Mechatronic design – Modelling and Control

Prof. Job van Amerongen
C.J. Drebbel Institute for Mechatronics
University of Twente

In this article, the modelling of mechatronic systems in relation to control will be described. Some examples will show the usefulness of the 20-sim software, which translates the multiple views that exist in mechatronics, into each other.

Introduction

As has been said before, the different specialists in the mechatronics arena speak different languages. To solve this problem, one could try to find a common language through which the specialists can talk to each other (see article by Van den Bosch). Alternatively, one could respect the multiple views and find, instead of a common language, a language that automatically translates the language of one discipline into one of another.

But let us start with some definitions. A lot has been said about mechatronics, but some new views could be added to the ones presented before. According to Buur, mechatronics is 'a technology that combines mechanics with electronics and information technology, to form both functional interaction and spatial integration in components, modules, products and systems'. Also well known is IRDAC's definition: 'Mechatronics is a synergistic combination of precision engineering, electrical control and systems thinking in the design of products and manufacturing processes.' Third, Van Brussel's definition: 'Mechatronics encompasses the knowledge base and technologies for the flexible generation of controlled motion.'

The keywords in these definitions are systems approach, synergy, some specific domains (mechanics, electronics, information technology) and flexibility. The systems approach implies that one should look at a system as a whole, and not add the controller only after the mechanics have been built. Synergy results from the fact that we can look at a problem from various points of view, thanks to the various specialists from specific domains that contribute to mechatronics. Therefore, we do not want to train mechatronic engineers – instead, we need specialists from the above-mentioned fields, with the additional ability to communicate with other disciplines.

Modelling plays a crucial part in this whole process. Since mechatronics is about physical plants and products – and not just about theory – those models should be closely related to the physical components that are part of the mechatronic system. If one wants to change the system, then one wants to change the actual properties, like masses and spring characteristics. Especially in an early stage of the designs, simple models are essential. For this, supporting software is needed, such as 20-sim.

Multiple Views

Ideally, the supporting software should take into account the multiple views that are associated with mechatronics. People from different domains will more or less understand each other's view; however, one view is always better – more appropriate – than the other. Also, different design stages may require different views. Possible models include:

- Physical models
- Bond graphs
- Control engineering models, e.g.:
  - Block diagrams
  - Bode, Nyquist diagrams
  - State space
  - Time domain

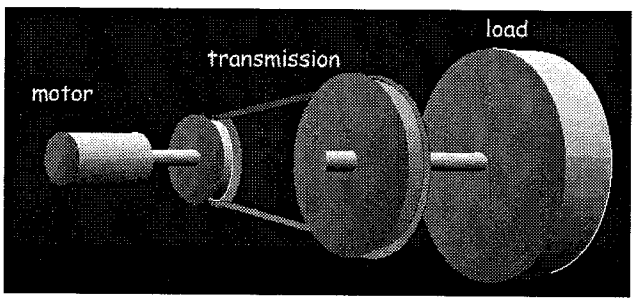

Fig. 1: Servo system

This can be illustrated with a simple servo system (Fig.1), that has been built in our lab as a demonstrator and is representative for many mechatronic systems. Some models of this system are depicted in the following figures: the components, a bond graph and a block diagram, resulting from the differential equations with which the system can be characterised. When written in

integral form, these equations can be used in a program like Simulink to simulate the system. In the physical domain, physical quantities like forces, velocities, voltages and currents are depicted rather than signals, so that it is easier to connect the components to other physical components.

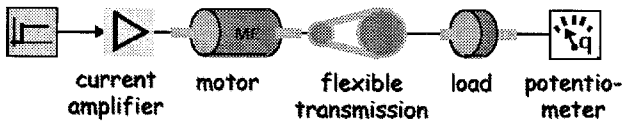

Fig. 2: Components of servo system

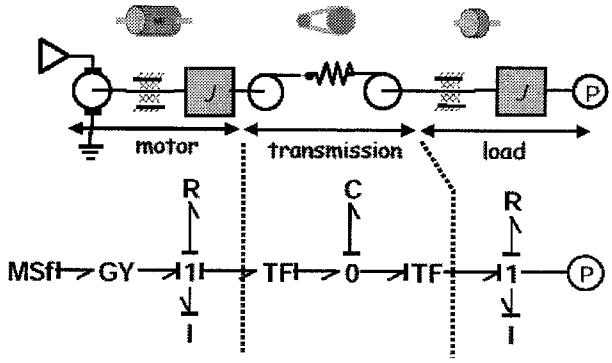

Fig. 3: Bond graph

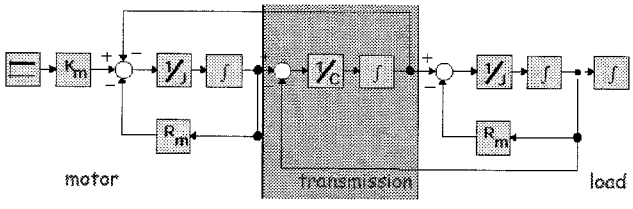

Fig. 4: Block diagram

When changing the properties of the system, one can arrive fast at a new representation of the system by changing the elements of the iconic diagrams. Alternatively, if you change for instance the complex poles of this system, all the other views of the system change as well. The various windows that may be open on a computer screen change simultaneously. Consequently, when one is sitting with a couple of people in front of the screen, one can discuss the system while looking at the preferred representation of the system.

The power of these simple simulations is that one can arrive at standard solutions for standard problems. Of course, it is also possible to implement more advanced controllers, such as a controller based on measurements of the load axis only and a subsequent Kalman filter. The measurements can be exported to Matlab, where the computations will be done; the result is the Kalman filter gain, which will be imported again in the 20-sim software, where the system is simulated. This would result in a nice controller, only based on measurements of the load axis position (Fig.5).

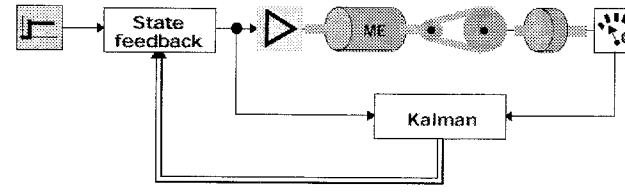

Fig. 5: Controller based on load axis

Other question: suppose there is a belt between the motor and the load. One may want to know what the force in the belt will be, to make sure that the belt will not break. Because of the physical modelling, it is very easy to monitor the force in the belt, and to minimise it. So: same system, just measuring the force in the belt. This can be achieved by tuning a setpoint generator (which normally is a system which has zeroes at the resonance frequencies, in order to avoid exciting those frequencies – in this case it appears that by tuning the zeroes a bit, the results become even better). See Fig.6.

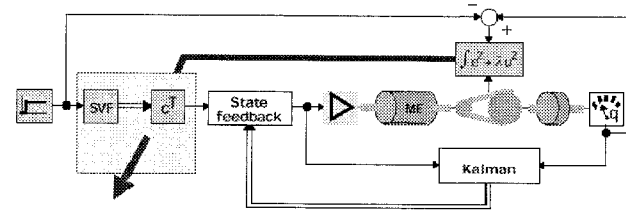

Fig. 6: Minimising the force in the belt

Standard solutions

Since we are working with more or less standard processes, it is possible to come up with standard descriptions and standard solutions. For instance, in a recent Ph.D. study (available on the web), Erik Coelingh examined 4th-order models that can be used for many systems – pure resonances, anti-resonance, and others. It is possible to make a menu-driven controller design and vary the physical parameters, in order to make a robust system, e.g. by applying QFT-techniques and studying the performance bounds.

Another example to which we can apply 20-sim is the Mobile Autonomous Robot Twente (MART; also mentioned by Koster). The idea was to design a robot that could move around in a factory and pick up components, to combine these in a product while driving around. The question was how to distribute the weight between the lower and the upper frame, in order to have the navigation system work properly. A model could help to answer this question. Again, multiple views can be used: equations, bond graphs, spring-damper submodels, et cetera. After a few iterations, the error between the top of the robot and the working platform had been reduced sufficiently, so that the robot could easily compensate for remaining errors. Bumps in the floor will excite the lower frame, but the upper frame is damped well enough to avoid vibrations there.

Motors

Motors can be divided into three groups: those driving belts, spindles, and linear motors. The three options have specific properties, making each suitable for a specific combination of velocity and accuracy requirements (Fig.7). Design issues typically associated with these motors:

- Belt: compliance
- Spindle: (Coulomb) friction
- Direct drive linear motor: cogging, (Coulomb) friction

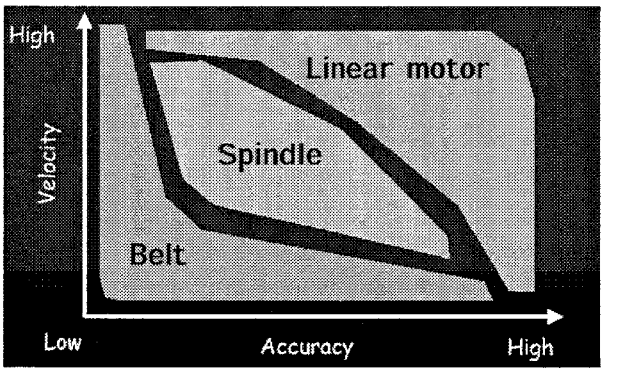

Fig. 7: (Linear) motors

Repetitive control and Learning feedforward control

When dealing with repetitive motions, one can apply repetitive control and iterative learning control. For instance, this has been applied to CD-players, to deal with the eccentricity. The idea is that if there is an error in one revolution, then this error may return in the next one. Using that information in a feed-forward loop, one can obtain a better controller. Similarly, in a neural network, if there are no disturbances but only a reference change, then the output of the system can be used in a feed-forward loop, thus relating the input of the network to time (time-indexed networks).

For non-repetitive motions it is better to use another input. The reference, and the first and second derivative of the reference may be used as input for the neural network, rendering the network a bit more complex. However, even with an arbitrary motion it is possible to achieve good control. Such a controller can be used for compensating non-linearities such as cogging. For instance, it has been used in a flight simulator, designed for Fokker, to compensate for friction. Also, it has been used in linear motors. The more or less sinusoidal disturbances due to cogging and the effects of friction were effectively neutralised: the error disappeared after a few motions. This type of control is called Learning Feedforward Control.

Conclusion

- Mechatronic design requires models with a clear relation to the physical parameters
- Multiple views are needed to allow discussion between specialists from different domains; this requires appropriate software
- Since problems can be classified, standard solutions can be sought
- 4^{th} order approximations will help to deal with major performance-limiting factors
- Certain classes of non-linearities can be adequately dealt with by learning feed-forward compensation