INTRODUCTION

A buck converter, also known as a step-down converter, is a type of DC-to-DC converter that decreases voltage while increasing current from its input (supply) to its output (load). It is classified as a switched-mode power supply, which means it uses switching elements to efficiently convert electrical energy. The primary function of a buck converter is to take a higher input voltage and convert it into a lower output voltage with high efficiency, often exceeding 90%. This makes buck converters particularly useful in various applications where power efficiency is critical.

https://www.rs-online.com/designspark/how-the-buck-converter-has-evolved

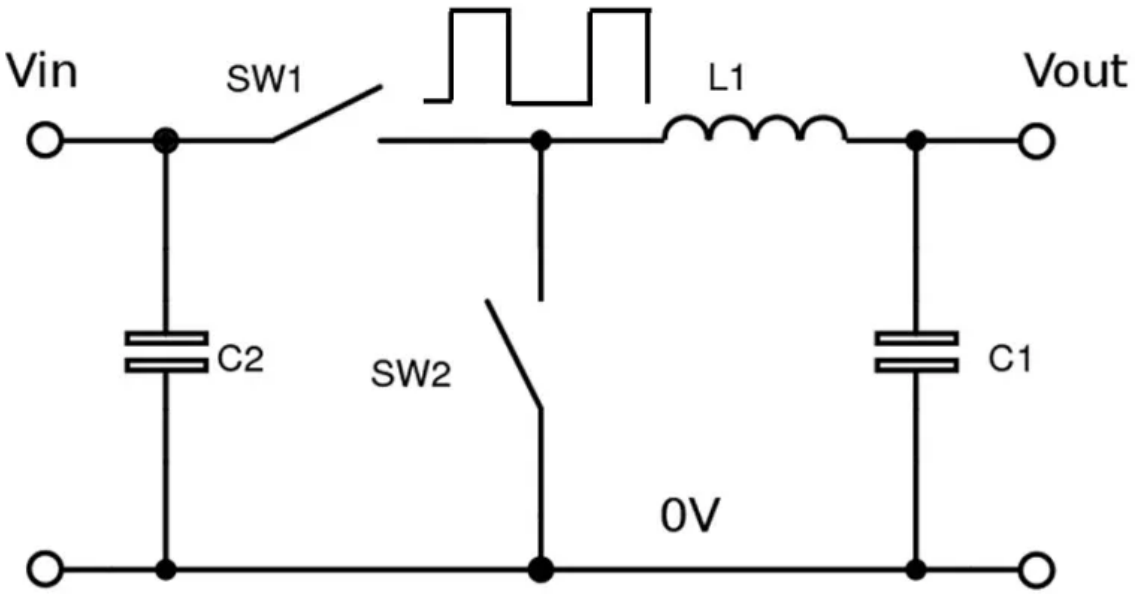


Figure 1: The basic elements of a buck converter

LCD BACKLIGHT CONTROL

When dealing with displays, backlight control is necessary.

Buck converters are suitable if the starting voltage is higher than the one needed to drive the LEDs:

https://www.analog.com/en/resources/analog-dialogue/articles/convert-a-buck-regulator.html

https://www.electronicdesign.com/technologies/power/whitepaper/21160211/analog-devices-low-emi-led-driver-features-integrated-switches-internal-pwm-dimming

https://www.ti.com/lit/wp/slva130/slva130.pdf?ts=1737985328095

As an example, the following display:

https://docs.rs-online.com/5ccc/A700000008505485.pdf

Has backlight LEDs arranged as follows:

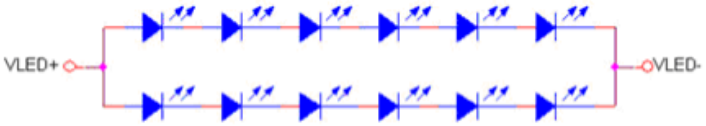
4.2 LED BACKLIGHT UNIT

Electrical characteristic of LED Back-light

The back-light system is an edge-lighting type with 12 LED.

The characteristics of the LED are shown in the following tables.

Item	Symbol	Min.	Typ.	Max.	Unit	Note
LED current	IL	--	40	--	mA	(2)
LED voltage	VL	--	19.8	--	V	
Operating LED life time	Hr	40K	50K	--	Hours	(1)(2)



LED Light Bar Circuit

In the context of LEDs, "buck current" refers to a method of controlling the brightness of an LED by using a buck converter circuit to deliver a constant, adjustable current to the LED, where higher current directly translates to increased brightness.

Essentially, the more current flowing through the LED, the brighter it will shine.

By adjusting the duty cycle of the buck converter, you can effectively control the average current supplied to the LED, enabling smooth dimming functionality.

Power electronics is a field that deals with the conversion and control of electrical power using electronic devices. It encompasses various applications, including DC-DC converters, AC-DC converters, inverters, and motor drives. The primary goal of power electronics is to efficiently manage electrical energy for various applications while minimizing losses and improving performance.

FPGAs allow to implement custom control algorithms tailored to specific applications without the need for dedicated hardware. This flexibility is crucial in power electronics, where different topologies and control strategies may be required based on the application.

https://www.xilinx.com/developer/articles/have-you-considered-fpgas.html

FPGAs can operate at high clock rates (often exceeding 100 MHz), enabling them to process data quickly. In power electronics, this capability allows for real-time monitoring and control of parameters such as voltage, current, and frequency, which are essential for maintaining system stability and efficiency.

Many modern power electronic systems utilize digital control techniques such as PID controllers or state-space methods.

https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/pel2.12158

FPGAs provide an ideal platform for implementing these algorithms, due to their ability to handle complex mathematical computations efficiently.

In many cases, FPGAs can be combined with analogue components (such as ADCs) to create a hybrid system that leverages both digital processing capabilities and analogue signal conditioning. This integration enhances the overall performance of power electronic systems.

https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/iet-pel.2013.0736

The integration of FPGA technology into power electronics design significantly enhances flexibility, speed, and performance while allowing for advanced control strategies tailored to specific applications.

SIMSCAPE

Simscape Electrical™ is a comprehensive tool developed for modelling and simulating electronic, mechatronic, and electrical power systems. It was formerly known as

SimPowerSystems™ and SimElectronics®. This software provides a rich library of components that allows engineers to simulate various applications, including electromechanical actuation, smart grids, and renewable energy systems.

https://uk.mathworks.com/products/simscape-electrical.html?s_tid=srchtitle_site_search_3_simscape

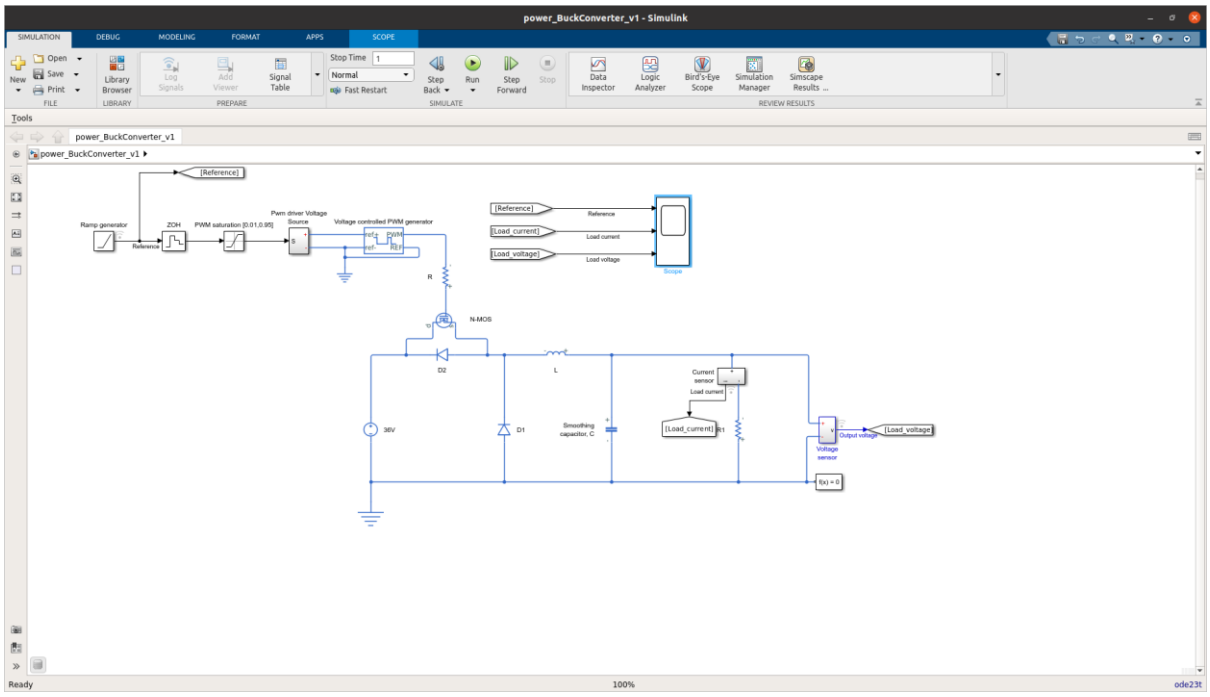
The tool includes models of semiconductors, motors, and other essential components that are crucial for the design and analysis of electrical systems. These libraries enable users to evaluate analogue circuit architectures effectively.

https://uk.mathworks.com/discovery/power-electronics-simulation.html

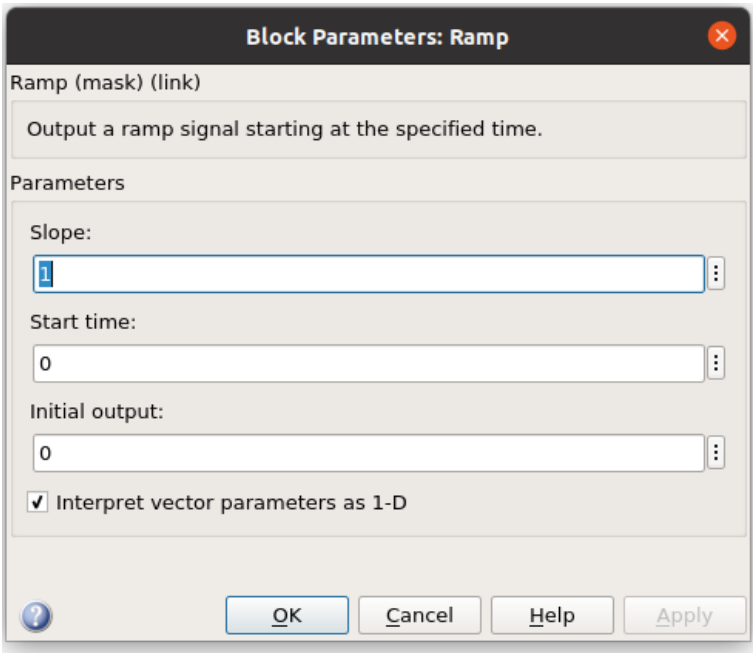
One of the primary advantages of using Simscape Electrical is its ability to conduct system-level simulations. This approach allows engineers to assess the performance of electrical systems before committing to more detailed SPICE models, which can be time-consuming. Users can parameterize their models using MATLAB variables and expressions, facilitating seamless integration with control systems designed in Simulink. This integration enhances the capability to develop complex control strategies for electrical systems.

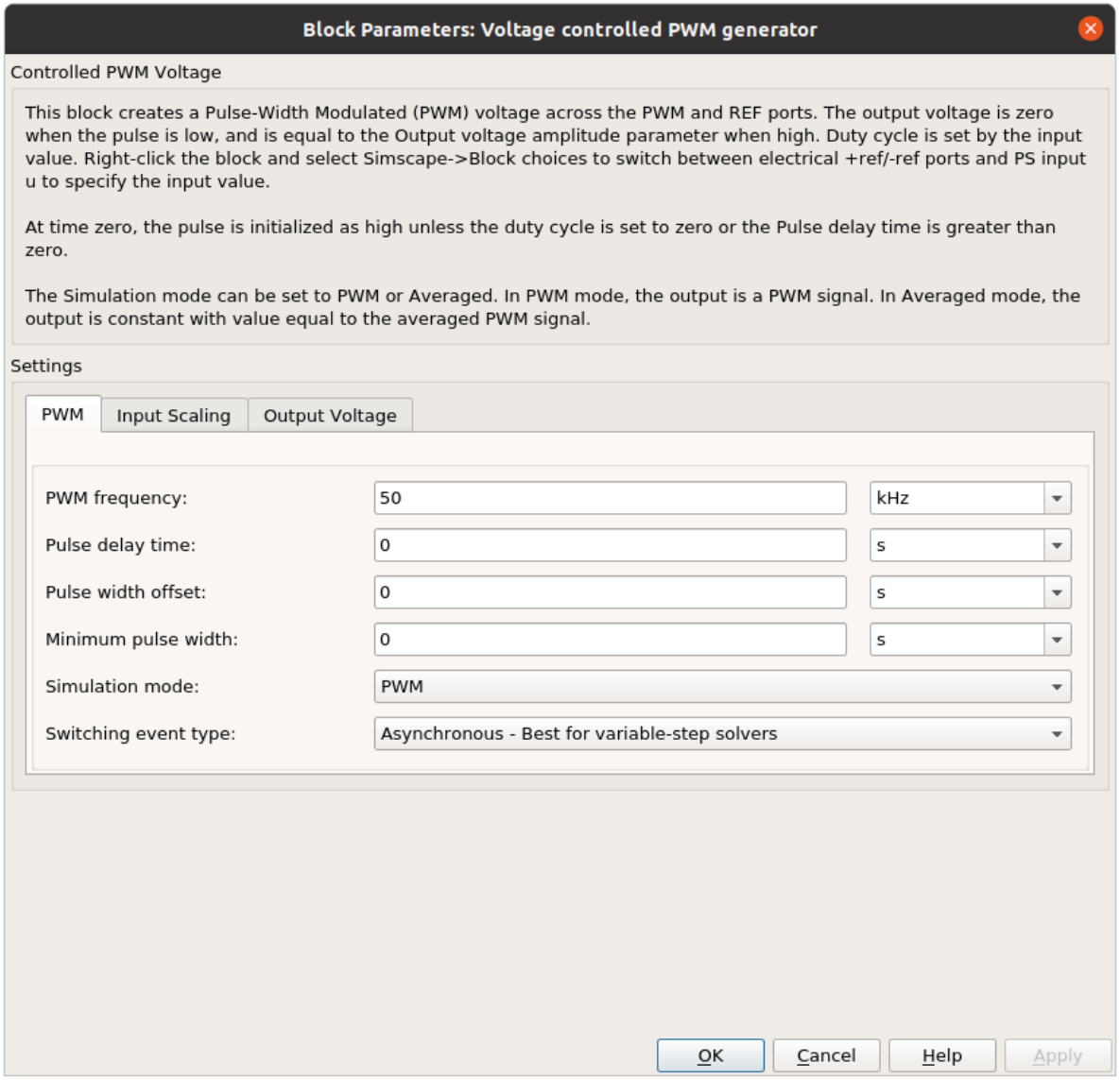
GOLDEN MODEL

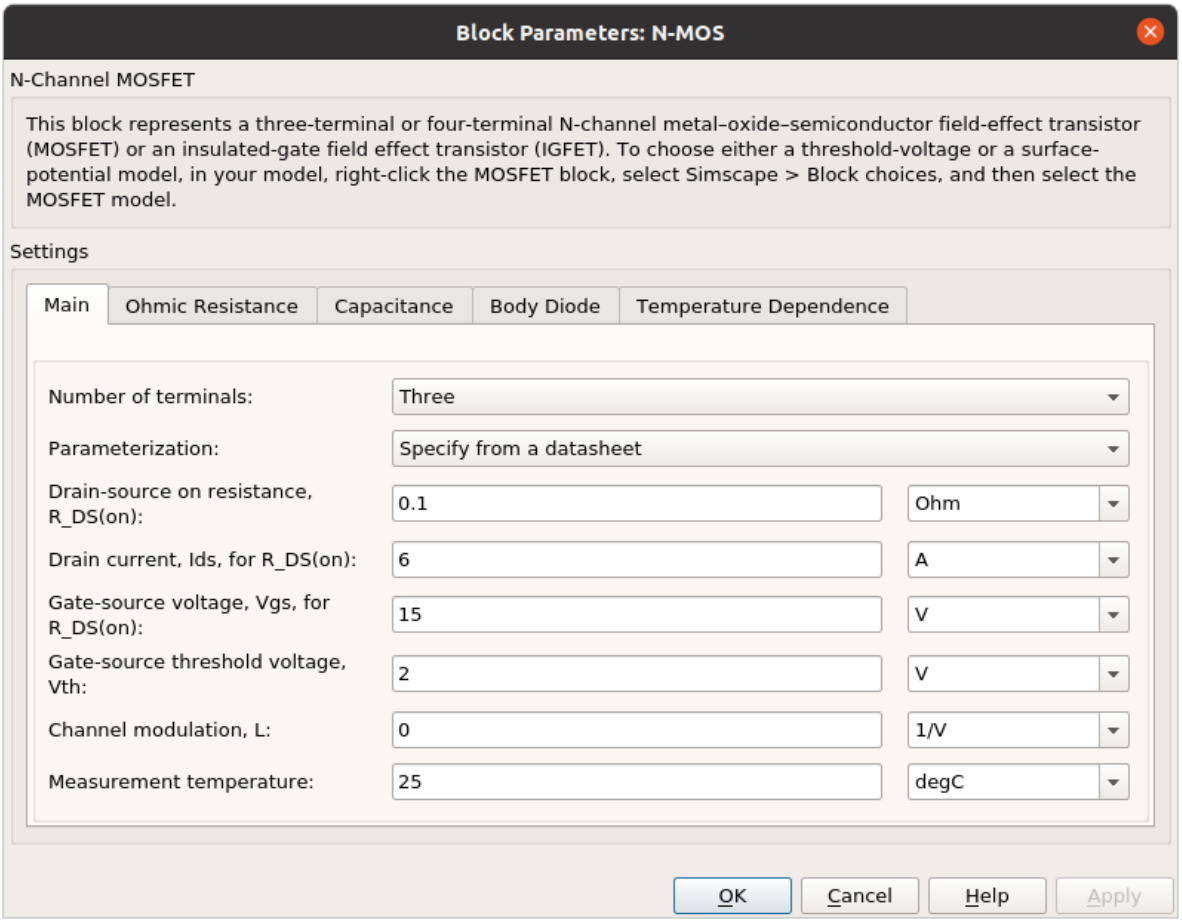
The “golden model” is the following Simulink model:

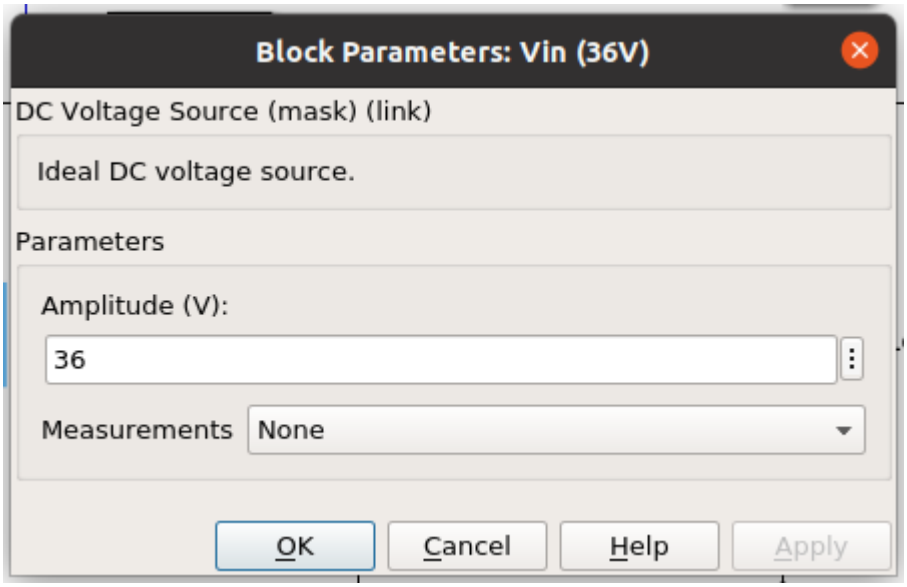


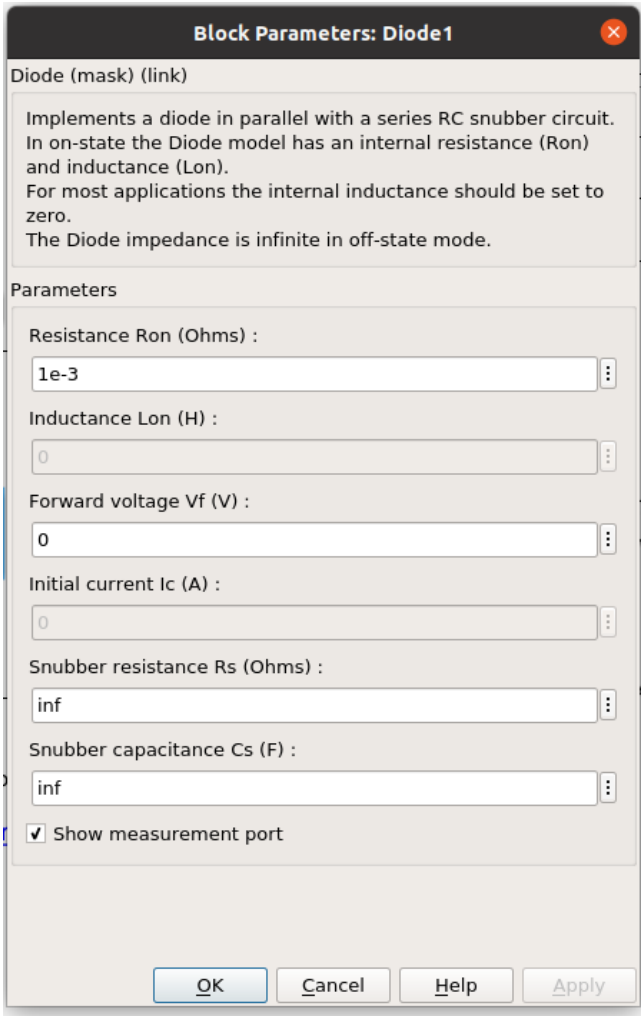
And is a slight variation of one of the many example models that come with Simscape Electrical.
A ramp is used to sweep all the possible PWM duty cycles.
It drives the PWM block that in turn drives the MOSFET.
The starting DC voltage is 36 V.
For the time being the load is resistive.
Details about key blocks are:

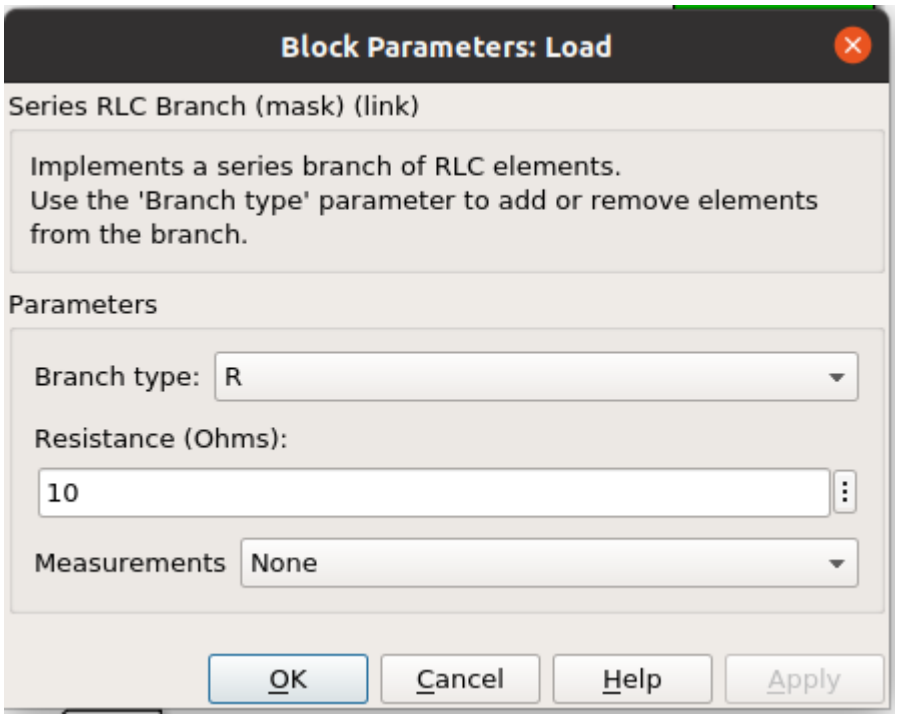


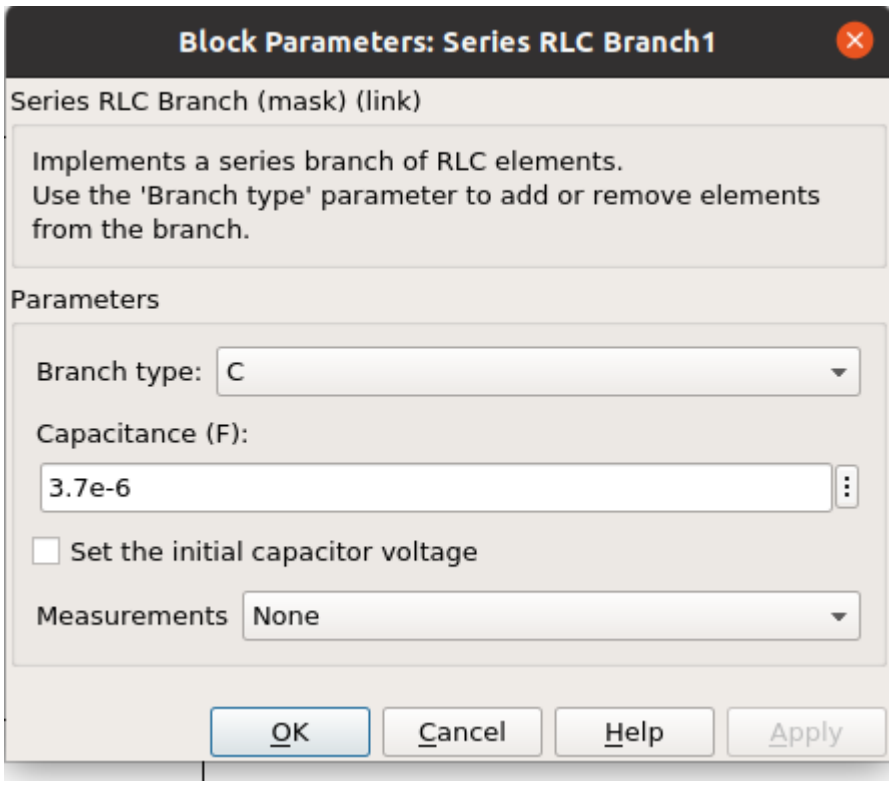

Block Parameters: Voltage controlled PWM generator
Controlled PWM Voltage
This block creates a Pulse-Width Modulated (PWM) voltage across the PWM and REF ports. The output voltage is zero when the pulse is low, and is equal to the Output voltage amplitude parameter when high. Duty cycle is set by the input value. Right-click the block and select Simscape->Block choices to switch between electrical +ref/-ref ports and PS input u to specify the input value.
At time zero, the pulse is initialized as high unless the duty cycle is set to zero or the Pulse delay time is greater than zero.
The Simulation mode can be set to PWM or Averaged. In PWM mode, the output is a PWM signal. In Averaged mode, the output is constant with value equal to the averaged PWM signal.
Settings
PWM
Input Scaling
Output Voltage
PWM frequency:
50
kHz
Pulse delay time:
0
s
Pulse width offset:
0
s
Minimum pulse width:
0
s
Simulation mode:
PWM
Switching event type:
Asynchronous - Best for variable-step solvers
OK
Cancel
Help
Apply

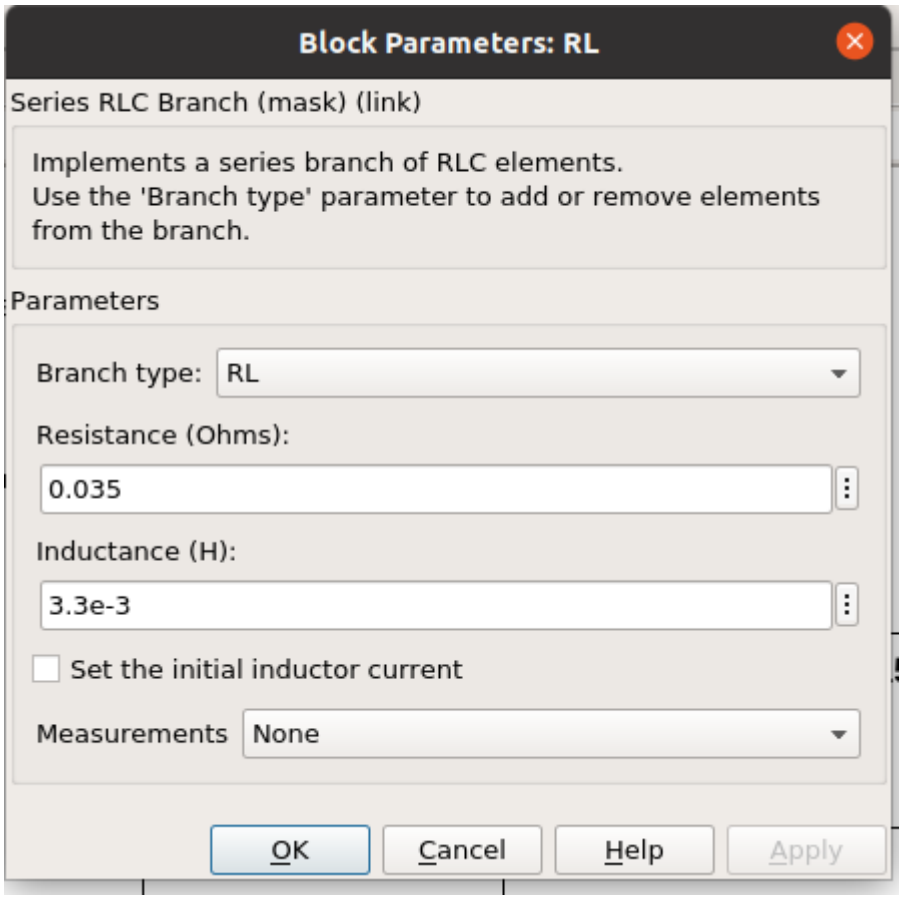

Block Parameters: N-MOS
N-Channel MOSFET
This block represents a three-terminal or four-terminal N-channel metal–oxide–semiconductor field-effect transistor (MOSFET) or an insulated-gate field effect transistor (IGFET). To choose either a threshold-voltage or a surface-potential model, in your model, right-click the MOSFET block, select Simscape > Block choices, and then select the MOSFET model.
Settings
Main
Ohmic Resistance
Capacitance
Body Diode
Temperature Dependence
Number of terminals:
Three
Parameterization:
Specify from a datasheet
Drain-source on resistance, R_DS(on):
0.1
Ohm
Drain current, Ids, for R_DS(on):
6
A
Gate-source voltage, Vgs, for R_DS(on):
15
V
Gate-source threshold voltage, Vth:
2
V
Channel modulation, L:
0
1/V
Measurement temperature:
25
degC
OK
Cancel
Help
Apply


Block Parameters: Vin (36V)
DC Voltage Source (mask) (link)
Ideal DC voltage source.
Parameters
Amplitude (V):
36
Measurements
None
OK
Cancel
Help
Apply

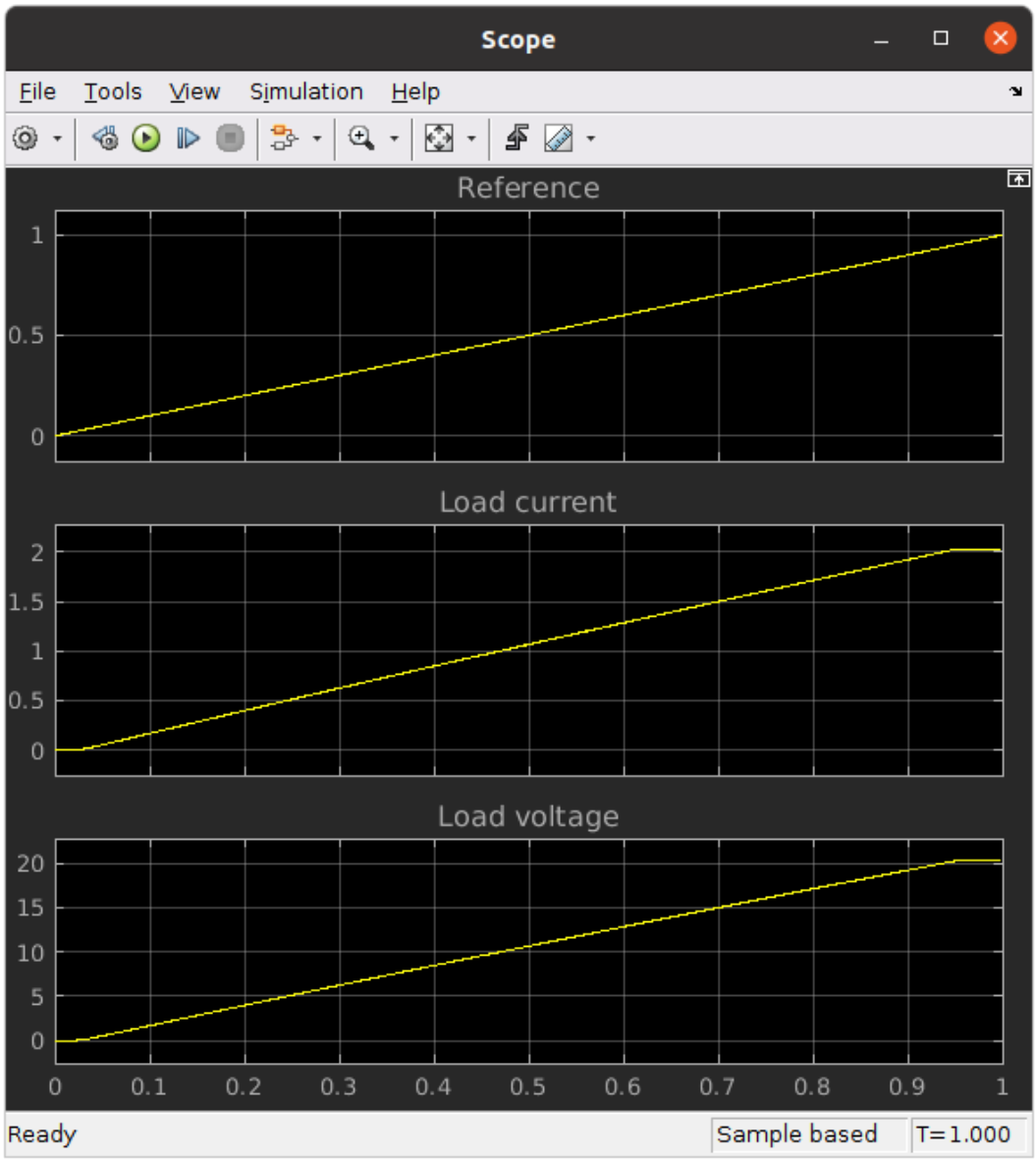

Block Parameters: Diode1
Diode (mask) (link)
Implements a diode in parallel with a series RC snubber circuit.
In on-state the Diode model has an internal resistance (Ron)
and inductance (Lon).
For most applications the internal inductance should be set to
zero.
The Diode impedance is infinite in off-state mode.
Parameters
Resistance Ron (Ohms) :
1e-3
Inductance Lon (H) :
0
Forward voltage Vf (V) :
0
Initial current Ic (A) :
0
Snubber resistance Rs (Ohms) :
inf
Snubber capacitance Cs (F) :
inf
Show measurement port
OK
Cancel
Help
Apply


Block Parameters: Load
Series RLC Branch (mask) (link)
Implements a series branch of RLC elements.
Use the 'Branch type' parameter to add or remove elements
from the branch.
Parameters
Branch type:
R
Resistance (Ohms):
10
Measurements
None
OK
Cancel
Help
Apply


Block Parameters: Series RLC Branch1
Series RLC Branch (mask) (link)
Implements a series branch of RLC elements.
Use the 'Branch type' parameter to add or remove elements from the branch.
Parameters
Branch type: C
Capacitance (F):
3.7e-6
Set the initial capacitor voltage
Measurements None
OK
Cancel
Help
Apply

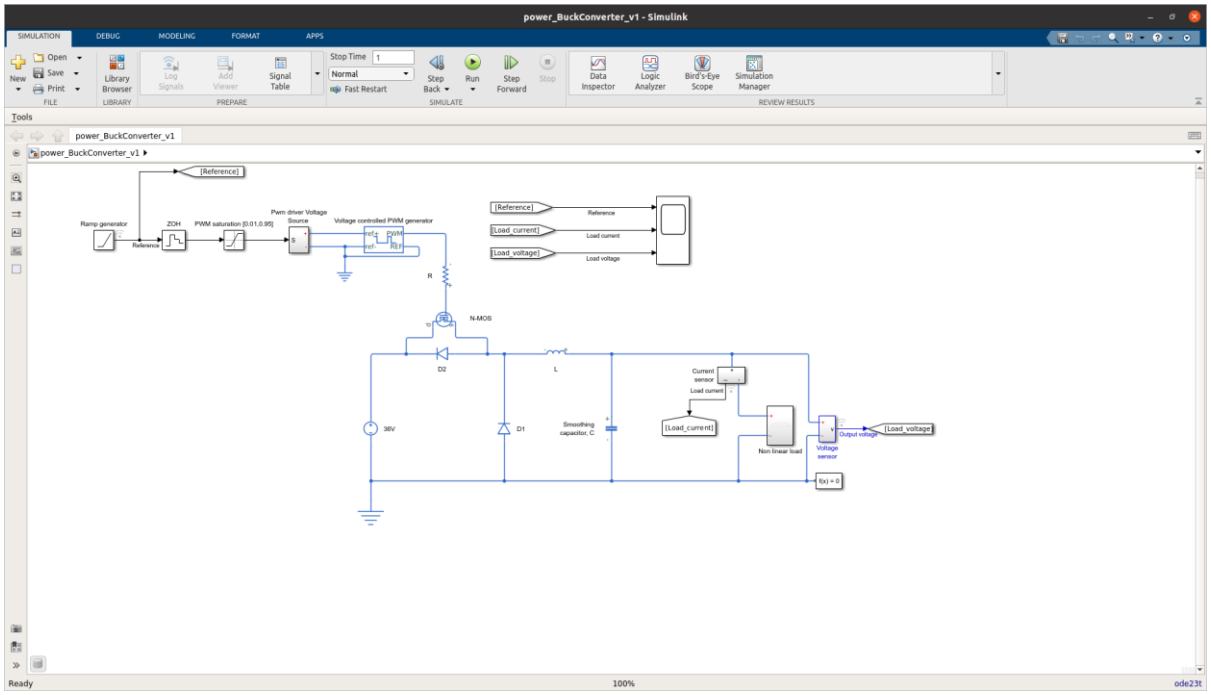

Block Parameters: RL
Series RLC Branch (mask) (link)
Implements a series branch of RLC elements.
Use the 'Branch type' parameter to add or remove elements from the branch.
Parameters
Branch type: RL
Resistance (Ohms):
0.035
Inductance (H):
3.3e-3
Set the initial inductor current
Measurements None
OK
Cancel
Help
Apply

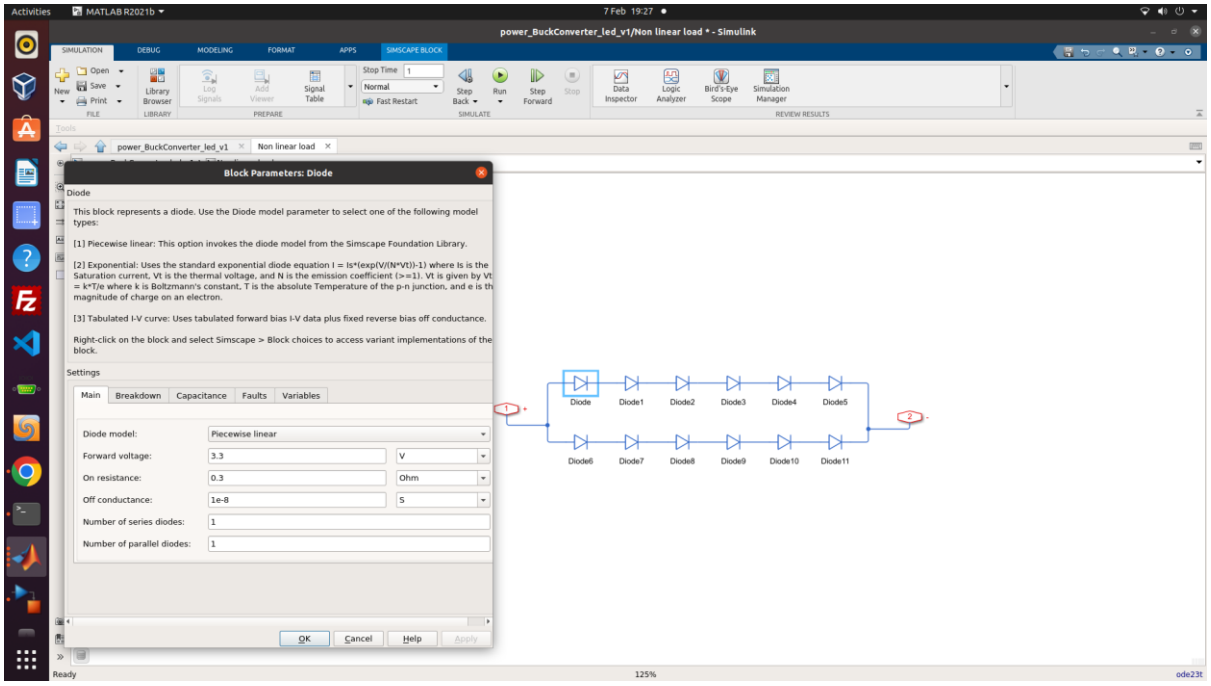
By sweeping the duty cycle from the minimum to maximum allowable values, we can see that the output current and voltage increase:



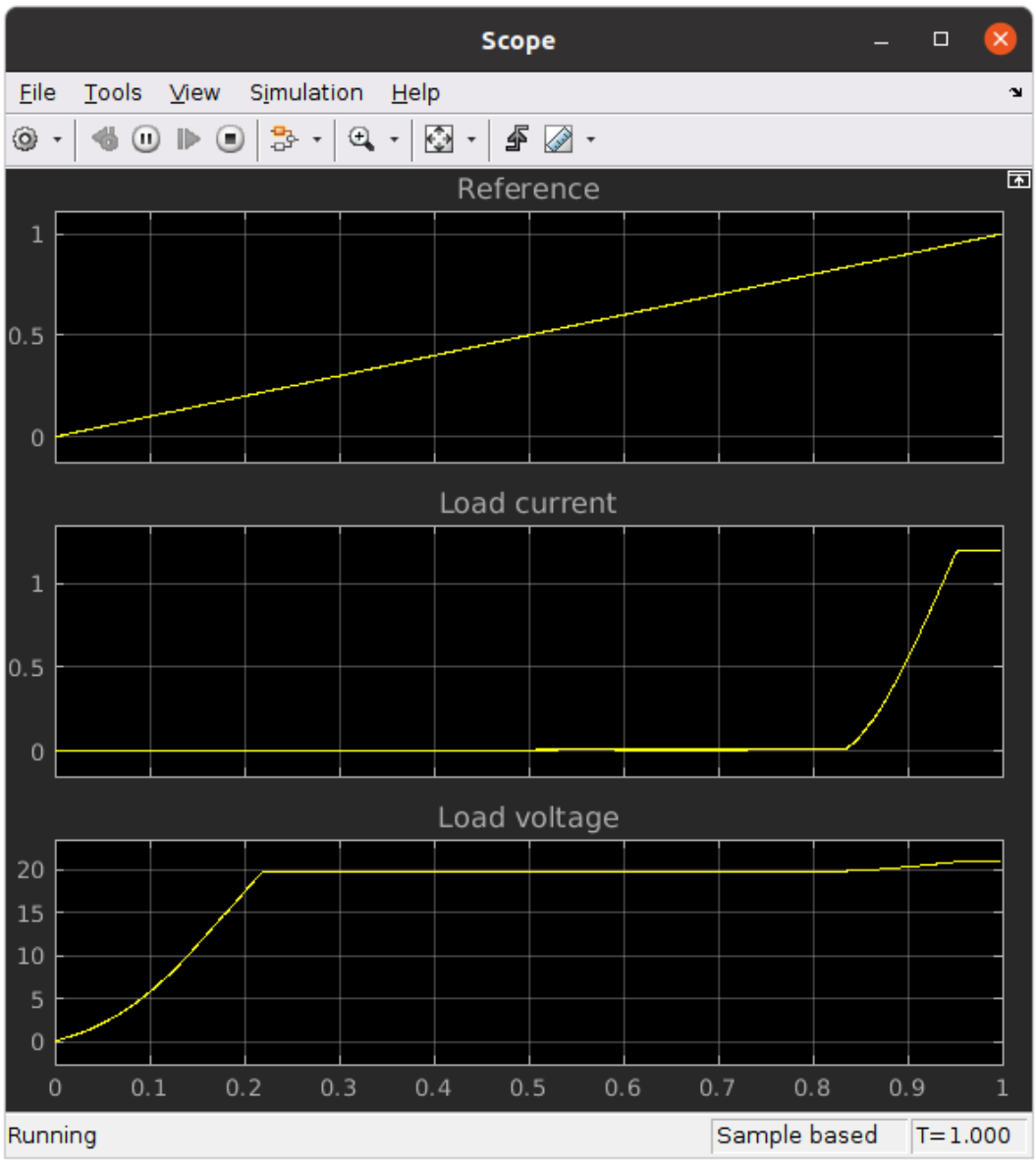
LED LOAD

Starting from the golden model, the resistive load is replaced with the equivalent model of the LCD LED backlight unit:





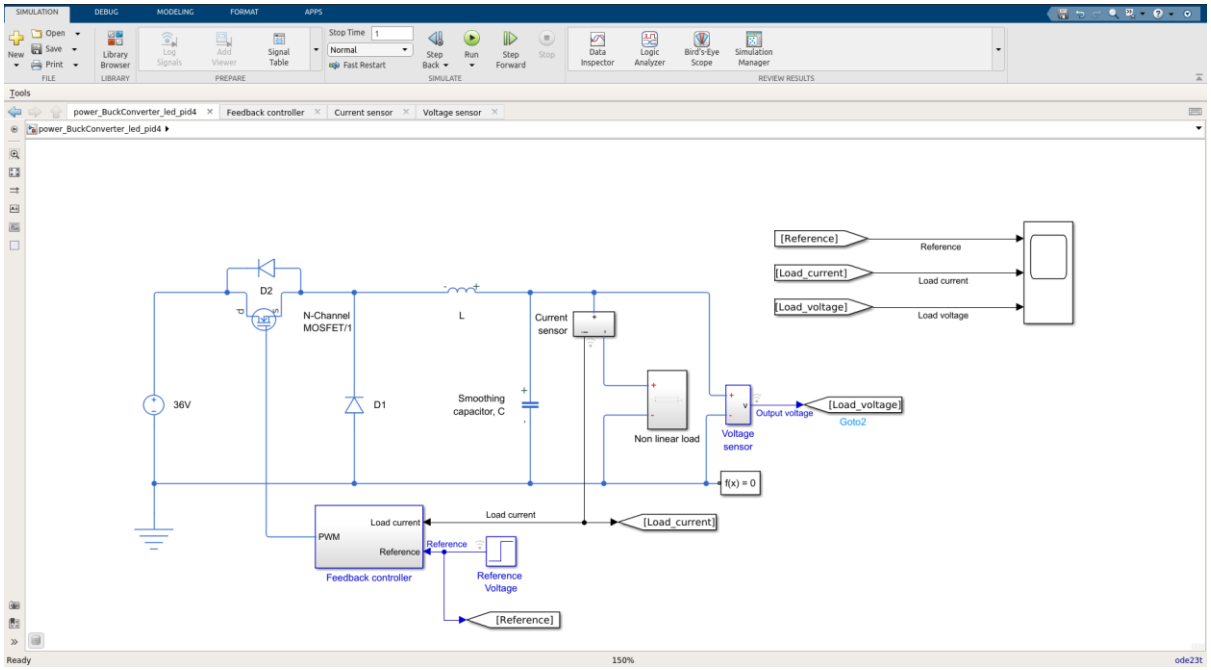
By running the model with a ramp driving the PWM generator, exercised from the minimum to maximum allowable duty cycle range, we get:



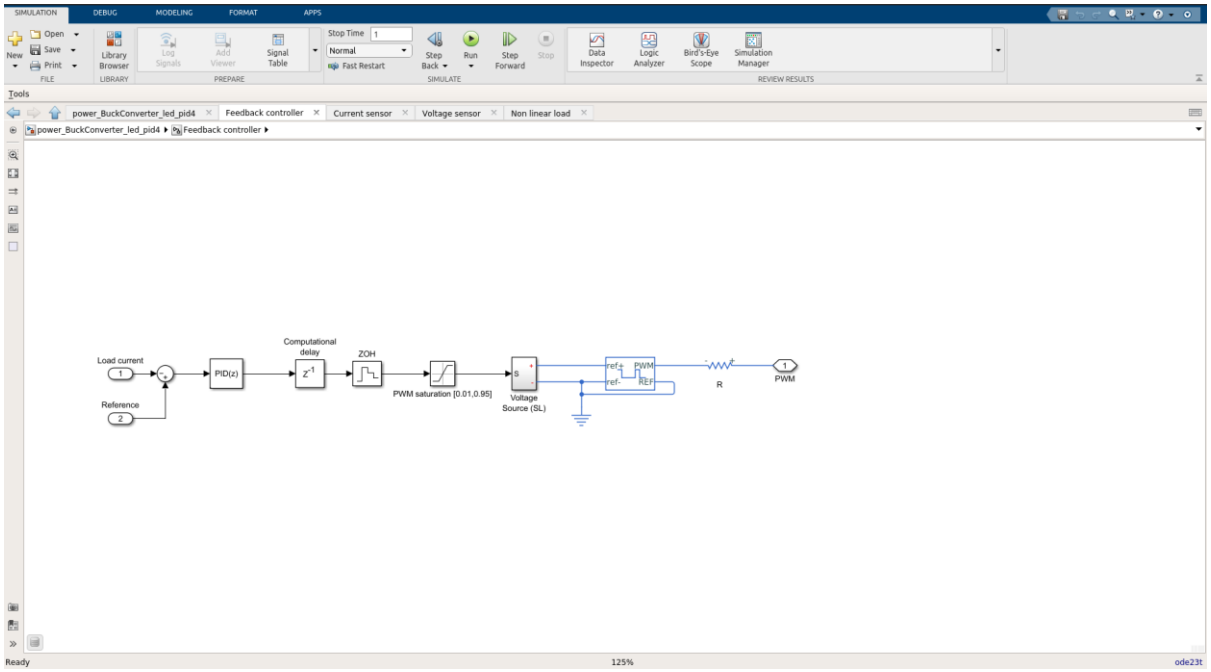
The current and voltage behaviour are nonlinear.

PID CONTROLER

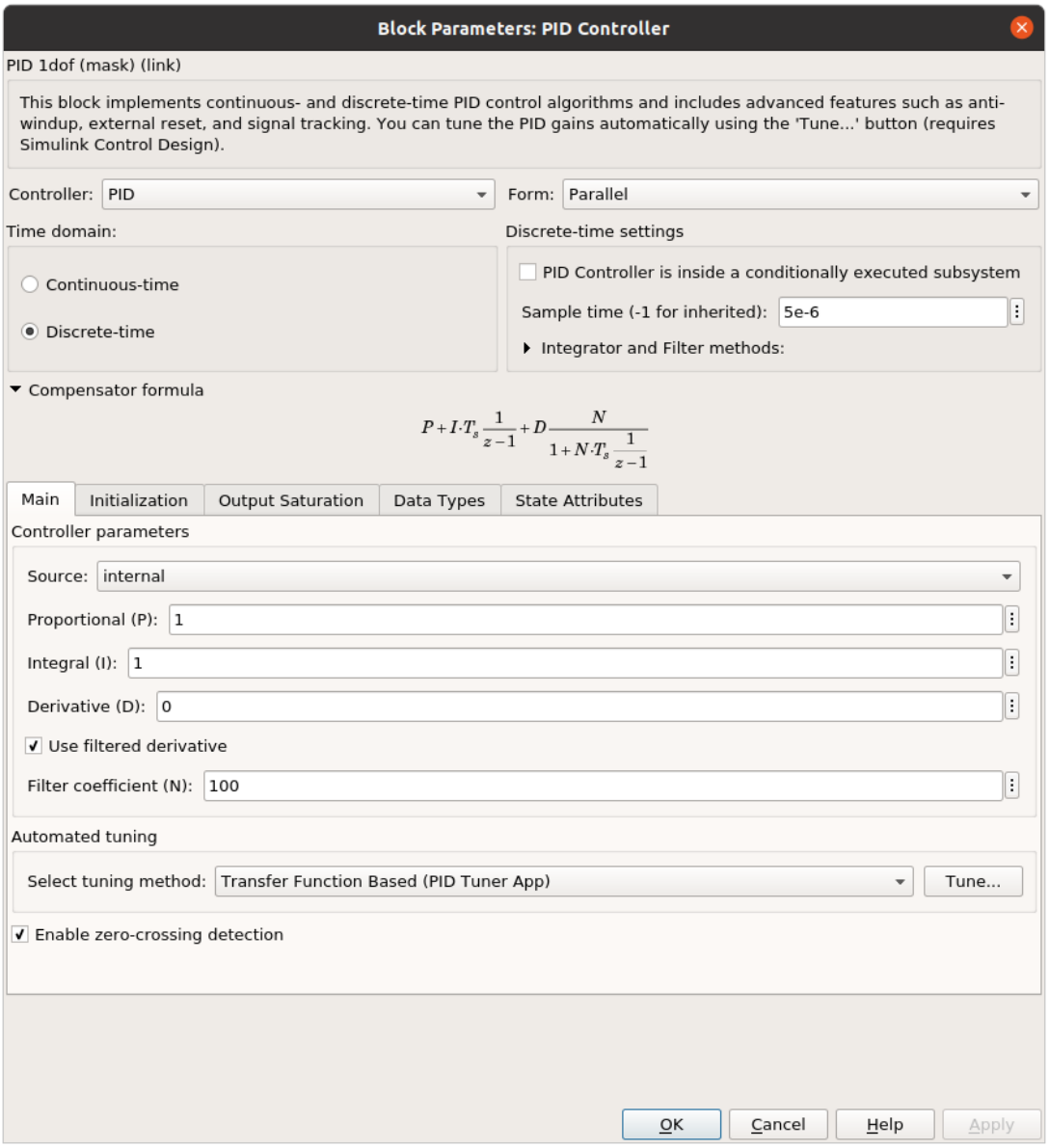
The golden model has been used as a starting point for the addition of a discrete time PID controller:



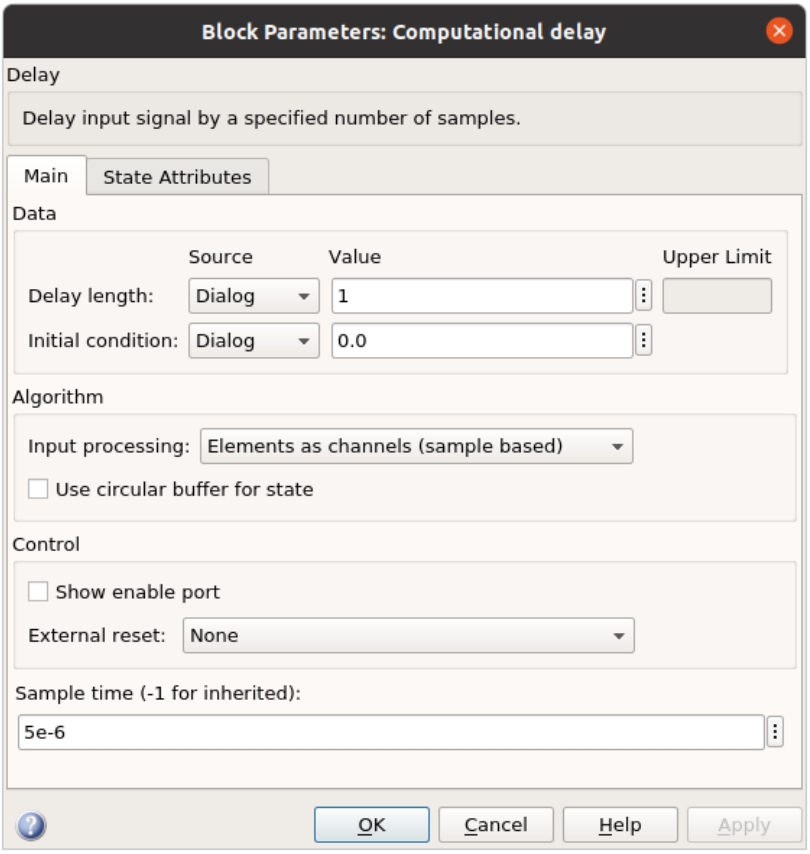
Details about the feedback controller follow:



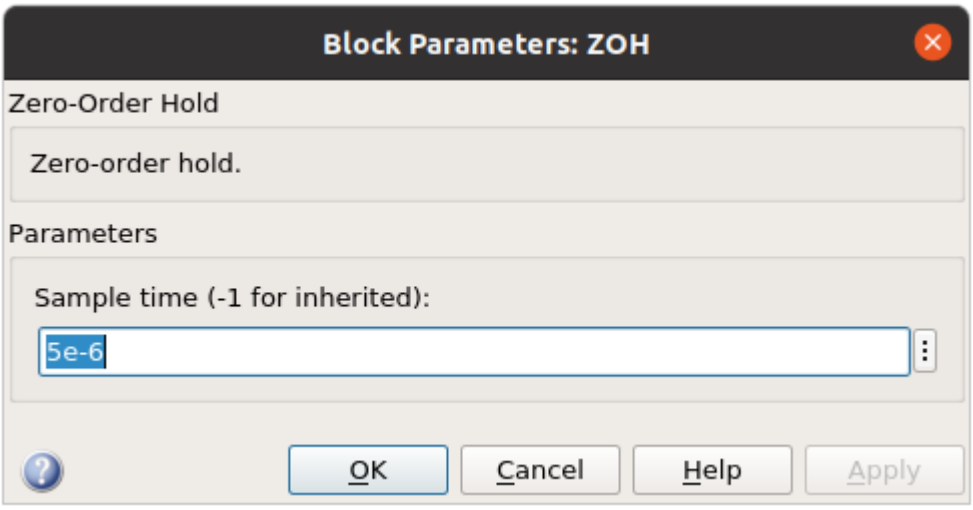
The current in the LEDs is being controlled via a discrete time PID.
Details about the PID follow:



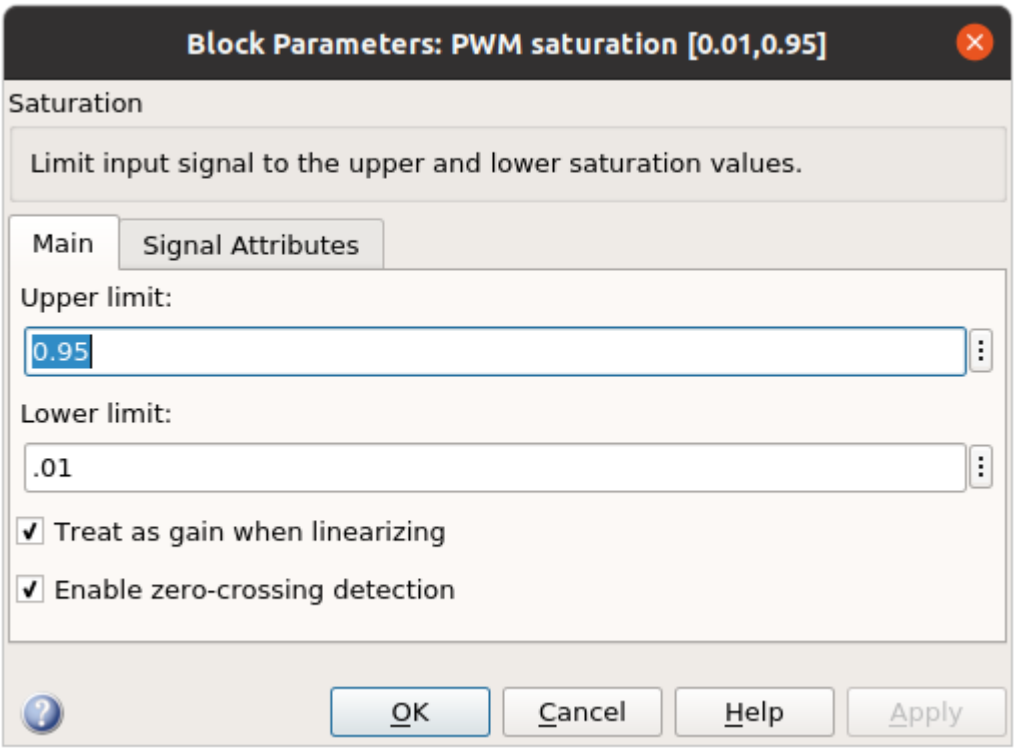
The PID sampling time is four times the 50 KHz PWM frequency.
Computational delays and sampling effects are modelled via a delay block:



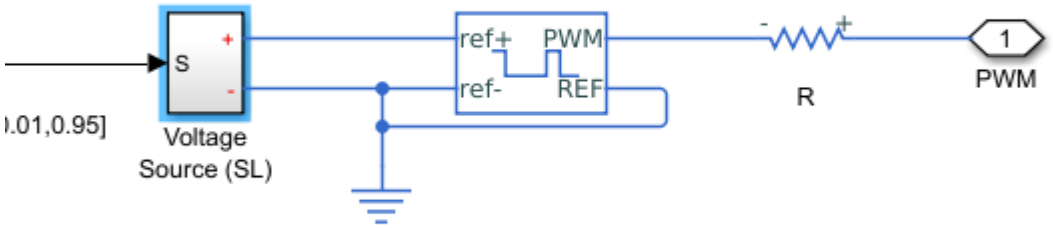
Sampling effects are modelled via a zero-order hold:

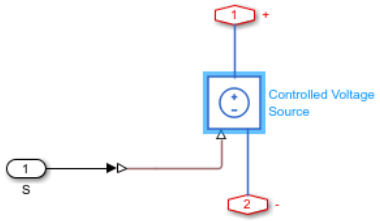


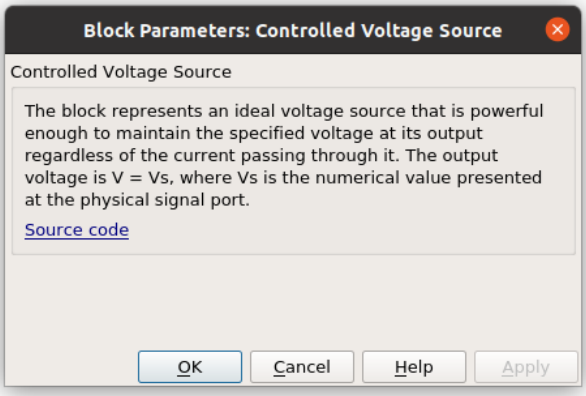
Saturation on the PWM driving signal is added:

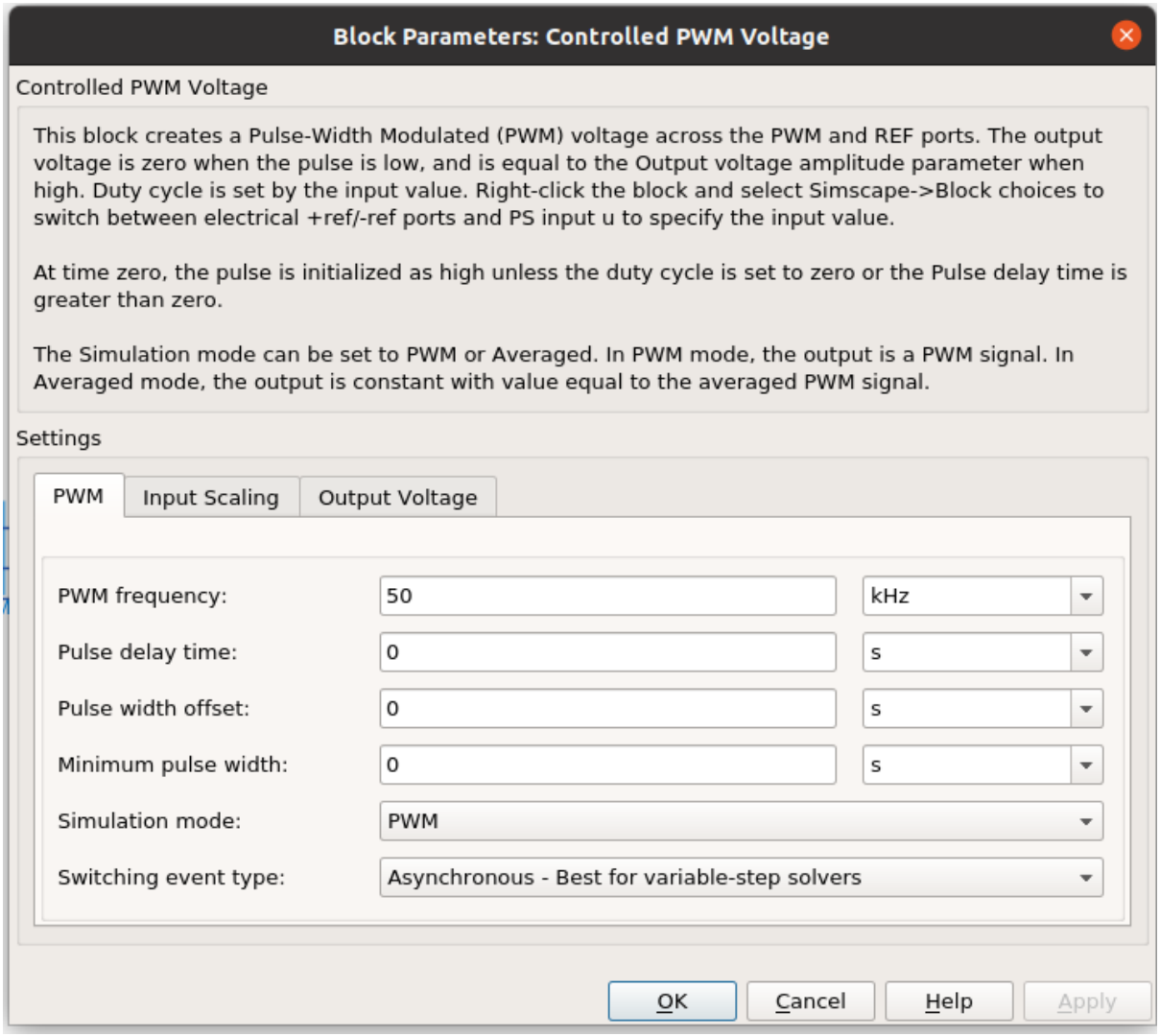


The following blocks in the diagram are modelling a PWM driver and a PWM converter.

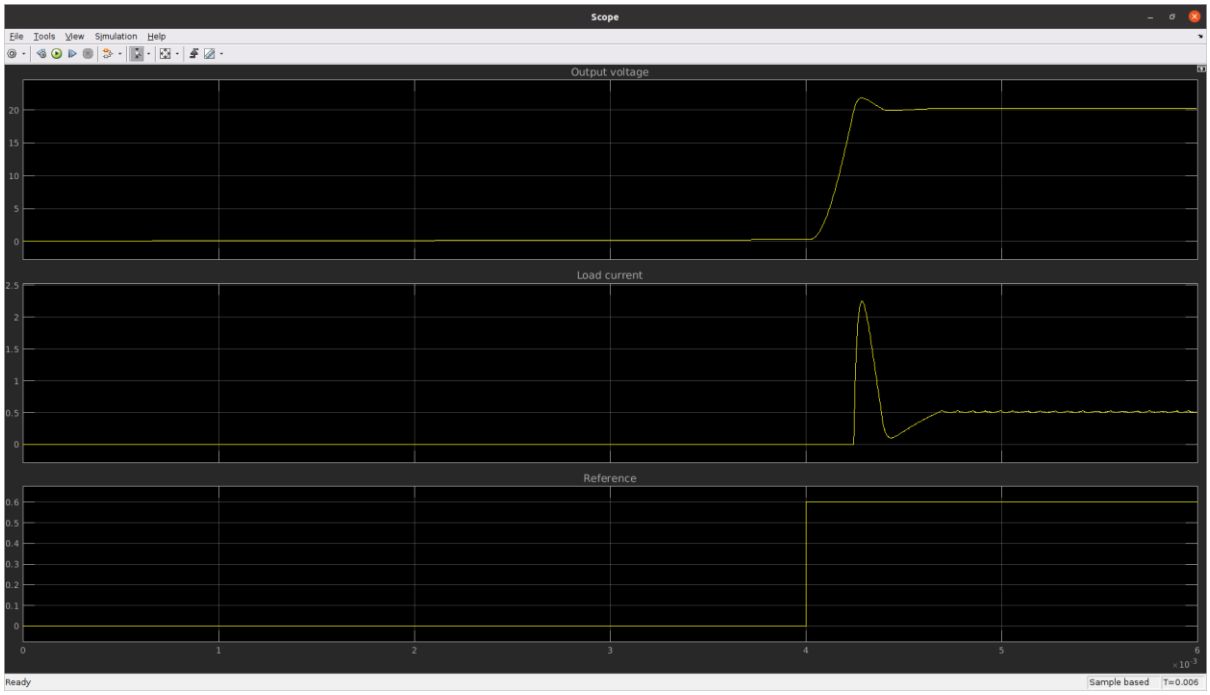


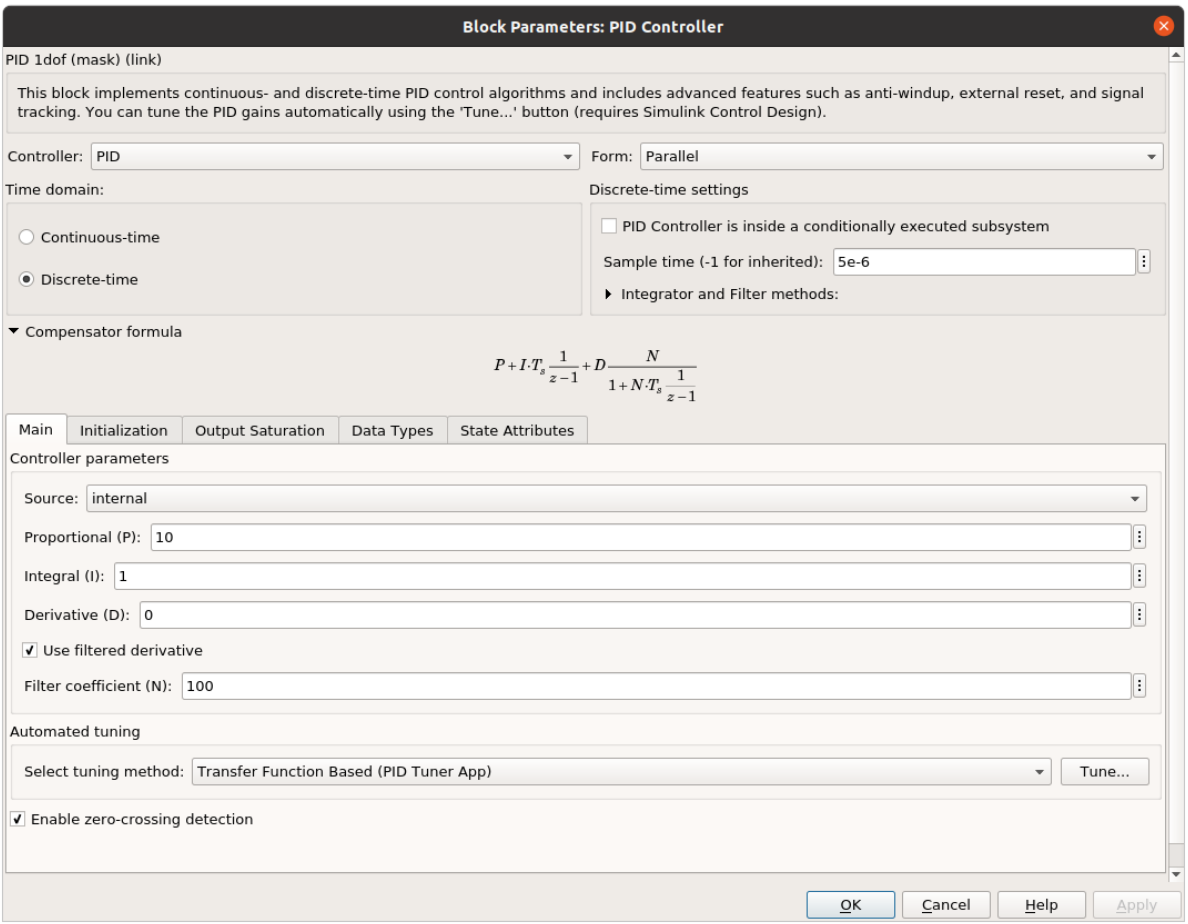

1
+
Controlled Voltage
Source
1
s
2
-


Block Parameters: Controlled Voltage Source
Controlled Voltage Source
The block represents an ideal voltage source that is powerful enough to maintain the specified voltage at its output regardless of the current passing through it. The output voltage is V = Vs, where Vs is the numerical value presented at the physical signal port.
Source code
OK
Cancel
Help
Apply


Block Parameters: Controlled PWM Voltage
Controlled PWM Voltage
This block creates a Pulse-Width Modulated (PWM) voltage across the PWM and REF ports. The output voltage is zero when the pulse is low, and is equal to the Output voltage amplitude parameter when high. Duty cycle is set by the input value. Right-click the block and select Simscape->Block choices to switch between electrical +ref/-ref ports and PS input u to specify the input value.
At time zero, the pulse is initialized as high unless the duty cycle is set to zero or the Pulse delay time is greater than zero.
The Simulation mode can be set to PWM or Averaged. In PWM mode, the output is a PWM signal. In Averaged mode, the output is constant with value equal to the averaged PWM signal.
Settings
PWM
Input Scaling
Output Voltage
PWM frequency:
50
kHz
Pulse delay time:
0
s
Pulse width offset:
0
s
Minimum pulse width:
0
s
Simulation mode:
PWM
Switching event type:
Asynchronous - Best for variable-step solvers
OK
Cancel
Help
Apply

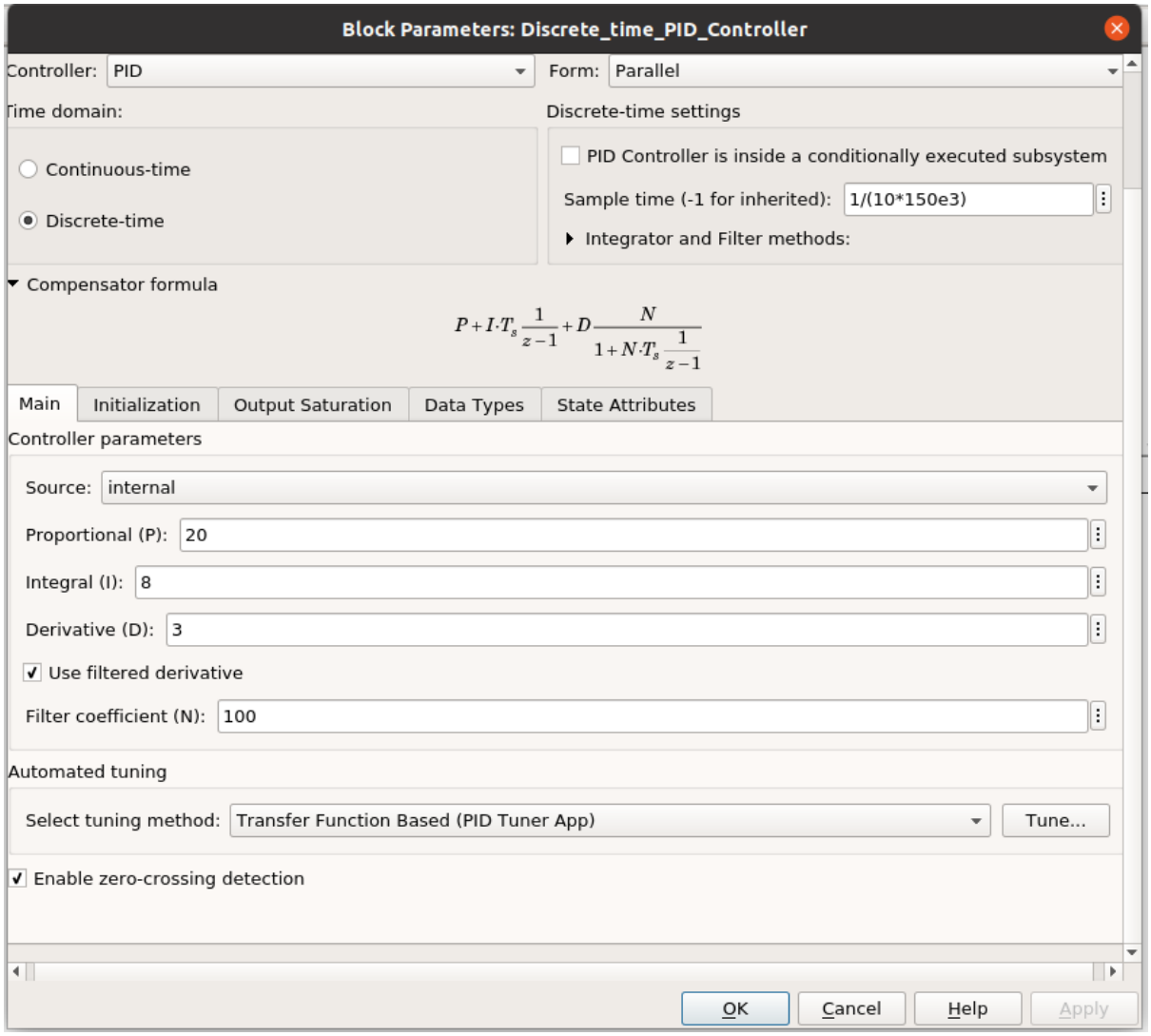
With a manual assignment of PID parameters shown below, we get a step response with overshooting:





PID AUTOTUNING

By opening the PID block and clicking the “tune” button:



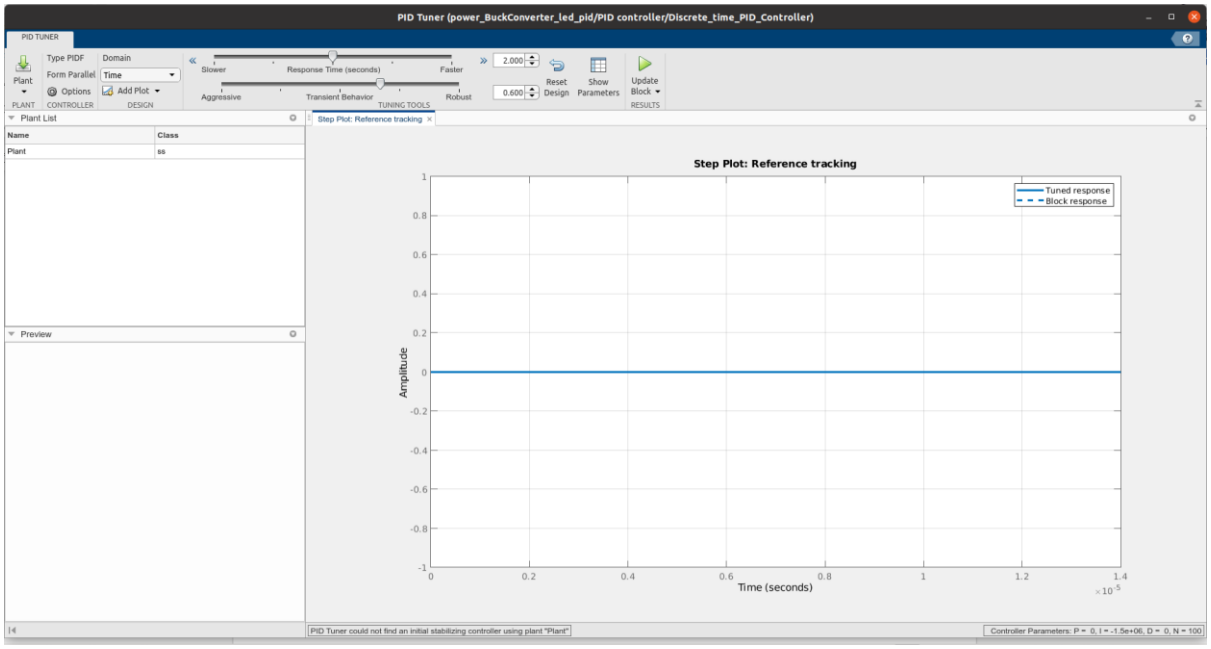
The tuner starts the tuning process for the PID, against the buck converter model.

To perform the tuning process, “System Identification Toolbox” and “Control System Toolbox” have been used:

https://uk.mathworks.com/products/sysid.html

https://uk.mathworks.com/products/control.html

By opening the following app:



As the model is nonlinear, the app does not work.

Linearizing a plant in Simulink is essential for simplifying complex nonlinear dynamics, applying established linear control techniques, performing stability analysis, evaluating performance metrics efficiently, and validating models against experimental data.

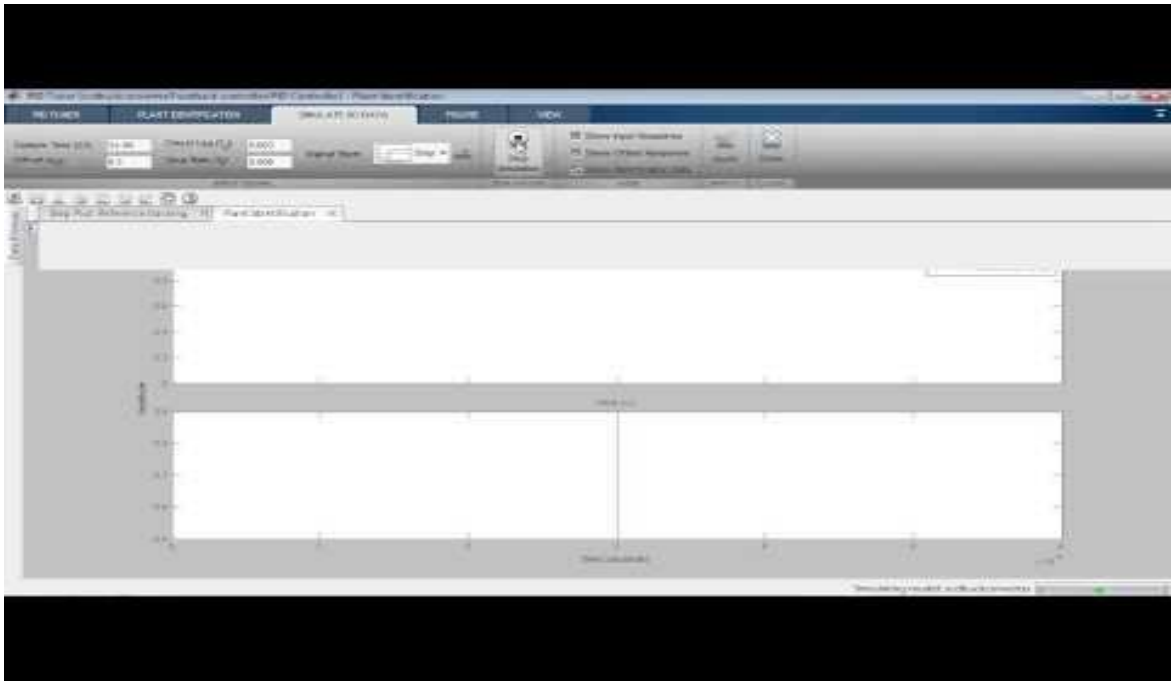
By following the instructions provided in:

https://uk.mathworks.com/help/slcontrol/ug/design-a-pid-controller-using-simulated-i-o-data.html

And in:

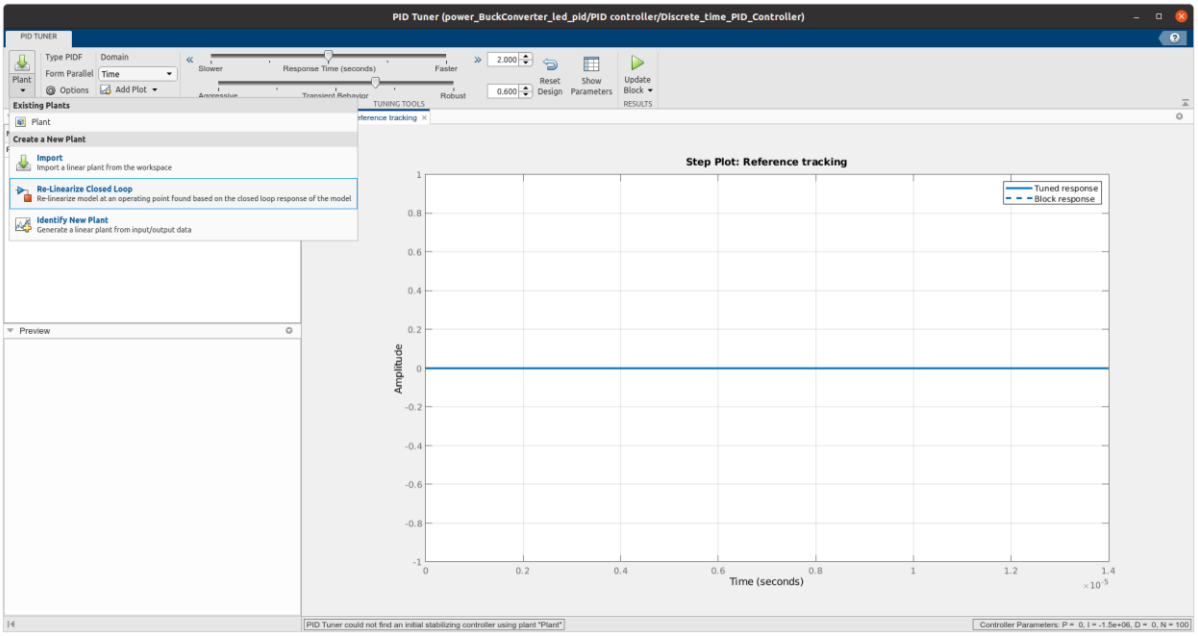
https://www.youtube.com/watch?v=5fXT7-8Rw84

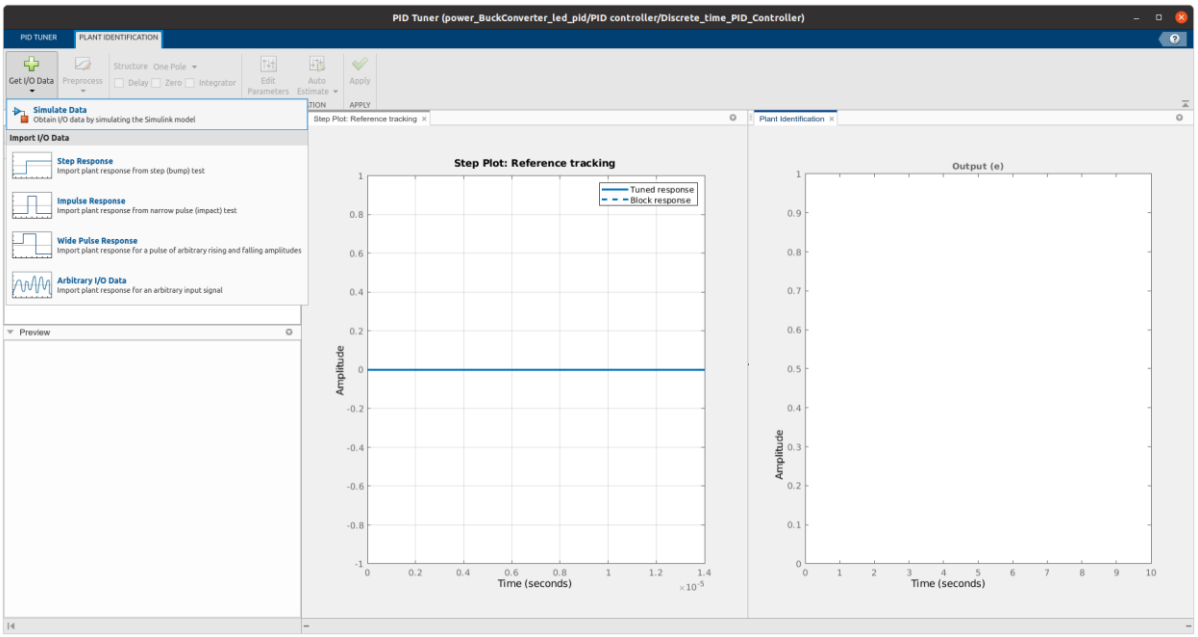
https://www.youtube.com/watch?v=TlaION68WAg&t=12s



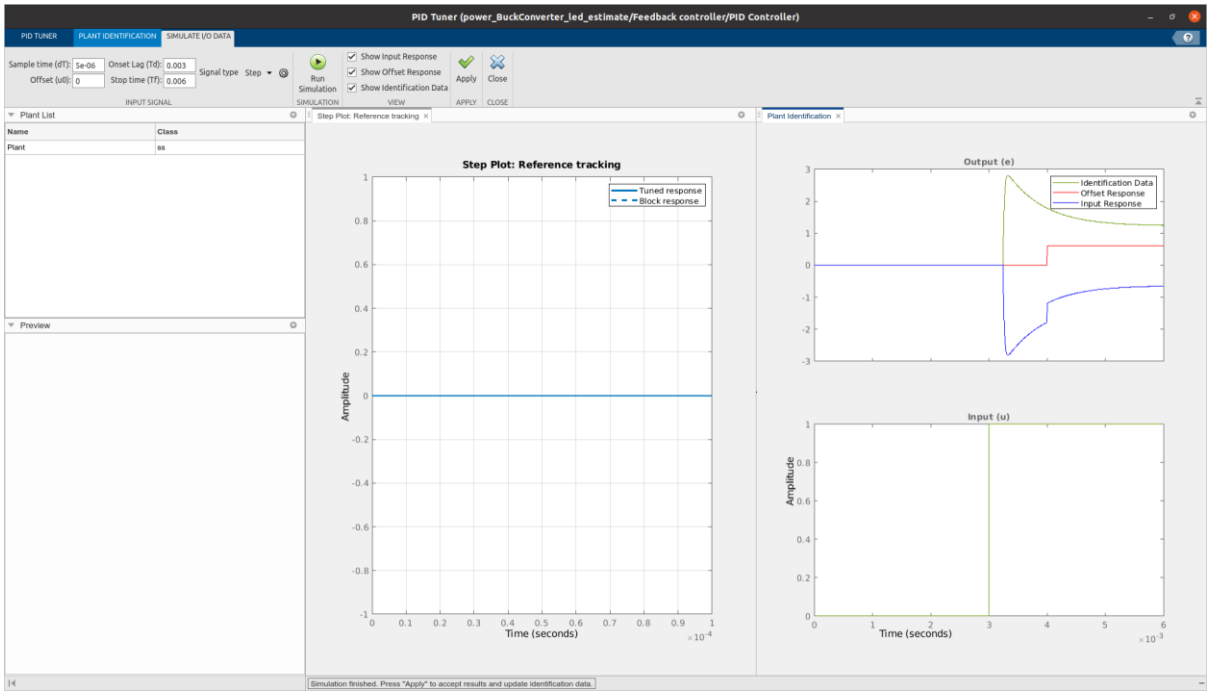
The tuning effort moves forward.

We need to identify the model of the buck converter, getting a linear estimate of the plant:

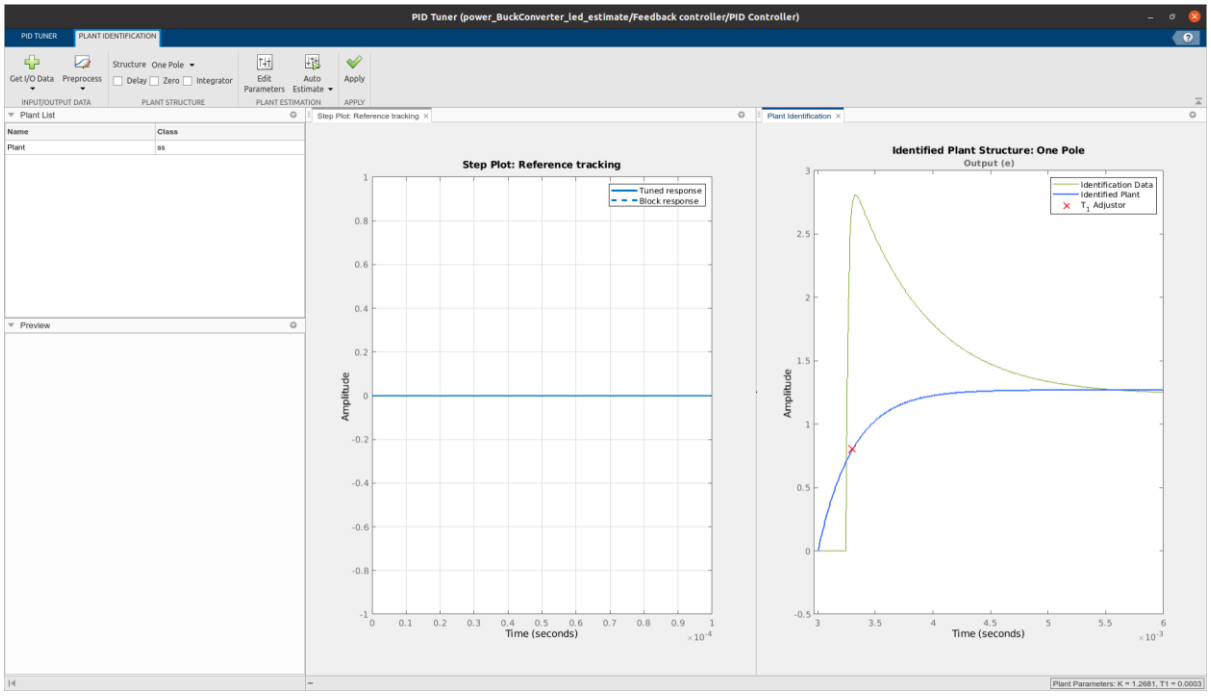




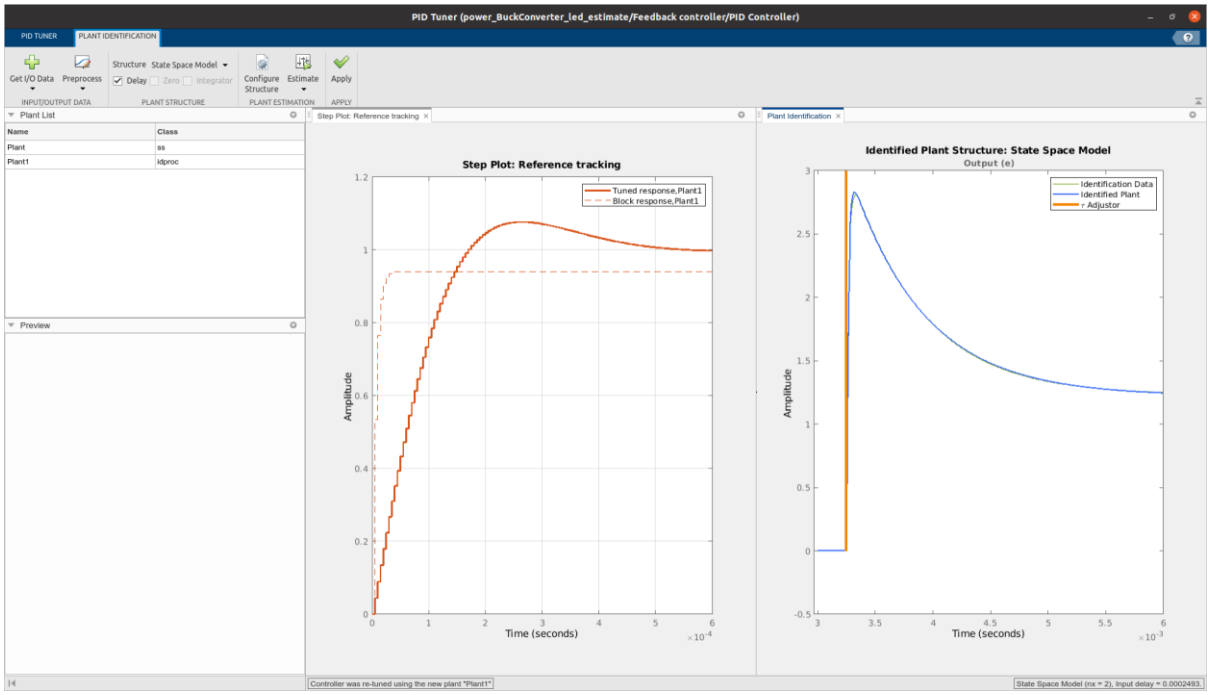
The model gets created by collecting I/O data via simulation:

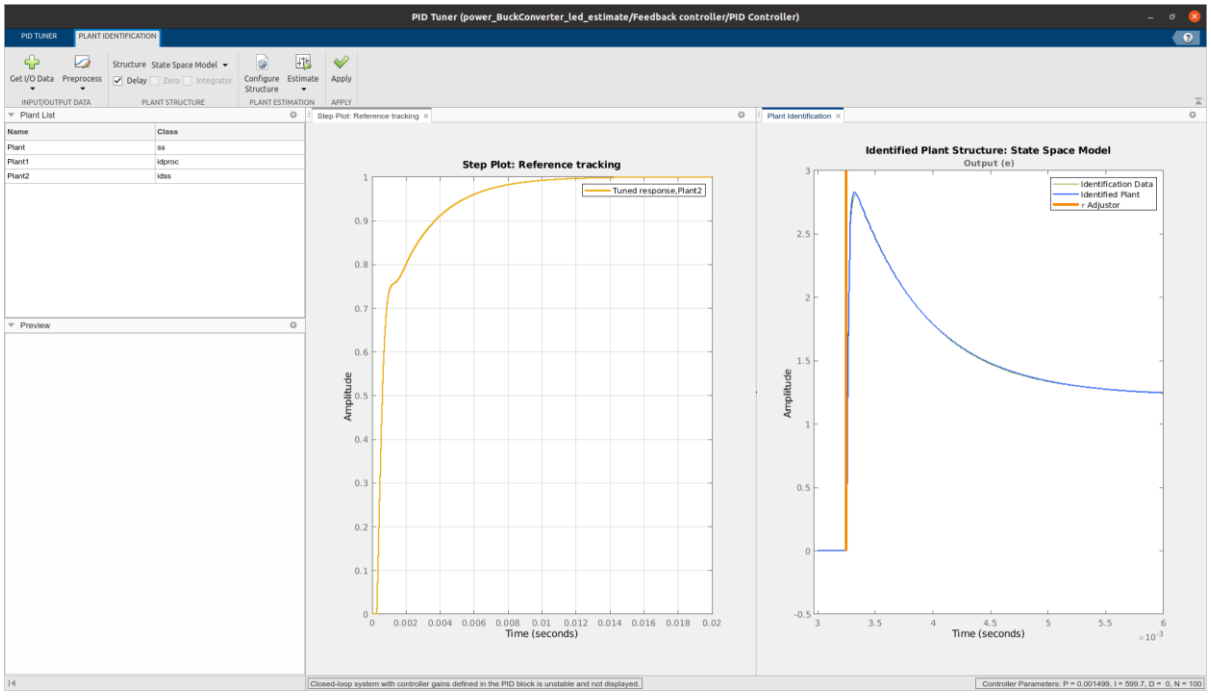


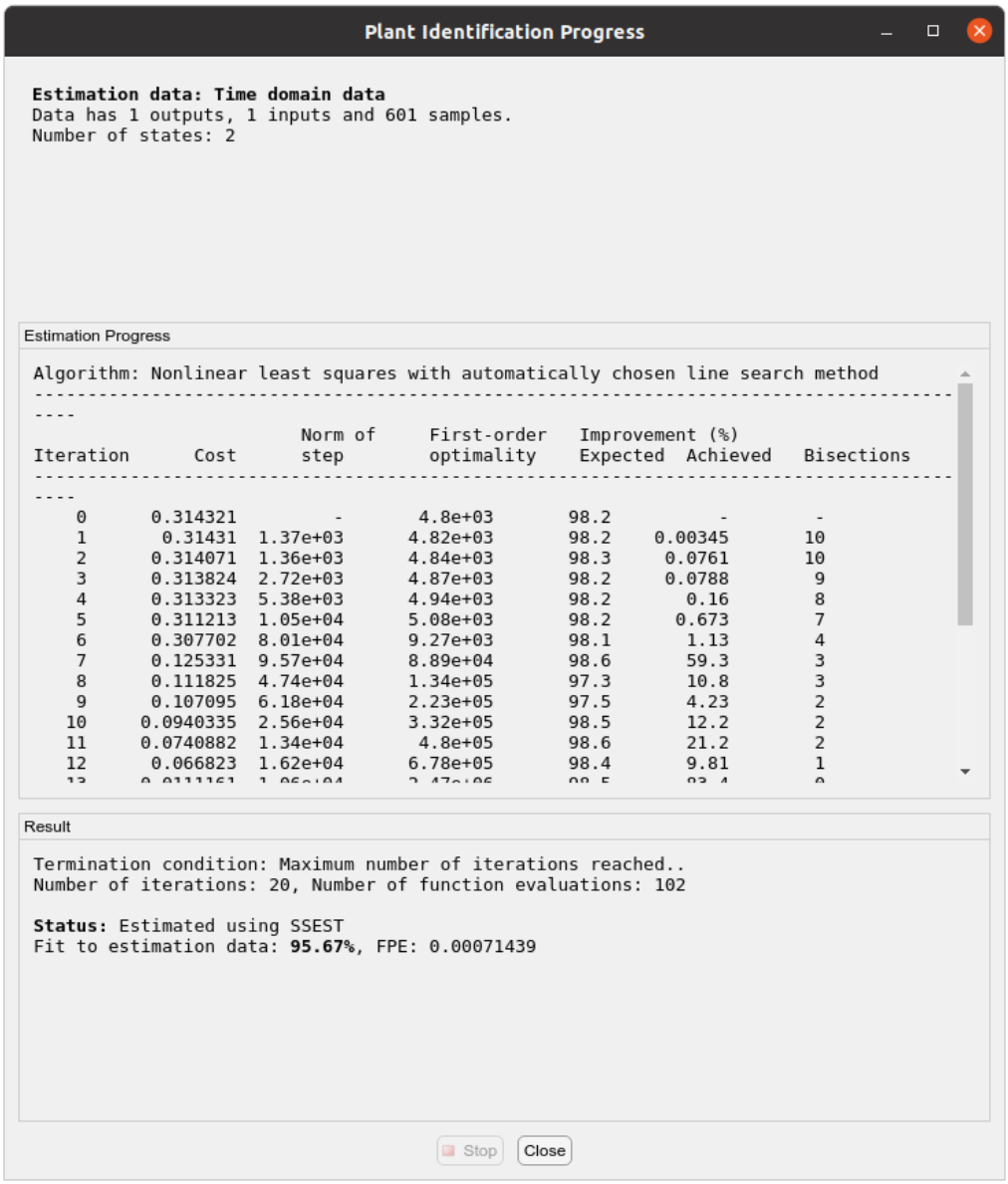
Applying the changes after the plant identification:



Pressing auto estimate, state space estimation with delay, the plant gets estimated:


PID Tuner (power_BuckConverter_led_estimate/Feedback controller/PID Controller)
PID TUNER
PLANT IDENTIFICATION
Get I/O Data
Preprocess
Structure
State Space Model
Delay
Zero
Integrator
Configure Structure
Estimate
Apply
INPUT/OUTPUT DATA
PLANT STRUCTURE
PLANT ESTIMATION
APPLY
Plant List
Name
Class
Plant
ss
Plant1
idproc
Preview
Step Plot: Reference tracking
Step Plot: Reference tracking
Tuned response,Plant1
Block response,Plant1
Amplitude
Time (seconds)
Plant Identification
Identified Plant Structure: State Space Model
Output (e)
Identification Data
Identified Plant
r Adjustor
Amplitude
Time (seconds)
Controller was re-tuned using the new plant "Plant1".
State Space Model (nx = 2), Input delay = 0.0002493.

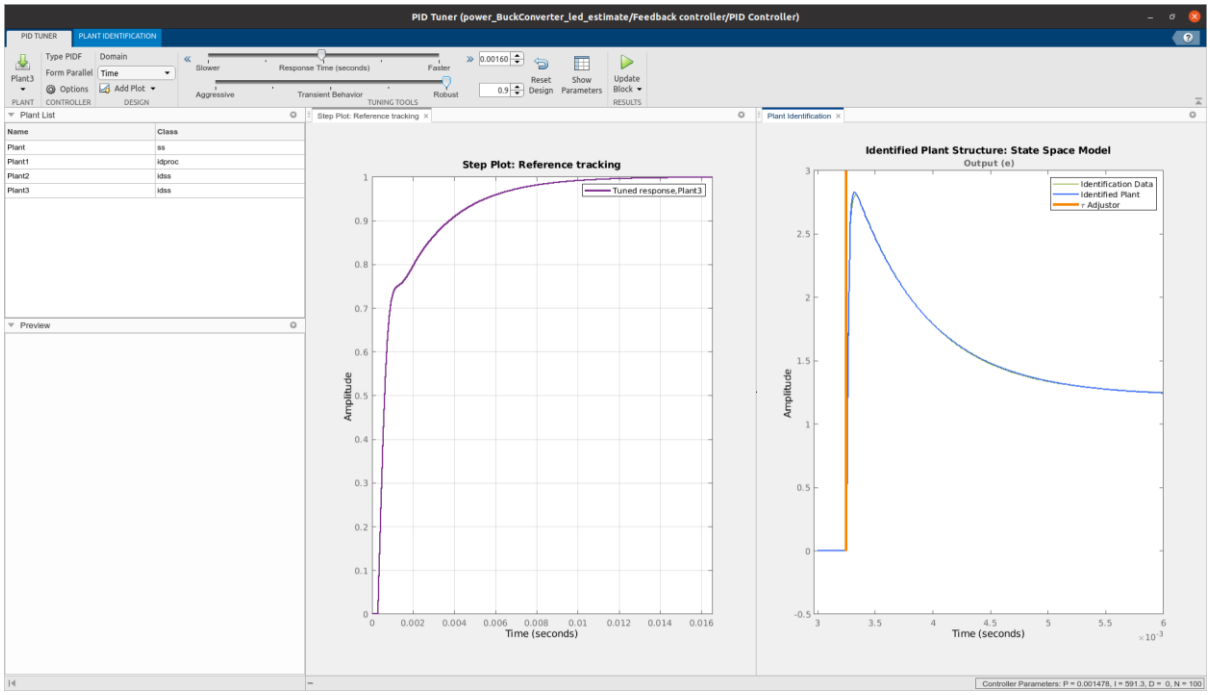

PID Tuner (power_BuckConverter_led_estimate/Feedback controller/PID Controller)
PID TUNER
PLANT IDENTIFICATION
Get I/O Data
Preprocess
Structure
State Space Model
Delay
Zero
Integrator
Configure Structure
Estimate
Apply
INPUT/OUTPUT DATA
PLANT STRUCTURE
PLANT ESTIMATION
APPLY
Plant List
Name
Class
Plant
ss
Plant1
idproc
Plant2
idss
Preview
Step Plot: Reference tracking
Step Plot: Reference tracking
Tuned response,Plant2
Amplitude
Time (seconds)
Plant Identification
Identified Plant Structure: State Space Model
Output (e)
Identification Data
Identified Plant
r Adjustor
Amplitude
Time (seconds)
Closed-loop system with controller gains defined in the PID block is unstable and not displayed.
Controller Parameters: P = 0.001499, I = 599.7, D = 0, N = 100



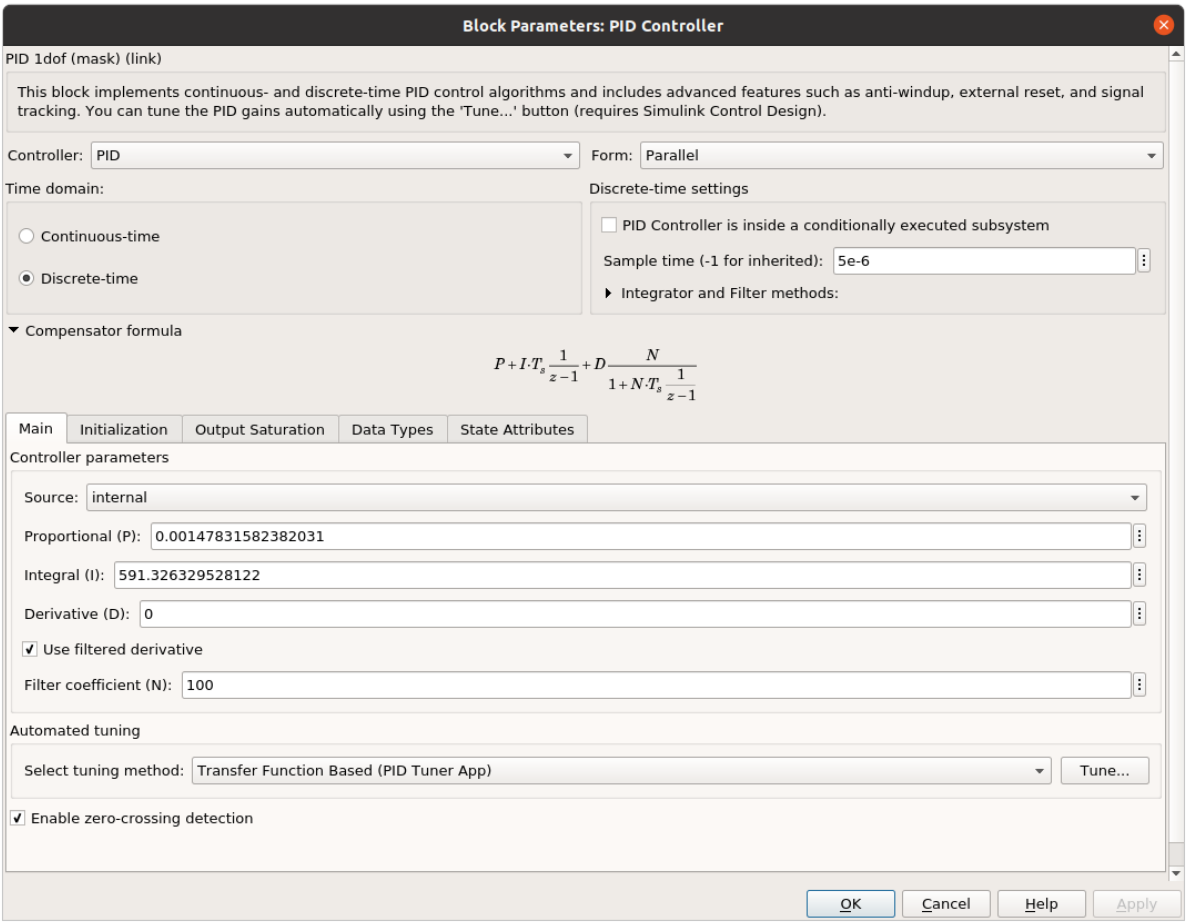
PID Tuner allows you to specify a plant structure, such as One Pole, Two Real Poles, or State-Space Model: the latter has given the best fit to simulation data.

https://uk.mathworks.com/help/slcontrol/ug/interactively-estimate-plant-from-measured-or-simulated-response-data.html

Now the PID tuning can be tackled:

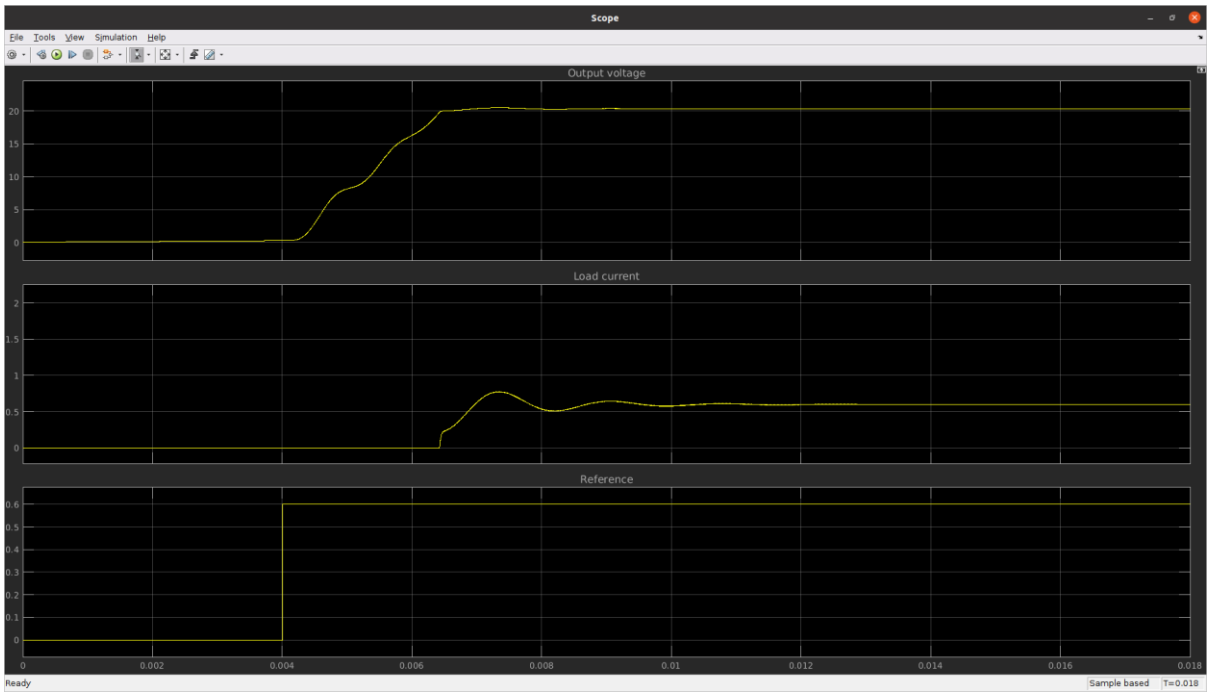


And changes to PID parameters can be applied:



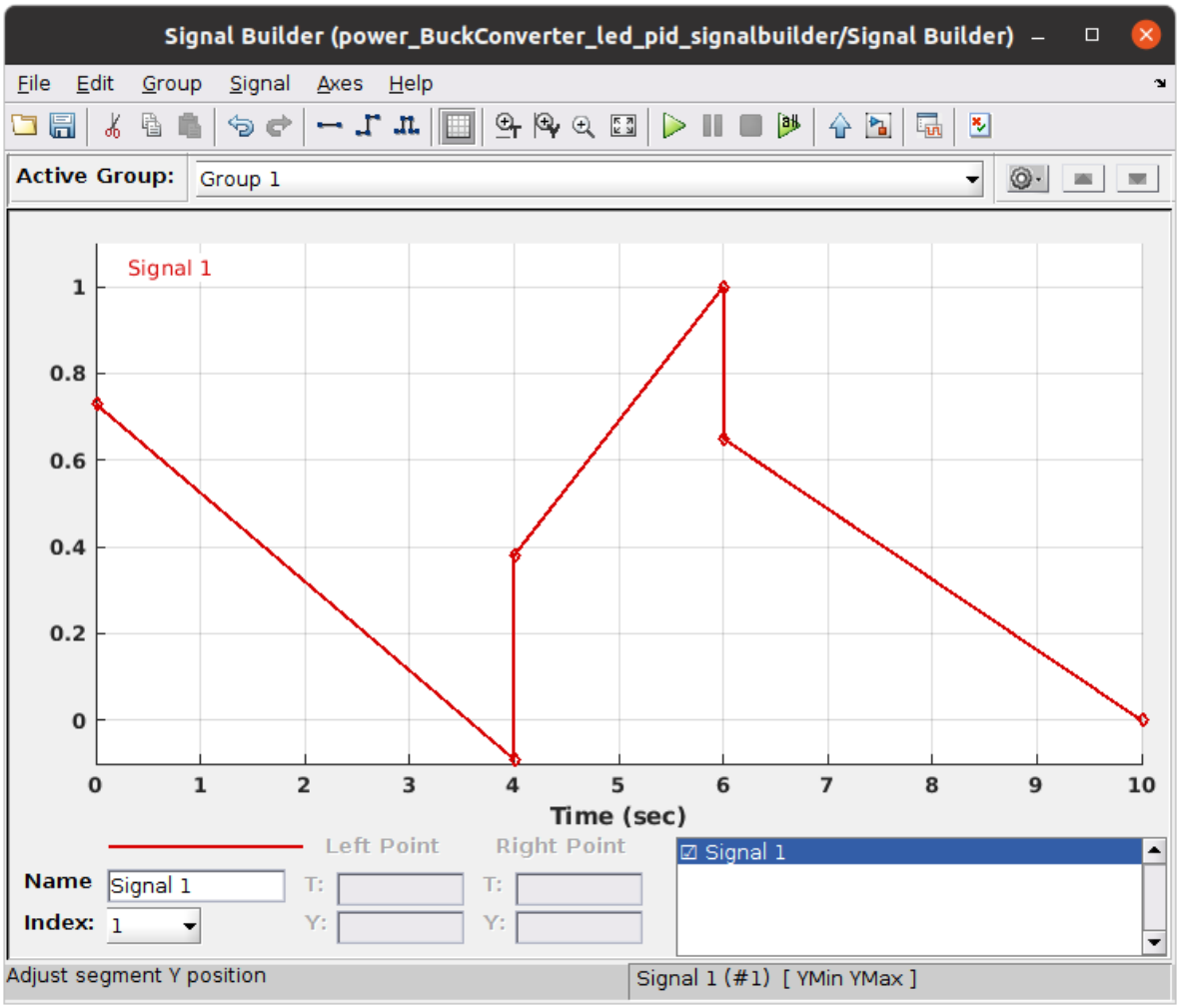
Now we can measure the performances of the nonlinear simulation, with the updated PID parameters.

Because of this, now the step response has changed:

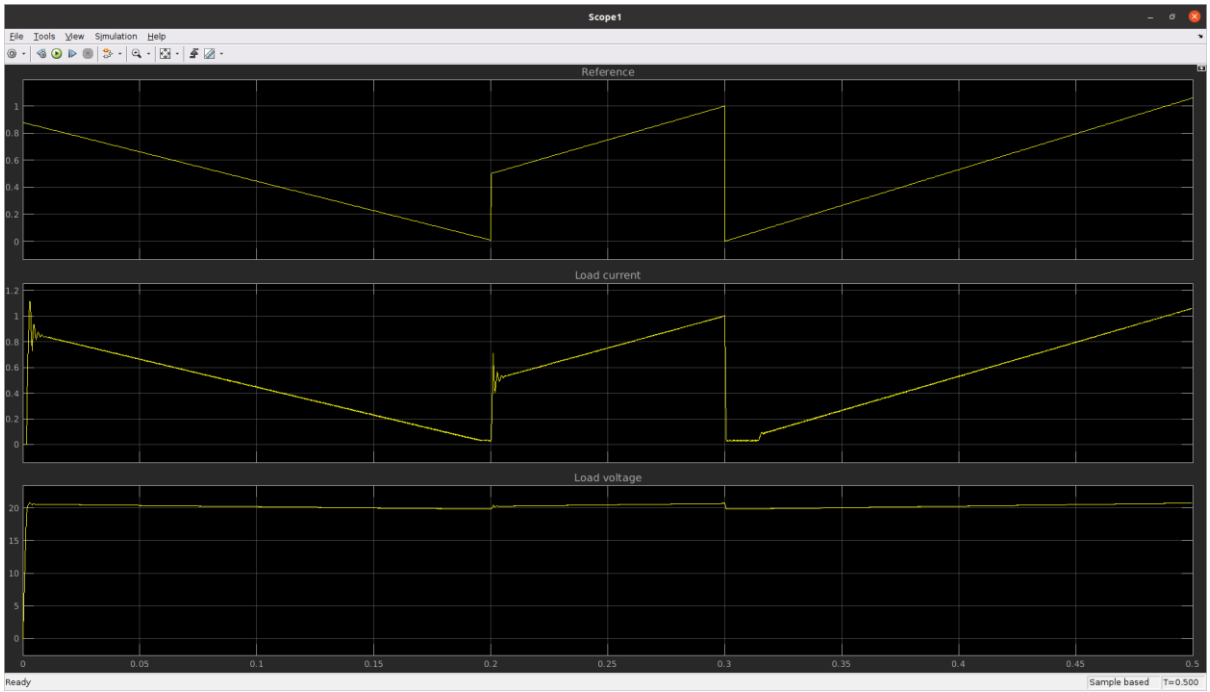


The overshoot on the current is smaller now, the PID is tuned.

By adding a signal builder block, more complicated input reference signals can be used as a reference, to see how the model behaves in such circumstances:



We can see that the current follows the reference quite well, but with some ringing in reference signal discontinuities:



FUZZY CONTROLLER

Fuzzy inference systems creation can be driven via Matlab scripts:

https://uk.mathworks.com/help/fuzzy/working-from-the-command-line.html

Or by using the fuzzy logic designer app GUI:

https://uk.mathworks.com/help/fuzzy/building-systems-with-fuzzy-logic-toolbox-software.html

Fuzzy logic control, is an advanced control strategy that mimics human reasoning to manage complex systems.

Fuzzy logic controllers (FLCs), use linguistic variables rather than precise numerical values to make decisions.

The controller evaluates inputs such as error (the difference between desired and actual output voltage) and change in error (the rate at which the error is changing).

These inputs, are “fuzzified” into categories, the FLC then applies a set of rules based on these fuzzy inputs to determine the appropriate control action.

It is possible to create models of a buck converter, and integrate fuzzy logic blocks, using the Fuzzy Logic Toolbox available in Simulink.

Advantages of Fuzzy Logic Control in Buck Converters are:

- FLCs are less sensitive to parameter variations and uncertainties compared to traditional controllers.
- They can effectively manage non-linearities present in buck converters due to their inherent design flexibility.
- Unlike PID controllers that require precise tuning of parameters, fuzzy logic systems can be adjusted intuitively through rule modifications without extensive mathematical modelling.
- FLCs can enhance transient response times and reduce overshoot during load changes or input fluctuations.
- They can easily adapt to different operating conditions without requiring complete redesigns or recalibrations.

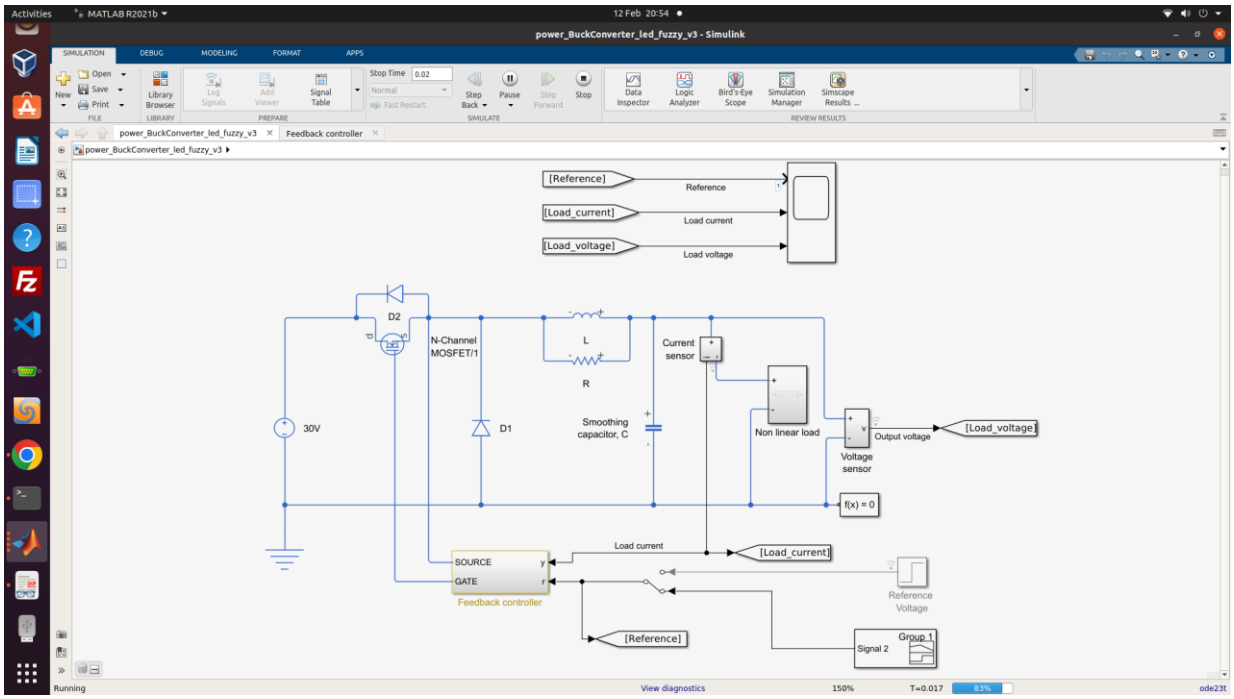
The following IEEE paper, describes in depth the practical implementation of a fuzzy logic buck converter controller:

https://www.researchgate.net/publication/342855119_DCDC_Buck_Converter_Using_Fuzzy_Logic_Controller

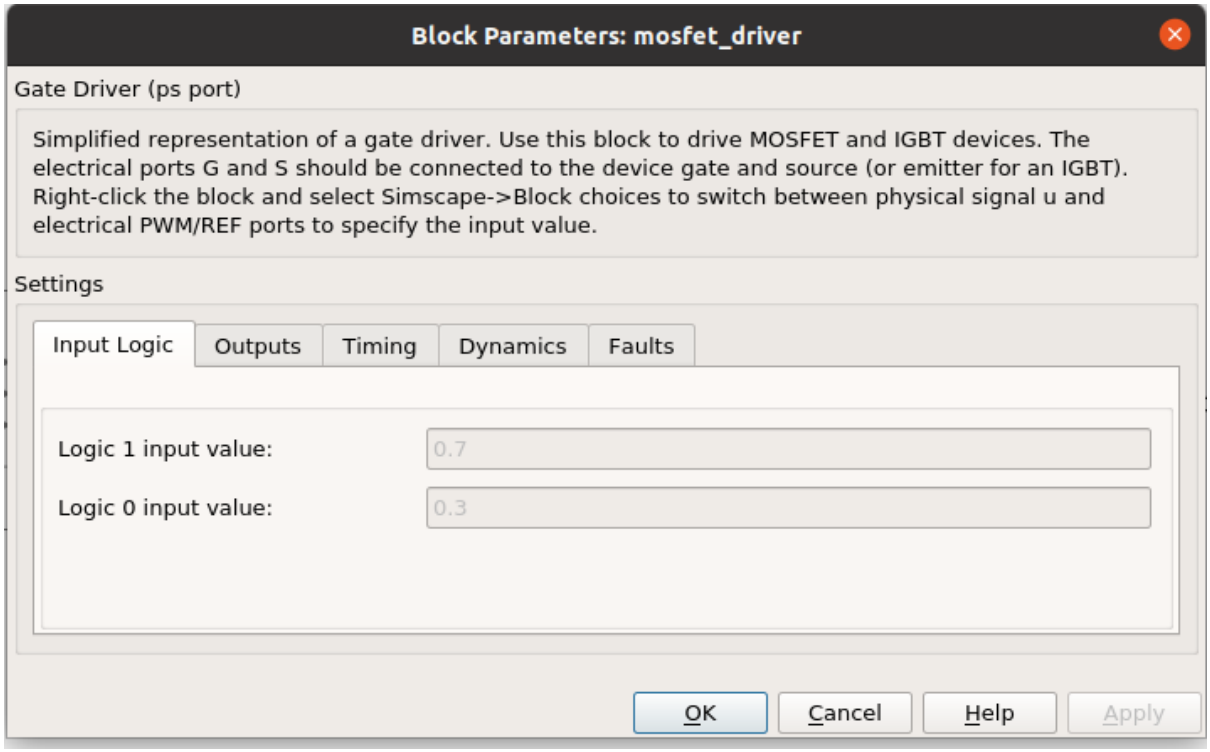
https://ieeexplore.ieee.org/document/9138084

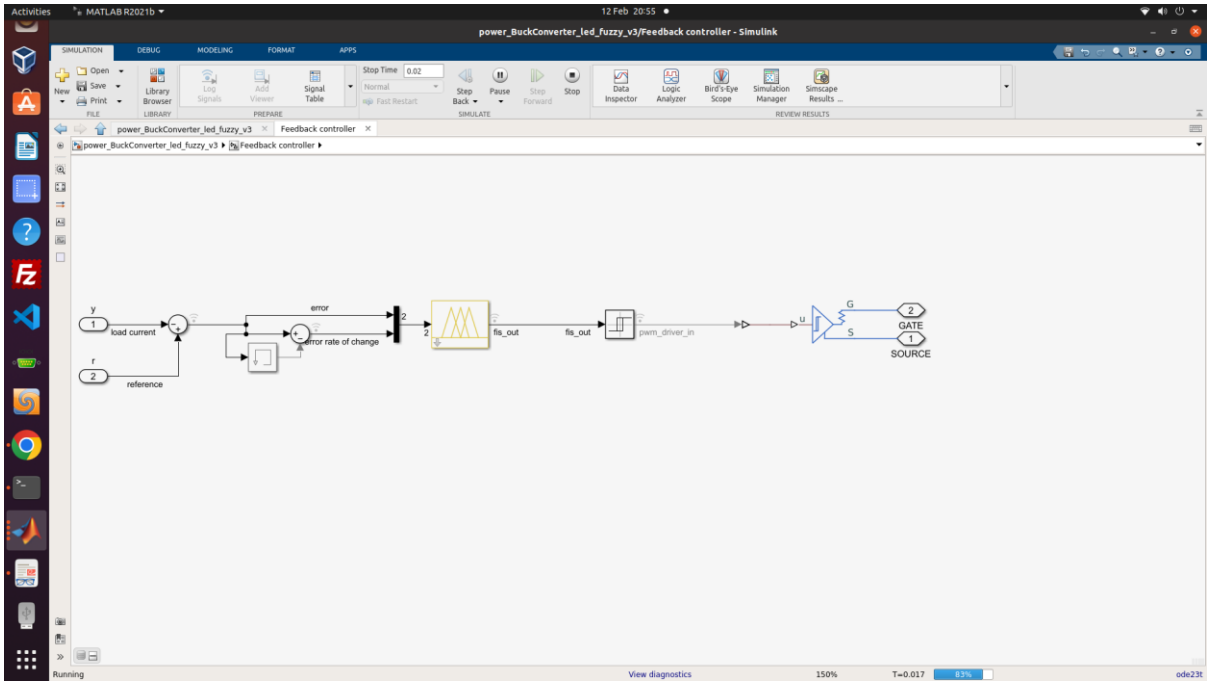
And it has been used as a source of inspiration to implement a fuzzy logic controller to replace the PID we talked about earlier.

The model has been modified to employ a fuzzy logic controller:



A mosfet driver block has been added:





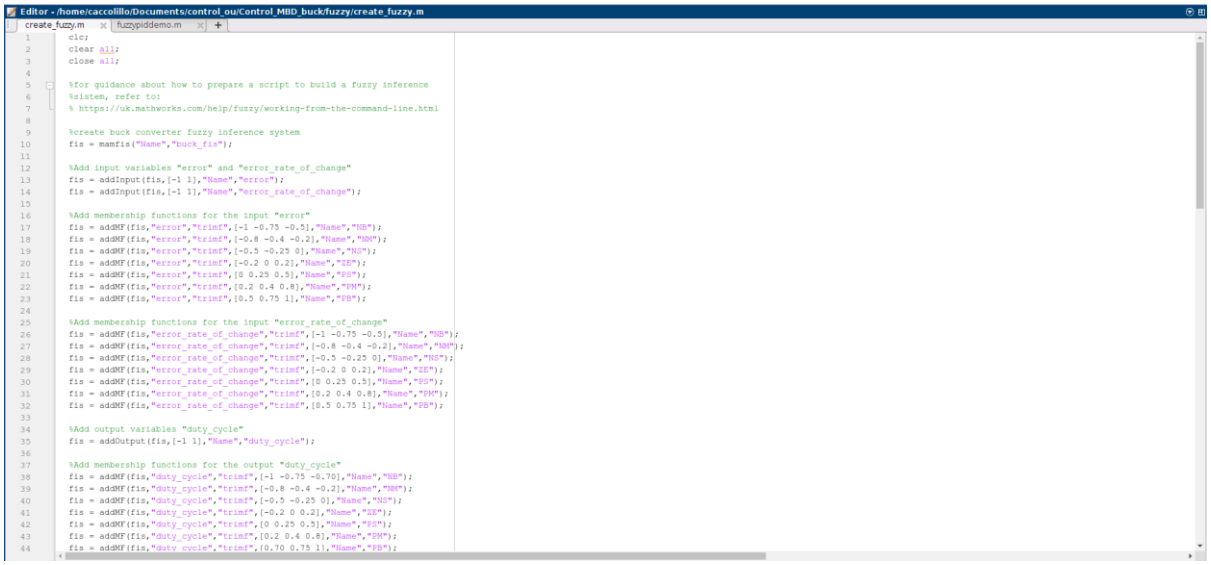
Mathwork's tool called "Fuzzy Logic Toolbox" has been used to build this model.

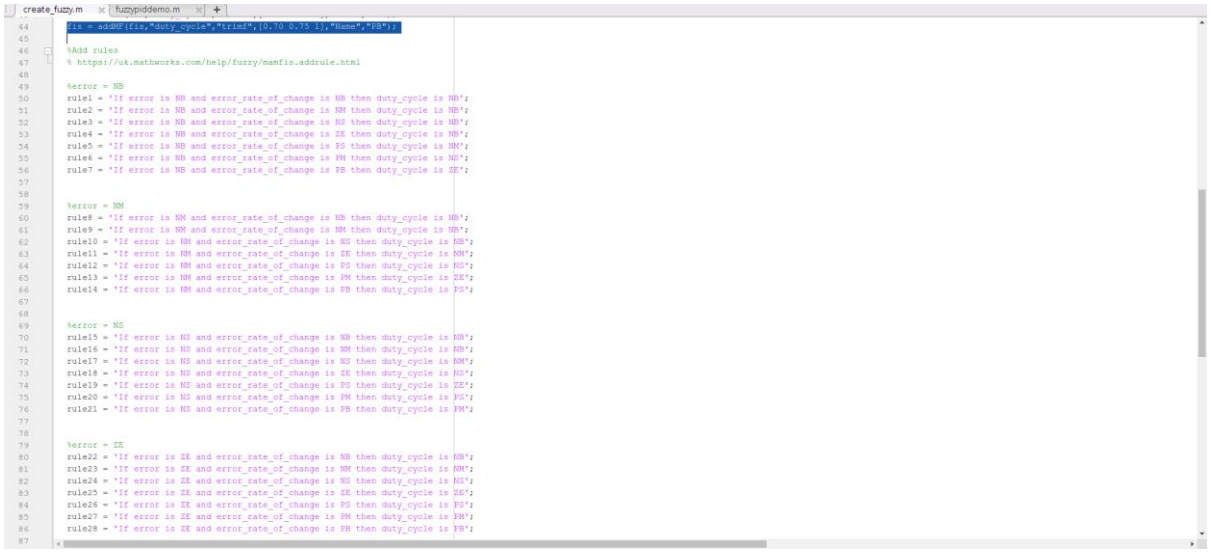
https://uk.mathworks.com/products/fuzzy-logic.html

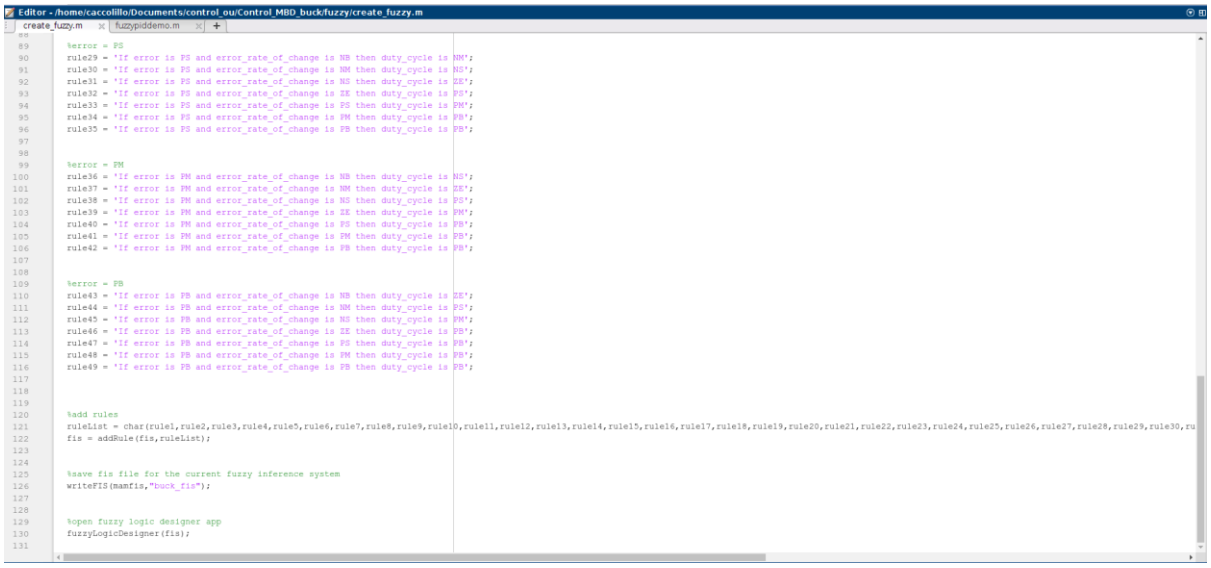
A matlab script, used as initialization function:

https://uk.mathworks.com/help/simulink/ug/initialization-function.html

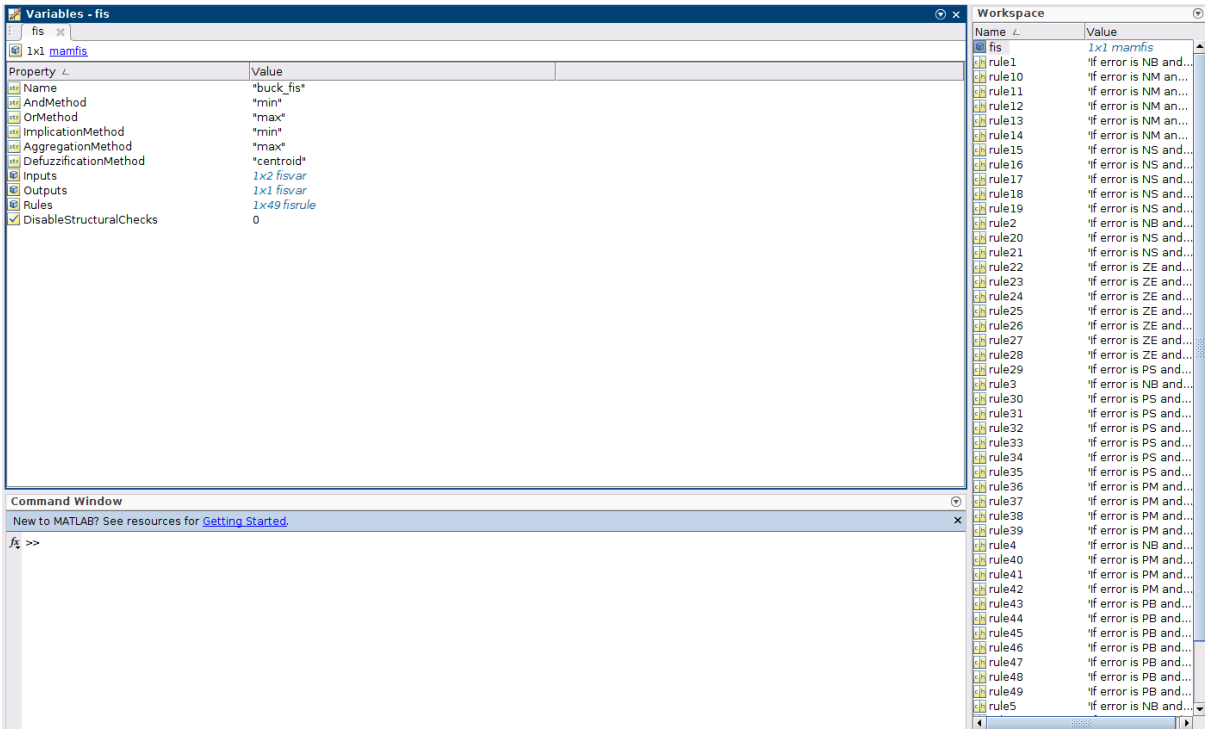
Has been prepared to create the fuzzy inference system described in the IEEE paper:



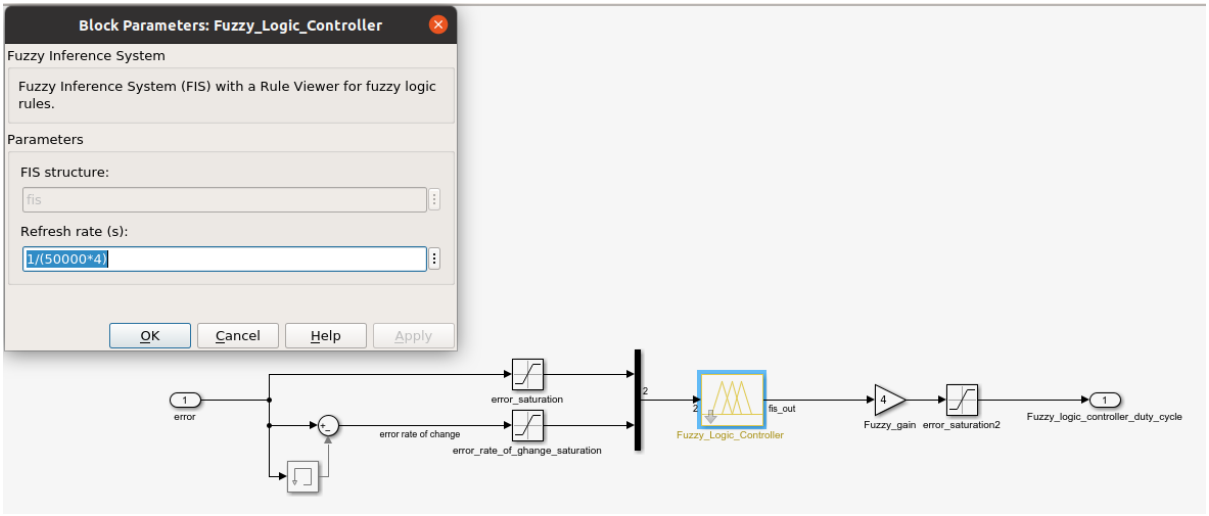




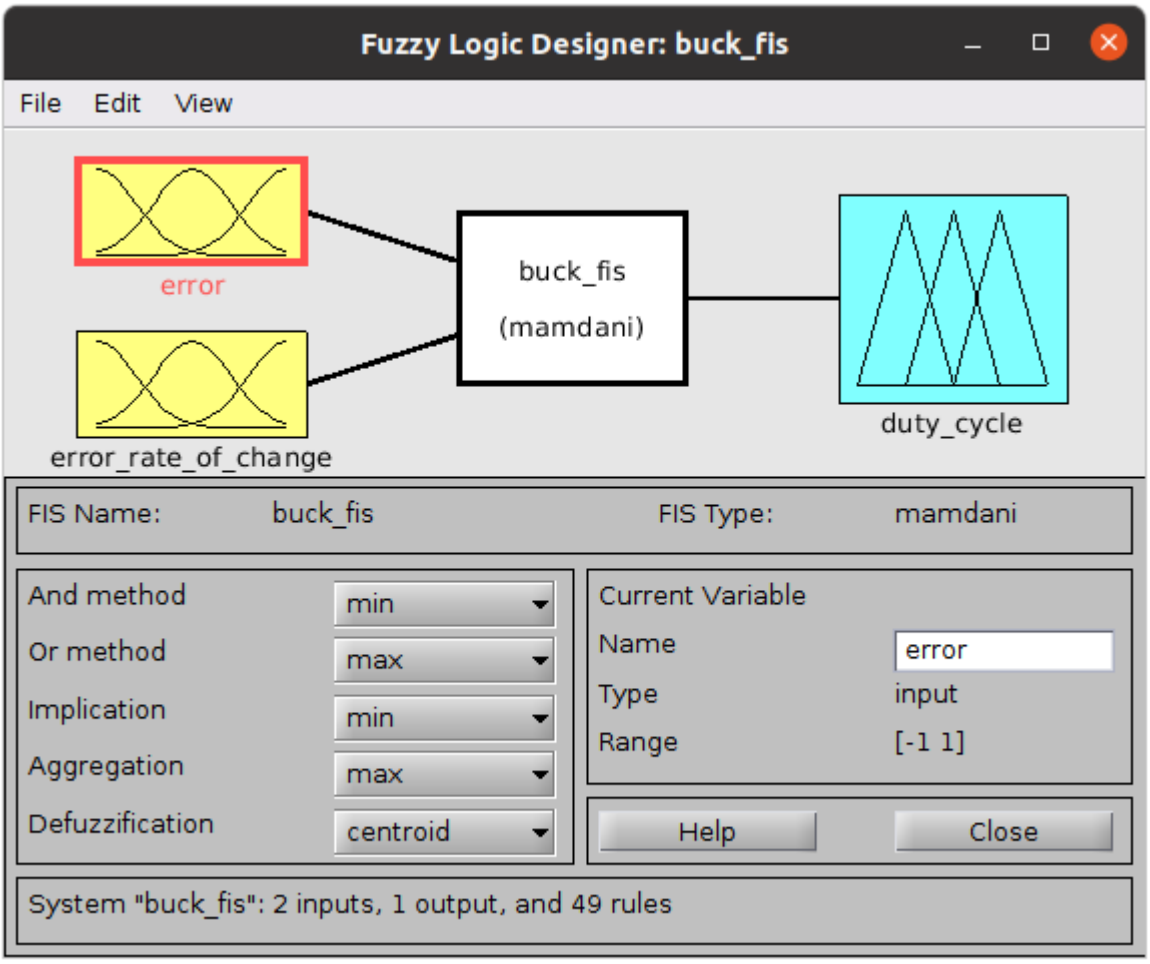
When launched it creates a “fis” object in the Matlab’s workspace:

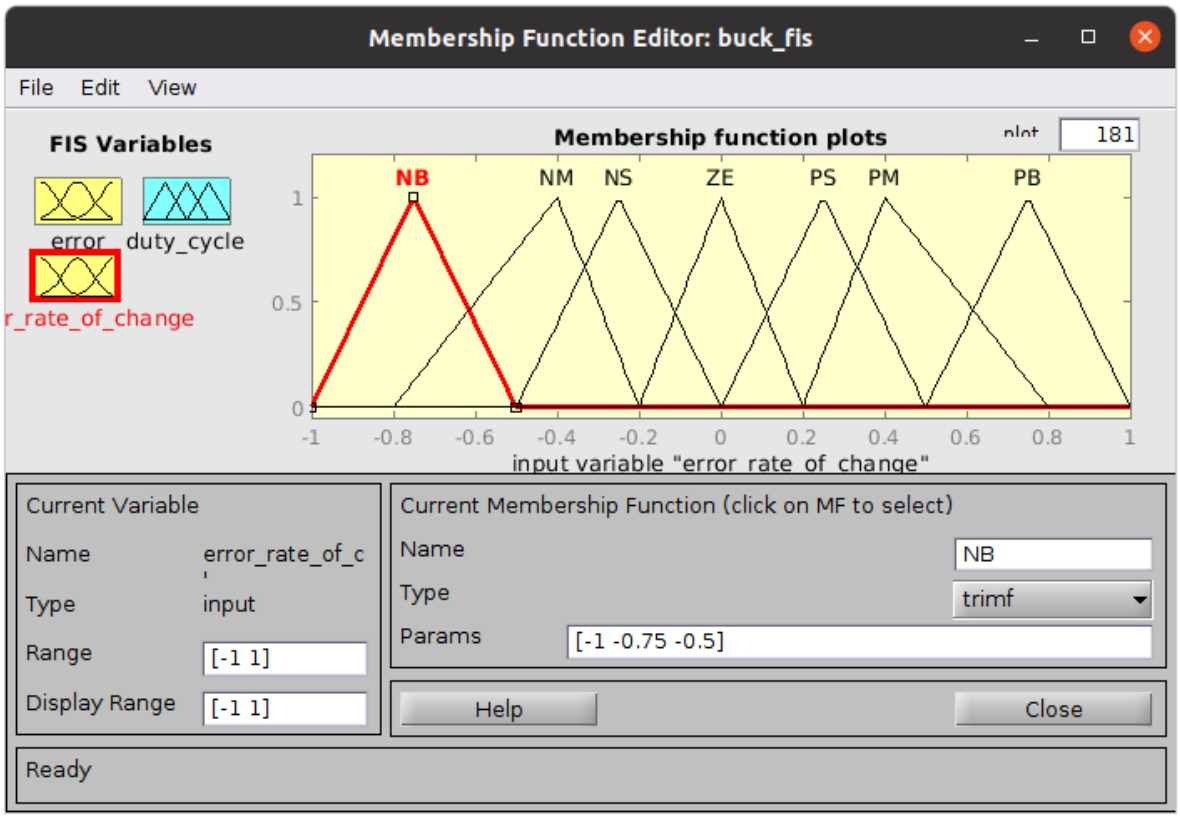


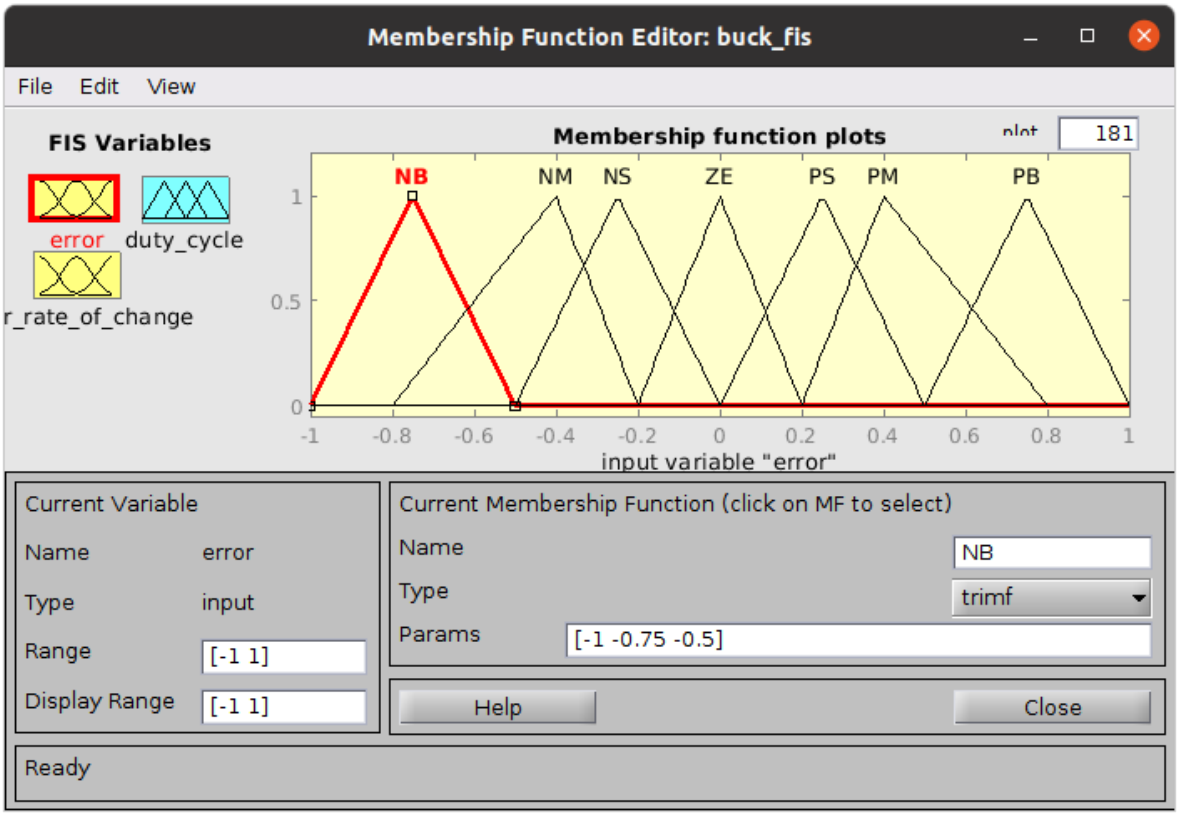
That is used in the simulink fuzzy inference system in the model:

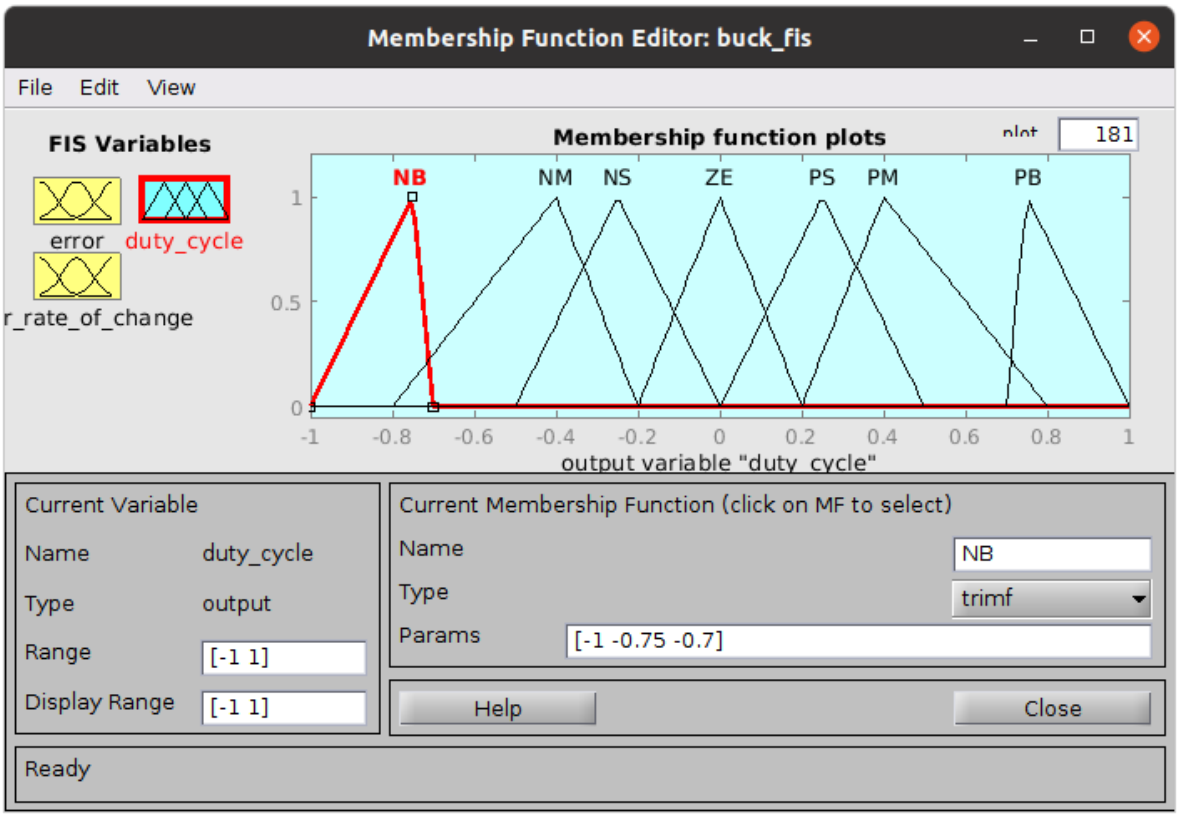


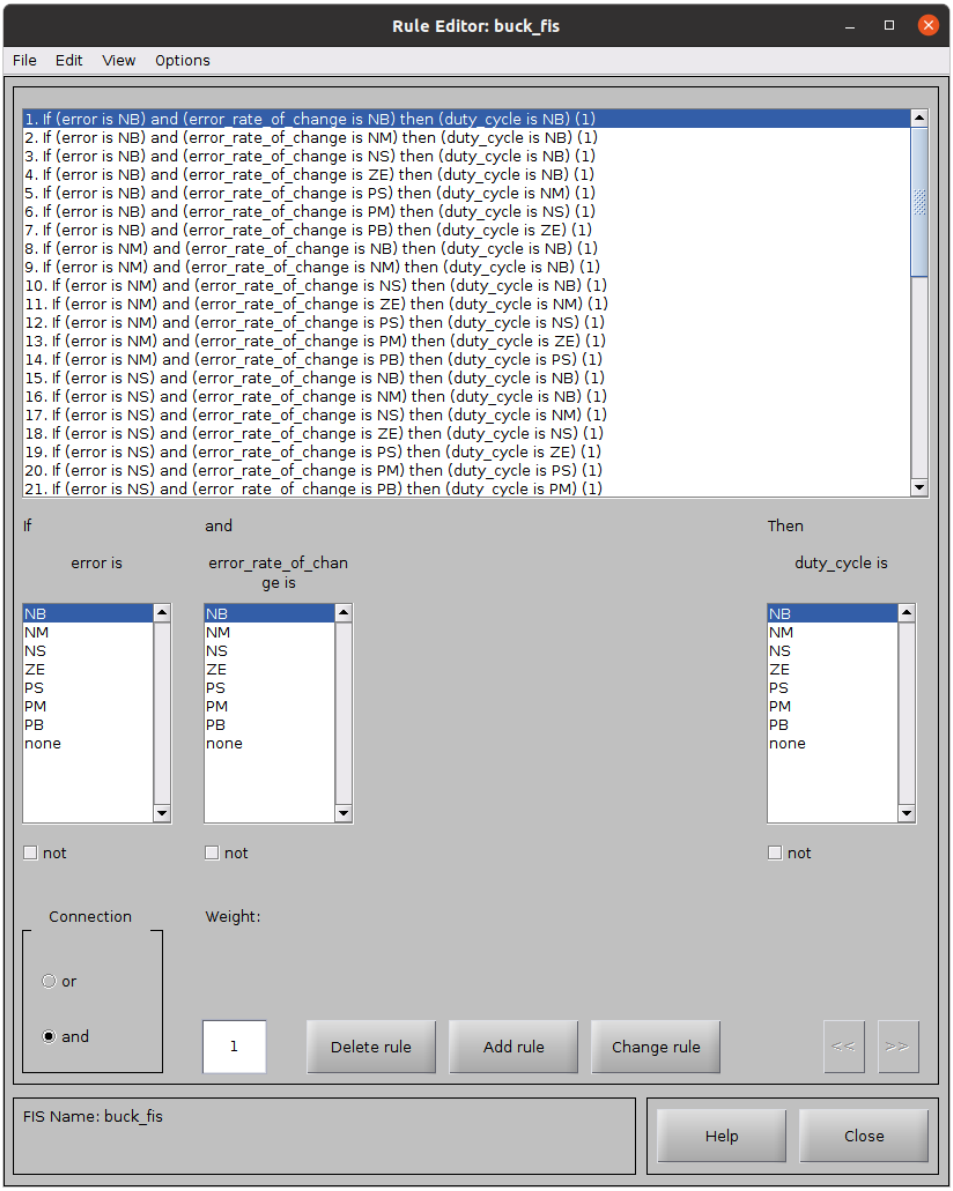
By opening the “Fuzzy Logic Designer” GUI, we can see internal details about the fuzzy controller described by the script, in an intuitive pictorial fashion:


Fuzzy Logic Designer: buck_fis
File Edit View
error
error_rate_of_change
buck_fis
(mamdani)
duty_cycle
FIS Name: buck_fis
FIS Type: mamdani
And method min
Or method max
Implication min
Aggregation max
Defuzzification centroid
Current Variable
Name error
Type input
Range [-1 1]
Help
Close
System "buck_fis": 2 inputs, 1 output, and 49 rules

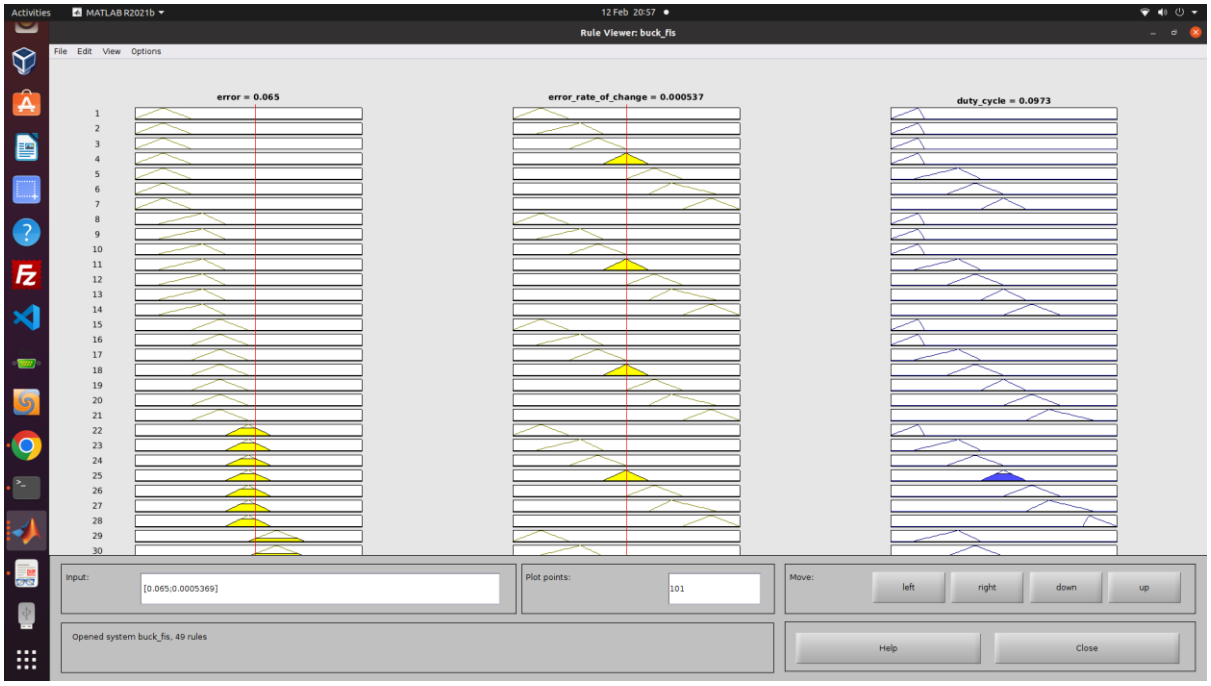

Membership Function Editor: buck_fis
File Edit View
FIS Variables
error
duty_cycle
r_rate_of_change
Membership function plots
plot 181
NB NM NS ZE PS PM PB
1
0.5
0
-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1
input variable "error_rate_of_change"
Current Variable
Name error_rate_of_c
Type input
Range [-1 1]
Display Range [-1 1]
Current Membership Function (click on MF to select)
Name NB
Type trimf
Params [-1 -0.75 -0.5]
Help
Close
Ready

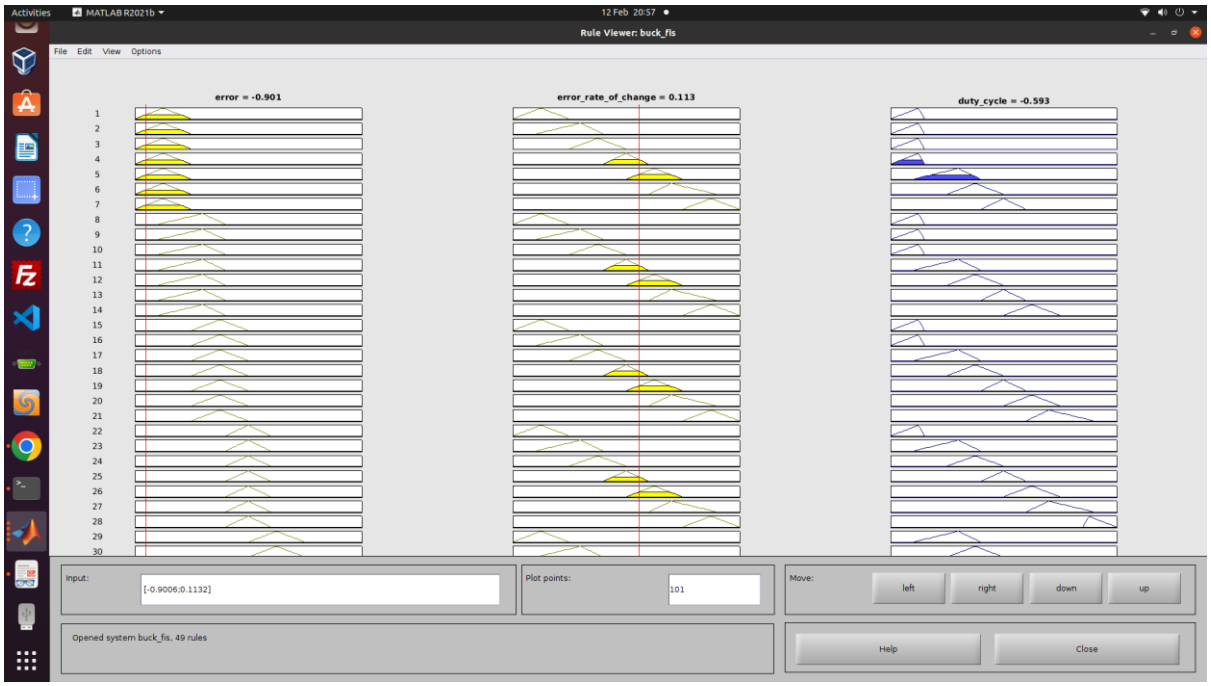

Membership Function Editor: buck_fis
File Edit View
FIS Variables
error
duty_cycle
r_rate_of_change
Membership function plots
plot
181
NB NM NS ZE PS PM PB
input variable "error"
Current Variable
Name error
Type input
Range [-1 1]
Display Range [-1 1]
Current Membership Function (click on MF to select)
Name NB
Type trimf
Params [-1 -0.75 -0.5]
Help
Close
Ready

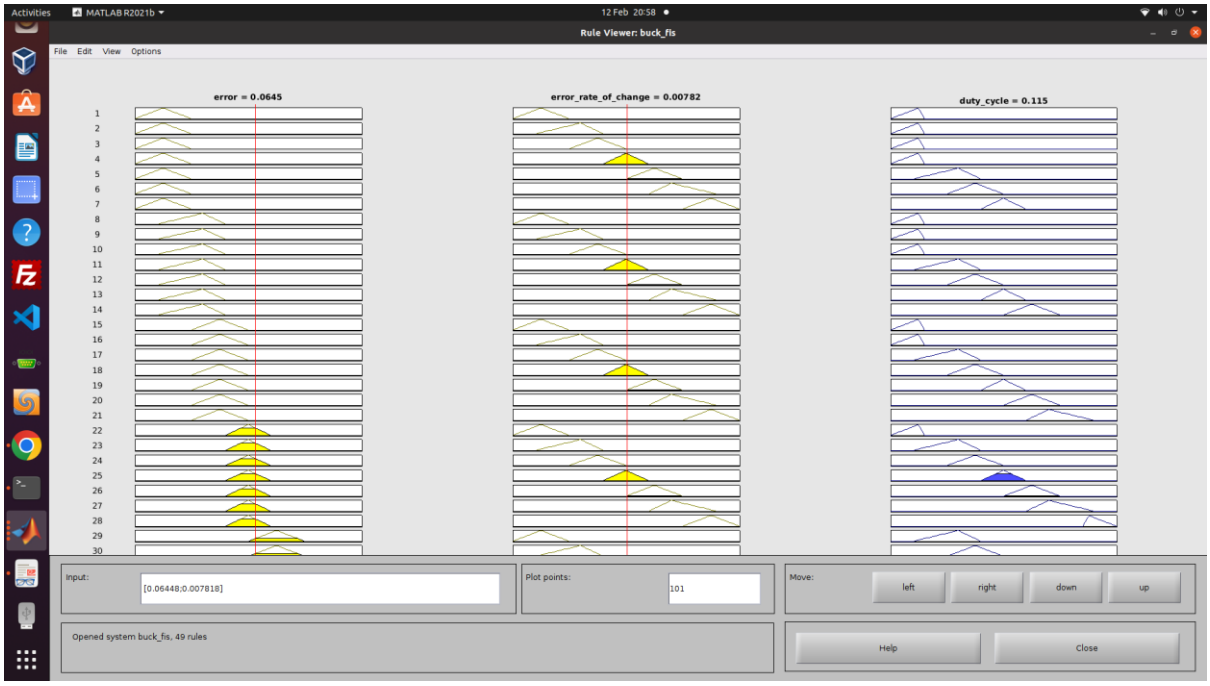

Membership Function Editor: buck_fis
File Edit View
FIS Variables
error
duty_cycle
r_rate_of_change
Membership function plots
plot
181
NB NM NS ZE PS PM PB
output variable "duty_cycle"
Current Variable
Name duty_cycle
Type output
Range [-1 1]
Display Range [-1 1]
Current Membership Function (click on MF to select)
Name NB
Type trimf
Params [-1 -0.75 -0.7]
Help
Close
Ready



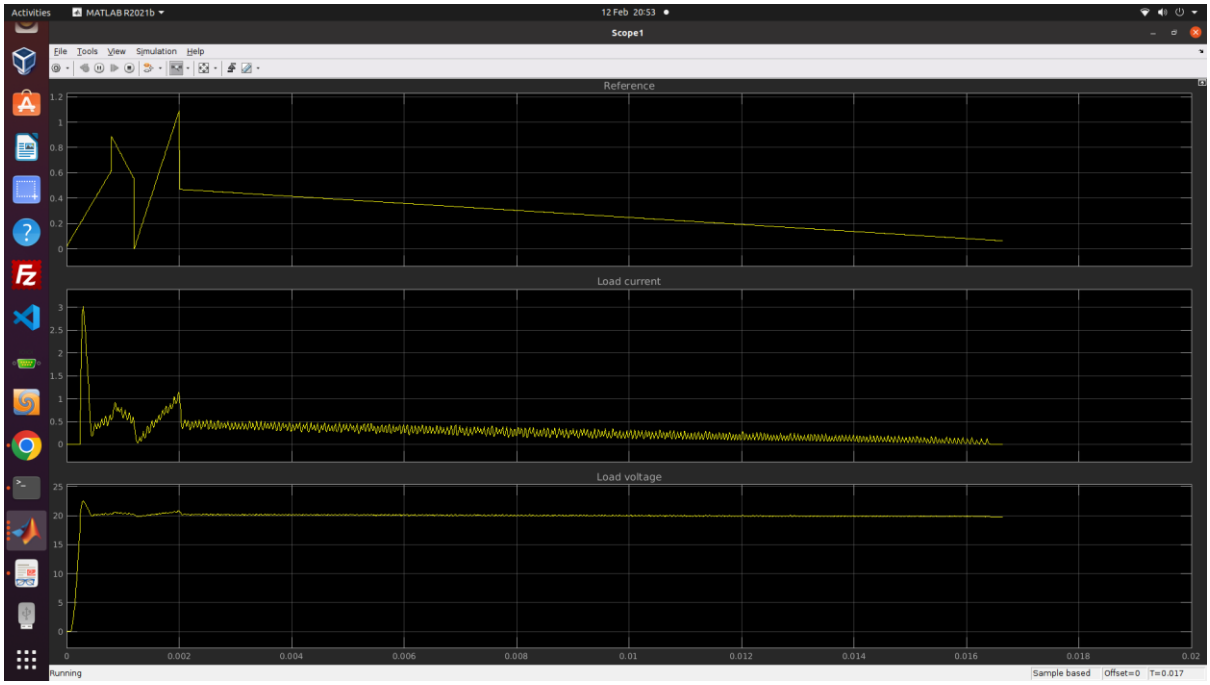
When the model runs, a window pops up showing in real time details about the “fuzzification” of the input variables and the “defuzzification” of the output:


Activities
MATLAB R2021b
12 Feb 20:57
Rule Viewer: buck_fis
File Edit View Options
error = 0.065
error_rate_of_change = 0.000537
duty_cycle = 0.0973
Input:
[0.065;0.0005369]
Plot points:
101
Move:
left
right
down
up
Opened system buck_fis, 49 rules
Help
Close


Activities
MATLAB R2021b
12 Feb 20:57
Rule Viewer: buck_fis
File Edit View Options
error = -0.901
error_rate_of_change = 0.113
duty_cycle = -0.593
Input:
[-0.9006;0.1132]
Plot points:
101
Move:
left
right
down
up
Opened system buck_fis, 49 rules
Help
Close

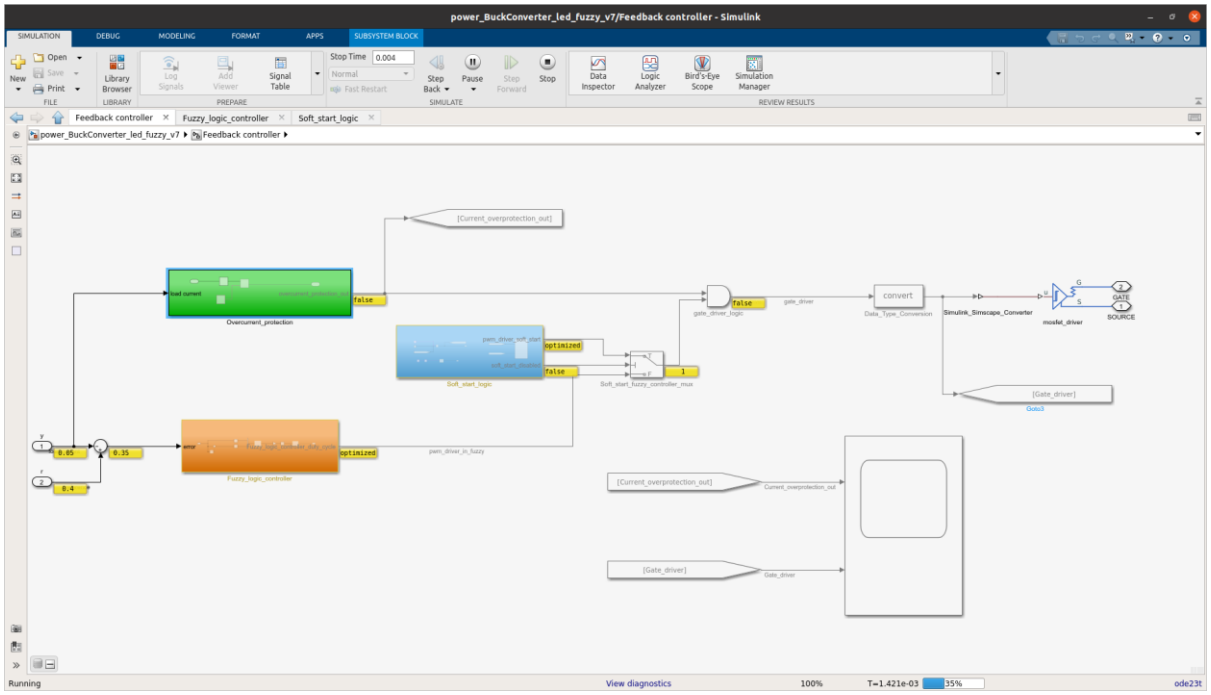


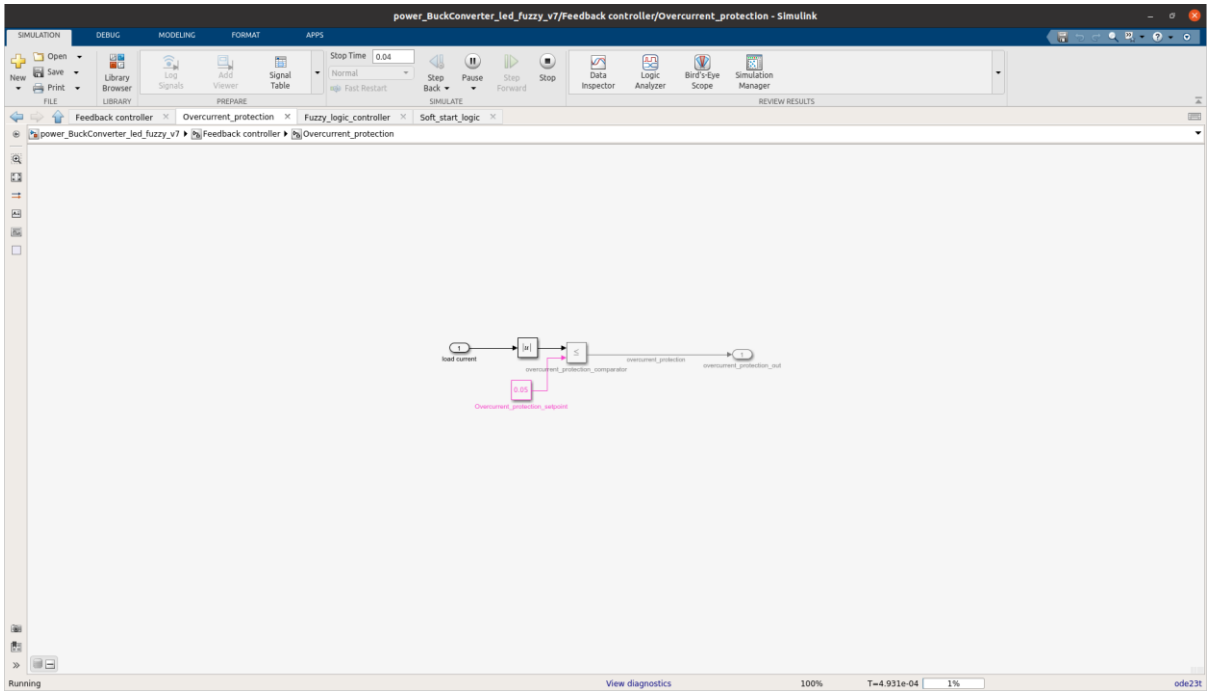
Simulating the model and looking at the waveforms, we can see that the LED current follows the reference input signal, with the drawback of having a huge overshoot at the converter start up:

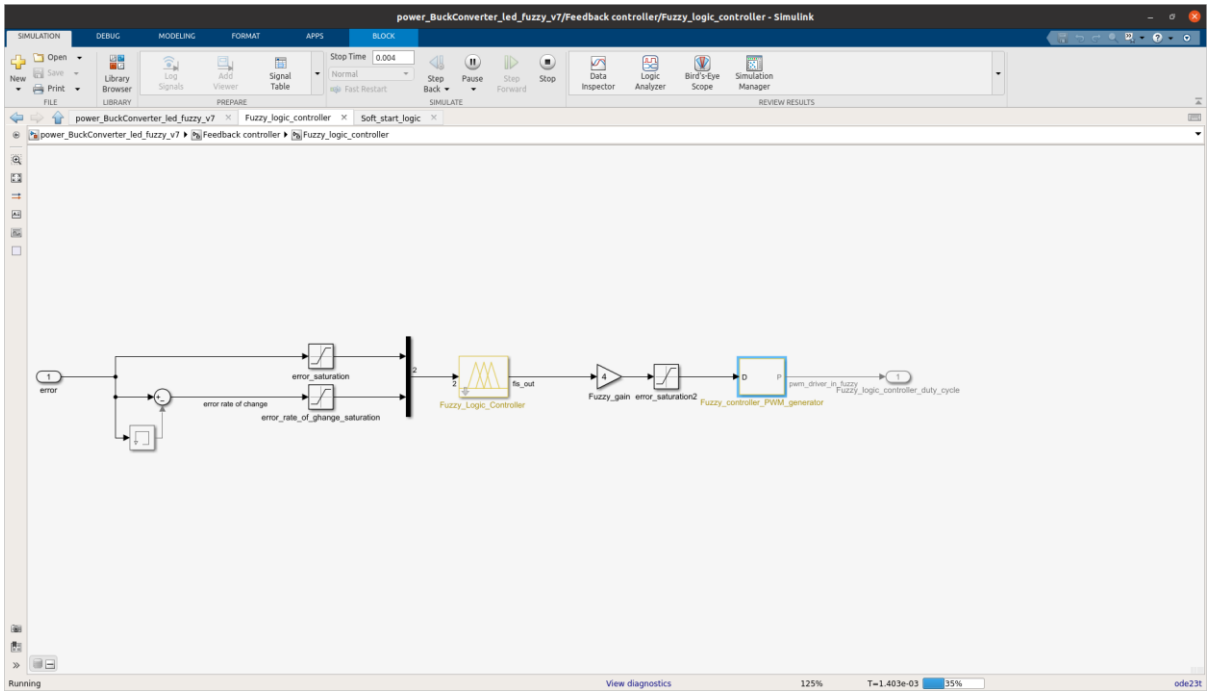


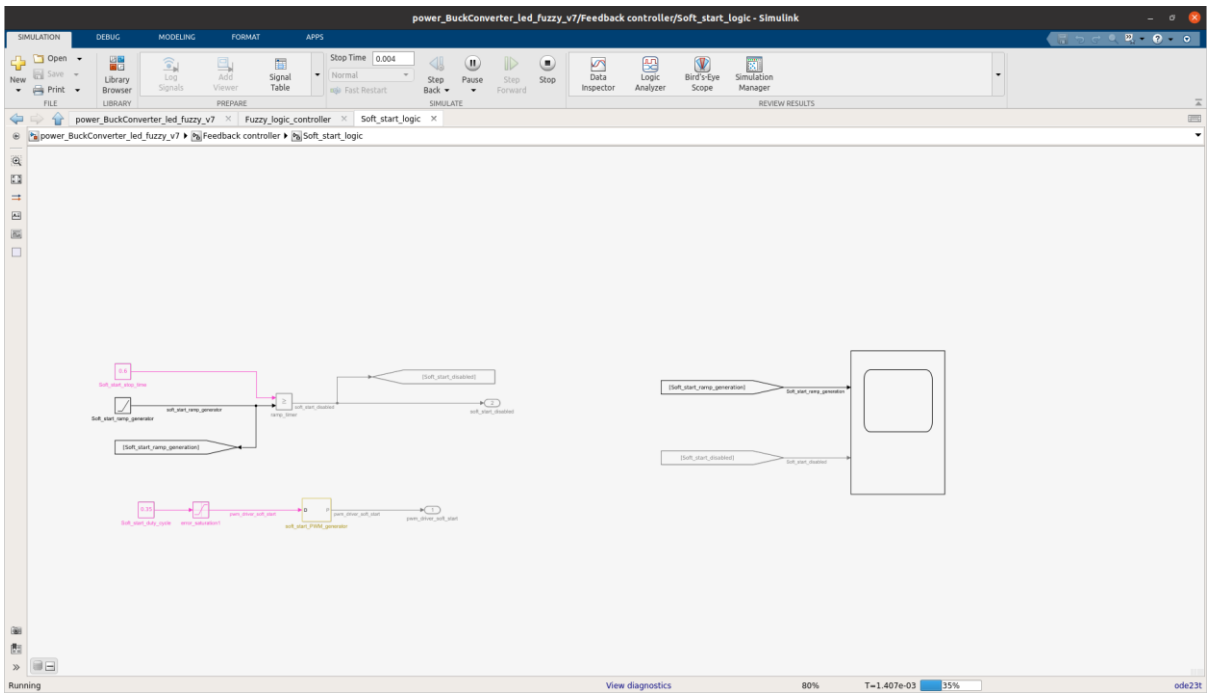
This has led to the design choice of introducing a soft start mechanism and an overcurrent detection protection circuitry.

Hence, the controller's model has been changed slightly, as shown below:

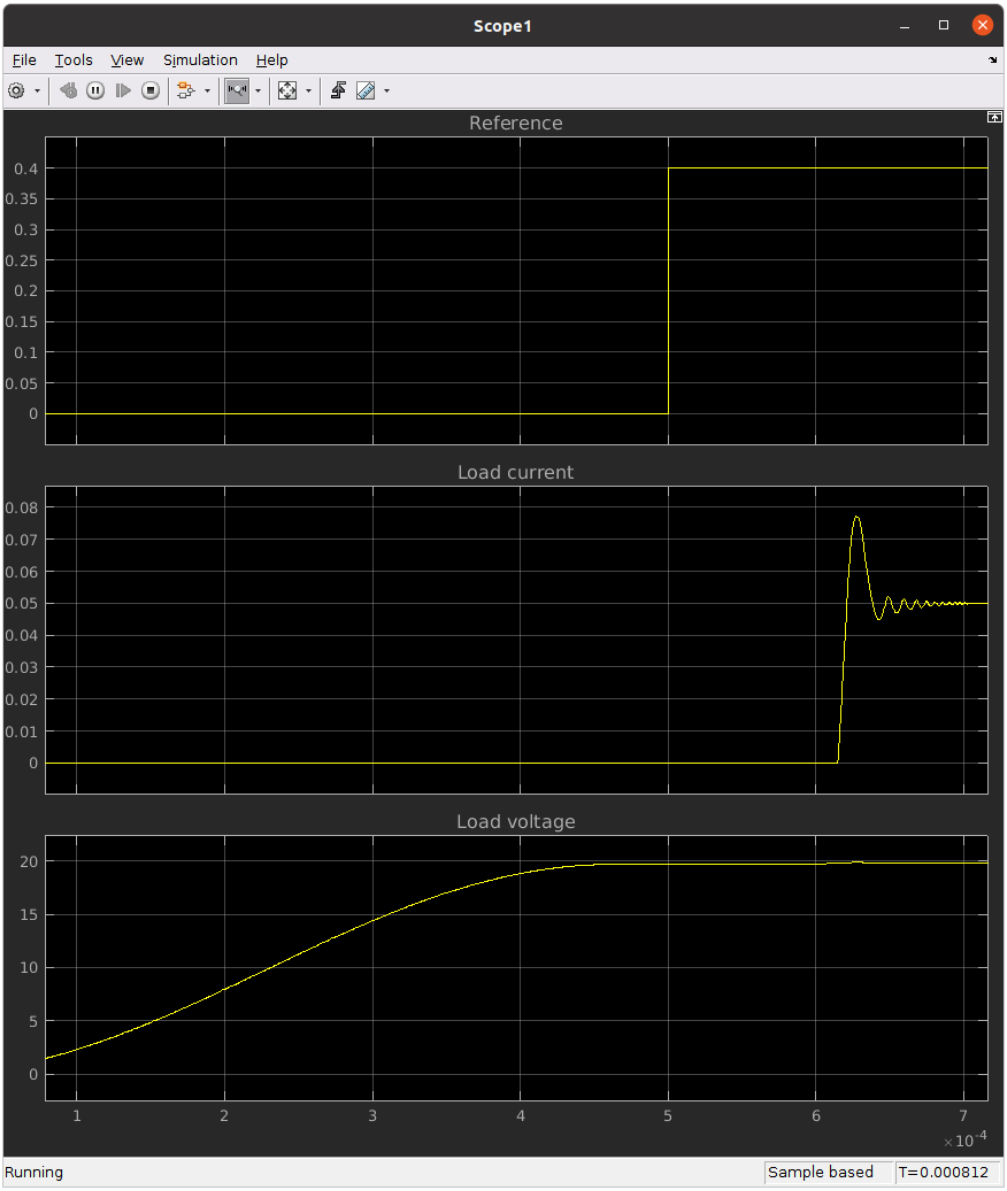

power_BuckConverter_led_fuzzy_v7/Feedback controller - Simulink
SIMULATION
DEBUG
MODELING
FORMAT
APPS
SUBSYSTEM BLOCK
Stop Time
0.004
Feedback controller
Fuzzy_logic_controller
Soft_start_logic
power_BuckConverter_led_fuzzy_v7
Feedback controller
[Current_overprotection_out]
load current
Overcurrent_protection
false
gate_driver_logic
gate_driver
convert
Data_Type_Conversion
Simulink_Simscape_Converter
mosfet_driver
GATE
SOURCE
[Gate_driver]
Goto3
pwm_driver_soft_start
soft_start_disabled
optimized
false
Soft_start_logic
Soft_start_fuzzy_controller_mux
1
0.05
0.35
error
Fuzzy_logic_controller
pwm_driver_in_fuzzy
0.4
Current_overprotection_out
Gate_driver
Running
View diagnostics
100%
T=1.421e-03
35%
ode23t


power_BuckConverter_led_fuzzy_v7/Feedback controller/Overcurrent_protection - Simulink
SIMULATION
DEBUG
MODELING
FORMAT
APPS
Stop Time
0.04
Feedback controller
Overcurrent_protection
Fuzzy_logic_controller
Soft_start_logic
power_BuckConverter_led_fuzzy_v7
Feedback controller
Overcurrent_protection
load current
|u|
overcurrent_protection_comparator
overcurrent_protection
overcurrent_protection_out
0.05
Overcurrent_protection_setpoint
Running
View diagnostics
100%
T=4.931e-04
1%
ode23t

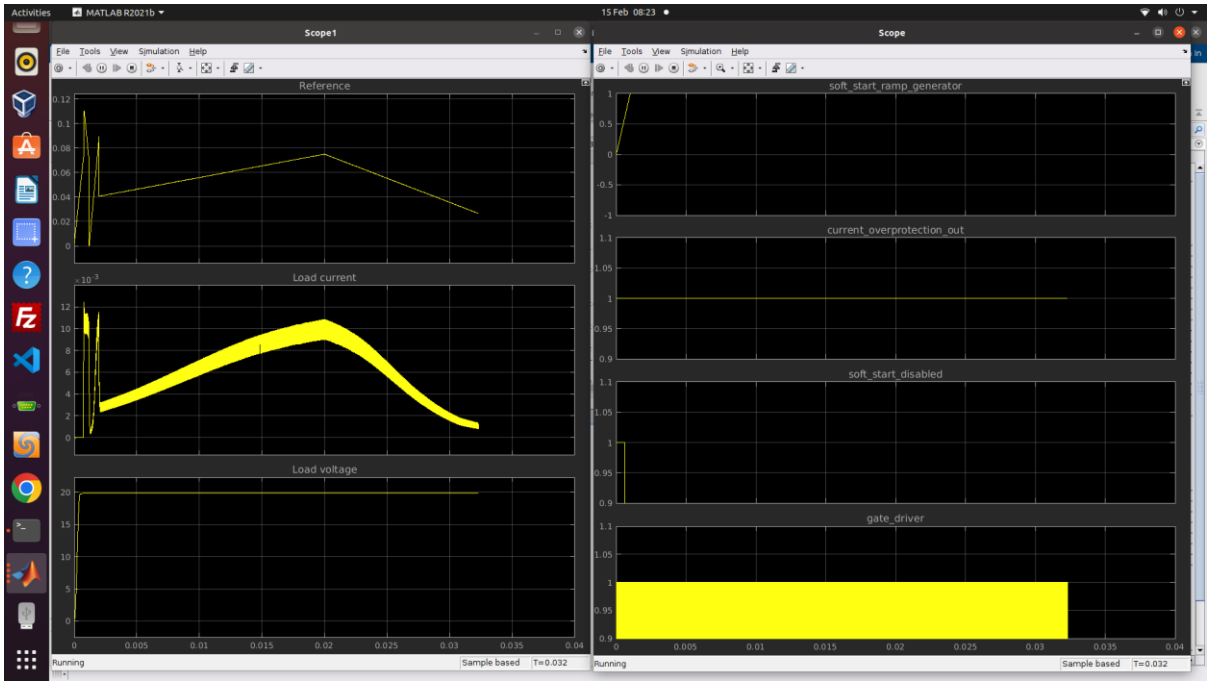




With these modifications, the step response shows a reduction in the overshoot:



By applying an arbitrary input reference signal, we can see that the load current follows it, but with some superimposed ripple:


Activities
MATLAB R2021b
15 Feb 08:23
Scope1
File Tools View Simulation Help
Reference
Load current
Load voltage
Running
Sample based
T=0.032
Scope
File Tools View Simulation Help
soft_start_ramp_generator
current_overprotection_out
soft_start_disabled
gate_driver
Running
Sample based
T=0.032