Modelling, Simulation and Controller Design for Mechatronic Systems with 20-sim 3.0

Job van Amerongen

Cornelis J. Drebbel Research Institute for Systems Engineering and
Control Laboratory, Electrical Engineering Department, University of Twente,
P.O. Box 217, 7500 AE Enschede, Netherlands

e-mail: J.vanAmerongen@el.utwente.nl, http://www.el.utwente.nl/amn

Modelling, Simulation and Controller Design for Mechatronic Systems with 20-sim 3.0

Job van Amerongen

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

Abstract: This paper discusses the use of the modelling and simulation program 20-sim for the analysis and design of mechatronic systems. Control engineers traditionally use transfer functions, block diagrams or state space descriptions. In mechatronics, where a controlled system as a whole has to be designed, it is advantageous that model parameters are directly related to physical components. In addition it is desired that models be reusable. Block-diagram-based simulation packages hardly support these features. 20-sim allows input of models in the form of equations, block diagrams, bond graphs and iconic diagrams. This will be illustrated by means of the design of a system that is representative for many mechatronic systems.

Keywords: mechatronics, modeling, simulation, controller design, CACSD

1 Introduction

Simulation is an important tool to evaluate the design of (control) systems. Originally computer simulations required conventional programming. One of the early programs especially designed for simulation was THTSIM (later TUTSIM), developed by the University of Twente in the 1970's. Tools to enter block diagrams graphically became available in programs as EASY5 (1976), Simulink (1991) and Vissim (1990). 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), originally known as CAMAS (1990) [15] and Dymola (1993).

This paper will illustrate some new features of version 3.0 of 20-sim (pronounce Twente Sim). 20-sim supports object-oriented modelling: each *object* is determined by equations and power and signal ports to and from the outside world [12]. Other *realisations* 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. Later they can be filled with realistic descriptions with various degrees of complexity (models can be polymorphic [16]). 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 express the dynamic properties of the system in different domains. This is achieved by coupling models by means of the *flow of energy* from one sub model to the other, rather than by *signals* such as voltage, current, force and speed. The energy flow or *power*, P, is commonly chosen as a combination of *two conjugated signals*, called effort and flow: $P = ef$ (Table 1). *Bond graphs* are a graphical tool to represent such systems. They gain popularity in the control community as well. The name originates from the *power bonds* that connect the sub models. This paper will shortly introduce the use of bond graphs. Additional information is available in [14] and in the references [1]–[13].

TABLE 1 Effort and flow in electrical and mechanical domain

domain	Effort (e)	flow (f)
electrical	Voltage (V)	current (i)
mechanical	Force (F), torque (T)	(angular) velocity v (ω)

2 Modelling

2.1. Top down modelling

In the design stage modelling of a mechatronic system starts with describing the components of the system. In practice mostly a combination of top-down and bottom-up modelling is used. 20-sim allows the use of components from a library, enabling fast design of new systems. Here we will start from scratch and consider as an example the design of a small servo system to be used as a laboratory experiment. The conceptual idea is given in figure 1. The system consists of a load, driven by a DC motor through a transmission. In addition, a power amplifier and a potentiometer to read the load angle are needed.

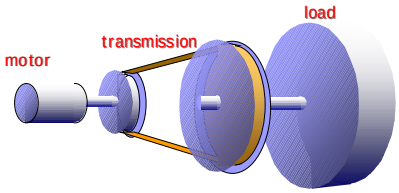

Figure 1 Concept of the laboratory set up

Modelling starts by entering a number of empty sub models (figure 2).

Figure 2 Empty sub models

Models filled with descriptions in the form of a block diagram cannot be connected without special precautions. Therefore, we consider the *energy flow* (*P*) between the sub models and connect them with so called *power bonds*. The sub models of the DC_motor (that converts electrical into mechanical power), the transmission and the load (both dealing with mechanical power) each get two power ports. The power amplifier and the potentiometer get one power port and one signal port. The latter form the interface between elements where power is dominant and elements where the information content of the signals is dominant. This allows that still empty sub models be connected to each other (figure 3a). With a little more effort we can edit the icons such they give a good idea about the contents sub model (figure 3b).

Figure 3a Sub models can be connected through ports

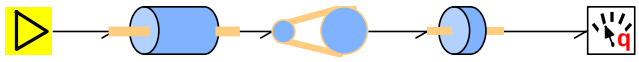

Figure 3b User definable icons make the model easier to interpret

2.2. Modelling of the DC-motor

Next we consider the contents of the different sub models. We construct an ideal physical model from ideal elements such as resistance and friction, inductance and inertia, etceteras. For the DC-motor we can draw the ideal physical model, or iconic diagram of figure 4. In the motor we have modelled the armature inductance and resistance, the transfer from electrical to mechanical energy, as well as the mechanical friction and inertia of the motor. An ideal voltage amplifier has been added to drive the motor.

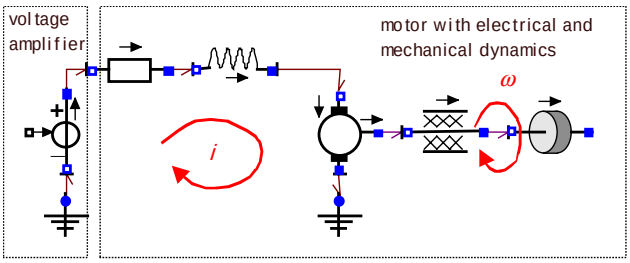

Figure 4 The component 'motor' decomposed into ideal physical elements

Figure 4 has been drawn from standard available ideal elements in the graphical editor of 20-sim. We can make a more abstract description in the form of a bond graph. Later, this will help with making proper choices in the modelling process.

All elements in the electrical part share the same current *i* (*flow*), all mechanical elements the same angular velocity ω (*flow*). In a bond graph *flows* are represented by a 1-junction. An **R**-element represents electrical resistance, an **I**-element inductance. The **GY**-element (gyrator) represents the transfer from electrical to mechanical power, an **MSe**-element a modulated voltage source. Half arrows indicate the direction of the positive energy flow. At a 1-junction the sum of all *efforts* is zero. In the mechanical domain an **R**-element represents friction and an **I**-element inertia. With these elements we can draw the bond graph of the motor (figure 5a). In fact the elements in figure 4 are sub models that hide bond graphs inside. This is made clear in figure 5a.

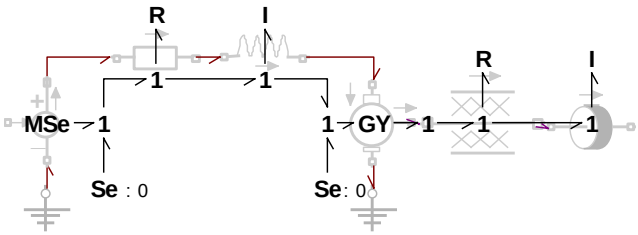

Figure 5a Bond graph representation superimposed on iconic diagram

After removing the iconic diagram we get figure 5b.

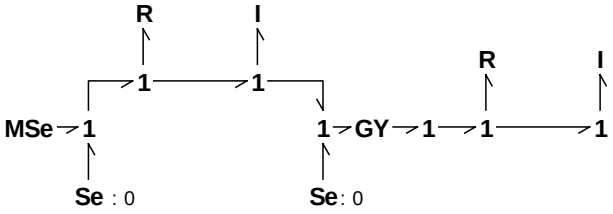

Figure 5b Bond graph of the motor

This bond graph can be simplified by means of some simple rules (figure 5c). 20-sim can do this automatically.

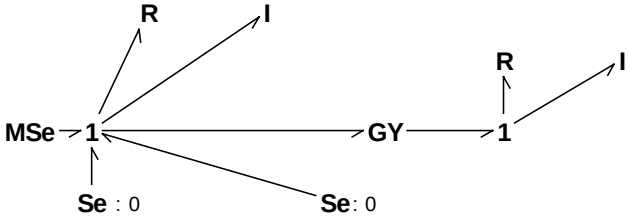

Figure 5c Bond graph of the motor after automatic simplification

The different elements can be rearranged to give the bond graph a better look. In addition, voltage sources with value zero, representing the reference ('grounds') can be removed. This yields the bond graph of figure 5d.

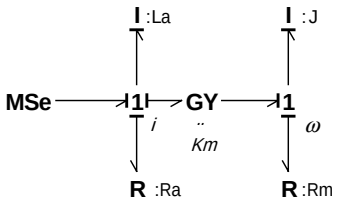

Figure 5d
Causal bond graph of a DC-motor

In figure 5d '*causal strokes*' at the begin or end of the half arrows indicate how the signals will be computed during simulation. A bond graph *without* causal strokes is a complete description of the dynamical behaviour of the underlying physical system, or a graphical representation of the *equations* that describe the system. At this point causal strokes seem only needed when we want to simulate the system by writing the equations as *assignment statements* or when we want to draw a block diagram. However, as we will see later, the causal strokes also give the modeller useful feedback about his modelling decisions. 20-sim automatically assigns the causality *during* the composition of a bond graph. If desired, the user can make the order of causal assignment visible.

From the iconic diagram of figure 4, or from the bond graph of figure 5d we can directly derive differential equations. At 1-junctions the sum of the efforts is zero:

$$\begin{aligned} U - L_a \frac{\mathrm{d}i}{\mathrm{d}t} - K_m \omega - R_a i = 0 \\ K_m i - J \frac{\mathrm{d}\omega}{\mathrm{d}t} - R_m \omega = 0 \end{aligned} \qquad (1)$$

To draw a block diagram an integral form is desirable:

$$\begin{aligned} i = \frac{1}{L_a} \int \left(U - K_m \omega - R_a i\right) \mathrm{d}t \\ \omega = \frac{1}{J} \int \left(K_m i - R_m \omega\right) \mathrm{d}t \end{aligned} \qquad (2)$$

Equations (2) and the block diagram of figure 6 correspond with the causal strokes in the bond graph.

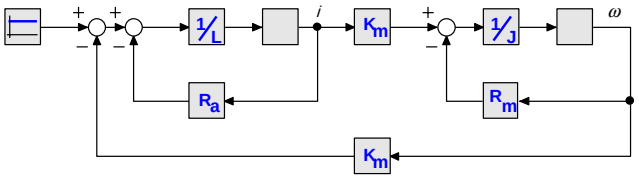

Figure 6 Block diagram of a DC-motor

Until so far, the whole modelling procedure was not very exciting. We have derived presentations of the DC-motor as iconic diagram, bond graph, several sets of equations and block diagram. All these forms can be entered as model representations in 20-sim (all figures where made with the 20-sim editors). Integral (eqn. (2)) as well as differential equations (eqn. (1)) can indeed be entered as equations (i.e. not as assignment statements). During generation of the simulation code 20-sim will automatically use the most appropriate form.

2.3. Modelling of the transmission and the load

We continue the modelling by adding the transmission and the load. The transmission is represented by a transformer, **TF**, ($\omega_2 = r\omega_1$, $T_1 = rT_2$), the load by another inertia and friction element. This leads to the bond graph of figure 8.

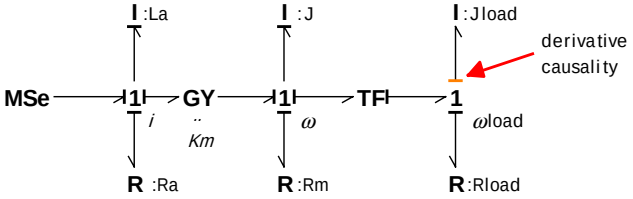

Figure 8 DC-motor plus transmission and load

When entering the additional elements in 20-sim we see that the causal stroke at the load inertia has got a red colour. In addition, it is at the other end of the bond than the causal strokes at the other I-elements. This indicates that this element can only be computed with *derivative causality*. This is not only a numerical problem: the model should be reconsidered because it has been over-simplified. Derivative causality indicates that the initial conditions of the two inertias are dependent of each other. They have been modelled as rigidly connected. In that case it makes sense to eliminate the transformer and combine the two inertias. Another option is to reconsider the model. A transmission by means of a belt is not a rigid connection. This can be modelled by adding flexibility (spring or compliance) in the form of a C-element in the model, allowing different speeds of the two inertias. The C-element is placed between two transformers representing the transformation from mechanical rotation to mechanical translation and vice versa (figure 9). The 'spring' has different velocities at both ends. This velocity difference is represented by a 1-junction, connected to a 0-junction (zero-junction). At a 0-junction the sum of all flows is zero. It represents effort.

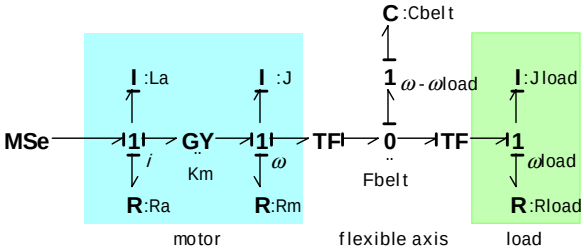

Figure 9 DC-motor plus flexible transmission and load

The C-element should not be added for numerical reasons only. If the two inertias are almost rigidly connected, it is better to combine them or to solve the causal problem numerically. A proper (but computationally more intensive) implicit integration algorithm in the simulator can solve the numerical problems. 20-sim indicates these options by means of the red causal stroke. The user can decide which option is most desirable to solve the problem.

When we translate the bond graph of figure 9 into equations and a block diagram, the advantages of using an energy-based approach become clear. The bond graph of figure 9 can directly be entered into 20-sim and is ready for simulation. Manually deriving equations and a block diagram requires the whole process of the former section to be repeated. There is only a limited reuse of earlier results. After eliminating the transformers, we find the following equations:

$$
\begin{aligned}
i &= \frac{1}{L_a}\Big(\big(U - K_m\omega_1 - R_a i\big)\mathrm{d}t \Big) \\
\omega &= \frac{1}{J}\Big(\big(K_m i - R_m\omega_1 - T_{spring}\big)\mathrm{d}t \Big) \\
T_{belt} &= \frac{1}{C'_{belt}}\Big(\big(\omega - \omega'_{load}\big)\mathrm{d}t \Big) \\
\omega'_{load} &= \frac{1}{J'_{load}}\Big(\big(T_{belt} - R_{load}\omega'_{load}\big)\mathrm{d}t \Big)
\end{aligned}
\qquad (3)
$$

These equations lead to the block diagram of figure 10. The different parts of this model are connected to each other in a rather complex way.

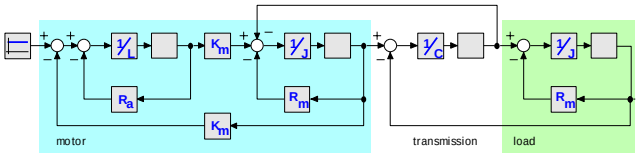

Figure 10 Block diagram of motor, transmission and load

It may be better to use a current amplifier rather than a voltage amplifier to steer the motor. In the bond graph of figure 11 this has been done. It leads to derivative causality of the inductance.

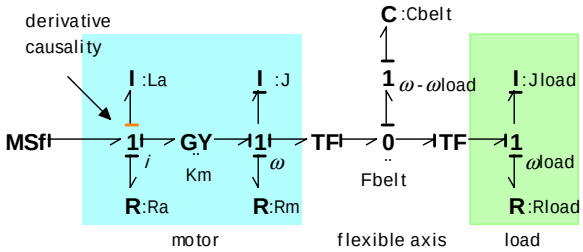

Figure 11 Bond graph of process with a current source

It indicates that we should model a current amplifier with internal resistance (1), remove the inductance from the model (2), or leave things as they are and use an implicit integration routine (3). The second solution is most reasonable. We are not interested in the high-frequency dynamics of the electrical circuit and the current amplifier was applied to eliminate the electrical time constant.

2.4. Top-down modelling of the complete system

With the different parts of the bond graph of figure 11 we can fill the still empty sub models of figure 3. This is shown in figure 12. Figure 12 also shows the hierarchical structure of the model. Each sub model contains other sub models (iconic diagram, bond graph or equation). The number of levels is only limited by the memory of the computer. At the lowest level there is always an equation (see the sub model of the friction of the load). The (signal) input of the current source allows connection to a signal generator or a controller. The potentiometer integrates the angular velocity and adds noise to the measurements.

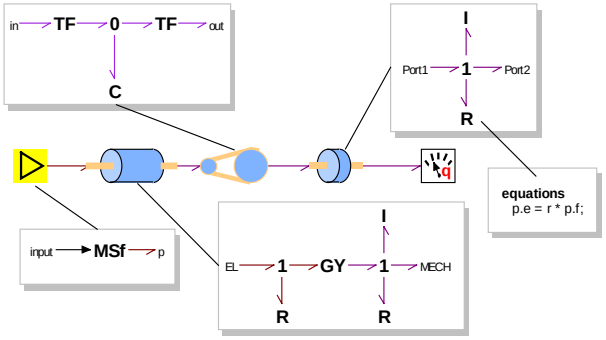

Figure 12 Model of DC-motor after filling the sub models

3 Controller design

3.1. Generating a state space model

Having a proper model available, we want to design an LQG-controller with Matlab. This requires a description of the system in state space form. This can be automatically generated in 20-sim. We extend the model of figure 12 with a signal generator (figure 13), start the simulator and choose parameters as present in a real experimental system (R_{load} = 0.1152 Nms, J_{load} = 0.056 kg m^2 ,n_{TF1} = 0.25, n_{TF2} = 1, C_{belt} =0.648 N^{-1}m, K_{motor} = 0.292 , R_{motor} = 0.0001 Nms, J_{motor} = 0.00262 kg m^2). Figure 14 gives the open-loop responses.

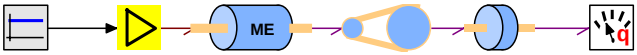

Figure 13 DC-motor with signal generator

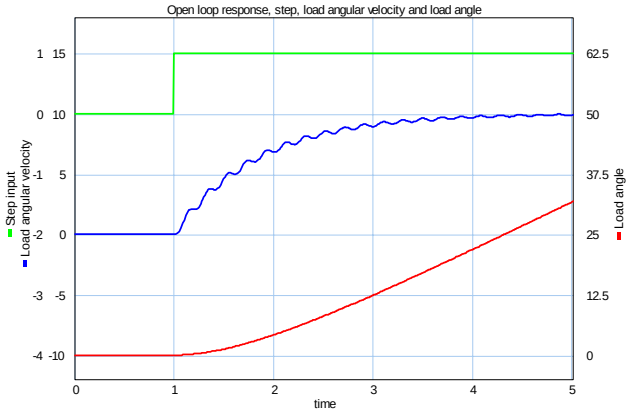

Figure 14 Open loop response (input, load velocity and load position)

The menu option *linearise model* yields a state-space description of the model, both, as a 20-sim model and as a Matlab m-file. Running the m-file in Matlab shows the transfer function of the model and a bode plot.

3.2. *Design of an LQG-controller*

In Matlab optimal controller gains of a state feedback controller as well as gains of a Kalman filter can be computed. State-feedback controllers as well as a Kalman filters are available as sub models in the State Space library of 20-sim. In figure 15 these elements have been added. The internal structure of the Kalman filter and the state-feedback controller, shown within the dashed lines, is just a complex icon. Inside the sub models are equations.

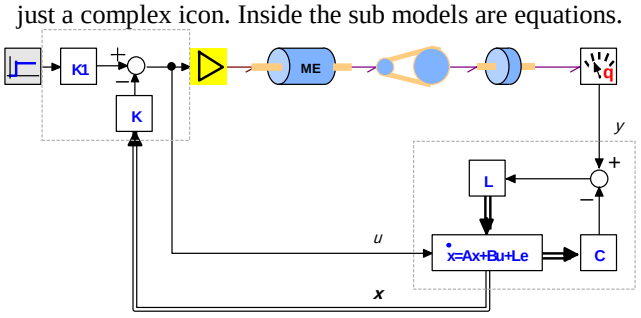

Figure 15 Process with state-feedback controller and Kalman filter

3.3. *Time events*

By adding an AD- and DA-converter the controller can also be implemented in digital form. Because 20-sim 3.0 supports time events, during simulation the samples will be accurately taken at the right moments. If the sampling instants do not coincide with the integration intervals, extra computations are automatically added. Figure16 shows the discrete system and figure 17 simulated responses.

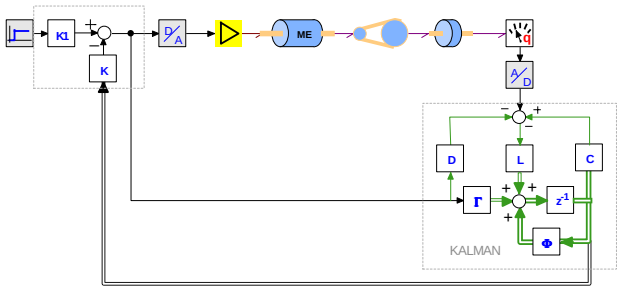

Figure 16 Process with discrete Kalman filter and discrete state-feedback controller

The Kalman filter and the state feedback controller can also be combined into one LQG-controller. It makes the diagram simpler but also less informative (figure 18).

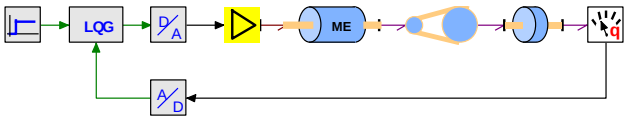

Figure 18 Process with discrete LQG-controller

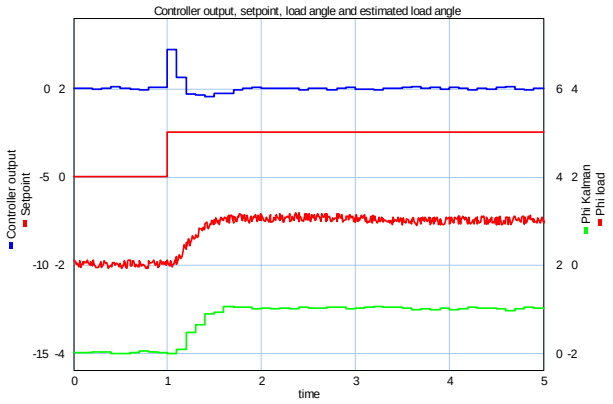

Figure 17 Responses of process with discrete Kalman filter and discrete state-feedback controller

3.4. *State events*

Time events happen at fixed time intervals. State events depend upon the states of the system. They can occur any moment. As an example we consider a transmission belt with limited strength. A too high tension will cause breaking of the belt. Because 20-sim supports state events, the moment the belt breaks will be accurately computed (figure 18). Sometimes animations are more informative than time responses. An animation of the experiment of figure 18 can be found at the www [14].

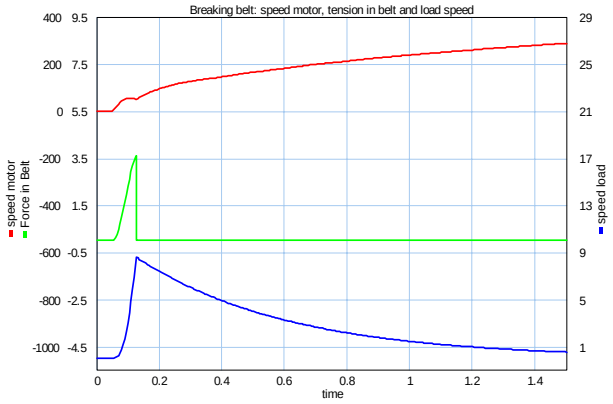

Figure 18 State event: breaking belt

3.5. *Optimisation*

The chance that the belt will break can be minimised, by minimising the variations in the tension in the belt. Modifying the set point of the control loop can prevent the resonance frequencies from being excited. A feed-forward network with complex zeros that cancel the resonant poles of the system is a possible solution (figure 19). The zeros of the network are tuned by means of an optimisation experiment. It minimises the tension in the belt, as well as the error signal between set point and angle of the load. The reduction of the forces in the belt before and after optimisation can be seen in figure 20.

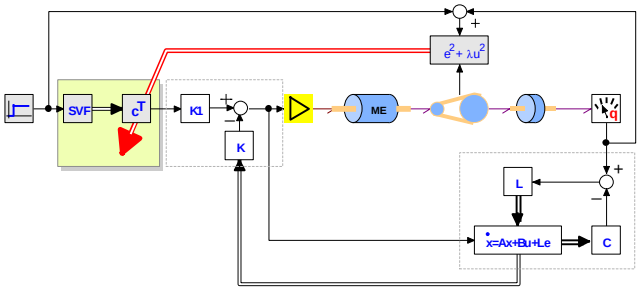

Figure 19 Optimization of the feed-forward compensator

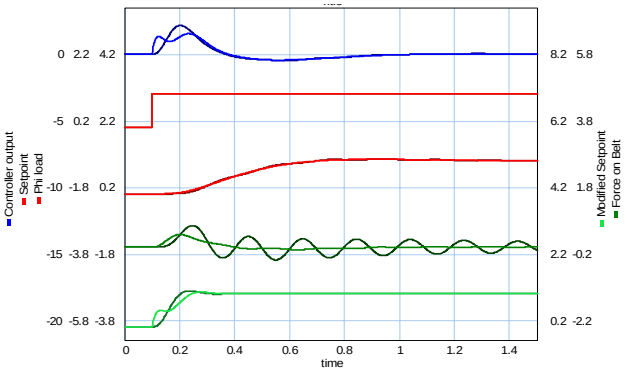

Figure 20 Proper tuning of the feed-forward controller reduces forces in the belt

4 Conclusions

In this paper the use of 20-sim for modelling and controller design of mechatronic systems has been demonstrated. The example of minimising the tension in the belt showed the advantage of having direct access to physically meaningful signals and parameters.

Because models can be entered as equations, bond graphs, block diagrams, iconic diagrams, or by any combination of these, appropriate views can be chosen for different users and for different design projects. The hierarchical structure of the models allows that details be hidden if desirable. Different realisations of a sub model can easily be exchanged, without the need to reconnect the sub model or to re-enter parameters. In an early stage of the design sub models can be kept simple to capture only the most important dynamics. Later models can be more precise and detailed.

The possibility to couple sub models by means of power ports is one of the most attractive features of 20-sim. It enables models to be re-used and connected to each other, without re-doing all the manual generation of equations. Sub models can be simple ideal elements or, for instance, a complete motor.

Output can be represented as *x-t* and *x-y* plots and as 3D-animations. The optimisation and multiple run facilities of the simulator allow parameter estimation, controller tuning or sensitivity analysis. The interface to Matlab make all its design and analysis tools easily available.

The above-mentioned features make 20-sim a tool for research as well as education. Iconic diagrams can help a novice to get used to the bond graph notation. The controller design examples presented here were given as an assignment to students in a course on digital control, optimisation and state estimation. Because 20-sim supports physical modelling in various domains it makes simulation of of real physical systems, including rather complex controllers easy.

A demonstration version of 20-sim can be downloaded from [14].

5 References

[1] Breedveld, P.C., “Physical systems theory in terms of bond graphs”, Ph.D. Thesis, University of Twente, 1984

[2] 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

[3] Cellier, F.E., Elmqvist, H. and Otter, M., “Modeling from physical Principles” in The Control Handbook, W.S. Levine, ed., CRC Press, Inc., pp. 99-108, 1996

[4] Cellier, F.E. “Continuous system modelling”, Springer-Verlag, 1991

[5] van Dixhoorn, J.J., and F.J. Evans, eds., Physical Structure in System Theory, Academic Press, 1974

[6] Gawthrop, P. and Lorcan Smith, L., “Metamodelling: Bond Graphs and Dynamic Systems”, Prentice Hall, 1996

[7] Karnopp, D.C., Margolis, D.L. and Rosenberg, R.C., “System Dynamics: A Unified Approach”, John Wiley and Sons, New York, 1990

[8] Karnopp, D.C., and R.C. Rosenberg, "Analysis and Simulation of Multiport Systems - The Bond Graph Approach to Physical System Dynamics", M.I.T. Press, 1968

[9] Paynter, H.M., “Analysis and design of engineering systems”. MIT Press, Cambridge, Mass., 1961

[10] Thoma, J, “Introduction to Bond Graphs and their Applications”, Pergamon Press, 1975

[11] Wellstead, P.E., “Introduction to Physical System Modelling”, Academic Press, 1979.

[12] 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, Mechatronics 98, J.Adolfson and J.Karlsén (Editors), Elsevier Science Limited, 1998

[13] Many references can be found at: http://www.ece.arizona.edu/~cellier/bg.html

[14] http://www.rt.el.utwente.nl/amn/mech2000

[15] Broenink, J.F., Computer-aided physical-systems modeling and simulation: a bond-graph approach, PhD-thesis, Unversity of Twente, 1990

[16] Vries, T.J.A. de, Conceptual design of controlled electro-mechaincal systems, PhD. Thesis, Unversity of Twente, 1994