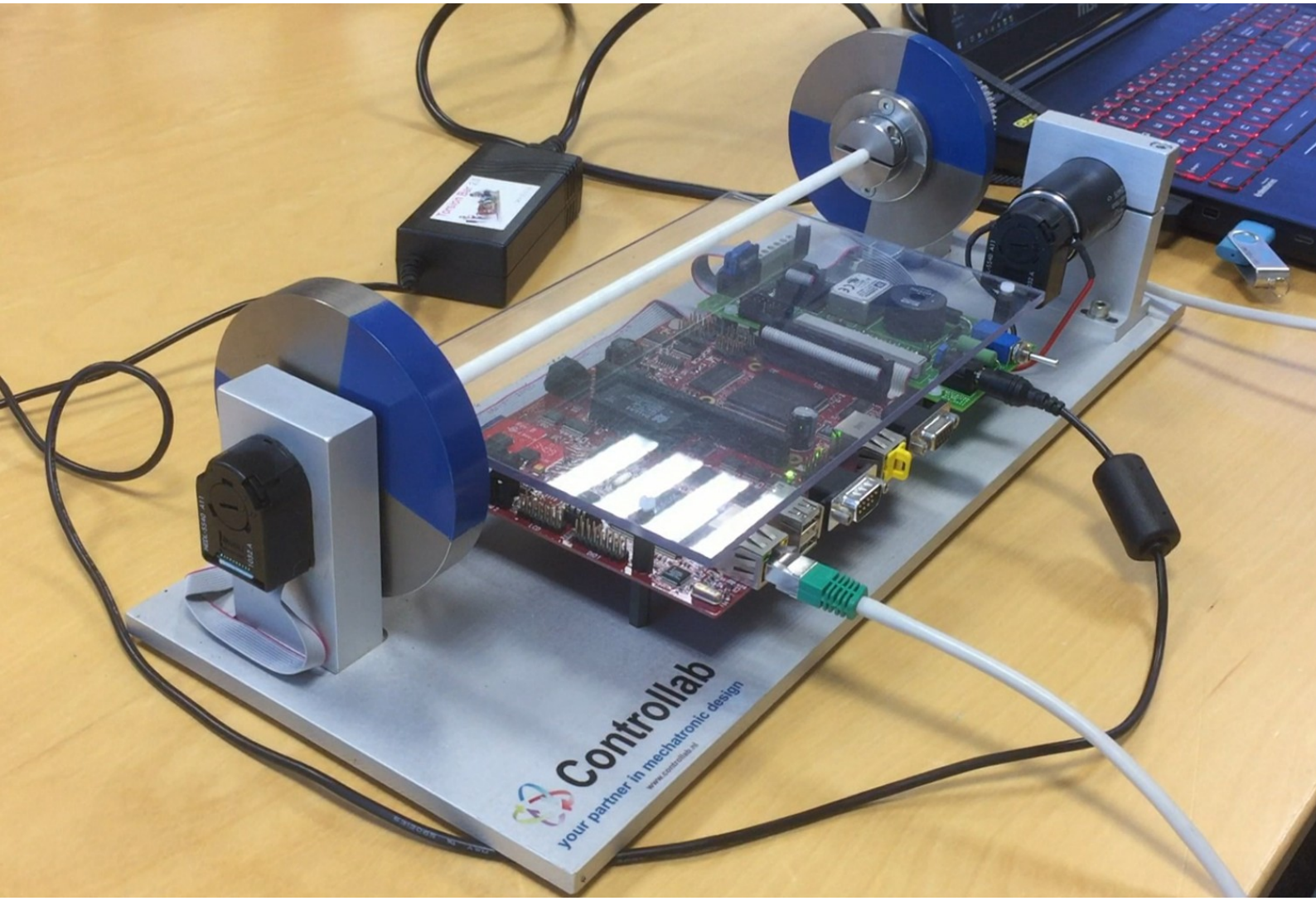
Seriously Improving Control



Snap Feedforward Control

C. Kleijn

Contents

Disclaimer

1. Introduction

To improve the accuracy of PID controlled machines, feedforward control can be added. With flexible machines this still leaves a significant tracking error. Snap feedforward is an efficient means to improve this tracking error. Snap is the fourth order derivative of the position and this type of feedforward has been described extensively by Boerlage (1). In this paper we will show the application of snap feedforward with a simple simulation model.

We will start with two masses connected rigidly to see how feedforward works. Then we will add flexibility and find that this will increase the tracking error significantly. We will counter this by adding snap feedforward. To demonstrate that snap feedforward also works in practical applications, we will demonstrate it on the Torsion Bar setup.

For the simulation models, 20-sim is used. In the appendix, you can find more information on 20-sim and instructions to download the package and the simulation models described in this paper.

1.1. PID Control

In many motion controlled machines, like printers, wafer stages or robots, a piece of the machine has to be transported following a predefined path with high accuracy. The control of these machines typically consists of an actuator applying a force onto the part to be moved. The actuator is most of the times controlled by a PID controller.

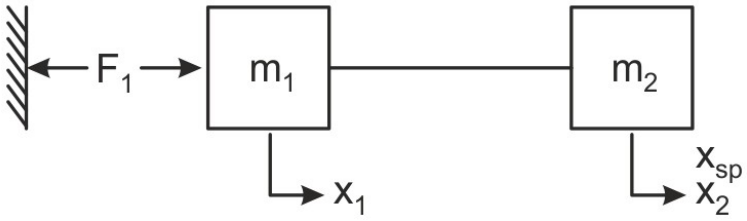


Figure 1: Example machine with two moving masses.

Suppose we have an example machine with two moving masses, as shown in figure 1. In this example we first start with two rigidly connected masses and add flexibility later. We want the second mass to follow a given setpoint position x_{sp}. A 20-sim model of this machine with a PID controller is shown below. At the left is a *MotionProfile* block that generates a position setpoint. The actual position of the first mass is subtracted to generate a position error and fed into the PID controller. The controller is connected to the force actuator.

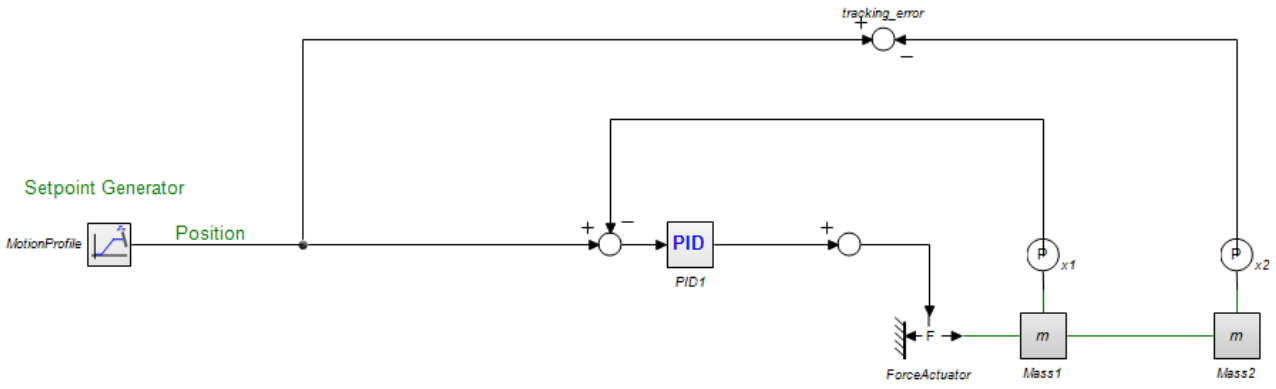


Figure 2: Two moving masses with a PID controller (model: PID control rigid.emx).

You can download the model *PID control rigid.em*x and run it in 20-sim.

In the model, the following parameter values were used:

```
m1      1 {kg}          // mass 1
m2      1 {kg}          // mass 2
kp      200000.0        // proportional gain PID controller
tauD    4.0 {ms}        // derivative time PID controller
taul    10.0 {ms}       // intergral time PID controller
beta    0.1             // tameness constant PID controller
```

In the model the distance *d* between both masses is ignored and assumed zero. In the rest of this paper we will continue to assume the distance is zero. A 9th order polynomial motion profile has been used to generate the position setpoint. The resulting plots are shown in figure 3. In the left plot the position setpoint (*x*), the double derivative (a) of the position setpoint and the fourth order derivative (*s*) of the position setpoint are shown. The middle plot shows the positions of both masses and the position setpoint. The right plot shows the tracking error with a maximum of 5 mm.

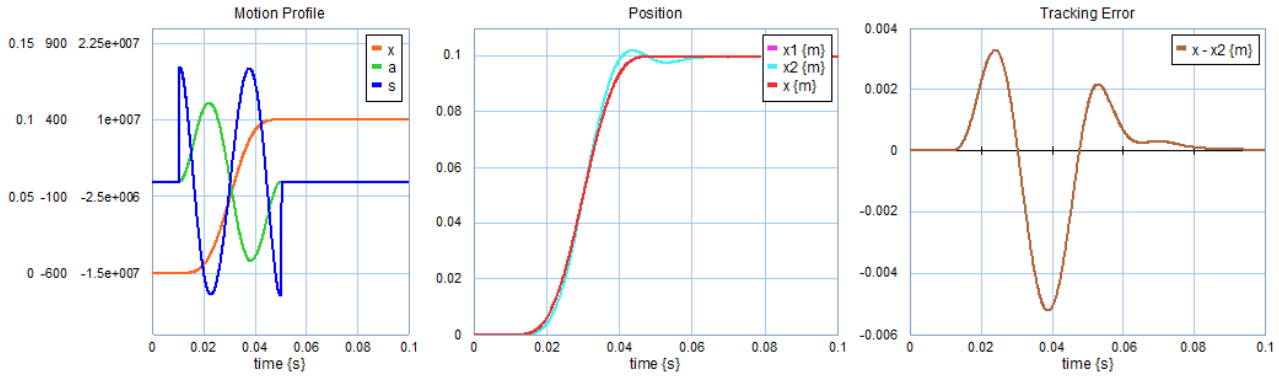


Figure 3: Simulation results for the PID controller (model: PID control rigid.emx).

1.2. Acceleration Feedforward

The resulting tracking error is significant during the motion. The error may be reduced if we apply feedforward. Feedforward is based on the fact that we know in advance the force that has to be applied, here an acceleration force:

$$F = m_1 \frac{d^2 x_1}{dt^2} + m_2 \frac{d^2 x_2}{dt^2} \tag{1}$$

The speed and acceleration of both masses are equal:

$$\frac{d x_2}{dt} = \frac{d x_1}{dt}$$
$$\frac{d x_2^2}{dt^2} = \frac{d x_1^2}{dt^2} \tag{2}$$

which simplifies the force of equation 1 to:

$$F = (m_1 + m_2)\frac{d^2 x_2}{dt^2} = (m_1 + m_2)\frac{d^2 x_{sp}}{dt^2} \tag{3}$$

In the 20-sim model *PID control with feedforward rigid.emx* this force is added next to the output of the PID controller. The double derivative (acceleration) of the position setpoint is multiplied with the sum of the masses (inside the *FF_Acc* block) and added to the force actuator.

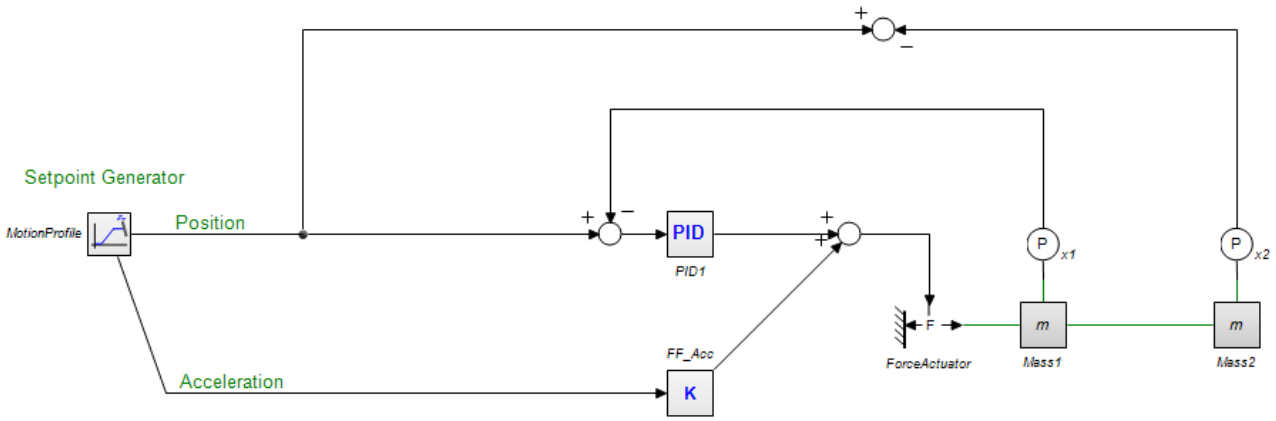


Figure 4: Two moving masses with a PID controller and feedforward (model: PID control with feedforward rigid.emx).

Theoretically the tracking will reduce to zero! In the simulation run the tracking error in the right plot of figure 5 is equal to the accuracy of the integration method.

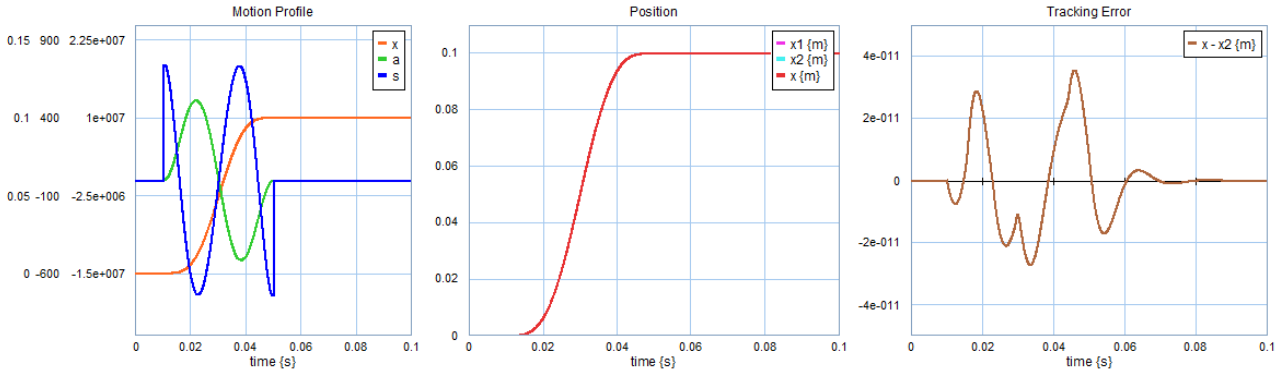


Figure 5: Simulation results for the PID controller (model: PID control with feedforward rigid.emx).

In practice the tracking error is never equal to zero, because of friction, flexibility, non-ideal actuators, limited sampling frequencies etc. In the next chapter we will investigate what will happen if the machine is flexible and how we can counter this by using the fourth order derivative of the position setpoint (called snap). This method of control is called snap feedforward. In the third chapter we will apply snap feedforward to the Torsion Bar setup to see how it works out on a practical machine.

2. Snap Feedforward

2.1. Flexibility

Every machine will have friction, flexibility, non-ideal actuators and other effects that make acceleration feedforward less effective. In this paper we will investigate the effect of flexibility in machines and see how we can counter this. For our example system this means that we will add a spring between both masses. Again we want the second mass to follow a setpoint position x_{sp}. We assume the unstrected spring length is zero.

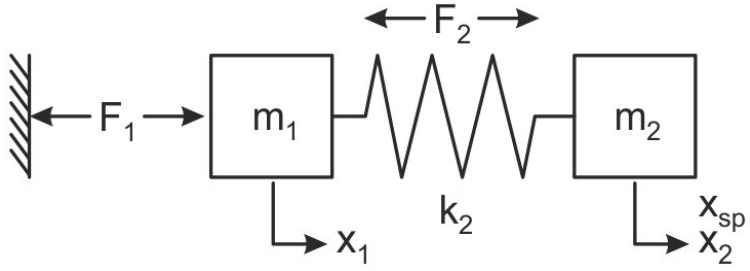


Figure 6: Example machine with two moving masses and stiffness.

2.2. Tracking Error

The model *PID control with feedforward.emx* has a flexibility between both masses. The stiffness *k2* is equal to 1.1e6 N/m resulting in a resonance frequency of 170 Hz. The unstretched spring length is equal to zero.

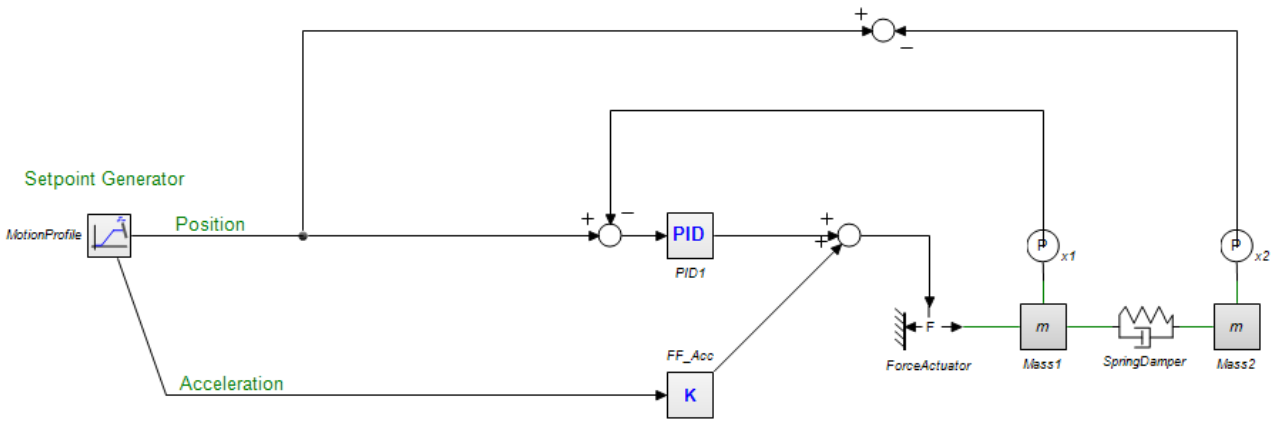


Figure 7: Feedforward control with flexibility (model: PID control with feedforward.emx).

Although the stiffness is quite high, the effect on the tracking error is significant. The tracking error increases to 0.5 mm!

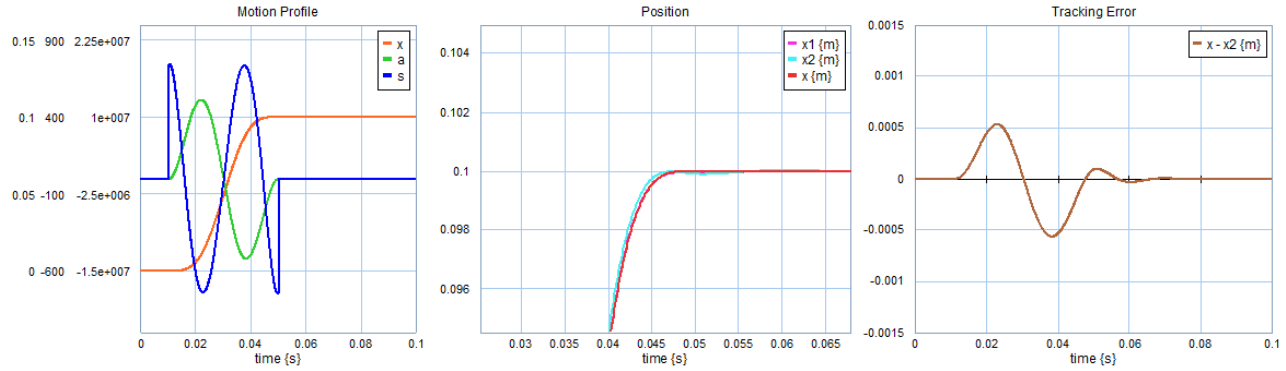


Figure 8: Simulation results for the model with flexibility (model: PID control with feedforward.emx).

The tracking error may significantly be reduced if snap feedforward is added that takes into account the flexibility.

2.3. Setpoint Position Correction

The first problem that we encounter with the spring added (see figure 6) is that most machines measure the position directly on the actuator (x_1) and not on the end effector (x_2). We want to control the end effector and make it follow a setpoint, but because of the spring, x_1 and x_2 are not equal. Therefore we have to find an equation that gives the position of the end effector x_2 into terms of the actuator position x_1.

The difference in position of both masses gives a spring force:

$$F_2 = k_2 \cdot (x_1 - x_2) \tag{4}$$

The spring force defines the acceleration of the second mass:

$$F_2 = m_2 \frac{d^2 x_2}{dt^2} \tag{5}$$

Combining equations 4 and 5 gives our description of x_1 into terms of x_2:

$$x_1 = x_2 - d + \frac{m_2}{k_2} \cdot \frac{d^2 x_2}{dt^2} \tag{6}$$

Because x_2 has to follow the setpoint x_{sp}, we can write:

$$x_1 = x_{sp} + \frac{m_2}{k_2} \cdot \frac{d^2 x_{sp}}{dt^2} \tag{7}$$

This means that the setpoint position for the PID controller acting on mass m_1 has to be corrected for the second derivative multiplied by the second mass and divided by the stiffness. In the 20-sim model, we will name the second derivative correction term *Corr_Snap*.

2.4. Force Feedforward

In the previous section, we have only determined the position correction. We still have to determine the actuator force F_1. To derive this we start with the net force on the first mass. It is equal to the actuator force minus the spring force:

$$F_1 - F_2 = m_1 \frac{d^2 x_1}{dt^2} \tag{8}$$

Combining equations 8, 4 and 6:

$$\begin{aligned} F_1 &= m_1 \frac{d^2 x_1}{dt^2} + k_2 \cdot (x_1 - x_2) \\ &= m_1 \frac{d^2 \left(x_2 + \frac{m_2}{k_2} \cdot \frac{d^2 x_2}{dt^2} \right)}{dt^2} + k_2 \cdot \left(x_2 + \frac{m_2}{k_2} \cdot \frac{d^2 x_2}{dt^2} - x_2 \right) \\ &= (m_1 + m_2) \cdot \frac{d^2 x_2}{dt^2} + \frac{m_1 m_2}{k_2} \cdot \frac{d^4 x_2}{dt^4} \end{aligned} \tag{9}$$

Because x_2 has to follow the setpoint x_{sp}, we can write:

$$F_1 = (m_1 + m_2) \cdot \frac{d^2 x_{sp}}{dt^2} + \frac{m_1 m_2}{k_2} \cdot \frac{d^4 x_{sp}}{dt^4} \tag{10}$$

Equation 10 gives the feedforward term that has to be added to the force actuator. It consists of an acceleration part (equal to equation 3) and a snap part. In the 20-sim model, we will name the acceleration term *FF_Acc* and the snap part *FF_Snap*.

2.5. Simulation

The model *PID control with feedforward and snap.emx* shows an implementation of the snap feedforward. In this model, the spring length *d* is equal to zero. The snap correction term *Corr_Snap* (see equation 7) is added to the setpoint for the PID controller. The snap feedforward term *FF_snap* (see equation 10) is added to the force actuator.

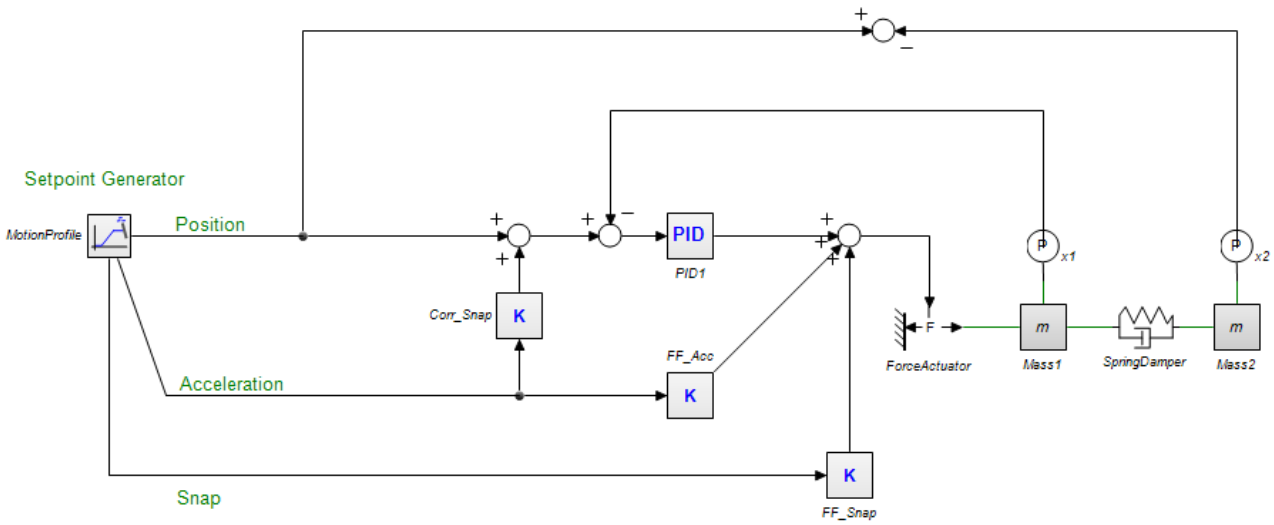


Figure 9: The two masses model with snap feedforward (model: PID control with feedforward and snap.emx).

The tracking error reduces from 0.5 mm to 0.01 mm!

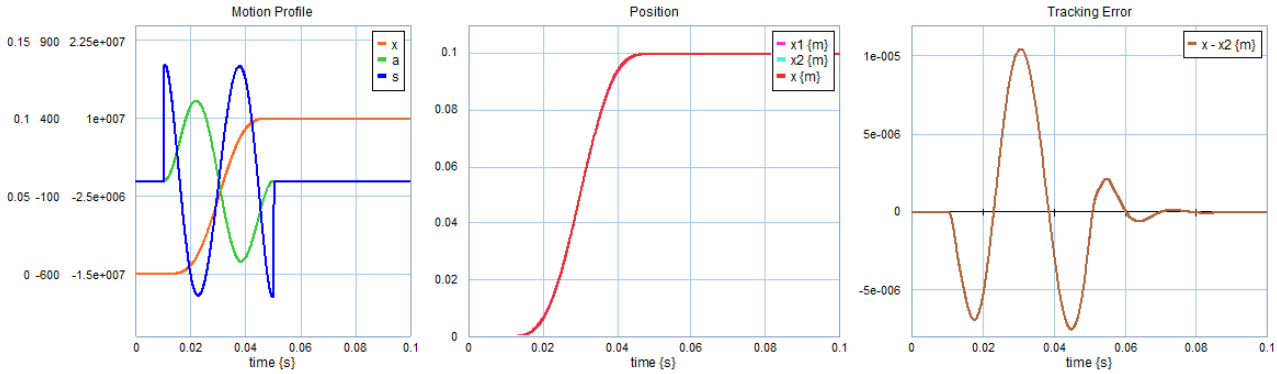


Figure 10: Simulation results for the model with snap (model: PID control with feedforward and snap.emx).

2.6. Robustness

The model *PID control with feedforward and snap.emx* works very well even though a spring damper is used while the snap feedforward is only based on a spring only system. You can inspect the damping value in 20-sim by clicking in the Simulator: *Properties - Parameters*. This will open the *Parameters* Editor which allows you to inspect the parameters of the model and change them.

The first three parameters are the parameters of the acceleration feedforward and snap feedforward controllers:

k2 = 1.14e6 {N/m}
m1 = 1.0 {kg}
m2 = 1.0 {kg}

You can effectively turn the snap feedforward off by making *k2* very high (e.g 1e20 {N/m}). You can effectively turn the acceleration feedforward off by making *m1* and *m1* zero.

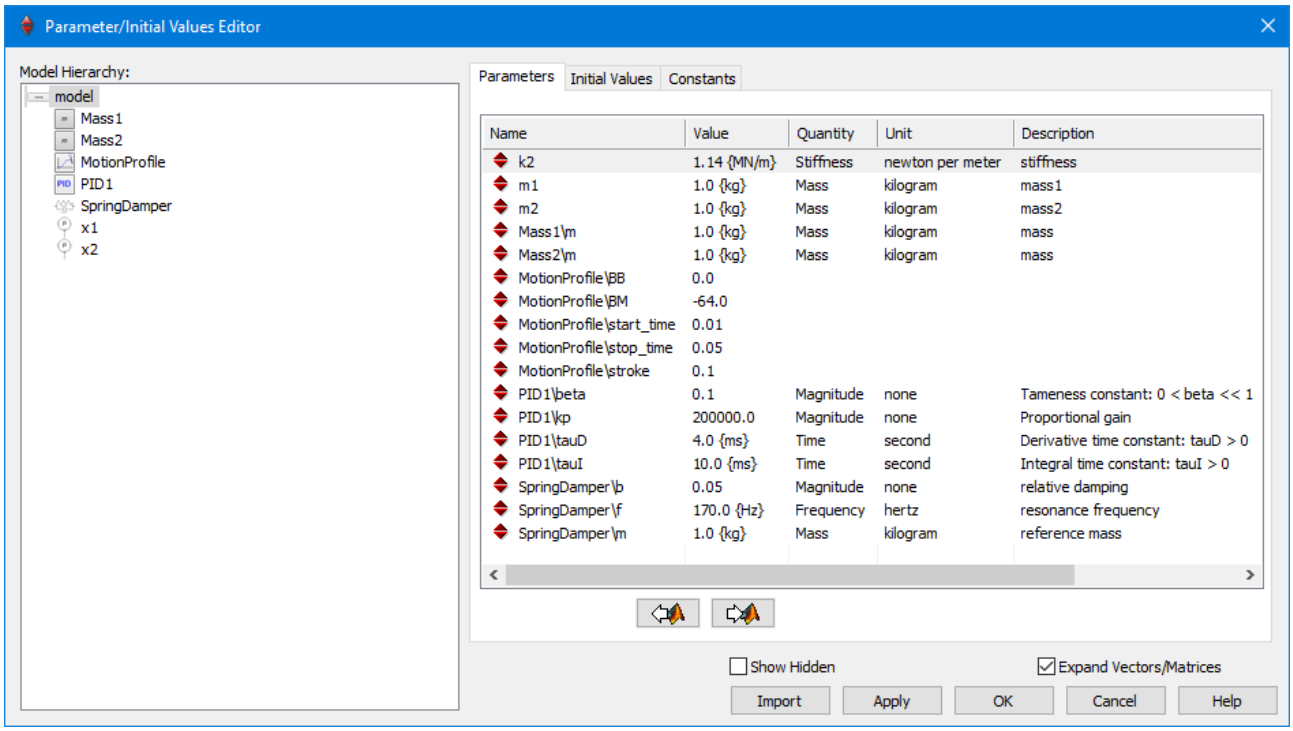


Figure 11: The parameter values in the model PID control with feedforward and snap.emx.

The damping value is given by *SpringDamper\b* = 0.05. If you increase the damping, the tracing error will increase but the snap feedforward will still be beneficial. This is in general so for snap control. On general machines it will have beneficial effects, if proper values are used.

2.7. Tuning

If the proper values for the snap feedforward are not known the following tuning procedure may be used:

1. Effectively turn-off snap feedforward by making the estimated stiffness k_2 very high.
2. Effectively turn-off feedforward control by setting the estimated masses m_1 and m_2 to zero.
3. Tune the PID controller until a good tracking error is obtained.
4. Give both estimated masses m_1 and m_2 a starting value based on a best guess and change their values until a minimal tracking error is obtained.
5. Reduce the stiffness k_2 until a minimal tracking error is obtained.

You can check this with the simulation model *PID control with feedforward and snap.emx*. Set the following parameter values:

k2 = 1e20 {N/m}
m1 = 0.0 {kg}
m2 = 0.0 {kg}

With these parameter values we have a properly tuned PID controller without feedforward control as shown in figure 12.

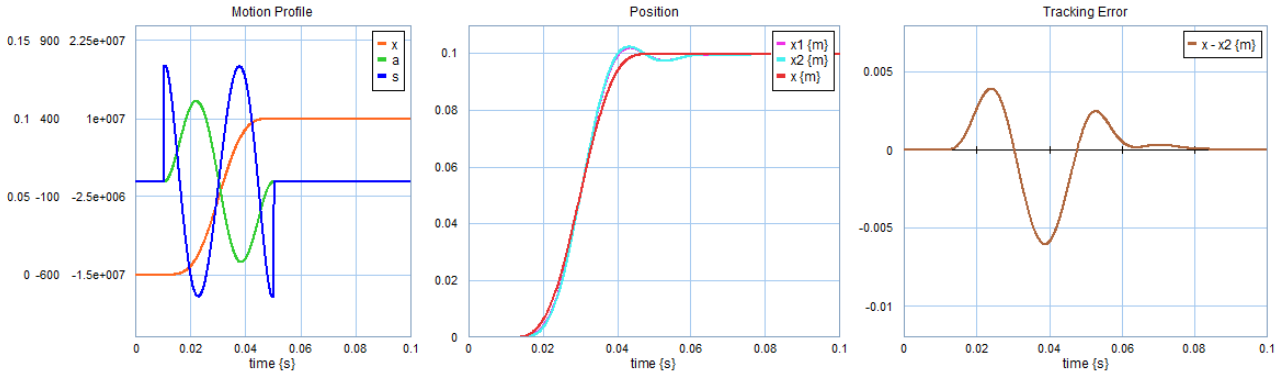


Figure 12: Simulation results for k2 = 1e20, m1 = 0, m2 = 0.

Suppose we have followed the tuning rule 4 and found for both masses a value of 1.05 kg. These values are 5% off the real values. We can investigate the effects by setting the following parameter values:

k2 = 1e20 {N/m}
m1 = 1.05 {kg}
m2 = 1.05 {kg}

The resulting simulation is shown in figure 13.

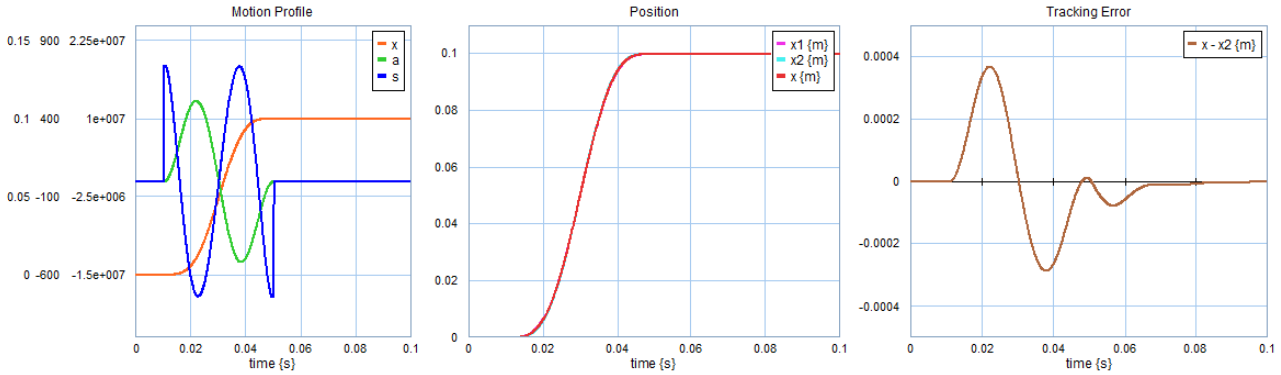


Figure 13: Simulation results for k2 = 1e20, m1 = 1.05, m2 = 1.05.

Now we will follow tuning rule 5 and ower the stiffness values. We start with

k2 = 3.0e6 {N/m}

If you gradually lower the stiffness value and run simulations, you will find that a minimal tracking error is found for:

k2 = 2.0e6 {N/m}

The resulting simulation is shown in figure 14.

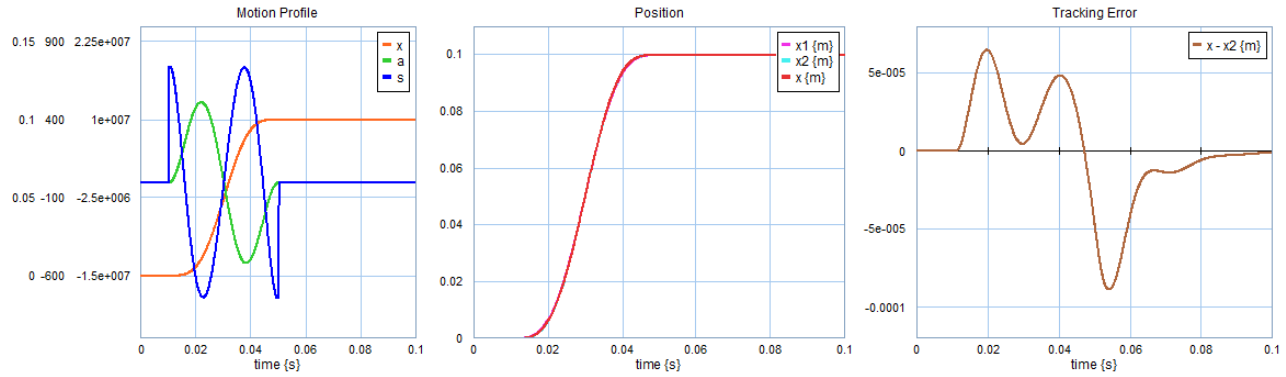


Figure 14: Simulation results for k2 = 2.0e6, m1 = 1.05, m2 = 1.05.

Because the masses are 5% off, the stiffness will also be off. Even so, the snap feedforward controller works pretty well! The tracking error has been reduced considerably.

2.8. Higher Order Flexibilities

Equation 7 and 10 suggests that higher order flexibilities can be countered by higher order derivatives of the setpoint. This is in more detail described in the paper of Boerlage (1). There it is explained that higher order derivatives are most of the times not required, because the higher resonant modes lie orders apart from the lowest resonant mode. It means that snap feedforward will give good results for many systems.

3. Torsion Bar

3.1. Introduction

The Torsion bar is a setup that is typical for an industrial machine and it contains all the elements that we can find in industrial machines like a servo motor attached to a belt and a rotating disk (*motor disk*). The disk is connected to another disk (*load disk*) via a flexible shaft. The flexibility of the axis is very high which results in a resonance mode that is so low, that you can see it with your own eyes. The setup is made robust to allow students to rotate the disks to induce the resonance mode by hand.

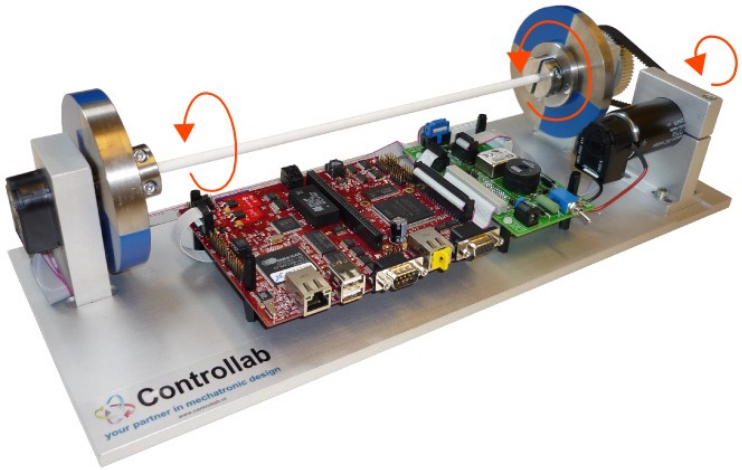


Figure 15: The Torsion Bar setup.

3.2. Acceleration Feedforward

The 20-sim model *Torsion Bar with PID and Feedforward.emx* is a very accurate model of this setup. You can run it and see the resulting motion in a plot and as a 3D animation.

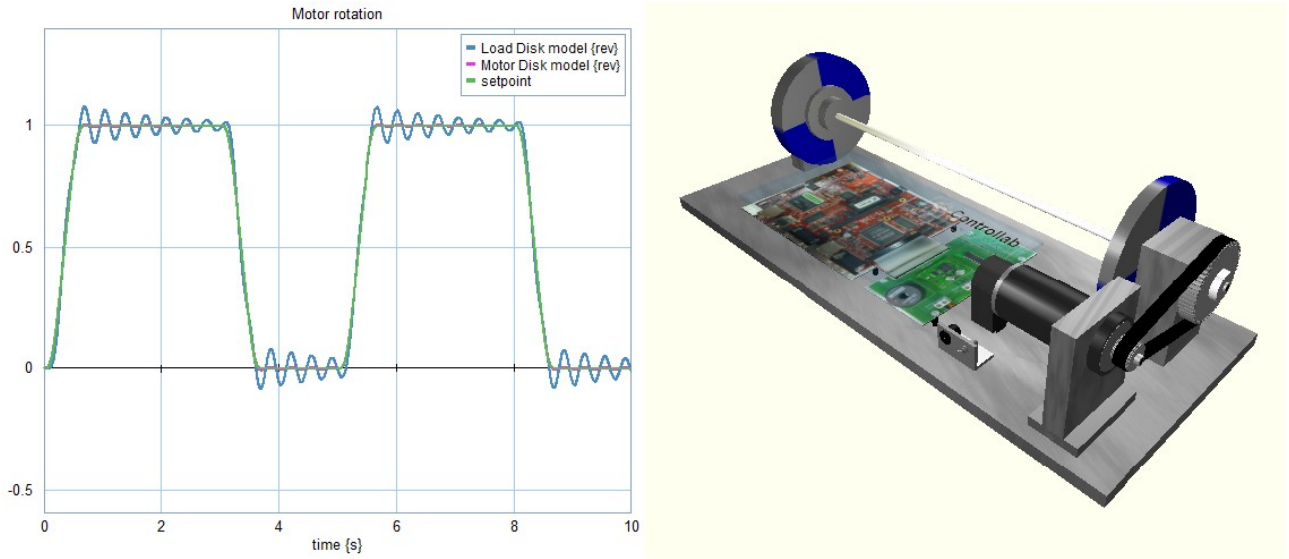


Figure 16: Simulation results for the Torsion Bar setup (model: Torsion Bar with PID and Feedforward.emx).

You can replay the animation in 20-sim, by clicking the green play button. The simulation shows that the motor disk is controlled pretty well but the load disk shows large vibrations. This is confirmed by experiments on the real setup. The video *Torsion Bar with PID and Feedforward.wmv* shows a run made with the setup.

3.3. Snap Feedforward

Now open the 20-sim model *Torsion Bar with PID and Snap Feedforward.emx*. This model has snap feedforward implemented.

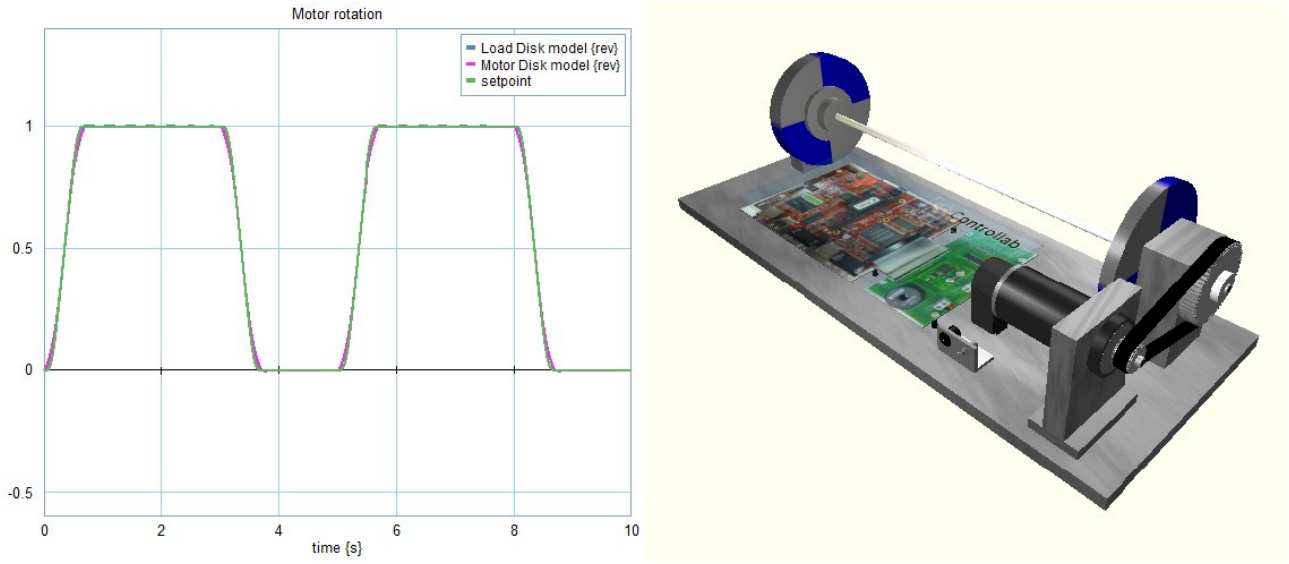


Figure 17: Simulation results for the Torsion Bar setup (model: Torsion Bar with PID and Snap Feedforward.emx).

The simulation results show that the vibrations of the load disk have significantly been reduced! You can also clearly see that the motor disk, during acceleration, is rotated in advance of the setpoint to make the load disk exactly follow the setpoint. During deceleration this is repeated vice-versa. The simulation results are confirmed by experiments on the real setup. The video *Torsion Bar with PID and Snap Feedforward.wmv* shows a run made with the setup.

Snap feedforward works very well on the Torsion Bar, even though the setup has many non-linearities like:

- A real non-ideal actuator
- significant friction
- discrete-time controller
- a limited range of measurement values through the encoders

The only observation made was, that the sample frequency may need to be increased a little, to get a good fourth order derivative of the position setpoint.

4. More Information

Controllab has over 20 years of experience in the design, testing and implementation of control systems. For more information on control systems design and the application of control systems in your company please contact Controllab.

Company:	Controllab
Address:	Hengelosestraat 500
	7521 AN Enschede, The Netherlands
Tel:	085-7731872
E-mail:	info@20sim.com
	info@controllab.nl
Web:	www.20sim.com
	www.controllab.nl

5. Appendix A: Literature

1. M. Boerlage, R. Tousain, M. Steinbuch, “Jerk derivative feedforward control for motion systems”, August 2004, Proceedings of the American Control Conference 5:4843 - 4848 vol.5, DOI: 10.1109/ACC.2004.182719

6. Appendix B: 20-sim

All the models described in this paper were created with 20-sim. You can download these models and a free demo version of 20-sim from the website:

https://www.20sim.com/blog/control-engineering/snap-feedforward/