MECHATRONIC DESIGN

Job van Amerongen

Drebbel Research Institute for Systems Engineering and
Control Laboratory, Faculty of Electrical Engineering,
University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands
J.vanAmerongen@el.utwente.nl, http://www.rt.el.utwente.nl/amn

Abstract: Mechatronic design requires that a mechanical system and its control system be designed as an integrated system. In order to make proper choices early in the design stage, tools are required that support modeling and simulation of physical systems -together with controllers- with parameters that are directly related to the real-world system. The 20-sim software, developed at the Control Laboratory of the Drebbel Institute is such a tool. 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

Mechatronics deals with the design of controlled mechanical systems. This implies that during the design phase the designer should be able to evaluate changes of the mechanical part as well as of the (electronic or 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 controller. Although control engineers have promoted the idea of such a systems approach for a long time, it is in mechatronics that many systematic attempts are being made to achieve this. This has lead to the availability of inexpensive, high-quality consumer products, such as CD- and DVD-players, video recorders etc. Also more professional equipment can benefit from a mechatronics design approach because this leads to more flexible, cheaper 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 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, present in many mechatronic systems, such as friction and cogging.

2. DESIGN SOFTWARE

During the design of mechatronic systems it is important that changes in the construction and the controller can be evaluated simultaneously. Although a proper controller will enable a cheaper construction to be built, 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 about the mechanical properties needed to achieve a good performance of the controlled system. This requires that already in an early stage of the design a simple model is available, that is able to make clear what the performance

limiting factors of the system are. Of course this can only be done when also a sufficiently sophisticated controller is designed for this conceptual model. The controller design includes the choice of the 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 because of the resonant poles proper selection of the signals to be used in the feedback is essential. Efforts have been made (Groenhuis 1991; Coelingh 2000a, 2000b) 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 (van Amerongen et al., 2000).

2.1 Simulation of mechatronic systems

Simulation is an important tool to evaluate the design of (control) systems. But not all simulation programs support physical modeling 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. There are programs that support modeling of *physical systems* in *particular domains*. Examples are PSPICE (electronic networks), ADAMS (mechanical systems) and SpeedUp (chemical processes). Recently also programs that allow physical modeling in *various physical domains* became available. They use an object-oriented approach that allows hierarchical modeling 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 (1990) as CAMAS and Dymola (1993).
In this paper version 3.1 of 20-sim (pronounce Twente Sim) will be used to illustrate the simultaneous design of construction and controller in a mechatronic system. 20-sim supports object-oriented modeling. Power and signal ports to and from the outside world determine each object (Weustink et al., 1998). Inside the object there can be other objects or, on the lowest level, equations. Other *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 modeling as well as bottom-up modeling. Modeling can start by a simple interconnection of (empty) sub models. Later they can be filled with realistic descriptions with various degrees of complexity. De Vries (1994) calls this polymorphic modeling. Sub models can be constructed from other sub models in hierarchical structures.
Modeling 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. The energy flow or *power*, *P*, is commonly chosen as the product of *two conjugated signals*, called effort (*e*) and flow (*f*):

$$P = ef \tag{1}$$

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

$$P = Fv, \text{ or } P = T\omega \tag{2}$$

$$P = ui \tag{3}$$

where F is force, v is velocity, T is torque, ω is angular velocity, u is voltage and i is current. *Bond graphs* (Breedveld, 1989, Gawthrop and Smith, 1996) are a graphical tool to represent such systems. They gain popularity in the control community as well. By using iconic diagrams the bond graph can be hidden for users not used to this representation. In this way the tool becomes available to a much wider class of users. Version 3.1 of 20-sim allows for model descriptions in the form of iconic diagrams, bond graphs, block diagrams and equations. Its use will be illustrated with a few examples.

3. DESIGN OF A SERVO SYSTEM

Coelingh (2000) and Coelingh et al. (2000) 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 a load driven by an electric motor, through a flexible transmission. An iconic diagram of this model is given in Fig. 1.

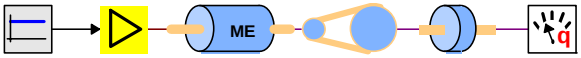

Fig. 1. Iconic diagram of a system consisting of a current amplifier, DC-motor, flexible transmission and load

Because of the flexible transmission, by means of a rubber belt, the resonance in this system is clearly visible in the step response (Fig. 2.).

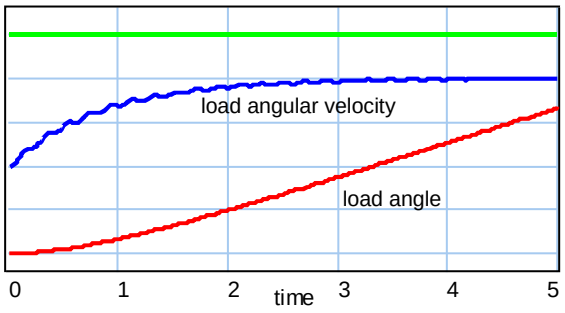

Fig. 2. Open loop responses

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. These models can be exported to Matlab, where they can be used, e.g. to compute an LQR or LQG controller. The Matlab results can easily be imported again in 20-sim. The diagram of the process together with an LQG-controller (Kalman filter and state feedback) is given in Fig. 3. and responses in Fig. 4.

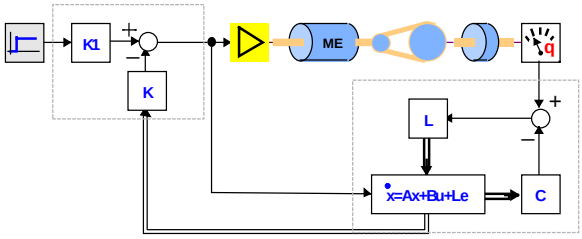

Fig. 3. Process with Kalman filter and state feedback

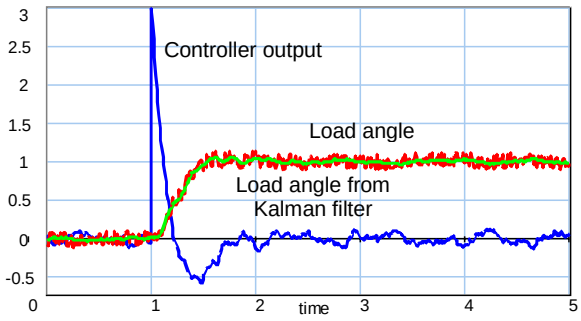

Fig. 4. Response of the LQG-controlled system

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. This is shown in Fig. 5. Notice that because of the use of a physical model, the force signal is easily available.

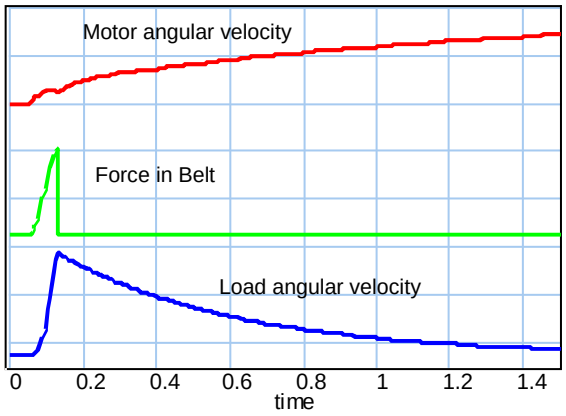

Fig. 5. Breaking of the belt

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 optimization algorithm explicitly minimizes the forces in the belt by tuning the location of the zeros. The model used for optimization and the forces in the belt before and after optimization are shown in Fig. 6. and Fig. 7.

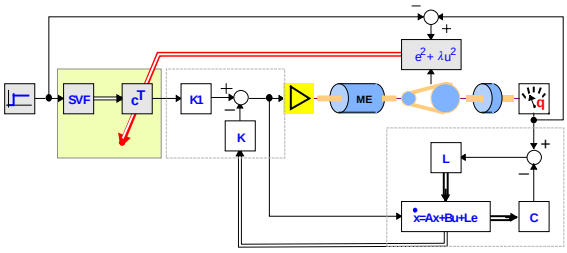

Fig. 6. Structure used for optimization

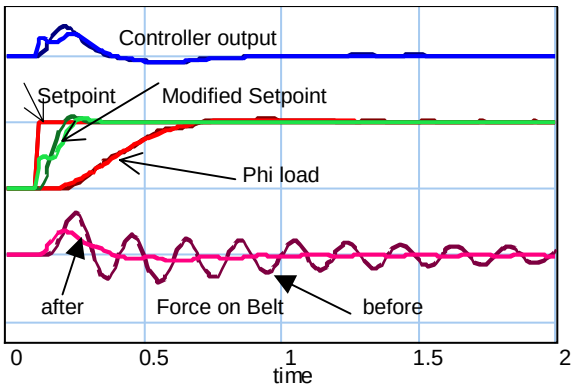

Fig. 7. Simulation results before and after optimization

4. DESIGN OF A MOBILE ROBOT

A typical example of the early design procedure is the conceptual design of a mobile assembly robot. 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

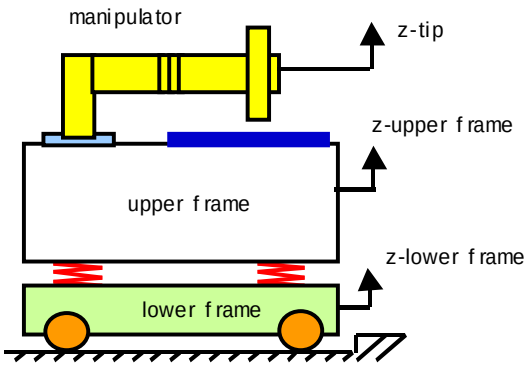

Fig.8. Conceptual design of the mobile robot

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 not prevent proper assembly. The dead reckoning used for the navigation requires that the wheels be very stiff. Damping of disturbances has to be realized 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 8.). The robot can be mounted at the upper frame and should have sufficiently bandwidth such that the position error (e_{tip}) between the tip of the robot (z_{tip}) and the upper frame ($z_{upper\ frame}$) is small enough.
The next step is to derive a simple (2D-) model, in order to have some parameters for the weight distribution and the stiffness and damping of the springs. In the model of Fig. 9. the robot is confronted with a bump in the floor at a speed of 1m/s.

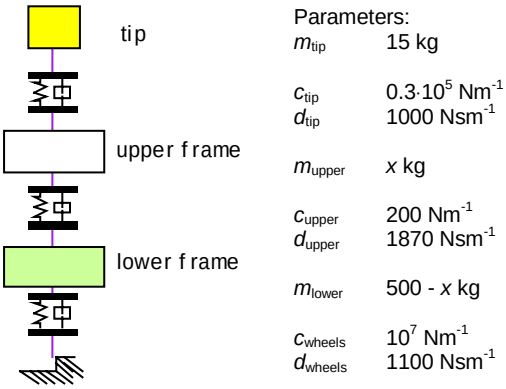

Fig. 9. Simple model with ideal physical elements to compute the error e_{tip}

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 are made, the only parameter to be varied is the weight distribution between the upper frame and lower frame. By using the optimization 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.

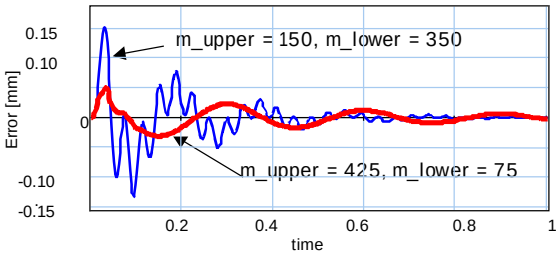

Fig. 10. Error of the tip before and after optimization of the weight distribution between upper and lower frame

A next step could be to optimize 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. 11.).

Fig. 11. The mobile robot after completion

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 (Arimoto, 1988). The basic idea is explained in Fig. 12.

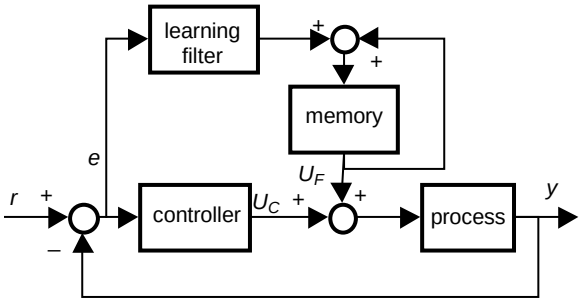

Fig. 12. Principle of Iterative Learning Control

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:

$$U_F^{k+1} = U_F^k + LE^k \qquad (4)$$

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.

A variation on this idea and even more straightforward is the Learning Feed-Forward Controller (*LFFC*) set up of Fig. 13.

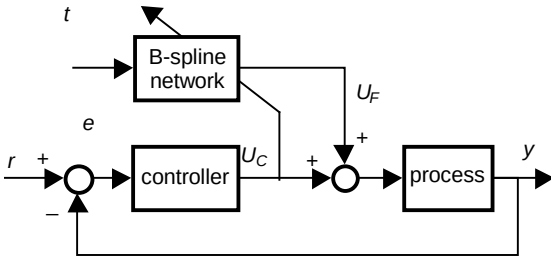

Fig. 13. Learning Feed-Forward Controller for repetitive motions

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. The use of 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 (2000) 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 (Otten et al., 1997)). In cooperation with Fokker Control Systems, it has been applied to compensation of (Coulomb) friction of a linear motor used in a flight simulator (Velthuis, 2000). It has also been applied to the tracking control of the Mobile robot of Section 4 (Starrenburg et al. 1996). The application to cogging compensation of the linear motor will be described a little bit more into detail. Fig. 14. shows the set up.

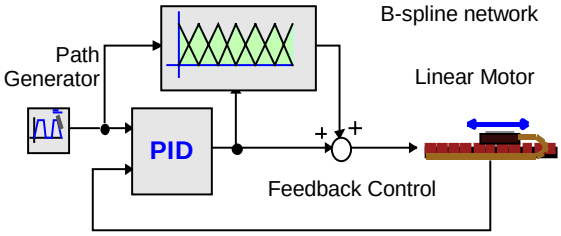

Fig. 14. Compensation of cogging with *LFFC*

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. 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.

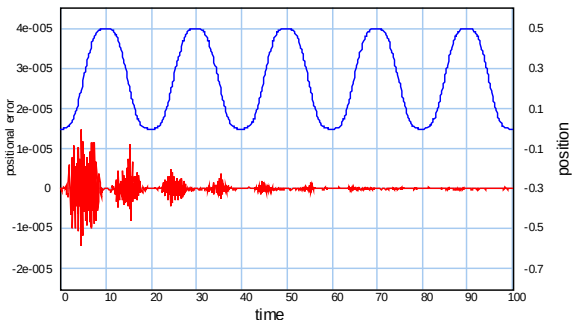

Fig. 15. Position and error signal during learning of the *LFFC*

From Fig. 15. it can be seen that learning is almost completed after 6 training cycles. Zooming in at the first ten seconds of Fig. 15. reveals the typical sinusoidal shape of the error due to the not yet properly compensated cogging (Fig. 16.).

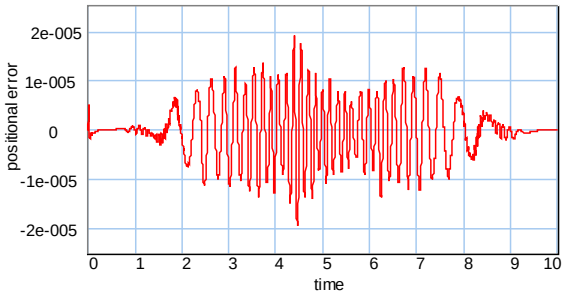

Fig. 16. The first ten seconds show the sinusoidal character of the error due to cogging

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 generalizing ability (Velthuis, 2000). Because of recently obtained results with respect to the stability of such systems, robust control systems can be designed.

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. A few examples showed how simulation software such as 20-sim could help to find proper parameters of mechanical components and of the controller. The third example showed how a learning controller based on a B-spline network could reduce non-linear effects such as cogging. A demonstration version of the software used in this paper can be obtained from www.20sim.com. The models used in this paper are available at www.rt.el.utwente.nl/amn/atlanta.

7. REFERENCES

Amerongen, J. van, H.J.Coelingh, T.J.A. de Vries, "Computer support for mechatronic control system design", Robotics and Autonomous Systems, vol. 30, nr. 3, pp. 249 - 260, PII: SO921-8890(99)00090-1, 2000

Arimoto, S, A Brief History of Iterative Learning Control. In: *Iterative Learning Control: Analysis, Design, Integration and Applications*, Kluwer Academic Publishers, pp. 3-7, 1988

Breedveld, P.C., Fundamentals of Bond Graphs, in *IMACS Annals of Computing and Applied Mathematics, Vol. 3: Modelling and Simulation of Systems*, Basel, 1989, pp. 7-14

Broenink, J.F., *Computer-aided physical-systems modeling and simulation: a bond-graph approach*, Ph.D.-thesis, University of Twente, 1990

Coelingh, H.J., *Design Support for Motion Control Systems*, Ph.D.-thesis, University of Twente, 2000, also http://www.rt.el.utwente.nl/clh/

Coelingh, H.J., T.J.A. de Vries, J. van Amerongen, Design Support for Motion Control Systems -Application to the Philips Fast Component Mounter, this conference, 2000

Gawthrop, P. and Lorcan Smith, L., *Metamodelling: Bond Graphs and Dynamic Systems*, Prentice Hall, 1996

Groenhuis, *A design tool for electromechanical servo systems*, Ph.D.-thesis, University of Twente, 1991

Oelen, W., *Modeling as a tool for design of mechatronic systems – design and realization of the Mobile Autonomous Robot Twente* Ph.D.-thesis, University of Twente, 1996

Otten, G. T.J.A. de Vries, J. van Amerongen, A.M. Rankers and E. Gaal, Linear motor motion control using a learning feedforward controller, IEEE/ASME transactions on mechatronics, vol. 2 nr: 3, ISSN 1083-4435, pp. 179-187, 1997

Starrenburg, J.G., W.T.C. van Luenen, W. Oelen and J. van Amerongen, Learning Feedforward Controller for a Mobile Robot Vehicle, Control Engineering Practice, Vol. 4, No. 9, pp.1221-1230, 1996

Velthuis, W.J.R., *Learning Feed-Forward Control – Theory, Design and Applications*, Ph.D.-thesis, University of Twente, 2000, also http://www.rt.el.utwente.nl/vts/

Vries, T.J.A. de, *Conceptual design of controlled electro-mechanical systems*, Ph.D.-thesis, University of Twente, 1994

Weustink, P.B.T., de Vries, T.J.A. and P.C. Breedveld, Object Oriented Modeling and Simulation of Mechatronic Systems with 20-sim 3.0, in *Mechatronics 98, J. Adolfson and J. Karlsén (Editors)*, Elsevier Science Limited, 1998