PERGAMON

Mechatronics 13 (2003) 1045–1066

MECHATRONICS

# Mechatronic design

Job van Amerongen *

*Faculty of Electrical Engineering, Mathematics and Computer Science, Drebbel Institute for Mechatronics and Control Laboratory, University of Twente, EL-RT, P.O. Box 217, 7500 AE Enschede, The Netherlands*

**Abstract**

Mechatronic design is the integrated design of a mechanical system and its embedded control system. In order to make proper choices early in the design stage, tools are required that support modelling and simulation of physical systems—together with the controllers—with parameters that are directly related to the real-world system. Such software tools are becoming available now. Components in various physical domains (e.g. mechanical or electrical) can easily be selected from a library and combined into a 'process' that can be controlled by block-diagram-based (digital) controllers. A few examples will be discussed that show the use of such a tool in various stages of the design. The examples include a typical mechatronic system with a flexible transmission, a mobile robot, and an industrial linear motor with a neural-network-based learning feed-forward controller that compensates for cogging.


## 1. Introduction

There are various definitions of mechatronics, but a very short one could be formulated as "*Mechatronics deals with controlled mechanical systems that are designed as a whole*" or "*Mechatronic design is the integrated design of a mechanical system and its embedded control system*". This implies that during the design phase the designer (or the design team) should be able to evaluate changes of the mechanical part as well as of the (electronic or embedded-software based) controller part. This allows that the requested performance can be obtained by optimally exploiting the possibilities in the mechanical domain as well as in the embedded controller. For

* Tel.: +31-53-489-2791; fax: +31-53-489-2223.
*E-mail address:* j.vanamerongen@utwente.nl (J. van Amerongen).
*URL:* http://www.ce.utwente.nl/amn.

0957-4158/$ - see front matter 
doi:10.1016/S0957-4158(03)00042-4

a long time control engineers have promoted such a systems approach realised now in mechatronics. This has led to the availability of inexpensive, high-quality consumer products, such as CD- and DVD-players, video recorders etc. Also professional equipment benefits from a mechatronics design approach because this leads to more flexible, less expensive or better performing machines. In this paper a few examples will be shown of projects that were carried out in recent years in the Drebbel Institute for Mechatronics of the University of Twente in the Netherlands. These projects include the development of tools for the design of mechatronic systems, including simulation tools that give the designer direct access to the physical parameters of the mechanical construction as well as those of the controller. An example of a mobile robot will illustrate how simulation can be applied in an early stage of the design process to fix some important parameters. The last example discusses the use of learning feed-forward control that compensates for some typical non-linearities such as friction and cogging that are present in many mechatronic systems.

## 2. Design software

During the design of mechatronic systems it is important that changes in the construction and the controller be evaluated simultaneously. Although a proper controller enables building a cheaper construction, a badly designed mechanical system will never be able to give a good performance by adding a sophisticated controller. Therefore, it is important that during an early stage of the design a proper choice can be made with respect to the mechanical properties needed to achieve a good performance of the controlled system. On the other hand, knowledge about the abilities of the controller to compensate for mechanic imperfections may enable that a cheaper mechanical construction be built. This requires that in an early stage of the design a simple model is available, that reveals the performance-limiting factors of the system. This can only be done when also a sufficiently sophisticated controller is designed for this conceptual model. The controller design includes the choice of appropriate sensors and sensor locations.

Many processes can be reasonably well controlled by means of PID-controllers. This is due to the fact that these processes can be more or less accurately described by means of a second-order model. Tuning rules, like those of Ziegler Nichols, enable less experienced people to tune such controllers. Relatively simple models can also describe many mechatronic systems. A mechatronic system mostly consists of an actuator, some form of transmission and a load. A fourth-order model can properly describe such a system. The performance-limiting factor in these systems is the resonance frequency. A combination of position and tacho feedback (basically a PD-controller) can be applied here as well. But due to the resonant poles proper selection of the signals to be used in the feedback is essential. Efforts have been made [1–3] to derive recipes for tuning such systems, in addition to selecting the proper feedback signals and fixing of the mechanical properties. Computer support tools are essential to enable less experienced designers to use these recipes [4].

### 2.1. Simulation of mechatronic systems

Still there is a gap between simulation software used for evaluation of mechanical constructions and software used for controller design. Mechanical engineers are used to finite element packages to examine the dynamic properties of mechanical constructions. It is only after reduction to low-order models (modal analysis) that these models can be used for controller design. On the other hand, typical control-engineering software does not directly support the mechatronic design process because in the modelling process the commonly used transfer functions and state-space descriptions often have lost the relation with the physical parameters of the mechanical construction. Tools are required that allow modelling of mechanical systems in a way that the dominant physical parameters (like mass and dominant stiffness) are preserved in the model and simultaneously provide an interface to the controller design and simulation tools control engineers are used to [2,3].

Simulation is an important tool to evaluate the design of (control) systems. But most simulation programs do not support physical modelling in a way that direct tuning of the physical parameters of the mechanical construction and those of the controller is possible as required in the design of mechatronic systems. Programs like Simulink (1991) and Vissim (1990) use block diagram representations. These representations only allow for an indirect and complex representation of the physical parameters and are therefore less suited for mechatronic design. But control engineers are so used to these packages that they hardly realise themselves what is needed to make life easier. Examples of programs that support modelling of *physical systems* in *particular domains* are PSPICE (electronic networks), ADAMS (mechanical systems) and SpeedUp (chemical processes). Recently also programs that allow physical modelling in *various physical domains* became available. They use an object-oriented approach that allows hierarchical modelling and reuse of models. The order of computation is only fixed after combining the sub-systems. Examples of these programs are 20-sim (1995), described by Broenink [5] as CAMAS (1990) and Dymola (1993).

In this paper 20-sim (pronounce Twente Sim) [1] will be used to illustrate the simultaneous design of construction and controller in a mechatronic system. 20-sim was developed at the Control Laboratory of the University of Twente as a tool for mechatronic design and to overcome the limitations of the software package TUTSIM that was developed by the group around 1970. In principle TUTSIM only supported block-diagram based models.

20-sim supports object-oriented modelling. Power and signal ports to and from the outside world determine each object [6]. Inside the object there can be other objects or, on the lowest level, equations. Various *realizations* of an object can contain different or more detailed descriptions as long as the interface (number and type of ports) is identical. This allows top–down modelling as well as bottom–up modelling. Modelling can start by a simple interconnection of (empty) sub-models.

[1] 20-sim, http://www.20sim.com.

Later they can be filled with realistic descriptions with various degrees of complexity. De Vries [7] calls this *polymorphic* modelling. Sub-models can be constructed from other sub-models in hierarchical structures.

Modelling in different physical domains requires that a core language be available to describe a system in different domains. This is achieved by coupling models by means of the *flow of energy*, rather than by *signals* such as voltage, current, force and speed.

This way of modelling is well suited for mechatronics system design. It will be illustrated with an example. We want to consider the design of a simple servo system, considering of a voltage source, a DC-motor and a mechanical load driven through a transmission (Fig. 1).

The transmission is disregarded for the time being. The belt is considered as infinitely stiff and the transformation ratio is taken care of by changing the motor constant. If a power amplifier driven by a signal generator describes the voltage source, we can draw the iconic diagram of Fig. 2. Note that all figures have been drawn with the aid of 20-sim. At this stage the different *components* in this model are still empty. But all components have electrical and/or mechanical "ports". With the proper interfaces (ports) defined, the components can be connected to each other.

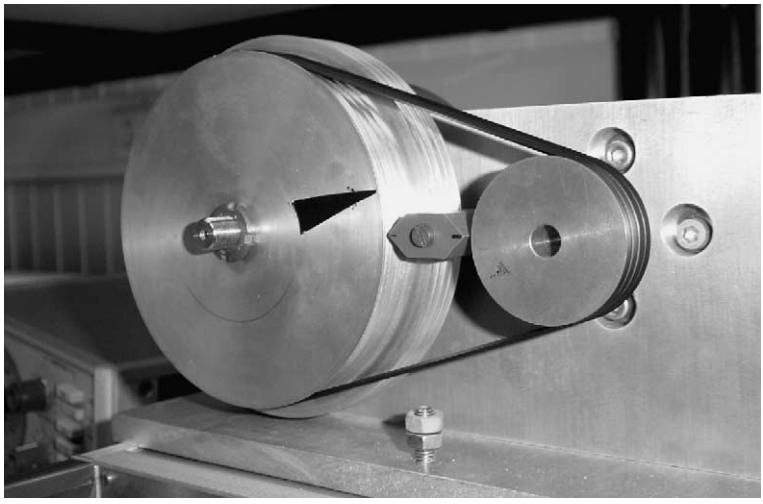

Fig. 1. Simple DC-servo system.

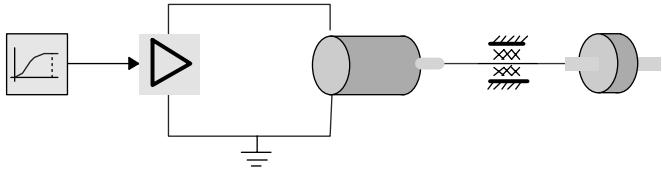

Fig. 2. Iconic diagram of the simple servo system.

In the next step we can detail the description of the DC-motor. One solution could be the description given in Fig. 3. The motor is now described by a number of *ideal physical elements*, each representing a basic physical relation. The motor has an electrical (EL) as well as a mechanical port (MECH).

Each of the *elements* in this figure can be described as an element with an electrical and/or mechanical port. This has been visualised in Fig. 4, which is obtained after redrawing Fig. 3. Each port is described by two signals. For the electrical elements these are the voltage difference over the element and the current through the element. For the mechanical elements these are the (angular) velocity and the torque. The products of these conjugated variables ($P = ui$ or $P = T\omega$) represent power.

The representation of Fig. 3 is the common way to draw diagrams of electrical circuits. Fig. 4 makes the idea of ports more clear. This way of thinking is made even

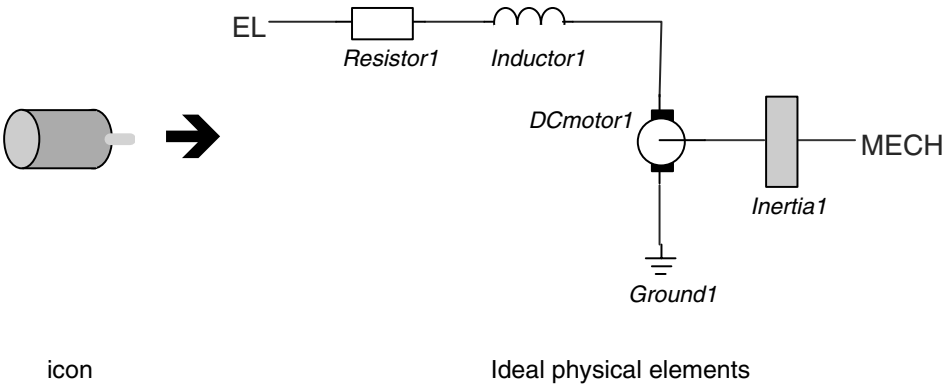


Fig. 3. Icon of the motor expanded to ideal physical elements.

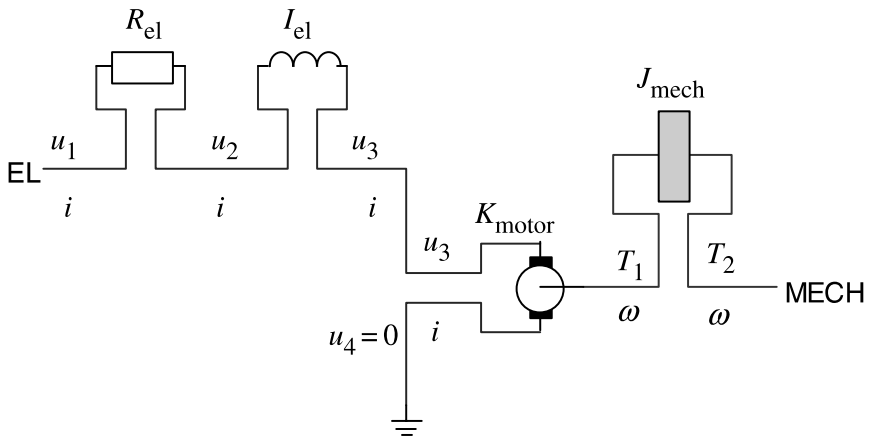


Fig. 4. Ideal elements of Fig. 3 drawn with power ports.

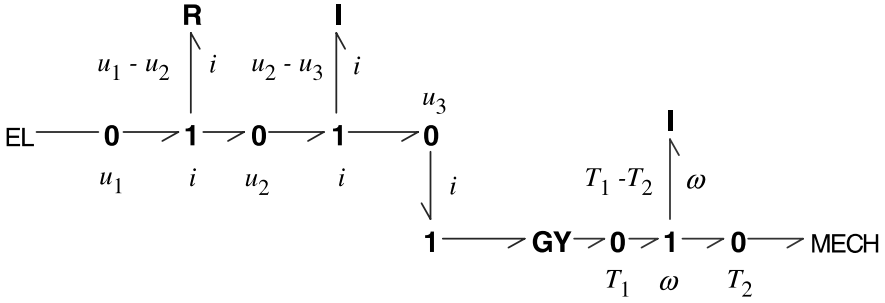


Fig. 5. Figs. 3 and 4 redrawn as a bond graph.

more explicit in so-called bond graphs [8,9] (Fig. 5). A bond graph is a domain-independent representation of a physical system. Each element (e.g. $R$ for electrical resistance or mechanical friction and $I$ for inductance or mechanical mass or inertia) is connected to the junction structure by a power bond. The junction structure consists of '1'-junctions that represent current (and voltage difference) and '0'-junctions that represent voltage in the electrical domain. In the mechanical domain '1'-junctions represent velocity or angular velocity and '0'- junctions force or torque. As long as these energy ports are being used the representations of Figs. 4 and 5 are equal. Use of bond graphs is not essential to use the port-based modelling approach. The iconic diagram representation is easier to read, the more abstract bond graph is a tool that represents the essential properties of a mechatronic system in a very compact way. In addition, it supports the process of finding a proper model as will be shown later. The bond graph of Fig. 5 is a one to one translation of the iconic diagram of Fig. 3 or Fig. 4.

This bond graph can be simplified: 20-sim is able to do that automatically (Fig. 6). This yields a more compact form. It is still directly related to and has the same physical parameters as the iconic diagram of Fig. 3.

If we go down a step further into the hierarchy, we arrive at the level of equations. For instance, an electrical resistor can be described by the equation:

$$u = Ri \tag{1}$$

where the variables $u$ and $i$ indicate the conjugated variables of the electrical port. Note that this is an equation and not an assignment statement. It could have been written equally well in the forms:

$$i = u/R \tag{2}$$


Fig. 6. Simplification of the bond graph of Fig. 5.

or

$$u - Ri = 0 \tag{3}$$

In a similar way the inductance can be described by the equations:

$$u = L\frac{\mathrm{d}i}{\mathrm{d}t} \tag{4}$$

or

$$i = \frac{1}{L}\int u\,\mathrm{d}t \tag{5}$$

In case of an *R*-element there is no preference for one of the forms. For the *I*-element the integral form is preferred in the simulations. 20-sim will automatically determine the preferred causal form and indicate this in the bond graph by so-called causal strokes (Fig. 7).

This form is well suited to derive the simulation equations. We will skip the process of deriving equations because 20-sim is able to do that automatically from either the iconic diagram or from the bond graph.

When port-based modelling is used, the energy flow or *power P* is commonly chosen as the product of *two conjugated signals*, generally called effort (*e*) and flow (*f*):

$$P = ef \tag{6}$$

Examples of this expression in the mechanical and electrical domain are:

$$P = Fv, \quad \text{or} \quad P = T\omega \tag{7}$$

$$P = ui \tag{8}$$

where $F$ is force, $v$ is velocity, $T$ is torque, $\omega$ is angular velocity, $u$ is voltage and $i$ is current.

One could say that the 20-sim software is internally based on bond graphs, but that the introduction of iconic diagrams enables users not experienced with using bond graphs to benefit from its advantages as well.

When we expand the complete Fig. 2 we obtain Fig. 8. Fig. 8 shows the model in the form of ideal physical elements and as a bond graph. When this model is processed a message pops up that indicates that inertia 2 in the iconic diagram has a dependent state. Or, when the bond graph is used, that *I*-element *I*3 in the bond graph has a dependent state. In the bond graph this is also visible by the orange

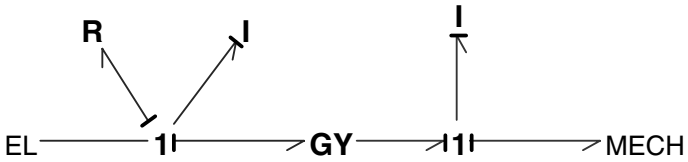


Fig. 7. Bond graph of Fig. 6 enhanced with causal strokes.

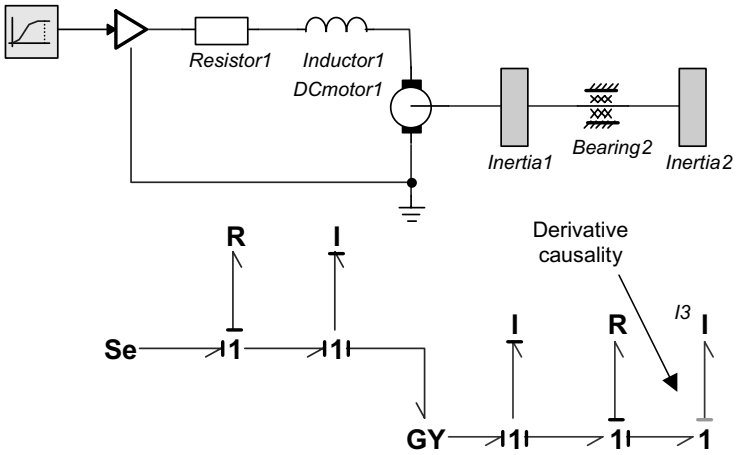


Fig. 8. Complete model in the form of ideal physical elements and as a bond graph.

causal stroke that is not at the end of the half arrow preferred for $I$-elements. Closer examination of the model explains why. In this model the two inertias always have the same speed and therefore, they are dependent. The (orange) causal stroke at the $I$-element and the messages indicate that in the simulation model this element can only be written in derivative form:

$$T = J\frac{\mathrm{d}\omega}{\mathrm{d}t} \tag{9}$$

There are several ways to deal with this problem.

1. The two inertias can be combined into one inertia. (If the symbolic simulation feature is selected this is done automatically in 20-sim. A message pops up that the dependency of the two inertias has been solved symbolically.)
2. The transmission can be added, including some flexibility in the belt.
3. Dealing with the dependent state can be done in the 20-sim simulator by means of an implicit integration algorithm.

If the flexibility is negligible solution 1 leads to the simplest model. 20-sim derives this model automatically when the model is processed for simulation. On the other hand, the provided warning suggests that when the flexibility of the belt cannot be disregarded the model could be extended with a spring element. This will solve the problem of dependent states as well. It should be noted that this should not be done for numerical reasons only. If the transmission were very stiff, this would result in high-frequency dynamics that would require a small integration interval in the simulator and thus lead to long computation times. On the other hand, if the flexibility is important, as it is in this system, the warning helps to draw the designer's attention to the fact that the model has been oversimplified.

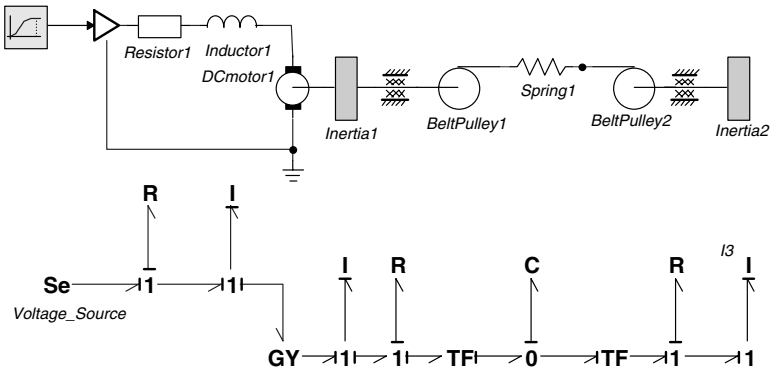


Fig. 9. Model extended with transmission.

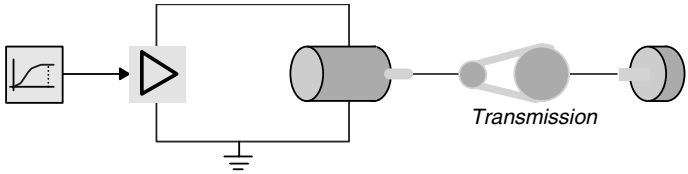


Fig. 10. Representation of Fig. 9 in the form of an iconic diagram with sub-models.

In the next figure the transmission, including a spring element, has been added, which leads to Fig. 9. Processing of this model does not produce any warnings. Again, the iconic diagram and the bond graph are equal. They contain the same information and it is up to the user, which one of the two representations is preferred. Both are given here to show the similarities.

This example illustrates how modern software can help to come up with a model that has the complexity that is needed for a particular problem. Physical models, in the form of an iconic diagram or as a bond graph may help in this modelling process. Both are based on connecting elements by means of power ports. The user can select the preferred view, whether this is a bond graph, an iconic diagram with ideal physical element or a view using higher lever sub-models. The latter is given in Fig. 10 for the system of Fig. 9.

In the next section it will be shown how to use this model for the design of controllers.

## 3. Design of a servo system

Coelingh [2] and Coelingh et al. [3] describe a structural design method for mechatronic systems. The method starts with reducing the conceptual design to a fourth-order model that represents the dominant properties of the system in terms of

the total mass to be moved and the dominant stiffness. This model still has physical meaningful parameters. In this model appropriate sensors are chosen, as well as a path generator. In the conceptual design phase a simple controller is developed and mechanical properties are changed if necessary. Then a more detailed design phase follows where also parameter uncertainties are taken into account. A robust controller is designed by means of quantitative feedback theory (QFT).

Here we will consider some simple aspects of the design of such a system in order to illustrate the advantage of the use of physical models. We consider the model discussed in Section 2, a load driven by an electric motor, through a flexible transmission. A current amplifier has replaced the voltage amplifier. The iconic diagram of this model was given in Figs. 9 and 10.

Because of the flexible transmission, by means of a rubber belt, the resonance in this system is clearly visible in the step responses of Fig. 11.

From the equations used for the simulation, 20-sim can automatically derive a model in the form of a state-space description, a transfer function or poles and zeros. An interface is provided to Matlab enabling, for instance, to automatically compute the gains of an LQR or LQG controller. The Matlab commands can be embedded in 20-sim formula models. If such commands are present, Matlab is automatically started and the results of the computations can be used in the 20-sim simulations. In this example it is not necessary to use Matlab, because the Matrix Ricatti equations needed to find the Kalman filter and controller gains can be solved faster in 20-sim.

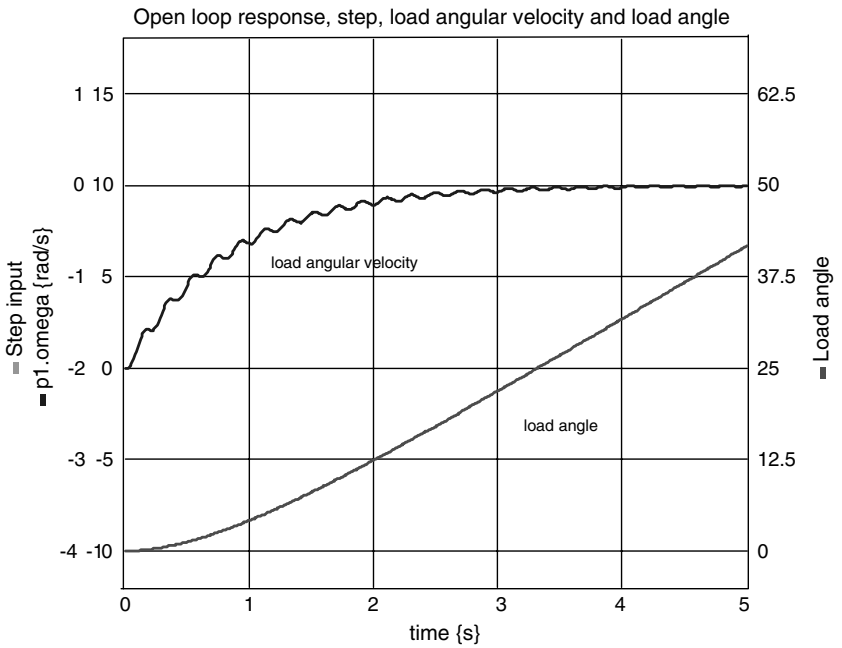


Fig. 11. Open loop responses.

The diagram of the process together with an LQG-controller (Kalman filter and state feedback) is given in Fig. 12 and some responses in Fig. 13.

But it is not always necessary to use such a Kalman-filter based optimal controller structure. A properly designed P(I)D controller can do the job as well, especially when the amount of noise is small. A first attempt could be to use only measurements of the load angle and load speed. This attempt fails, because this leads to a badly damped or unstable system as can be seen from the root locus for variations in the gain of the velocity feedback. 20-sim can easily determine the transfer function between the motor current and the load speed from the responses of Fig. 11. The input output relation is given in state-space form, as a transfer function or as poles and

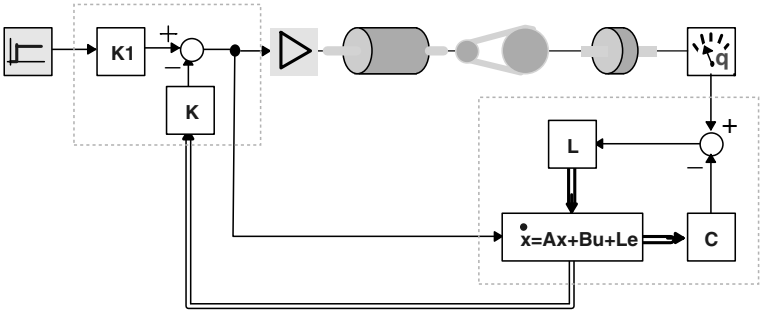


Fig. 12. Process with Kalman filter and state feedback.

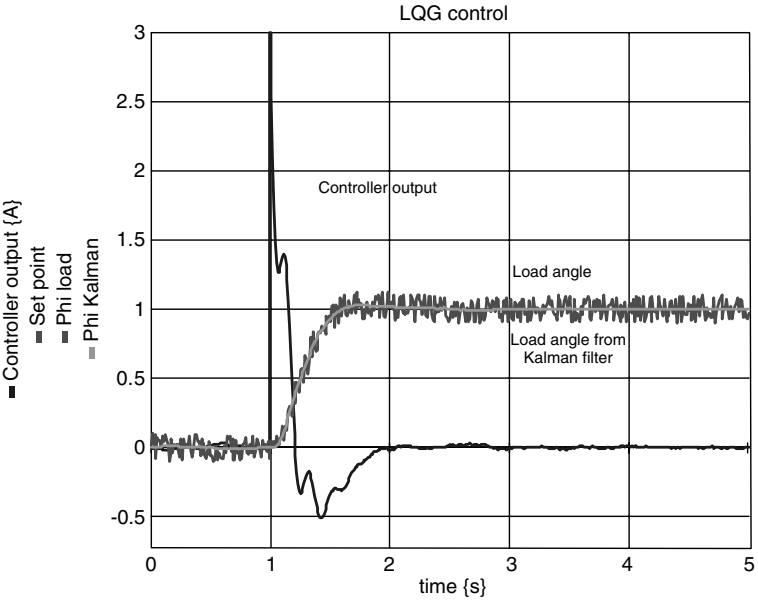


Fig. 13. Response of the LQG-controlled system.

zeros. The matrix form is also available in symbolic form, thus enabling a direct relation with the physical parameters in the controller design phase. From either of these representations various frequency characteristics, a pole-zero plot and a plot of the root locus can be generated (Fig. 14).

Fig. 14 clearly shows that even a small amount of velocity feedback will lead to an unstable system. It is well known that feedback of the motor speed is a better so-

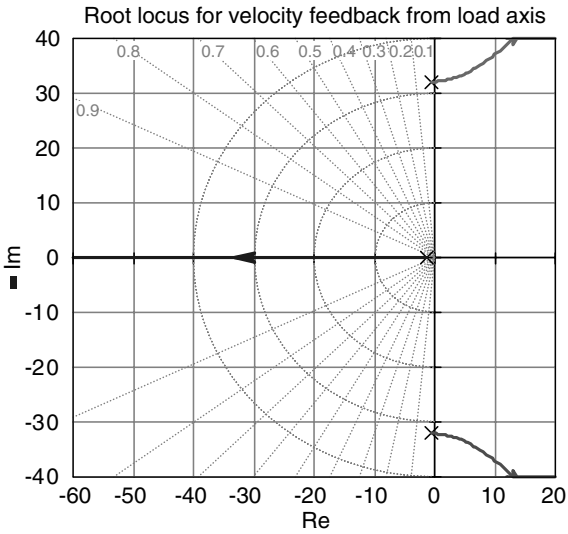


Fig. 14. Root locus for velocity feedback of load axis.

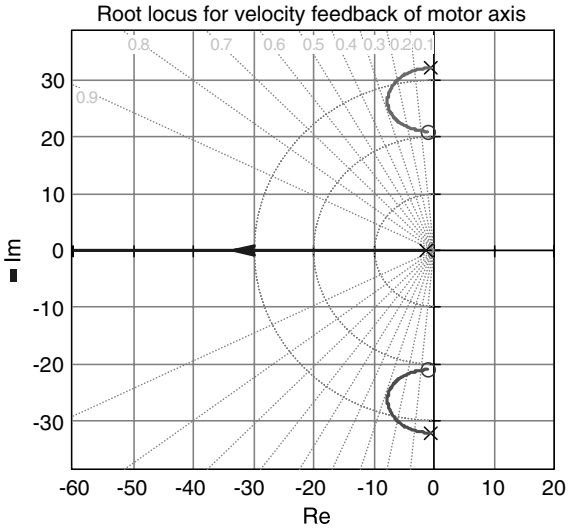


Fig. 15. Root locus for velocity feedback of motor axis.

lution. Using again the model of Figs. 9 and 10 to determine the transfer from input current to motor speed yields the root locus of Fig. 15.

Complex zeros now accompany the complex poles and because they are close together their influence on the response will be almost negligible. The branch of the root locus on the real axis now shows the desired behaviour: moving the dominant pole to the left in the *s*-plane. Combining the feedback of the motor speed with feedback of the load angle yields the PD-controller structure of Fig. 16 and the responses of Fig. 17. Except for the noise there is not much difference with the responses of the system with the Kalman filter, although the PD-controlled system is simpler.

Suppose that the belt in the transmission has a limited strength. Using the state-event feature of 20-sim the moment of breaking of the belt can be exactly determined.

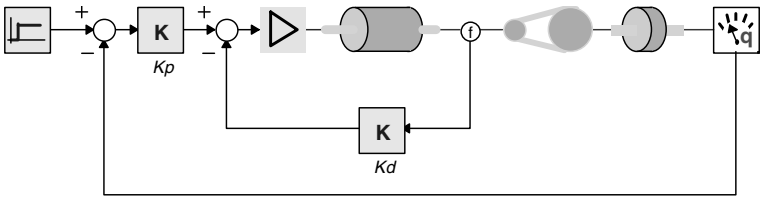


Fig. 16. Servo system with PD-controller.

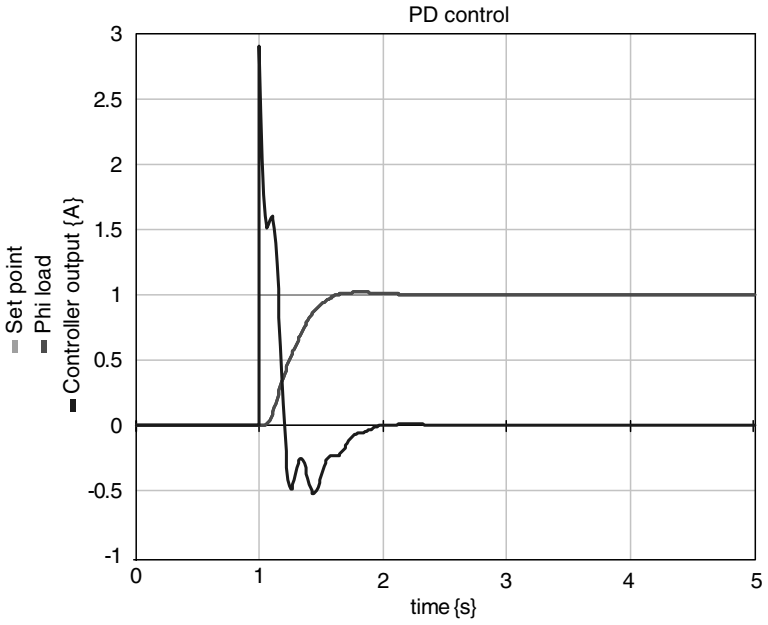


Fig. 17. Responses of the system of Fig. 16.

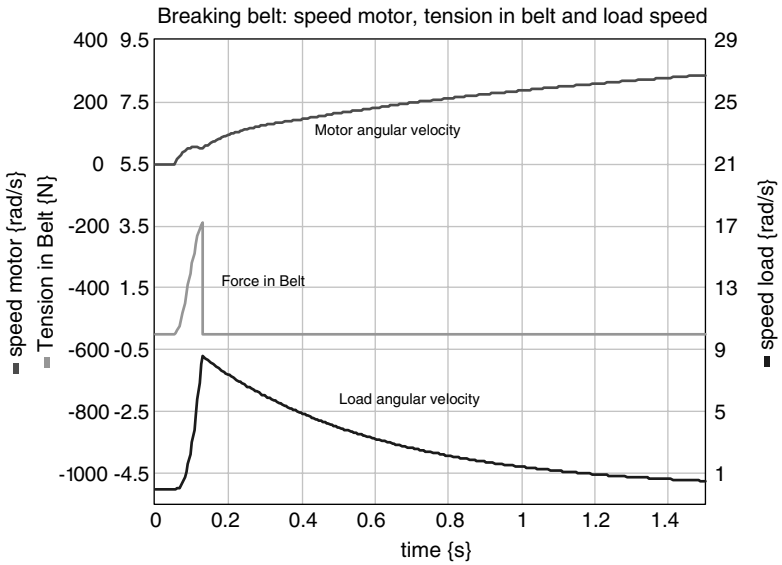


Fig. 18. Breaking of the belt.

This is shown in Fig. 18. Notice that because of the use of a physical model, the force signal of the belt is easily available.

Using a path generator can prevent excessive forces in the belt. In its basic form the path generator smoothes the reference step and filters out frequencies that excite the resonance frequencies. Further improvement is possible by adding zeros to the path generator that suppress those frequencies explicitly. The optimal situation with this configuration is achieved when an optimisation algorithm explicitly minimizes the forces in the belt by tuning the location of the zeros. The model used for optimisation and the forces in the belt before and after optimisation are shown in Figs. 19 and 20.

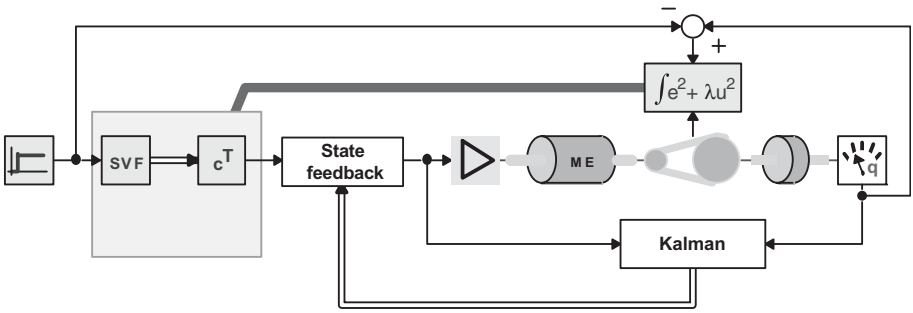


Fig. 19. Structure used for optimisation.

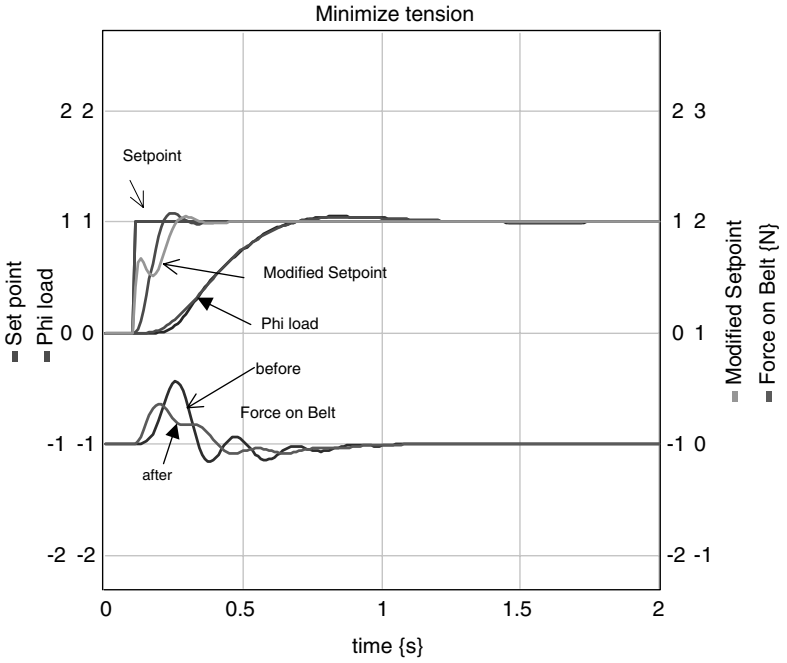


Fig. 20. Simulation results before and after optimisation.

## 4. Design of a mobile robot

A typical example of the early design procedure is the conceptual design of a mobile assembly robot [10]. Such a robot should be able to collect parts all around a production facility and do the assembly while driving. Because a high accuracy is required between the gripper of the robot and the surface where the parts are located, it is important that floor irregularities and vibration modes of the structure do

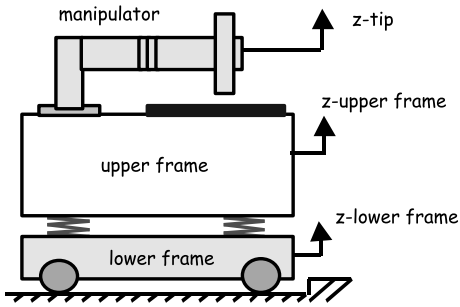


Fig. 21. Conceptual design of the mobile robot.

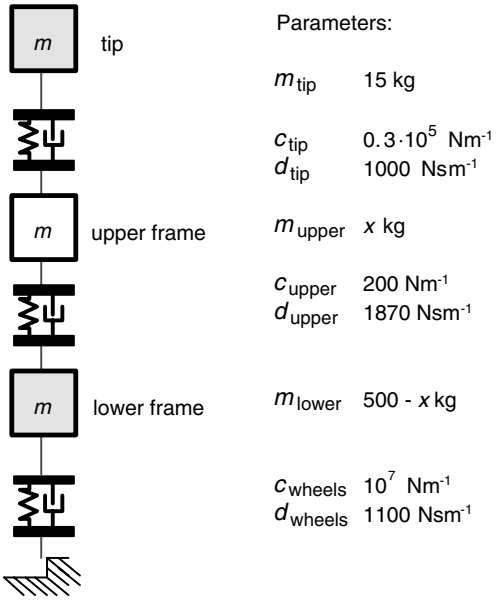


Fig. 22. Simple model with ideal physical elements to compute the error $e_{tip}$.

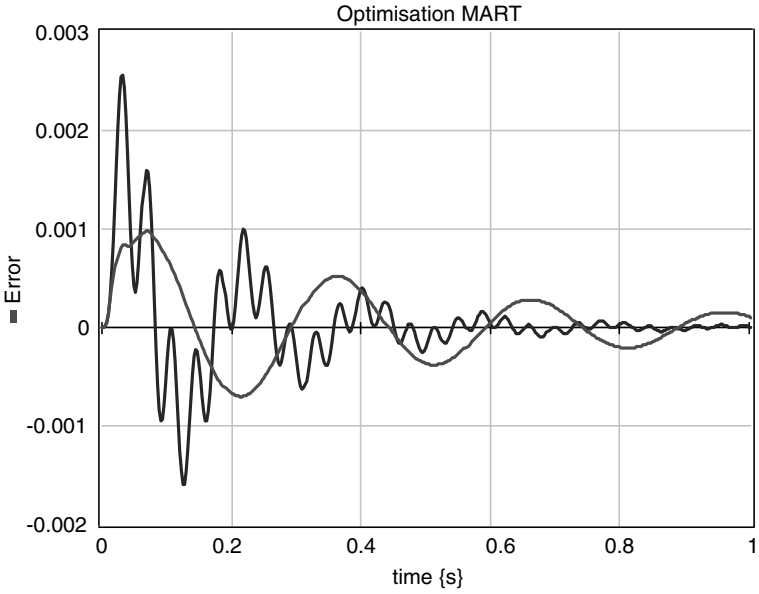


Fig. 23. Error of the tip before and after optimisation of the weight distribution between upper and lower frame.

not prevent proper assembly. The dead reckoning used for the navigation requires that the wheels be very stiff. Damping of disturbances has to be realised by another

means of suspension. This has led to the concept of an upper frame and a lower frame, connected by means of springs (Fig. 21).

The robot can be mounted at the upper frame and should have sufficiently bandwidth such that the position error ($e_{\text{tip}} = z_{\text{tip}} - z_{\text{upper frame}}$) between the tip of the robot ($z_{\text{tip}}$) and the upper frame ($z_{\text{upper frame}}$) is small enough.

The next step is to derive a simple model, in order to have some parameters for the weight distribution and the stiffness and damping of the springs. In the model of Fig. 22 the robot is confronted with a bump in the floor at a speed of 1 m/s.

Based upon the payload—mainly the weight of the batteries—the total mass of the vehicle was estimated to be 500 kg. Stiffness and damping of the wheels follow from the demands for the accuracy of the position estimation. The mass and bandwidth of the controlled manipulator were already known from other studies, yielding the effective stiffness and damping for the robot tip. When also initial estimates of the stiffness and damping of the springs between the upper and lower frame

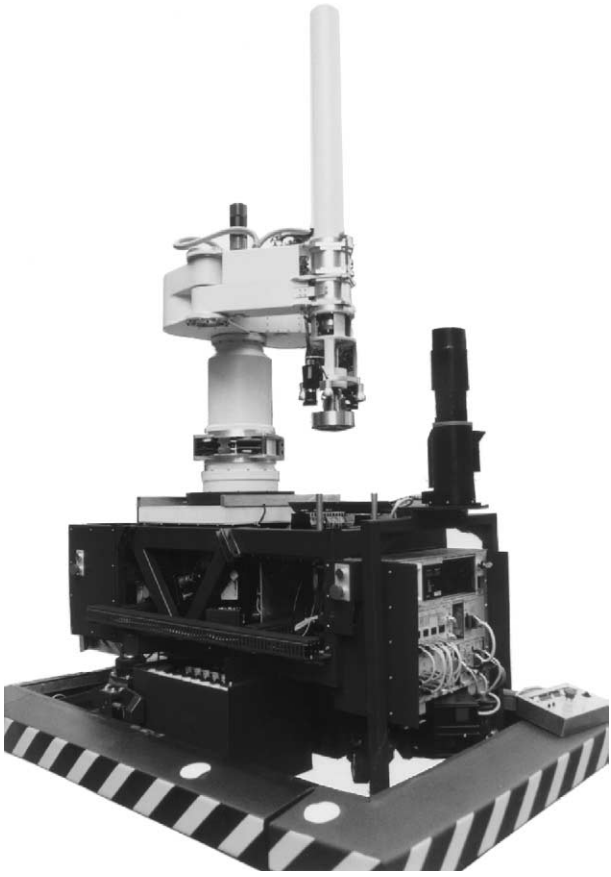

Fig. 24. The mobile robot after completion.

are made, the only parameter to be varied is the weight distribution between the upper frame and lower frame. By using the optimisation feature of 20-sim, the optimal weight distribution can easily be found. In order to minimize the error between the tip of the robot and the upper frame, the weight has to be placed as much as possible in the upper frame (Fig. 23).

A next step could be to optimise the properties of the suspension between upper and lower frame. This will further improve the error. This decision made in a very early stage of the design directed other design decisions. After completion of the project it appeared that the different parameters of the final construction were close to these early estimates (Fig. 24).

## 5. Learning feed-forward control

Many mechanical systems suffer from non-linear effects that limit the accuracy that can be achieved. Friction and cogging are two examples. A (linear) feedback controller can diminish the influence of non-linearities, but complete compensation may be difficult. For systems that perform repetitive motions, an iterative learning controller can help to further improve the performance [11]. The basic idea is explained in Fig. 25.

When only the feedback loop is present and under the assumption that there are no disturbances, the error signal and thus the controller signal $U_C$ will be the same for each repetitive motion. It is obvious that the accuracy can be improved when in the next motion the controller signal from the former cycle is used as a feed-forward signal, $U_F$. The feedback will generate a signal that further compensates for the remaining error by updating the feed-forward signal $U_F$ with the formula:

$$U_F^{k+1} = U_F^k + LE^k \tag{10}$$

where $L$ is the transfer function of the learning filter. The superscript $k$ denotes the $k$th repetitive motion. The signal $U_F$ should converge to a feed-forward signal that compensates for all repetitive errors. An example of a situation where such errors are present is, for instance, a CD-player that has to compensate for the eccentricity of the disk.

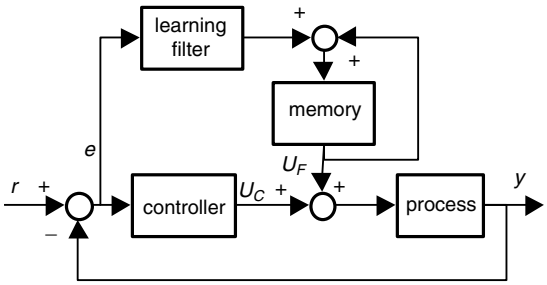


Fig. 25. Principle of iterative learning control.

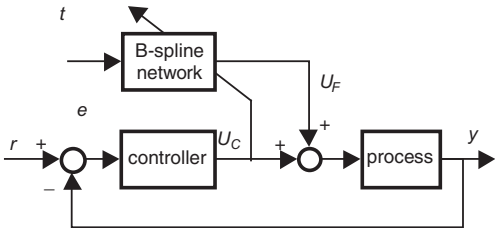


Fig. 26. Learning feed-forward controller for repetitive motions.

A variation on this idea and even more straightforward is the learning feed-forward controller (LFFC) set up of Fig. 26.

When the feed-forward signal would be perfect, the output of the controller would be zero. This implies that this output can be used as a training signal for the B-spline network. Using an adaptive B-spline network as a feed-forward network, enables that complex non-linear characteristics can be learned. The input of the B-spline network is the time $t$ that is reset each time a new motion starts. This is called a time-indexed LFFC. Instead of the time, also the reference signal and its derivatives—obtained from a path generator—could be used as index for the network (path-indexed LFFC). The advantage of this structure is that after proper training the LFFC can successfully be used for non-repetitive motions as well. Velthuis [12] has given a stability analysis for time-indexed as well as path-indexed LFFC-controllers. The stability analysis is relatively easy for the time-indexed case. For the path-indexed case it is more complex and some heuristics are required to guarantee a stable system. The main issue is that the number of B-splines should not be too large. On the other hand a sufficiently dense B-spline distribution is desired for an accurate approximation of the non-linear process. In cooperation with Philips LFFC has successfully been applied to compensate for cogging in an industrial Linear Motor [13]. In cooperation with Fokker Control Systems, it has been applied to compensation of (Coulomb) friction of a linear motor used in a flight simulator [12]. It has also been applied to the tracking control of the Mobile robot of Section 4 [14]. The application to cogging compensation of the linear motor will be described a little bit more into detail. Fig. 27 shows the set up.

A linear motor is controlled by means of a PID-controller, while a B-spline neural network is present to learn the inverse motor model, including the non-linearity due to cogging. The latter is the effect that DC-motors with permanent magnets are subject to more or less sinusoidally shaped varying forces that depend on the position of the translator with respect to the stator. If these forces really had a sinusoidal shape, they would be easily to compensate for by means of a feed-forward compensator. However, this would require magnets with very similar magnetic properties and accurate spacing of the magnets. Both solutions are expensive. An alternative is to design a controller that learns the disturbance pattern and compensates it by means of a learning feed-forward compensator. An additional advantage is that such a system can also be used to compensate for other non-linear effects, such as friction.

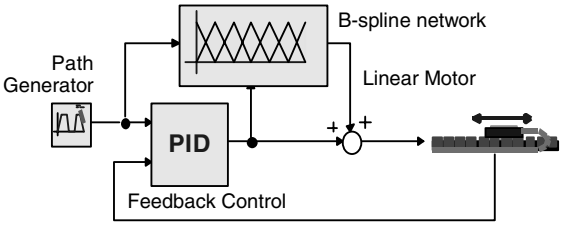


Fig. 27. Compensation of cogging with LFFC.

This has also been demonstrated in a part of a flight simulator (a control stick) where friction forces spoil the feeling of a realistic simulation especially at almost zero speed.

From Fig. 28 it can be seen that learning is almost completed after six training cycles. Zooming in at the first 10 seconds of Fig. 28 reveals the typical sinusoidal shape of the error due to the not yet properly compensated cogging (Fig. 29).

From these experiments it can be concluded that LFFC is an attractive method to compensate for non-linearities that are present in mechatronic systems, such as cogging and friction. The use of B-spline neural networks results in fast convergence, relatively low computational effort and a good generalising ability [12]. Because of recently obtained results with respect to the stability of such systems, robust control systems can be designed.

A mechatronic view on this design problem raises the question whether it is possible to use the same techniques to build a less expensive linear motor, when maximum accuracy is not the main goal of the design. This has been investigated in a recent research project [15]. Similar accuracy as in presently produced motors, but with less expensive components and less demanding assembly specifications can be achieved.

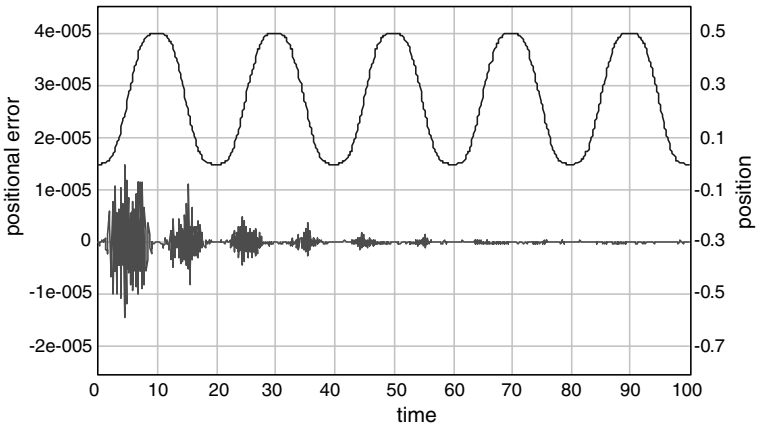


Fig. 28. Position and error signal during learning of the LFFC.

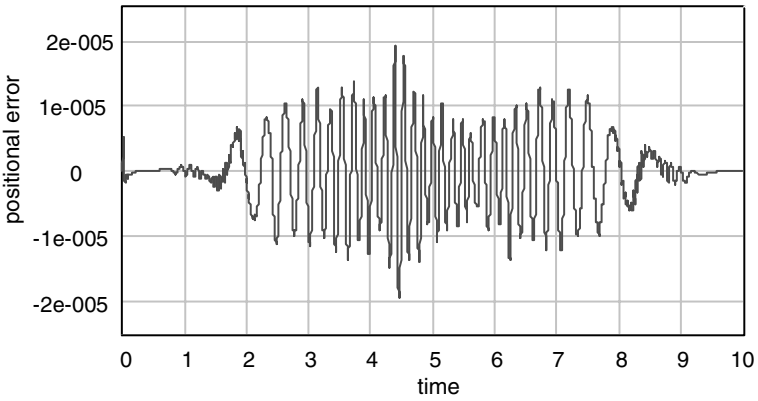


Fig. 29. The first 10 seconds show the sinusoidal character of the error due to cogging.

## 6. Conclusions

This paper has discussed the need for an integrated approach during the design of the mechanical and control parts of a mechatronic system. It requires models that besides the parameters of the controller, maintain all relevant physical parameters of the construction itself. A few examples showed how simulation software that is based on physical system models, such as 20-sim, could help to find proper parameters of mechanical components and of the controller. The example of learning feed-forward control showed how a learning controller based on a B-spline network could reduce non-linear effects such as cogging. It was also mentioned that solutions for achieving certain accuracy could be achieved by improving the mechanical part of the system, but also by means of improved control.

A demonstration version of the latest version of the 20-sim software used in this paper can be obtained from www.20sim.com. [1] Some of the models used in this paper are available at www.ce.utwente.nl/amn/JournalMechatronics. These models require 20-sim version 3.3.

## References

[1] Groenhuis H. A design tool for electromechanical servo systems. PhD thesis, University of Twente, 1991.
[2] Coelingh HJ. Design support for motion control systems. PhD thesis, University of Twente, 2000, also http://www.ce.utwente.nl/clh/.
[3] Coelingh HJ, de Vries TJA, van Amerongen J. Design support for motion control systems application to the Philips fast component mounter. In: Mechatronics Forum 7th International Conference, Mechatronics 2000, Atlanta, GA, USA.
[4] van Amerongen J, Coelingh HJ, de Vries TJA. Computer support for mechatronic control system design. Rob Auton Syst 2000;30(3):249–60. Available from: S0921-8890(99)00090-1.
[5] Broenink JF. Computer-aided physical-systems modeling and simulation: a bond-graph approach. PhD thesis, University of Twente, 1990.

[6] Weustink PBT, de Vries TJA, Breedveld PC. Object oriented modeling and simulation of mechatronic systems with 20-sim 3.0. In: Adolfson J, Karlsén J, editors. Mechatronics 98. Elsevier Science Limited; 1998.
[7] de Vries TJA. Conceptual design of controlled electro-mechanical systems. PhD thesis, University of Twente, 1994.
[8] Breedveld PC. Fundamentals of bond graphs. In: IMACS Annals of Computing and Applied Mathematics, vol. 3: Modelling and Simulation of Systems, Basel, 1989. p. 7–14.
[9] Gawthrop P, Lorcan Smith L. Metamodelling: bond graphs and dynamic systems. Prentice Hall; 1996.
[10] Oelen W. Modeling as a tool for design of mechatronic systems—design and realization of the Mobile Autonomous Robot Twente. PhD thesis, University of Twente, 1996.
[11] Arimoto S. A brief history of iterative learning control. In: Iterative learning control: analysis, design, integration and applications. Kluwer Academic Publishers; 1988. p. 3–7.
[12] Velthuis WJR. Learning feed-forward control—theory, design and applications. PhD thesis, University of Twente, 2000, also http://www.ce.utwente.nl/vts/.
[13] Otten G, de Vries TJA, van Amerongen J, Rankers AM, Gaal E. Linear motor motion control using a learning feedforward controller. IEEE/ASME Trans Mech 1997;2(3):179–87, ISSN 1083-4435.
[14] Starrenburg JG, van Luenen WTC, Oelen W, van Amerongen J. Learning feedforward controller for a mobile robot vehicle. Control Eng Practice 1996;4(9):1221–30.
[15] Kruif BJ, de Vries TJA. Improving price/performance ratio of a linear motor by means of learning control. In: 2nd IFAC Conference on Mechatronic Systems, Berkeley, CA, USA, December 9–12, 2002. p. 501–6.