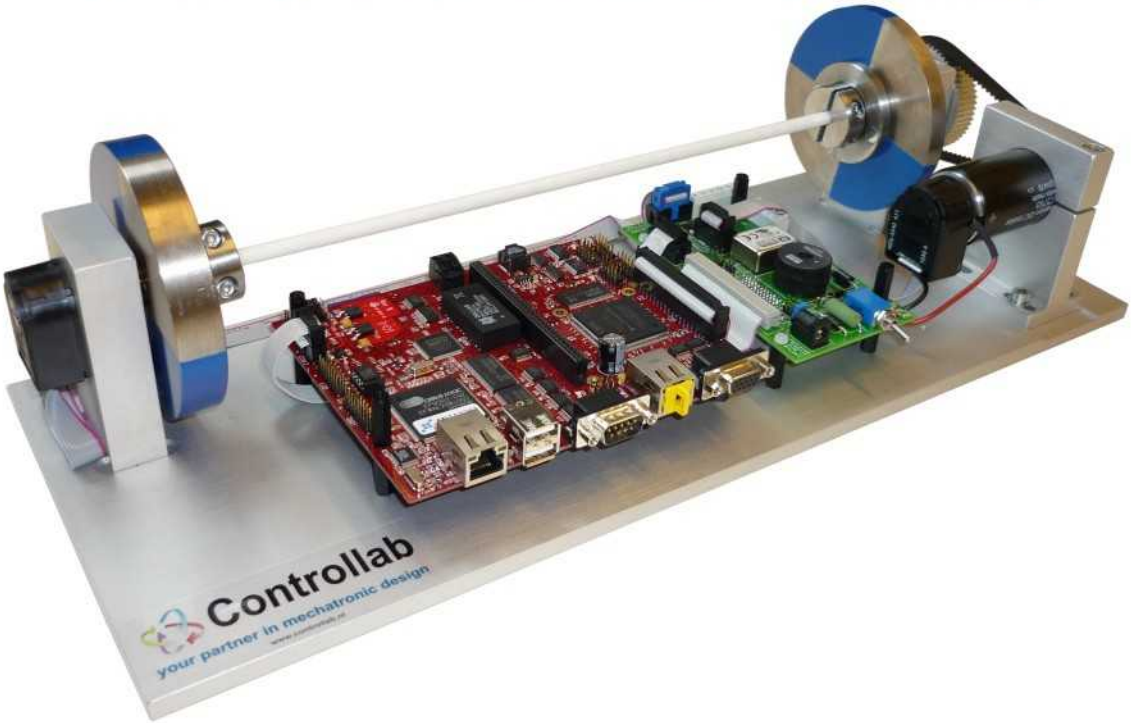
Torsion Bar



Torsion Bar 2.0 Reference Manual

Windows® XP / Vista / 7

Torsion Bar 2.0

Author: Ir. C. Kleijn, Ir. H. Differ, Ir. M. A. Groothuis

Disclaimer

This manual describes the Torsion Bar Setup.

Reference

Kleijn, C.
Torsion Bar 2.0 Reference Manual
Enschede, Controllab Products B.V., 2011
ISBN 978-90-79499-10-6

Information

Controllab Products B.V.
Address: Hengelosestraat 705
7521 PA Enschede
the Netherlands
Phone: +31-53-4836434
Fax: +31-53-4337415
Internet: www.20sim.com
www.controllab.nl
E-mail: info@20sim.com

Table of Contents

1 Introduction

1.1 What is the Torsion Bar 2.0?

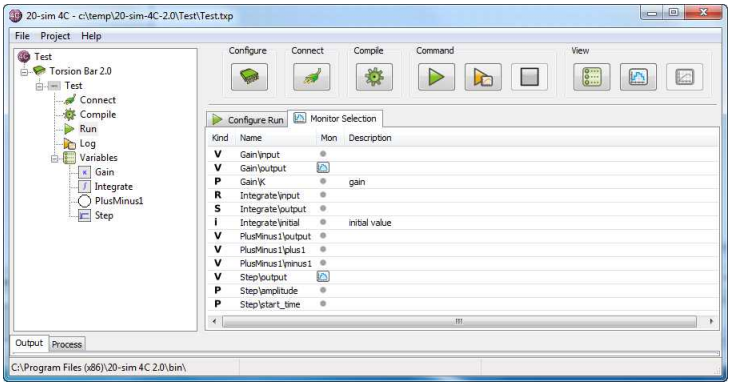
The Torsion bar is a setup that is typical for an industrial machine and it contains all the elements that we can find in industrial machines:

The typical elements we can find in an industrial machine.

The Torsion Bar can be operated in combination with the software packages 20-sim and 20-sim 4C. It comes in two versions. Version 1.0 has a voltage amplifier and is used for teaching machine dynamics and the basics of control engineering. This manual describes version 2.0, that is used for teaching machine dynamics and advanced control.

Operator Display

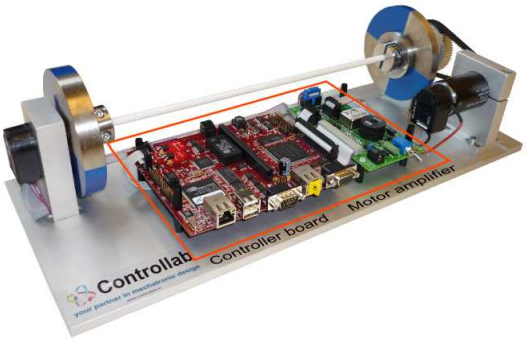
The Torsion Bar is equipped with an ethernet connection that allows communication with your local PC. The software 20-sim 4C is the operator display that supports the human-machine interaction.



The software 20-sim 4C allows you to operate the Torsion Bar.

Controller and Amplifier

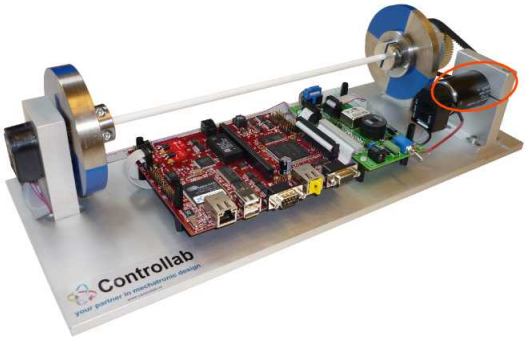
The controller and amplifier control the rotation of the disks by steering the servo motor in a controlled way. The controller is an embedded board with an ARM9 CPU and an FPGA for generating a PWM motor setpoint. The current amplifier generates a current that is following the PWM setpoint from the controller.



The motor controller and amplifier.

Servo Motor

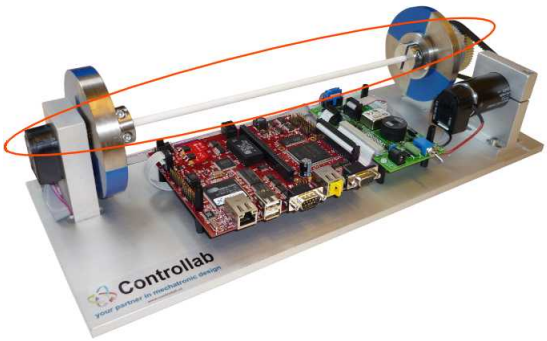
The servo motor is the actuator of our setup. It is a DC motor that drives the disks, and it is steered by a current that comes from the current amplifier.



The servo motor.

Drive train

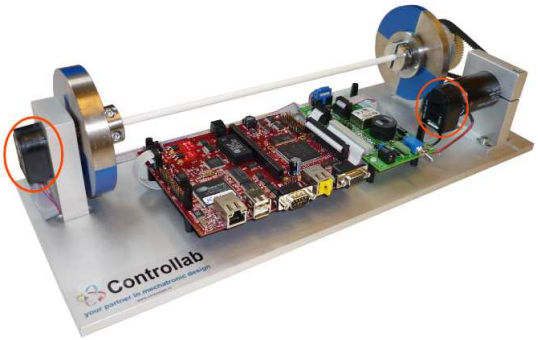
The drive train contains the moving parts of the setup. The drive train consists of two disks connected by a flexible beam. One disk is driven by the servo motor via a stiff belt. The stiffness of the flexible beam is deliberately chosen very low to ensure that the resonance and anti-resonance frequencies are visually apparent: the amplitudes are large and the frequencies are low.



The drive train.

Sensors

Two sensors measure the rotation of the motor and the large disk. The sensor readings are used by the motor controller to steer the motion of the drive train. The sensor readings also allow the operator panel to display the behavior of the setup in detail. In the torsion bar setup the sensors are two rotary encoders, that are connected to counters on the controller board.



The sensors.

Requirements

- A Windows PC with ethernet connection.
- 20-sim 4.1 Professional or higher.
- 20-sim 4C 2.0 or higher.

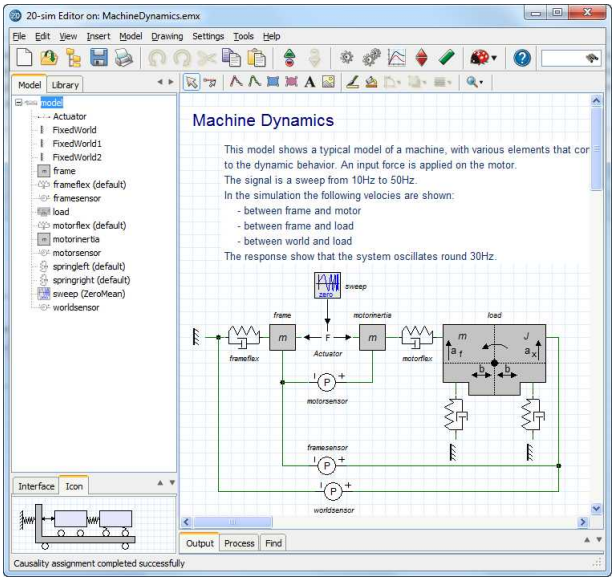
Versions

The Torsion Bar comes in two versions:

- Torsion Bar 1.0: This version is equipped with a voltage amplifier. It is used for teaching the basics of machine dynamics and control.
- Torsion Bar 2.0: This version is equipped with a current amplifier. It is used for teaching the basics of machine dynamics and control and advanced machine control.

1.2 What is 20-sim?

20-sim is a modeling and simulation program that runs under Microsoft Windows. With 20-sim you can simulate the behavior of dynamic systems, such as electrical, mechanical and hydraulic systems or any combination of these. 20-sim fully supports graphical modeling, allowing to design and analyze dynamic systems in a intuitive and user friendly way, without compromising power. 20-sim supports the use of components. This allows you to enter models as in an engineering sketch: by choosing components from the library and connecting them, your engineering scheme is actually rebuilt, without entering a single line of math!



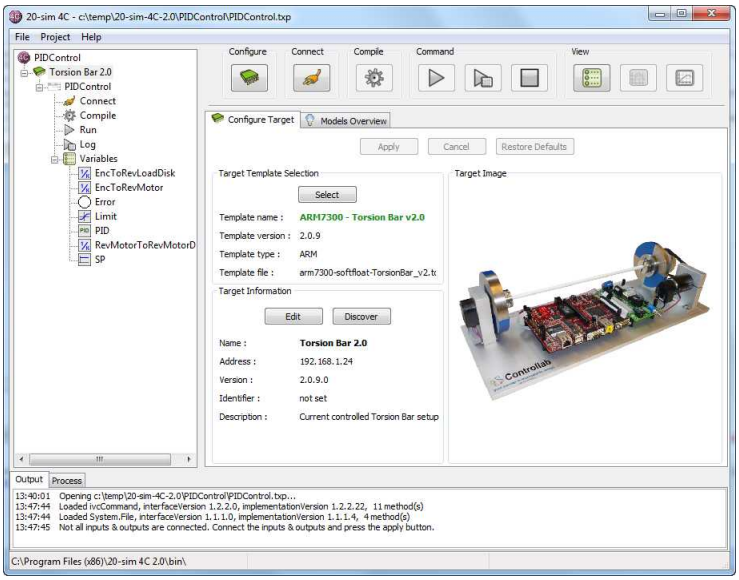
The software package 20-sim, with a machine model loaded.

20-sim will be used to create controller models for the Torsion Bar. These models can be exported to 20-sim 4C as C-code. More information on 20-sim can be found on the website www.20sim.com.

1.3 What is 20-sim 4C?

20-sim 4C is a prototyping environment that allows you to connect 20-sim models to physical systems. The models can be executed as real-time C-code on targets like PC's or ARM-9 based processor boards. This allows you to perform various tasks:

- Measurement and Calibration: From 20-sim 4C you can export C-code that will operate and read sensor signals. The signals can be inspected or logged for analysis in external programs.
- Machine Control: With 20-sim 4C you can export machine control code to external targets to control the operation of machines. 20-sim 4C acts as the operator interface by allowing you to start and stop controllers and change parameters during run-time.
- Rapid Prototyping: Models that are created in 20-sim can be exported to 20-sim 4C with the click of a button and executed on a target with a second click. The results can be logged automatically and imported in 20-sim to compare with the simulation model. This makes 20-sim 4C a valuable tool for rapid prototyping.



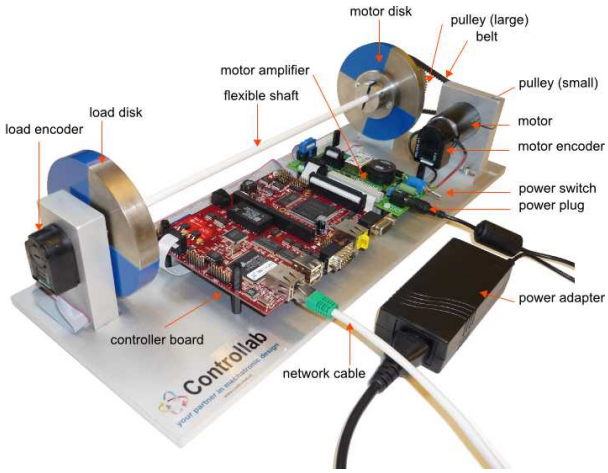
20-sim 4C helps you to get 20-sim models running as C-code on real-time processors.

20-sim 4C will be used to export the (cross-)compiled C-code to the Torsion Bar. From 20-sim 4C we can start and stop the code, monitor variables and take measurements. More information on 20-sim 4C can be found on the website www.20sim4C.com.

2 Getting Started

2.1 Components

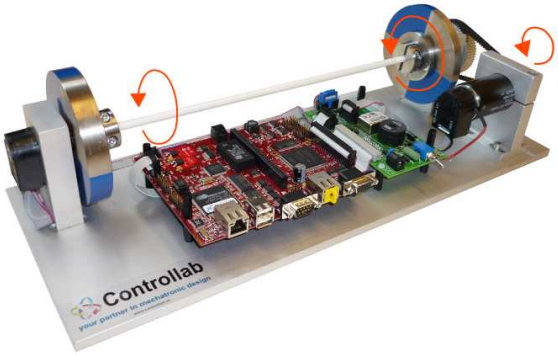
The Torsion Bar Setup is built out of various components.



The components of the torsion bar.

Motor Encoder

The motor encoder is an incremental encoder that gives 2000 pulses per revolution. The motor encoder is connected to a counter on the controller board that is initialized at 0 at the startup. The maximum counting frequency is 18.75 MHz. The counting software uses doubles which makes the total number of counts in practice unlimited. The counter is listed in 20-sim 4C as Motor - Quadrature counter. The positive direction of rotation is given in the next picture.



The direction of positive rotation for the load disk, motor disk and motor.

If you want to calculate the rotation of the motor disk out of the motor encoder, you have to divide the counts by 3.75 to account for the belt ratio.

Load Encoder

The load encoder is an incremental encoder that gives 2000 pulses per revolution. The encoder is connected to a counter on the controller board that is initialized at 0 at the startup. The counting software uses doubles which makes the total number of counts in practice unlimited. The counter is listed in 20-sim 4C as Load - Quadrature counter. The positive direction of rotation is given in the picture above.

Load Disk

The load disk is made out of brass with a specific density of 8500 kg/m3. The load disk consists of a main disk and a smaller disk on which a clamp is mounted to hold the flexible shaft. The total mass and rotational inertia are shown in the next table.

Disk	diameter mm	width mm	Mass kg	Inertia kg.m^2
Main Disk	100	16	1.07	1.34E-003
Small Disk	50	5	0.08	2.61E-005
Clamp	30	13	0.08	8.79E-006
Total			1.23	1.37E-003

Motor Disk

The motor disk is made out of brass with a specific density of 8500 kg/m3. The motor disk consists of a main disk and a smaller disk on which a clamp is mounted to hold the flexible shaft. The total mass and rotational inertia are shown in the next table.

Disk	diameter mm	width mm	Mass kg	Inertia kg.m^2
Main Disk	100	10	0.67	8.34E-004
Small Disk	50	5	0.08	2.61E-005
Clamp	30	13	0.08	8.79E-006
Total			0.83	8.69E-004

Pulley (Large)

The large pulley is made by the company Gates (type: Powergrip GT2 Sprocket 3MR-60S-09) and has 60 teeth. The pulley is made out of aluminium with a specific density of 2700 kg/m3. The total mass and rotational inertia are shown in the next table.

Disk	diameter mm	width mm	Mass kg	Inertia kg.m^2
Main Disk	56	13.5	0.09	3.52E-005
Bevel 1	32	9	0.02	2.50E-006
Total			0.109	3.77E-005

Pulley (Small)

The small pulley is made by the company Gates (type: Powergrip GT2 Sprocket 3MR-16S-09) and has 16 teeth. The pulley is made out of aluminium with a specific density of 2700 kg/m3. The total mass and rotational inertia are shown in the next table.

Disk	diameter mm	width mm	Mass kg	Inertia kg.m^2
Main Disk	12	14	0.004	7.70E-008

Flexible Shaft

The flexible shaft is made out of the material Delrin. The specific parameters are:

length (mm)	295
diameter (mm)	6.3
modulus of elasticity E (Gpa)	3
poison ratio μ	0.4
shear modulus G (Gpa)	1.07
stiffness (Nm/rad)	0.56

The shear modulus G is calculated out of the modulus of elasticity as:

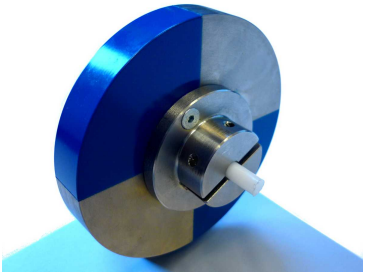
$$G = \frac{E}{2 \cdot (1 + \mu)}$$

The poison ratio μ is not known. A good estimate is 0.4. This makes the calculated stiffness also an estimate:

$$k = \frac{G \cdot \pi \cdot d^4}{32 \cdot l}$$

Shaft Pieces

If the flexible shaft is removed, the clamps will give an unbalance when the disk is rotated. You can use the small shaft pieces to place in the clamp opening. If you now close the clamp it will be balanced.



Use a shaft piece to balance the clamp.

Belt

The belt between the motor shaft and the motor disk is a Gates Powergrip GT2 3MR-300-09 belt. The belt has 100 teeth with a pitch of 3 mm. The total length of the belt is 300 mm. The total weight of the belt is 7.6 g. The stiffness of the belt is not given, by the Gates company. If the belt pretension of the belt is high enough, we can assume that the belt stiffness is high enough to neglect. Therefore the belt can be modelled as an ideal transmission with friction. The friction of the belt depends highly on the pretension of the belt and environmental parameters (temperature, humidity). Accurate values can only be found by identification.

Transmission Ratio

The transmission ratio of the belt can be calculated by dividing the number of teeth of both pulleys as 60/16 = 3.75.

Motor

The motor is a Maxon RE35 motor with type number 273753. The motor has a torque constant of 38.9 mNm/A, a resistance of 1.23 ohm and an inductance of 0.340 mH. The rotor inertia is 67.6 g.cm2.

Amplifier

The amplification is done using an Elmo Whistle Motor driver. It is basically a PWM-driven current amplifier that operates with a peak and continuous current of 2.25A with switching frequency of 22 kHz. An inductance of 0.220 mH is put in series to increase the motor inductance. The amplifier behaves like an ideal current amplifier for currents up to a maximum of 2.25 A.

Controller Board

The controller board is the TS-7300 of Technologic Systems. This is compact single board computer based upon the Cirrus EP9302 ARM9 CPU and set of on-board peripherals. The board is equipped with an SD-card that contains the Operating System. The TS-7300 is connected with the load encoder, the motor encoder, the PWM amplifier and the red and green LED's.

- Encoders: Both the load encoder and the motor encoder are connected with counting logic. The results are available as counting signals with 2000 counts per revolution.
- PWM output: The PWM (Pulse Width Modulation) output signal has a range of -1 to +1. The signal is converted by the on-board logic into a PWM signal that is connected to the amplifier. As a result the PWM output signal of -1 will result in an amplifier current of -2.25A and a signal of +1 results in an amplifier current of +2.25A. Values in between are unidirectional PWM-fits. For low and medium frequencies the amplifier can be regarded as a current amplifier with an amplification factor of 2.25 and a maximum of 2.25A.
- LED's: Two on-board LED's (read and green) can put on or off. An output value larger than 0 will make a LED shine. For zero values and negative values the LED will be powered off.

The supply voltage of the controller board is 5V. This voltage is supplied by a voltage regulator that is located next to the amplifier. The ARM-9 CPU of the board uses RTAI Linux as operating system to run the controllers code. The maximum sample frequency of the controller code depends on the complexity of the controller. In practice the upper limit is 2kHz for simple controllers and 200 Hz for more complex controllers.

Power Adapter

The power adaptor requires an input voltage of 100 to 240V alternating current with a frequency of 50 or 60 Hz. The adaptor gives a DC output voltage of 24 V and a maximum current of 2.5 A.

Cross Cable

A cross cable is available to connect your PC to the Torsion Bar directly, without the need of a network switch.

2.2 Installation

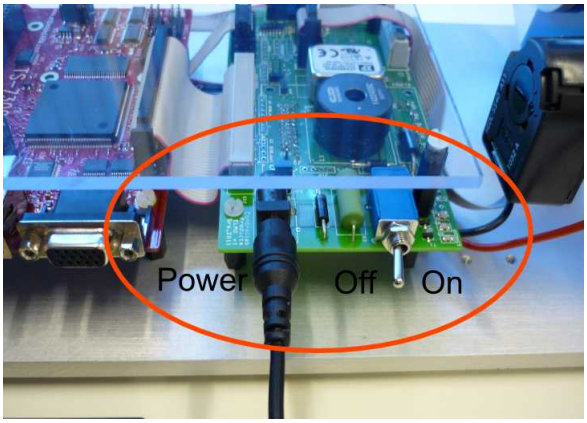
For the proper operation of the Torsion Bar, a PC computer is required that is connected to the Torsion Bar either directly with a cross cable or via a network. On the PC computer, the software packages 20-sim Professional and 20-sim 4C have to be installed. This section describes how to make the various installations. In the next sections we will run two models to test if the Torsion Bar is working properly.

Power

1. Check if the correct power adapter was supplied.

The Torsion Bar is delivered with a power adaptor that yields 24 V output. Depending on the country, a 220 V or a 110 V input is used. Check if the power adaptor has the correct input voltage.

2. Plug in the power adapter.



Plug in the power cable just below the motor. Right of the power connector is the power switch.

The output of the power adaptor can be connected just below the motor. The input of the power adaptor has to be connected to you local 220 V or 110 V supply.

3. Switch on the power.

The power switch is located right to the power cable connector. If you switch it on some LED's on the controller board should be turned on.

Cross Cable

For the communication between the Torsion Bar and the host PC you can use a cross cable or a network cable. A cross cable is a direct connection with the PC. A network cable is indirect connection because it connects the Torsion Bar with a network switch or other network components. Using a cross-cable to connect the host PC to the Torsion Bar is the recommended approach for first time users.

1. Plug the cross cable in the most left connector of the controller board.

The left connector is used for the network connection.

2. The other end of the cross cable must be plugged into your PC computer.

It may take several minutes before your PC will notice that a cross cable is connected. So be patient if 20-sim 4C cannot make a connection immediately.

Network Cable

1. Plug in the network cable in the most left connector of the controller board.
2. The other end of the network cable must be plugged in into a switch or hub of the network.

Use a 'normal' network cable to connect the Torsion Bar Setup via a switch or hub to your network. If your network provides a DHCP server the configuration is automatic. If your network does not provide a DHCP server make sure your network is configured in the 192.168.1.x range and that your PC computer has a proper connection to the network.

Software

On the PC computer two software packages have to be installed: 20-sim 4.1 Professional or higher and 20-sim 4C 2.0 or higher:

1. Install 20-sim 4.1 Professional or higher. For installation guidelines, see the 20-sim Getting Started manual.
2. Install 20-sim 4C 2.0 or higher. For installation guidelines, see the 20-sim 4C Reference manual.

20-sim Models

In the examples, 20-sim models are used to generate C-code. These models have to be copied onto your PC computer.

1. Copy the example files from the CD-Rom (e.g. *D:\Torsion Bar\20-sim*.**) to a local folder (e.g. *Documents\Torsion Bar\20-sim*.**).

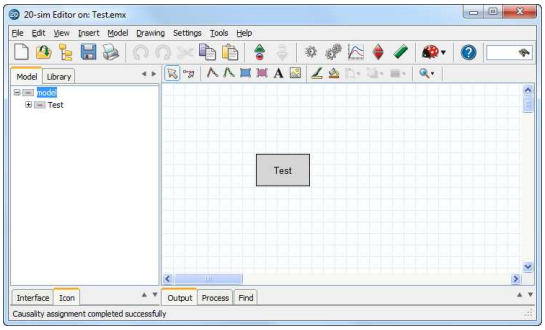
Please check that you have read and write access to the local folder.

2.3 Running a Simple model

To test if the installation was successful, we will run two example models in 20-sim and generate C-code that will be transferred to 20-sim 4C to run on the Torsion Bar. The first model is called *Test.emx* and has no external inputs or outputs.

Exporting the 20-sim Model to 20-sim 4C

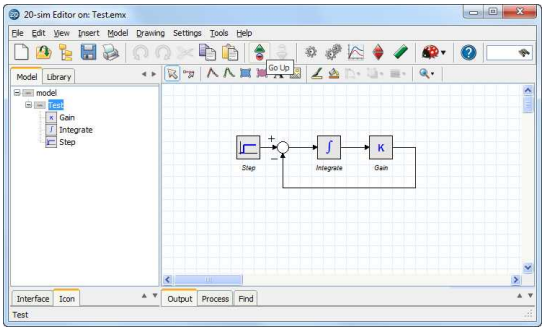
1. Open 20-sim and load the model *Getting Started\Test.emx*. Please check the installation section if you can not find this model.



The model Test.emx loaded in 20-sim.

The model *Test.emx* contains one submodel (named Test). As you can see it is a block diagram model of a first order system, that contains no external inputs or outputs.

2. Select the submodel Test and from the Model menu select Go Down.



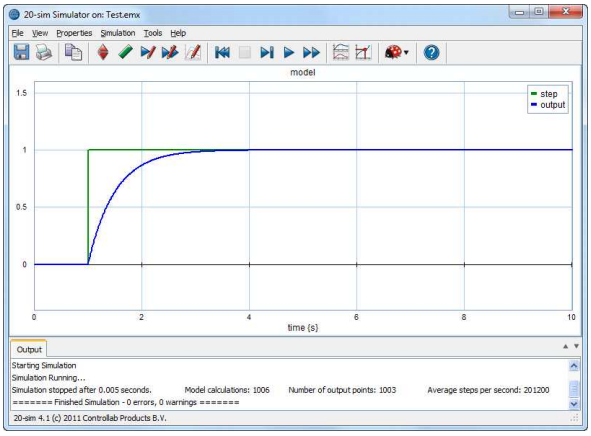
The submodel Test contains a block diagram model with no external inputs or outputs.

3. From the Model menu select Start Simulator.

Now the 20-sim Simulator opens showing an empty plot.

4. In the Simulator from the Simulation menu select Run to start a simulation.

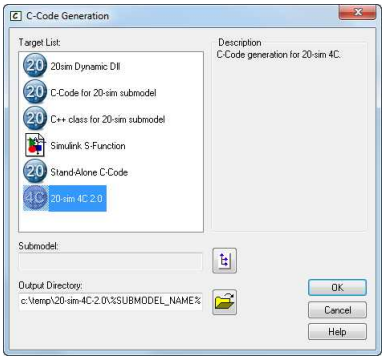
You should see a stepwise change in the setpoint signal and a first order response of the system.



The simulation results of the model Test.emx.

We will generate C-code from this model and run it on the Torsion Bar.

5. In the Simulator, from the Tools menu select Real Time Toolbox and C-code Generation. The C-code Generation dialog will pop-up.

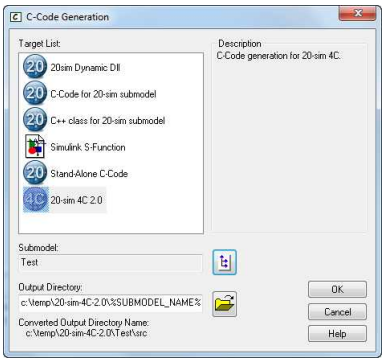


The C-code generation menu in 20-sim.

If you do not see the 20-sim 4C target, please have a look at the Troubleshooting section.

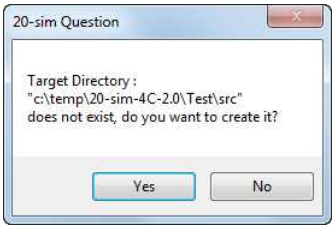
6. Select the 20-sim 4C 2.0 Target.

7. Click on Submodel button  and select the submodel *Test*. The C-code Generation dialog should look like:



The C-code generation menu in 20-sim.

8. Click the OK button to close the dialog.
9. Now the model will be exported to 20-sim 4C as C-code. All code files are stored in a temporary folder. A dialog may appear asking you to open such a folder:

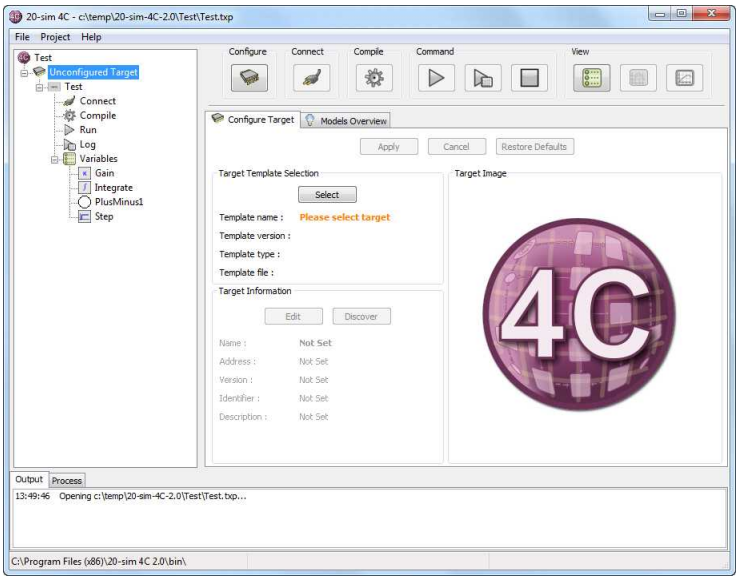


The folder for temporary code files.

10. Click Yes to create the folder (or go to step 6 to enter another directory).

20-sim 4C Settings and Monitoring

If everything works fine, 20-sim 4C will be opened, with the *Test* model loaded. If 20-sim 4C does not open, please check the troubleshooting section. We will enter the settings to get the model running on the Torsion bar and choose the variables that we would like to monitor.

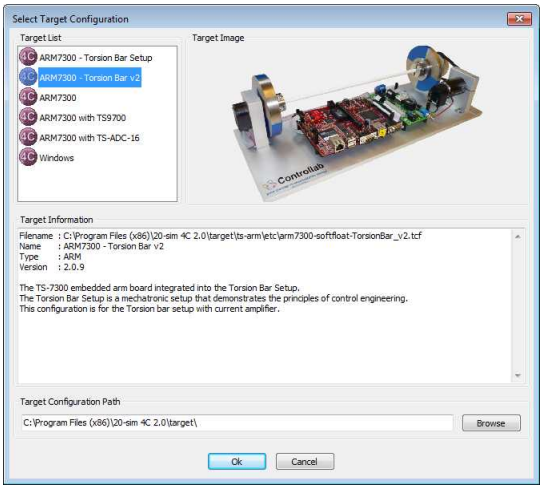


20-sim 4C.

Configure

11. Click on the Configure button  .

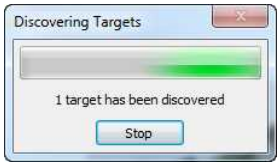
12. Click on the Select button, select the ARM7300 - Torsion Bar V2.0 target and click OK.



Select the ARM7300 - Torsion Bar Setup target.

Clicking OK from the previous step should automatically trigger 20sim 4C to search for IP-addresses of connected Targets.

13. If not, Press the Discover button to initiate this action. The IP address of the target can also be entered manually.



Finding the IP-address.

If a firewall is installed, a warning message may pop-up:

A Windows firewall message when you try to find the target.

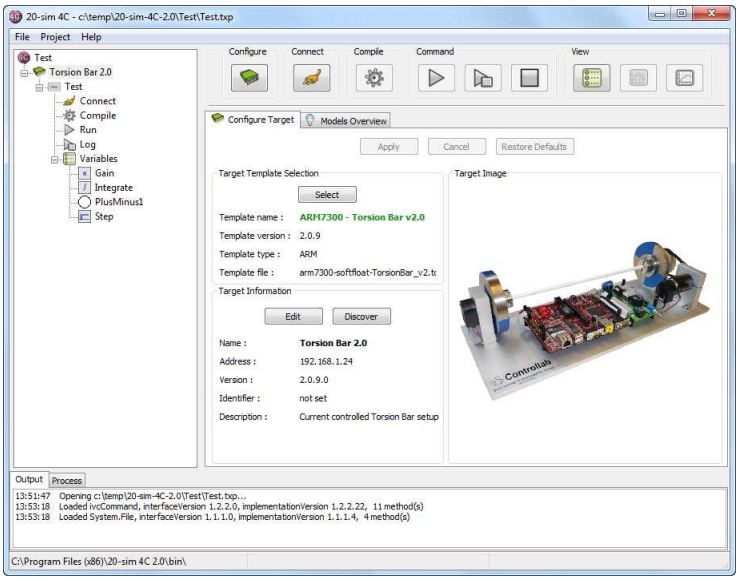
14. Click the Allow access button (or a similar button if another firewall message is shown) if the name of the program is 20-sim 4C and the publisher is Controllab Products B.V. Repeat step number 12.

If nothing happens, please check if your firewall allows 20-sim 4C to be opened.

15. A list of IP-addresses of connected targets will be shown. If more than one target is listed, please choose the correct one.

16. Click on the Apply button.

If everything was installed and connected successfully, the *Configure* button  should turn into green.



The configuration was entered successfully.

If the *Configure* button has turned to red please check the Troubleshooting section.

Connect

17. Click on the Connect button .

18. Our model does not have external inputs or outputs that we have to connect so click the Apply button. Now the connect button should turn green.

Compile

19. Click on the Compile button .

Now the *Compile* button should turn into green. If it has turned into red please check the Troubleshooting section for help.

Command

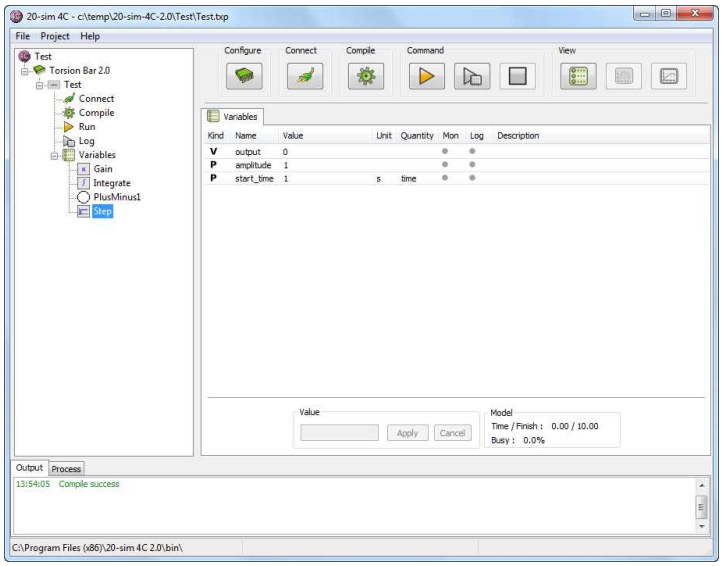
Our model has no external inputs or outputs. To make sure its runs properly we have to monitor its internal variables during running. That is why we will first choose the variables that we will monitor and then run the model.

20. In the tree-view click Variables.

21. Click on the plus sign of the Variables item. Now it will expand showing the available submodels.

22. Click on the Step submodel.

Now a list of two parameters (amplitude and start_time) and one variable (output) should be visible.

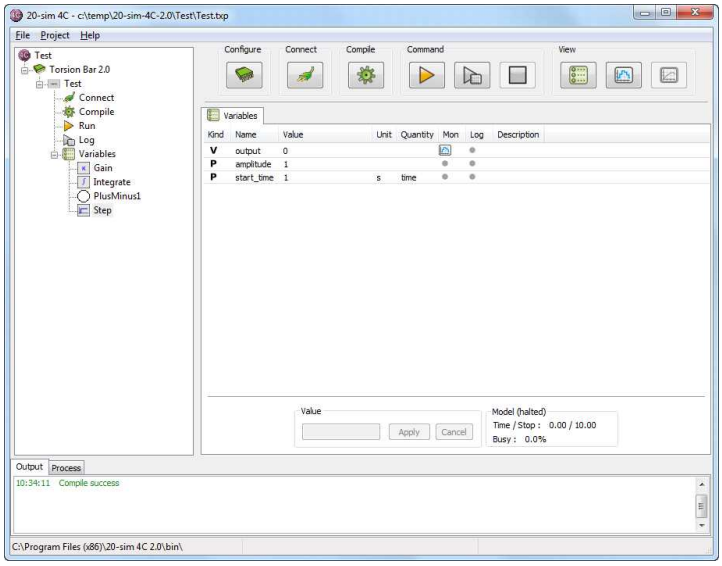


The list of parameters and variables of the submodel Step.

The variable *output* has two gray dots in front. You can click these dots to select the variable for monitoring or logging.

23. With your mouse pointer, go to the variable *output* and click on the grey dot in the *Mon* column to select it for monitoring.

Now 20-sim 4C should look like:



The list of parameters and variables of the submodel Step.

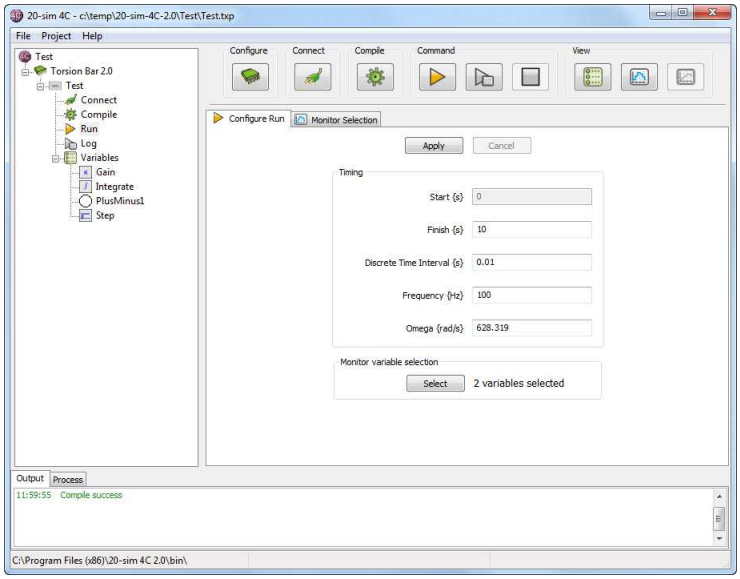
Now we will choose a second variable for monitoring.

24. Click the submodel Gain and select the variable *output* for monitoring.

25. To see the variables that have been chosen for monitoring we have to click the Run item.

26. Now two tabs, Configure Run and Monitor Selection, will appear.

In the Configure Run tab we can set the time of a run and the sample time.

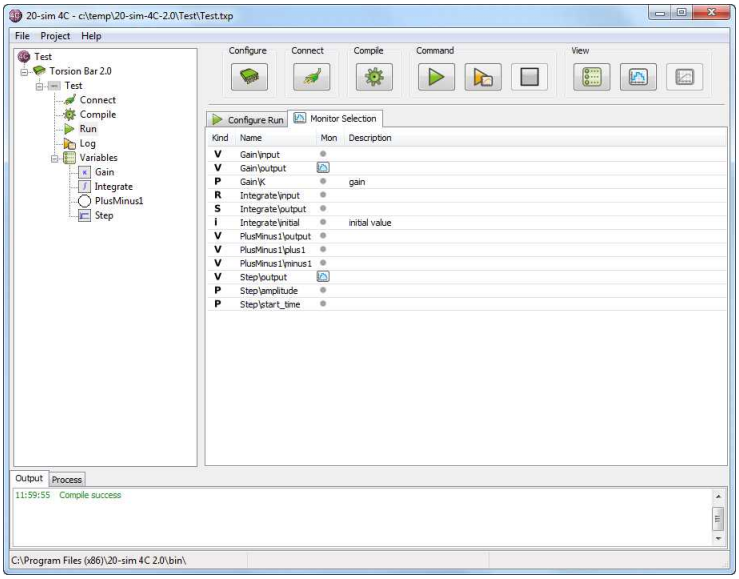


You can set the time of run and clock frequency (sample frequency).

27. Click on Apply to accept the current settings. The Run button  should now turn into green.

28. Click on Monitor Selection tab to see the selected variables for monitoring.

Now 20-sim 4C should look like:



The list of monitored variables.

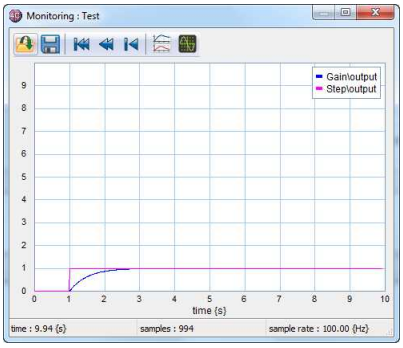
Now we will open a plot and start to run the model on the target.

29. Click on the Monitor button  to show a plot.

Now an external plot will appear.

30. In 20-sim 4C click on the Run button  to start running the model on the target.

Now you should see the following plot after 10 seconds of running. The plot shows a stepwise change in the setpoint signal and a first order response of the system.



The monitored variables, shown in a plot.

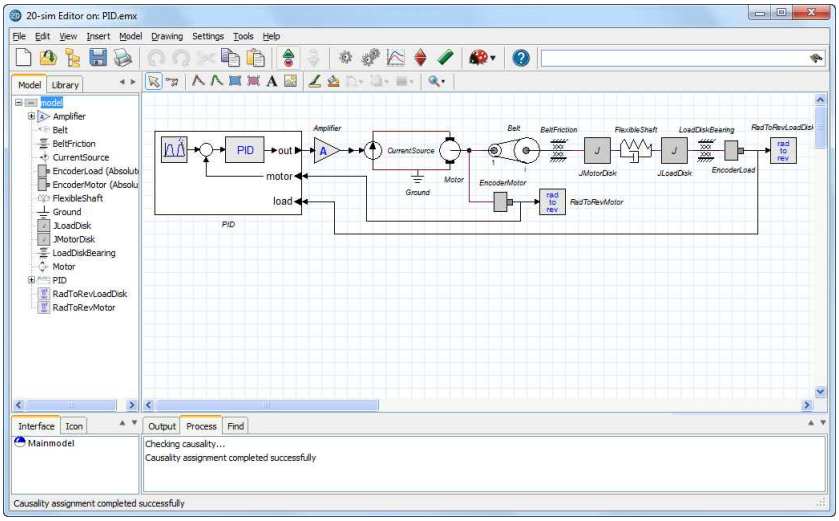
If your results are similar, the Torsion Bar is properly installed. If the results are not similar but some other variable is shown in the plot, check if the correct variables were chosen for monitoring.

2.4 Running the Torsion Bar

The second model that we will use to check if the Torsion Bar is operating well is called *PID.emx*. This is a model of the complete setup with a PID-controller. We will make a fresh start in every section, so make sure that you close all open windows of 20-sim and 20-sim 4C.

Exporting the 20-sim Model to 20-sim 4C

1. Open 20-sim and load the model *Getting Started\PID.emx*.



The model PID.emx loaded in 20-sim.

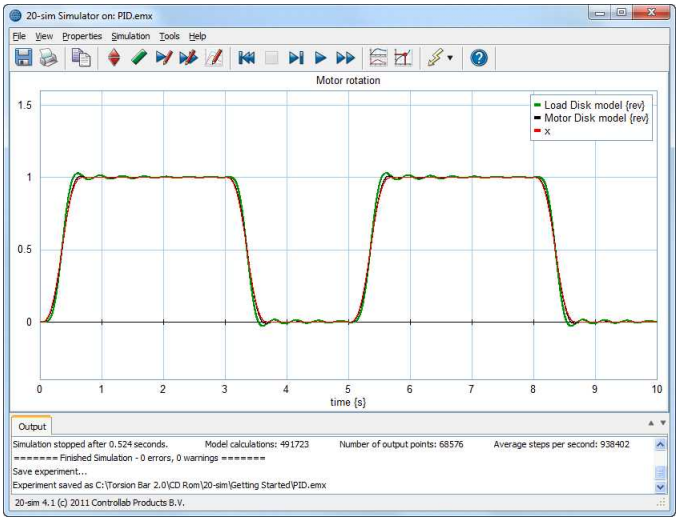
The model PID.emx shows the complete setup with a controller block. The controller block consists of a setpoint generator and a PID controller. The block has two inputs (EncoderLoad, EncoderMotor) and one output (PWM).

2. From the Model menu select Start Simulator.

Now the 20-sim Simulator opens showing an empty plot.

3. In the Simulator from the Simulation menu select Run to start a simulation.

You should see the rotation of the disk close to the motor (Motor Disk) and the disk at the end (Load Disk).



The simulation results of the model PID.emx.

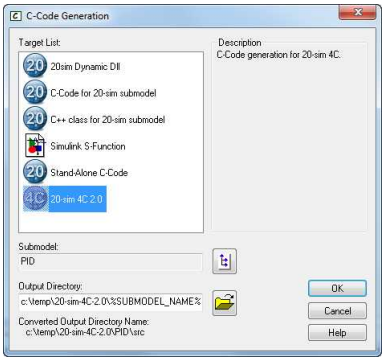
From the controller part of this model, we will generate C-code and run it on the Torsion Bar.

4. In the Simulator, from the Tools menu select Real Time Toolbox and C-code Generation.

The C-code Generation dialog will pop-up.

5. Select the 20-sim 4C 2.0 target.

6. Click on Submodel button and select the submodel PID. The C-code Generation dialog should look like:



The C-code generation menu in 20-sim.

7. Click the OK button to close the dialog.

20-sim 4C Settings and Logging

If everything works fine, 20-sim 4C will be opened, with the PID model loaded. We will enter the settings to get the model running on the Torsion Bar and choose the variables that will be logged during the run.

Configure

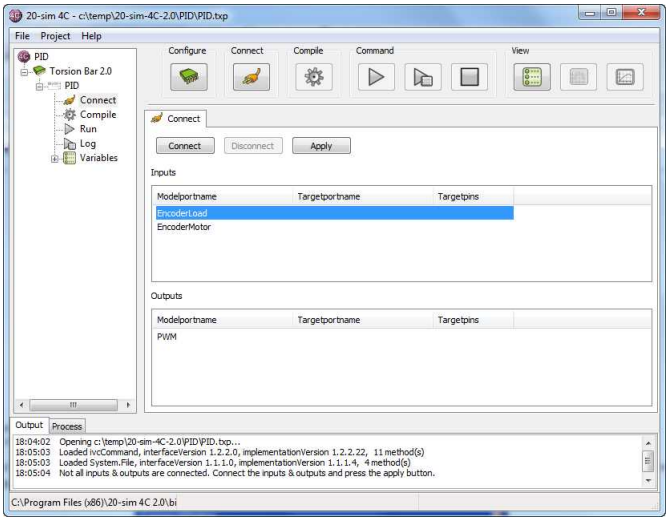
8. Click on the Configure button  .
9. Click on the Select button to select the ARM7300 - Torsion Bar V2.0 target and click OK.
10. Clicking OK from the previous step should automatically trigger 20sim 4C to search for IP-addresses of connected Targets. If not, Press the discover button to initiate this action. IP-address of the target can also be entered manually.
11. Click the Apply button.

Now the *Configure* button  should turn into green.

Connect

The controller model has two inputs (EncoderLoad, EncoderMotor) and one output (PWM). We have to match these inputs and output with the corresponding hardware inputs and outputs.

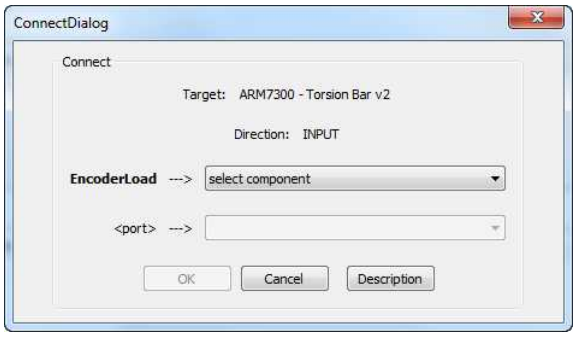
12. Click on the Connect button  .



Connect the inputs and outputs of a model with the inputs and outputs of the target.

13. Select the EncoderLoad input.
14. Click on the Connect button.

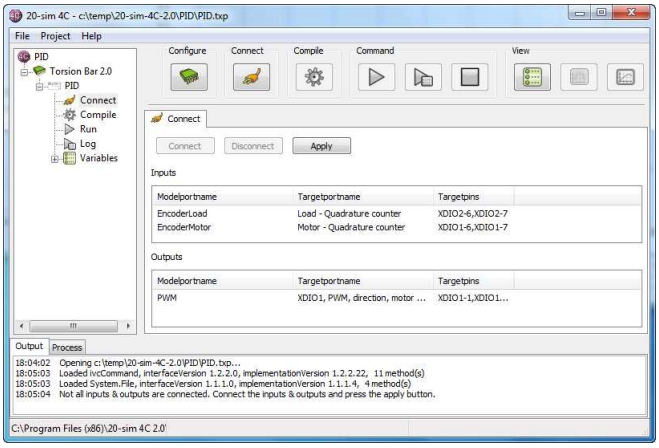
The Connect dialog will appear. In this dialog you can select one out of a set of matching hardware inputs. The first field shows the selected input (EncoderLoad) and the corresponding hardware input (select component).



Use the Connect dialog to choose a matching hardware input or output.

15. Click on the arrow at the right of the first field and from the drop down list choose Load - Quadrature counter.
16. Click OK to close the Connect dialog.
17. Repeat step 12 to 15 and match EncoderMotor with Motor- Quadrature counter.
18. Repeat step 12 to 15 and match PWM with Motor Current.The model inputs and outputs are matched with hardware inputs and outputs.

Now 20-sim 4C should look like:



All connections are now defined.

19. Click the Apply button.

Now the connect button  should turn green.

Compile

20. Click on the Compile button  . It should turn green  .

Command

21. Click on the Run button  .

In the Configure Run tab the run settings are listed. We will leave them as they are.

22. Click the Apply button. The run button should turn green  .

During monitoring, variables are updated much slower than the sample time. To inspect the variables every sample, we have to use the logging facility. We will first choose the variables that should be logged and then run the model.

23. In the tree-view click Variables.

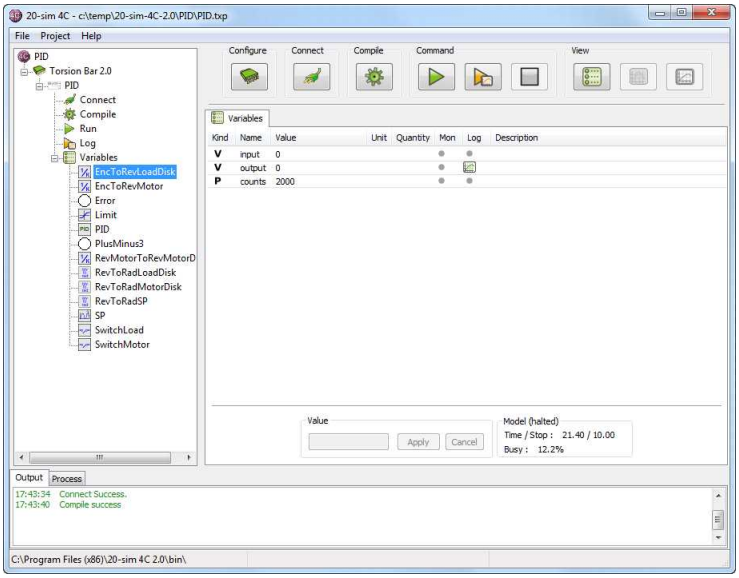
24. Click on the plus sign of the Variables item. Now it will expand showing the available submodels.

25. Click on the RevMotorToRevMotorDisk submodel.

26. With your mouse pointer go to the variable *output* and click on the grey dot to select it for logging.

27. Repeat the same procedure for the submodel *EncToRevLoadDisk* and variable *output*.

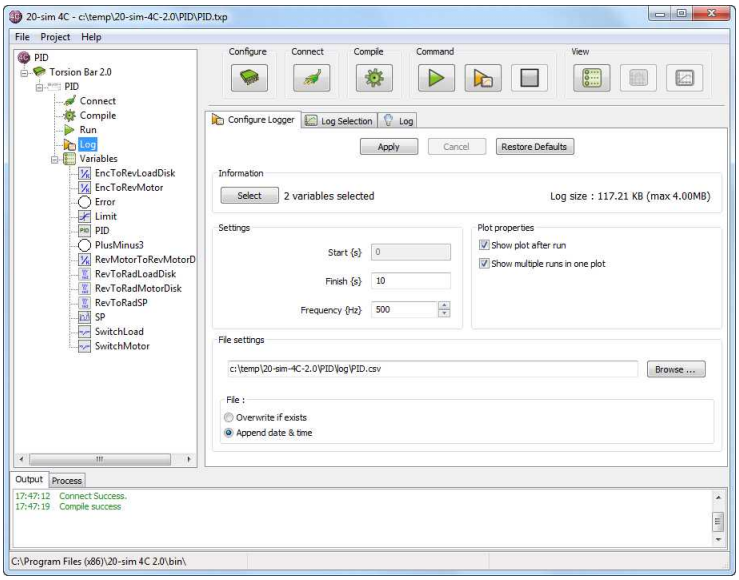
Now 20-sim 4C should look like:



Select the variables that you want to log.

28. In the tree-view click Log.

In this tab, you can set the location of the log-file and some additional logging variables:



You can specify where the log file should be stored.

Now we will open a plot and start to run the model on the target.

29. Click the Apply button.

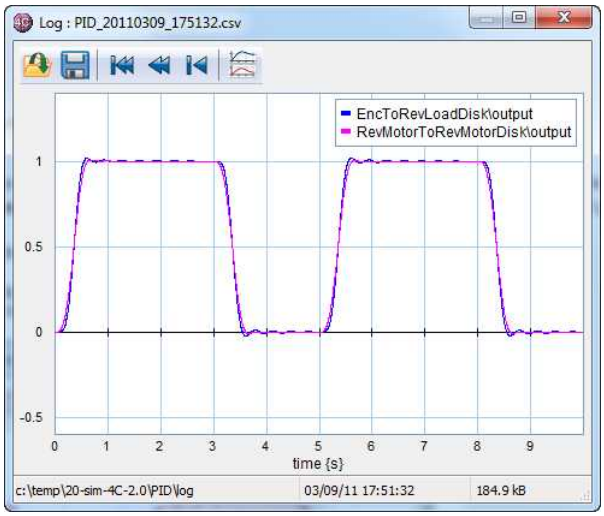
Now the Run with Logging button  should be green. We will run the model and inspect the logged variables after the run.

Note

The model that we will run should turn both disks one turn forward and after a few seconds one turn back. The disk at the end of the torsion bar will show clear vibrations at the end of a turn. During the first runs keep your hand close to the power switch! Immediately Turn off power if you suspect the Torsion Bar is not properly working and check the trouble shooting section.

30. Click on the Run with Logging button  to start running the model on the target.

Now you should see a plot showing the rotation of the motor disk and the load disk.



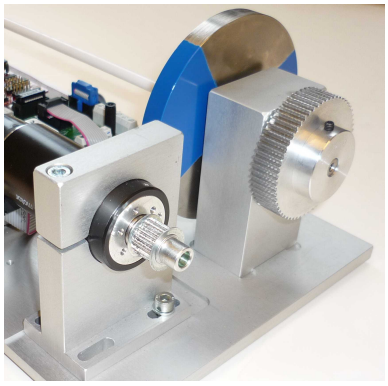
The logged variables, shown in a plot.

If your results are similar, the Torsion Bar is properly working. If the results are not similar but some other variable is shown in the plot, check if the correct variables were chosen for logging. If still the results are not similar, you have to check manually if the encoders are working properly and if the motor is working properly.

3 Testing

3.1 Testing the Components

The components of the Torsion Bar are connected using hexagonal screws. This allows an easy removal and reassembling of the flexible shaft and the belt. We will use this feature to test the proper working of the encoders and the motor.



Various parts of the Torsion Bar can be removed.

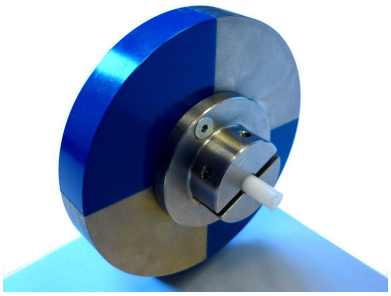
Note

Please take care to turn off the power and remove the power plug before you start assembling or disassembling!

3.2 Removing the Shaft

The flexible shaft can be removed by loosening the clamp of both disks. Without the shaft you can rotate the load disk and check the operation of the load encoder.

- Please turn off the power and remove the power plug before you start assembling or disassembling!
- If you want to run experiments with the shaft removed, make sure that you place the shaft pieces in the clamps to get them balanced for disk rotations.

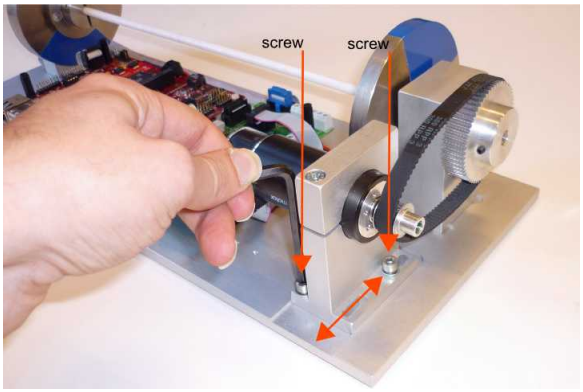


Use a shaft piece to balance the clamp.

- Please take care not to use excessive force when tightening the screws. Otherwise you may damage the threads.

3.3 Removing the Belt

The screws of the motor mount can be loosened so you can slide the motor towards the motor disk. This will release the belt and allow you to remove the belt completely.

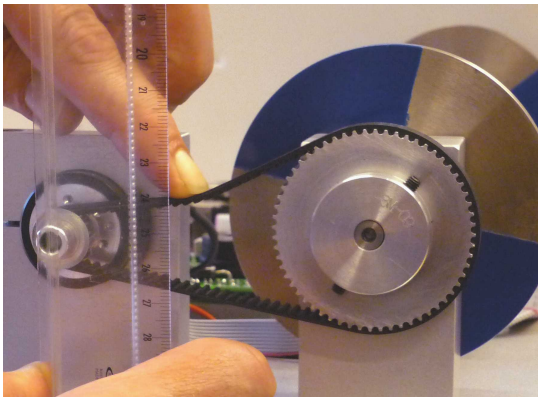


Loosen the motor screws to release the belt.

Without the belt, you can rotate both disks freely. Without the belt you can also do experiments with a free running motor.

- Please turn off the power and remove the power plug before you start assembling or disassembling!
- Please take care not to use excessive force when tightening the motor screws. Otherwise you may damage the threads.

Please pull the motor mount sufficiently when reassembling the belt. Otherwise the belt will have too little pretension and make a very flexible coupling between the motor and motor disk. As a general rule of thumb the belt should not deflect more than 3 mm when you press heavily with a finger.

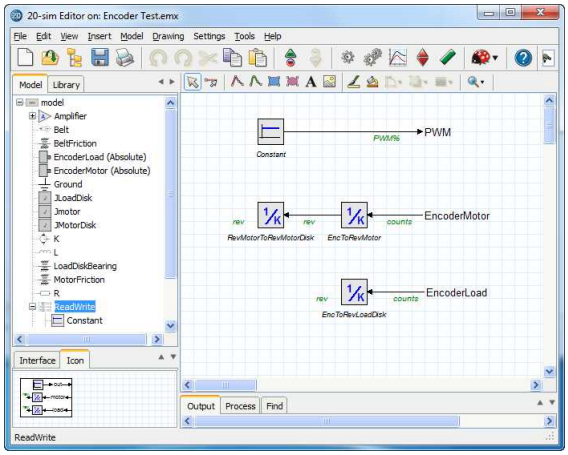


Make sure that the belt deflection is not too large when mounting it back.

3.4 Testing the Encoders

You can test the encoders by removing the flexible shaft, rotating the disks and measuring the response. We will make a fresh start in every section, so make sure that you close all open windows of 20-sim and 20-sim 4C.

1. Remove the flexible shaft.
2. Open 20-sim and load the model *Testing\Encoder Test.emx*.
3. Select the submodel *ReadWrite* and from the Model menu select Go Down.



The submodel ReadWrite has a constant PWM output and reads the encoder signals.

The submodel ReadWrite creates a constant PWM output (in our case a zero output) and reads both encoders. The encoder values are given in counts. By a proper division we get the rotation in revolutions. The motor revolution is divided by the transmission ratio to get the revolution of the motor disk. We will export the submodel ReadWrite as C-code to 20-sim 4C and use it to check the encoders:

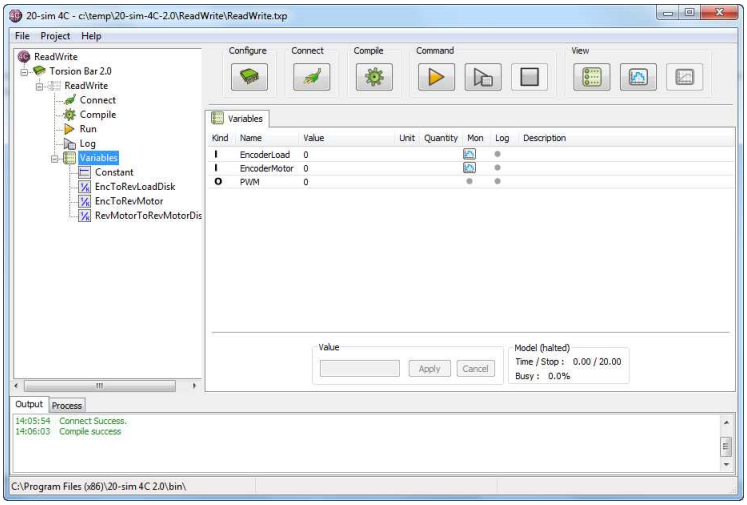
4. Start the Simulator and export the submodel *ReadWrite* to 20-sim 4C.

Now 20-sim 4C will be loaded with the model *ReadWrite* loaded. We will set the target parameters, match the model inputs and outputs to the hardware inputs and outputs, and select the encoder variables for monitoring.

5. Click on the Configure button  and enter the proper configuration parameters. Click on the Apply button and the Configure button turns green.
6. Click on the Connect button  .
7. Match the input *EncoderLoad* with *Load-Quadrature* counter.
8. Match the input *EncoderMotor* with *Motor-Quadrature* counter.
9. Match the output PWM with *Motor Current*.
10. Click on the Apply button and check if the Connect button turns green.
11. Click on the Compile button  and check if the Compile button turns green.

12. In the tree-view click Variables and select the variables *EncoderLoad* and *EncoderMotor* for monitoring.

Now 20-sim 4C should look like:



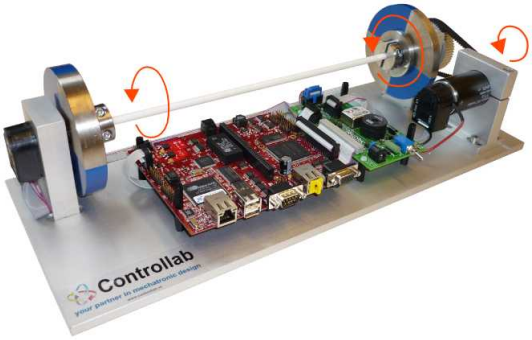
The list of monitored variables.

13. Click on the Run button.

In the Configure Run tab the run settings are listed. We will leave them as they are.

14. Click the Apply button.

Now we will start to run the model on the target. As soon as the target has started, we will (1) rotate the load disk manually and then (2) rotate the motor disk manually! Make sure you rotate both disks in the positive direction and exactly one revolution. Take your time, you have 20 seconds to finish the rotation.



The direction of positive rotation for the load disk, motor disk and motor.

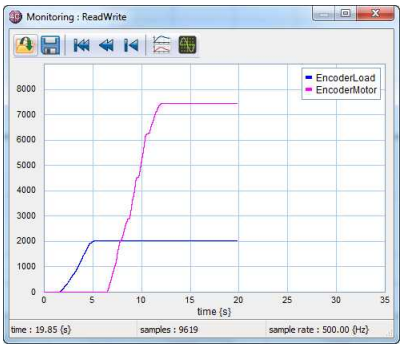
15. Click on the Monitor button  to show a plot.

16. Click on the Run button  to start running the model on the target.

17. Rotate the load disk manually one revolution.

18. Rotate the motor disk manually one revolution.

Now you should see the following plot after 20 seconds of running. The plot shows a change of the load encoder from 0 to about 2000. After the change of the load encoder the motor encoder changes from 0 to about 7500. This means the load disk has made one revolution and the motor 3.75 which is equal to the ratio of the belt drive.



The monitored variables, shown in a plot.

If your results are similar, the encoders are operating properly. If your results are different, please check the troubleshooting section.

19. If you like you can change the selection of monitored variables. Please select *EncoderToRevLoadDisk\output* and RevMotorToRevMotorDisk\output for monitoring. A similar run with both disks manually rotated should result in a plot with both disks changing from 0 to 1 revolution.

20. If you are ready, please take care to put back the flexible shaft.

3.5 Testing the Motor

If both encoders are working properly, you can test the motor by giving it a nonzero PWM output. We will use a small sinusoidal output that will make the motor accelerate and decelerate with moderate velocity.

1. Remove the belt and make sure the motor is fixed.

We will use a similar model as in the previous section to test the motor.

2. Open 20-sim and load the model *Testing\Motor Test.emx*.

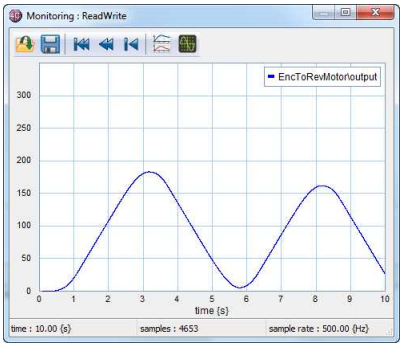
3. Start the Simulator and export the submodel *ReadWrite* to 20-sim 4C.

Now 20-sim 4C will be loaded with the model ReadWrite loaded.

4. In the tree-view click Variables and select the variable *EncoderToRevMotor\output* for monitoring.

5. Click on the Run button  to start running the model on the target.

Now the motor should start to run slowly in the positive direction and then back. You should see a sinusoidal motion after 10 seconds of running.



The monitored motor revolution.

The measured signal may vary from the figure and the simulation, because the rotation is very much affected by the motor friction. As long as you see a similar sinusoidal motion that starts to run in the positive direction, everything is fine and the motor and amplifier are working properly. If your results are totally different, please check the troubleshooting section.

6. Now mount back the belt.

4 Measurement and Control

4.1 Measurement

The Torsion Bar contains two rotary encoders for measurement of the rotation of the load disk and the motor. We will investigate the use of these encoders.

What is the accuracy of the encoders?

Calculation

The encoders give 2000 pulses per revolution. The maximum deviation is half a count. Theoretically the accuracy is therefore equal to:

$$a = \frac{360}{2 \cdot 2000} = 0.09\,[\text{deg}]$$

If we convert this to radians we get an accuracy of:

$$a = \frac{2 \cdot \pi}{2 \cdot 2000} = 0.00157\,[\text{rad}]$$

The load disk has a radius of 100 mm. An accuracy of 0.00157 radians then corresponds to a displacement at the edge of the disk of:

$$a = 0.00157 \cdot 50 = 0.078\,[\text{mm}]$$

A displacement of 1 mm will therefore give about 1 / 0.078 = 13 ticks for the load encoder. The motor is coupled to the motor disk through a belt with a gear ratio of 3.75. A displacement 1 mm will therefore give about 3.75 * 1 / 0.078 = 48 ticks for the motor encoder.

Measurement

1. Run the exercise *Testing the Encoders* of the Testing chapter.
2. Select the variable *EncoderLoad* and *EncoderMotor* for monitoring.
3. Click on the Run button  to start running the model on the target.
4. With your hand turn the load disk a little. How much do you have to turn the disk before the load encoder changes one count?

You will see that a slight touch will be enough to make the encoder change the encoder counts. If you push the outer surface of the load disk 1 mm up, the load encoder will change about 10 ticks. If you do the same with the motor disk, the motor encode will change about 50 ticks.

What is the maximum speed of the encoders?

Calculation

The maximum counting frequency is 18.75 MHz. Theoretically the maximum speed of the encoders is therefore:

$$v = \frac{18.75 \cdot 10^6}{2000} = 9375\,[\text{rev/s}]$$

Measurement

It is not practical to measure the maximum speed of the encoders. Speeds over 20 rev/s could be potentially dangerous for the flexible shaft.

What is the maximum speed of the motor?

Calculation

The theoretical speed of the motor can be calculated with the maximum output voltage and the motor constant:

$$v_{\max} = \frac{24}{0.0389} = 617\,[\text{rad/s}] = 5891\,[\text{rpm}]$$

In practice this speed will be limited due to mechanical friction and power losses in the motor.

Measurement

The maximum speed of the motor can be measured by applying maximum current and logging the motor rotation. In practice, however, most current amplifiers will not run very well for a motor without load. That is why we will not make an attempt to do a measurement run.

4.2 Bang-Bang Control

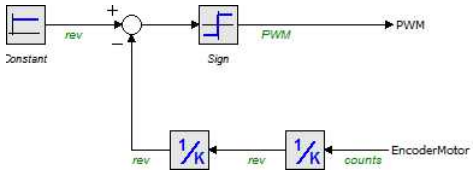
The PWM output to the current amplifier acts as a throttle for the motor. A value of 1 corresponds with the maximum current of 2.25 A. With a motor constant of 38.9 mNm/A that will give a maximum torque of 0.09 Nm. The closer the value is to 1 the higher the torque of the motor. If the PWM output is negative, the torque is in the other direction. We can use this knowledge to make the disks rotate a specific number of rotations.

Theory

Let us assume we want the motor to rotate 1 revolution. A control strategy we could use to accomplish this is:

1. Start with a PWM output of 0.
2. If the motor disk rotation is less than 1, set the PWM output to 1. A positive torque of 0.09 Nm will drive the disks forward.
3. If the motor disk rotation is equal to 1, set the PWM output to 0. No torque applied to the load.
4. If the motor disk rotation is larger than 1, set the PWM output to -1. A negative torque of 0.09 Nm will drive the disks back.
5. Go to step 2.

As you see the output switches between -1 and +1 until the motor disk has reached the desired rotation. That is why this strategy is also known as bang-bang control. We can represent the strategy by the following block diagram.



Bang-bang Control.

The constant block represent the desired rotation of 1 revolution. The actual motor disk rotation is subtracted from this value. The result is an input of the sign function. The sign function gives an output of +1 if the input is larger than 0, an output of -1 if it is smaller than zero and an output if 0 if the input is zero.

Now suppose we haven't reached the desired rotation, then the input of the sign function is a positive value and the output is therefore equal to 1. In a same manner we will get an output value of -1 if the motor disk rotation is larger than the desired rotation. The output of the sign function is 0 when the motor disk rotation is exactly equal to the desired rotation.

Practice

We will investigate the behaviour of bang-bang control using our Torsion Bar. We will not use a controller that switches between -1 (minimum torque) and +1 (maximum torque) because that will cause large accelerations. In stead we will use a controller that switches between -0.2 (moderate negative torque) and +0.2 (moderate positive torque).

1. Open 20-sim and load the model *Measurement and Control\Bang-Bang Control.emx*.

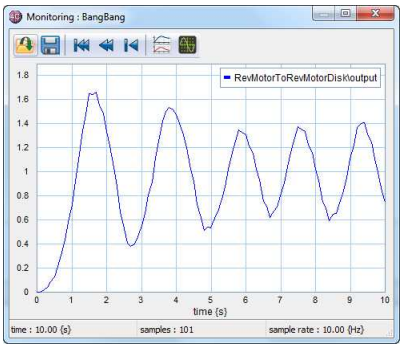
2. Start the Simulator and export the submodel BangBangController to 20-sim 4C.

Now 20-sim 4C will be loaded with the model BangBangController loaded.

3. Make the proper settings in 20-sim 4C (see the Getting Started section to see how) and select the encoder variable *RevMotorToRevMotorDisk\output* for monitoring.

4. Click on the Run button  to start running the model on the target.

Now the motor disk starts to run quickly to about 1.6 rotations, then go back to its initial position, forward again and so on. The system is in oscillation!



The monitored motor disk revolution.

Note: depending on the pretension of the belt, the oscillations that you measure may have a larger or smaller amplitude.

Why is the system in oscillation?

A good start for the investigation of bang-bang control is to monitor the value of the PWM output.

5. Add the variable *PWM* for monitoring.

6. Click on the Run button  to start running the model on the target.

You will see that the PWM output is set to its maximum value of +0.2 at the start. The motor disk starts to speed up until the desired rotation is reached. At that time the motor disk has a high velocity which will decrease because we now have a negative throttle (PWM output = -0.2). At about 1.6 rotations, the motor disk comes to a halt and starts to accelerate in the other direction. The motor disk will pass the desired value and come to about 0.4 rotations. This cycle is repeated on and on.

If you take a close look at the log plot you will see that the motor encoder changes value every 0.1 s. This is because we have set the sample frequency to 10 Hz. If the motor reaches its desired position in the middle of this interval, nothing happens and the motor will continue to speed up too long. That the cause of the oscillation.

What can we do to decrease the oscillation?

The oscillations found with bang-bang control are a combination of a fixed output and limited switching times. This makes the Torsion Bar behave like a ship that can only go at a fixed speed ahead or back and the captain is on the bridge only at fixed time intervals. Every time he is on the bridge he will see if the ship is at its desired position. If the position is reached he will stop the engines. If the ship is too far he will set the throttle to reverse.

There are two thing the captain can do to make the ship stop more accurately. He can decrease the speed of the ship or visit the bridge more often. In our controller this means:

- Decrease the PWM output.
- Increase the sample-time.

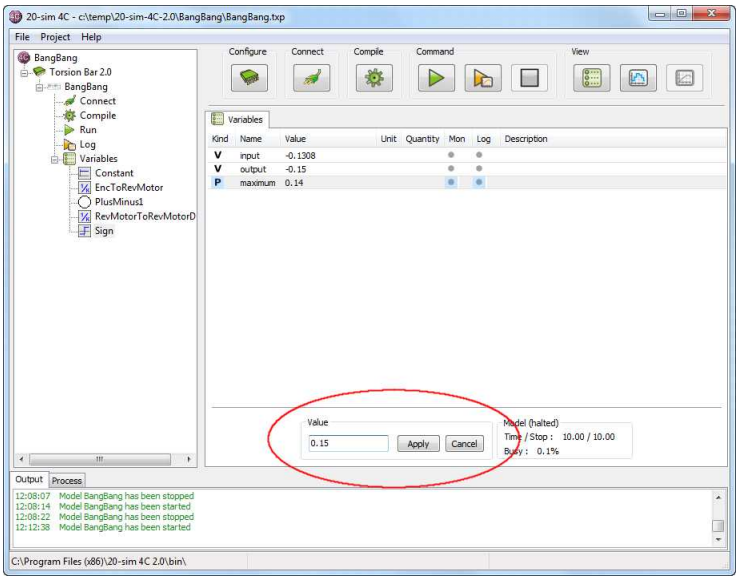
You can do both actions in 20-sim 4C. We will first decrease the PWM output.

7. In the tree-view click Variables.

8. Select the variable Sign / maximum.

At the bottom of the window there is a box named Value.

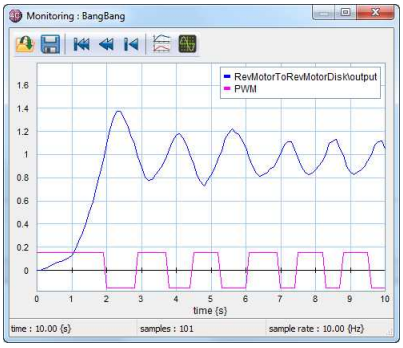
9. You can change the value here from 0.2 to 0.15.



You can change parameter values in the Value box.

10. Click Apply and do another Run.

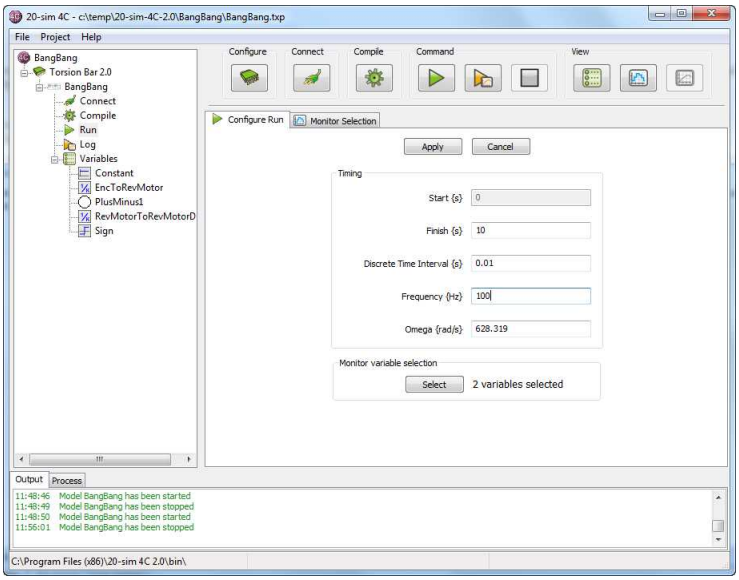
The motor disk will now get to the position more slowly, but the oscillations will decrease in amplitude.



Bang-bang control with a lower output value.

Still the switching times do not change. We will try to tackle this by increasing the sample time.

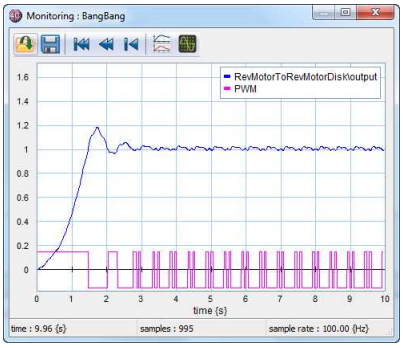
11. Select the Configure Run tab and set the sample frequency to 100 Hz.



You can change the sample frequency in the Configure Run tab.

12. Click on the Apply button and do another Run.

When you investigate the results on the Torsion Bar, you will the that the oscillations will decrease in amplitude and increase in frequency. The oscillations never damp out completely.

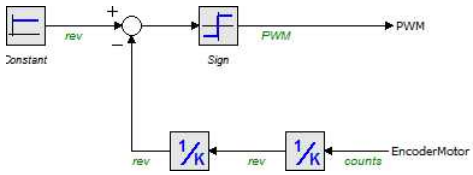


Bang-bang control with an increased sample frequency.

The presented bang-bang controller is a very poor controller. It does not use knowledge of the system and only allows a very limited set of PWM values. In the next topics we will see more advanced controllers in action.

4.3 Feedback Control

Bang-bang control contains features that we see in most controllers. This is clear when we have look at the control scheme.



Bang-bang Control.

Setpoint

The input of the control scheme is the constant block. The output of this block is the desired rotation of the motor, i.e. the goal that we want to achieve. In control theory this is called the setpoint. The setpoint is not restricted to a constant value. It can for example be a signal that ramps up from 0 to 1 and back. The motor will then rotate from zero to 1 and back. A setpoint can be any goal that we want to achieve, varying from motor rotation to motor speed etc. A good setpoint should be something that can be measured, directly or indirectly.

Measured Value

If the goal of a controller is to follow the setpoint, we need some sensor signal to verify how good the system is working. This is called the measured value. The measured value will be related to the setpoint. If the setpoint is the motor rotation, the measured value could be the motor encoder output. In most cases we need to do some calculations to get the measured value in the units of the setpoint. E.g. the encoder ticks have to be divided by 2000 to get the number of rotations. In some cases we can only obtain a measured value indirectly. E.g. if we want to know the rotation of the motor disk, we can divide the rotation of the motor by the belt ratio of 3.75.

Error

In bang-bang control, the setpoint is compared with a measured value. This is done by subtracting the measured value from the setpoint. The outcome is the error between the setpoint and the measured value.

Output

We have used the PWM output signal for the Torsion Bar. The PWM output is a measure of the voltage that is supplied to the motor and can be used to steer the motor forwards and backwards. Every control scheme will have an output signal that is connected to actuators of the system. The actuators should be able to bring the system in the desired state. Otherwise the control scheme will fail.

Controller

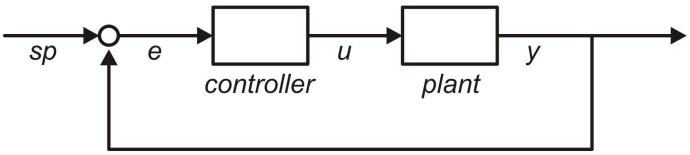
In bang-bang control, the error is transformed into the output by a sign function. This called the controller. A controller can be much more complex than a simple sign function, but every controller should be able to translate an error signal into the an output signal (i.e. the actuator action) that brings the system closer to the setpoint (reduces the error).

Plant

We will give the name Plant to the system that we would like to control. The plant represents the hardware and software that lies between the controller output (u) and the measured value (y).

Feedback Loop

The control scheme with a setpoint, a measured value and an error is called feedback control. This name is chosen because of the feedback loop from the system by using the measured value.



Feedback Control.

A general layout of feedback control is given in the picture. A setpoint signal (sp) is compared with a measured value (y) to give yield an error (e). The controller translates the error (e) into an output (u) which is connected to the actuator(s) of the plant.

Feedback control is so common in control theory, that the control schemes are mostly named after the controller that is used inside the scheme. If we speak about a PID controller in a machine, we mean a scheme with a feedback loop and a PID controller inside.

4.4 Dead Beat Control

We have seen that bang-bang control will lead to oscillations. These oscillations can be decreased by increasing the sample time and decreasing the throttle. Increasing the sample time is not always possible and decreasing the throttle will lead to a slower system. To get a good performing controller we need to change our control scheme.

To find a new control scheme we will start with the metaphor of our ships captain. Bang-bang control is like a ship that can only go full speed ahead or full speed back and the captain is on the bridge only at fixed time intervals. Every time he is on the bridge he will see if the ship is at its desired position. If the position is not yet reached he will set the throttle to full speed ahead. If the position is reached he will stop the engines. If the ship is too far he will set the throttle to full reverse.

The first thing the captain could do is to change from full speed ahead to full speed back, way before the desired position is reached. If he does this at the right instant, the ship will come to rest exactly at its desired position. Theoretically this is the fastest way to get a ship from A to B. It is called deadbeat control. In practice there are some drawbacks:

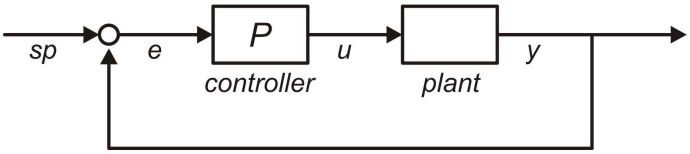
- The captain has to be on the bridge at the exactly the right time, otherwise he will switch too late. This can only be established if he is on the bridge often. In other words, the sample time should be high.
- If the ship has a lot of drag from the water, the theoretical switching time will bring the ship to a standstill way before the desired position. In other words, if the system has unknown effects, deadbeat control will bring the system not exactly to the desired position.
- The ships engines may get damaged if we change from zero to full throttle instantly. Most systems have this problem. They behave much better if we pull the throttle lever gently.

This makes deadbeat control only useful for very robust systems which are predictable and have a control system with a high sample frequency.

4.5 Proportional Control

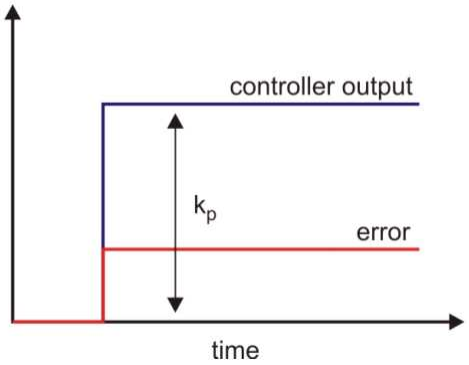
Theory

We continue with the metaphor of our ships captain. We have seen that bang-bang control will lead to oscillations, and that dead beat control is undesirable. Another thing the captain could do is to set the throttle in proportion to the distance that should be covered. If the ship is far away from the desired position, he will set the throttle to full ahead. If he comes closer to the desired position, he will release throttle to gently reduce the ships speed. If the ship goes to far he will do the opposite: make the ship go backwards gently. The scheme for this control strategy is shown in the next figure and is called proportional control.



Proportional Control.

The controller of a proportional control scheme gives an output that is proportional to the error:



The controller output is in proportion to the error.

Using formula's the controller output u is equal to:

$$u = k_p \cdot e$$

The parameter kp is called the proportional gain. It determines how the error e will lead to an output u.

Practice

We will investigate the behaviour of proportional control using the Torsion Bar.

1. Open 20-sim and load the model *Measurement and Control\Proportional Control.emx*.

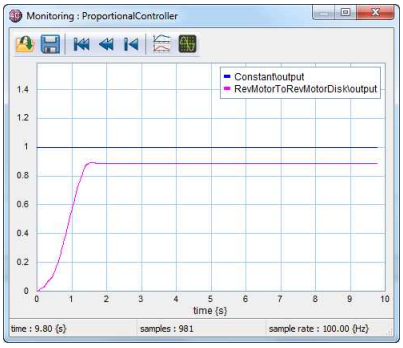
2. Start the Simulator and export the submodel ProportionalController to 20-sim 4C.

Now 20-sim 4C will be loaded with the model *ProportionalController* loaded.

3. Make the proper settings in 20-sim 4C (see the Getting Started section to see how) and select the variables *Constant\Output* and *RevMotorToRevMotorDisk\output* for logging.

4. Click on the Run with Logging button  to start running the model on the target.

Now the motor should start to run almost one rotation and then come to a halt.

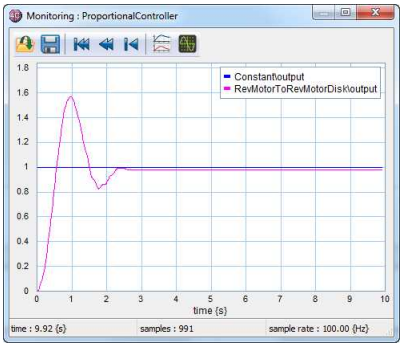


Proportional Control with a gain of 0.2.

As you can seen from the log plot, the motor comes to a halt at almost the desired setpoint. We can try to increase the accuracy by increasing the proportional gain.

5. Change the value of the parameter *Gain\K* to 0.5 and do another run.

You should see something like the next plot. The motor disk is now going too far and then runs back until it reaches a stable value of around 1 rotation.

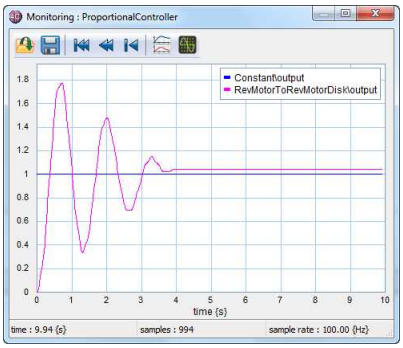


Proportional Control with a gain of 0.5.

The setpoint, one rotation, will never be reached exactly. We can try to improve this error by increasing the proportional gain even more.

6. Change the value of the parameter *Gain\K* to 1.0 and do another run.

Now you will see results like the next figure: the Torsion Bar starts to oscillate.



Proportional Control with a gain of 1.0.

For small values of the proportional gain, the measurement will be different from the simulation. This is caused by non-linear friction in the belt, that is not part of the model. In the chapter on identification, we will look at this friction. Nevertheless for both the simulation model and the Torsion Bar setup, we find that if we increase the proportional gain:

- At first the error will be reduced.
- Eventually oscillations will occur and things go out of hand.

5 Identification

5.1 Introduction

In the previous chapters we have used a 20-sim model to generate the C-code for 20-sim 4C. These models consists of a controller and a representation of the Torsion Bar. If you have tried to run a simulation in 20-sim, you have noticed that the models predict the behaviour of the Torsion bar pretty well. This allows us to test any new controller on the model first, before trying it out on the hardware. This is in fact what we have done. We started with a good model of the Torsion Bar and then tested all controllers with this model. When we got all the errors and mistakes out, we tried it for real. A good model of a machine is therefore very useful for the design of a controller that should work on the machine. Some controllers can only be designed if we have a proper machine model!

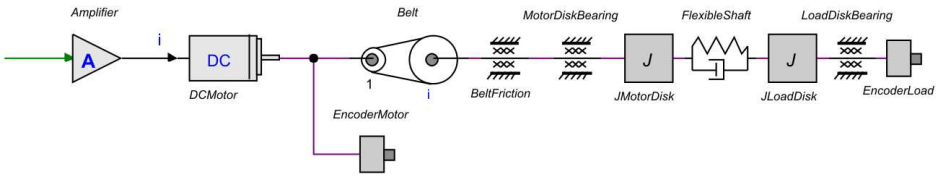
Getting a model that pretty well predicts how a machine will behave is not an easy job. If the machine only exists in the drawing room, we have to apply physics and mathematics to predict how it will behave if it is going to be built. If a machine exists and is available for us to use, we can do experiments to derive a predictive model. This is called identification.

Roughly speaking, there are two methods of identification: component identification and black box identification. With component identification we use dedicated experiments on the machine, math, data sheets, colleagues and all possible means to gather information of the system. With this information we build a model of the system. In black box identification we assume there is no information at all. We give the system a stimulus and measure the response. Out of both we try to create a mathematical model of the system.

In this chapter we will explain how you can do component identification to obtain a model of the Torsion Bar. Most of our effort we will spend on component identification, because a lot of knowledge on the various component of the Torsion Bar is available. Black Box identification requires a lot of mathematical knowledge. Therefore we only show how it can be done but do not go into details.

5.2 Component Identification

In the previous chapter, we have seen that friction plays an important rol in the behavior of the Torsion Bar setup. The models that we have used, only contain viscous friction models, which are not very accurately describing the real behaviour. In this chapter we will look at the all the components of our Torsion Bar model, and find out if they accurately describe the real setup.



General model of the Torsion Bar.

The model that we will use as a starting point is shown above. The creation of such a model is called modeling. Creating good models requires skill and a good mathematical background. The kind of model that we make of a system, depends on the accuracy that is required. The more accurate a model should be, the more complex it gets and the more time we need to identify all components. The model that is presented above is a good trade-off between accuracy and complexity.

The components of this model are:

- amplifier: An amplifier converts the controller output, which is a low power data signal, into a high power output voltage.
- motor: 20-sim has a special toolbox that can generate motor models from datasheet parameters. We have used the datasheet parameters of the Maxon RE35 273753 DC-motor to generate a motor model.
- motor encoder: the motor encoder counts the motor rotation.
- belt: the belt transfers the motor power to the motor disk.
- motor disk: the motor disk is the first load for the motor.
- flexible shaft: the flexible shaft transfers some of the motor power to the load disk.
- load disk: the load disk is the second load for the motor.
- load encoder: the load encoder counts the rotation of the load disk.

5.3 Encoders

A detailed description of the encoders is given in the introduction. Both encoders give a rotation in counts and use 2000 counts per revolution. The encoders are so light, compared to the motor and load disk, that we can neglect their inertia and friction. A good encoder model is available in the 20-sim library.

5.4 Motor

The servo motor is modelled usign the 20-sim Servo Motor Editor. The 20-sim Servo Motor Editor is a tool to generate dynamic models of servo motors for the use in 20-sim. These models describe the complete dynamic behaviour of servo motors, including the electrical, mechanical and thermal behaviour. The dynamic models are generated automatically from data files containing commercially available motors, but you can also enter your own motor parameters.

The motor that is used in the Torsion Bar is Maxon RE35 motor (type number 273753). The motor parameters can be found in the data sheets of Maxon Motors:

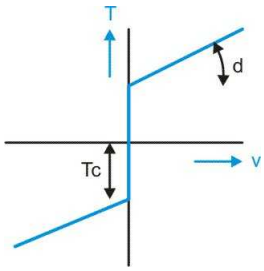
Resistance	1.23 ohm
Inductance	0.034 mH
Motor constant	38.9 mNm/A
Inertia	6.96e-6 kg.m2
damping	4e-7 Nm.s/rad
Friction	4.0e-4 Nm/rad

The inductance of the motor is very low (0.034 mH). That is why an additional inductance of 220 mH has been connected to the motor wiring to get an improved response. To compensated for this inductance we have increased the value that we have entered in the Servo Motor Wizard.

5.5 Load Disk

Theory

The load disk is modelled as a rotational inertia and friction caused by the bearings. The rotational inertia can be calculated (1.37e-3 kg.m2) but the friction is unknown. Friction is hard to model in detail because of the strong dependency on temperature, humidity and lubrication. We will use a relatively simple friction model with only viscous and coulomb friction.



The friction torque as function of the disk speed.

The corresponding parameters are:

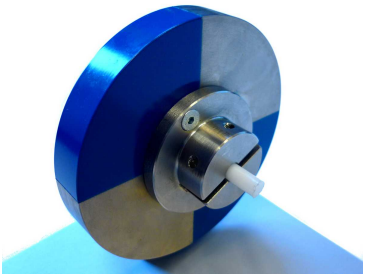
d (N.m.s/rad)	damping (viscous friction value)
Tc (N.m)	coulomb friction torque

The values of these parameter are unknown. We have to do an experiment to find them. The simplest experiment is to remove the flexible shaft and give the load disk a twist. The disk will start to rotate with a certain speed which will get lower and lower due to the friction. If we measure the rotation and compare that with a simulation, we could use the simulation to find the parameter values. The proper parameters values are found when the simulation results match with the measurements.

Practice

We will remove the flexible shaft, so the load disk can rotate freely.

1. Remove the shaft and insert a shaft piece in the Load disk to balance it.



Use a shaft piece to balance the clamp.

2. Open 20-sim and load the model *Identification\Load Disk Rotation.emx*.
3. Start the Simulator and export the submodel Measurement to 20-sim 4C.

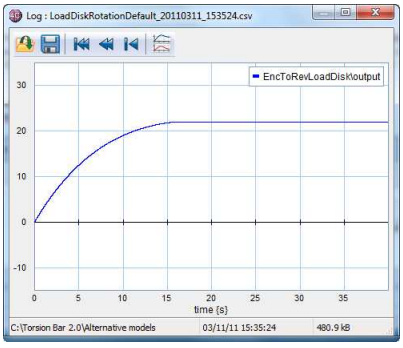
Now 20-sim 4C will be loaded with the model Measurement loaded.

4. Make the proper settings in 20-sim 4C (see the Getting Started section to see how)
5. Select the encoder variable *EncoderToRevLoadDisk\output* for logging.
6. Remember the location and name of the log-file. We are going to use it later on.

We will give the load disk a twist so it starts spinning. We will measure the rotation of the disk from its initial speed until it comes to a halt. Try to give the disk a rotation that will bring it to a halt just before the logging stops. The experiment gives us 40 seconds for logging, so you might try several times before you get it right. In the plots we show a disk that has a positive speed.

7. Give the load disk a twist so it starts spinning.
8. Click on the Run with Logging button  to start running the model on the target.

After 40 seconds a plot should appear like the next plot:



Log plot of rotating load disk.

If your plot shows negative values, you have twisted the load disk in the negative direction. This is not a problem. You can use it equally well to identify the friction parameters. We will go back to 20-sim to compare our log-file with a simulation model

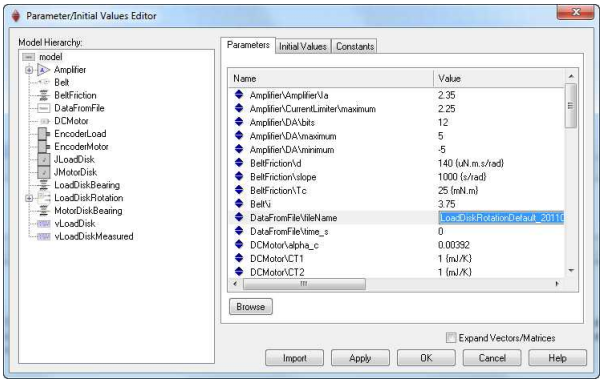
9. Open 20-sim and load the model *Identification\Load Disk Rotation.emx*.

As you can see, the model contains a file input block (DataFromFile) to compare the simulated load disk rotation with the measured rotation. The shaft has been removed in the model, just like on the real setup. A default measurement is used that will be replaced by your own measurement.

10. Start the Simulator.

11. In the Simulator, from the Properties menu select Parameters.

12. Select the parameter DataFromFile\filename.



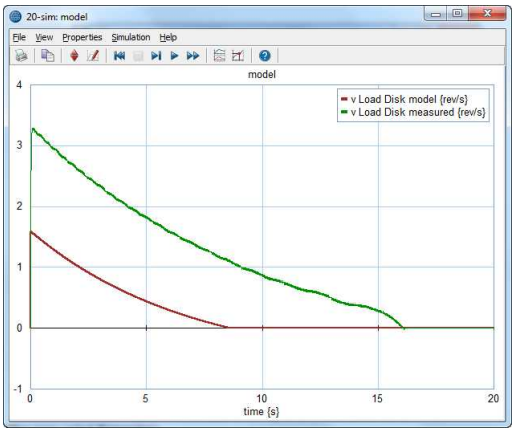
In the Parameters/Initial Values Editor you can change the filename of the log file.

13. A Browse button will be visible. Click it to find the name and location of our log-file (see step 7).

14. Click OK to close the Parameters/Initial Values Editor.

15. From the Simulation menu select Run to start a simulation.

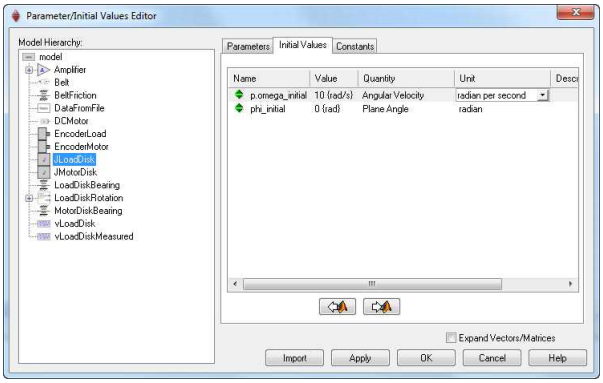
Two plots are shown, one of the load disk rotation and one of the load disk speed. We will focus on the speed plot since this the most handy plot for curve fitting. As you will see the initial speed of the model is 1.5 revolution per second. This is different from the speed of the measurement, which is 3.2 revolution per second in our example.



The speed of the measured disk (green) and simulated disk (brown).

16. From the Properties menu select Parameters.
17. Select the initial value *JLoadDisk\p.omega_initial*.

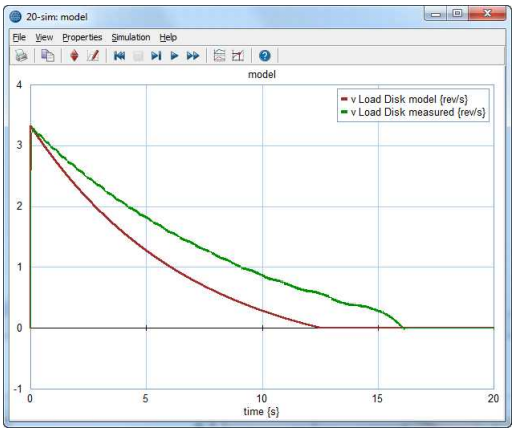
Note: Initial values a shown in the second tab as shown in the next figure. You can select the unit of rotational speed. We will use revolution per second.



You can change the initial speed of the Load Disk.

18. Give the initial value JLoadDisk\p.omega_initial the same starting value as your measurement.
19. Click OK to close the Parameters/Initial Values Editor.
20. From the Simulation menu select Run to start a simulation.

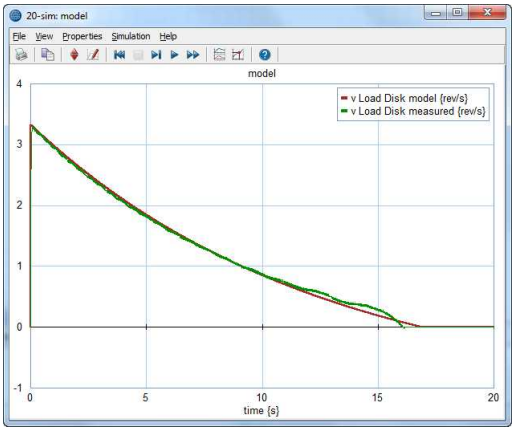
Now the starting speed of the simulated disk and measured disk should be equal. We will change the friction values to make the curve of the simulated fit precisely on the measured curve.



Equal starting speeds for the measured and simulated disk.

21. From the Properties menu select Parameters.
22. Select the parameter *LoadDiskBearing \ d* and change its value.
23. Select the parameter *LoadDiskBearing \ Tc* and change its value.
24. Click OK to close the Parameters/Initial Values Editor.
25. From the Simulation menu select Run to start a simulation.

The simulated curve will be different from the previous simulation. As you will see it is better to change one parameter value at a time and see the effects before changing the other parameter value. In the end you will find the values that make the simulated curve fit the measured curve pretty well.



Equal curves for the measured and simulated disk.

Friction is very hard to model. Parameter values are strongly depending on the age of a bearing, the lubrication, temperature, humidity etc. Typical values are shown in the next table:

	min value	max value
d (μN.s.m/rad)	50	200
Tc (μN.m)	500	1500

Typical friction parameter values for the load disk bearing.

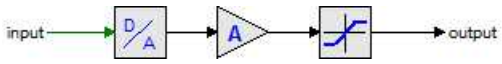
26. If you are ready, please put back the flexible shaft.

5.6 Motor Disk

The motor disk is modelled as a rotational inertia and friction caused by the bearings. The rotational inertia is the combined inertia of the disk (8.69e-4 kg.m2) and the large pulley (3.77E-005 kg.m2) which gives a combined total of 9.07e-4 kg.m2. The motor disk is mounted in the same bearings as the load disk. Therefore we will use the same friction parameters as with the load disk.

5.7 Amplifier

The amplifier is a PWM (Pulse Width Modulation) driven Elmo whistle drive that converts an input signal into a 2.25A output signal. The amplifier behaves like an ideal current amplifier for currents up to 2.25A. The input signal of the amplifier is a PWM signal, that is converted by the amplifier into a current.



The amplifier model.

In our model, we neglect the high frequency switching (20 kHz) of the PWM signal and consider it as a signal with a values between -1 and +1. It is generated by the controller, therefore it is a discrete signal that can change value every sample time. In our amplifier model this signal is converted into a continuous time signal by a DA-converter. The continuous time signal is multiplied by a gain of 2.25 and limited to a value of 2.25 A.

5.8 Flexible Shaft

Theory

The flexible shaft is modelled as a spring damper with two parameters: the rotational stiffness and the viscous damping. The damping is unknown. The stiffness parameter might be calculated as:

$$k = \frac{G \cdot \pi \cdot d^4}{32 \cdot l}$$

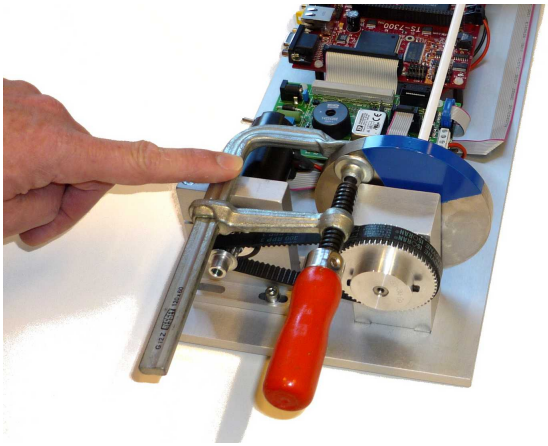
with a shear modulus G = 1.07 (GPa), a diameter d = 6.3e-3 (m) and a length l = 0.295 (m) this will give a stiffness of k = 0.543 (N.m/rad). The shear modulus however, is an approximate value and may be different for our Torsion Bar.

We can conclude that the rotational stiffness and the viscous damping are not (exactly) known. Therefore we have to do an experiment to find the them. The simplest experiment is to clamp the motor disk so it cannot spin and give the load disk an initial rotation. If we release the load disk it will start to spin forwards and backwards many times and gradually come to a halt. If we measure the rotation and compare that with the simulation, we can use the simulation to find the parameter values. The proper parameters values are found when the simulation results match with the measurements.

Practice

1. Open 20-sim and load the model *Identification\Flexible Shaft Twist.emx*.
2. Start the Simulator and export the submodel Measurement to 20-sim 4C.
3. Now 20-sim 4C will be loaded with the model Measurement loaded.
4. Make the proper settings in 20-sim 4C (see the Getting Started section to see how)
5. Select the encoder variable *EncoderToRevLoadDisk\output* for logging.
6. Remember the location and name of the log-file. We are going to use it later on.

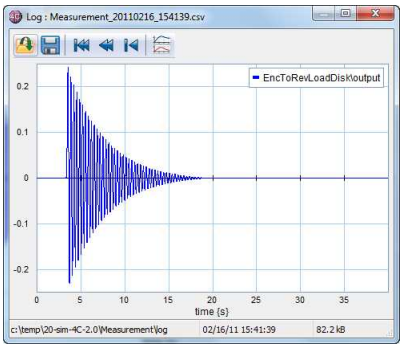
After the logging is started, we will give the load disk an initial rotation. The motor disk should not move. The best way to do this is to use a tool and clamp the motor disk as indicated in the next picture.



Use a tool to clamp the motor disk.

7. Clamp the motor disk to prevent it from spinning.
8. Click on the Run with Logging button  to start running the model on the target.
9. Immediately after the logging has started, give the load disk a rotation of quarter revolution and release it.

The load disk will now start to move forwards and backwards. After 40 seconds the logging is finished and a plot should appear like the next plot:



Log plot of rotating load disk.

If your plot shows negative values, you have twisted the load disk in the negative direction. This is not a problem. You can use it equally well to identify the friction parameters. We will go back to 20-sim to compare our log-file with a simulation model.

10. Open 20-sim and load the model *Identification\Flexible Shaft Twist.emx*.

As you can see, the model contains a file input block (*DataFromFile*) to compare the simulated load disk rotation with the measured rotation. The motor disk is clamped to the ground and a torque actuator gives the load disk a twist just as with the real setup. A default measurement is used that will be replaced by your own measurement.

11. Start the Simulator.

12. In the Simulator, from the Properties menu select Parameters.

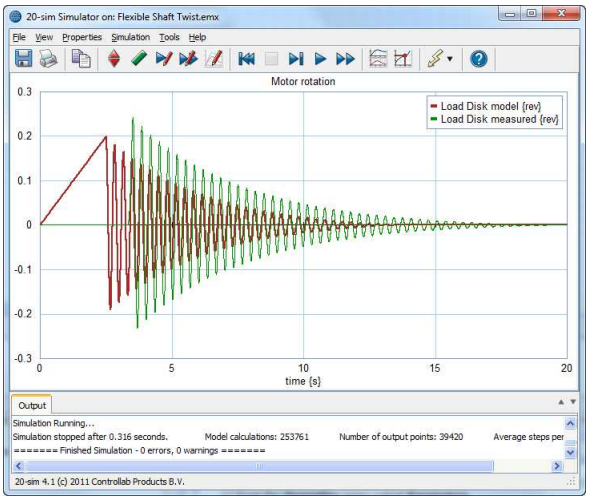
13. Select the parameter DataFromFile\filename.

14. A Browse button will be visible. Click it to find the name and location of your log-file (see step 6).

15. Click OK to close the Parameters/Initial Values Editor.

16. From the Simulation menu select Run to start a simulation.

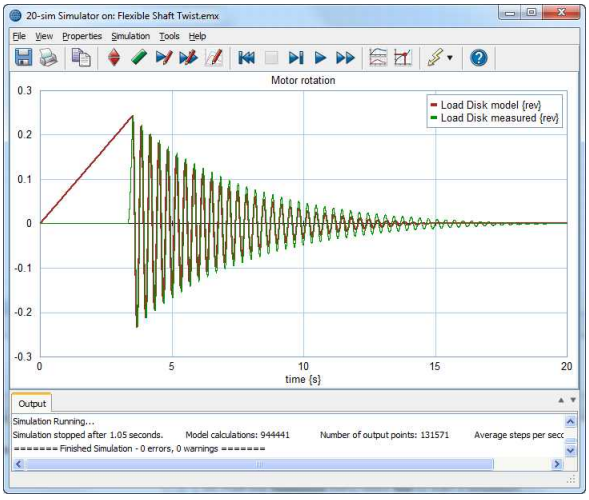
A plot is shown of the measured and simulated rotation of the load disk. As you will see the initial twist of the load disk in the model is a quarter revolution. The disk is released at 5 seconds. This is probably different from the measurement. That is why we will give the simulation model the same twist as the measurement and the same release time.



The rotation of the measured disk (green) and simulated disk (brown).

17. From the Properties menu select Parameters.
18. Change the parameter values *TorqueActuator\rotation* and *TorqueActuator\start_time* to the values that where found in the measurement.
19. Click OK to close the Parameters/Initial Values Editor.
20. From the Simulation menu select Run to start a simulation.

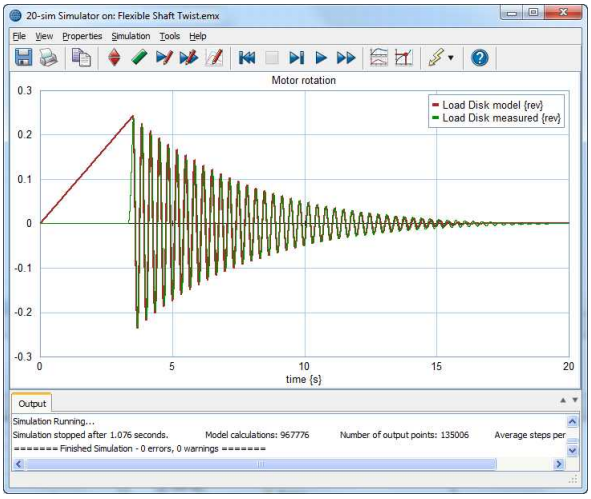
Now the initial twist and release time of the simulated disk and the measured disk should be equal. We will change the stiffness and damping of the flexible shaft to make the curve of the simulated fit precisely on the measured curve.



Equal starting rotation for the measured and simulated disk.

21. From the Properties menu select Parameters.
22. Select the parameter *FlexibleShaft\d* and change its value.
23. Select the parameter *FlexibleShaft\G* and change its value.
24. Click OK to close the Parameters/Initial Values Editor.
25. From the Simulation menu select Run to start a simulation.

The simulated curve will be different from the previous simulation. As you will see it is better to change one parameter value at a time and see the effects before changing the other parameter value. In the end you will find the values that make the simulated curve fit the measured curve pretty well.



Equal curves for the measured and simulated disk.

The shear modulus and damping can vary with the temperature, humidity etc. Typical values are shown in the next table:

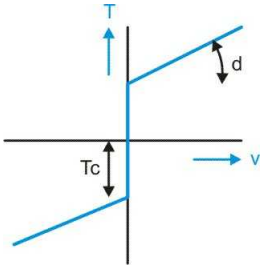
	min value	max value
d (μN.s.m/rad)	400	600
G (Gpa)	1.05	1.15

Typical parameter values for the flexible shaft.

5.9 Belt

Theory

The belt is modelled as an ideal transmission with friction. The friction is modelled with the same friction model that was used for the load disk, with only viscous and coulomb friction.



The friction torque as function of the disk speed.

The corresponding parameters are:

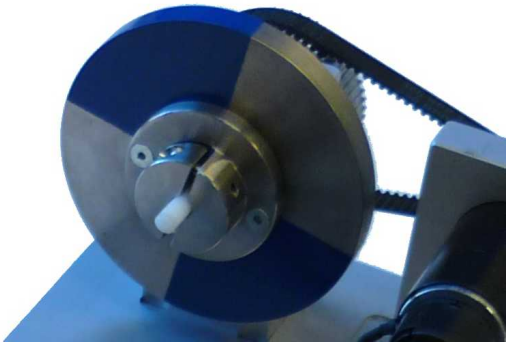
d (N.m.s/rad)	damping (viscous friction value)
Tc (N.m)	coulomb friction torque

The values of these parameter are unknown. We have to do an experiment to find them. In this experiment we will supply the motor with a block wave voltage. We will measure the rotation of the motor. To prevent the flexible shaft and load disk from interfering with the experiment, we will remove the flexible shaft. Because all other components of the model are known, we can use the simulation to identify the belt friction parameters. The proper parameters values are found when the simulation results match with the measurements.

Practice

We will remove the flexible shaft, so the motor disk has no interference with the load disk.

1. On the Torsion Bar, remove the shaft and insert a shaft piece in the motor disk to balance it.



Use a shaft piece to balance the clamp.

2. Open 20-sim and load the model *Identification\Belt Identification.emx*.
3. Start the Simulator and export the submodel Measurement to 20-sim 4C.

Now 20-sim 4C will be loaded with the model Measurement loaded.

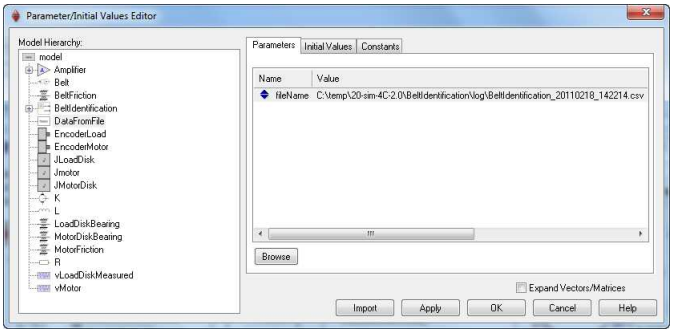
4. Make the proper settings in 20-sim 4C (see the Getting Started section to see how)
5. Select the encoder variable *RevMotorToRevMotorDisk\output* for logging.
6. Remember the location and name of the log-file. We are going to use it later on.
7. Click on the Run with Logging button  to start running the model on the target.

After 40 seconds the logging is finished and a plot should appear.

8. Go back to the model Belt Identification.emx in the 20-sim Editor.

As you can see, the model contains a file input block (DataFromFile) to compare the simulated motor rotation with the measured rotation. The shaft has been removed in the model, just like on the real setup. A default measurement is used that will be replaced by your own measurement.

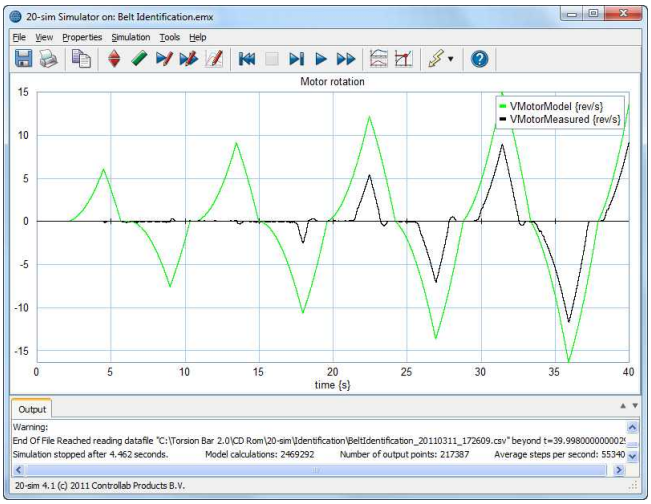
9. Open the Simulator.
10. In the Simulator, from the Properties menu select Parameters.
11. Select the parameter *DataFromFile\filename*.



In the Parameters/Initial Values Editor you can change the filename of the log file.

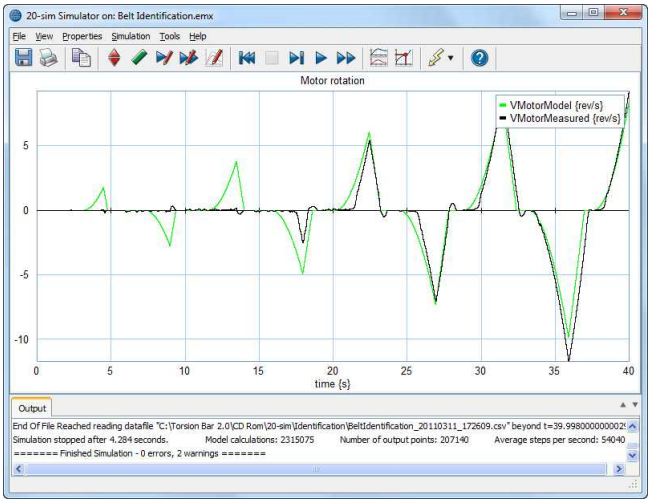
12. A Browse button will be visible. Click it to select the name and location of your log-file (see step 7).
13. Click OK to close the Parameters/Initial Values Editor.
14. From the Simulation menu select Run to start a simulation.

Two plots are shown, one of the motor rotation and one of the motor speed. We will focus on the speed plot since this the most handy plot for curve fitting. As you will see the motor speed is stepwise increased. The simulated speed is different because the belt friction parameters are not correct.



The speed of the measured disk (green) and simulated disk (brown).

15. From the Properties menu select Parameters.
16. Select the parameter *BeltFriction \ d* and change its value.
17. Select the parameter *BeltFriction \ Tc* and change its value.
18. Click OK to close the Parameters/Initial Values Editor.
19. From the Simulation menu select Run to start a simulation.



Equal curves for the measured and simulated motor speed.

The simulated curve will be different from the previous simulation. As you will see it is better to change one parameter value at a time and see the effects before changing the other parameter value. In the end you will find the values that give the best match of the simulated model and the measurements. The friction parameters that you will find depend strongly on the pretension of the belt, so don't worry if your measurement are a little different from the plots shown here. You will however find that the coulomb friction torque is the dominant parameter for the belt friction. Typical values that you may find are shown in the next table:

	min value	max value
d (mN.s.m/rad)	0.05	5
Tc (mN.m)	10	50

Typical parameter values for the belt friction.

If you look very carefully you will notice that the coulomb friction value should be higher for small movements. This is a phenomenon that is called stiction. Just when the belt is about to move, its resistance against the movement is the strongest. The model *Belt Identification with Stiction.emx* has a friction component that includes stiction. You can load it and try it out to see the difference.

20. If you are ready, please take care to put back the flexible shaft.

5.10 Black Box Identification

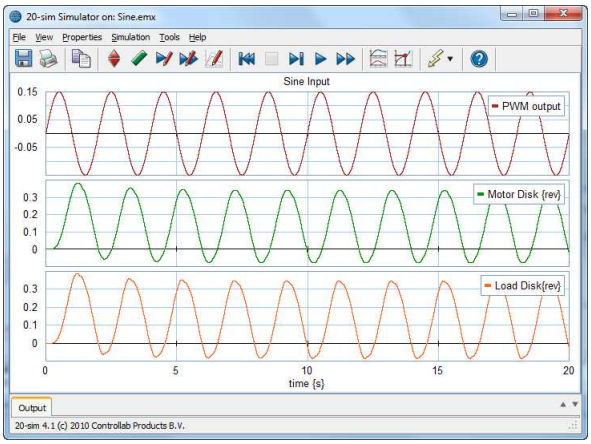
With black box identification we give the system a stimulus and measure the response. Out of both we try to create a mathematical model of the system. For the Torsion Bar the most obvious stimulus is to connect a certain signal to the PWM output. The response could be the motor encoder or the load encoder. A wide variety of input signals can be used. We will not go into much detail here and only use a sinusoidal input.

Sine

If we use a sine input signal and measure the response, we can create a frequency response.

1. Open 20-sim and load the model *Identification\Sine.emx*.
2. Start the Simulator and from the Simulation menu select Run.

Now you will see the response to a sine.



The response to a sine is a sine with a change in phase and amplitude.

Apart from the startup phase, the response to the sine is a sine with a change in phase and amplitude. For each frequency you will find another change in phase and amplitude:

3. In the Simulator from the Properties menu select Parameters.
4. Change the value of *SineOutput\Sine\f* to 3 {Hz} and click OK.
5. Start the Simulator and from the Simulation menu select Run.

Now you will see that the response of the system is still a sine but with different amplitude and phase. If you plot the changes of amplitude and phase against the frequency, you have found a frequency response. The frequency response can be used to create a model of the Torsion Bar.

Sine Sweep

Creating a frequency response using sine inputs is a tedious job, which can be automated using a sine sweep. This is a signal that starts with a sine of which the frequency is slowly increased.

1. Open 20-sim and load the model *Torsion Bar\Identification\Sine Sweep.emx*.
2. Start the Simulator and export the submodel SineSweep to 20-sim 4C.

Now 20-sim 4C will be loaded with the model SineSweep loaded.

3. Make the proper settings in 20-sim 4C (see the Getting Started section to see how).
4. Select the variables *PWM* and *RevMotorToRevMotorDisk\output* and *EncoderToRevLoadDisk\output* for logging.
5. Remeber the location and name of the log-file. We are going to use it later on.
6. Click on the Run with Logging button to start running the model on the target.

After 40 seconds the logging is finished and a plot should appear.

7. Go back to the model Sine Sweep.emx in the 20-sim Editor.

As you can see, the model contains a file input block (*DataFromFile*) to compare the simulated motor rotation with the measured rotation. A default measurement is used that will be replaced by your own measurement.

8. Open the Simulator.

9. In the Simulator, from the Properties menu select Parameters.

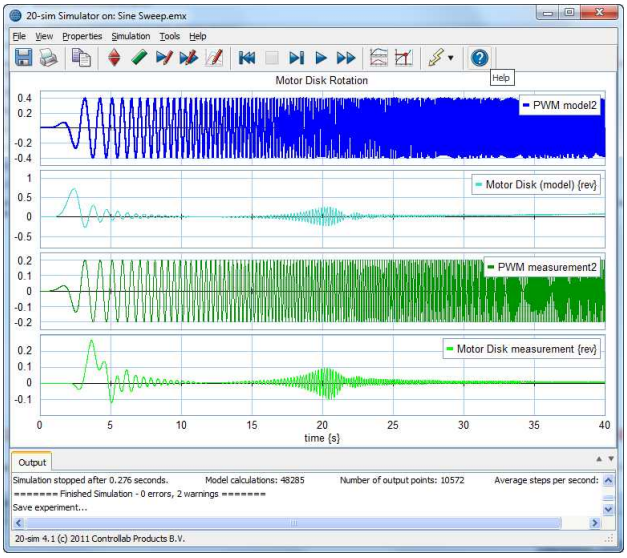
10. Select the parameter *DataFromFile\filename*.

11. A Browse button will be visible. Click it to enter the name and location of your log-file (see step 7).

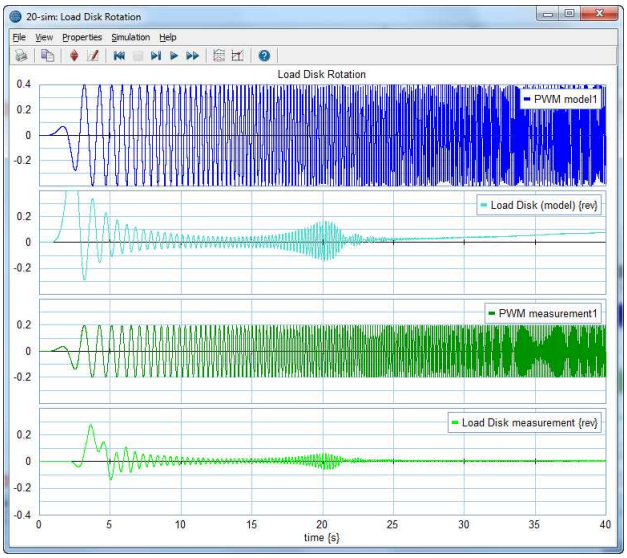
12. Click OK to close the Parameters/Initial Values Editor.

13. From the Simulation menu select Run to start a simulation.

Two plots are shown, one of the motor disk rotation and one of the load disk rotation. The PWM output is a sine with that constant amplitude of 0.2 that starts with a low frequency (about 1 Hz) which is increased continuously (up to 10 Hz). The plots show the PWM output and the response of the motor disk and load disk.



The simulated (blue) and measured (green) motor disk rotation.

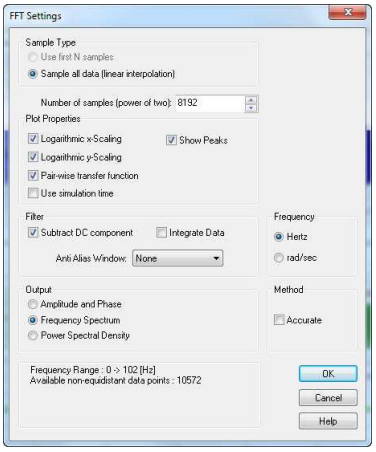


The simulated (blue) and measured (green) load disk rotation.

20-sim allows you to generate a bode plot out of these curves.

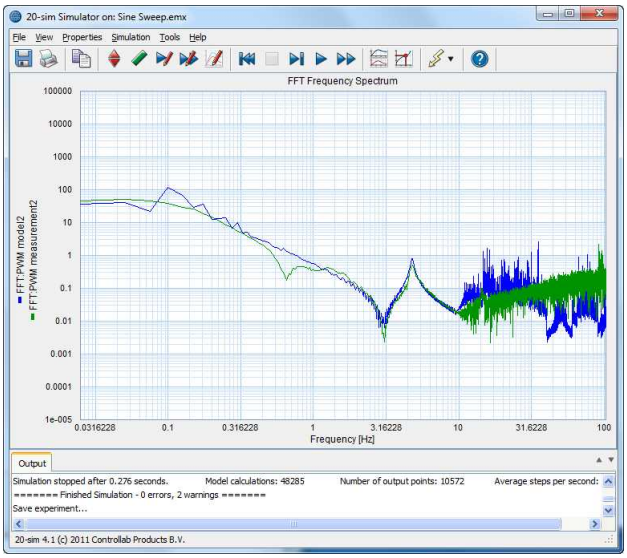
14. Put your mouse pointer on a plot and from the right mouse menu click FFT Analysis.

15. Choose the following settings.

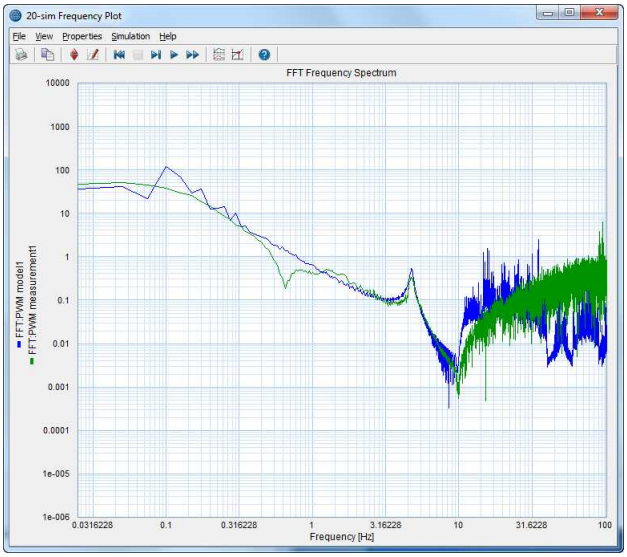


16. Click OK.

Now a bode plot will be created showing the amplitude as a function of the frequency. We are interested in the frequency range from 1 to 10 Hz.



Bode plot of the simulated (blue) and measured (green) motor disk rotation.



Bode plot of the simulated (blue) and measured (green) load disk rotation.

We can make some useful observations:

- Increasing amplitude for low frequencies: although the PWM output has a constant amplitude, the amplitude of the motor disk rotation and load disk rotation is not constant. The lower the frequency, the larger the amplitude gets.

- Decreasing amplitude for high frequencies: For frequencies above 5 Hz, the amplitude of the rotation of both disks gradually decreases.
- Zero amplitude at 3.3 Hz: Around 3.3 Hz there is no motor disk rotation at all. This is called the anti-resonance frequency.
- Maximum amplitude at 5.1 Hz: Around 5.1 Hz the motor and load disk rotation increases to a local maximum. This is called the resonance frequency.

Using the bode plots (and lot of math) we can find a mathematical model with exactly the same response. A well know system that gives a similar bode plot is a mass-spring-mass system. The equivalent in the rotational domain is a set of inertias coupled by a spring. The resonance frequency and anti-resonance frequency of such a system can be calculated as:

$$f_{anti-resonance} = \frac{1}{2 \cdot \pi} \cdot \sqrt{\frac{k}{J_2}}$$

$$f_{resonance} = \frac{1}{2 \cdot \pi} \cdot \sqrt{\frac{k \cdot (J_1 + J_2)}{J_1 \cdot J_2}}$$

If we fill in the inertias of the motor disk, the load disk and the stiffness of the flexible bar we find an anti-resonance frequency of 3.2 Hz and a resonance frequency of 5.1 Hz.

Tip

You can investigate the anti-resonance frequency and resonance frequency manually on the Torsion Bar.

1. Switch off the power and remove the belt.

2. Start to rotate the motor disk forwards and backwards in a gentle motion.

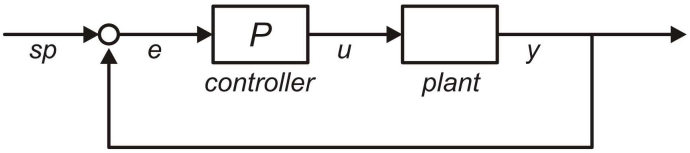
At around 3 motions per second you should be able to get the load disk into heavy motion, almost without moving the motor disk. This is the anti-resonance frequency. At around 5 motions per second you should see the load disk going forward and the motor disk in the opposite direction. This is the resonance frequency.

6 PID Control

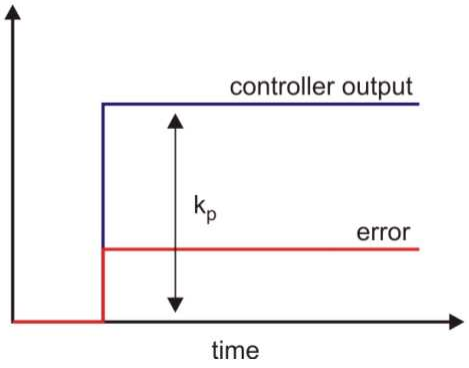
6.1 P-Control

Theory

A proportional controller, or shortly a P-controller, gives an output that is proportional to the error.



Proportional Control.

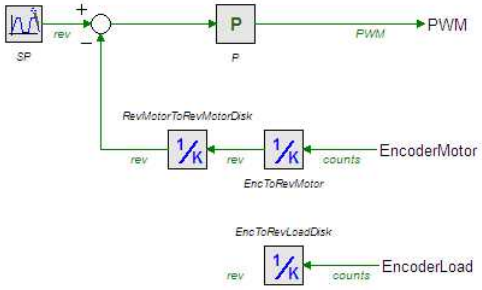


The controller output is in proportion to the error.

In other words, the controller output u is equal to:

$$u = k_p \cdot e$$

The parameter kp is called the controller gain. It determines how the error e will lead to an output u. A zero error will give a zero output. That is why a slight error always remains with a P-controller, even with high controller gains. In the chapter on Measurement and Control, we have introduced a proportional controller that regulated the motor rotation. We will make some small changes to this controller. We want the controller to regulate the motor disk rotation. Therefore we have to find a sensor that gives the motor disk rotation. We will not add an extra encoder but calculate this rotation out of the motor rotation. I.e. we have to divide the motor axis rotation by the belt ratio of 3.75 to get the motor disk rotation. With this change the control scheme will look like:



Block diagram of the proportional controller.

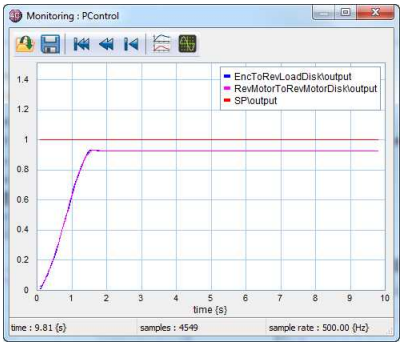
Practice

1. Open 20-sim and load the model *PID Control\P.emx*.
2. Start the Simulator and export the submodel PControl to 20-sim 4C.

Now 20-sim 4C will be loaded with the model Controller loaded.

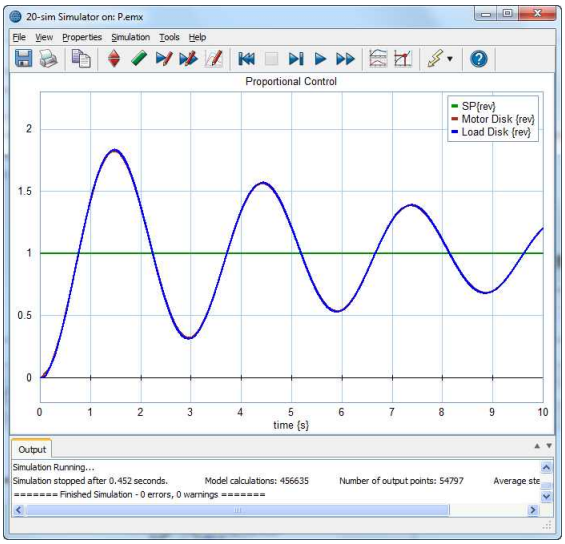
3. Make the proper settings in 20-sim 4C and select the variables *SP\output*, *EncoderToRevLoadDisk\output* and *RevMotorToRevMotorDisk\output* for monitoring.
4. Click on the Run button  to start running the model on the target.

Now the motor should start to run almost one rotation:



Proportional Control with a gain of 0.2.

As you can seen from the log plot, the motor disk follows the setpoint to almost one rotation. If you compare the results with the simulation, there is a considerable difference.

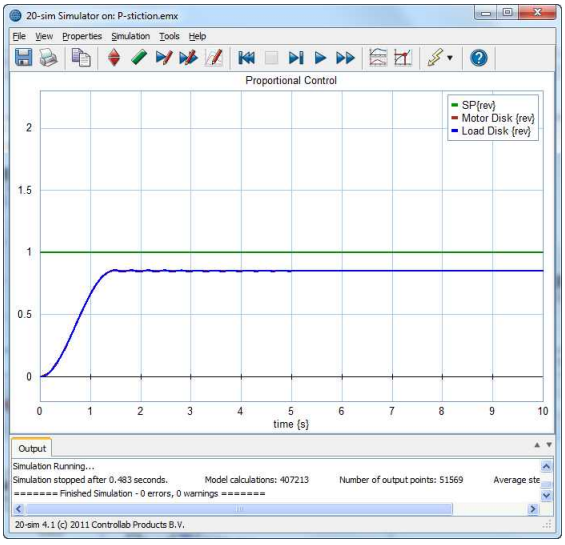


The simulation results show considerable difference with the measurements.

The simulation model uses a simple friction model to represent the belt. In the chapter on identification, we have seen that the belt should be modeled by a more complex friction model that includes stiction. To see the difference load the model *P-stiction.emx*.

5. Open the model *P-stiction.emx* and run a simulation.

The results now resemble the measurements.



A non-linear friction model improves the simulation.

The model with the non-linear friction shows a much better resemblance to the measurements than the model with linear friction. In practice however, linear models make good sense:

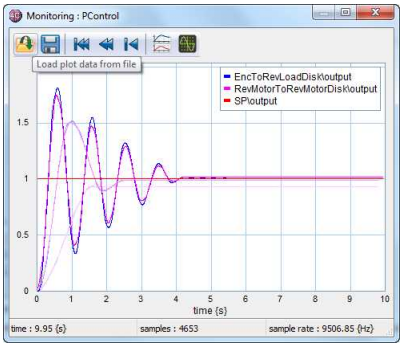
- Many design controller design methods can only be used for linear models.
- A well tuned controller will minimize the effects of friction, which makes both linear and non-linear models behave the same.

That is why we will not pay attention to the differences between the simulation and the measurement, and focus on the measurement results for the remainder of this chapter.

6. In 20-sim 4C increase the parameter *P \ kp* to a value of 0.5 and make a new run.

7. In 20-sim 4C increase the parameter *P \ kp* to a value of 2.0 and make a new run.

As you will see, increasing the proportional gain will make disks reach the setpoint more exactly but also increase the overshoot and result in oscillations.



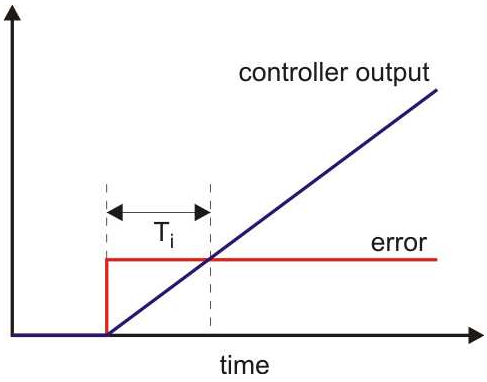
Proportional control with a gain of 0.2, 0.5 and 2.0.

To counter the overshoot, we can reduce the proportional gain, but that will make the load disk respond more slowly. Therefore we will look for other solutions.

6.2 PI Control

Theory

A proportional controller will give an output that is proportional to the error. If that input is not enough to get the actuator into motion, the error remains. Integral control is a kind of control that will increase the output, if an error remains. This is shown in the next plot.

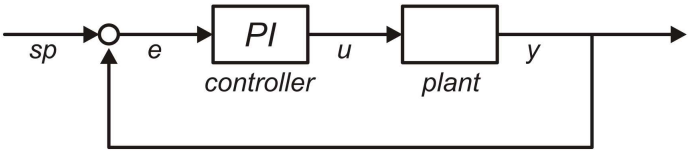


The controller output is the integral of the error.

The controller output u for an integral controller is equal to:

$$u = \frac{1}{T_i} \cdot \int e$$

The constant *Ti* is called the integral time. This is a time constant that indicates how much time is needed to create and output that is equal to the error. The smaller *Ti*, the quicker the response of the integral controller. In practice Integral control is always combined with proportional control. The proportional part assures that the system has a good response and the integral part will assure that it eventually reaches the setpoint.



Proportional and Integral Control.

Practice

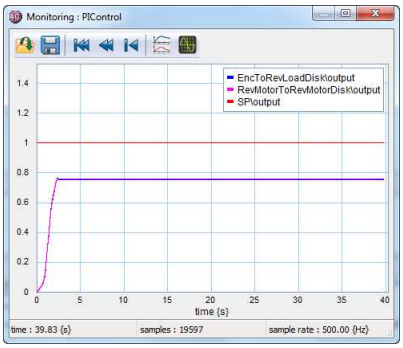
1. Open 20-sim and load the model *PID Control\PI.emx*.
2. Start the Simulator and export the submodel PIControl to 20-sim 4C.

Now 20-sim 4C will be loaded with the model Controller loaded.

3. Make the proper settings in 20-sim 4C and select the variables SP\output, *EncoderToRevLoadDisk\output* and *RevMotorToRevMotrDisk\output* for monitoring.
4. Click on the Run button to start running the model on the target.

Now the motor should start to run almost one rotation. Depending on the belt pretension, your results may vary a little.

5. In 20-sim 4C increase or decrease the proportional *P/kp* gain, until your results look like:

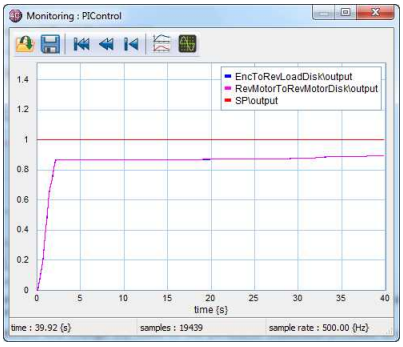


Proportional Integral Control with a gain of 0.15 and an integral time of 10000 s.

The proportional gain is deliberately set to a small value which gives a final error is about 0.2. The resulting PWM output is too small to overcome the friction. The integral time is set to 10.000 seconds to make it ineffective. We will set the integral time to an effective value of 80 seconds and inspect the results.

6. In 20-sim 4C change the value of the parameter *PI\taui* to a value of *8* and do a new run.

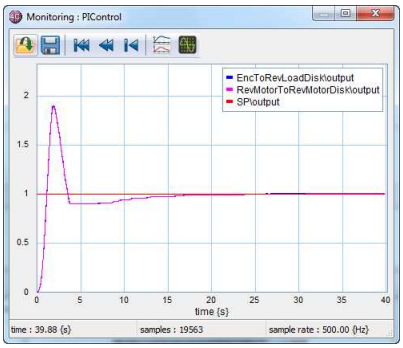
You should see something like the next plot. The motor disk starts to creep slowly towards the setpoint.



Proportional Integral Control with a gain of 0.1 and an integral time of 80 s.

The problem with integral control is that is accumulates the error during the whole run. if you make the integral time too small, the error that occurs at first second will lead to an overreaction of the integral controller: overshoot.

7. In the 20-sim Simulator, from the Properties menu select Parameters and change the value of the parameter *Controller\PI\taui to a value of 2* and do a new run.



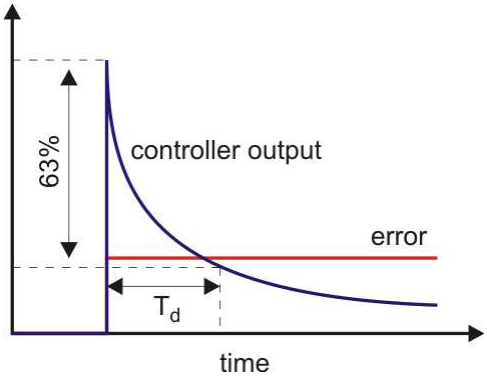
Proportional Integral Control with a gain of 0.15 and an integral time of 4 s.

As you will see, adding integral control (by lowering the integral time) will result in overshoot. The overshoot will turn into oscillation if you add more integral control.

6.3 PID Control

Theory

Integral control gives a slow response to the error between the setpoint and motor disk rotation and will lead to overshoot and oscillation. What we need is a controller that will give a quick response when the error between the setpoint and motor disk rotation starts to rise. This is called derivative control. With derivative control, the output is equal to the derivative of the error.

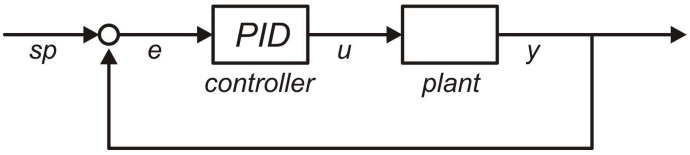


The controller output is the derivative of the error.

In other words, the controller output u is equal to:

$$u = T_d \cdot \frac{de}{dt}$$

The constant *Td* is called the derivative time. This time indicates how long in time a step change in the error will have its effect on the controller output. The large *Td* , the larger the response of the derivative controller. In practice derivative control is always combined with proportional and integral control. The proportional part assures that the system has a good response, the integral part will assure that it eventually reaches the setpoint and the derivative part will assure a quick response to changes in the setpoint.



Proportional, Integral and Derivative Control.

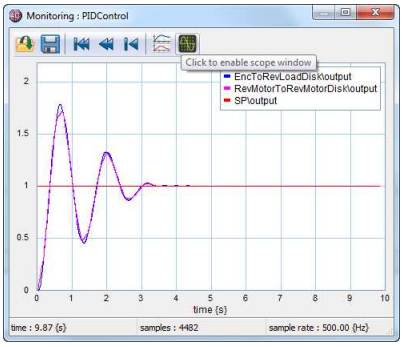
Practice

1. Open 20-sim and load the model *PID Control\PID.emx*.
2. Start the Simulator and export the submodel *PIDControl* to *20-sim 4C*.

Now 20-sim 4C will be loaded with the model Controller loaded.

3. Make the proper settings in 20-sim 4C and select the variables *SP\output*, *EncoderToRevLoadDisk\output* and *RevMotorToRevMotrDisk\output* for monitoring.
4. Click on the Run button  to start running the model on the target.

Now the motor should start to run one rotation with serious overshoot. Depending on the belt pretension, your results may vary a little.

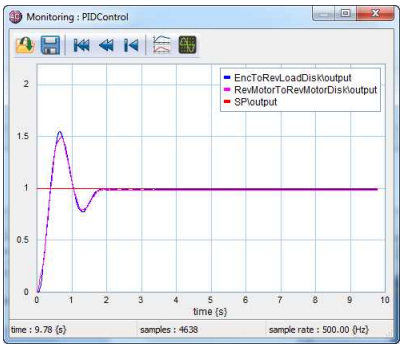


PID control with a gain of 1 and an integral time of 10000 s and a derivative time of 1 ms.

The integral time is set to 10000 s to make it ineffective. The derivative time is set to 50 ms to make it ineffective as well. We will increase the derivative time to see what effects it will have.

5. In 20-sim 4C change the value of the parameter *PID\tauD* to a value of 0.05 and do a new run.

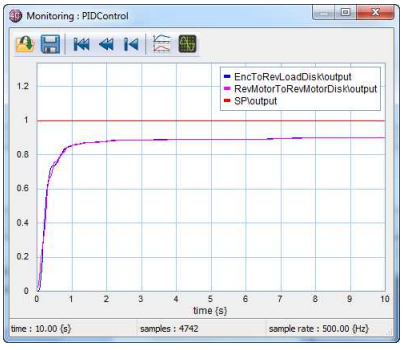
You should see something like the next plot. The overshoot is reduced!



PID control with a gain of 1 and an integral time of 10000 s and a derivative time of 50 ms.

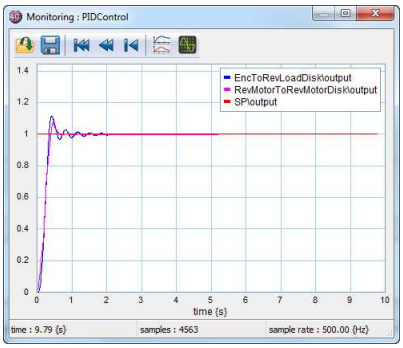
Derivative control can lead to oscillations in systems that have a lot of noise in the measured signals. For the Torsion Bar, increasing the derivative will only lead to too much damping.

6. In 20-sim 4C change the value of the parameter *PID\tauD* to a value of 0.5 and do a new run.



PID control with a gain of 1 and an integral time of 10000 s and a derivative time of 0.5 s.

With the help of derivative control, you can add more damping to the response. This will allow you to increase the proportional gain. Adding a little integral control will bring the disks accurately to the setpoint. The next figure gives an impression of the results that you can achieve.



PID control with a gain of 20 and an integral time of 1 s and a derivative time of 0.1 s.

7 Troubleshooting

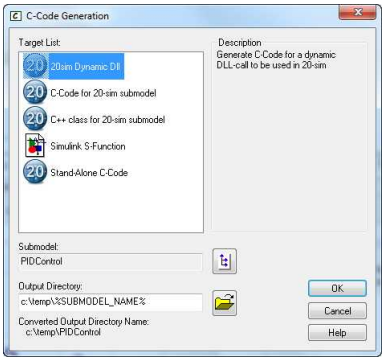
7.1 Errors

20-sim

First check if 20-sim was properly installed. For installation guidelines, see the 20-sim Getting Started manual.

I do not see the 20-sim 4C target

If you are generating C-code in 20-sim and see the following list of targets, you have forgotten to change the ini-file for C-code Generation.



The 20-sim 4C target is not listed.

Please read the *Installing 20-sim* topic in the 20-sim 4C reference Manual, how to set the correct location of the ini-file.

I cannot find the 20-sim models

The Torsion Bar comes with a CD-Rom which contains 20-sim models (e.g. *D:\Torsion Bar\20-sim*.**). Please copy these models to your local PC. We recommend to copy them to the Windows *Documents* folder.

Dependent states

If you make changes to your model, it may have dependent states. This is not a problem in 20-sim, because it has solvers to cope with this. When you generated C-code from a model with dependent states you will get the following warning.

The C-code generation menu in 20-sim.

You may ignore this warning if you are sure that the c-code that you are generating does not contain this dependent state. I.e. it is located in another part of the 20-sim model.

I cannot find the log file

If you cannot find the log file for use in 20-sim please check:

- Did you select variables for logging in 20-sim 4C?
- Did you select Run with Logging in 20-sim 4C?
- Click the *Configure Logging* tab to see what filename and location you have chosen in 20-sim 4C.

20-sim 4C

First check if 20-sim 4C was properly installed. For installation guidelines, see the 20-sim 4C Reference manual.

Configure Button is red

If the Configure button is red  , no connection could be made to the target.

- Make sure the Torsion Bar is powered on. If a fatal crash has occurred, switch the power off for and after a few seconds switch it on again. Then try to click on the IP-address button to search for an IP-address again.
- Make sure you have a cross cable or network cable connected to your PC and hardware.

Compile Button is red

If the Compile button  stays red, compilation was not finished successfully.

- Check if the ARM7300 - Torsion Bar Setup was chosen.

If that fails, please contact your local distributor for help.

Motor is not Running

If the motor is not running, please check the following items:

- Read the Testing section and run the motor test model to test if the sensors and actuator are properly working.
- Please check if the PWM output is connected with any output of your model. Check if that output signal has a non-zero value (choose that signal for monitoring).

20-sim 4C does not find an IP-address

Check the Network section for help.

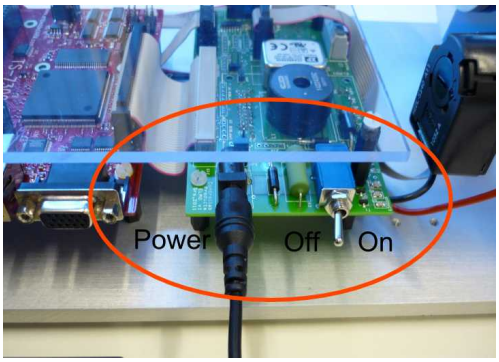
I see multiple IP-Addresses

Check the Network section for help.

Torsion Bar

Power

Make sure the power is on. To switch the power on, please connect the power adaptor and press the switch down. You can check if the power is on because the LED's on the controller board should start to shine.



Plug in the power cable just below the motor. Right of the power connector is the power switch.

There is only one way the output of the power adaptor can be connected and that is just below the motor. The input of the power adaptor has to be connected to you local 220 V or 110 V supply. The power switch is located right to the power cable connector.

Reassembling Parts

When you want to reassemble parts of the Torsion Bar, please take care to follow the instructions.

Results are Different

If the results of monitored variables or logged variables are different from the plots that are shown in the various examples, please check the following items.

- Read the Getting Started section and run the example models to test if the Torsion Bar is properly working.
- Read the Testing section and run the example models to test if the sensors and actuator are properly working.
- Please check if the correct variables were chosen for monitoring or logging.
- Check if you have opened the plot window before starting a run.
- Some models use only the motor encoder for input. Make sure that you have not connected the encoder input of the model with load input of the Torsion Bar by mistake!
- The belt friction plays an important role in the response of the Torsion Bar. Check if you have mounted the belt correctly.

Simulations show different results

You can use the Torsion bar model in 20-sim to test your controllers. If you find out the simulation results are different from the results that you find on the Torsion bar please check the following items.

- Check if you have used the correct simulation model. You can copy the model from the example models that come with the torsion bar.
- The belt friction plays an important role in the response of the Torsion Bar. Check if you have mounted the belt correctly.

The Torsion Bar is making a lot of noise

When the belt has been removed and reassembled the pretension may be too weak. This causes the motor to get unstable and make a lot of noise. Please check if you have reassembled the belt correctly.

7.2 Network

20-sim 4C does not find an IP-address

All network related problems result in the same issue, 20-sim 4C does not find an IP-address. Please check the following conditions :

1. First of all, make sure that a network is connected between your PC and your target. A cross cable if connected directly between your target and your PC, a network cable if connected via a network switch or hub.
2. Check if the Torsion Bar is connected and is powered on.
3. Check if the led's at both network connections are on (is the cable inserted well?)

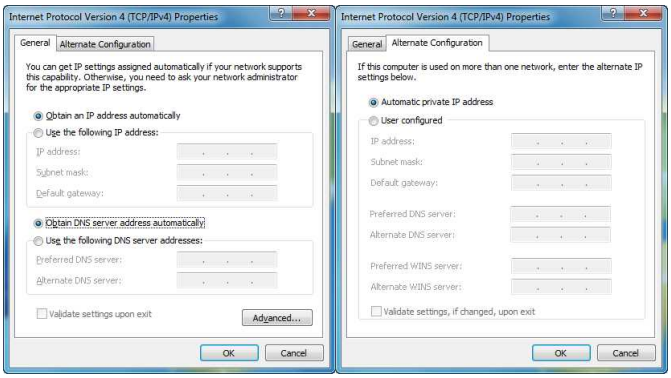
If the LED's at the network connector are off, it is not properly inserted. Re-insert until the LED's start shining, indicating good network connection.

4. Check with your network administrator of system administrator that the network is not blocking traffic between your PC and your target. Turn-off your firewall temporarily to check if this is the problem.
5. If multiple networks are connected to your PC, disable all networks that are not connected to the Torsion bar.
6. If no DHCP server is used, the Torsion Bar will fallback on the automatic private address of 169.254.254.254. Make sure the network interface on the host pc has a network IP-address in the same range.

Cross Cable

If all conditions are met but 20-sim 4C does not find an IP-address, use a cross-cable to connect the target with your PC.

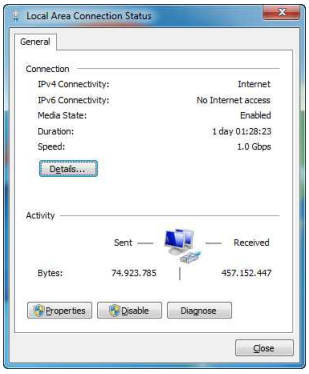
1. First of all, make sure that a cross cable is connected between your PC and your target.
2. Check if the target is connected and is powered on.
3. Open the TCP/IP Properties of your connection. Go to Windows Help and search for "configure TCP/IP settings" to find out how you can open the TCP/IP Properties.
4. Make sure the following settings are chosen.



The proper settings for the TCP/IP properties of you connection.

5. Open the *Local Area Connection Status* window of your connection. Go to Windows Help and search for "local area connection" to find out how you can open the *Local Area Connection Status* window.

The status of your network adapter should look like the next figure.

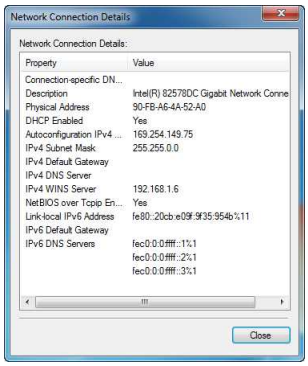


The status of your network.

Do not worry about the warning. Windows will try to obtain an automatic private address. This may take some time (1 minute in some cases) before the connection is established.

6. Click on the Details button.

If the private address was obtained successfully, the IP address should be in the range 169.254.x.x.



Check your IP address.

7. Close all network windows and return to 20-sim 4C.

If after one minute a connection cannot be made, try the following steps:

- Turn-off your firewall temporarily to check if this is the problem.
- Close all other network connections (do not forget your wireless network) to check if this is the problem.
- Enter the IP address 169.254.254.254 manually in 20-sim 4C.

My Firewall gives a Warning

If you click the Discover button of the Configure Target tab, 20-sim 4C will scan the network for connected targets. An active firewall will respond the first time by giving a warning message:

The Windows firewall message when the Refresh button is clicked.

You should allow 20-sim 4C access to the network to find the connected targets. If you do not allow access, the correct IP-address can not be shown.

1. Click the Allow Access button (or a similar button if another firewall message is shown) if the name of the program is 20-sim 4C and the publisher is Controllab Products B.V.

2. Please contact you network administrator if the name of the program is not 20-sim 4C or the publisher is not Controllab Products B.V.

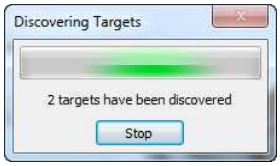
3. Please contact you network administrator if you do not have permission to change the firewall settings.

What to do with multiple IP-Addresses?

20-sim 4C automatically searches for targets which are connected to a network. If more than one IP-address is listed, it means that several targets are connected to the network. To find out the address of a target, if more than one address is listed do the following:

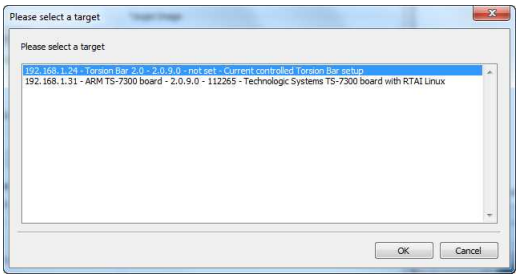
1. Switch the power on of your target.
2. In 20-sim 4C click on the Configure  button.
3. In the Configure Target tab, Click the Discover button.

20-sim 4C will now start a search to find the IP-addresses of connected targets.



Finding connected targets.

After a few seconds the search is ready and a list of targets is shown.



Click the target that you would like to use.

1. Select your target and click OK.

Index

The Torsion Bar is a setup that is great for teaching mechatronics, machine dynamics and control. The setup is typical for an industrial machine and it contains all the elements that we can find in industrial machines.

- Operator display
- Motor controller and amplifier
- Servo motor
- Drive train
- Sensors

The Torsion Bar can be used in combination with the software 20-sim Professional and 20-sim 4C. With this software you can automatically generate C-code to control the Torsion Bar and perform many tasks.

- Making measurements
- Running various controllers
- Identification
- Running your own code