OPAL-RT Training course

OP4510

Real-Time Simulation Fundamentals with RT-LAB

Dongyu Li

TU Delft

Agenda

1. Getting started
2. Complete workflow

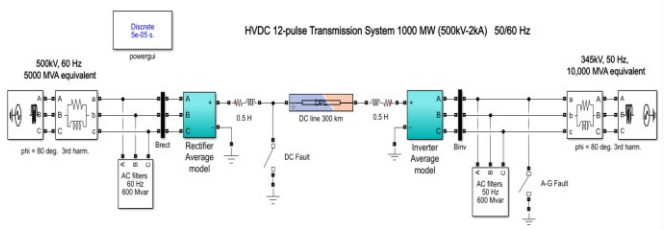
Agenda

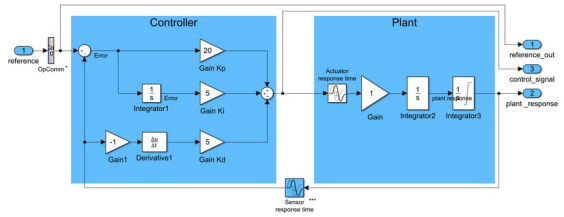
1. Getting started
   - Fundamental concepts
   - Target configuration
   - First example

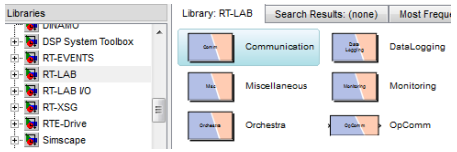
2. Complete workflow

Getting started – Fundamental concepts

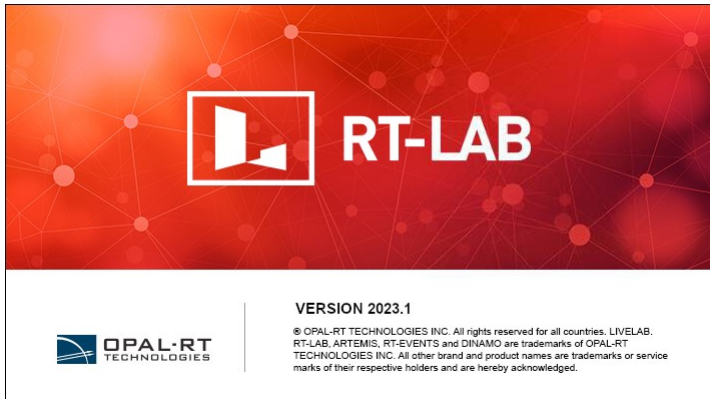
Simulink model + OPAL-RT libraries



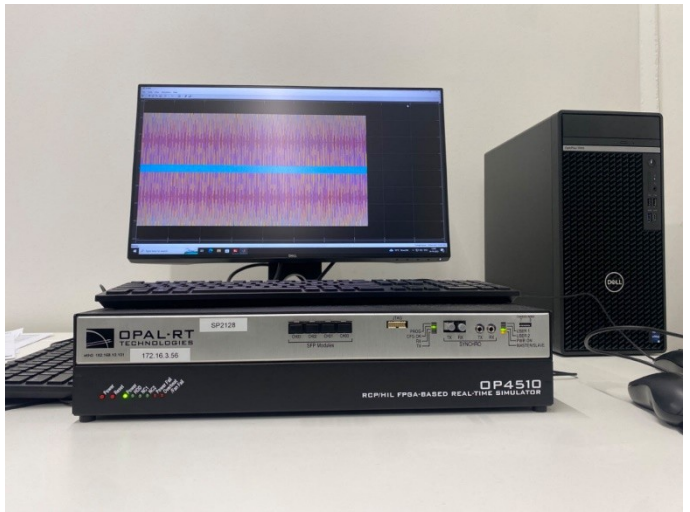




OPAL-RT's software RT-LAB

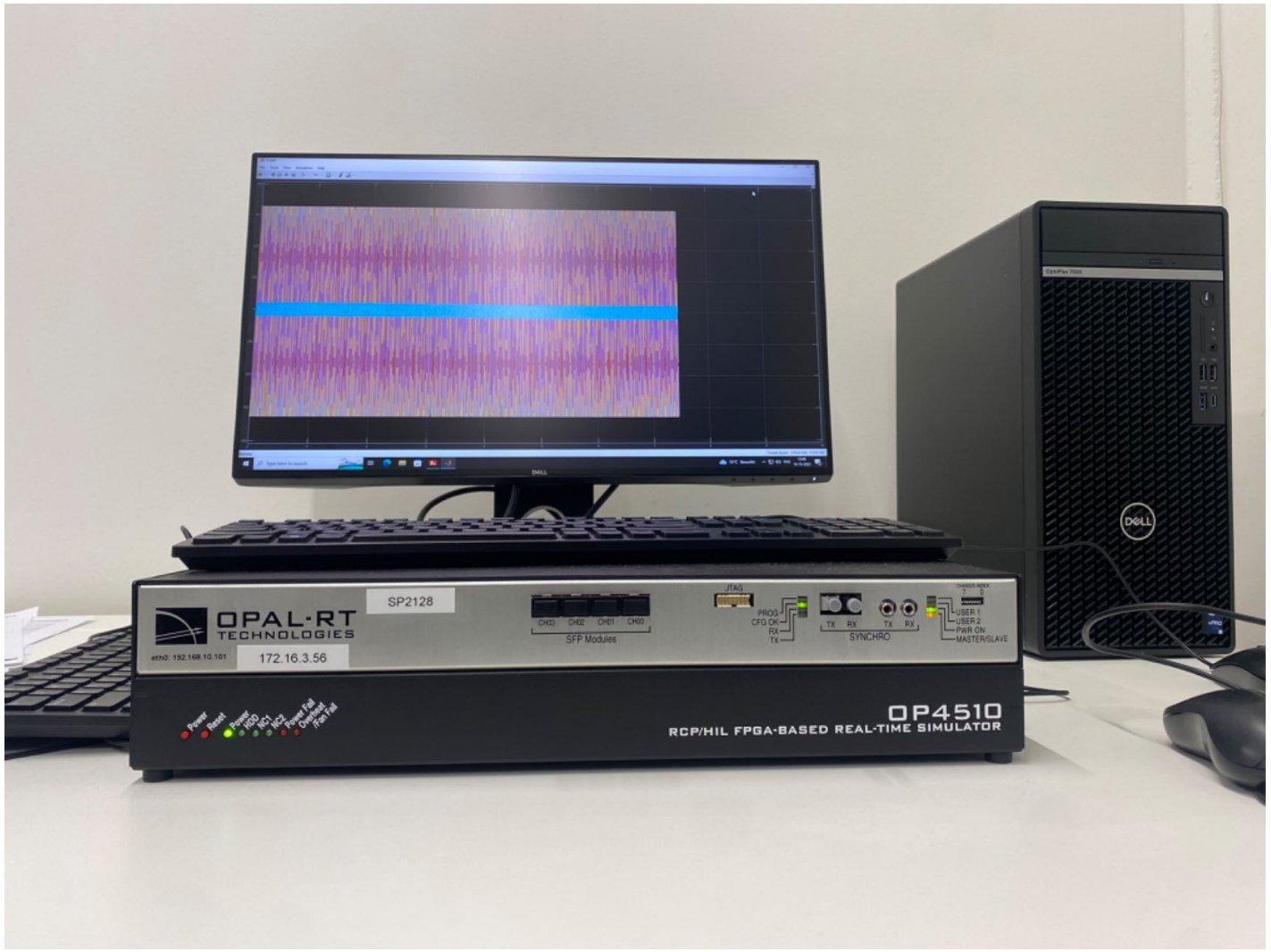


Real-time or accelerated simulation



Getting started – Fundamental concepts

TCP/IP



Host PC

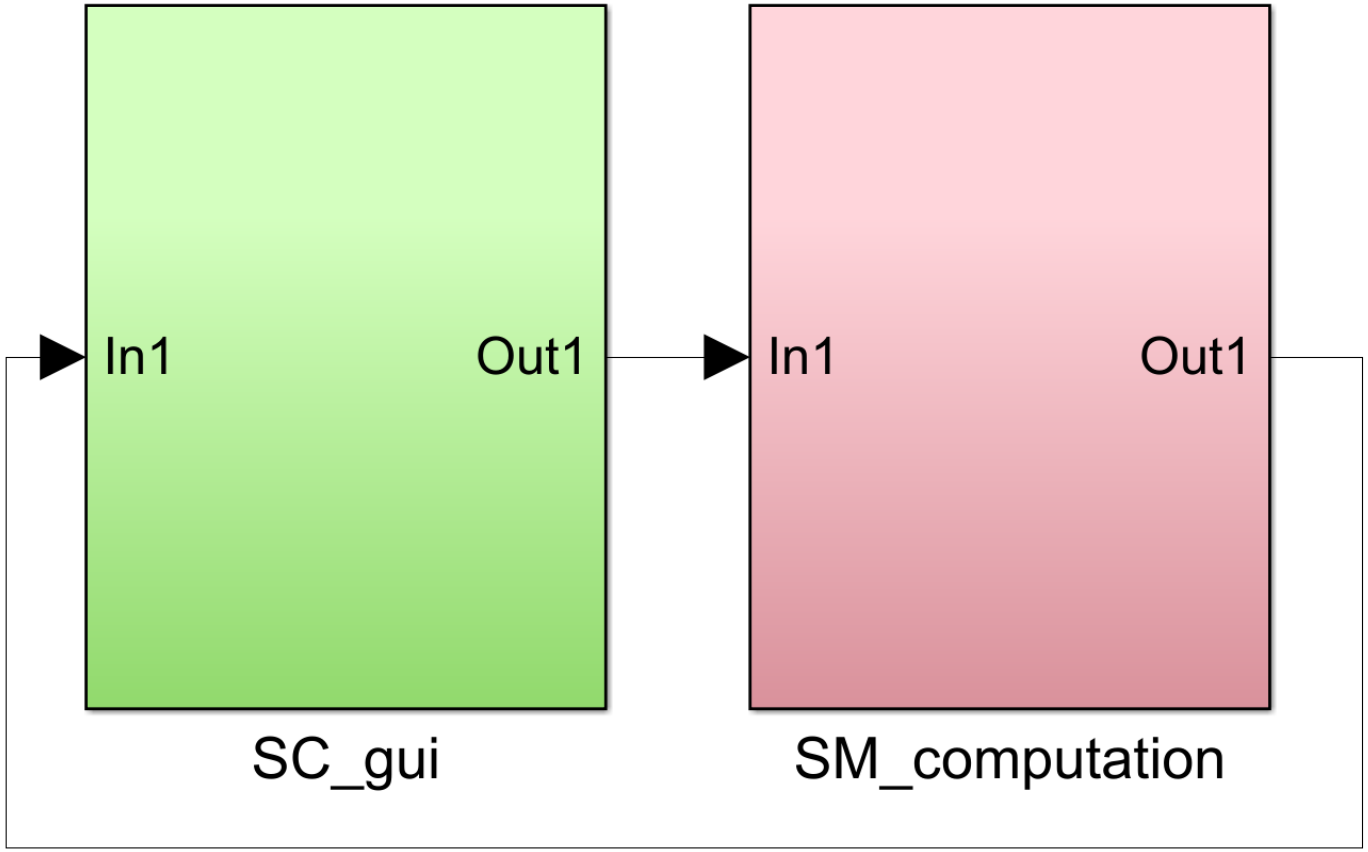
1. Design of model
2. Simulation management
3. Graphical interface

Real-time simulator

1. Model execution
2. Data logging
3. I/O management

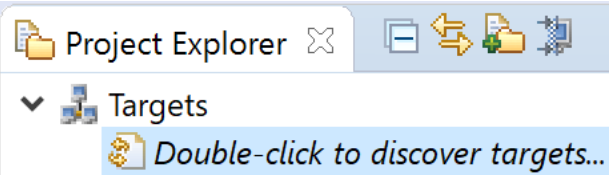
TU Delft

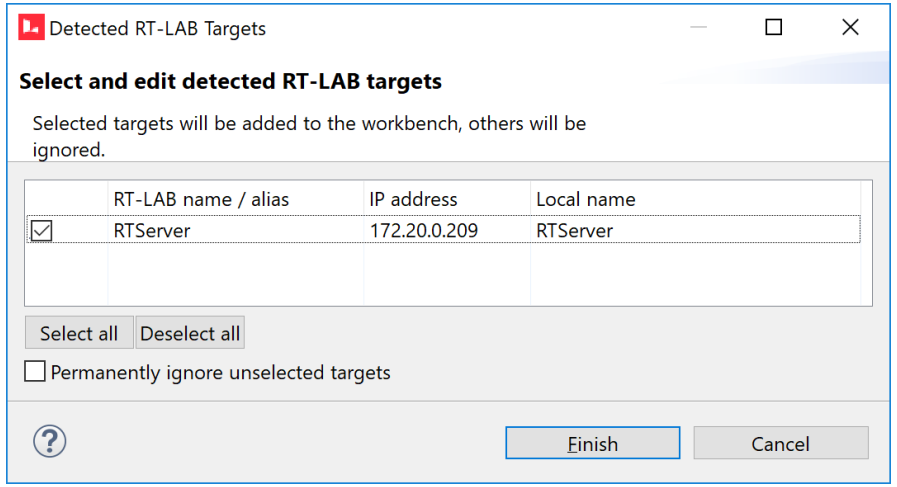
Getting started – Fundamental concepts

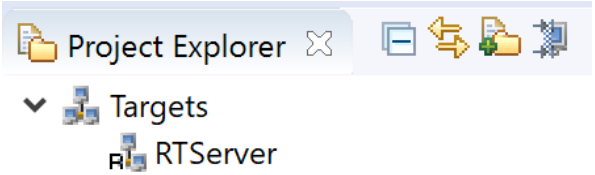


Getting started – Target configuration

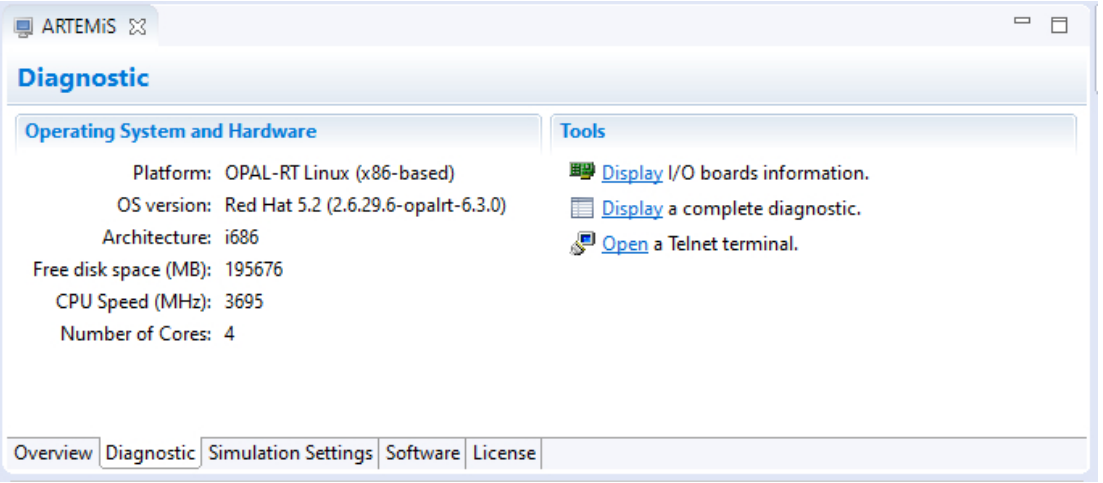
Host PC and target need to be connected through Ethernet with compatible IP addresses





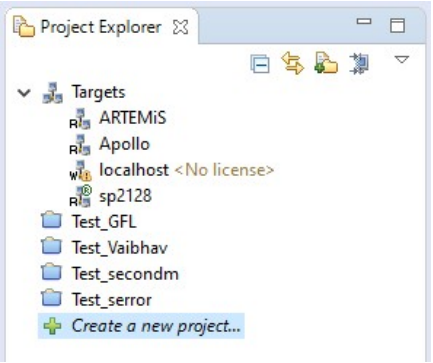


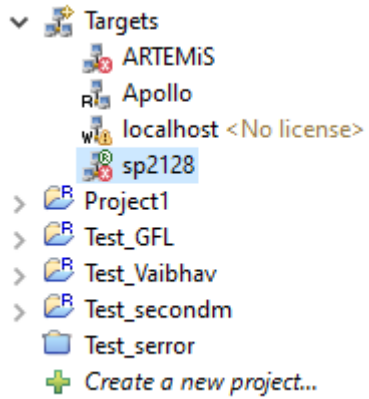
Target information

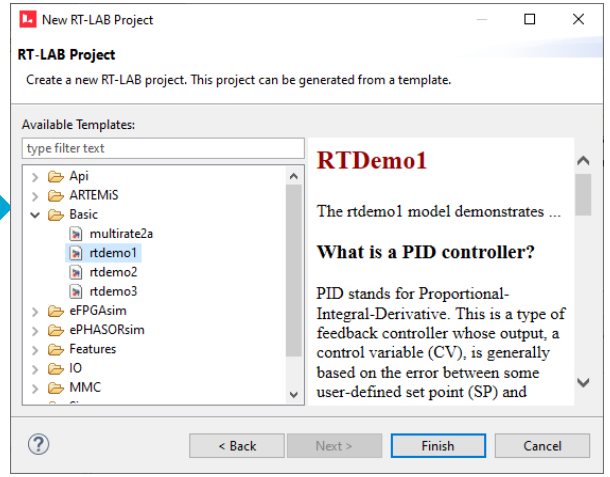


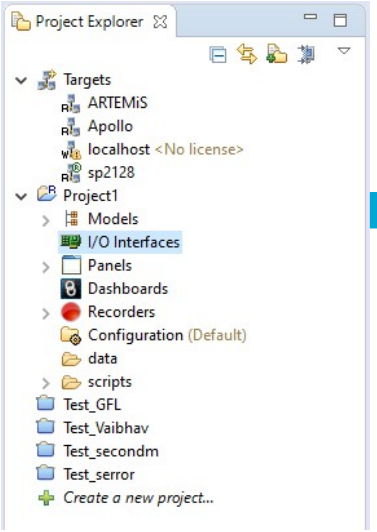
Getting started – First example

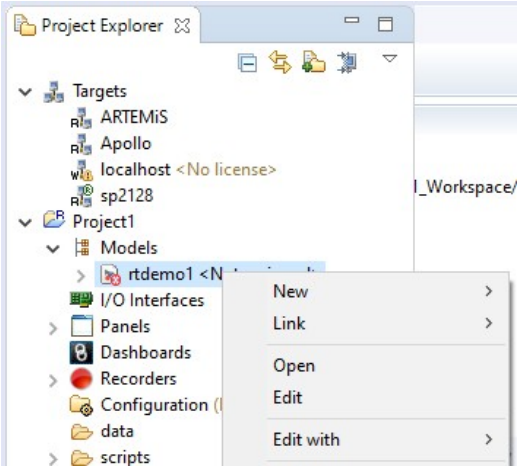
Launch RT-LAB and **create a new project** with an example model (Basic > rtdemo1), **edit** the model with Simulink and explore it

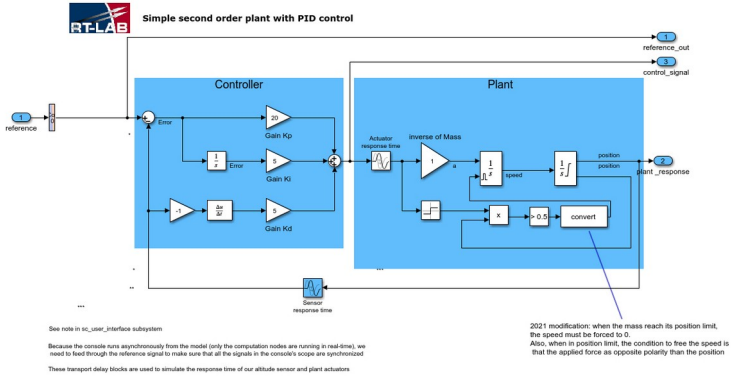


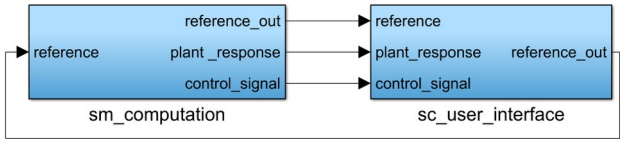








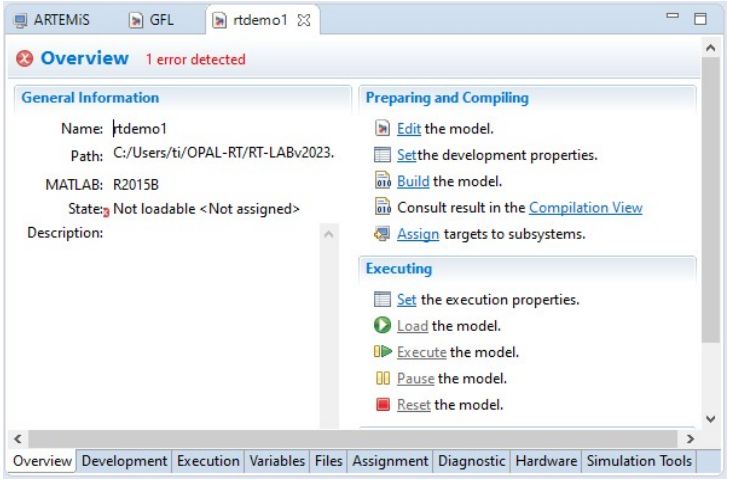


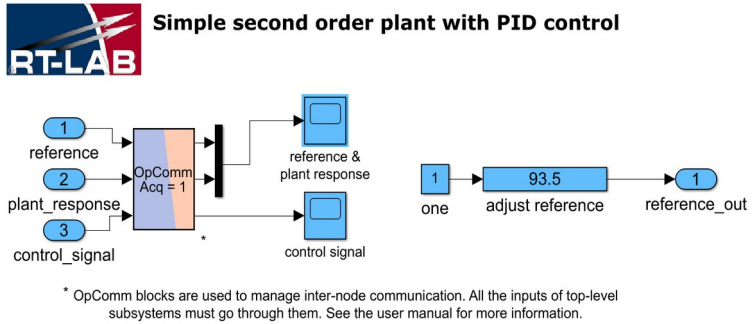


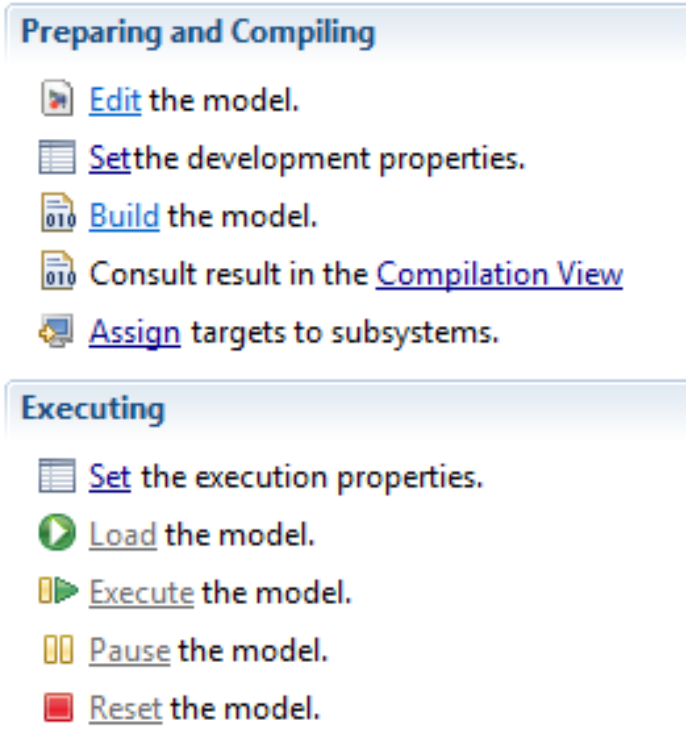
TU Delft

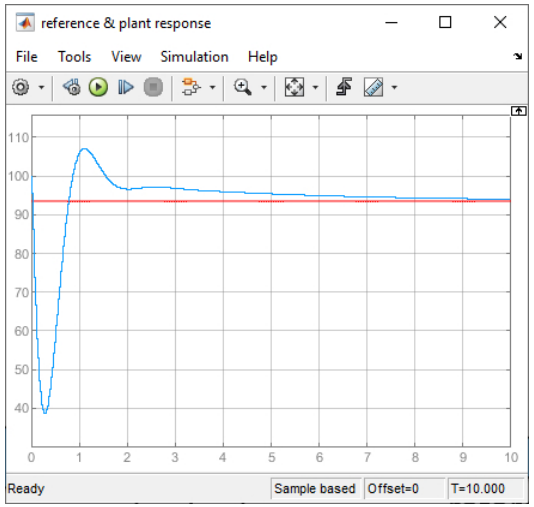
Getting started – First example

- Build - convert the model into a Linux executable
- Load - upload the executable to the simulator
- Execute - start the simulation









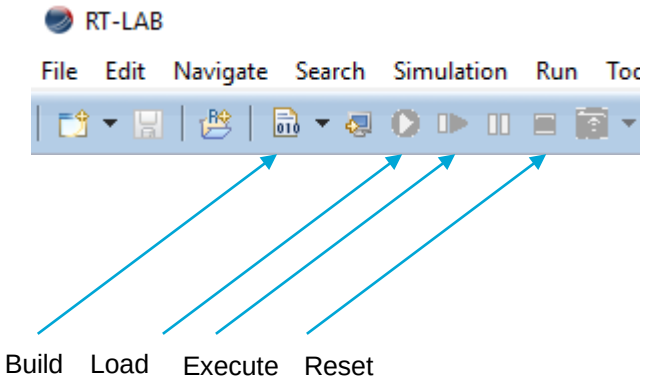
TU Delft

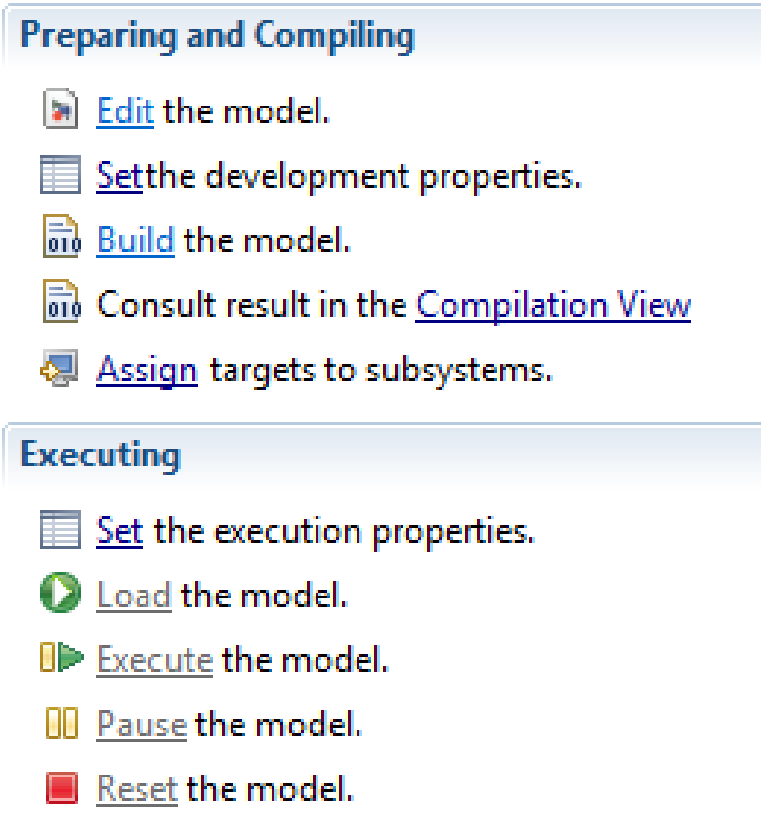
Getting started – First example

- Reset – Stop the simulation

After resetting, the simulator is in idle mode.

<Reset> does not reboot the simulator, it just stops the simulation





Agenda

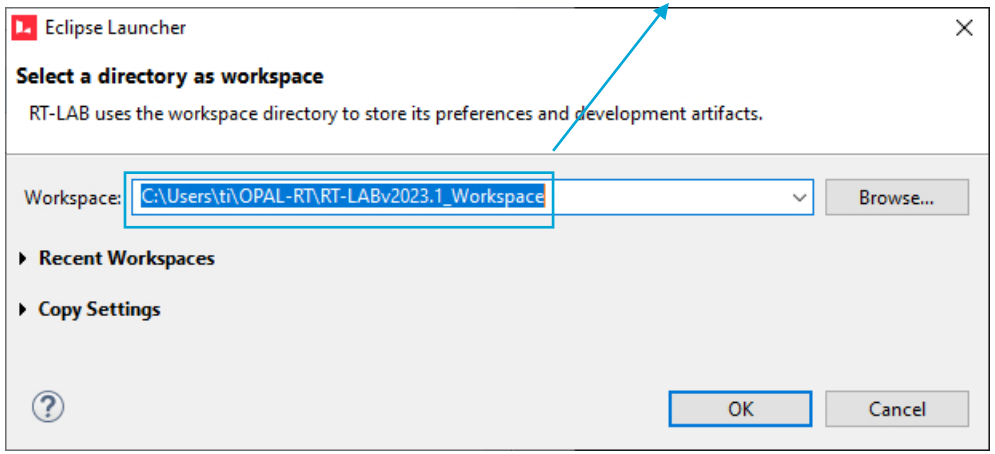
1. Getting started
2. Complete workflow
   - Creating a project
   - Model structure – subsystems
   - Simulation parameters
   - Building and running the model
   - Execution performance
   - Probe control
   - Parameters, variables & signals
   - Data logging
   - Panels

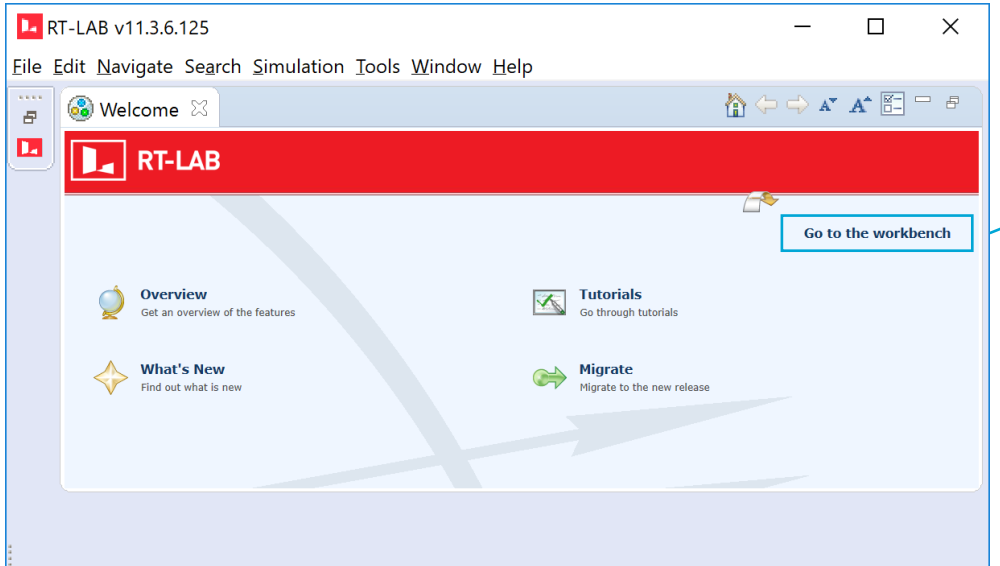
Complete workflow – Creating a project

- Projects and models - launching RT-LAB

RT-LAB.exe

RT-LAB work folder

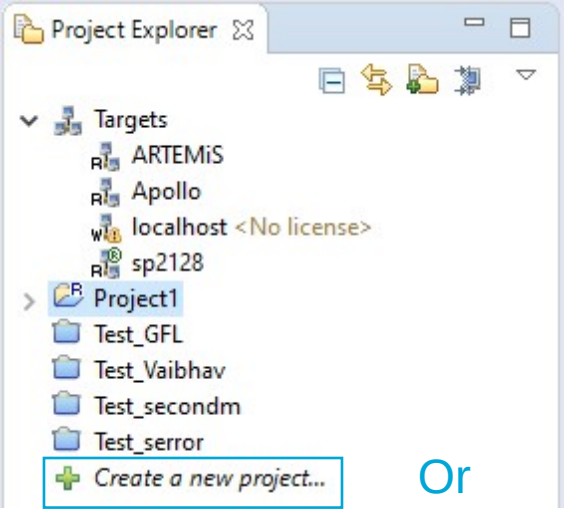


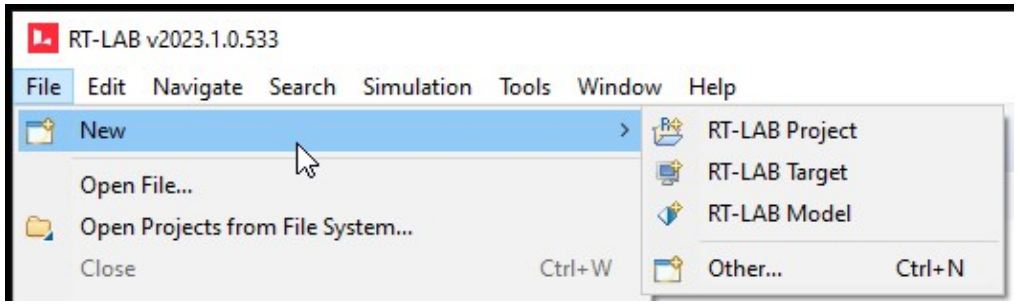


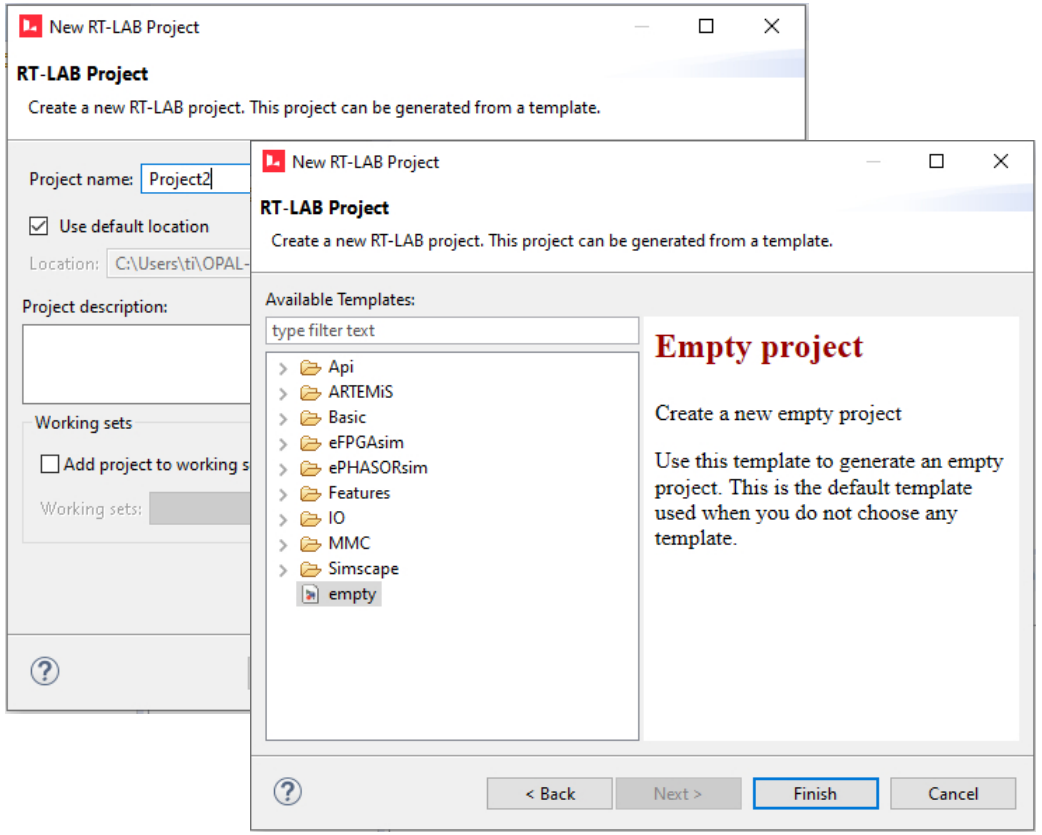
1st launch: <Go to the workbench>

Complete workflow – Creating a project

- Projects and models – creating a project

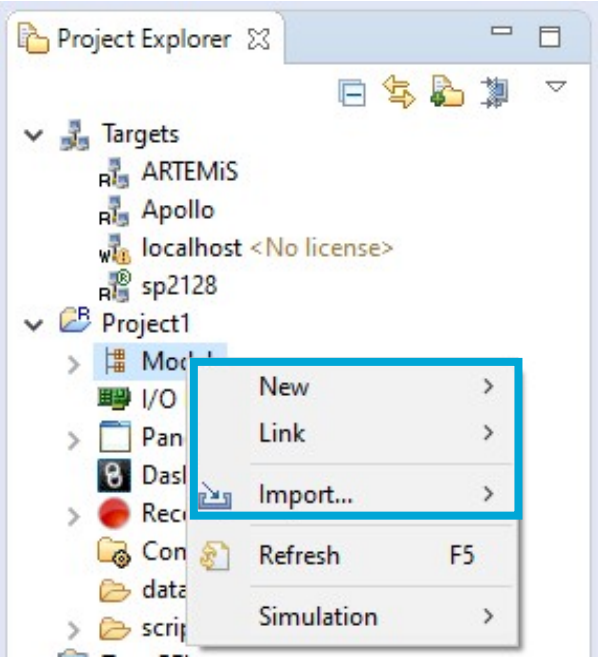






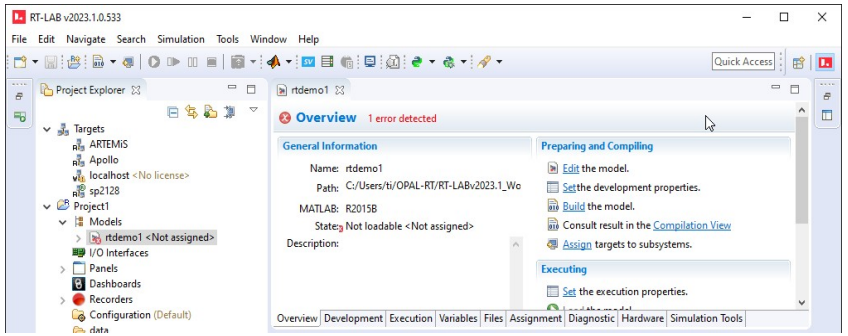
Complete workflow – Creating a project

- Projects and models – adding a project

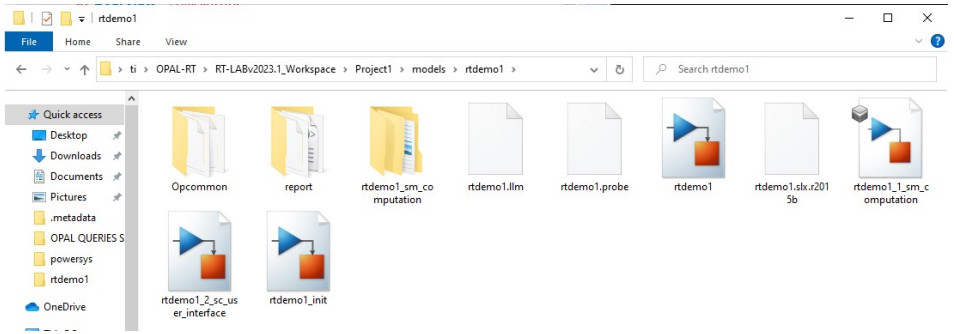


New: Create a new model from scratch
Link: Point to an existing model
Import: Copy an existing model to the RT-LAB workspace

C:/Users/ti/OPAL-RT/RT-LABv2023.1_Workspace/Project1/models/rtdemo1/rtdemo1.slx

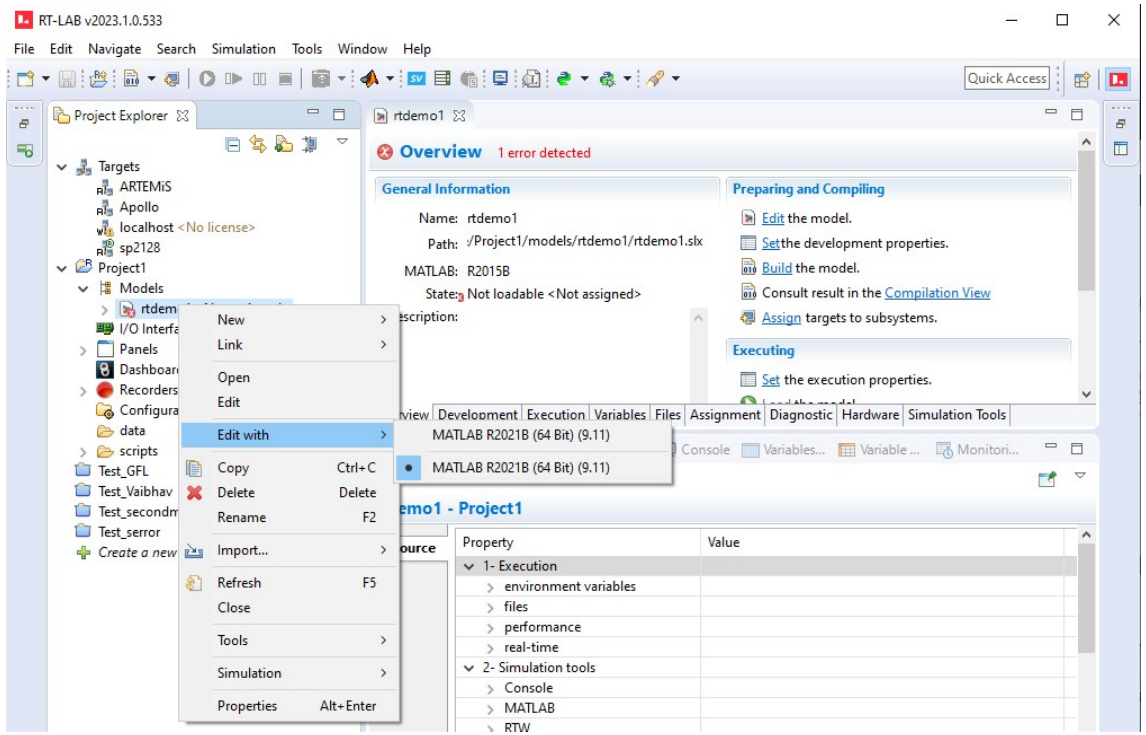


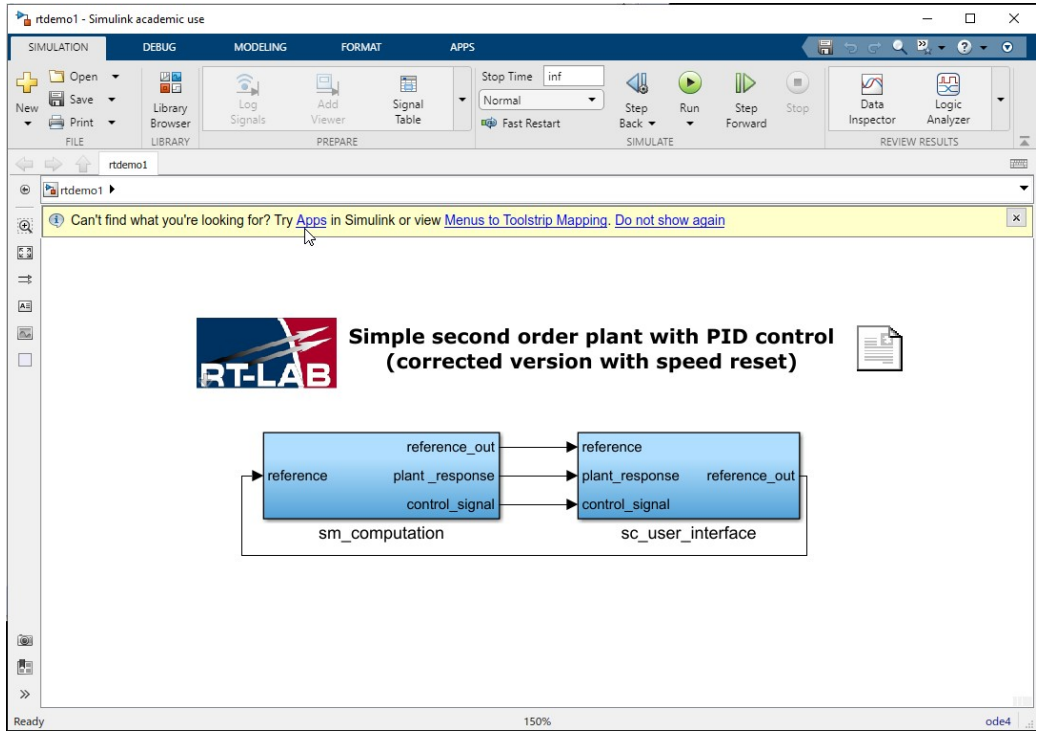
Model folder



Complete workflow – Creating a project

- Projects and models – opening a project

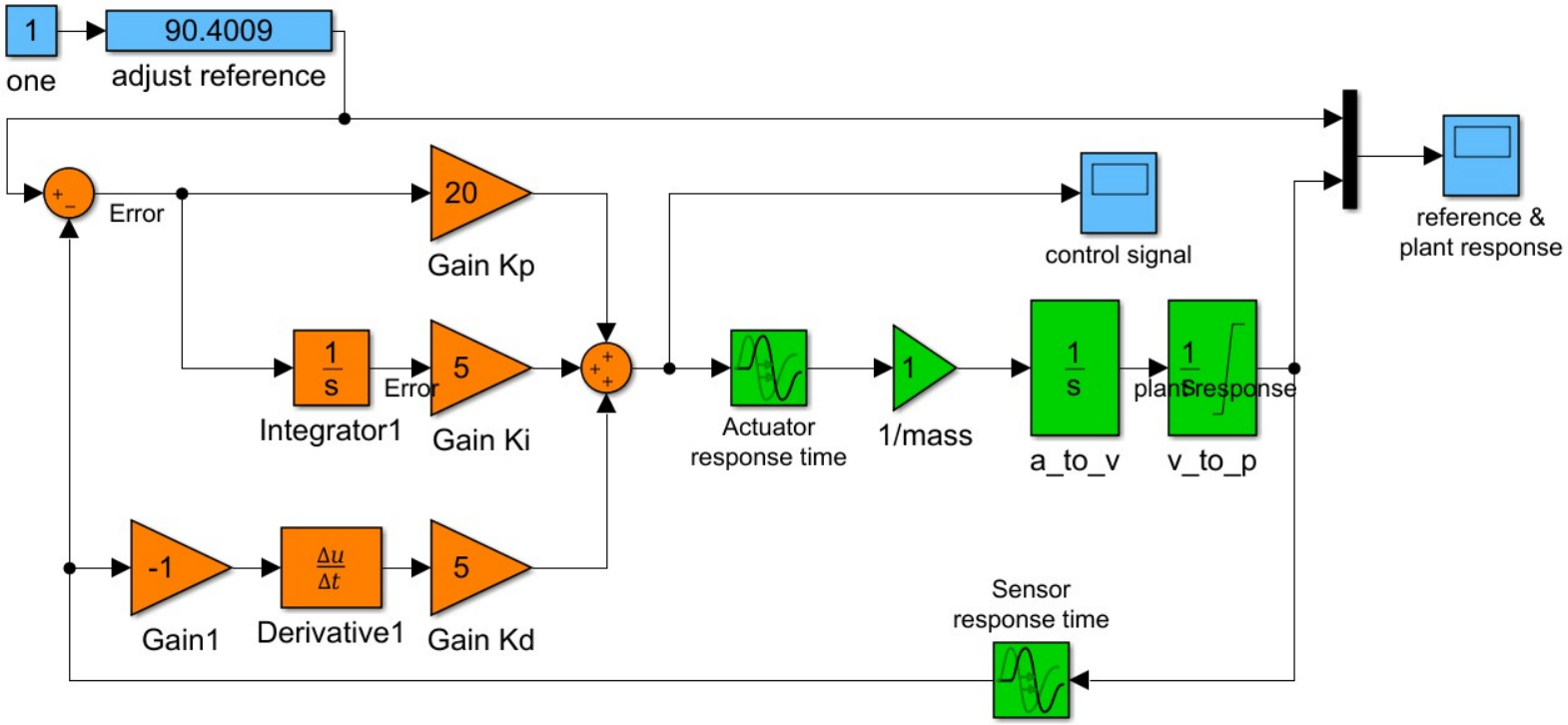




Complete workflow – Creating a project

Hands-on 1

1. Create a project in RT-LAB
2. From the Models section, import model rtdemo1.slx
3. Edit the model with Simulink



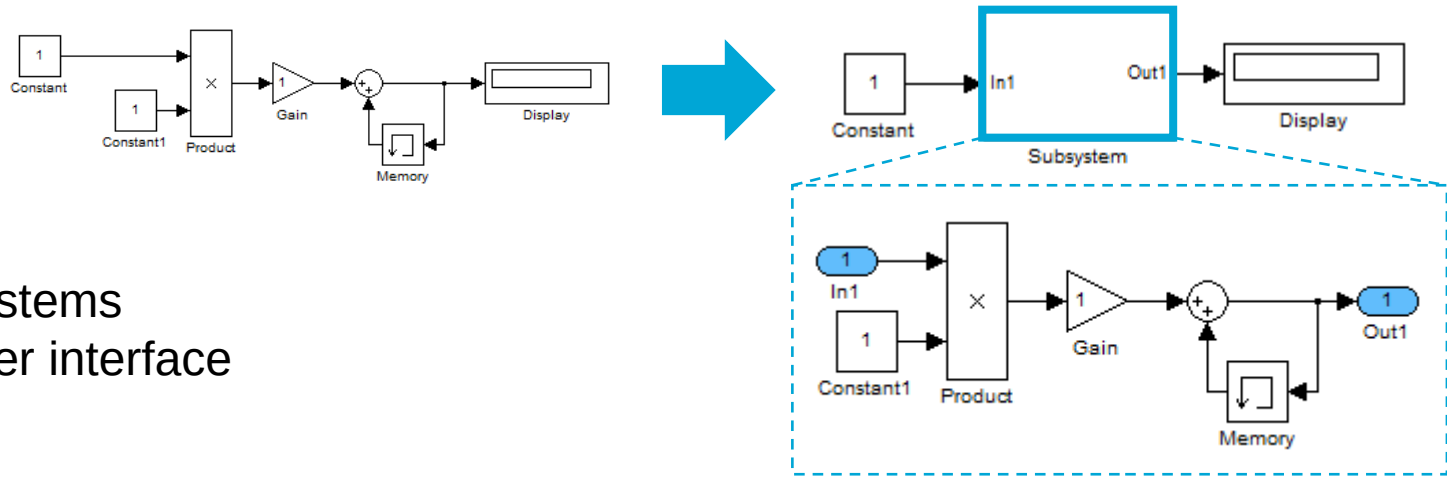
Complete workflow – Model structure - Subsystems

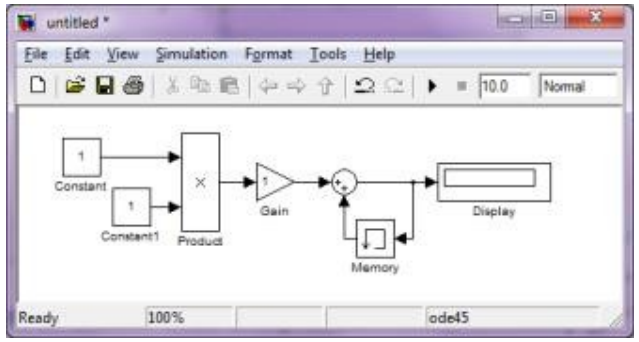
Grouping into subsystems

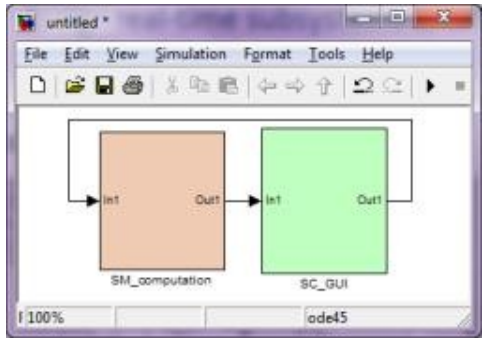
1. Simplify the model by grouping blocks
2. Establish hierarchical block diagram
3. Keep functionally related blocks together

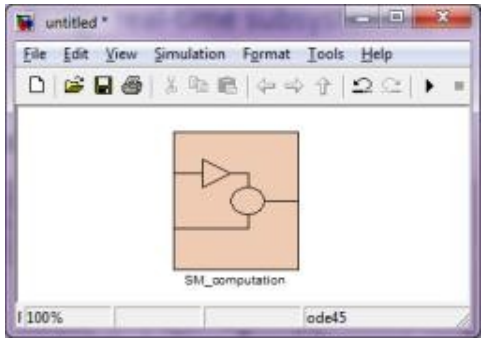
---- Top-level of a Simulink model: only subsystems

---- Distinguish computation and graphical user interface







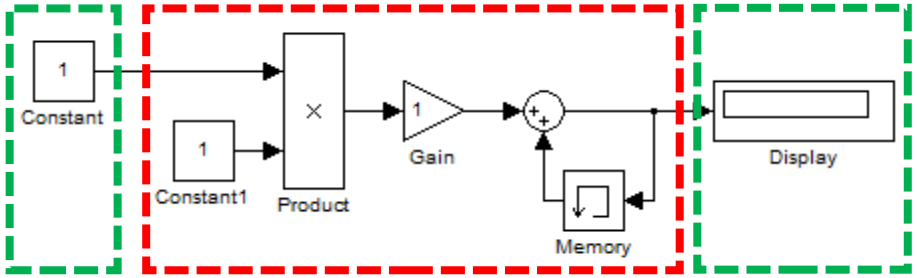


Complete workflow – Model structure - Subsystems

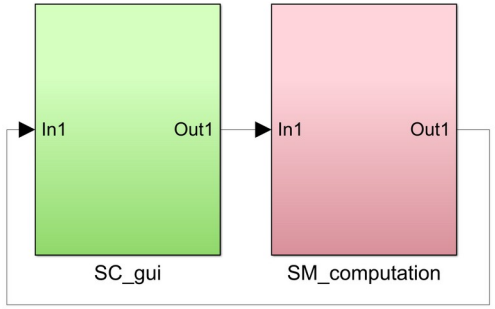
Grouping into subsystems

SC_anyname
1. Graphical interface – not computation. Runs on laptop/desktop.
2. Mainly contains scopes, displays, constants, manual switches

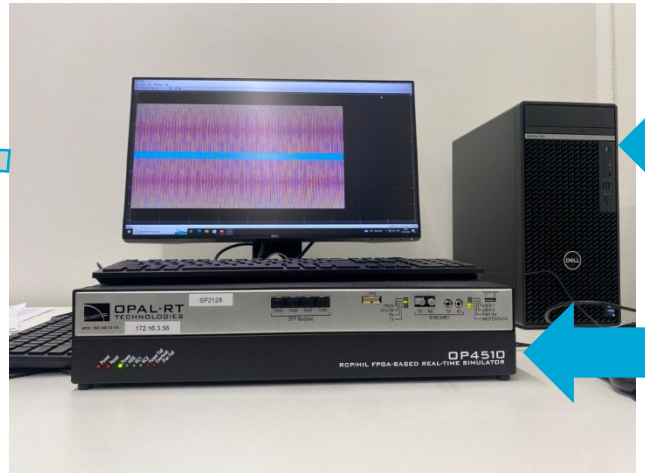
SM_anyname
3. Real-time or accelerated computation (mathematical, logic, I/Os, signal generation, physical models).
4. Uses 1 CPU core of simulator.
5. To use more CPU cores, define extra SS_subsystems (1 per extra core)

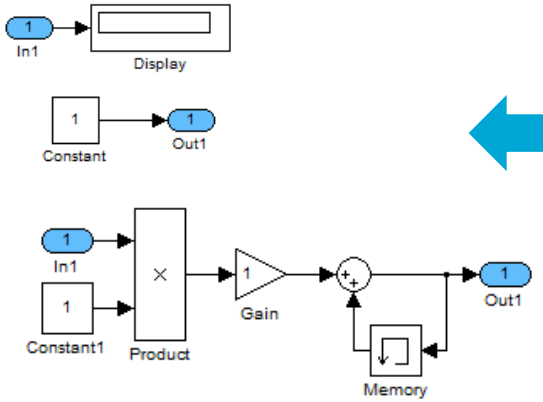


GUI Computation GUI



TCP/IP





Complete workflow – Model structure - Subsystems

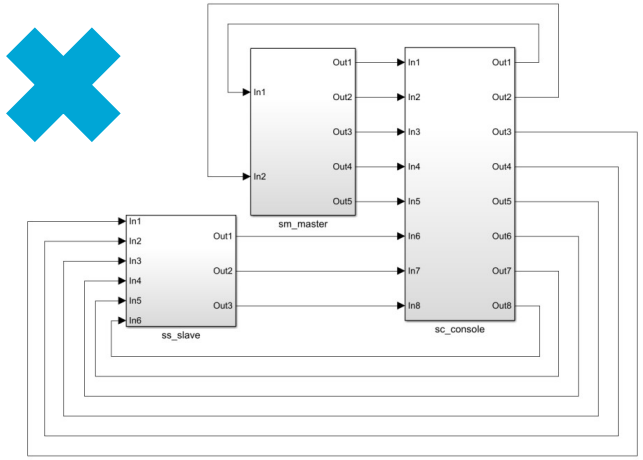
Grouping into subsystems

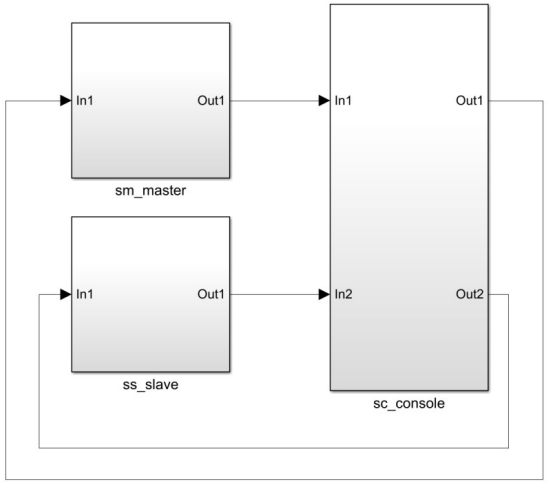
Bad practices

1. Exchanging many signals between subsystems may worsen model readability and maintenability.

Good practices

2. Buses or muxed signals can be exchanged between subsystems.
3. It’s preferable to exchange such signals to simplify the model top-view and improve maintenability.





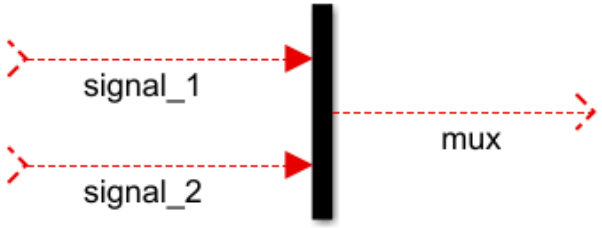
Complete workflow – Model structure - Subsystems

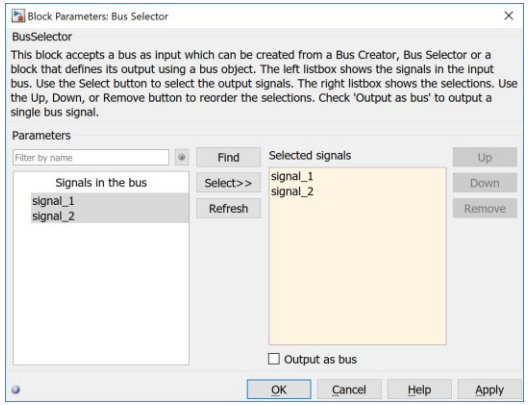
Bad practices

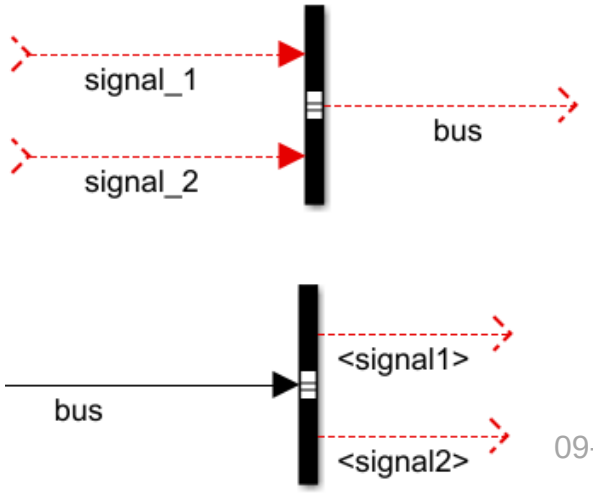
1. **Mux** and **Demux** blocks allow the gathering and break up of signals on a single <wire>.
2. Signal management is based on the indexes of signals.
3. Signal name is not propagated
4. Not the best solution for readability and maintenability.

Good practices

5. **Bus Creator** and **Bus Selector** blocks allow the gathering and break up of signals on a single <wire>.
6. Signal name is propagated.
7. Signals can be selected.
8. Readability and maintenability are improved.



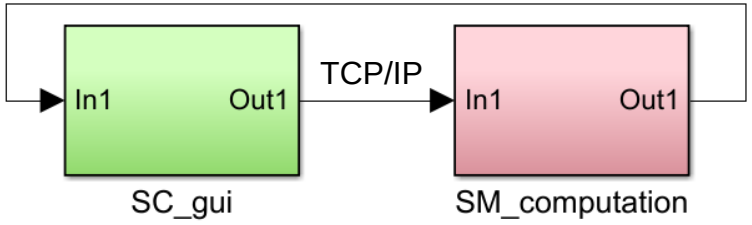


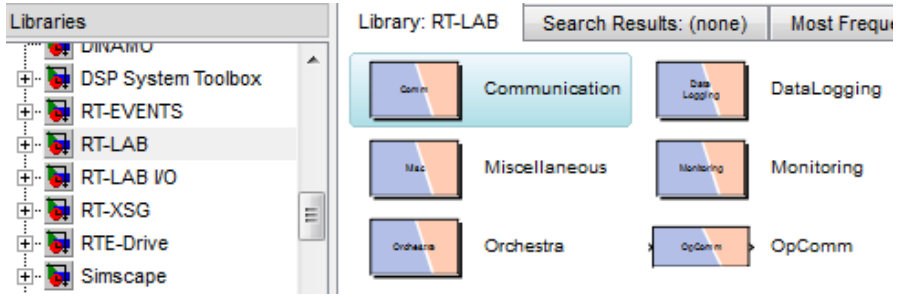


Complete workflow – Model structure - Subsystems

OPCOMM

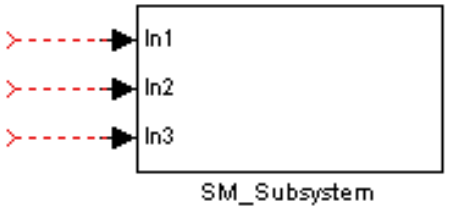
- Responsible for TCP/IP communication between host and target.

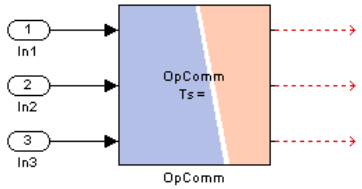


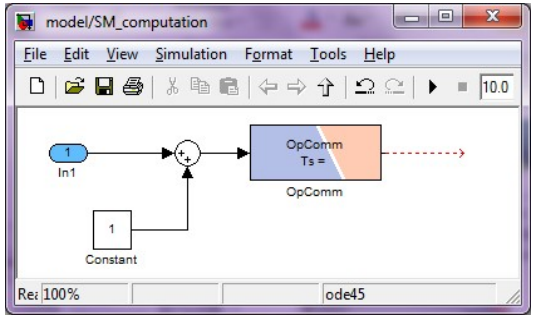


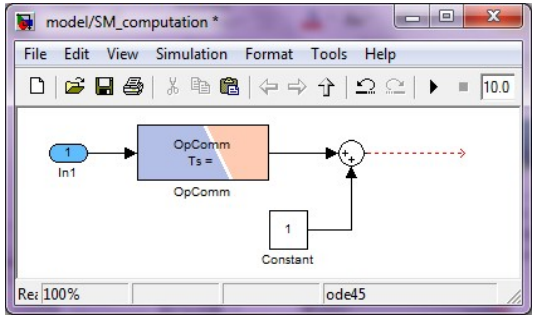
Has to be placed

- Inside SM, SS, SC
- All inports connected to OpComm



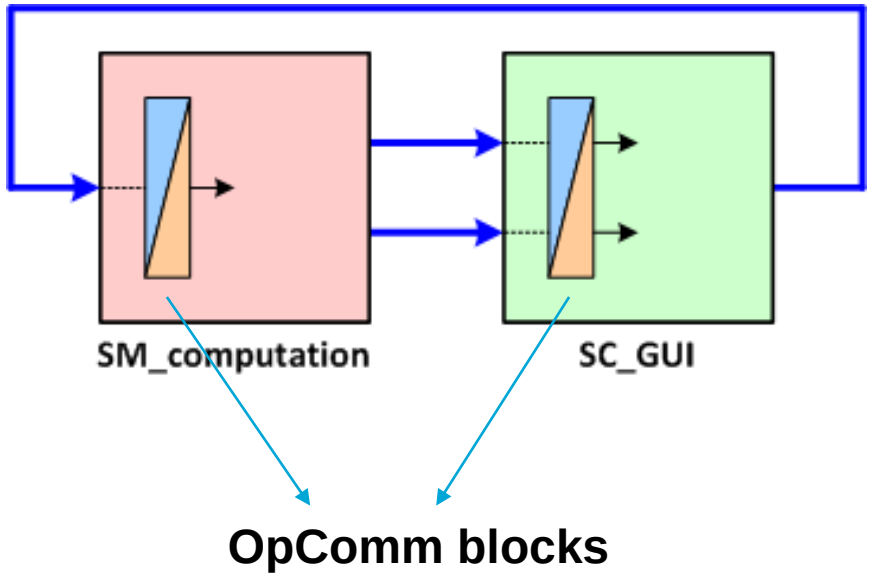




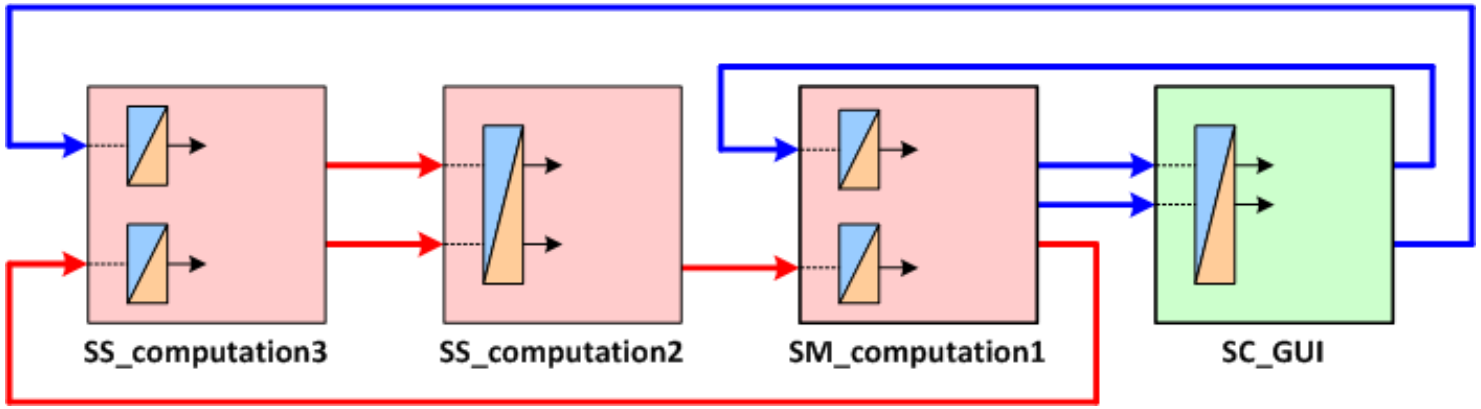


Complete workflow – Model structure - Subsystems

Typical case
- SM running on 1 CPU core.



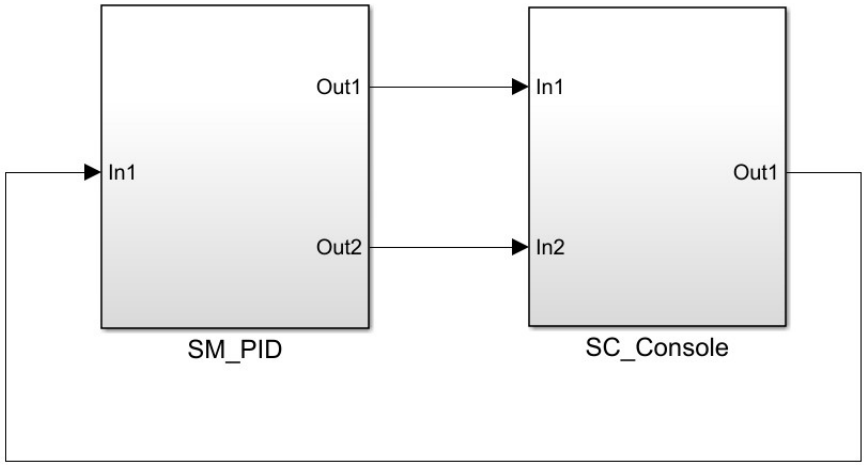
Multiple subsystem case
- Each SM and SS on 1 CPU core
- All synchronized signals (red) grouped into 1 OpComm
- All TCP/IP signals (blue) grouped into 1 OPComm

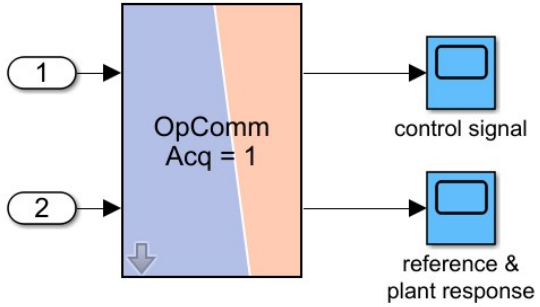


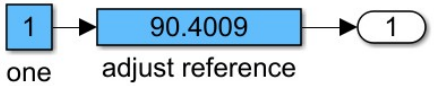
Complete workflow – Model structure - Subsystems

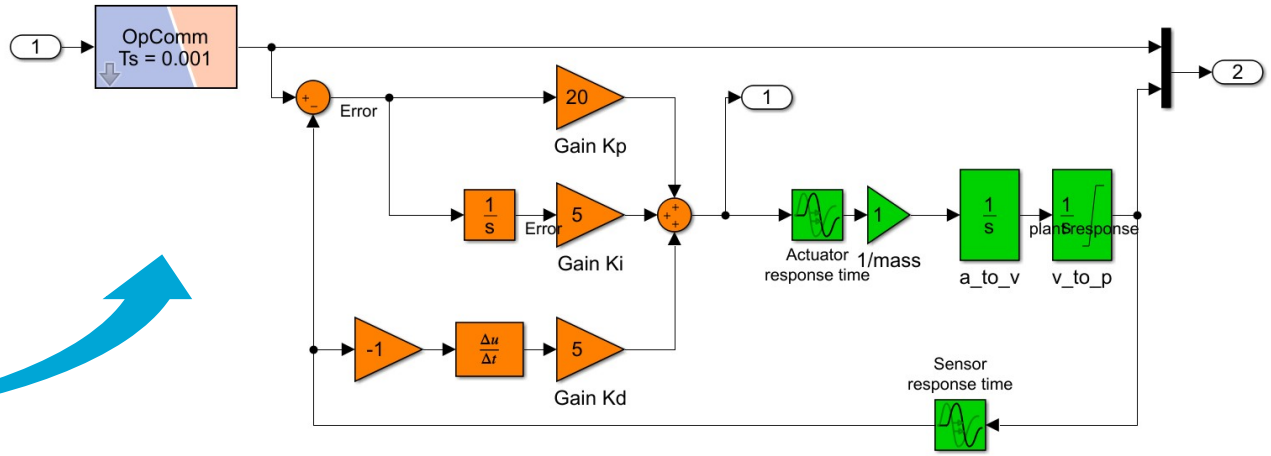
Hands-on 2

1. From model rtdemo1.slx, create subsystems SM_ and SC_
2. Add OpComm blocks to the model







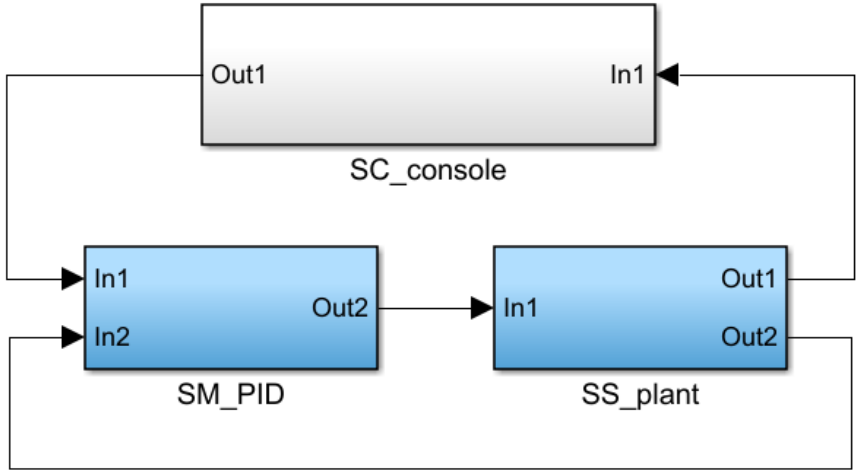


TU Delft

Complete workflow – Maximizing parallel execution

Let's assume a multiple-subsystem case

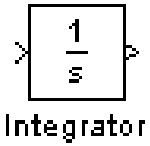
1. SM and SS are running on 2 different CPU cores…
2. … but it does not mean they run in parallel !



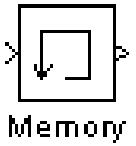
State - Definition

Output (signal) which is computed only from preceding inputs or outputs.

Example blocks with states



$$y_z = y_{z-1} + x_{z-1} \Delta t$$



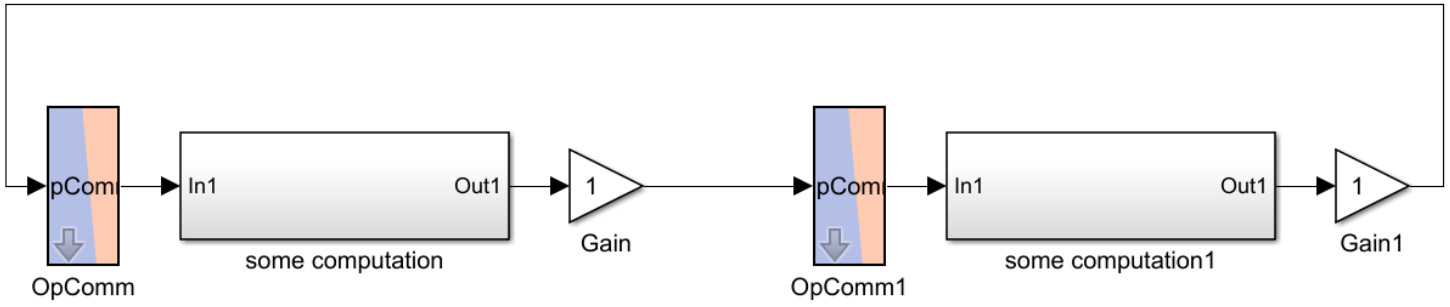
$$y_z = x_{z-1}$$

Example block without a state

$$y_z = A x_z$$

Complete workflow – Maximizing parallel execution

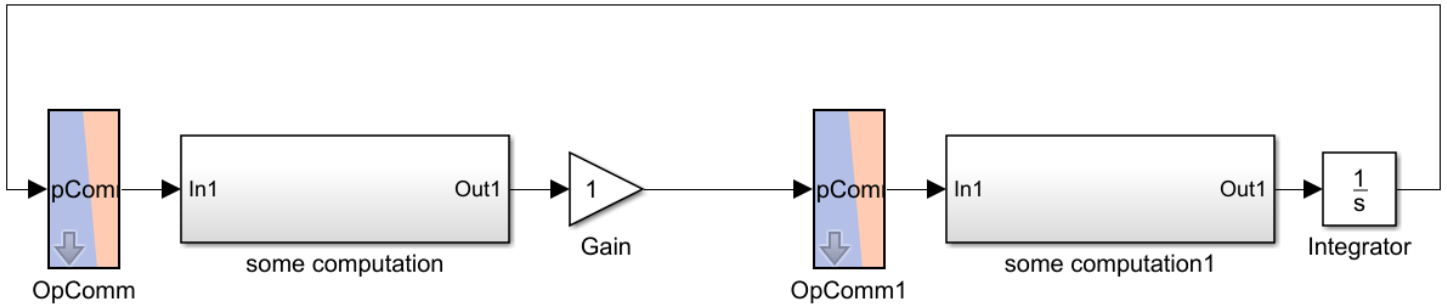
Deadlock - <will not execute> case



SM_computation1 SS_computation2

There is no « entry point» for the computation : each subsystem waits for the other one.

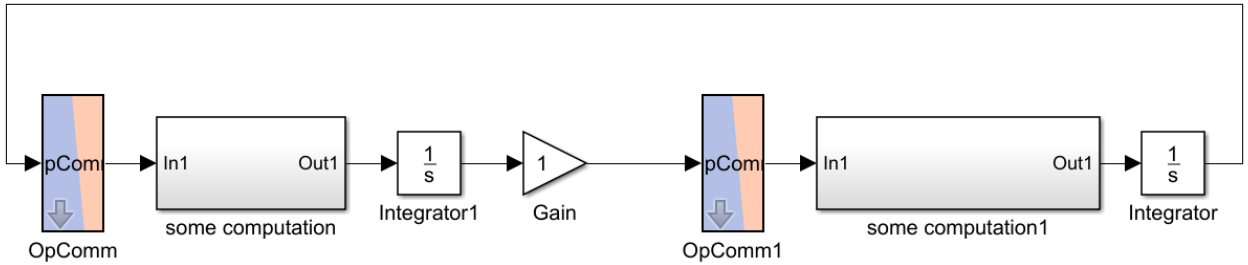
Serial execution – Bad case



SM_computation1 SS_computation2

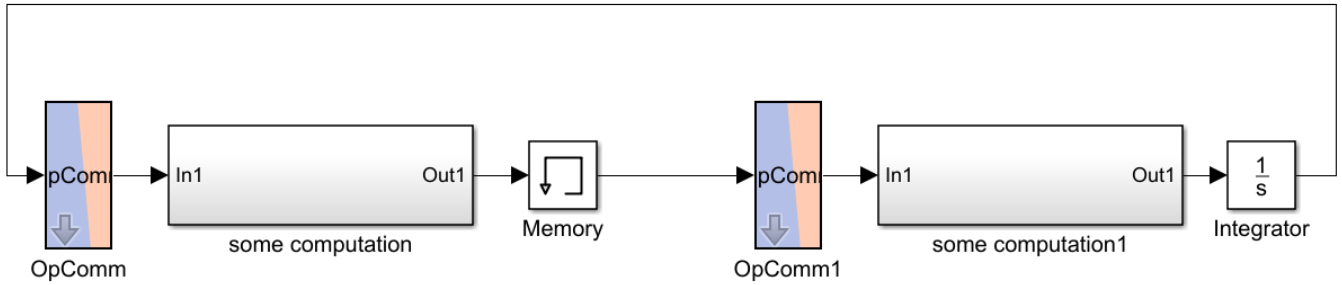
Complete workflow – Maximizing parallel execution

Partially parallel execution – Intermediate case



SM_computation1 SS_computation2

Full parallel execution – Best case

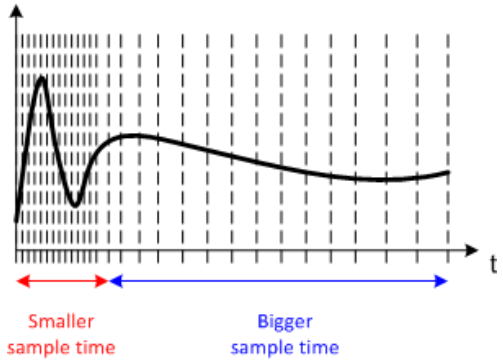


SM_computation1 SS_computation2

Complete workflow – Simulation parameters

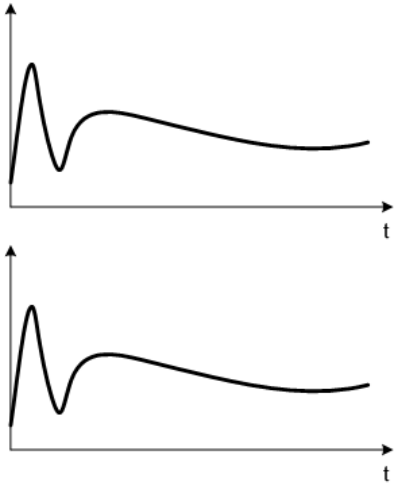
Variable-step solver

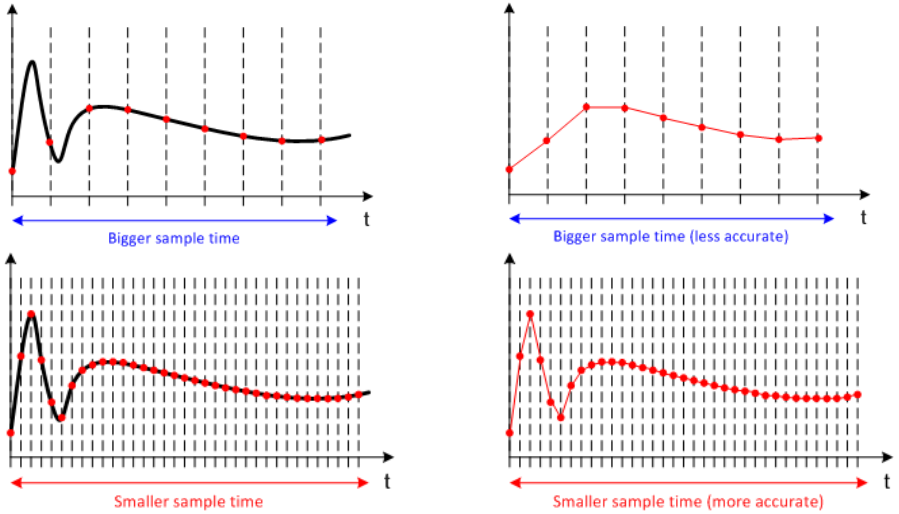
- Automatic setting of Ts
- Many iterations of same step
- Not deterministic

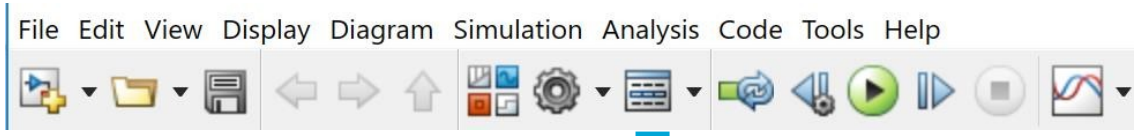


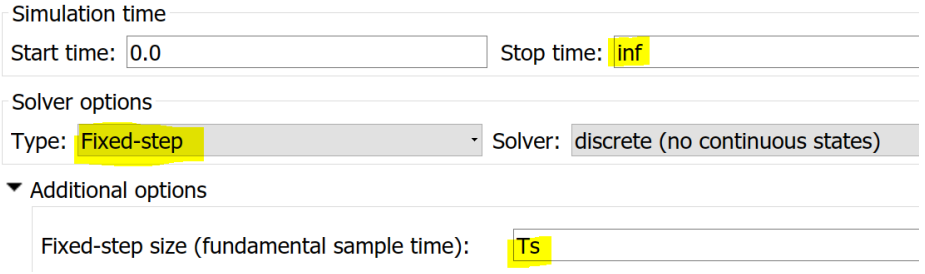
Fixed-step solver

- Fixed-step solver is mandatory for RT simulation
- Set time step Ts
- Stop time = inf







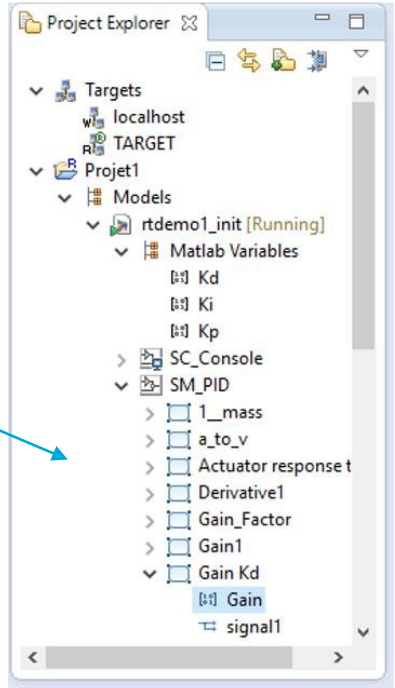


TU Delft

Complete workflow – Simulation parameters

Optimization options

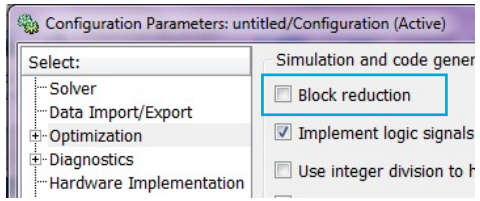
By default, Simulink optimizes the code it generates with the code generator. This optimization can lead to some blocks' outputs or some parameters not being available for selection in the Project Explorer.

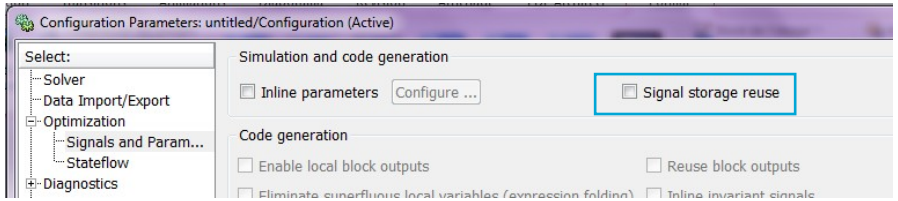


Missing parameters or signals !

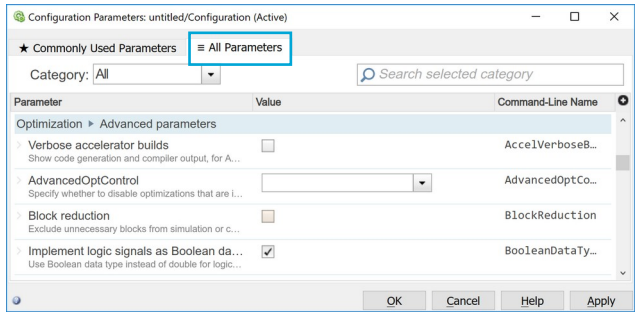
Setting simulation parameters

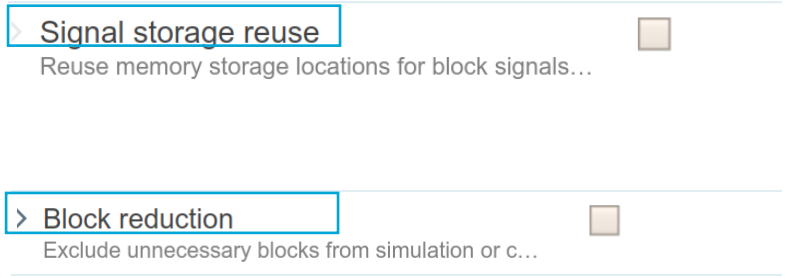
Older MATLAB versions





Newer MATLAB versions





Complete workflow

Before moving to the next step (build and execute model in the simulator), please make sure:

- The model has been saved
- The model runs offline

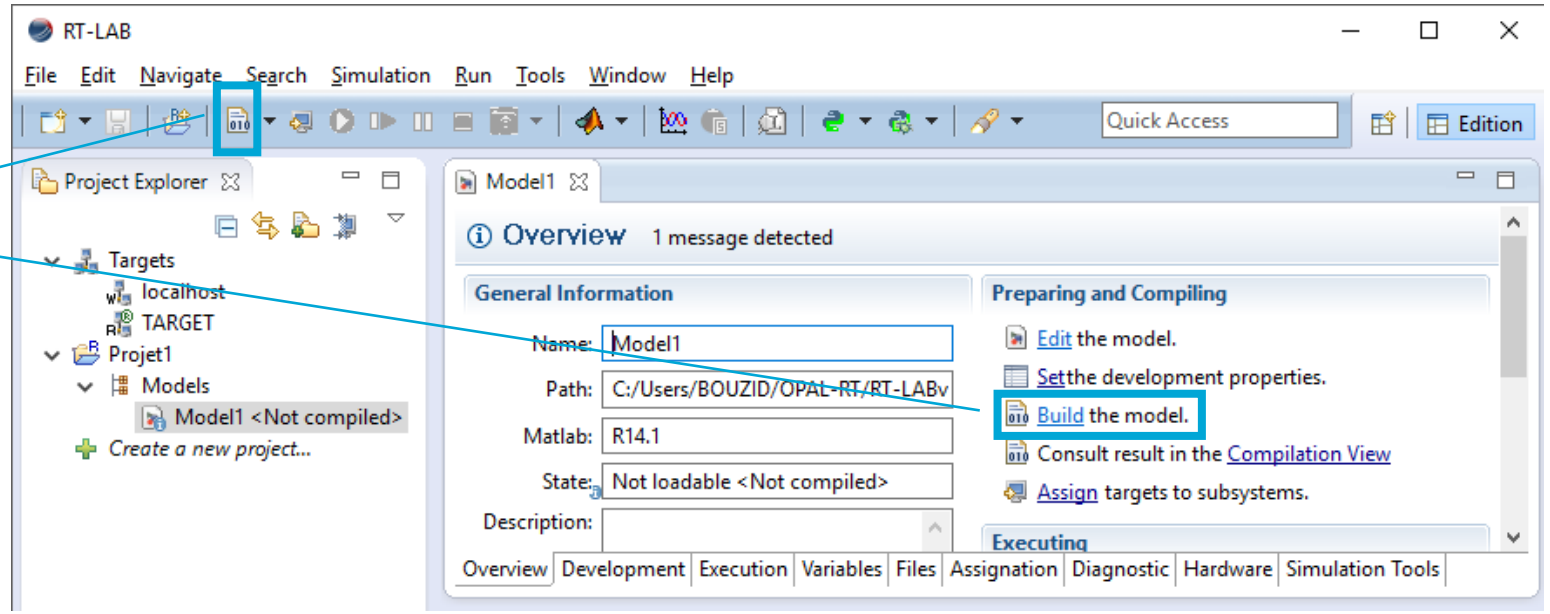
1. Create subsystems
   - Isolate computation and GUI blocks
   - Computation blocks: mathematical, logic, signal generation, transfer functions, physical models
   - GUI blocks: display, scope, constant, manual switch
2. Rename subsystems
   - SC_xxx : Graphical interface
   - SM_xxx : Main computation subsystem. Uses 1 CPU core
   - SS_xxx : Additional computation subsystem. Uses 1 extra CPU core
3. Add OpComm blocks
   - Inside SM, SS, SC
   - Usually 1 OpComm per subsystem
   - All inports go through OpComm blocks (an OpComm can have multiple ports)
4. Simulation parameters
   - Fixed step solver
   - Set the value of Ts
   - Stop time = inf

Complete workflow – Building & running the model

Building a model

1. During this step, RT-LAB:
2. Checks the Simulink model
3. Separates the subsystems
4. Generates code from subsystems
5. Transfer the code to the target, compiles it and links it with libraries
6. Transfer back the executable file to host PC

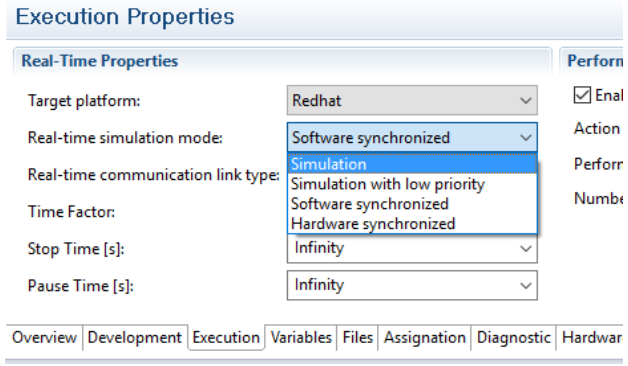
Build the model



Complete workflow – Building & running the model

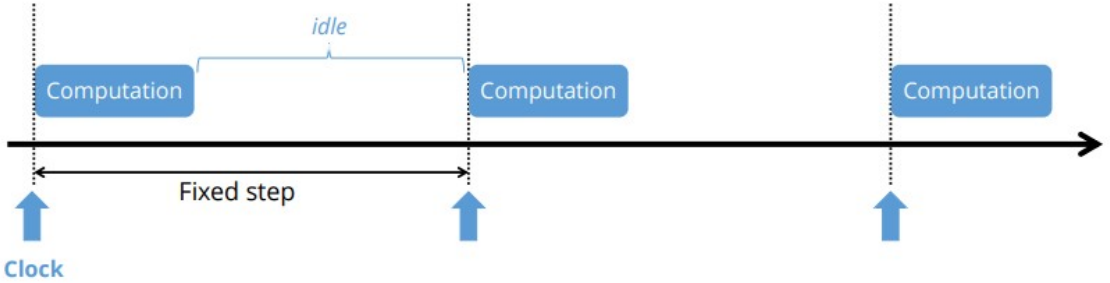
Preparing model execution – Simulation modes

1. Simulation: <as fast as possible> mode
2. Simulation with low priority: <as fast as possible> mode for Windows targets
3. Hardware Synchronized: synchronized mode (real-time), FPGA clock (using I/Os)
4. Software Synchronized: synchronized mode (real-time), CPU/OS clock (no I/Os)



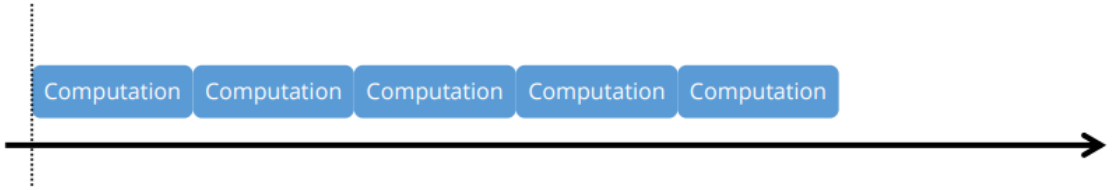
Preparing model execution – Simulation modes

1. Hardware/Software Synchronized
2. A clock signal synchronized the simulation



Preparing model execution – Simulation modes

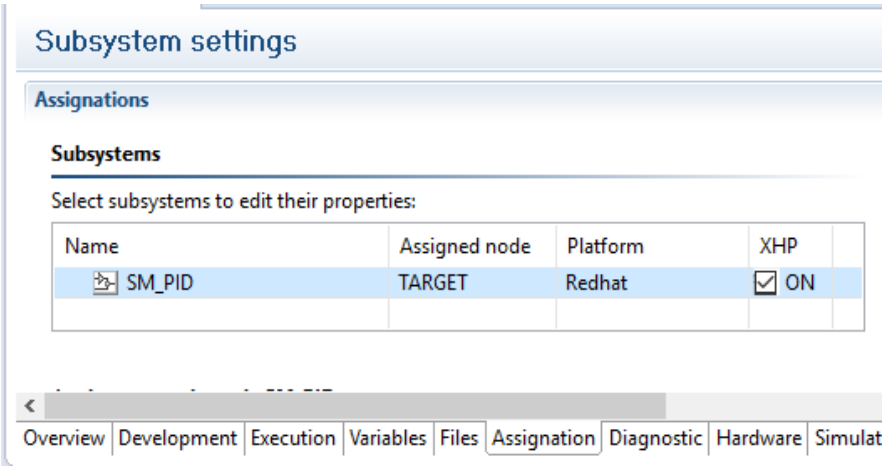
1. Simulation/Simulation with low priority
2. Simulation runs as fast as possible

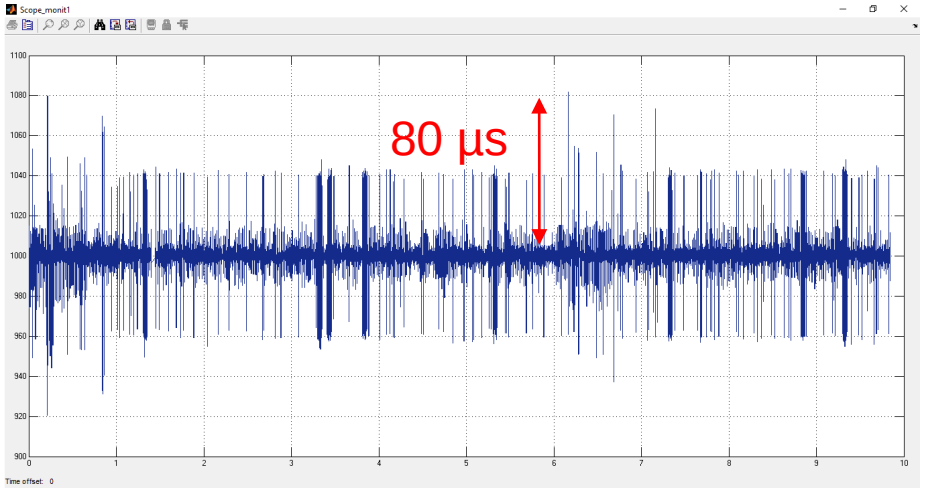


Complete workflow – Building & running the model

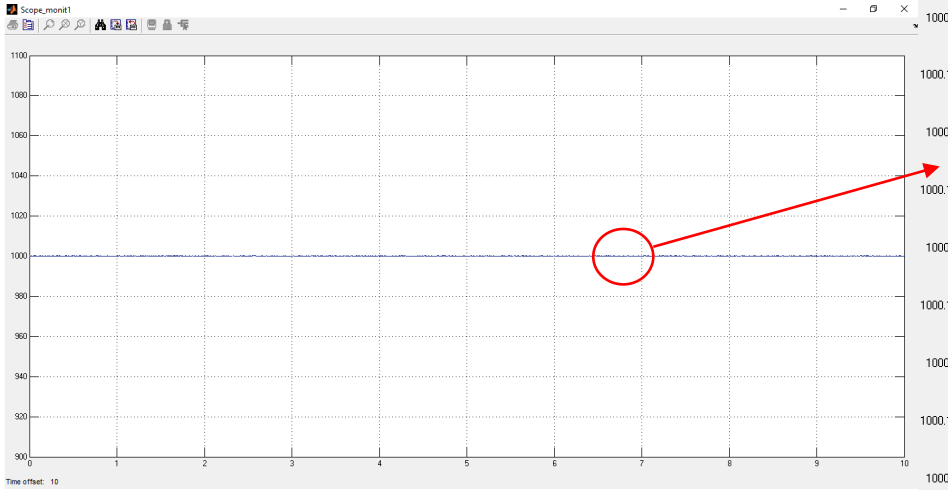
Preparing model execution – XHP: eXtra High Performance

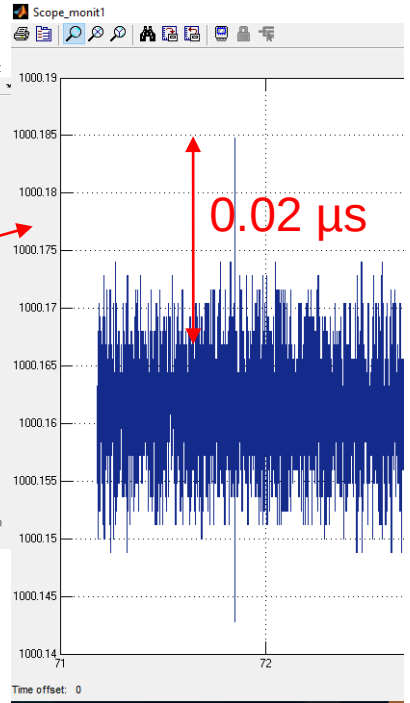
1. Used during real-time simulation (Software/Hardware synchronized)
2. Strongly recommended for small time steps (Ts < 100 µs)
3. Shields the CPU cores and reserves them for the models
4. Prevents interruptions from other OS processes
5. Considerably reduces jitter
6. Accelerates the simulation





Without XHP





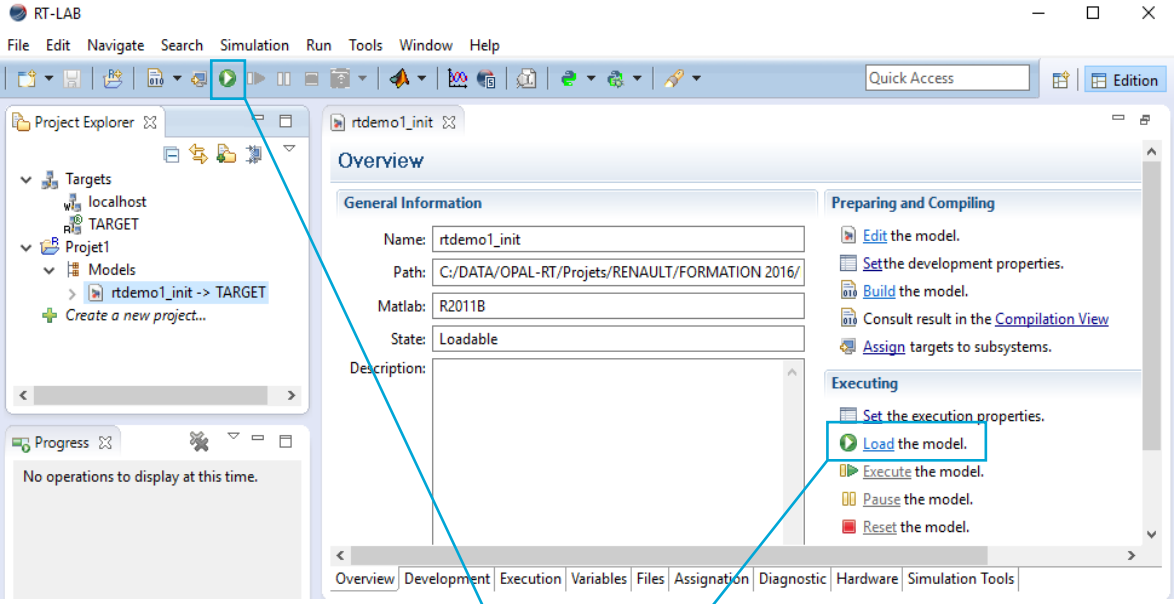
With XHP

TU Delft

Complete workflow – Building & running the model

Load

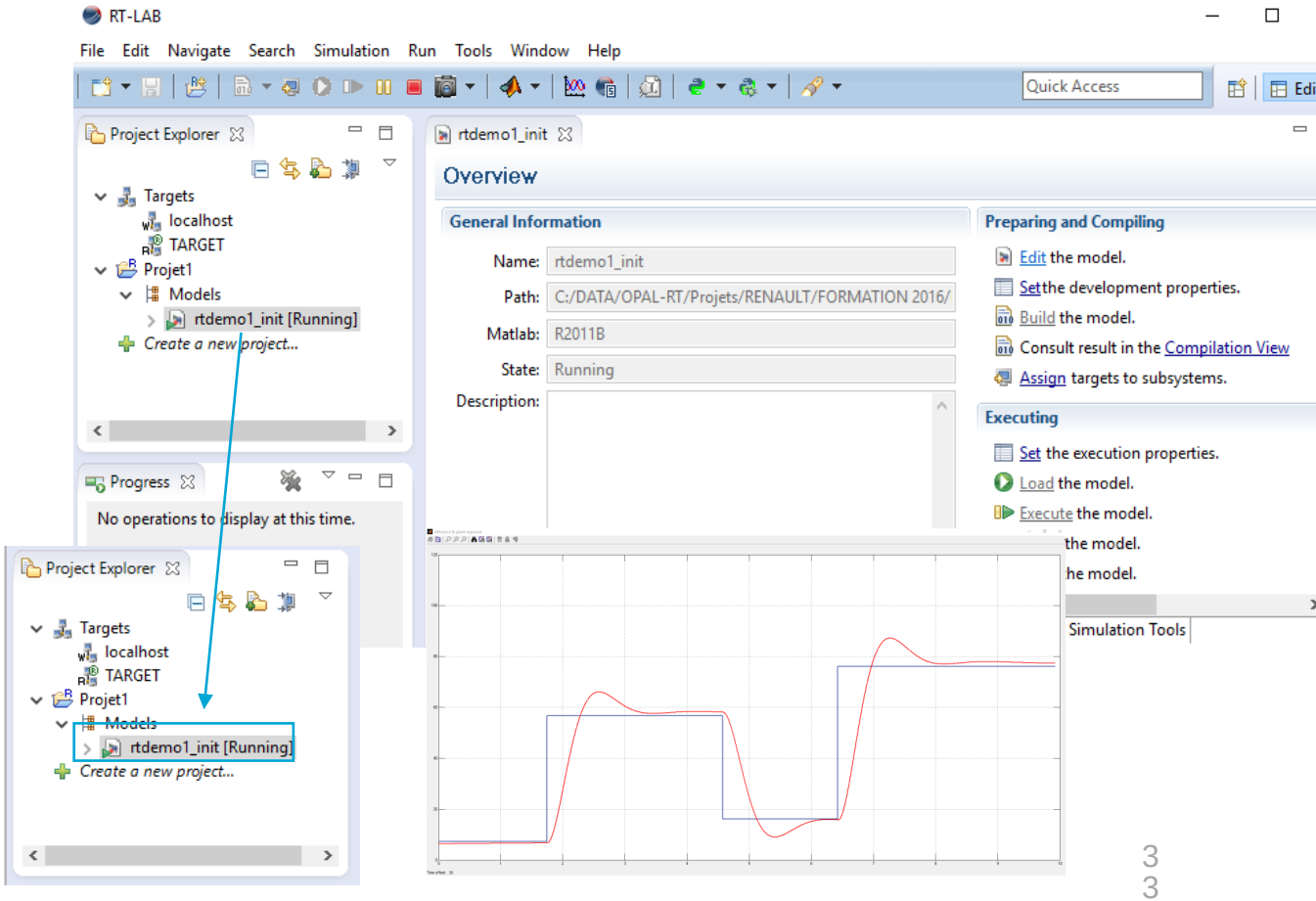
1. Uploads the executable file to the target
2. Allocates cores and memory
3. Launches the user interface (Simulink console)



Load the model

Execute

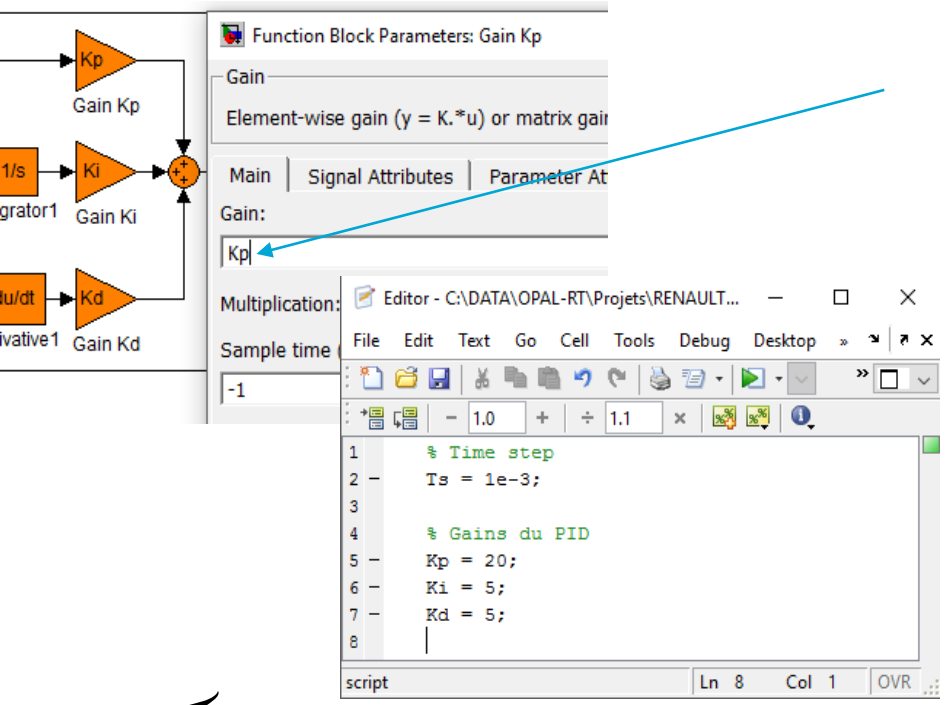
1. Starts the simulation



Complete workflow – Parameters, variables & signals

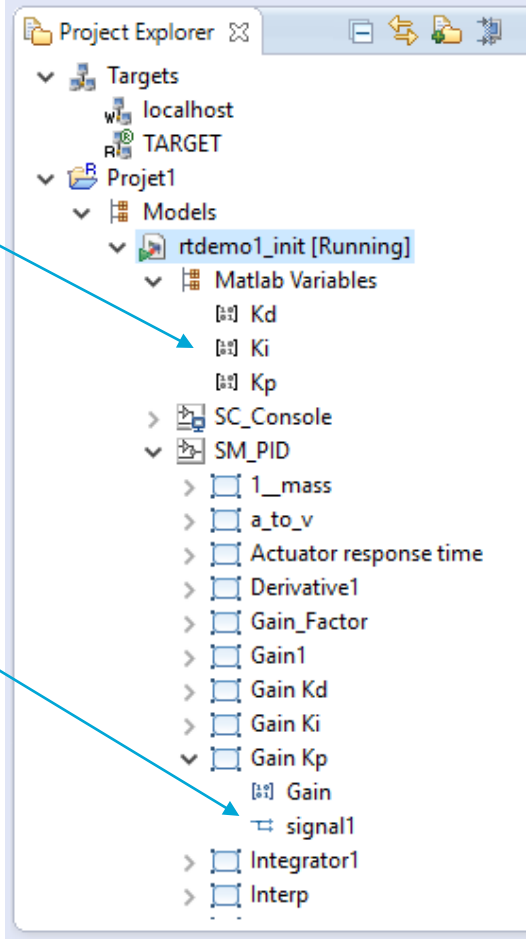
Parameters

RT-LAB allows the access (write mode) to block parameters of SM and SS subsystems during the simulation



Parameters defined as symbolic variables
(~ global variables)

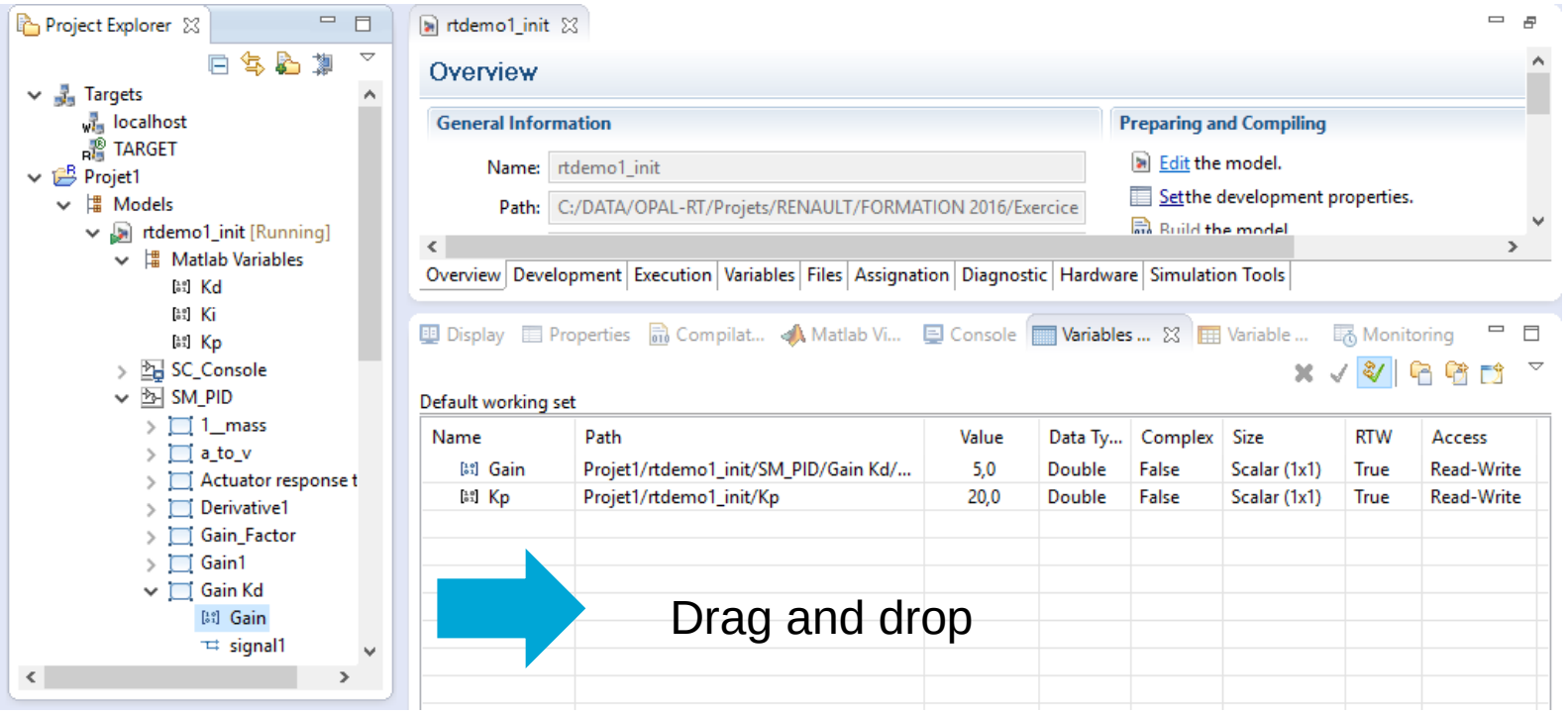
Internal parameters of blocks



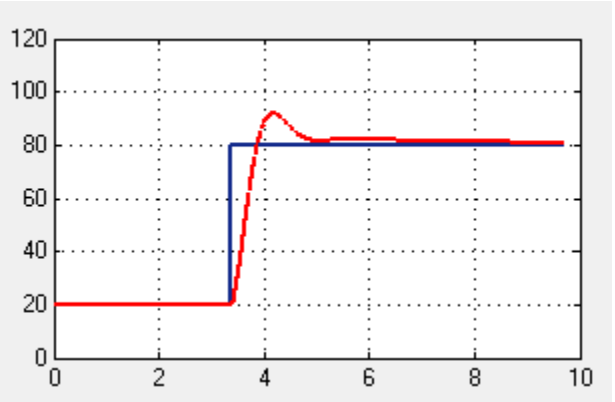
TU Delft

Complete workflow – Parameters, variables & signals

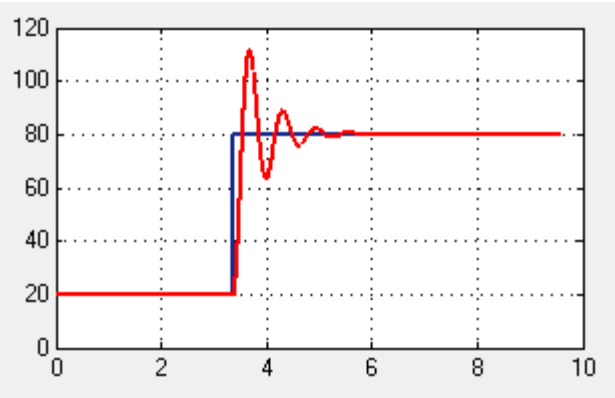
Parameters – Variable Table



Kp = 20



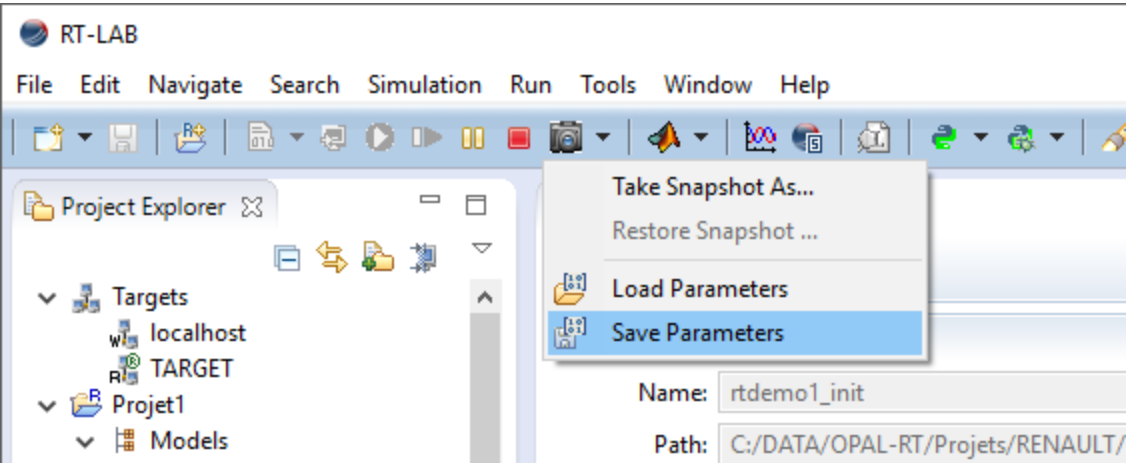
Kp = 100

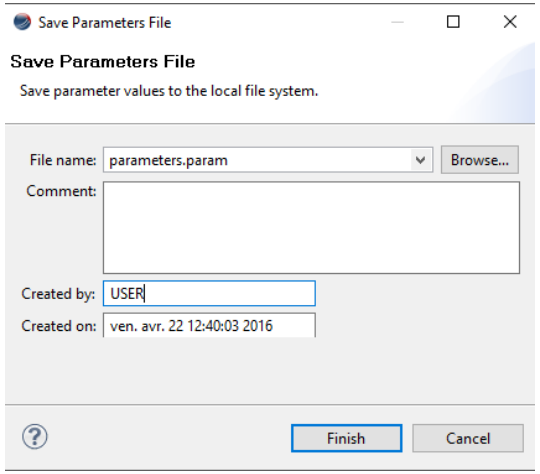


Complete workflow – Parameters, variables & signals

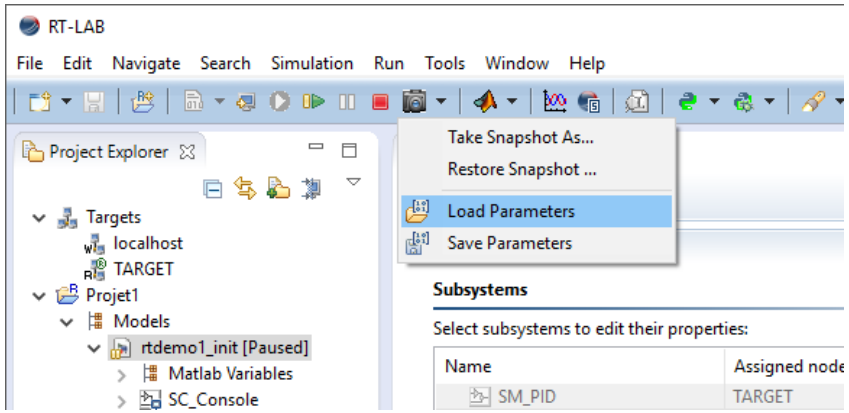
Parameters – Saving & Loading parameters

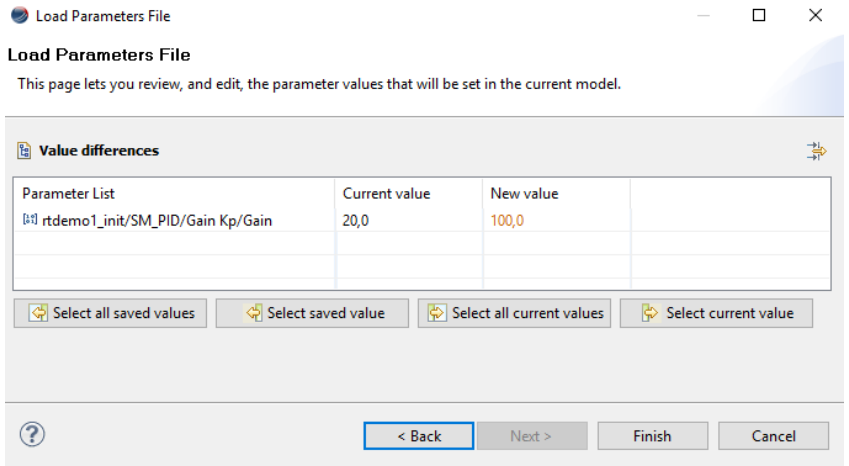
As soon as the model is reset, the modified parameter values are lost.





Parameters can be saved in a file and relocated on a later simulation.



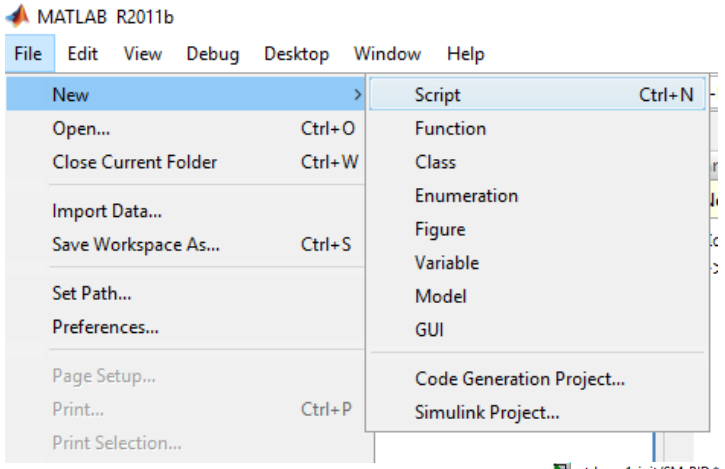


TU Delft

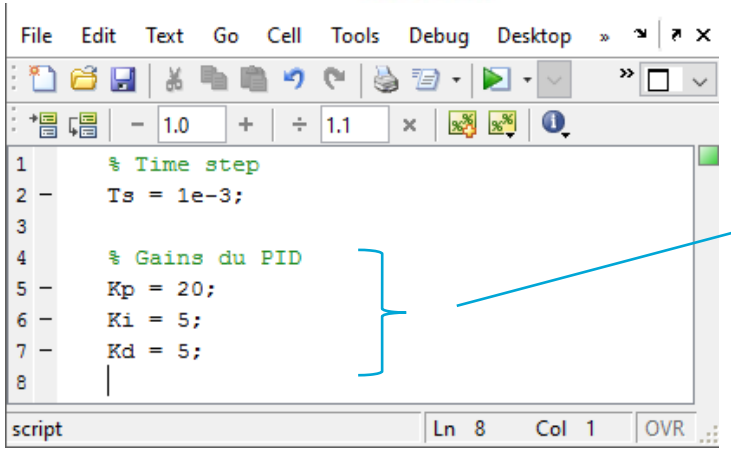
Complete workflow – Parameters, variables & signals

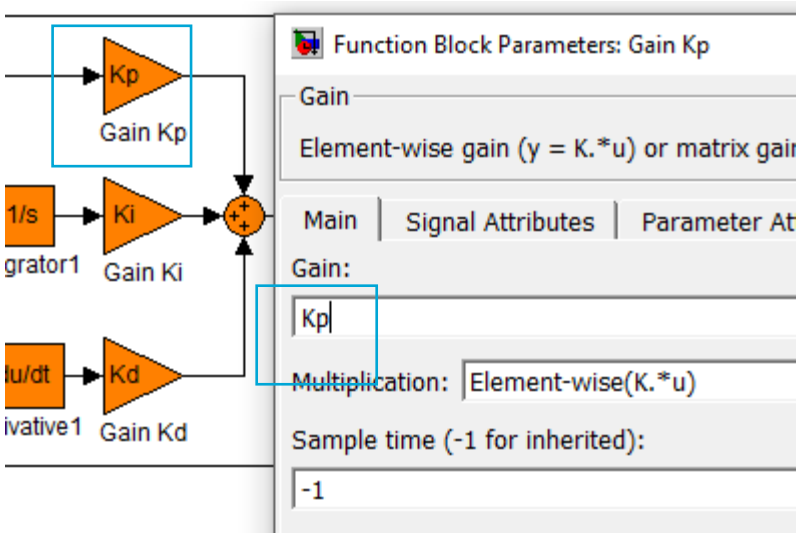
Parameters – Declaring variables in M-Files

MATLAB script *.m, initializes model variables

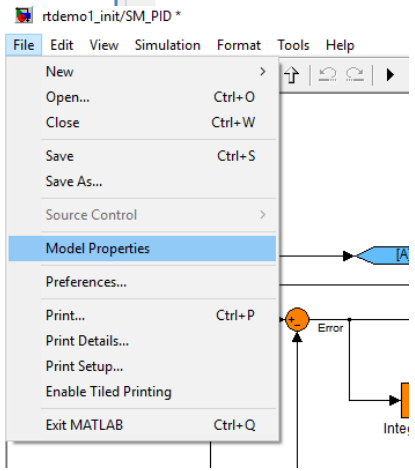


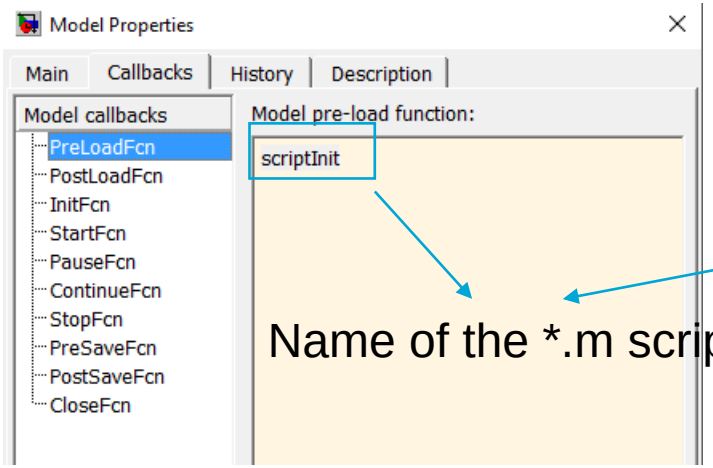
scriptInit.m



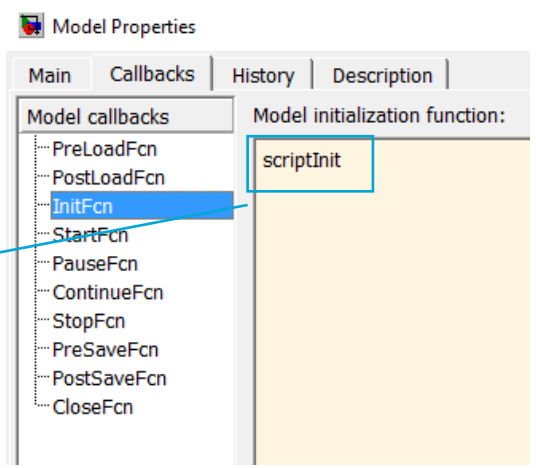


Linking the *.m file to the Simulink model using callbacks





Name of the *.m script

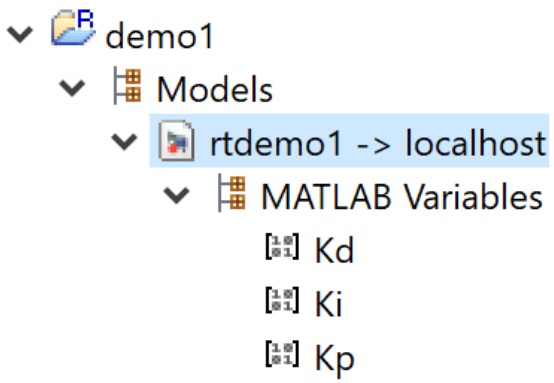


TU Delft

Complete workflow – Parameters, variables & signals

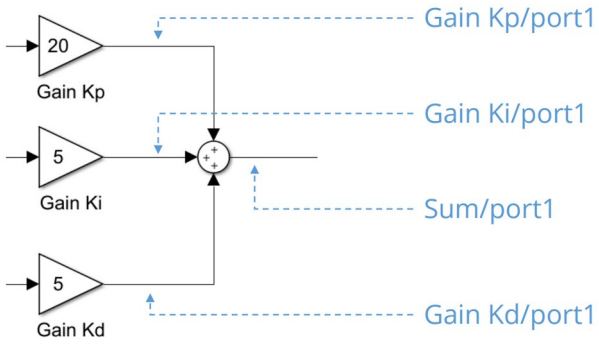
Parameters – Declaring variables in M-Files

1. Declaring the script in the « InitFcn» callback allows the call of the script when running the Simulink model offline (« Play » in Simulink).
2. Declaring the script in the « PreLoadFcn » callback allows the use of the script by RT-LAB during the « Build » process.
3. After building the model, the variables are visible in the Project explorer.

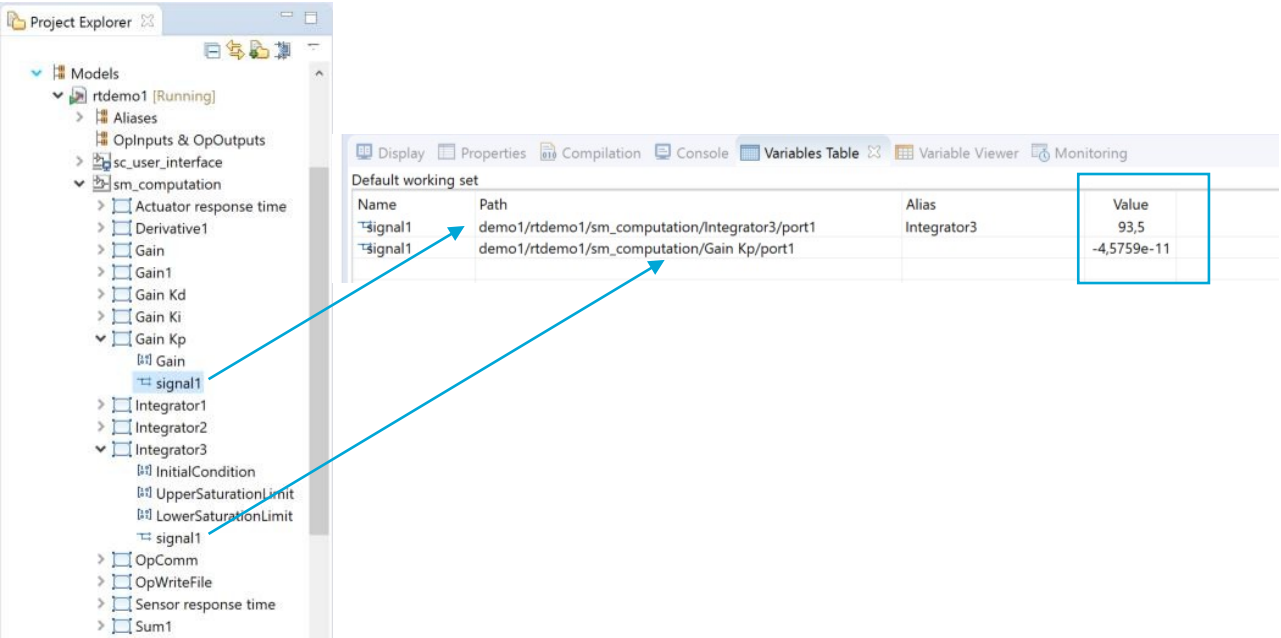


Signals – Definition

Signals are defined in Simulink as the outputs of blocks



Signal values can be monitored in the Variable Table

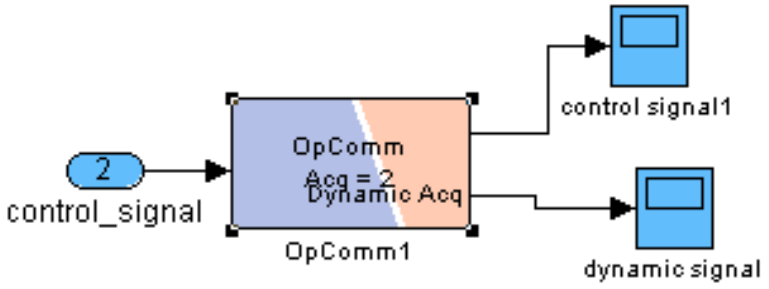


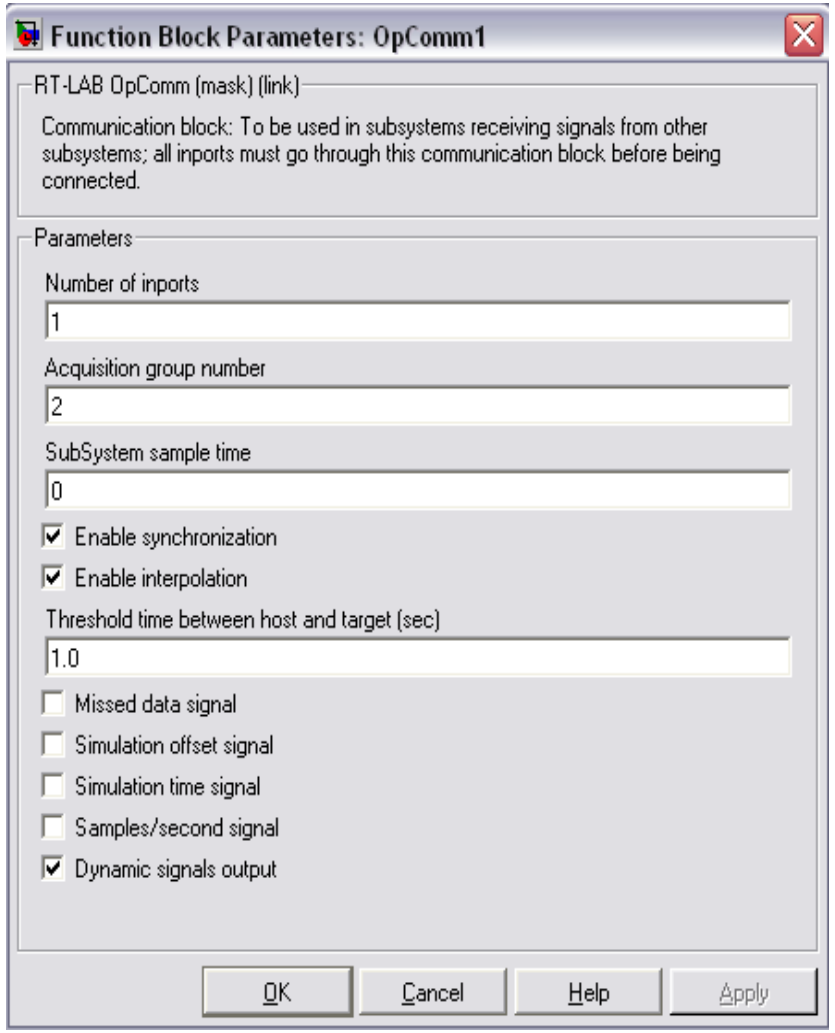
TU Delft

Complete workflow – Parameters, variables & signals

Signals – Dynamic signals

- RT-LAB allows the selection of signals for visualization to be done dynamically, at run time. This is called dynamic signal acquisition.
- To enable Dynamic Signals, in the OpComm block of the SC subsystem, simply check the Dynamic signals output. This will add an extra output to the block.



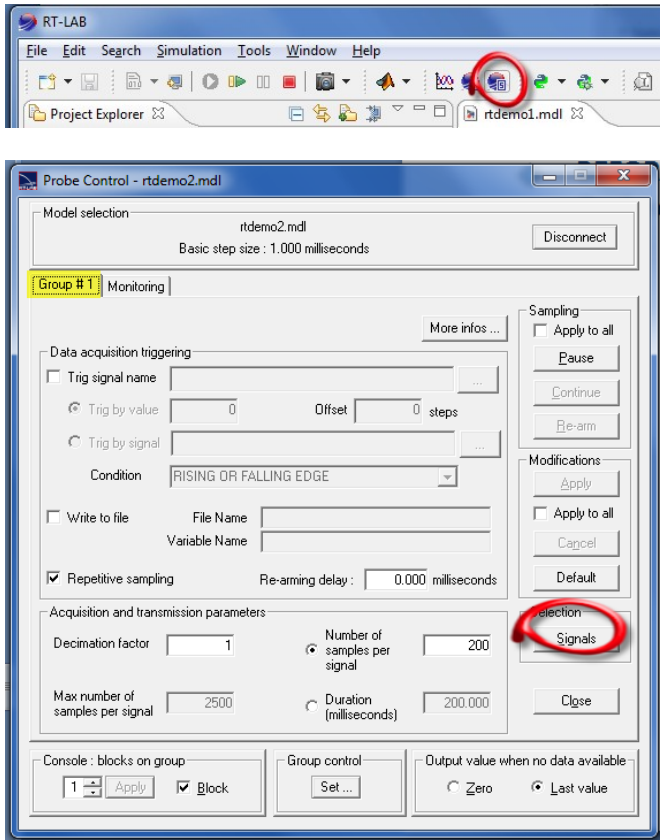


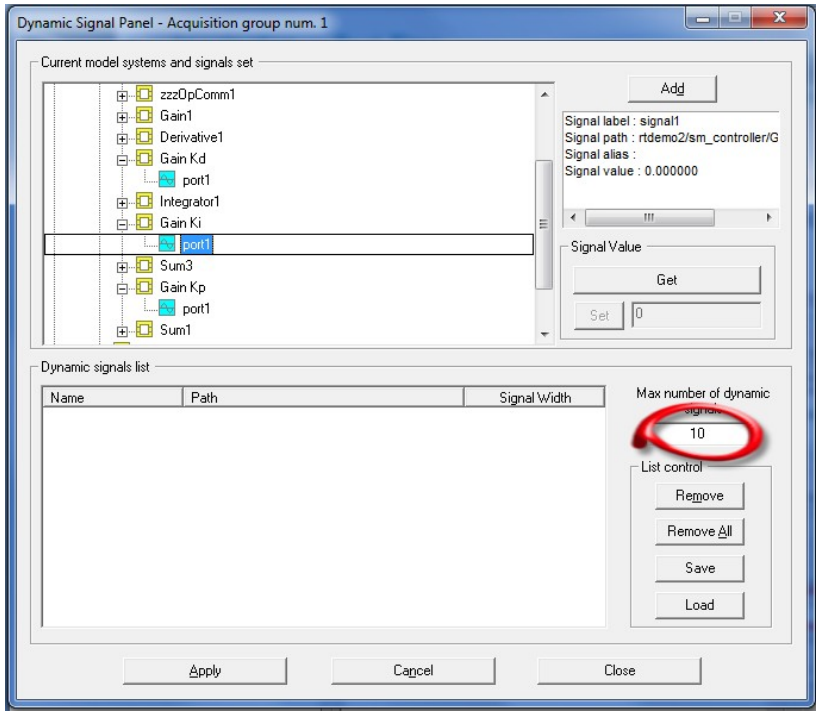
TU Delft

Complete workflow – Parameters, variables & signals

Signals – Dynamic signals

- Before loading the model, bring up the dynamic signals dialog from the Probe Control panel.
- Set the maximum number of dynamic signals you need. This is necessary before loading for the memory allocation of the acquisition buffer.



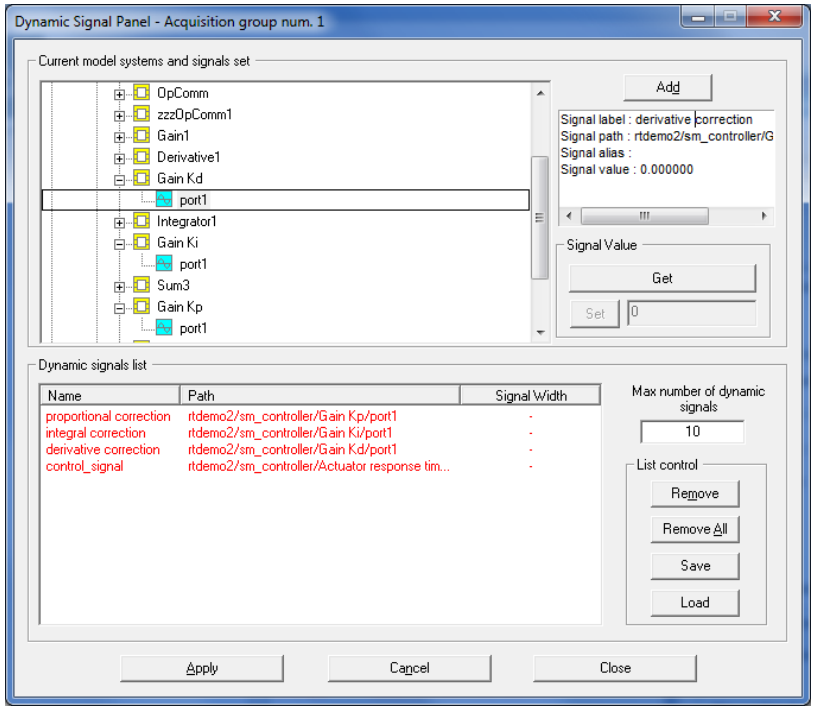


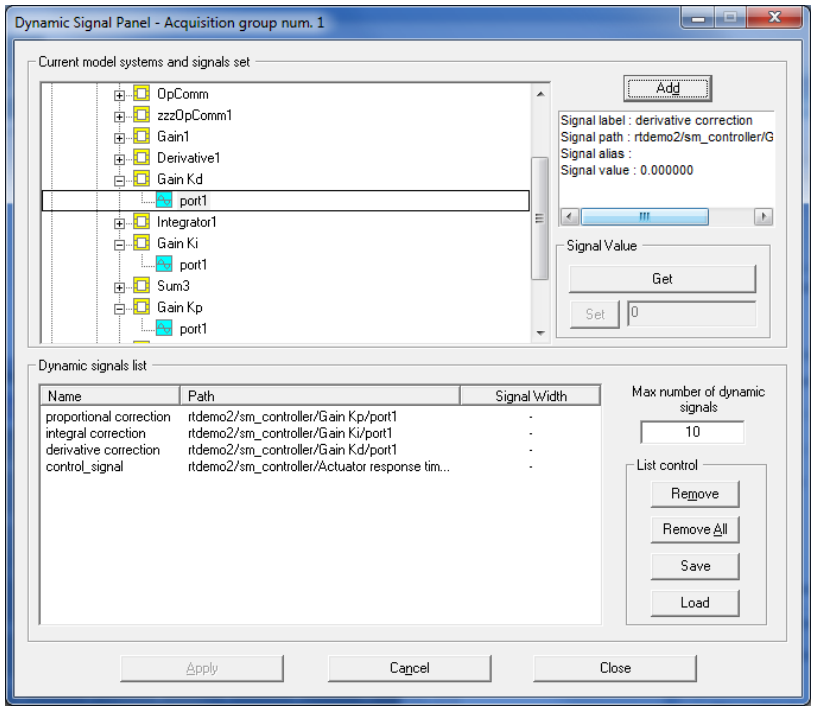
TU Delft

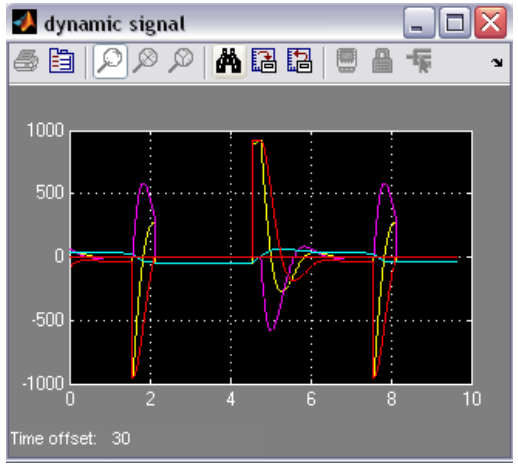
Complete workflow – Parameters, variables & signals

Signals – Dynamic signals

- During run-time, select the signal you want to visualize from the tree list.
- Press “Apply” and you will immediately see the selected signals in the Simulink console.







TU Delft

Bedankt voor uw aandacht

Dongyu Li