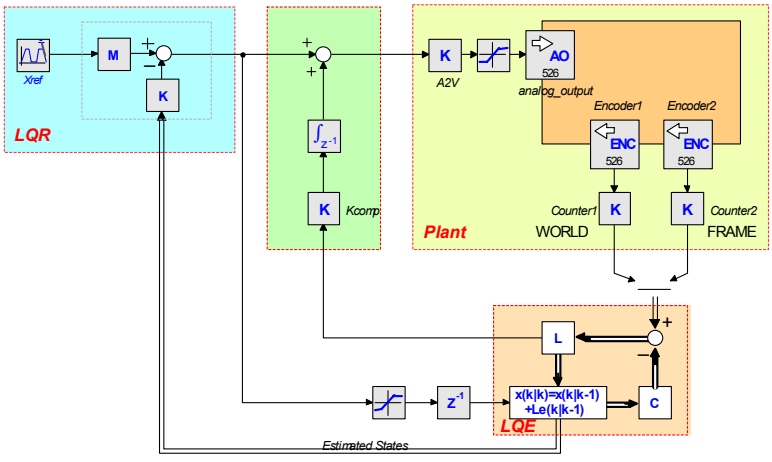


Figure 60: Testing integrator as addition to LQG controller

The gain K_{comp}, that determines the amount of error feedback, is tuned in the real-time environment by comparing the reference signal with the measurements and minimizing the error. This procedure requires manual tuning of K_{comp}. This procedure can be optimized and automated in a later stage.
The results of the addition of an integrator to the LQG control algorithm are substantial. Tuning parameter K_{comp} led to a value of -5000. Figure 61 shows the comparison of the LQG controlled system and the PID controlled system with the LQG system expanded with an integral term.

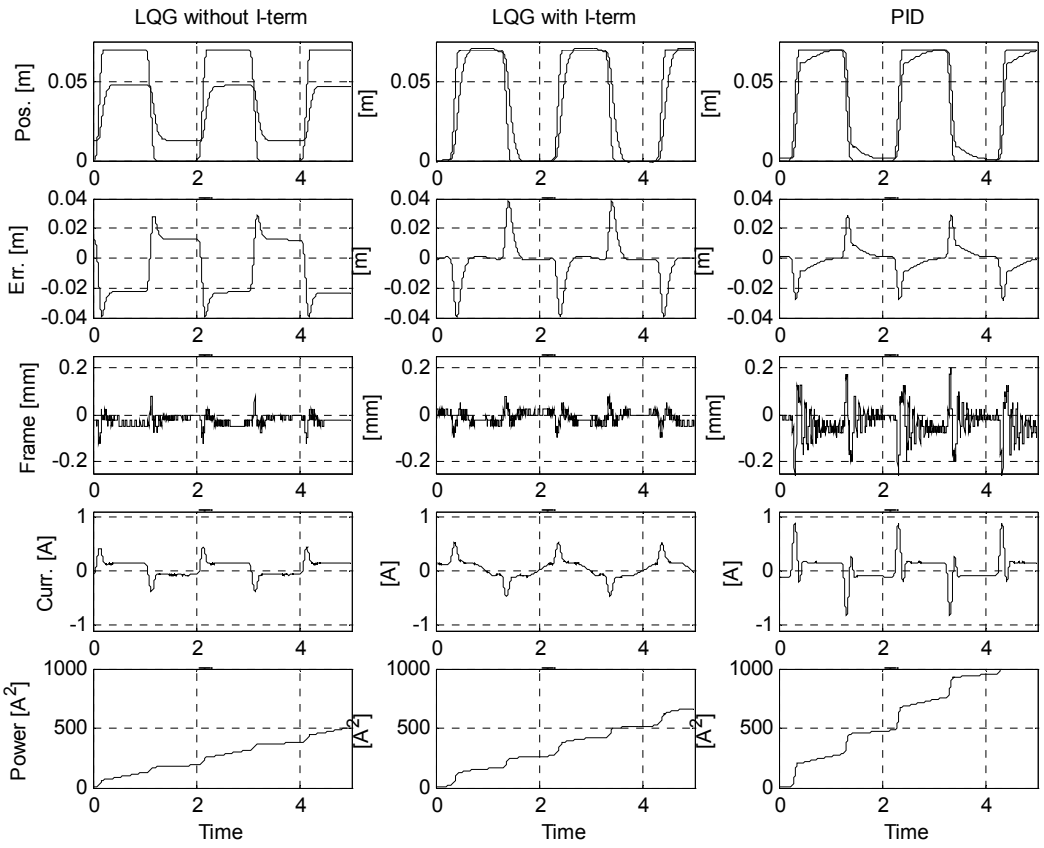


Figure 61: Comparison of LQG, LQG with integrator and PID controlled systems

The expanded LQG control algorithm (defined as LQG+ control algorithm) shows significant reduction of the static error. The LQG+ system also outperforms the PID system in the time needed to reduce the static error. The frame vibration, maximum current consumption and power usage is comparable with the LQG controlled system, but it outperforms the PID system at all these areas. See the following table for an overview of the performance of the three control algorithms.

	LQG	**LQG+**	**PID**
Max. tracking error [cm]	2.9	3.8	2.8
Static error [cm]	2.3	~0	~0
Settling time* [ms]	∞	0.21	0.53
Max. frame mov. [mm]	0.13	0.1	0.28
Max. current [A]	0.44	0.52	0.88
Power usage [A^2]	505	665	1200

On the whole, it can be concluded that the LQG+ system outperforms the PID controlled system on all defined criteria (settling time, maximum frame movement, maximum current usage, power usage) except the maximum tracking error.

*: Settling time is defined as the time in which a position error less than 2% of the setpoint is reached.

6.3 State machine

In addition to the normal operation of running control systems on the mechatronic demonstrator, a mathematical model has been developed that automatically performs 'homing' – a process with which the absolute position of the slider is obtained. The objective is to always perform a homing action before the to be tested control system is started. A state machine is designed specifically for that task. Here are the four states that the state machine contains.

1. Initial State
2. Perform Homing
3. Delay
4. Perform NormalOperation (= testing user defined control system)

The addition of the state machine places a software shell over the control system that will be tested by users. The top-layer of the entire system now looks as in Figure 62.

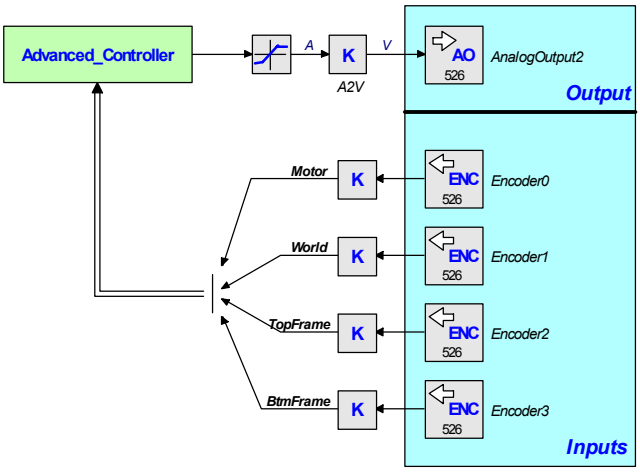


Figure 62: Top layer of improved 20-sim model

All measurements are available in the 'Advanced_Controller'. One hierarchic level lower shows the content of the 'Advanced_Controller'-block – see Figure 63.

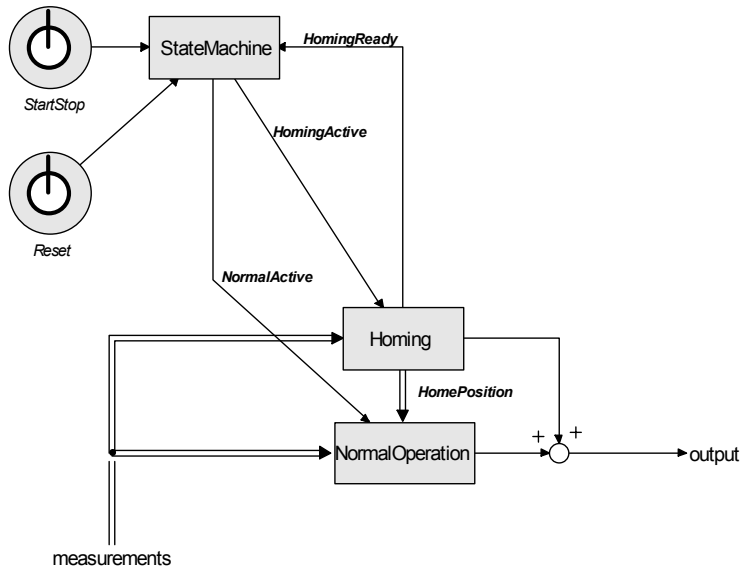


Figure 63: Advanced Controller

The state machine determines its internal state based on the switches Start/Stop and Reset and on the signal 'HomingReady', which is generated by the 'Homing' procedure at the end of its routine. The signals HomingActive and NormalActive indicate whether a certain sub process is set active.

6.3.1 Homing

The goal of a homing procedure is to obtain the absolute position of the slider for all four sensor readings. The encoders used in the system are incremental encoders, which only provide relative position information. The strips and encoder do offer possibilities in reading a index pulse located at a certain position of the strips. This index pulse can be read in software and processed. The procedure of the homing action is based on this index pulse.

1. Slider moves to the right until it passed the index pulse
    a. Internal PID controller is implemented to control the homing velocity
2. Slider moves to the left until the predefined 'nullposition' is reached
3. The software determines the offset that is required to zero all sensor counter values at this 'nullposition'
4. Offset values are stored for the rest of the time the model is used
    a. Sensormeasurements minus offset values yield absolute positions

Figure 64 depicts the homing procedure.

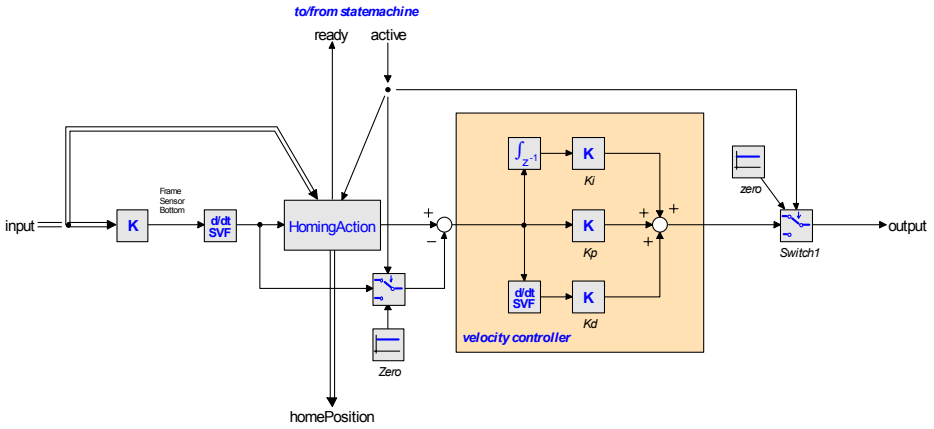


Figure 64: Sub process: "Homing"

The block 'HomingAction' contains also a statemachine that determines what action is to be undertaken (go right to indexpulse, go left to nullposition or determine offsets).

6.3.2 Normal operation

The sub process 'Normal operation' can be adjusted by the user to test any desired model in practice. The offsets defined by the homing procedure are subtracted from the sensor measurements to obtain the absolute sensordata. The block 'Normal operation' is depicted in Figure 65.

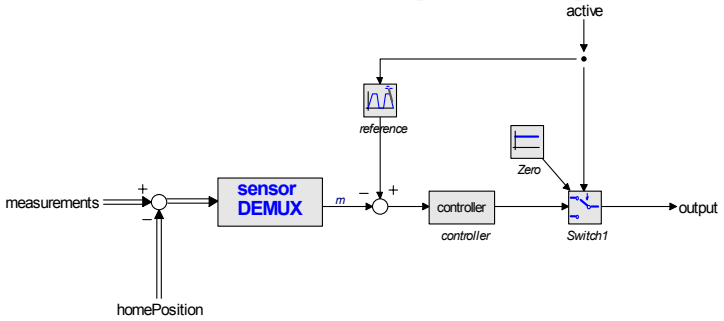


Figure 65: Sub process: "Normal operation"

6.3.3 Safety system

A software safety system has not yet been developed for the mechatronic demonstrator. It is recommended that such a system is designed before the system will be used for actual demonstration purposed. A safety system could consist of continuous checks, which activate a safety procedure if one of these two situations occur.

1. Position breach – if the slider moves outside the 'safe' area, the emergency procedure is called
2. Position & velocity breach – If the slider moves with a velocity higher than defined as maximum at a position near the boundaries

The safety procedure could consist of taking over the control of the analog output and actively decelerating the slider. This procedure could be implemented as extra sub process for the state machine: if a safety violation occurs, the deceleration will take place.

7 Conclusions & Recommendations

7.1 Conclusions

This Masters of Science assignment resulted in a real demonstration setup on which various control systems can be tested. The demonstrator that has been created is a compact, integrated machine that requires only two cables to be connected for normal operation. Installation of Linux in combination with the CosaTeq real-time server required considerable amount of manual work. After getting the system to run, it proved to be a reliable system with extensive remote capabilities. Compiled models created in 20-sim can be uploaded, deleted, started and stopped via an Ethernet connection. Parameters can be changed in real time and the variables are sent to the client PC for monitoring or processing. A special feature of one of the client tools is the ability to present an animation of the setup that shows the movement of the demonstrator in real time. The client tool made by controllab products has been extensively used for demonstration purposes. Improvements to this program have been suggested and implemented to make the tool even more user friendly.
Simulation models have been created with PID and LQG control algorithms and have been tuned to control according to an equal mathematical criterion. These control systems have been tested on both the demonstration setup of CLP as the new mechatronic demonstrator. Tests on the demonstrators showed that nonlinear friction (stiction, coulomb friction) decreases the performance of an LQG control algorithm considerably. By some intuitive maintenance actions, such as greasing the slider and loosening the geared belt, the negative effects of nonlinear elements could be decreased. More importantly, the mathematical model that was used appeared to be not competently describing the real process, due to nonlinearities in the physical setup. An addition to the LQG control algorithm was implemented to compensate for differences between process and model. The effects of this integrator, connected to the input of the plant were considerable. This so called LQG+ control algorithm has outperformed the PID controlled system at static error reduction, reduction of frame vibrations, maximum current usage and overall power usage. Even though an LQG(+) control system is more time consuming to implement, proper use of it will give an increase in performance.
A high-level control system environment has been developed in 20-sim, which contains a homing action to obtain the absolute position of the slider. It is a versatile development environment for testing various control systems in the future.

7.2 Recommendations

7.2.1 Hardware

Even though the Mechatronic Demonstrator has become a fairly small and integrated machine, the setup is still quite heavy – too heavy to be carried far by hand. First point of improvement therefore goes to weight reduction. The frame could be constructed of smaller tubes and the steel plating could have a smaller thickness. Furthermore, the power amplifier could be replaced by a smaller, less over-dimensioned variant. Probably the same power supply could be used for both motor and electronics. If the preference for off-the-shelf components is dropped, the power amplifier might also be home built (e.g. with an H-bridge). Final improvement of the current system is the addition of a more intelligent emergency braking system. The current hardware safety system consists of two end-stop switches and a relay that shuts down the motor amplifier when a safety switch is pressed. A more advanced system would actively brake the motor and could give some sort of feedback to the controller.
As an extension of the system, one could place multiple PC104 CPU-boards on the demonstrator to experiment with parallel processing. One CPU-board could be running for instance the observer, whereas another board would handle the control task. Parallel processing can also be put into action for an emergency-monitoring task. How to perform communication between these boards is another interesting research topic that could be investigated on this demonstration setup.

7.2.2 Software and Control

More control algorithms can be tested on the Mechatronic Demonstrator, such as MRAS, feed-forward, iterative learning control systems and key sample machines. For some control algorithms, an extensive mathematical model is required. Further research on the competency of the currently used model is advised, to extract a proper model of the system. Especially the friction in the system is interesting to identify in more detail.
However, it can be improved even more, for instance in automatic scaling of the graphs.
Finally, some bugs were discovered in the code generator toolbox of 20-sim. There still exists a bug in setting vectors and matrices as favorite, which will be improved in future versions of 20-sim. Finally the commercial CosaTeq software could in the future be replaced by own software, such as the programs written by Buit (Buit, 2005). This would exclude the need for Comedi device drivers which are significantly time consuming to develop. Furthermore, in some cases functionality of the hardware is limited by the use of a Comedi interface (such as in the case of reading index pulses and writing to registers directly).
The operating system of the mechatronic demonstrator should still be changed from a SuSE to a Debian operating system to fasten the boot and shutdown procedure. Furthermore, the system can then be shut down without risking damage to the operating system.
The propositions for a Kalman filter wizard can be put in practice by actually developing such a wizard for 20-sim.

7.2.3 Questions for lab works

The final recommendation concerns the student projects and/or lab works that the Mechatronic Demonstrator is also intended for. A number of suitable questions for students should be formulated. Some of the problems encountered during the design and testing of the demonstrator are suitable for this purpose, such as the following.

1. What problems are caused by linearisation around zero of a coulomb friction element?
2. Show the effects of changing (or unknown) mass and other model parameters in LQG controllers.
3. Demonstrate the effects of noise on real measurement systems (compare LQG with for instance PID)
4. Demonstrate the effects of low sample frequencies in practice
5. Implement add-on systems to an LQG controller to compensate for nonlinearities

It is recommended to give more thought to possible assignments that could be set up for projects and lab works performed on the Mechatronic Demonstrator.

Appendix I – Budget

Part	Type	Supplier	Amount	*Price each*	*Price*
PC/104	Via Eden 600MHz, M570 BAB	Seco	1	€ 342	€ 432
Cable kit	CabKit570	Seco	1	€ 49	€ 49
IO-board	Sensoray 526	Sensoray	1	€ 357	€ 357
Amplifier	4-Q-DC ADS 50/5		1	€ 277	€ 277
RAM	PC133 SODIMM 512Mb		1	€ 116	€ 116
FlashDisk	Compact Flash Card 1Gb		1	€ 87	€ 87
FlashCard conv.	CFC-ADA-1 CF	Kontron	1	€ 39	€ 39
Rail Guide	LLMHS 9 TA 300mm	SKF Multitec	1	€ 200	€ 200
Linear Strip	LIN-250-12-4	US Digital	3	€ 37	€ 111
Encoder	EM1-0-250	US Digital	3	€ 22	€ 66
Pulley	TP7A6MW2-24	Eltromat	2	€ 29	€ 58
Tandriem	TB7EF2-400	Eltromat	1	€ 21	€ 42
Motor	RE-35, 188777	Maxon motor	1	€ 175	€ 175
Motor Encoder	HEDS 5540-110513	Maxon motor	1	€ 55	€ 55
Construction	Quote according to mech.dep.		1	€ 2.650	€ 2.650
Various	Electronics, cables etc.		1	€ 250	€ 250
Total				€ 4.706	€ 4.964

Appendix II – Proposed Kalman filter wizard

Introduction

With an integrated Kalman filter qizard in the simulation package 20-sim a Kalman filter can easily be created. The aim is to facilitate also the less experienced users in generating a powerful tool as this state estimator. The user will be guided through a number of steps in order to implement a suitable Kalman filter: continuous or discrete and with flexible inputs and outputs.

The demands on a Kalman filter wizard have been set up as follows:

1. Support for continuous and discrete KF
2. Restriction to linear KF
3. Both symbolic and numeric KF
4. Both dynamic and static solution of Ricatti equation
5. Easy setting and adjusting of system and KF parameters (ABCD and QR matrices)

Since the Wizard only generates linear Kalman filters, the starting point in building a Kalman filter is 20-sim's "Linear System Editor". Model information can be entered here, after which specific Kalman filter parameters and properties can be set.

This document starts with a general overview of the user interface and functionality of the wizard itself. Secondly several configurations for implementation of this wizard in 20-sim are discussed. Concluding, an implementation of the Kalman filter in a simulation environment will be treated

Contents of the Wizard

When the wizard is called, a number of windows is presented sequentially to the user.

1. Presentation of system matrices, indication of time domain (continuous or discrete) and optionally the sample time
2. Requesting estimator parameters (KF matrices 'Q' and 'R'). Dimensions of matrices are preset
3. Asking for dynamic or static solution of the 'Ricatti equation'
4. Summary of results – overview of KF properties
5. Kalman filter is added to 20-sim model with proper icon

Example of the wizards' setup

Kalman Filter Wizard
Continous time (/ Discrete time)

System equations:
$dx = Ax + Bu$
$y = Cx + Du$

0	1.25	0	0	0
-5988	-11.25	10	0	-5.7
0	3.75	-10	0	5.7
0	0	3.333	0	0
0	0	0	1	0

System overview:
k steering signals
n states
m measured inputs

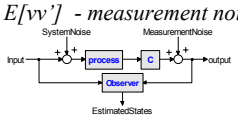
(Sample frequency: .. Hz)

1 / 4

No user input is required in this stage… only presentation of system parameters!

Kalman Filter Wizard
Enter estimator parameters

$Q = E[ww'] - system\ noise$
$R = E[vv'] - measurement\ noise$



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

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

System equations:
$dx = Ax + Bu + w$
$y = Cx + Du + v$

2 / 4

Q and R are initialized as unity matrices. Dimensions are preset (see below)

Kalman Filter Wizard
Solving Ricatti Equation

○ **Static**
Steady state result of ricatti equation will be used in the Kalman Filter. Error covariance matrix P and Kalman gain matrix L are set as parameters.

◉ **Dynamic**
The Ricatti equation will be solved dynamically in the Kalman Filter. P and L are set as variables and will be calculated in the course of the simulation. Speed up factor "λ" will be added to parameter list.

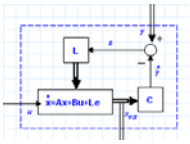
3 / 4

Static: L,P calculated once
Dynamic: Q,R,λ:par. L,P:var.

Kalman Filter Wizard
Summary

A continuous (discrete) Kalman Filter has been created, with the following properties:
k steering signals
n states
m measured inputs
(The system sample time is …Hz.)

The element has been added to the 20SIM editor. For more help, click "go down" on the icon.



4 / 4

Four different icons are available (discrete / continuous and single/multibond input for measurement signals.

Figure 66: Proposition for KF-wizard

Dimensions of signals, variables and parameters

u	:	related to # of columns of B
x	:	related to # of columns/rows of A
y	:	related to # of rows of C
Q, P	:	square matrices - related to # of elements of x.
R	:	square matrix - related to # of elements of u
L	:	#columns: related to # of elements of y, and #rows: related to # of elements of x
λ	:	scalar

Icons for 20-sim

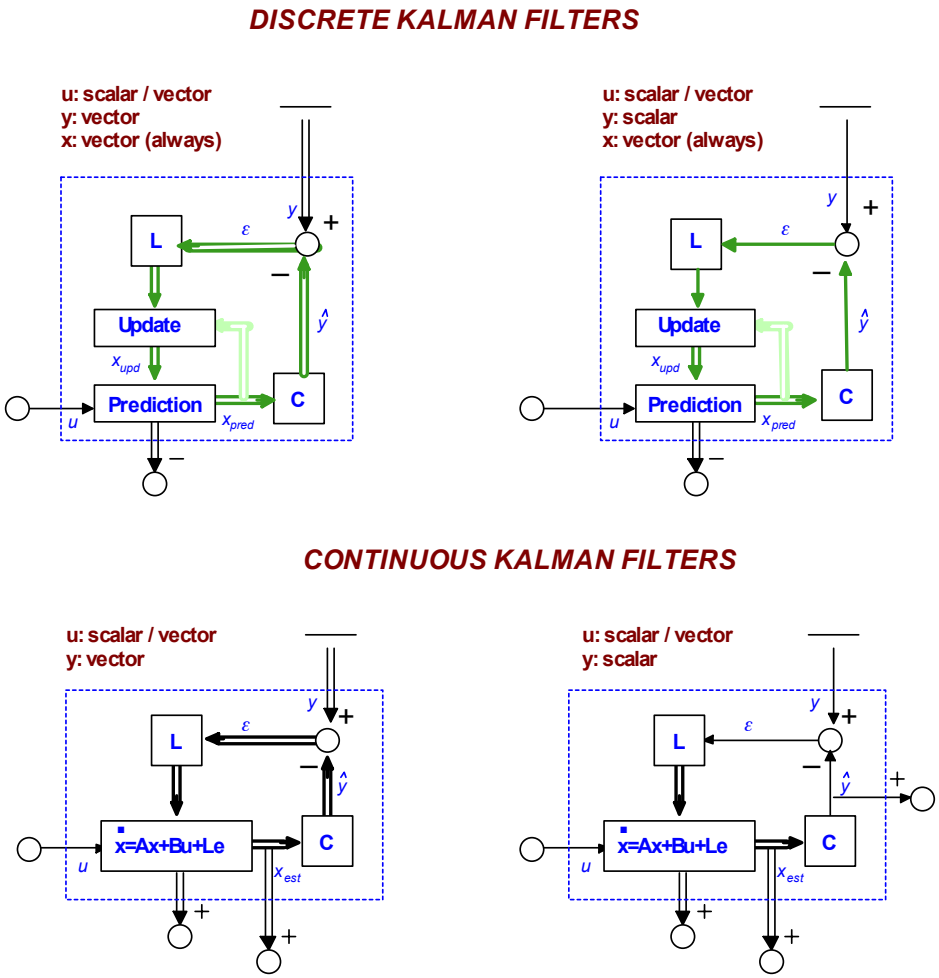


Figure 67: Kalman filter wizard icons

Implementation in 20-sim

Several configurations of the implementation in 20-sim are possible. A list of these possibilities including advantages and disadvantages are presented below:

Suggestion 1

Button in Linear System Editor with which KF-wizard can be called

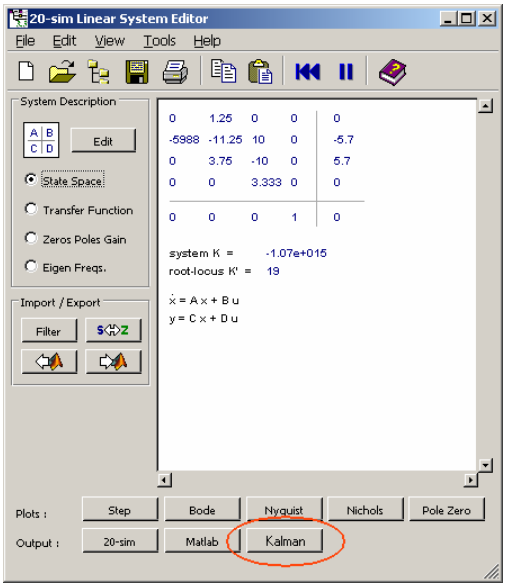


Figure 68: KF proposition 1

This configuration makes clear that the Kalman filter is a part of the linear system editor, which is easily understandable for the users. However, it is a 'one-way' system. Changing the system matrices (A, B, C, D) can not be done easily after generating the KF; if the user wants to 'enter' the submodel, the wizard itself will be called (and not the LSE, where ABCD can be altered).

Suggestion 2

The second suggestion is the integration of the Kalman filter inside the LSE. The current LSE could be 'extended' into an LSE in which the Kalman filter parameters are included. This could be enabled by means of a toggle box:

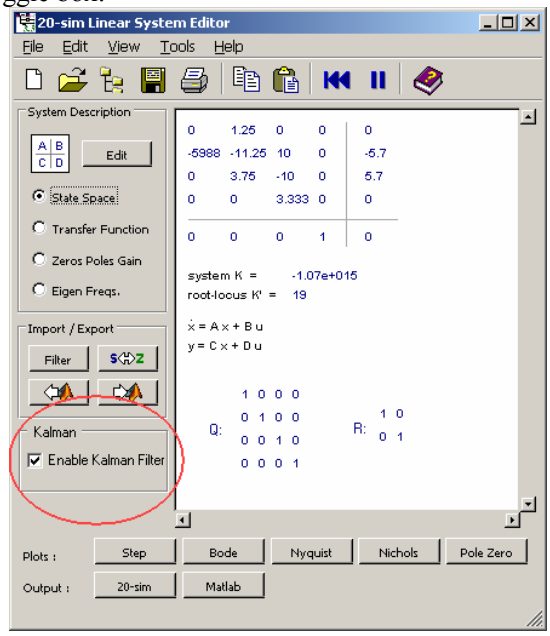


Figure 69: KF proposition 2

This incorporates actually two versions of the LSE: the currently available LSE and an extended version if a Kalman filter is desired. All functionality (including afterwards changing of ABCD) can be included, but the user friendliness might be at stake with this concept.

Suggestion 3
Another option is building an entirely separate wizard. A new menu item could be created from which the Kalman filter wizard could be invoked. This is a userfriendly option and very insightful for the user. However, building a Kalman filter wizard from scratch make this a labour intentensive option.

Suggestion 4
This final suggestion is a combination of the two previous suggestions: building a separate Kalman filter wizard *based on* the (existing) Linear System Editor:

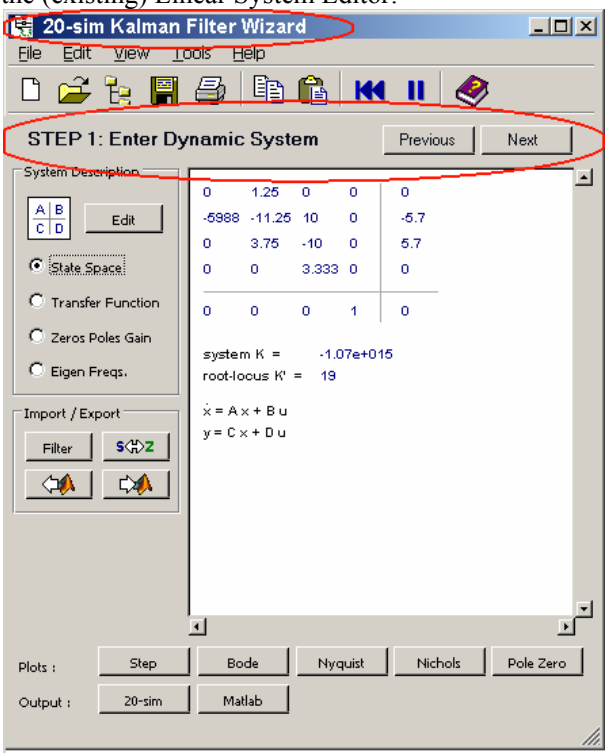


Figure 70: KF proposition 3

This is the most desired option; userfriendlyness is high since the wizard is separated from the LSE. The existing LSE software can be used to form the basis of the wizard. Caution has to be taken with the communication however: the LSE is developed in Smalltalk, whereas the wizard-functionality should be implemented in C++. Communication between these two frameworks is a point of concern, but these obstacles are expected to be overcome.

Furthermore, a link must be made between the Linearization toolbox and the Kalman filter wizard. It might be possible to invoke the Kalman filter wizard after the linearization process.

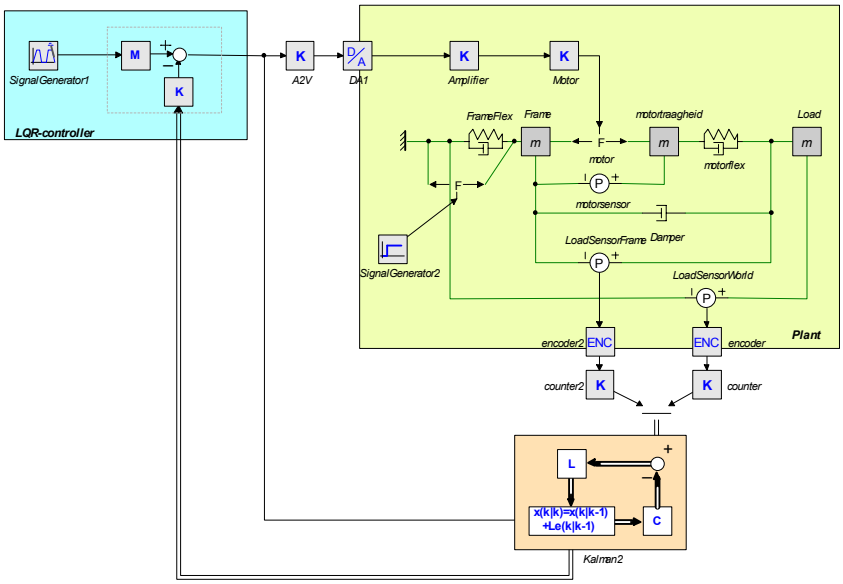
Testing

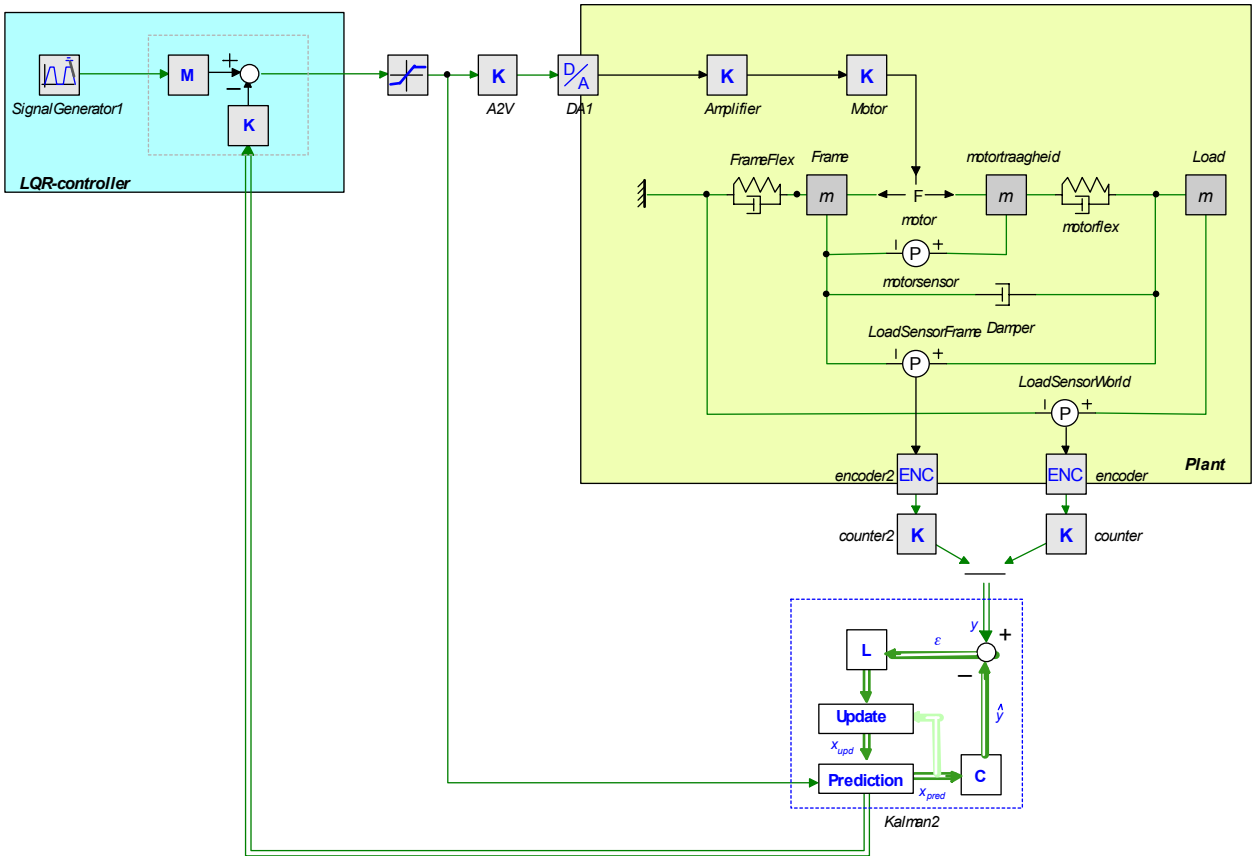
Basics of discrete Kalman filter

$$\vec{y}_{est} = C \cdot \vec{x}_{pred} + D \cdot \vec{u}$$

$$\vec{x} = \vec{x}_{pred} + L \cdot \vec{\varepsilon}$$

$$next\left(\vec{x}_{pred}\right) = A \cdot \vec{x} + B \cdot \vec{u}$$





Solving discrete Ricatti Equation

$$next(P, P0) = A \cdot (I - L \cdot C) \cdot P \cdot A^T + Q$$

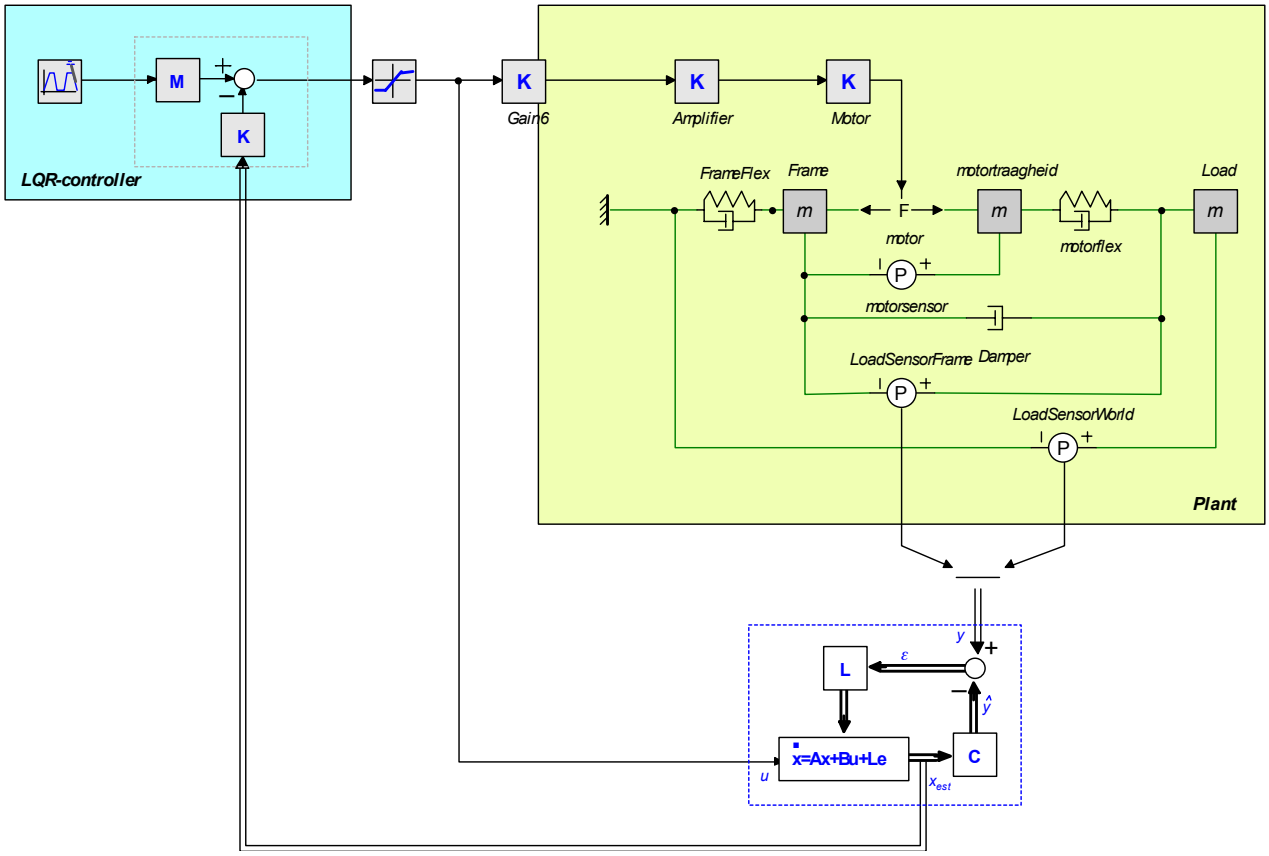
$$L = P \cdot C^T \cdot (C \cdot P \cdot C^T + R \cdot I)^{-1}$$

Continuous

Basics of continuous Kalman filter

$$d(\vec{x})/dt = A \cdot \vec{x} + B \cdot \vec{u} + L \cdot \vec{\varepsilon}$$

$$\vec{y}_{est} = C \cdot \vec{x} + D \cdot \vec{u}$$



Solving Ricatti Equation

$$d(P, P_0)/dt = A \cdot P + P \cdot A^T + G \cdot Q \cdot G^T - P \cdot C^T \cdot R^{-1} \cdot C \cdot P$$

$$L = (C \cdot P)^T \cdot R^{-1}$$

Appendix III – Wiring diagram

This appendix gives information about the wiring of the mechatronic demonstrator. The electronics in the setup basically consist of the following parts:

1. Power supply for PC104 (mini PC power supply)
2. Power supply for motor amplifier (self built)
3. PC104 CPU board
4. PC104 Sensoray 526 I/O board
5. Motor amplifier
6. Motor
7. Motor Encoder
8. Three Encoders on slider
9. Side panel
10. 230V and Ethernetconnectors at back panel

Power supplies

Both power supplies are connected to the (fused) 230Volts inlet that is located in the backplane of the demonstrator. The power supply for the motor amplifier is home-built, according to the schematic of Figure 71:

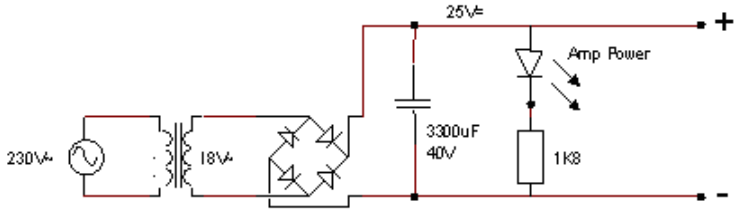


Figure 71: Schematic of motor amplifier power supply

The 'Amp Power'-LED is located on the side panel.

PC104 CPU board

The CPU board is supplied by +5Volts and ground from the "mini PC power supply", located on the right of the base plate of the mechatronic demonstrator. Cables are connected for Ethernet, USB0, USB1, mouse, keyboard and video, according to the connector diagram in Figure 72. The Sensoray 526 I/O board is connected to the PC104-bus located at the lower side of the CPU board.

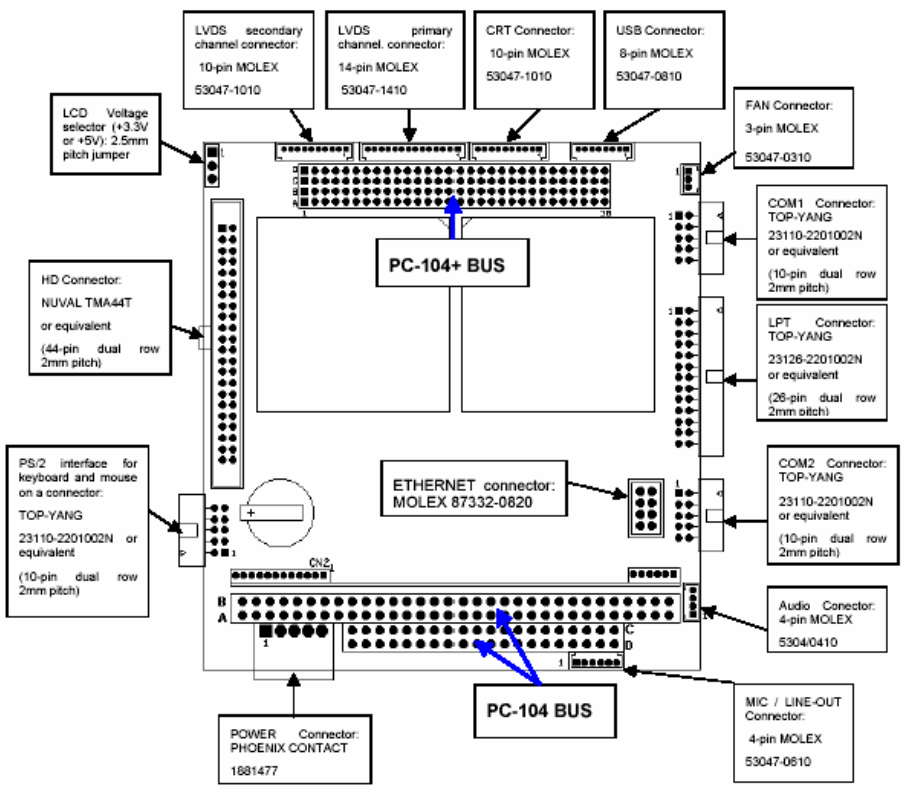


Figure 72: Connections on M570 PC104 CPU board (top view)

PC104 Sensoray 526 I/O board

The I/O board is placed directly on top of the CPU board via their PC104-bus. Furthermore, the analog and digital interfaces are used. See Figure 73 for their locations:

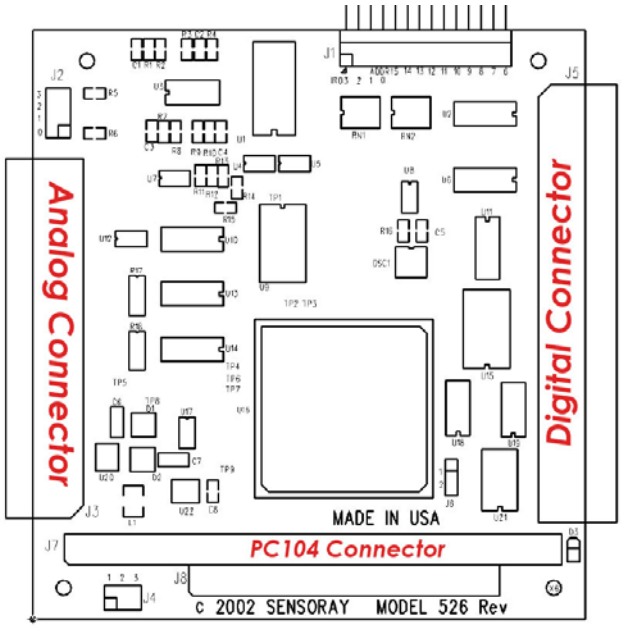


Figure 73: Connectors on Sensoray 526 I/O board (top view)

Digital Circuitry

The digital circuitry consists of:

1. Three encoders on slider
2. One encoder on motor
3. Two status LED's ('Server Online' and 'Model Started'; both located at side panel)
4. Digital part of Sensoray 526 I/O board
5. Possible future expansion: end stop switches

The sensors are located according to Figure 74:

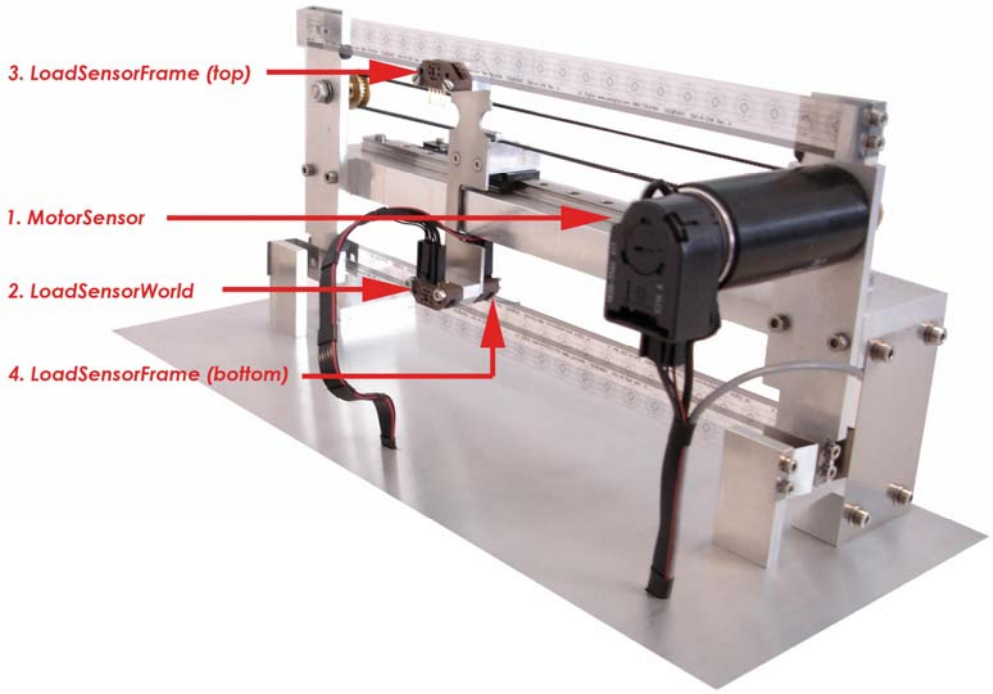


Figure 74: Locations of sensors (rear view)

Here are the connection diagrams of both type encoders:

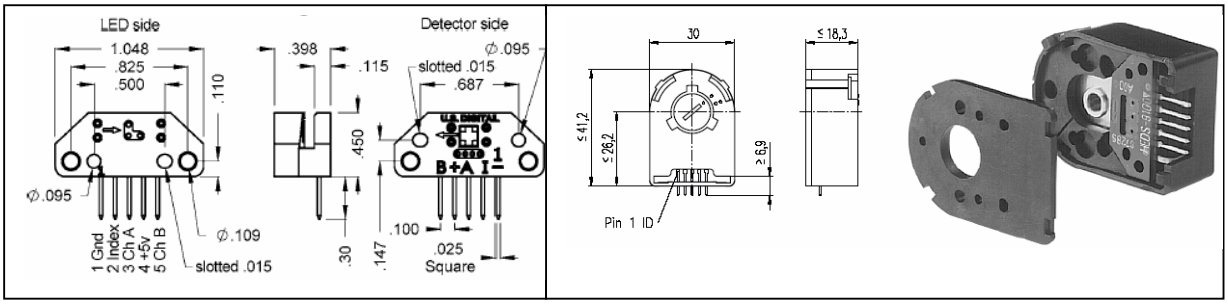


Figure 75: Connection diagrams slider sensors EM1-0-250 (l) and motor sensor HEDS55 (r)

The pinouts of both sensors are equal:

1. Ground
2. Index
3. Ch. A
4. +5V
5. Ch. B

Connection diagram signals on digital connector I/O board:

SENSORAY526-Dig.Conn.(J5)		CABLE	COMPONENTS ON MACHINE	
Pin	**Signal**	**Color**	**Pin**	**Signal**
1	Clock A 0 -			
2	Clock A 0 +		3	Encoder 0 Clock A+ (MOTOR ENCODER)
3	Clock B 0 -			
4	Clock B 0 +		5	Encoder 0 Clock B+
5	Index 0 -			
6	Index 0 +		2	Encoder 0 Index +
7	Count Enable 0	Local ??		+5V / n.c.
8	Couter Output 0			
9	Encoder 0 power (+5V)		4	Encoder 0 power (+5V)
10	Ground		1	Encoder 0 Ground
11	Clock A 1 -			
12	Clock A 1 +		3	Encoder 1 Clock A+ (WORLD ENCODER)
13	Clock B 1 -			
14	Clock B 1 +		5	Encoder 1 Clock B+
15	Index 1 -			
16	Index 1 +		2	Encoder 1 Index +
17	Count Enable 1			+5V / n.c.
18	Couter Output 1			
19	Encoder 1 power (+5V)		4	Encoder 1 power (+5V)
20	Ground		1	Encoder 1 Ground
21	Clock A 2 -			
22	Clock A 2 +		3	Encoder 2 Clock A+ (FRAME ENC. TOP)
23	Clock B 2 -			
24	Clock B 2 +		5	Encoder 2 Clock B+
25	Index 2 -			
26	Index 2 +		2	Encoder 2 Index +
27	Count Enable 2			+5V / n.c.
28	Couter Output 2			
29	Encoder 2 power (+5V)		4	Encoder 2 power (+5V)
30	Ground		1	Encoder 2 Ground
31	Clock A 3 -			
32	Clock A 3 +		3	Encoder 3 Clock A+ (FRAME BOTTOM)
33	Clock B 3 -			
34	Clock B 3 +		5	Encoder 3 Clock B+
35	Index 3 -			
36	Index 3 +		2	Encoder 3 Index +
37	Count Enable 3			+5V / n.c.
38	Couter Output 3			
39	Encoder 3 power (+5V)		4	Encoder 3 power (+5V)
40	Ground		1	Encoder 3 Ground
41	DIO0			LED ‘Server Online’
42	DIO1			LED ‘Model Started’
43	DIO2			Reserved for Endstop-Switch1
44	DIO3			Reserved for Endstop-Switch2
45	DIO4			
46	DIO5			
47	DIO6			
48	DIO7			
49	WDT relay 0			
50	WDT relay 1			

Analog Circuitry

Consists of:

1. Motor
2. Motor Amplifier (connection + settings)
3. Analog part of Sensory 526 I/O board
4. Power Supply

See Figure 76 for a top-view of the Maxon ADS 50-5 motor amplifier:

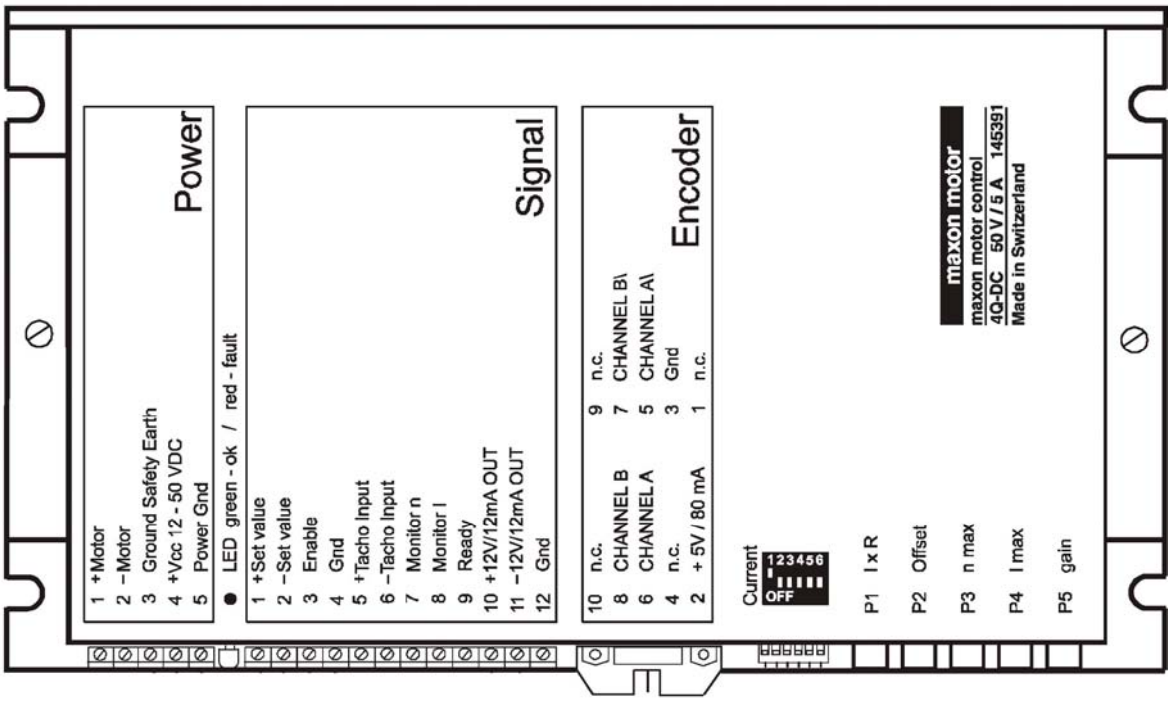


Figure 76: Layout of Maxon motor amplifier

The motor amplifier is set as 'Current Amplifier' (dipswitch 1 active) and is tuned to give a current output 1A at 10V input voltage.

Connection table:

MOTOR AMPLIFIER		
Pin	**Signal**	**Signal**
A1	+ Motor	Motor pos. conn.
A2	- Motor	Motor neg. con
A3	Ground Safety Earth	Earth (=base metal =earth 230V)
A4	+ Vcc 12-50 V	Power supply "Motor Amplifier" output
A5	Power GND	Power supply "Motor Amplifier" ground
B1	+ Set value	Analog Output0: An. Connector Sensoray 526 - pin 23
B2	- Set value	Return0: GND Analog Connector Sensoray 526 - pin 25
B3	Enable	B10 (+12V)
B4	Gnd	
B5	+ Tacho input	
B6	- Tacho input	
B7	Monitor n	
B8	Monitor i	
B9	Ready	
B10	+12V/12mA out	B3 (enable)
B11	-12V/12mA out	
B12	GND	

Side panel

The side panel is located on the left of the mechatronic demonstrator. It contains the on/off switch, connectors for the PC104 peripheral equipment and several status LEDS. See Figure 77:

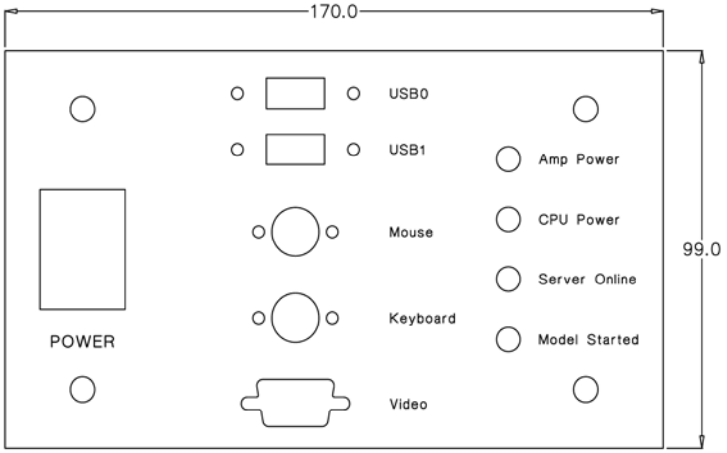


Figure 77: Side panel layout

Appendix IV – Manual for ProSys-RT Installation

Introduction

Cosateq ProSys-RT is a real-time extension for a SuSE Linux system. A RT model frame makes it possible to execute 20-sim models with hard real-time requirements. The MDI server permits to exchange models, data and parameters over a network interface in real-time.

Cosateq Regelungs- und Systemtechnik Johannes-Jung-Str. 7 88239 Wangen Mail: info@syscongroup.de

Details

Cosateq ProSys-RT consists of the following software parts:

Linux side

- Precompiled Linux 2.4.18 kernel with RTHAL5 patch (from RTAI)
- Precompiled RTAI 24.1.9
- Precompiled Comedi 0.7.65 & Comedilib 0.7.16
- MDI server (model upload, start, modify parameters)
- /home/model directory with scripts for model control
- tftp client (not installed by Cosateq, but it should be there)
- Webserver for model control

Windows side

- 20-sim code generation template
- RT-Linux compiler (Cygwin with gcc 2.95 Windows-to-Linux cross compiler)

The following actions must be undertaken at the windows client PC:

1. Install 20-sim (current version: 3.6.04)
2. Install RT-Linux-Compiler
3. Install TFTP server
4. Copy Templates directory to local disk
5. Install Client tools (CosaTeq or CLP-tools or both)
   Note: in some cases the CosaTeq client tools require LabView Runtime Engine 7.0 in order to run. Install this if required.
6. Replace existing 'version.h' by newer version to:
   C:\Program Files\cygwin\usr\local\linuxpc\i386-pc-linux-gnu\include\linux
7. Set CodeGenerator in 20-sim to use new templates
   - tools, option, codegeneration
8. Set TFTP server root directory
   - make directory c:\rtlinux
   - view, options
   - set to: c:\rtlinux

Experience

Remark: Points below are based on Cosateq Prosys-RT 2.1 and SuSE 8.2.

- ProSys-RT 2.1 expects a SuSE based Linux system. Trying to run the installation software on a different Linux distribution does not result in a proper installed RT environment => REQUIRES SUSE LINUX. Installation on a different Linux distribution requires a lot of handwork.

- Installation of SuSE on the Seco PC/104 pc's is not straightforward. The installation kernel does not support the on-board chipset => installation from CD does not work. Besides this, installation of SuSE on a 1 GB Compact Flash card did not work at all.

- Although Cosateq uses GNU based opensource software they do not deliver the used kernel, RTAI and Comedi sources on their CD. Adding a new driver for unsupported I/O hardware to Comedi requires these sources.

- The ProSys-RT MDI server requires a library that it available only when installing the graphical user interface. (The library is not installed by the ProSys-RT installation.

- The used 2.4.18 kernel does not compile anymore with the current (>2.95) gcc compilers.

- Using a kernel source with a different version requires manual adaptation of the ProSys-RT startup scripts => Own RTAI patched kernel with Comedi drivers do not work by default.

- For ProSys-RT, the kernel, RTAI and Comedi need to be compiled with a gcc 2.95 compiler, which is not delivered with SuSE anymore.

- Using a different kernel version at the Linux side requires also a manual adaptation of the CygWin compilation environment at the Windows side. (kernel sources at both sides need to be the same).

Literature

ADSP (2002), "*VisualDSP++ 3.0 C/C++ Compiler and Library Manual for ADSP-219x DSPs, revision 4.0"*, Analog Devices, Inc., Norwood, Mass.

Amerongen, J. van, (2005), (http://www.drebbel.utwente.nl/)

Amerongen, J. van, (2001), "Regeltechniek", University of Twente, Enschede

Amerongen, J. van, Vries, T.J.A de, (2001), "Digitale regeltechniek", University of Twente, Enschede

Åström, K.J., Hägglund, T., (1995), "PID Controllers: Theory, Design and Tuning", Instrument Society of America, Research Triangle Park

Buit, E., (2005), "PC104 stack mechatronic control platform", University of Twente, Enschede

Coelingh, H.J., (1997), "Automated conceptional design of controllers for mechatronic systems", University of Twente, Enschede

Controllab Products B.V., (2005) (http://www.20sim.com/)

CosaTeq, (2005), (http://www.cosateq.de/)

Groothuis, M.A., (2004), "Distributed HIL simulation for BodeRC", University of Twente, Enschede

Kleijn, C., (2003), "Ontwerp Demo Opstelling", internal rapport Controllab Products B.V., Enschede

Kopetz, H., (1997), "Real-time systems, design principles for distributed embedded applications", Kluwer academic publishers, Boston

TCPIPguide, (2005), (http://www.tcpipguide.com/free/t_TrivialFileTransferProtocolTFTP.htm)