Lesson 5
Simulink

Instructor:
ir drs E.J Boks
Elecrical Engineering
Embedded Systems Engineering
phone:(026) 3658173
Room: D2.03
e-mail:ewout.boks@han.nl

Contents

- Introduction into Simulink
- How simulink works
- Simulation results analysis
- Simulink debugger
- Tutorial and exercise: creating a control model

introduction into Simulink

- What is Simulink:

Simulink is a software package for modelling, simulating, and analyzing dynamic systems.

It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two.

Systems can also be multirate, i.e., have different parts that are sampled or updated at different rates.

Why Simulink ?

- In the last few years, Simulink has become the most widely used software package in academia and industry for modeling and simulating dynamic systems.
- Simulink encourages you to try things out. You can easily build models from scratch, or take an existing model and add to it. Simulations are interactive, so you can change parameters on the fly and immediately see what happens. You have instant access to all the analysis tools in MATLAB?, so you can take the results and analyze and visualize them. A goal of Simulink is to give you a sense of the *fun of modelling* and simulation, through an environment that encourages you to pose a question, model it, and see what happens.

Why Simulink ?

With Simulink, you can move beyond idealized linear models to explore more realistic nonlinear models, factoring in friction, air resistance, gear slippage, hard stops, and the other things that describe real-world phenomena. Simulink turns your computer into a lab for modeling and analyzing systems that simply wouldn't be possible or practical otherwise, whether the behavior of an automotive clutch system, the flutter of an airplane wing, the dynamics of a predator-prey model, or the effect of the monetary supply on the economy.

Simulink start

To start simulink, type simulink at the cmd prompt.

Look around in the various model components. Try to build a simple model yourself.

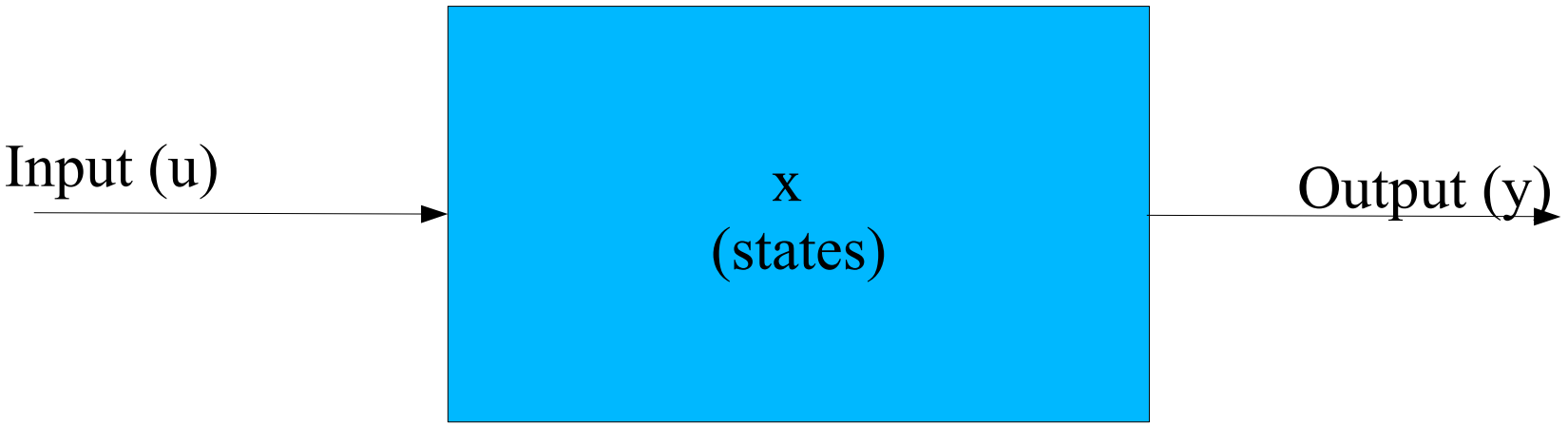
Jump right in !

Exercise:

Build yourself a model to view a sine wave, its derivative and the magnitude of the sine wave.

Tip : find all necessary components first by browsing the model library.

Simulink internals



Blocks can have *states* and are therefore persistent in that case

Blocks can also be stateless and do not require memory in that case

State and stateless configuration is determined by block parameters

Data input into Simulink models

Data may be input into models using the following methods:

- Use any of the data generation sources.
- Data already present in the Matlab/Simulink workspace can be used as input to the model. Use the Workspace I/O pane to specifiy the input data. This data can be either:
  - An array of real (i.e non complex) data.
  - A structure consisting of *time* and *signals* fields.

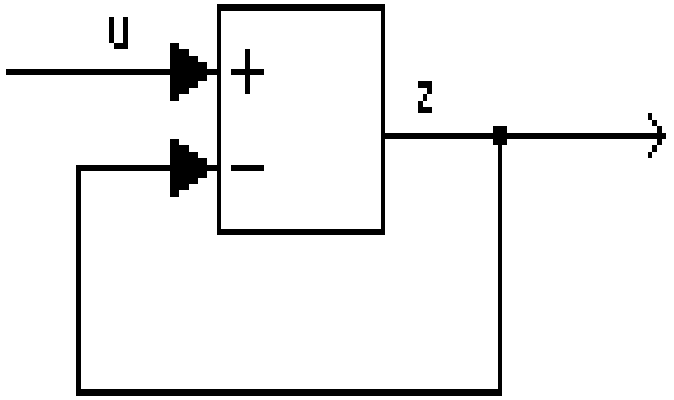
Dynamic systems start-up

- To perform the simulation of a dynamic system, Simulink discretizes the system and performs a step-to-step computation of every block's internal state.
- At simulation start-up, Simulink
  - Evaluates the parameters to determine all necessary information
  - Flattens the model hierarchy by replacing virtual subsystems with their contents.
  - Sorts the block execution order
  - Sets the non-specified sample times
  - Allocates and initialises the computer memory required

Simulink at execution time

- Simulink computes the system states with intervals equal to the step size which is computed using the selected *solver*. There are two classes of solvers:
  - Fixed step solver. The step size is entered by the user and the solver aids in the solution of the system equations and states. Use this if you need to explicitly state a step size (e.g when simulating a digital system).
  - Variable step solver. The solver computes the required step size, based on the prescribed resolution or precision. A faster computation than with the first solver is often achieved without sacrificing performance.
- Zero crossing detection is required to compute certain important events (discontinuities) in a simulation. Therefore, an additional zero crossing detector computes discontinuties that were not sampled with the computed/set step size.

Execution time: algebraic solutions

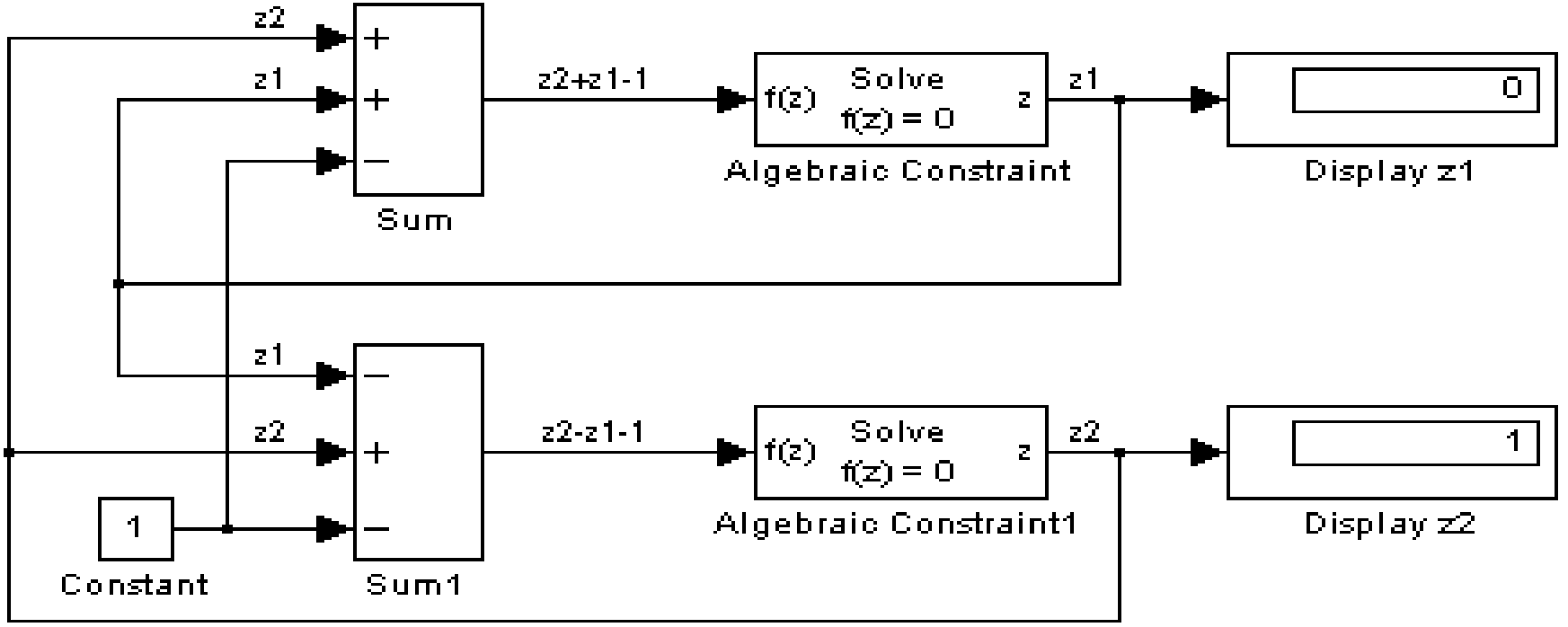


In order to resolve recursive systems, at the time of block evaluation the output must be used as an input. This is not always possible, as certain blocks are stateless and have no memory. In that case, the algebraic constraint block is required.

The algebraic constraint block

Below is an implentation where stateless blocks are given a state using the algebraic constraint block.

The block is also a handy way to specify initial conditions of variables.



Subsystems

It is possible to group blocks into a separate system, called a subsystem. This is helpful because:

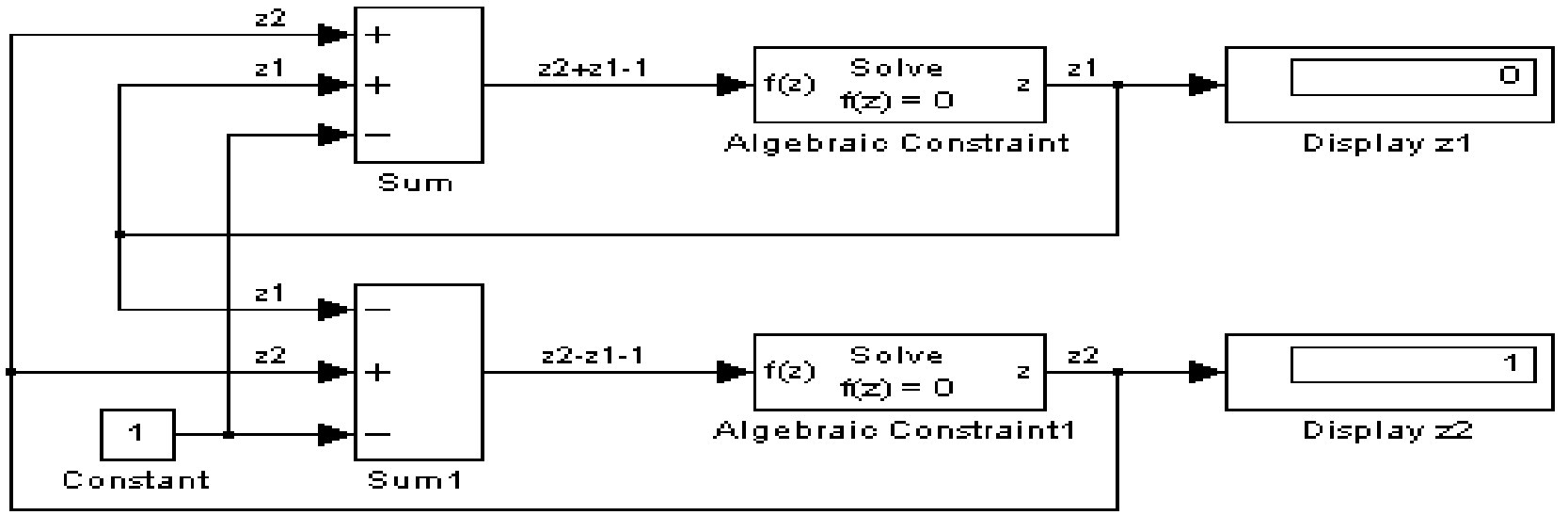
- It reduces the number of blocks in your display.
- It builds functionally related blocks into a handy container, allowing you to maintain oversight.
- It enables a layered approach to your design where subsystems represent layers.

Subsystems are built by:

- Creating a subsystem and opening it.
- Grouping blocks together and lumping them into a new subsystem.

Algebraic constraint in subsystems

- At initialization, subsystems are normally substituted into the main system. When using the subsystem recursively, this can lead to evaluation problems.
- In that case, add the trigger symbol to the subsystem



Signals and virtual signals

- Signals are the lifeblood of Simulink. They are either one or two dimensional arrays which are output at the frequency of the simulation step.
- Virtual signals are signals that represent other signals graphically. They are used to simplify a diagram.

Result evaluation

Results can be inspected by:

- Either the scope block or the XY block.
- supplying return values to the workspace using the out block. This can be done by specifying the the output parameters in the Workspace I/O pane (under simulation parameters)
- Supplying the return values directly to the workspace block.

Simulink debugger

- Simulink has a debugger to enable an exact nailing-down of run-time problems.
- To start the debugger, type sldebug <model> at the cmd prompt.
- All commands are listed after typing ? in the debugger.
- The major commands are also depicted as icons on the debugger.

Debugger major features

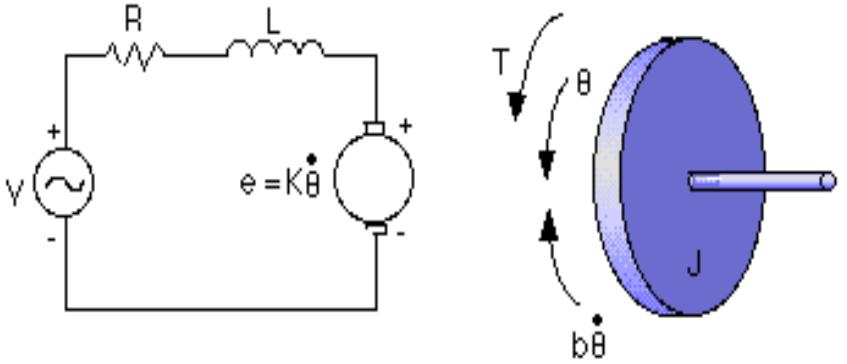
- Stepping through the simulation
- Setting break-points at:
  - At blocks
  - At time steps
  - At non-finite values
  - At zero crossings
- Displaying I/O blocks:
  - Algebraic loop information
  - System states
  - Integration information
  - Model execution order and non-virtual system information

Example model creation

The following example illustrates how to build a Simulink model based on an engineering problem.

Problem case: design a PID controller for a motor positioning system. The input voltage should lead to a set angle of the positioning system.

Motor problem



A common actuator in control systems is the DC motor. It directly provides rotary motion and, coupled with wheels or drums and cables, can provide transitional motion. The electric circuit of the armature and the free body diagram of the rotor are shown in the figure above.

The motor position and velocity equations are:

$$J\frac{d^2\theta}{dt^2}+b\frac{d\theta}{dt}=Ki$$

$$L\frac{di}{dt}+iR=V-K\frac{d\theta}{dt}$$

Laplace equation of motor

This leads to the following transfer function between Voltage as the input setting and position (θ):

$$\frac{\theta}{V} = \frac{K}{s\left((Js+b)(Ls+R)+K^2\right)}$$

$$\frac{\theta}{V} = \frac{K}{JLs^3+(JR+Lb)s^2+(bR+K^2)s}$$

Laplace equationof motor

Also, assume these parameters for the motor:

- Moment of inertia of the rotor (J) = $3.2284*10^{-6}$ $kg.m^2/s^2$
- damping ratio of the mechanical system (b) = $3.5077*10^{-6}$ N/ms^{-1}
- electromotive force constant (K) = 0.0274 Nm/A
- electric resistance (R) = 4 Ω
- electric inductance (L) = $2.75*10^{-6}$H
- input (V): Source Voltage
- Output (θ): position of shaft
- The rotor and shaft are assumed to be rigid

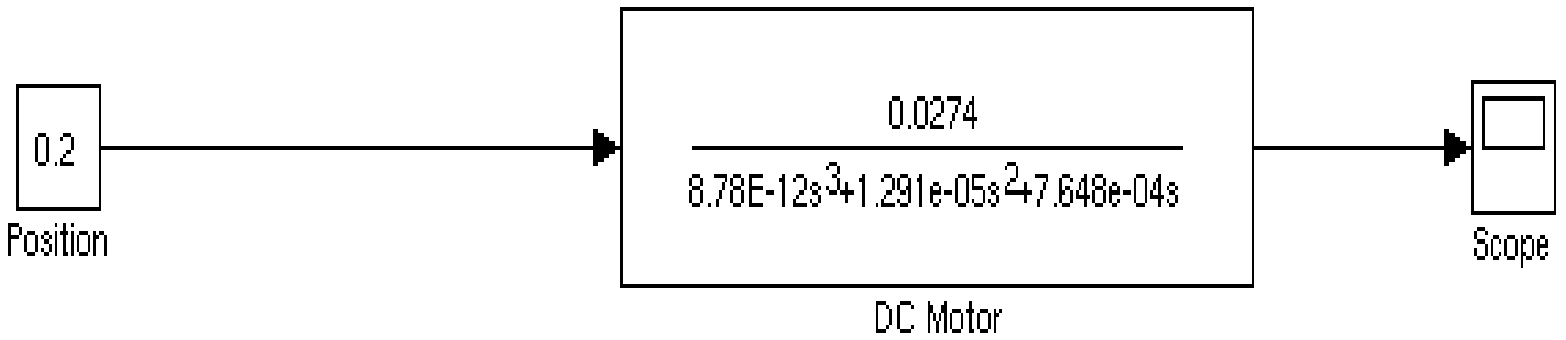
Design an open loop system

Exercise1:

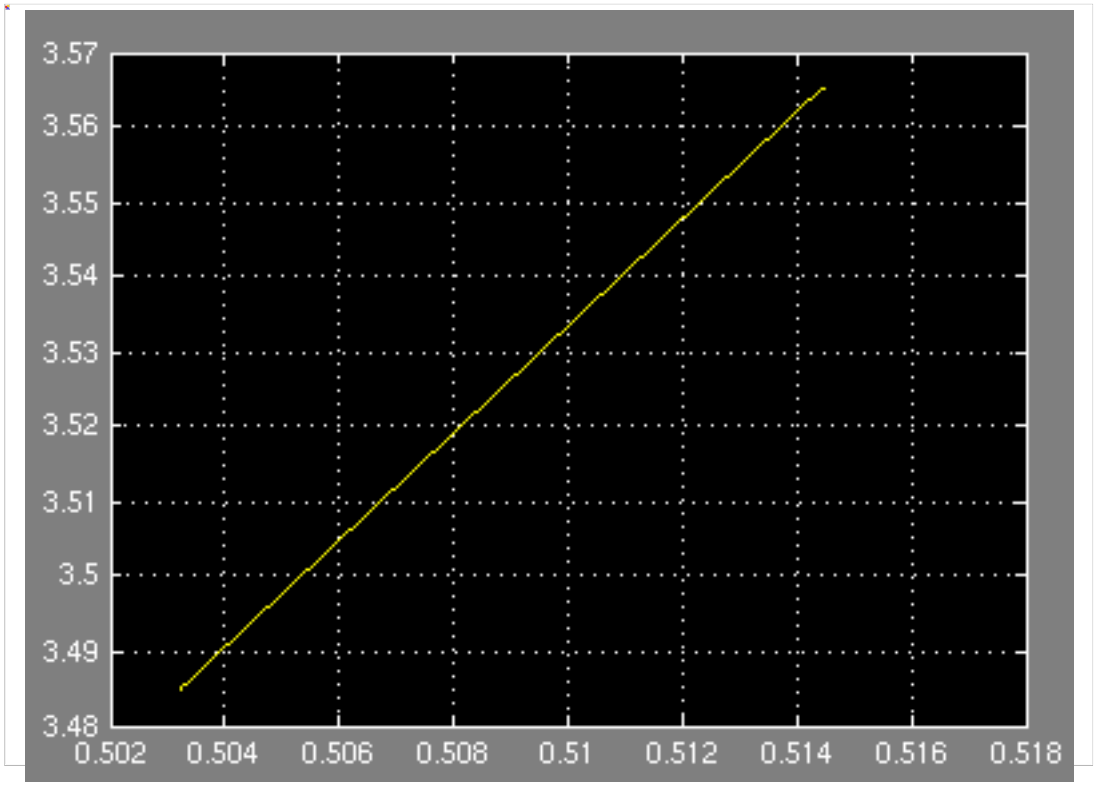
Design an open loop system in Simulink using the transfer function in continuous

Show that the motor position system is unstable.

Open Loop layout



Open loop plot



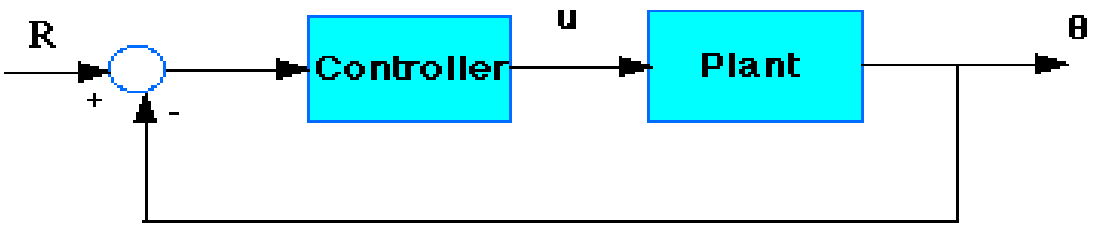
Add feedback to the system

- In order to stabilize the system, add a feedback mechanism to the system.
- Use a PID controller for this purpose. A PID controller is characterised by the following transfer function:

$$K_p + \frac{K_i}{S} + K_d S$$

Adding feedback

A system (*plant*) with an added controller in a feedback loop looks like:



Exercise 2

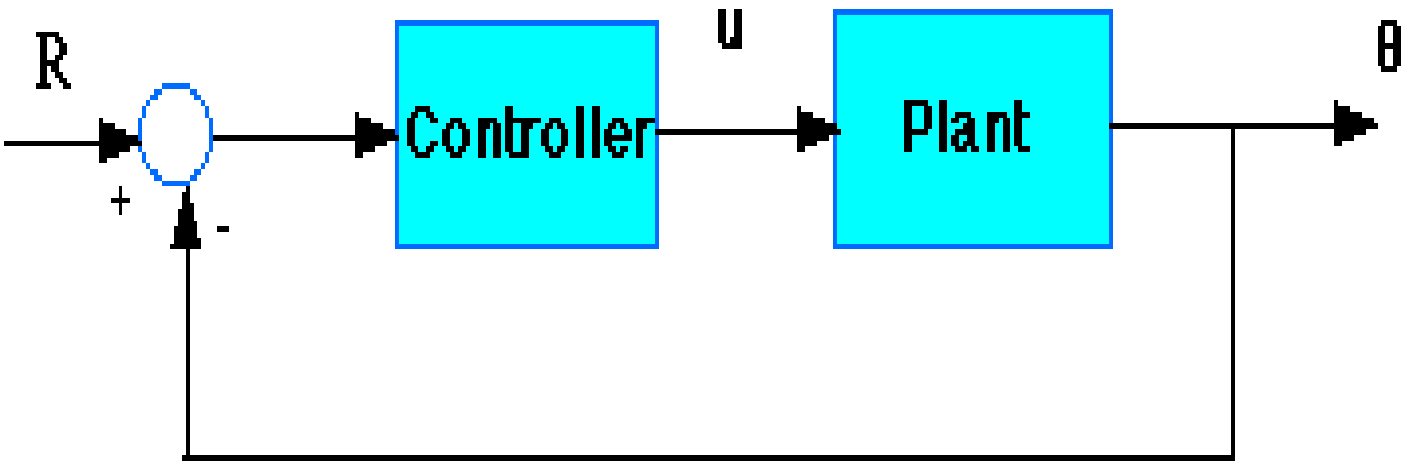
Exercise 2: add a feedback mechanism to the DC motor.

Later, insert a PID controller

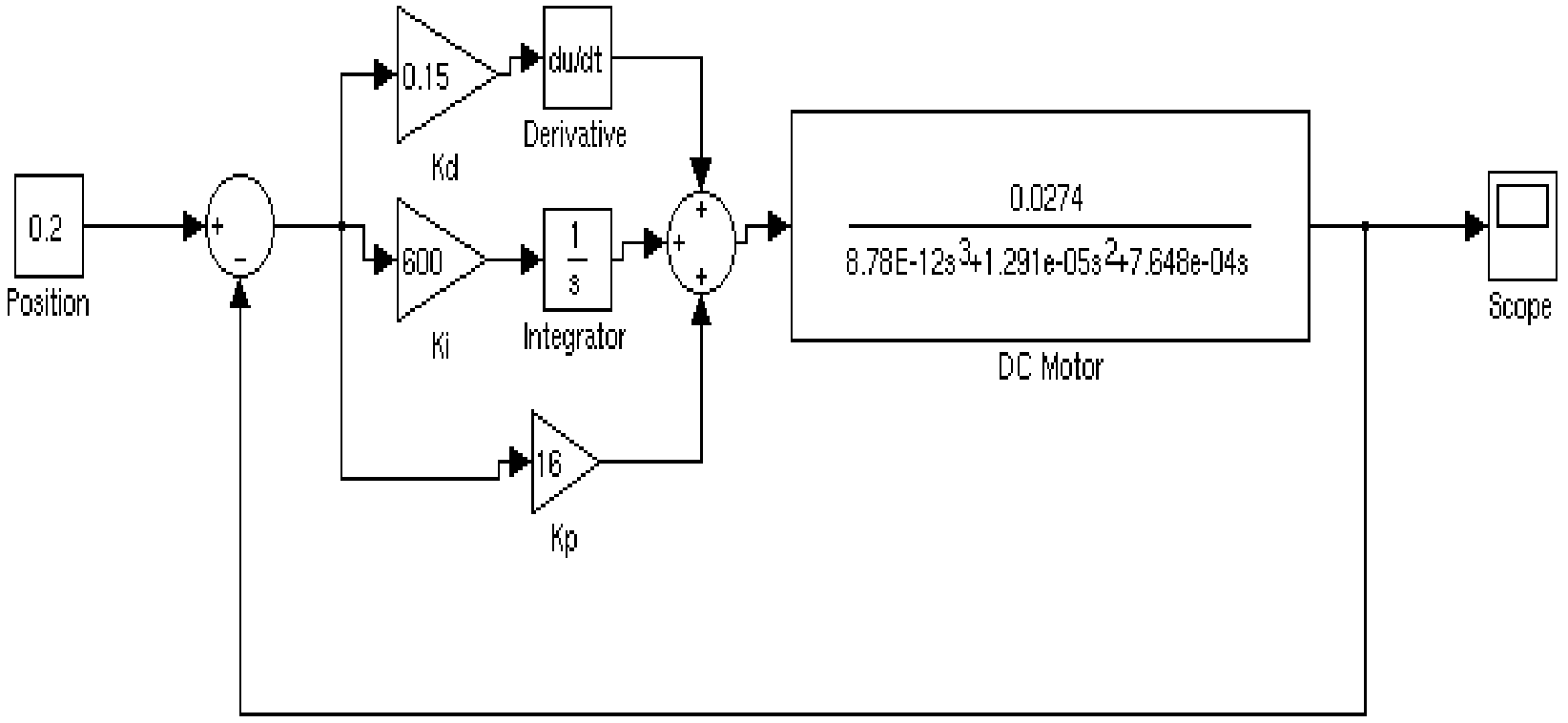
Verify with these parameters that the position setting is stable now:

- K_p=16
- K_i=600
- K_d=0.15

Closed loop with PID design

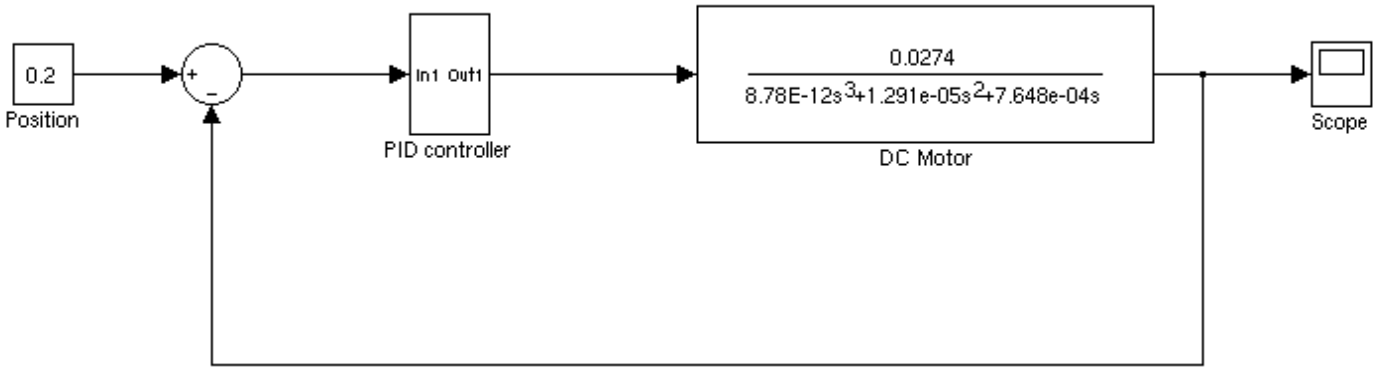


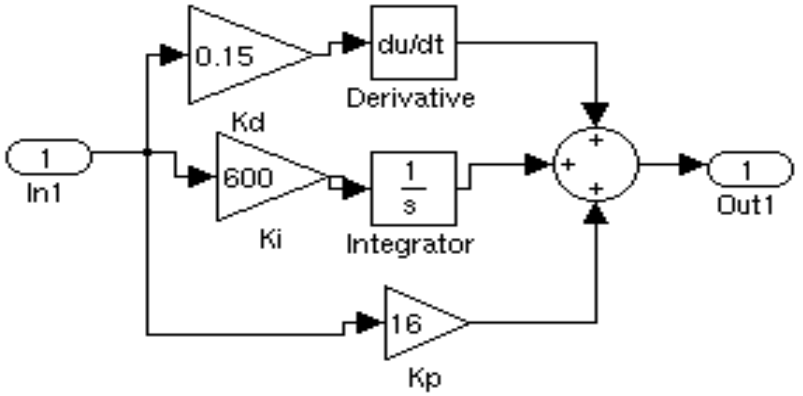
Closed Loop plot withPID controller



Using a subsystem

The system can be simplified using subsystems:





End

End of lesson 5