ALMA MATER STUDIORUM

UNIVERSITÀ DI BOLOGNA

Scuola di Ingegneria e Architettura

Corso di Laurea Magistrale in Ingegneria Meccanica

DIN

Dipartimento di Ingegneria Industriale

TESI DI LAUREA

in

Modeling and control of internal combustion engines
and hybrid propulsion systems

DEVELOPMENT OF A MODEL-IN-THE-LOOP SIMULATION ENVIRONMENT FOR VIRTUAL PRE-CALIBRATION OF CHARGE AND BOOST ENGINE CONTROLS

Relatore:
Ill.mo Prof. Ing. Nicolò Cavina

Correlatori:
Prof. Ing. Davide Moro
Prof. Ing. Enrico Corti
Ing. Paolo Scarpato
Ing. Klajdi Mustafaj
Ing. Luca Venturoli

Candidato:
Alberto Cesare Barbon
0000952167

0.1

ABSTRACT

The internal combustion engine development process is associated with high costs and longtime due to its multi-domain nature.

The simultaneous involvement of disciplines like mechanics, gas dynamics, and chemistry makes testing difficult for control purpose.

X-in-the-loop testing helps reducing research and development costs while significantly improving the time-to-market reactivity.

Model-in-the-loop is the early fully virtual step for achieving faster and safer hardware operations.

0.2

Acknowledgements

The project this dissertation will focus on, would not have been possible without the support of many people.

I wish first to thank my academic tutor Ill.mo Prof. Ing. Nicolò Cavina for his invaluable supervision, support, and patience throughout the past months and for his words of encouragement.

The internship that was carried out would not have been as engaging and as helpful without the support of Prof. Ing. Claudio Forte and NAIS host company, to whom I am grateful for giving me physical hospitality, kindness, and valuable advice both technical and personal.

I thank Automobili Lamborghini for allowing the thesis to be carried out. Ing. Paolo Scarpato, Ing. Klajdi Mustafaj, and Ing. Luca Venturoli not only have they been great teachers but also the best colleagues in the past few months. Their lessons and their expertise let a huge improvement in my technical skills and my know-how.

Table of contents

List of abbreviations

MiL	Model-in-the-loop
SiL	Software-in-the-loop
PiL	Processor-in-the-loop
HiL	Hardware-in-the-loop
CAN	Controller area network
ECU	Engine control unit
EGR	Exhaust gas recirculation
MAP	Manifold absolute pressure (sensor)
MAF	Mass air flow (sensor)
GPF	Gasoline particulate filter
VVT	Variable valve timing (actuator)
HFM	Hot film (air) mass (sensor)
FRM	Fast running model
MFB50	50% of fuel mass burnt
DoE	Design of experiments
RMSE	Root mean square error

Table of figures

1

Introduction

X-in-the-loop nomenclature groups all the testing activities through which parts of a system are joined together during the development process. Some systems can be virtually reproduced, others can be physically implemented.

Typically, there are two different environments either virtual or physical: the plant and the system. The plant consists of the domain to which the reaction of the system is gauged. The system pursues the target it is supposed to achieve according to given stimulus from the plant.

Loop testing is subdivided according to its hardware components:

MiL – Model-in-the-loop. The plant and the controller are both implemented virtually to verify the proper work of the control logic.

SiL – Software-in-the-loop. Once the model works in the previous stage, software code is generated thought integrated compilers to replace the controller and the plant block.

PiL – Processor-in-the-loop. The code is flashed into an embedded processor and runs in a closed-loop chain with the simulated plant. This step helps to identify processor characteristics bottleneck and glitches.

HiL – Hardware-in-the-loop. The plant is compiled and simulated thought a real-time computer. This step allows checking also communication protocols such as the CAN bus. The delay introduced by physical communication, for instance, can make the controller unstable if it is not correctly captured in the previous phases.

The dissertation will focus on the early development stages of the control logic of a new high-performance turbocharged internal combustion engine. With that purpose, a model-in-the-loop environment was built for testing and validating the engine control unit (ECU) operations.

Therefore, the plant is the virtual engine model whereas the system is the Simulink translation of some selected ECU software modules for this specific application.

Finally, some tuning operations on the engine control unit software will be described highlighting the usage of the simulation environment for faster and safer testing bench campaigns.

2

Charge control literature analysis

In this chapter the software architecture of the engine control unit (ECU) will be described. The dissertation will focus on two main engine areas: air charge and boost control. The first one is based on the reference paper provided by the ECU supplier and the second on a company owned control scheme.

2.1 Principles of charge determination

2.1.1 Definitions

According to the supplier definition, the air charge is the mass of the air that is introduced in the combustion chamber after the intake valve closing. Through lambda parameter, the quantity of the fuel that burns during the cylinder cycle is defined.

Accurate and precise air charge determination deals with:

- Maximum possible torque
- Combustion repeatability (low cycle to cycle variability)
- Exhaust gasses aftertreatment
- Engine efficiency

In particular, the ECU supplier considers the relative charge, which is the actual air charge compared to the air mass that would be present in the cylinder according to standard conditions. The agreement that the software uses, refers to:

- $p_0 = 1013hPa$
- $T_0 = 273K$

The determination of the relative charge starts from the definition:

$$Ch_{rel} = \frac{m_{cylinder}}{m_{STD}} \cdot 100 \qquad 2.1.1.2$$

According to the perfect gas law:

$$m_{cylinder} = \frac{p_{cyl}\, V_{eff}}{R\ T_{cc}} \qquad 2.1.1.3$$

$$m_{STD} = \frac{p_0\, V_{cyl}}{R\, T_0} \qquad 2.1.1.4$$

Considering the cylinder pressure as the sum of the air and the inert gasses partial pressure (p_{EGR}), the 2.1.1.5 definition is valid:

$$Ch_{rel} = \frac{p_{cyl} - p_{EGR}}{p_0} \frac{T_0}{T_{cc}} \frac{V_{eff}}{V_{cyl}}\ 100 \qquad 2.1.1.5$$

2.1.2 Charge components

To precisely compute the reactive oxygen quantity, the control software must account for the charge composition. In particular, the software supplier differentiates the inert gasses contribution depending on the origin:

- Reaspirative exhaust gas in the intake manifold from the cylinder
- Internal residual gas in the cylinder from the previous combustion
- External addition of inert gas due to high/low-pressure EGR devices
- Internal residual gasses due to camshaft timing adjustment

The external addition of inert gas depends on the air/fuel ratio of the previous combustion:

- $\lambda = 1$, the exhaust composition consists only of inert gas
- $\lambda < 1$, the exhaust composition contains unburnt fuel
- $\lambda > 1$, the exhaust gas contains unreacted oxygen

2.1.3 Charge determination

The charge determination control must:

- Ensure precise and accurate determination
- Obtain coherent results regardless the sensor type or failure

- Ensure simple calibration process
- Diagnose faults
- Keep reasonable computational effort and a low number of sensors

The charge estimation can be provided either via sensor readings or physical models. The engine sensors from which the charge is typically computed are:

- Anemometer mass flow sensor (MAF)
- Manifold pressure sensor (MAP)

The physical model is typically based on the throttle valve model via the isentropic orifice flow theory.

Anemometer air mass determination

The sensor directly provides the measurement of the air mass flowing into the intake manifold with a heated element. If the element is a wire, its resistivity is measured and correlated to the mass flow. If the element is a film, the temperature difference across the heating element is considered for the estimation.

Because of the nature of the measurement, the response is not reliable when fast transients occur due to the thermal capacity of the sensor and the storage behavior of the intake volumes. The maximum accuracy is reached on stationary or on slow dynamic runs.

The cylinder relative charge calculation is represented in *figure 1*:

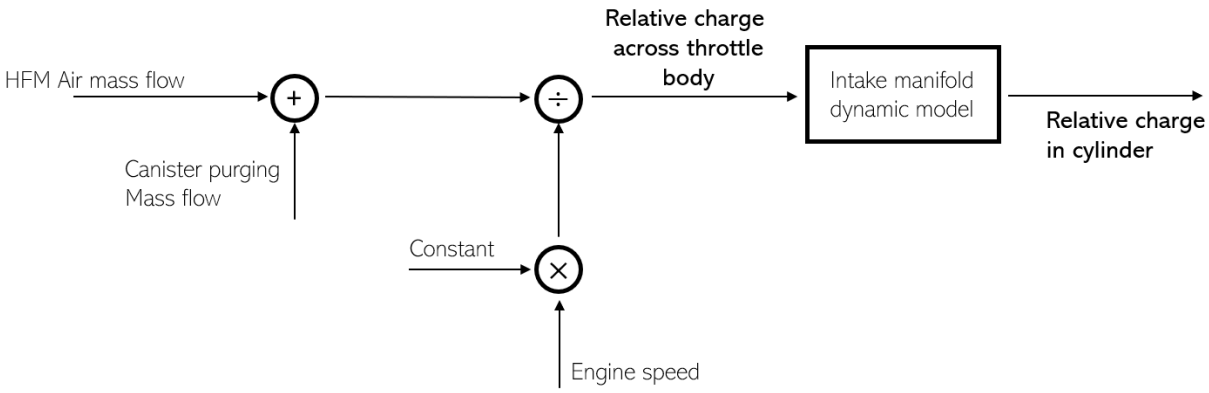


Figure 1, relative charge calculation

The mass flow from canister purge operation is added over the air mass flow signal coming from the sensor, obtaining the total mass flow flowing into the intake manifold. The constant term accounts for engine displacement, and the intake manifold dynamic model accounts for the storage behavior.

Manifold pressure sensor air mass determination

The charge calculation through manifold pressure sensor takes place considering that the pressures in the intake manifold and in the cylinder are equal when the intake valve is about to close.

Considering the 2.1.1.5, the relative charge formula now contains the measured manifold pressure and the modeled inert gas partial pressure:

$$Ch_{rel} = \frac{p_{man} - p_{iEGR}}{p_0} \frac{T_0}{T_{cc}} \frac{V_{eff}}{V_{cyl}} 100 = (p_{man} - p_{iEGR}) \cdot K \text{ [1]} \qquad 2.1.3.1$$

The relation is represented graphically in *figure 2*:

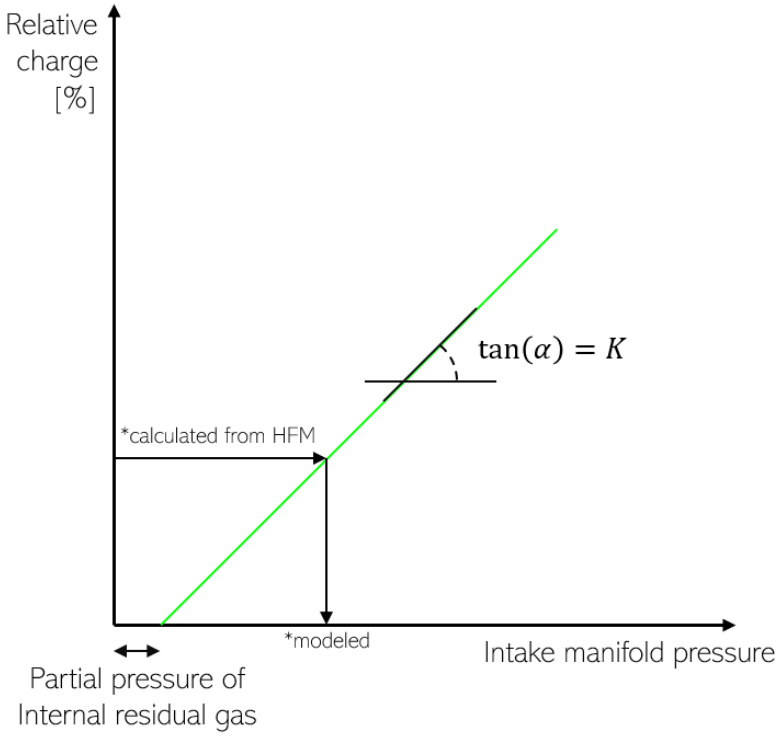


Figure 2, pressure to relative charge relation

Throttle valve model charge determination

The throttle valve model is based on the isentropic nozzle flow physical relation. It depends on the gas properties, on the ratio between the downstream and the upstream blade pressure ratio, on the cross-sectional area, and on the discharge coefficient.

[1] Conversion factor from pressure to relative charge

The formula which the model relies on follows:

$$\dot{m} = A(\alpha)\,\rho_{US}\,C_D\sqrt{R\,T_{US}\,\frac{2\gamma}{\gamma-1}\left[\beta^{\frac{2}{\gamma}} - \beta^{\frac{\gamma+1}{\gamma}}\right]} \qquad 2.1.3.2$$

β: ratio between downstream and upstream pressure

γ: air specific heat capacity

C_D: discharge coefficient

$A(\alpha)$: cross-sectional area as function of throttle blade angle

Figure 3 represents graphically the 2.1.1.7 equation.

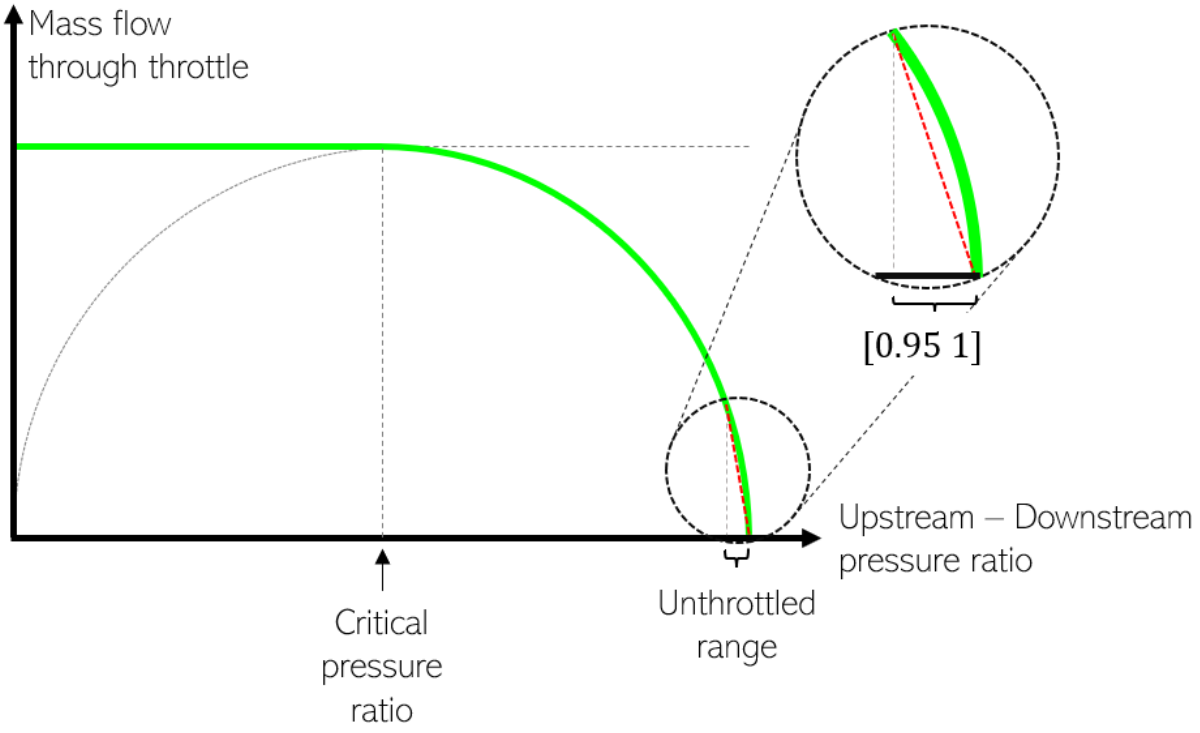


Figure 3, isentropic orifice mass flow

As the figure shows, when the pressure ratio drops below a critical threshold (0.528 for fresh air), the mass flow assumes a constant value regardless the ratio. The mass flow in that range reaches the sonic conditions and depends on the cross-sectional area and on the gas properties only.

The figure highlights also the unthrottled zone. When the pressure ratio goes from 0.95 to 1, the gradient of the curve tends to be infinite. Considering that the software runs with discrete steps, the calculation is numerical unstable due to large mass flow changes compared to small pressure ratio variations. The control software substitutes a straight line in the parabola keeping large the running step and low the computational effort.

2.1.4 Combustion chamber temperature model

The air heats up during its path from the air filter into the combustion chamber due to charger compression and residence time into the engine bay. The temperature of the gas in the combustion chamber at intake valve closing, required by the charge determination, cannot be, however, directly measured.

The control scheme provides a model for the temperature determination that relies on coolant and intake air temperature sensors.

In the simplest case of naturally aspirated engines, the air is heated up due to the heat transferred from the wall to the airstream. The heating depends on the temperature difference between the air and the intake walls. The convection coefficient depends on the air speed (the slower the air flows, the higher is the transferred heat). The implementation of the physical behavior is described with the 2.1.4.1 and the 2.1.4.2 equations.

$$T_{cc} = T_{intake} + \Delta T \qquad 2.1.4.1$$

$$\Delta T = (T_{wall} - T_{intake}) \cdot \frac{\alpha_w \cdot \Delta\tau}{C \cdot m_{int}} \qquad 2.1.4.2$$

T_{intake}: temperature of the intake air

T_{wall}: intake port wall temperature

α_w: thermal convective coefficient

$\Delta\tau$: residence time of air mass

C: air specific heat capacity

m_s: considered air mass

Moreover, the warm-up phase must be carefully considered as the engine coolant temperature does not provide a wall temperature with enough accuracy.

2.1.5 Functionalities of the air path control

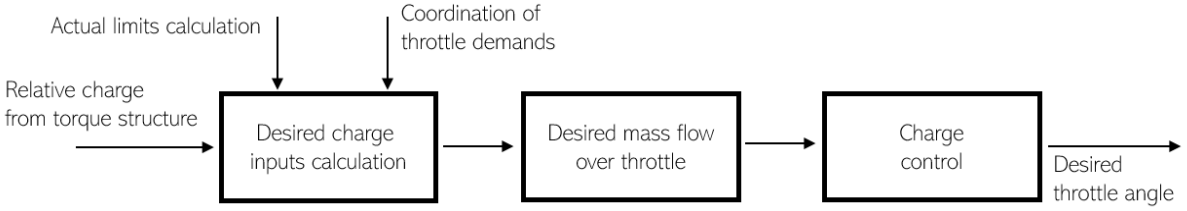


Figure 4, air control functionalities

The charge control system converts the desired relative charge coming from the torque structure into a corresponding throttle angle and, eventually, wastegate opening setpoint.

The first square from the left of *figure 4* includes the desired relative charge calculations from the torque structure. The coordination and the prioritization between the different torque requests take place.

The second block converts the torque request into the target mass flow considering all the contributions to manifold charge such as the canister purging and inert gasses recirculation flow. The deviation between the target and the actual charge comes from the charge controller correcting the module output.

The third block on the right of *figure 4* contains the inverse throttle valve model that outputs the target throttle angle which is filtered, coordinated, and limited.

3

Boost control literature analysis

Boost control runs in the ECU environment independently of the charge control. The main software is linked to boost module through a wrapper that resolves the dependencies between the two systems. The same signals are often modeled in both the controls allowing precise tuning of each system according to the different performances targets.

3.1 Principles of boost control

Figure 5 summarizes the logic behind the boost control implementation. On the left are represented the physical inputs coming from sensors or other software models. Boost control uses these inputs to feed models for requested signals calculations. Closed-loop and open-loop controls are the core of the scheme, the systems output a wastegate actuator setpoint, which is coordinated in the prioritization module. The details on the individual functions will be described in the following chapters.

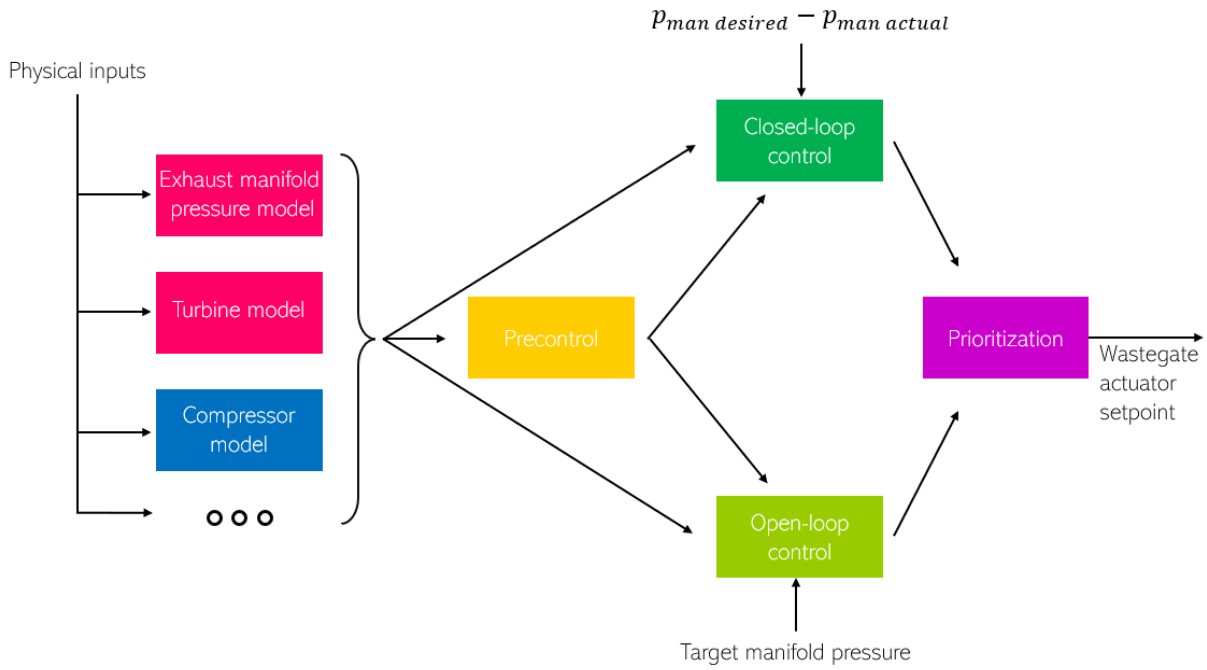


Figure 5, boost control functionalities

3.1.1 Intake path model

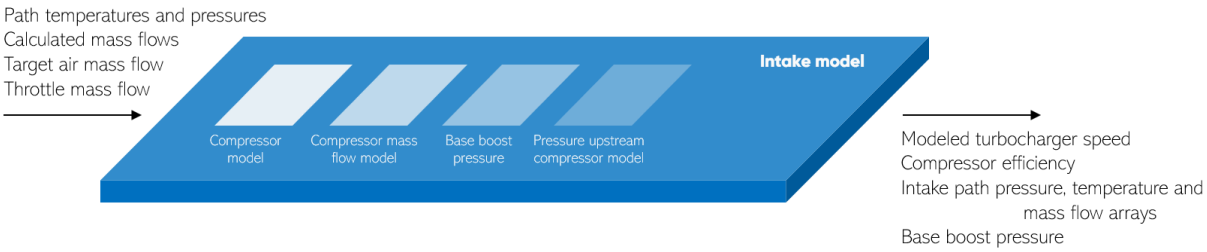


Figure 6, intake path functionalities

Intake path model takes most of its inputs from sensors or physical modeled signals. It uses the air heat capacity (function of the air temperature), the air path temperatures (intercooler, manifold, and compressor), and ambient conditions. The module uses the modelled and the targeted air mass flows computed with other systems.

Base boost pressure output is involved in:

- Closed-loop control
- Turbocharger speed modeling
- Component protection factors
- Modeled efficiency
- Open-loop control

3.1.2 Exhaust path model

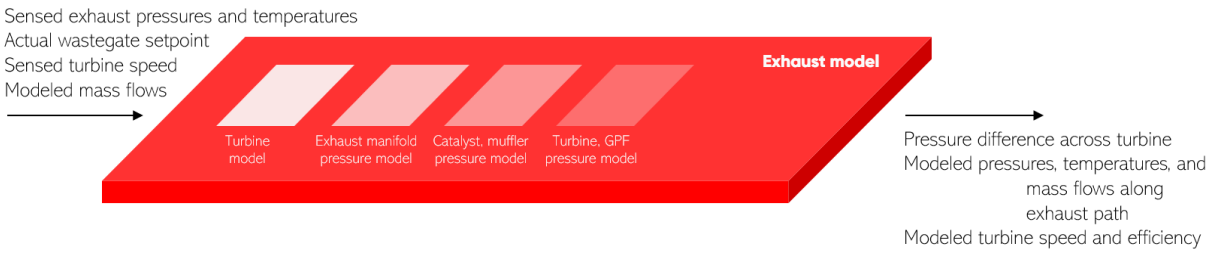


Figure 7, exhaust model functionalities

Exhaust path model includes the turbine physical model, the exhaust manifold pressure model and four separate modules for each exhaust elements (turbine, catalyst, muffles and GPF).

Like the intake path model, most of the inputs come from physical readings. In this case, the involved signals are:

- Turbine temperature and pressure
- Exhaust heat capacity
- Exhaust mass flow
- Catalyst temperature and pressure

The outputs are used mainly in open loop wastegate control and in component protection limitations.

3.1.3 Pre-control

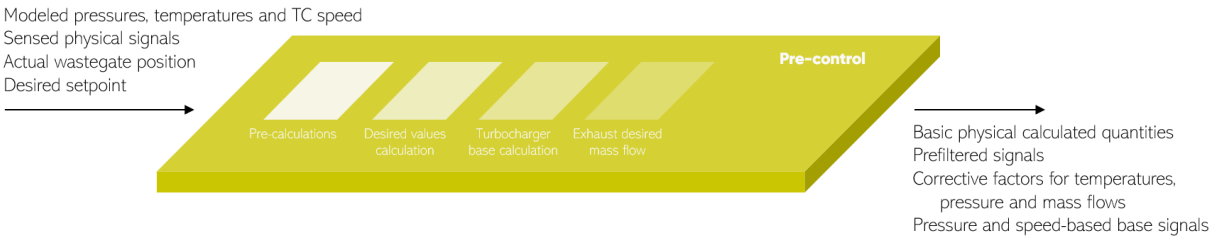


Figure 8, precontrol functionalities

The system is a general-purpose module that manipulates inputs to provide requested quantities for all the other blocks. For example, it updates the actual quantities for the air heat capacity via temperature interpolation and supplies correction factors for pressures and mass-flows.

The pre-control module uses both model and map-based approach.

3.1.4 Open-loop control

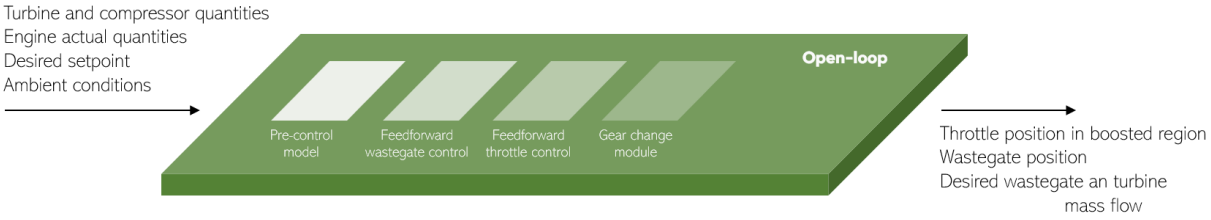


Figure 9, open-loop functionalities

Open-loop block generates the wastegate actuator setpoint to reach the target boost pressure without the feedback correction path. It relies on the pressure across the air path and the turbocharger, its actual efficiency, and the computed mass flows.

It is possible to switch between either map-based or model-based calculations. The first depends upon the conversion of the actual engine speed and the actual upstream compressor pressure into the desired wastegate actuator position. The model-based approach relies on the physical model of the turbine valve flow.

Furthermore, the module supplies the control of the throttle valve in boosted engine operating points, and the position of the wastegate during gear changing.

The open-loop control response to target changes is fast as it does not depend on the system measurement. If the system is well-calibrated, it allows to reach the target conditions without any other action but with a very long transient.

Typically, when the system reaches the target, the driving signal consists mainly of the open-loop control contribution.

3.1.5 Closed-loop control

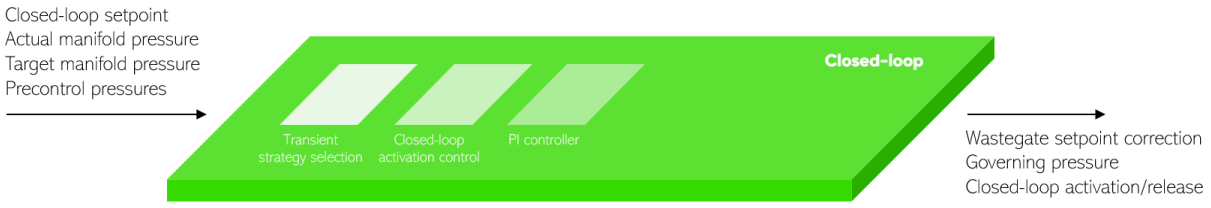


Figure 10, closed-loop functionalities

Closed-loop control generates the additive correction for the wastegate actuator setpoint to reach the desired boost pressure target with the shortest possible transient. Its output is based upon the difference between the target and the actual manifold pressure.

Unlike the open-loop control, the time response is typically slower as the controller relies also on the integral of the control deviation.

When the difference tends to zero and the target is almost reached, the wastegate setpoint contribution is typically negligible with respect to the open-loop output.

3.1.5 Prioritization control

Figure 11, prioritization functionalities

The control prioritizes and combines the wastegate requests coming from the boost systems according to its specific tuning. The module, indeed, includes the most common implementations and the required functions. For instance, it is possible to switch between variable geometry turbine and wastegate valve boost control depending on the specific engine tuning. Hence, it is possible to choose the electric or pneumatic actuated valve.

4

Simulink translation

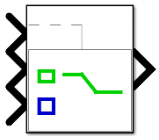
A substantial part of the activity consisted of reproducing the charge and boost ECU control systems in the Simulink environment according to reference papers. The task execution can be divided into three steps:

- Modeling
  The engine control software is translated into the Simulink block language maintaining the supplier structure with the highest possible accuracy.
- Single module validation
  Supply the modeled block is possible when a recorded engine run is available. With given inputs, a validated module must match the recorded equivalent outputs.
- Assembly validation
  Simulink models merging highlights the cumulative error when a signal is computed through multiple paths.

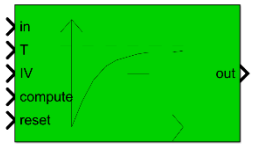
Modeling

The company provided a Simulink translation library to simplify the first step, which contains the fundamental Simulink blocks for quickly reproducing most of the desired ECU behaviors.

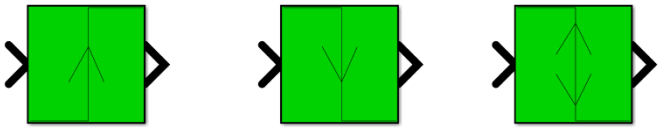
Mostly used blocks are described below.



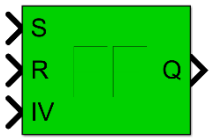
Switch, it works according to inverted Simulink's logic. When the first input is *true*, the last input passes. If the first input is *false*, the second input passes.



Lowpass filter cuts the desired frequency from the input signal according to the given time constant provided at the *T* port. *Compute* logic port actuates the filter and *reset* logic port resets the filter to the initial value provided at the *IV* port.

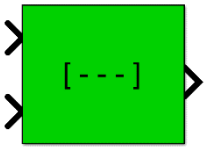


From the left, Edge Rising, Edge Falling, and Edge Bi. These blocks output a *true* logic signal with one simulation step duration when the input respectively rises, falls, or both of behaviors. It reproduces the interrupt microcontroller concept.



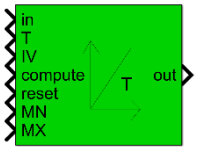
Flip Flop memory reproduces the elementary sequential circuit. It is the most basic memory circuit. Truth logic table follows:

S	R	Q	State
0	0	Memory state	No change
0	1	0	Reset to IV
1	0	1	Set
1	1	-	Invalid



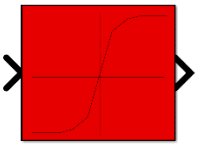
Array block reproduces the behavior of the ECU's array logic. In write configuration, it writes the value of the first input port on the memorized array at the second input port index. In read configuration, it reads the value of the first input port array at the second input port index.

The block, however, is not able to completely reproduce the ECU behavior because it is often needed to write on existent arrays. Many workarounds were implemented during the modeling phase to reproduce the correct assignment.



Integrator block integrates the input according to the given time constant provided at the *T* port. It can be actuated or deactivated by the *compute* logic port, and it can be reset to the initial value by the logic signal to *reset* port to the value provided with the *IV* port. The output can be limited through *MN* and *MX* ports.

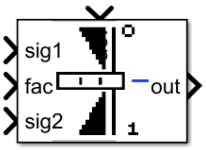
GetBit block outputs the binary value of the decimal number provided at the *CW* port at the index from *bit* port.



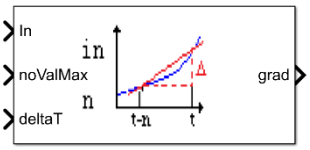
1D map consists of a one-dimensional Simulink lookup table block that interpolates the input value according to calibration specification.

2D map consists of a two-dimensional Simulink lookup table that interpolates the input values according to calibration specification. The interpolating matrix is transposed to be compliant with the ECU software.

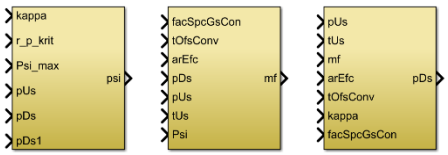
Some base blocks were added to the library during the boost modeling process:



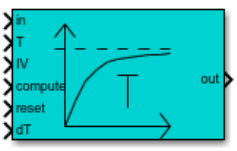
Mixer outputs the combination between the two inputs according to the *fac* port by the relation $out = fac \cdot sig1 + (1 - fac) \cdot sig2$.



Get gradient calculates the mean gradient of the signal provided at the *In* port, according to the timestep duration by *deltaT* port.



Throttle flow, the three blocks are the implementation of the forward and backward isentropic orifice equation calculating the mass flow through throttle body, the pressure downstream the blade, or the throttle angle with given conditions.



Lowpass Filter dT accounts for different simulation timesteps through the dT input port.

Single module validation

The process consists of recording ECU signals during a bench engine run or an on-board drive to apply trusted inputs to the Simulink model. Its output is then compared to the recorded raster highlighting which error the model is making.

Many comparing locations are introduced in the software by the ECU supplier allowing easier debugging process even with intermediate calculations.

There are two objects that are needed during the module validation process:

- Acquisition dataset
- Calibration dataset

The first dataset is provided in the form of a *.dat* text file containing the time series of all the required benchmark signals. A MATLAB script converts the text file into a workspace structure reproducing the following fields:

Workspace data object

Signal 1
x array
v array

Signal 2
x array
v array

Signal n
x array
v array

The *x array* contains the time sequence that matches the samples *v array.*

The acquisition data is retrieved in the Simulink model through a *from workspace* block which outputs the element from the *v array* corresponding to the actual simulation time.

Workspace calibration object

Calibration 1
x array
v array

Calibration 2
x array
y array
v array

Calibration n
x array
v array

The calibration dataset if provided in the form of a *.DCM* file from INCA export tool. A MATLAB script converts the calibration text file into a workspace structure as the acquisition data. This time, the calibration structure can have up to three fields in the case of 2D lookup table (two axis and one values vector).

When both the acquisition and the calibration datasets are loaded in the Simulink workspace, the model can run.

The Simulink solver settings must mimic the way in which the software is executed in the engine control unit even if its microcontroller and the desktop environment have different processor architectures.

The most suitable option available in the Simulink settings is the fixed step execution with discrete state solver, as reported in *figure 12*.

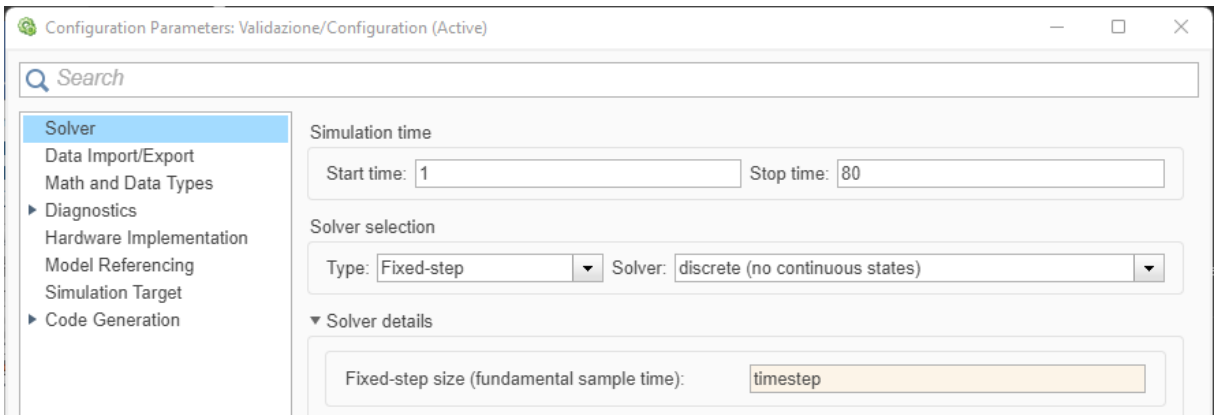


Figure 12, Simulink solver settings

The output of the validation process is a plot like the one represented in *Figure 13* that shows the model trace, the acquisition benchmark, and the percentage error between them according to *formula 4.0.0.1*:

$$e_{\%} = \frac{V_{modeled} - V_{dataset}}{V_{dataset}} \cdot 100 \qquad 4.0.0.1$$

Typically, if the offset between the model and the acquisition dataset is horizontal the model delays the signal due to different programming language implementation.

If the offset is vertical and constant, a summation error is likely responsible. On the other side, if the offset entity is proportional to the signal value (the higher the value, the higher the offset) a product or division operation is responsible for the error.

The agreement that this dissertation will keep in the validation chapters is highlighted in the *figure 13* where the upper plot draws the Simulink output with the blue line and the validation signal with the orange line. The lower plot reports the percentage error.

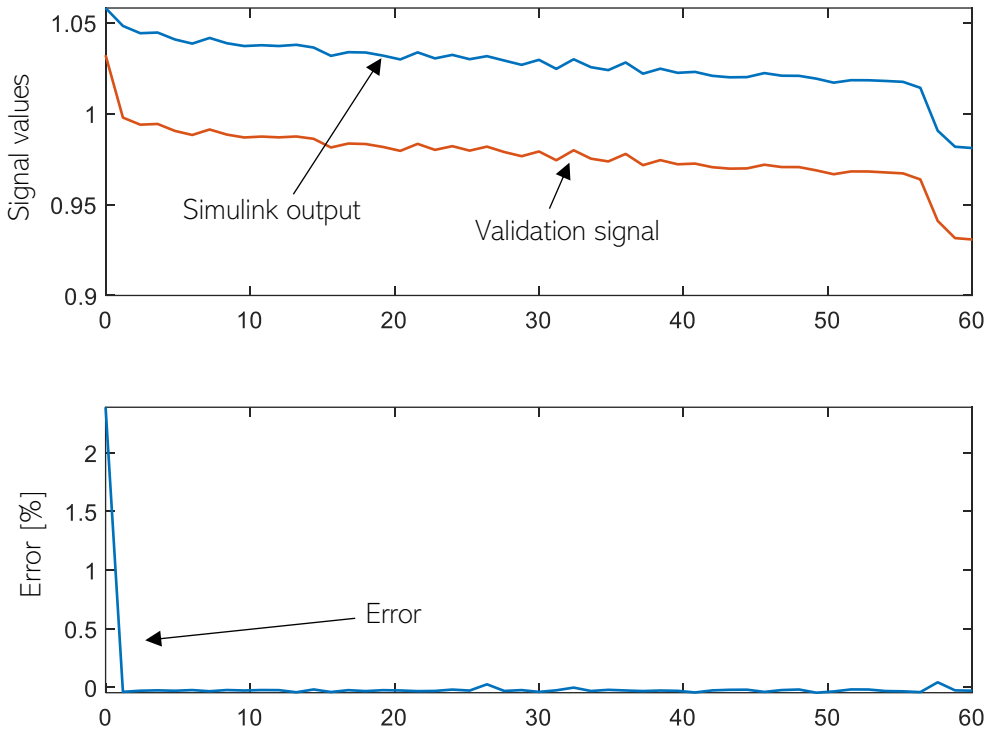


Figure 13, validation example

The error threshold under which the model is usually considered fine is 2% even if every signal must be carefully assessed.

For instance, in some cases the threshold can be higher due to the numerical instability with very small ($| < 10^{-1}|$) signal values. When the validation dataset signal is close to zero, the error can reach very high values even if the offset is low. The phenomenon can be explained by observing the denominator of the formula 4.0.0.1: the error tends to infinite when the acquisition value tends to zero.

In other situations, the error can reach very high peak values despite a good matching. *Figure 14* reports a qualitative example of huge error peaks even with a small delay.

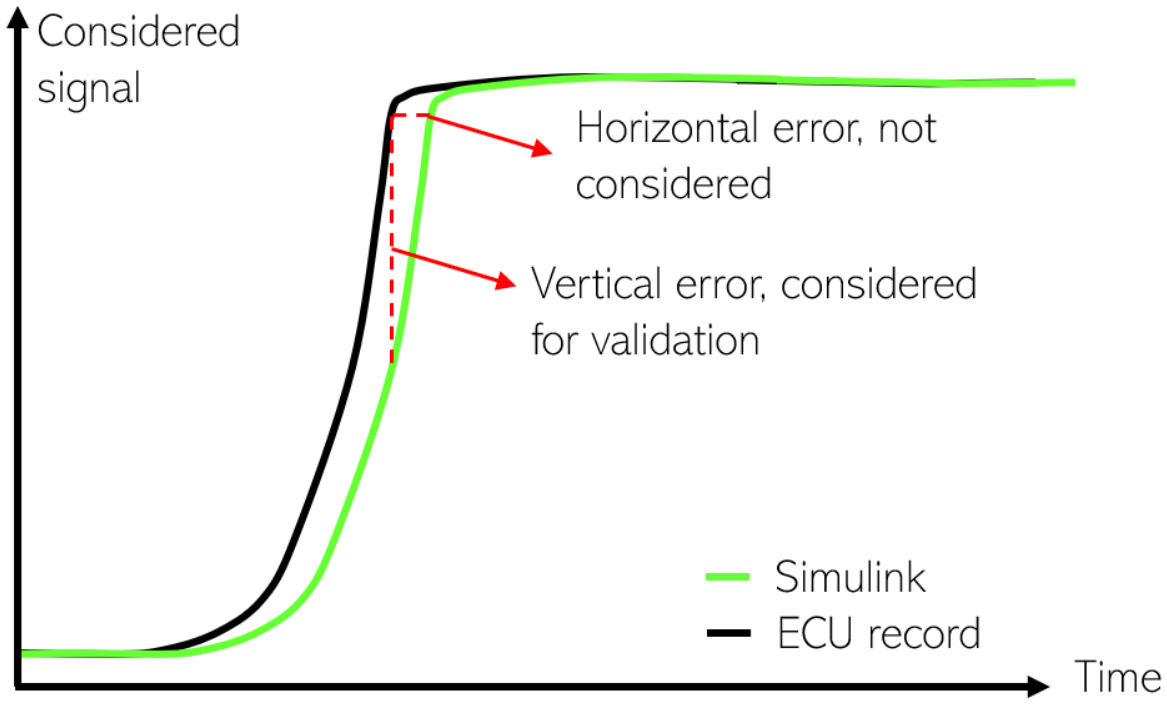


Figure 14, validation errors

Assembly validation

The last step of the Simulink modeling activity involves all the previously validated models. Joining blocks so that the outputs of one are the inputs of another highlights the cumulative errors.

For instance, the intake manifold pressure model relies on the charge exchange control block, which considers the pressure downstream exhaust valve calculation. From the goodness of the pressure signal is therefore possible to bench the accuracy of three Simulink implemented models. Both criteria and thresholds of single model validations are applied for cumulative observations.

Finally, the assembly model looks like the *figure 15* representation.

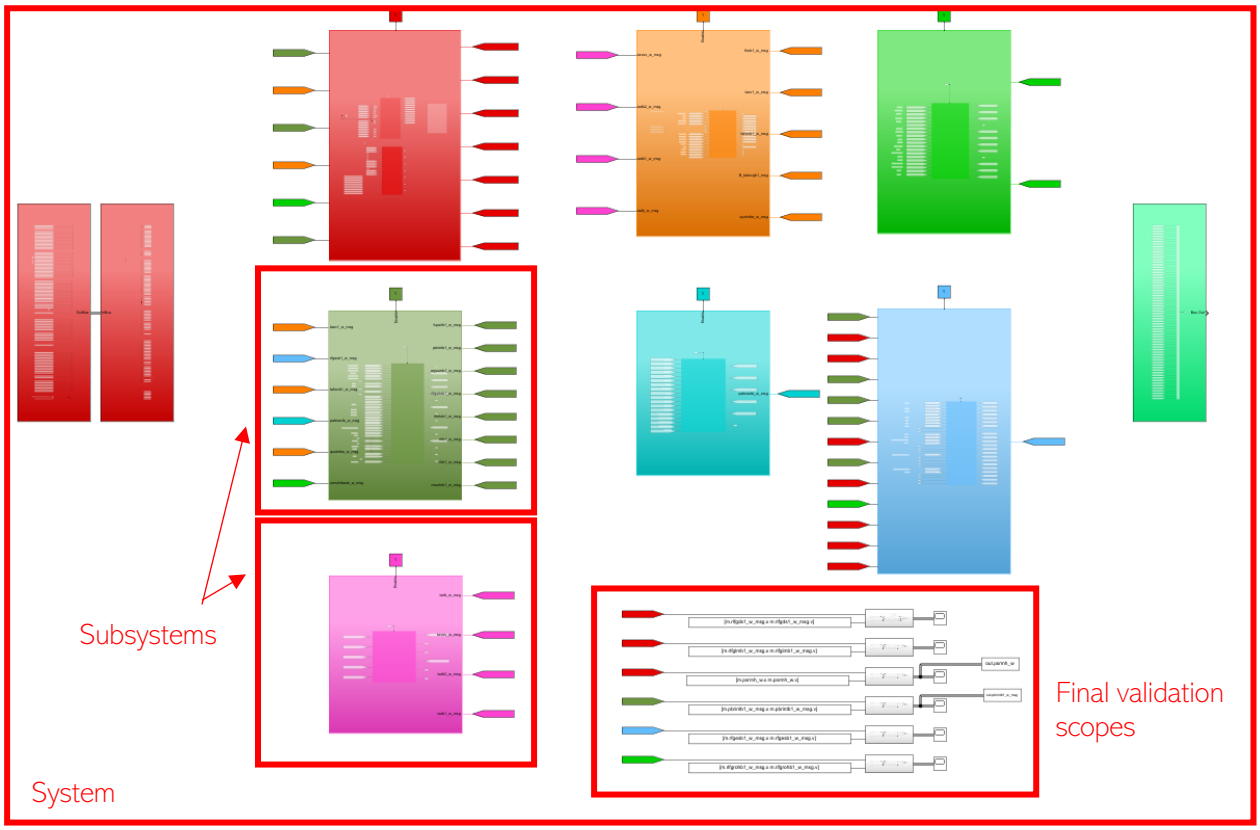


Figure 15, Simulink assembly example

Task management

The tasks outcome consists of a Simulink library containing all the modeled and the validated subsystems. Three organization levels are available according to software datasheet, control systems and function subsystems as represented in *figure 16*.

The library can be used directly from the Simulink library browser for easy and fast implementation through native *slblocks.m* MATLAB script as *figure 17* shows.

Models are accessible from two sources:

- Standalone subsystems

  Single function control block
- Connected system

  All the subsystems are inserted and connected one to each other. An input wrapper between the system and the Simulink workspace makes a custom implementation faster.

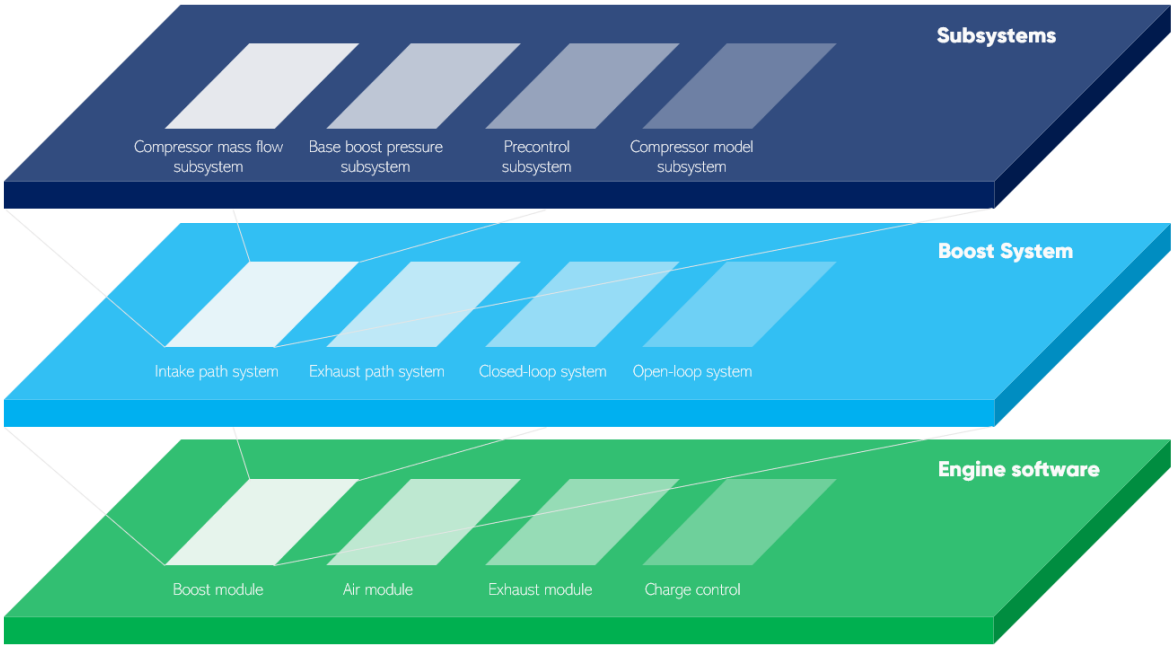


Figure 16, hierarchical structure

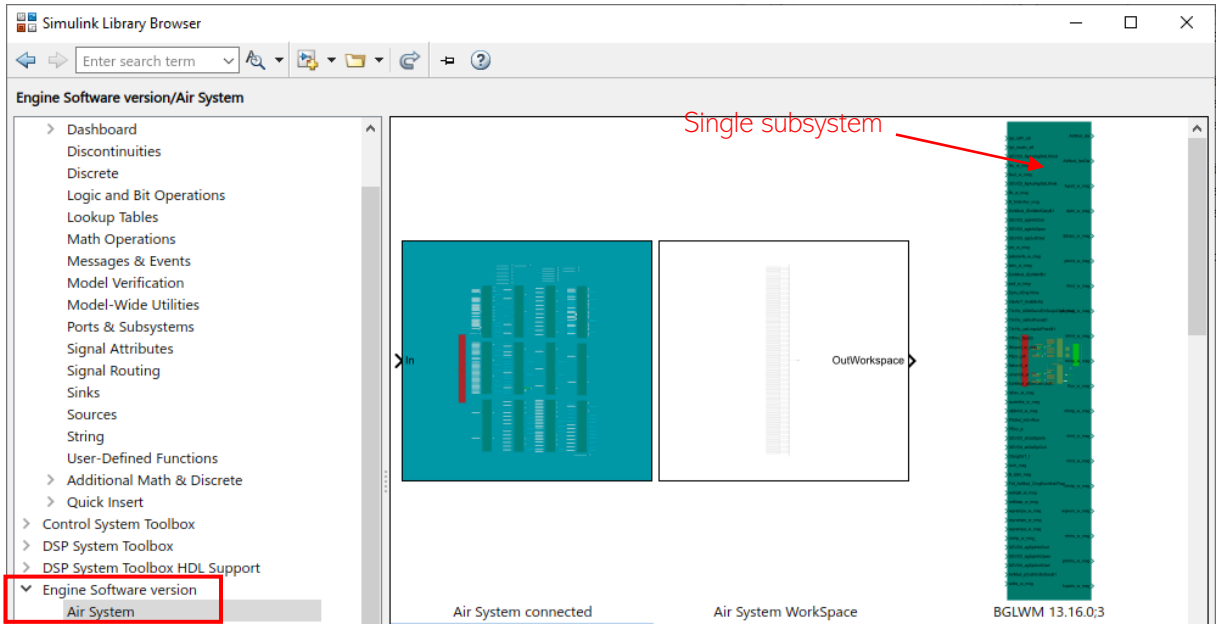


Figure 17, Simulink library browser

4.1 Charge task

4.1.1 Charge exchange through manifold pressure

The subsystem models the air charge from the manifold pressure sensor through the calculation of the inert residual gasses partial pressure and from the factor for pressure to load conversion. The same calculations are performed for the target and the subsequent

prediction of the air charge. Through actual VVT setpoint, the model considers the effect of the scavenging operating mode on the relative charge value.

Figure 18 represents the subsystem functions and dependencies.

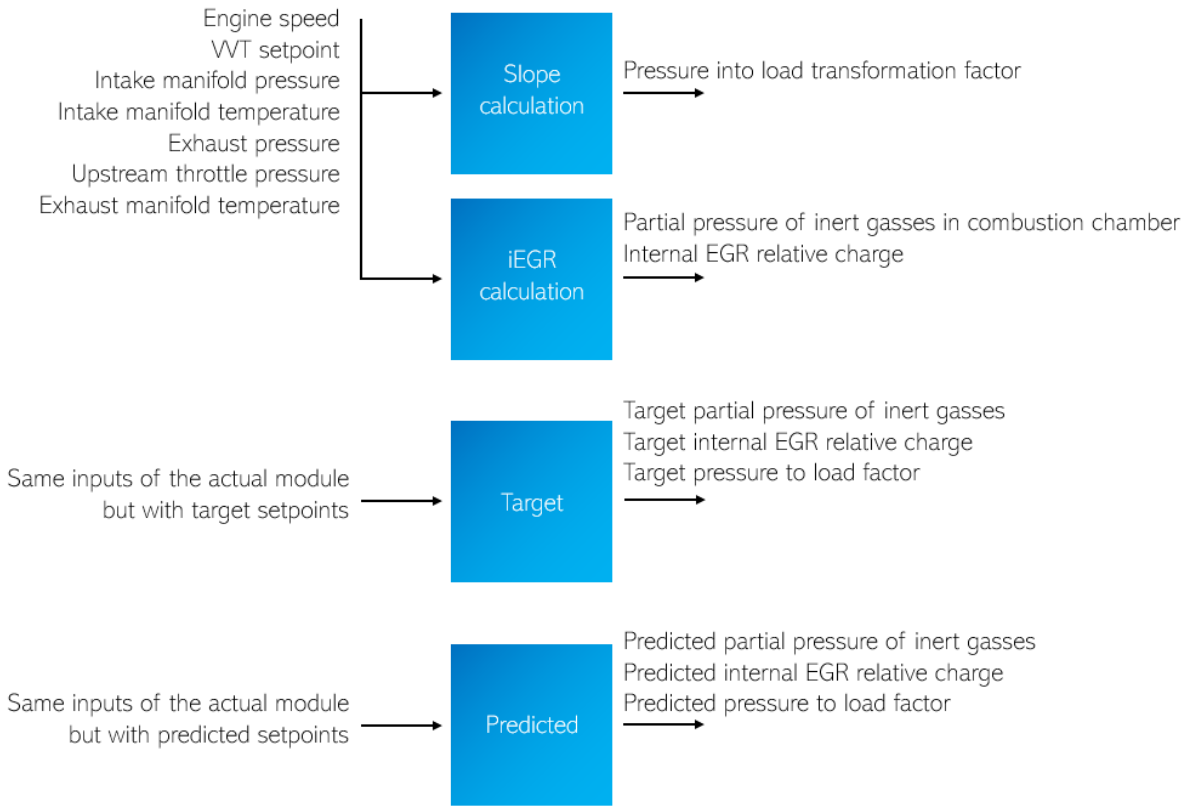


Figure 18, charge exchange blocks scheme

The conversion factor for pressure into charge correctly returns an error under 0.1% after the first steps. The behavior of the partial pressure of the residual gasses signals is worse. In this case, the initial transient is longer than the factor's one while still acceptable. The error spans under 1% which is sufficient for considering the subsystem implementation fine.

Most important signals validation is reported in *figure 19* and *figure 20*.

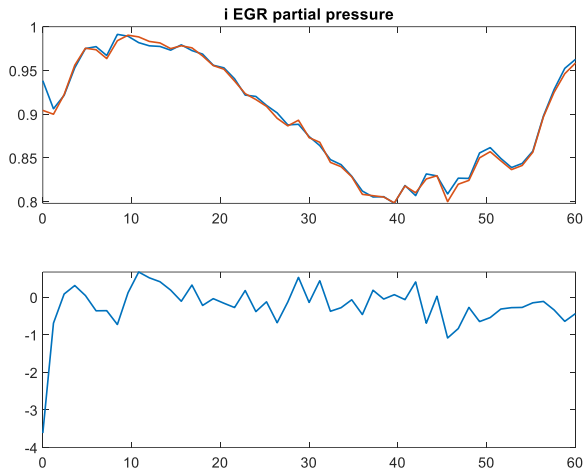


Figure 20, EGR validation

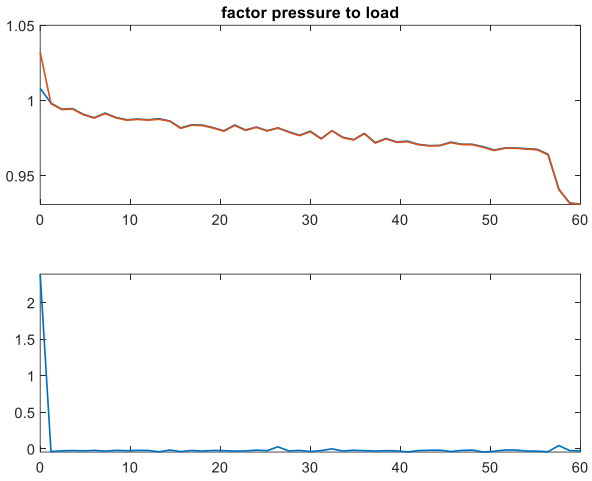


Figure 19, factor validation

4.1.2 Selection between manifold pressure or throttle blade model charge sensor

The subsystem switches between two different paths for computing the charge value: throttle body model or manifold pressure sensor. In both cases, it accounts for the storing effect of the manifold volume as represented in *figure 21.*

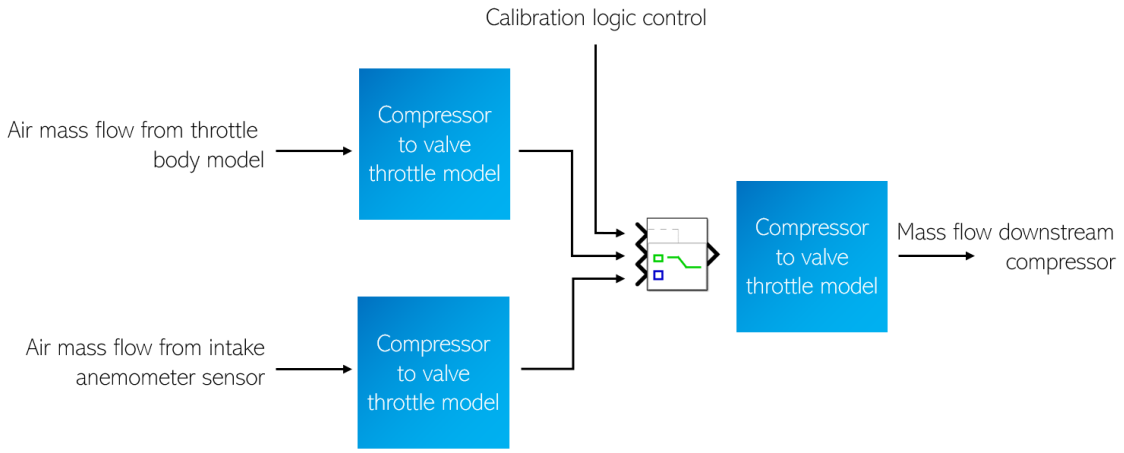


Figure 21, charge sensor selection blocks scheme

The main output, the mass flow downstream the compressor, is validated according to *figure 22*.

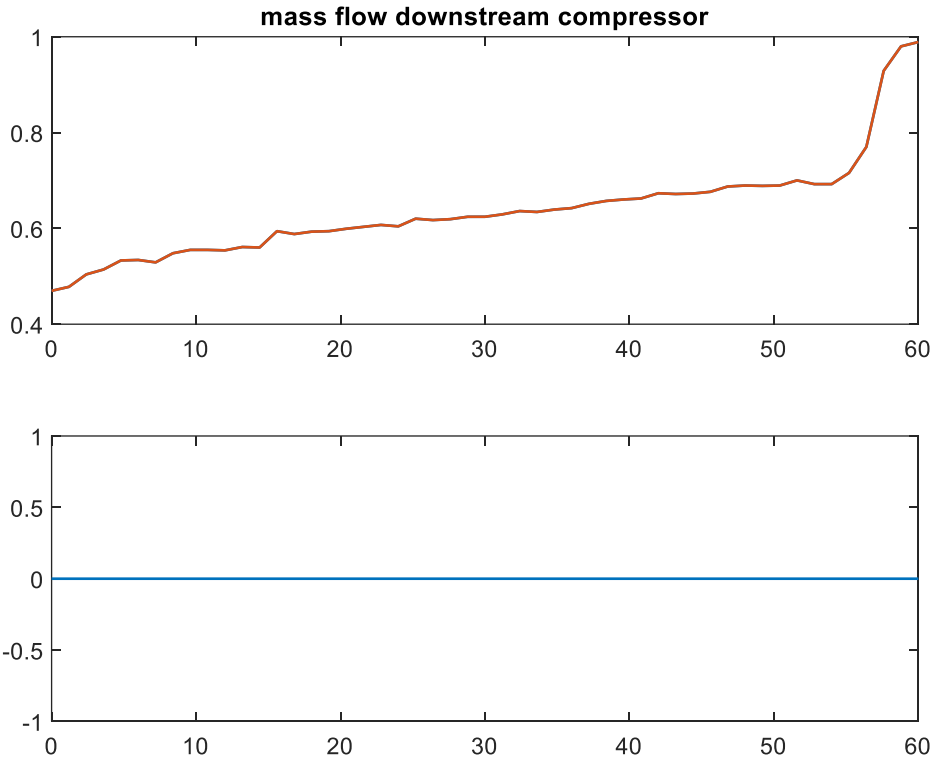


Figure 22, compressor mass flow validation

The error is constantly null because the model directly outputs one of its inputs according to calibration dataset.

4.1.3 Pressure downstream the exhaust valve model

The subsystem computes three different values: the exhaust manifold pressure, the target one, and the actual exhaust valve downstream pressure. The module can be calibrated either to be used as a physical model or as a calibration lookup table interpolating the exhaust mass flow and the actual boost pressure.

Figure 23 summarizes the calibration choice between the physics-based and the map-based exhaust model.

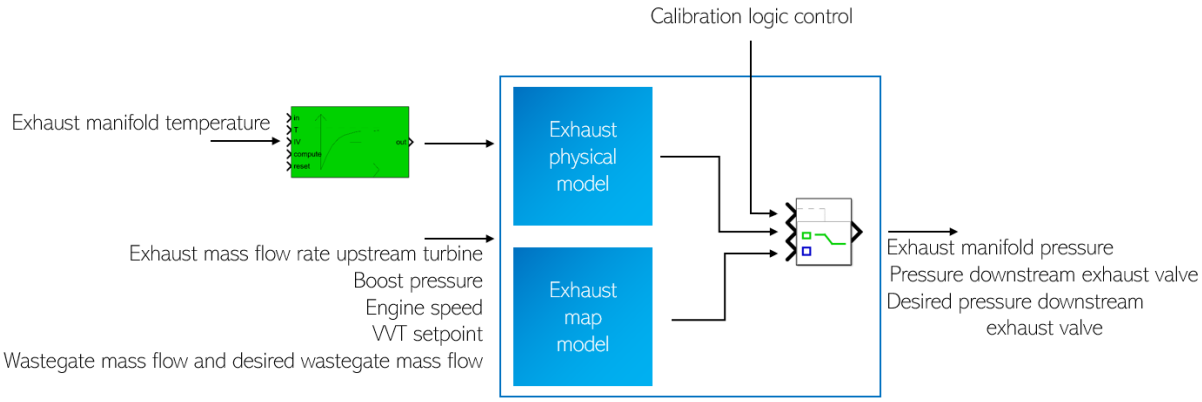


Figure 23, exhaust pressure blocks scheme

Validation of the three main signals is reported in *figure 24*, *figure 25*, and *figure 26*.

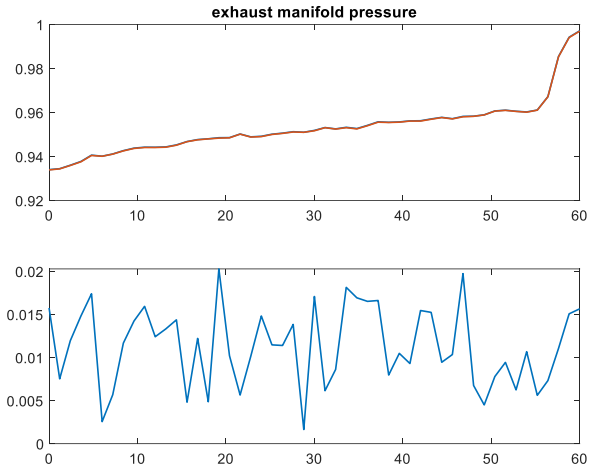


Figure 25, exhaust manifold pressure validation

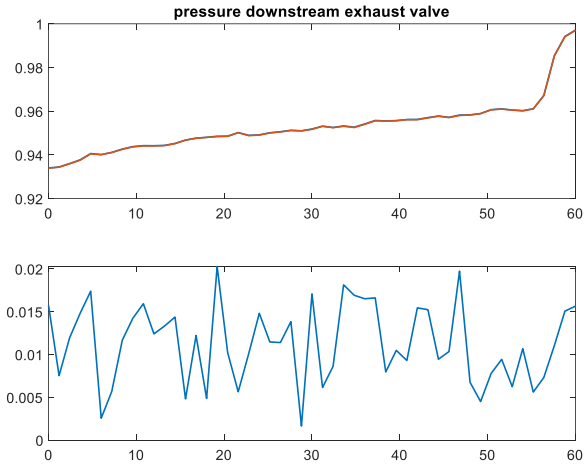


Figure 24, exhaust valve ds pressure validation

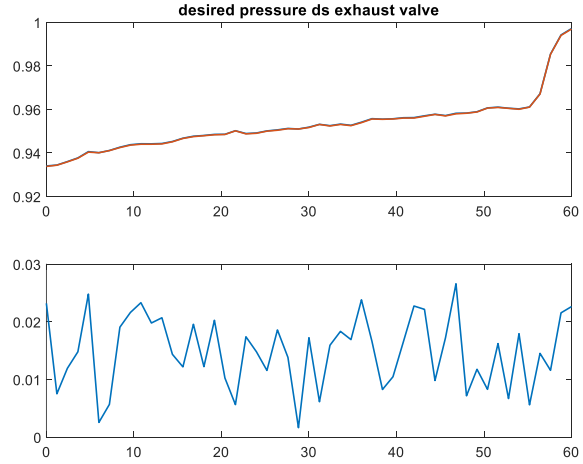


Figure 26, exhaust valve ds target validation

In all the three plots, the error is much lower than the critical threshold, even if the actual calibration chooses the physics-based approach involving much more operations than the map-based one.

4.1.4 Calculation of the fresh air in the combustion chamber

According to the calibration dataset the subsystem switches the source from which computing the combustion chamber relative air charge.

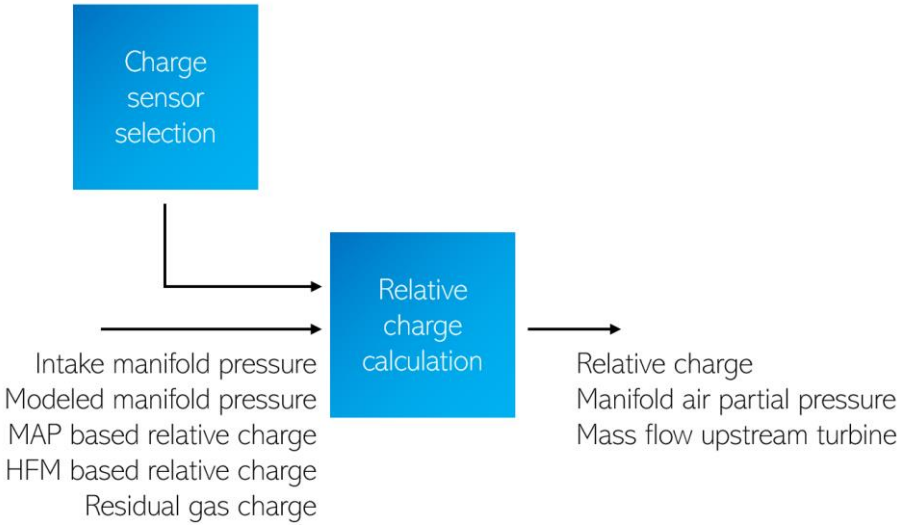


Figure 27, fresh air calculation blocks scheme

The most representative validation scopes are reported in *figure 28*, *figure 29*, *figure 30*, and *figure 31*.

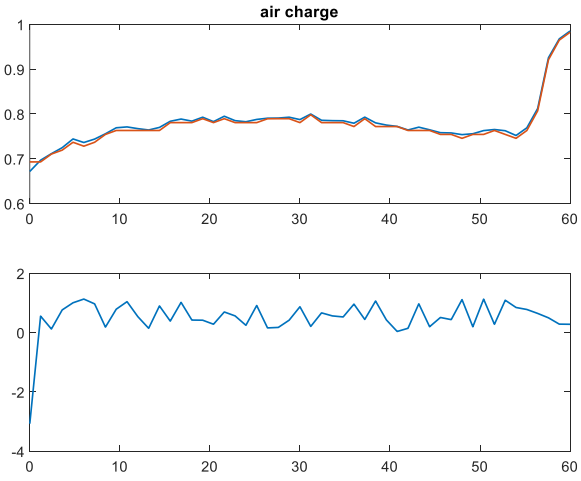


Figure 28, air charge validation

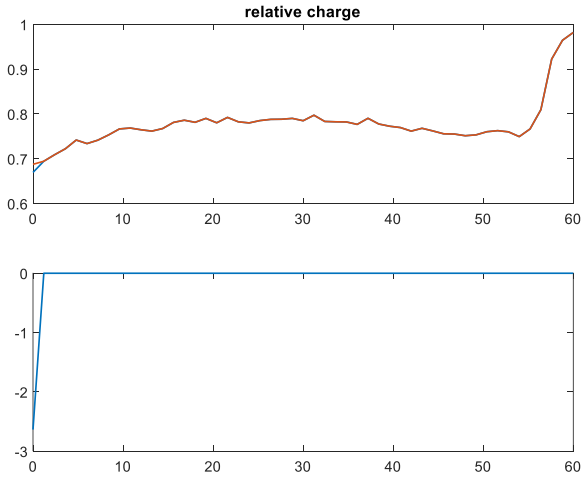


Figure 29, relative charge validation

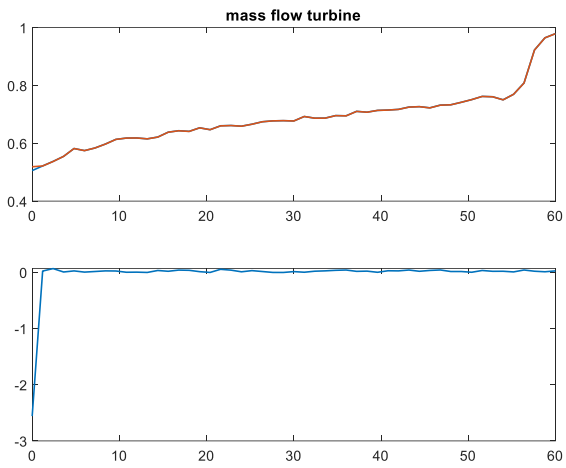


Figure 31, mass flow turbine validation

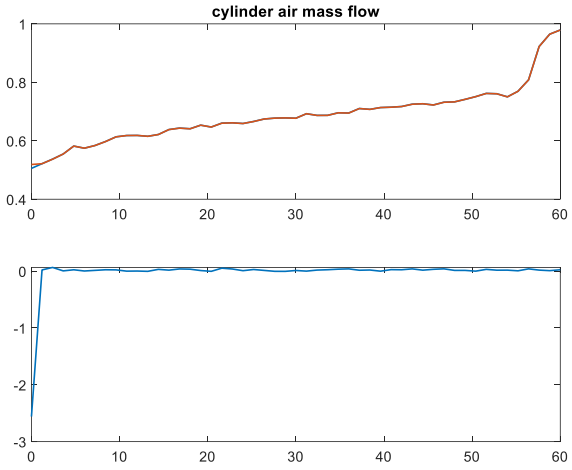


Figure 30, cylinder air mass flow validation

The initial error peak is representative of the transient behavior of filters, integrators or delays due to lack of initial conditions specifications. After the first seconds, all the signals error is below the critical threshold

4.1.5 Intake manifold pressure model

The intake manifold pressure model is a critical element of the model-in-the-loop environment because it depends on the accuracy of many other models. The block is structured according to the intake manifold storing behavior. It is constituted by an integrator that updates the manifold mass every step according to incoming and outcoming air flows.

The time constant of the integrator depends on the ratio between the combustion chamber and the intake manifold volumes.

Figure 32 shows the block scheme representation highlighting the dependence on the output itself. This type of dependence is called *algebraic loop*.

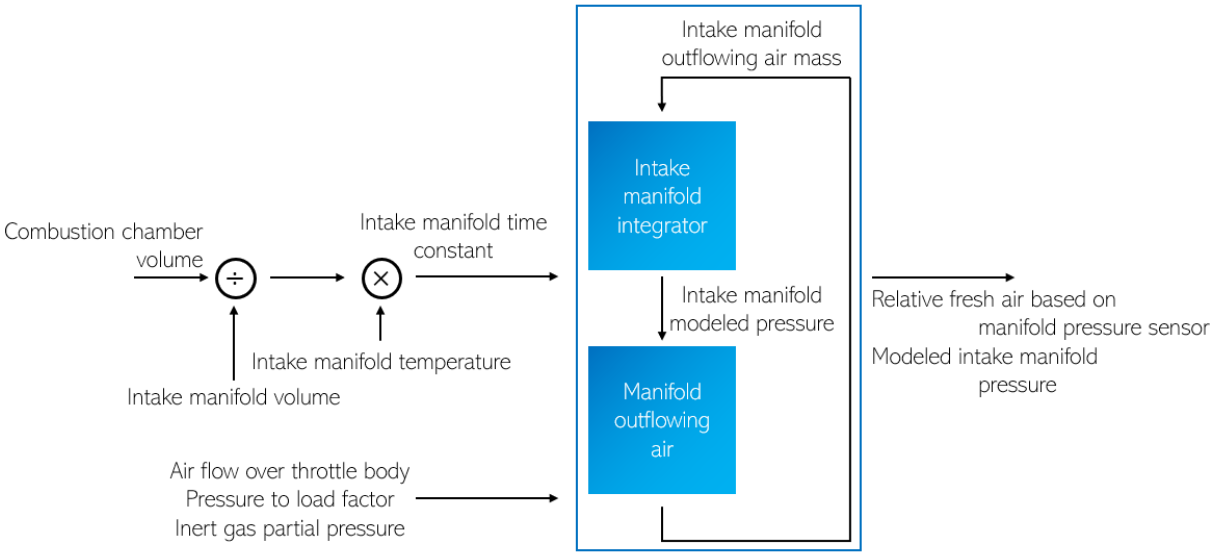


Figure 32, intake manifold pressure model blocks scheme

A magnification of the datasheet compliant model output specification is reported in *figure 33*:

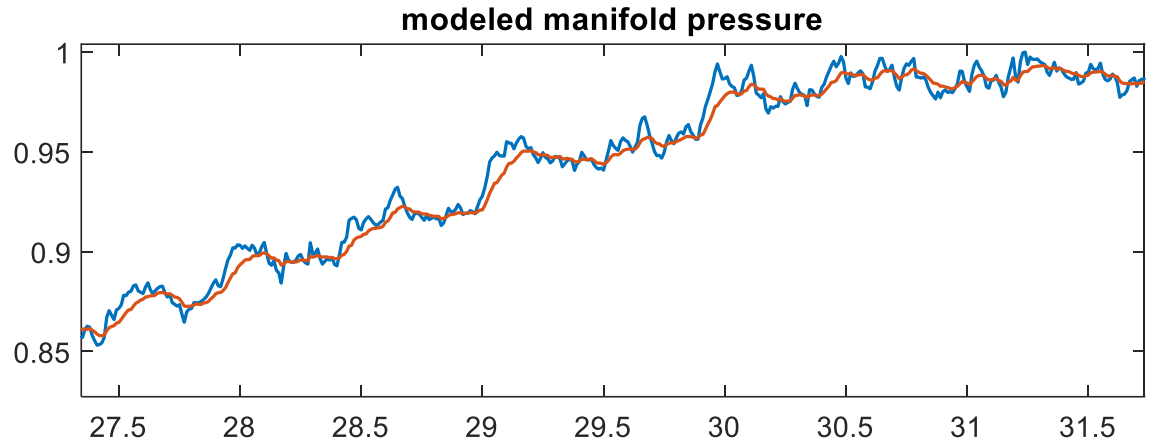


Figure 33, modeled manifold pressure error

Even though the error is smaller than the 2% threshold, the Simulink model is not able to correctly capture the dynamic of the validation signal, with its peaks and valleys.

The origin of this phenomenon proves how Simulink algebraic loops management works. As MathWorks documentation reports, an algebraic loop occurs when a signal loop exists only with direct feedthrough blocks within the loop. This means that Simulink needs the value of the block input signal for computing its output at the current time step. Such signal loop creates a circular dependency of the block outputs and inputs in the same time step. This results in an algebraic equation that needs to be solved at each iteration, demanding additional computational effort.

ECU software, instead, resolves loops in a different way. It simply takes the value of the loop signal from the previous step, without sub-iterations along the elementary execution step.

To simulate the behavior in the model block, introducing a memory or a unit-delay Simulink block in the feedback path is sufficient.

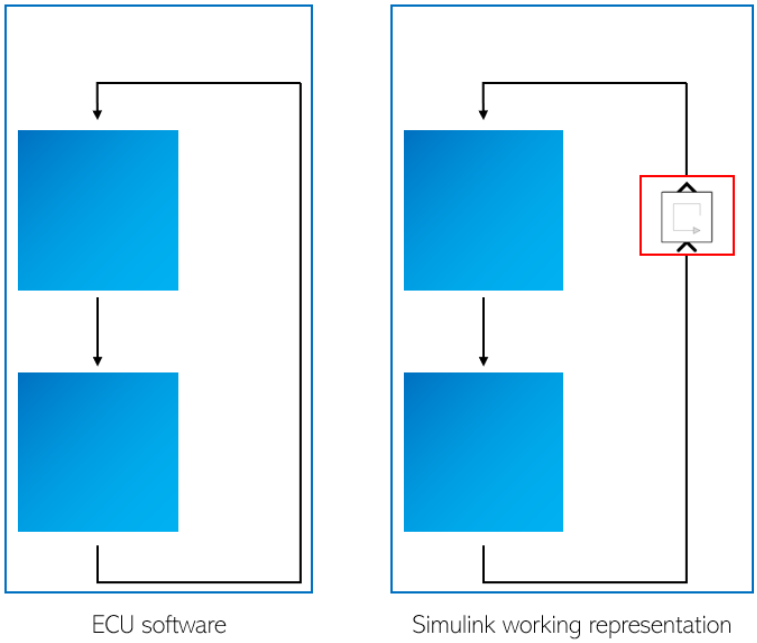


Figure 34, artificial delay

Figure 34 shows the delay implementation in the subsystem.

Thanks to this fixing, it is possible to correctly reproduce the model behavior, as *figure 35* and *figure 36* show.

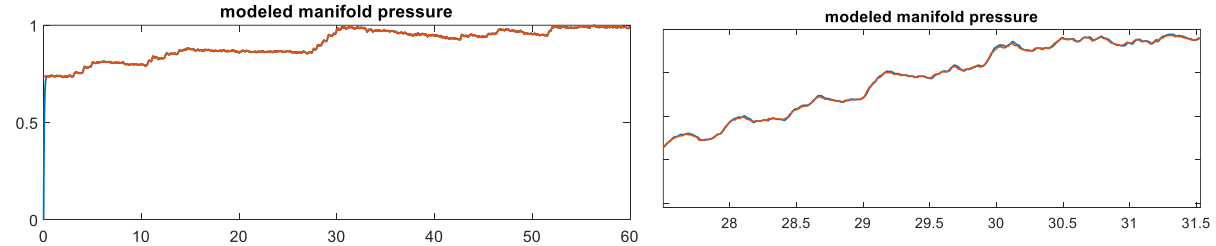


Figure 35, modeled manifold pressure magnification

Figure 36, modeled manifold pressure validation

4.1.6 Intake manifold inflow model

The subsystem calculates the relative charge flowing into the intake manifold based on HFM sensor signal or throttle valve mass flow model according to the calibration dataset, as *figure 37* shows.

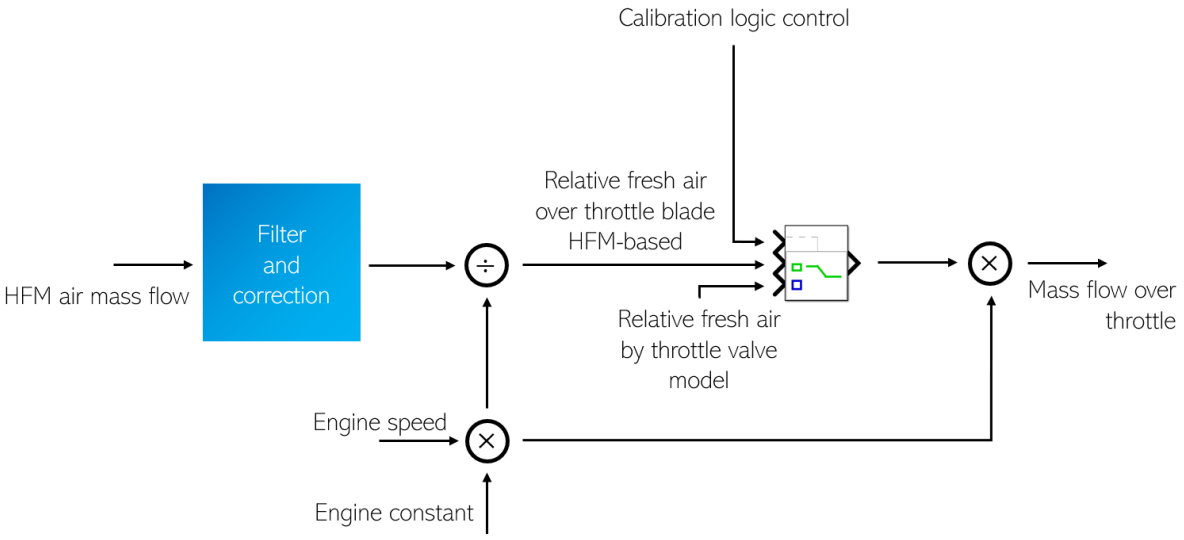


Figure 37, intake manifold inflow model blocks scheme

figure 38 and *figure 39* report the most relevant validation plots.

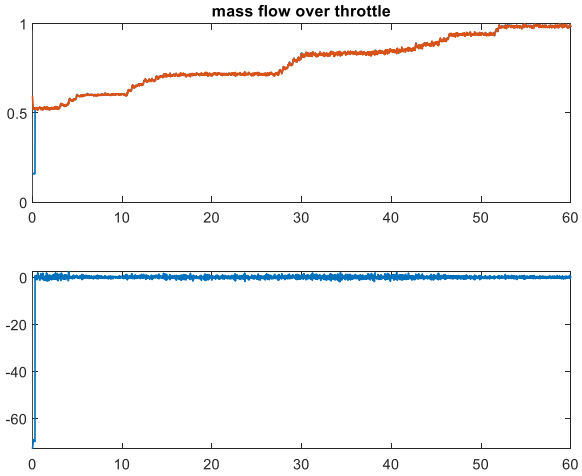


Figure 39, throttle mass flow validation

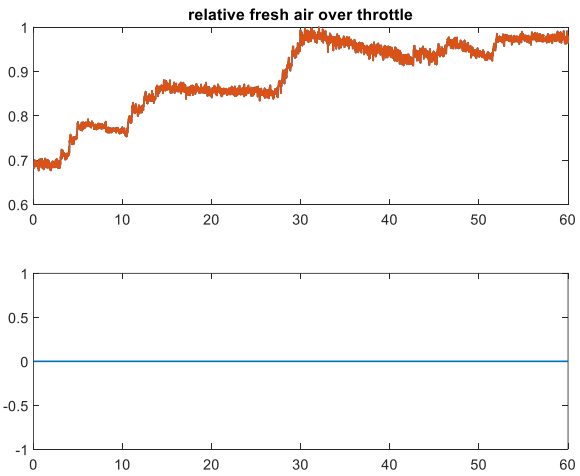


Figure 38, relative throttle fresh air

The behavior of the model is satisfactory as the error is below the 2% critical threshold.

4.1.7 Intake manifold temperature model

The module is constituted by two paths according to the action of the exhaust gas recirculation devices. In both cases it consists of a simple filter between the intake valve temperature model and its dependencies.

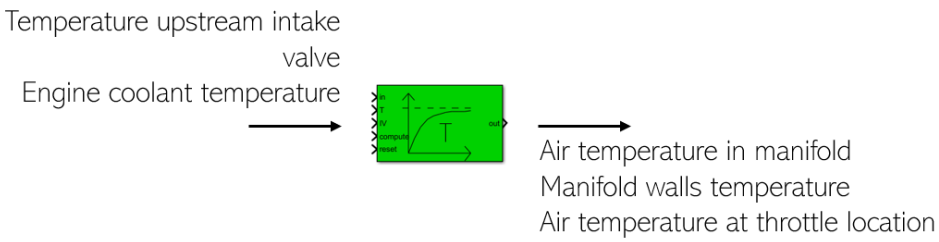


Figure 40, intake manifold temperature model blocks scheme

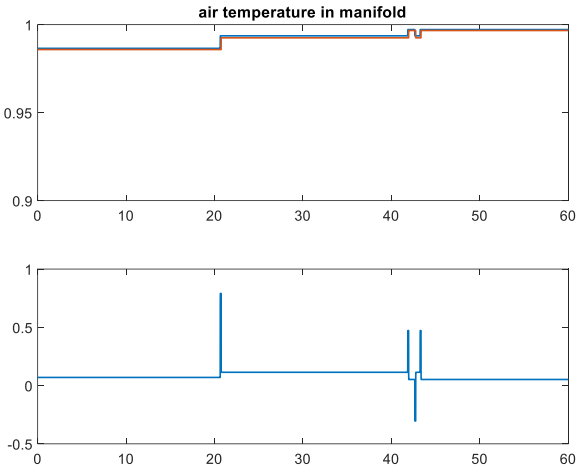


Figure 41, manifold temperature validation

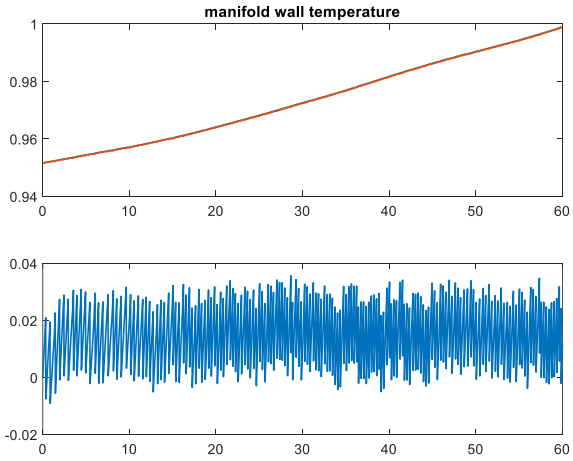


Figure 42, manifold wall temperature validation

Even if the module is simple, the error is small but not null. This behavior is due to non-perfect lowpass filter agreement between the ECU software and the Simulink interpretation. X-in-the-loop simulations, in many cases, retain errors since a virtual simulation of the hardware architecture is performed.

4.1.8 Intake manifold temperature compensation

The subsystem computes the temperature of the aspirated air into the combustion chamber. It accounts limit conditions like the warm-up and the step load transients.

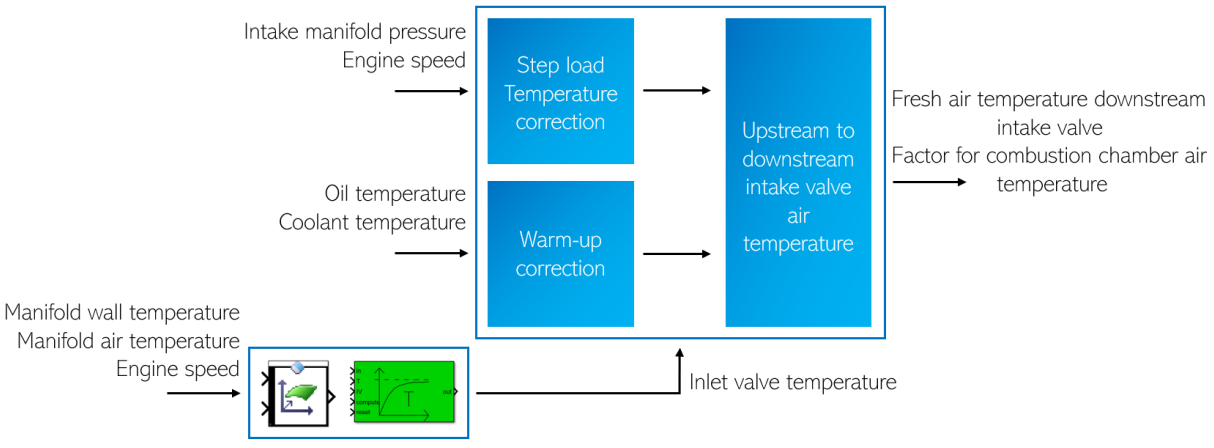


Figure 43, intake manifold temperature compensation blocks model

Output plots are reported in *figure 44* and *figure 45*.

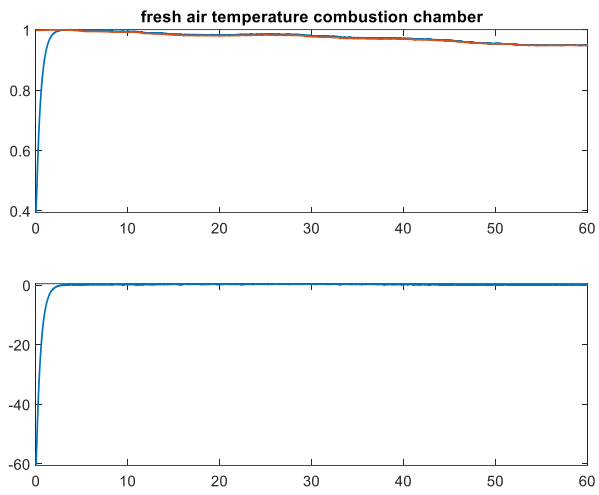


Figure 45, combustion chamber fresh air validation

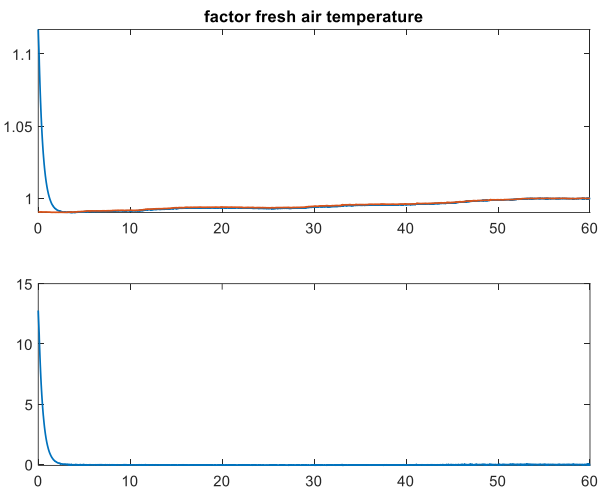


Figure 44, fresh air temperature factor validation

Apart from the long initial transient the subsystem does not show issues. The origin of the phenomenon is the filter initialization. In this case, null value is provided to the lowpass filters as the reference paper does not contain specifications. The initial execution of the model, therefore, considers zero degrees air temperature but, after few steps, the correct trend is reached.

4.1.9 Air system assembly

This chapter describes the next hierarchical step respect to the previous ones. The second activity phase consists of assembling the previously validated blocks to figure out the cumulative error entity and pattern.

Figure 46 summarizes the dependencies of the modules by themselves.

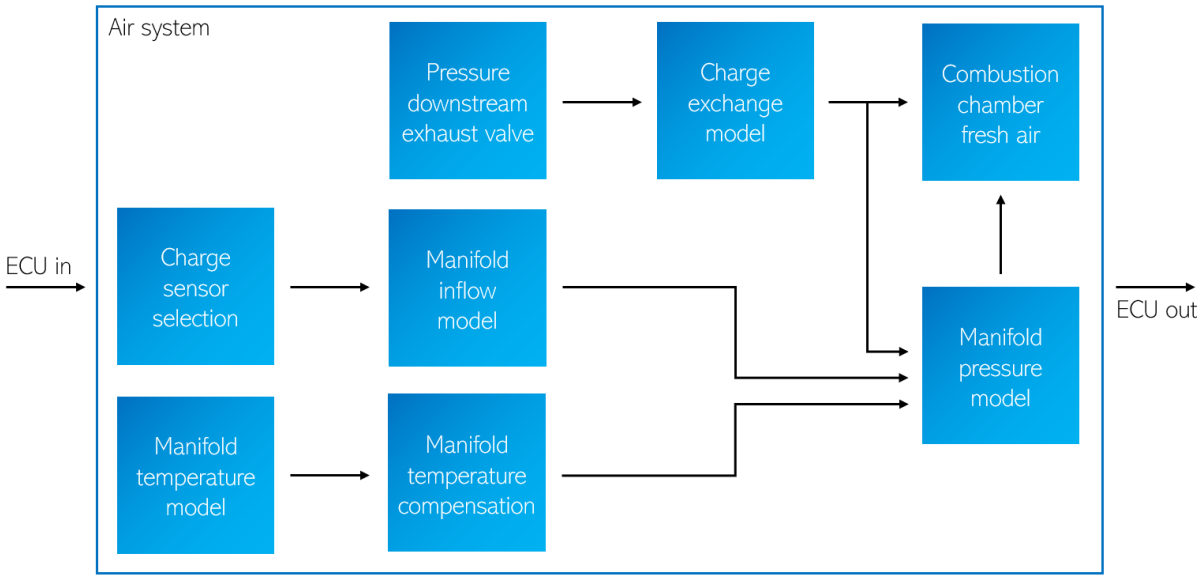


Figure 46, air assembly blocks scheme

The equivalent implementation of the block scheme, which is available in the Simulink library, is reported in the *figure 47*.

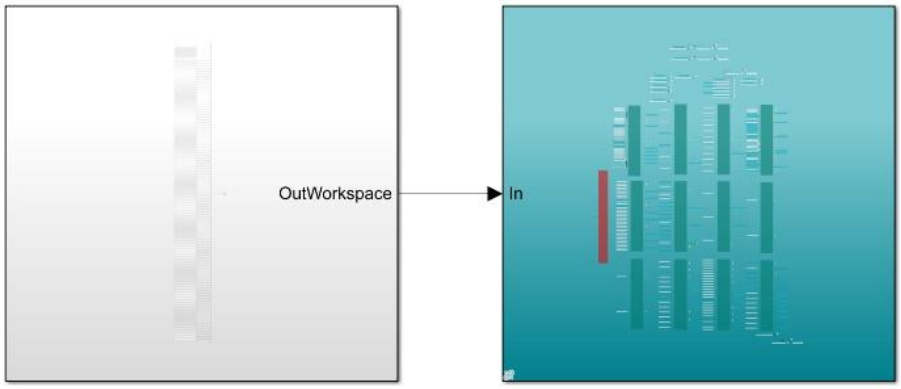


Figure 47, air assembly Simulink implementation

According to *figure 46*, the last involved modules in the calculous chain are the *Manifold pressure model* block, the *Combustion chamber fresh air* block, and the *Charge exchange* model. Therefore, validating the outputs of these blocks gives a measure of the implementation validity.

As previously shown, the *Charge exchange model* computes the *EGR manifold partial pressure*, *figure 48*, and the *factor for pressure to load conversion*, *figure 49*.

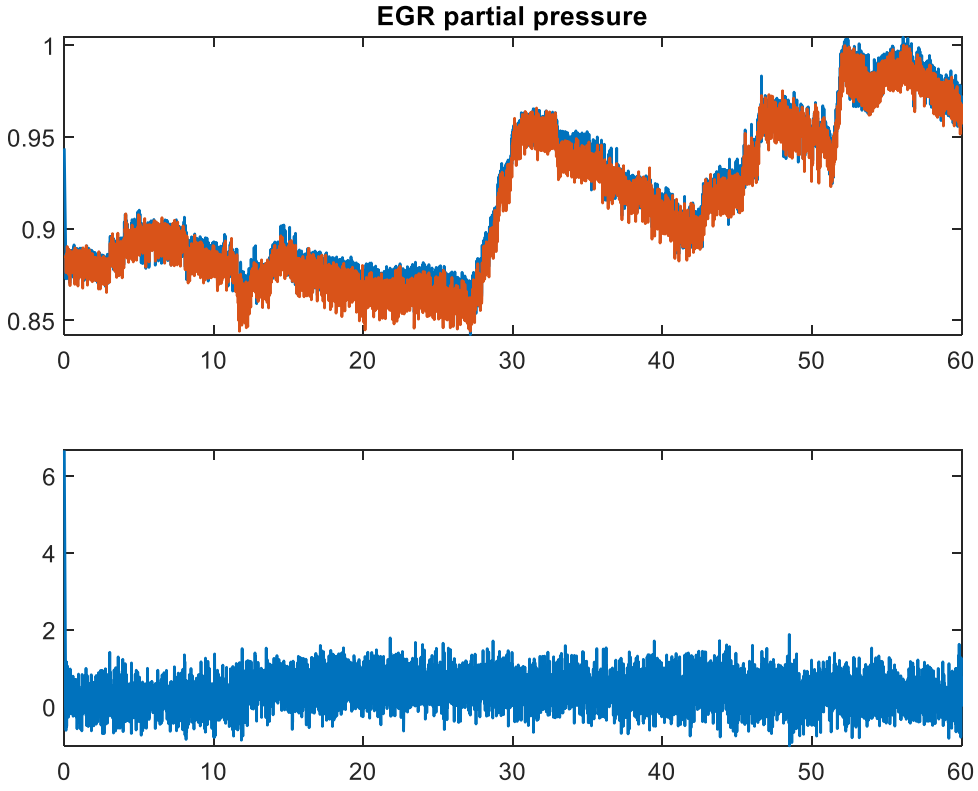


Figure 48, air assembly EGR partial pressure validation

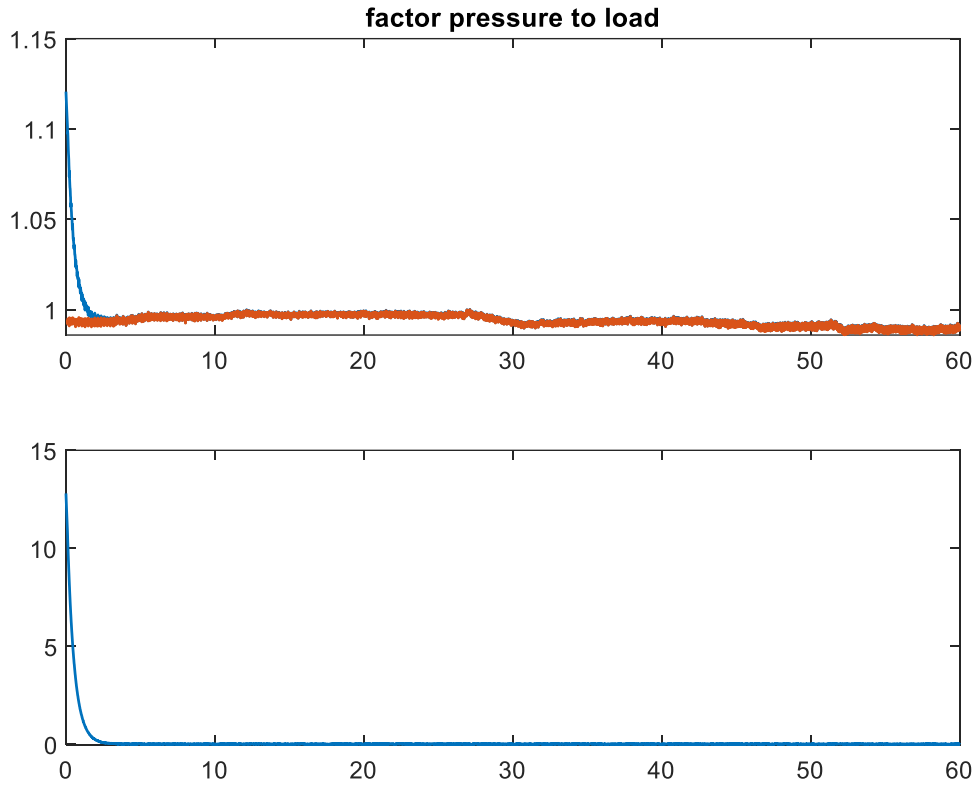


Figure 49, air assembly pressure to load factor validation

The *manifold pressure model* outputs the intake manifold modeled pressure (*figure 50*).

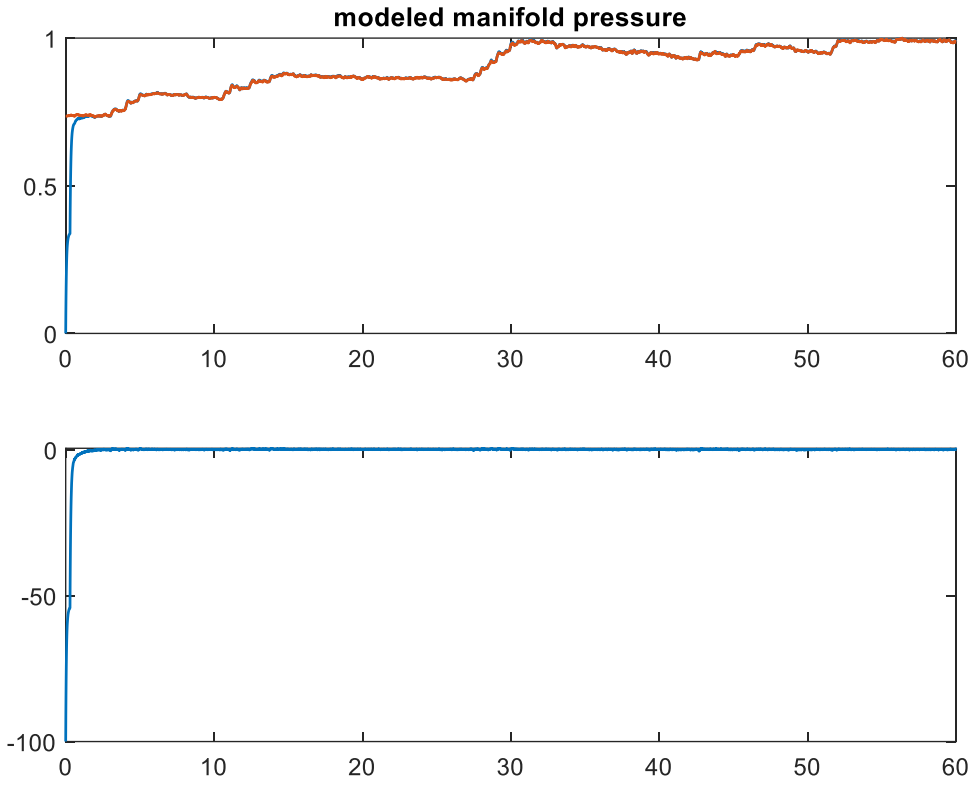


Figure 50, air assembly intake manifold pressure model validation

Finally, the *Combustion chamber fresh air* block calculates the pressure downstream the exhaust valve (*figure 51*).

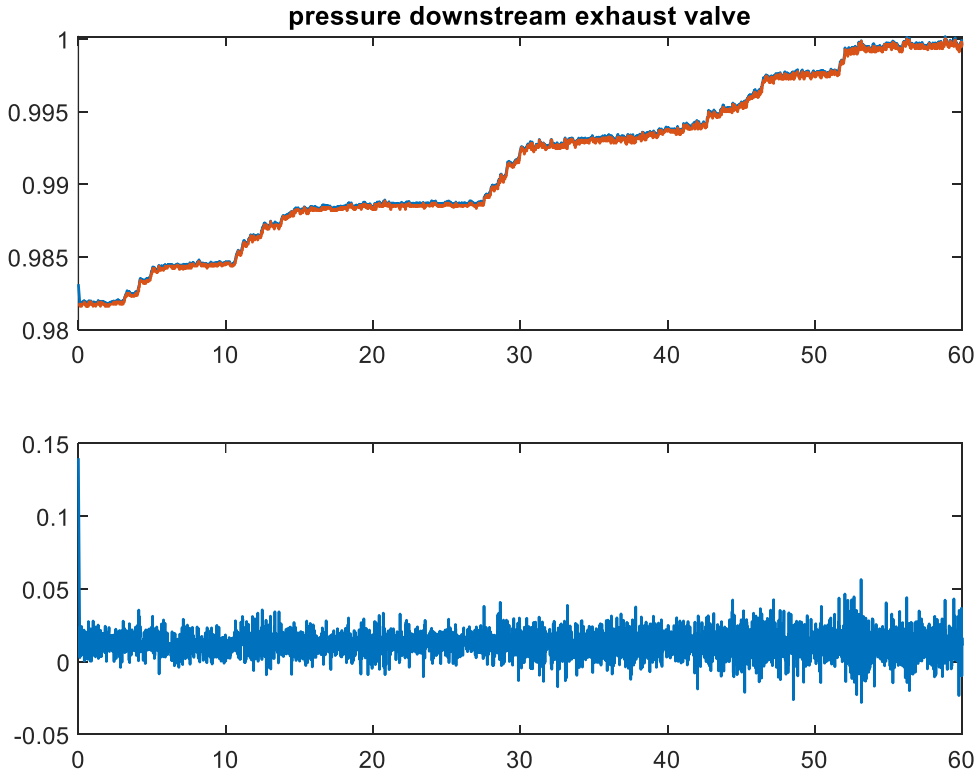


Figure 51, air assembly pressure downstream exhaust valve validation

The considered plots meet the validation criteria with the percentage error below the 2% critical threshold. The EGR partial pressure signal shows a small degradation compared to the standalone run, but it is still acceptable.

As reference, the accuracy of the combustion chamber relative charge signal is reported in *figure 52* as long as it is involved in fundamental calculations such as the injected fuel, the variable valve timing operations and the spark advance selection.

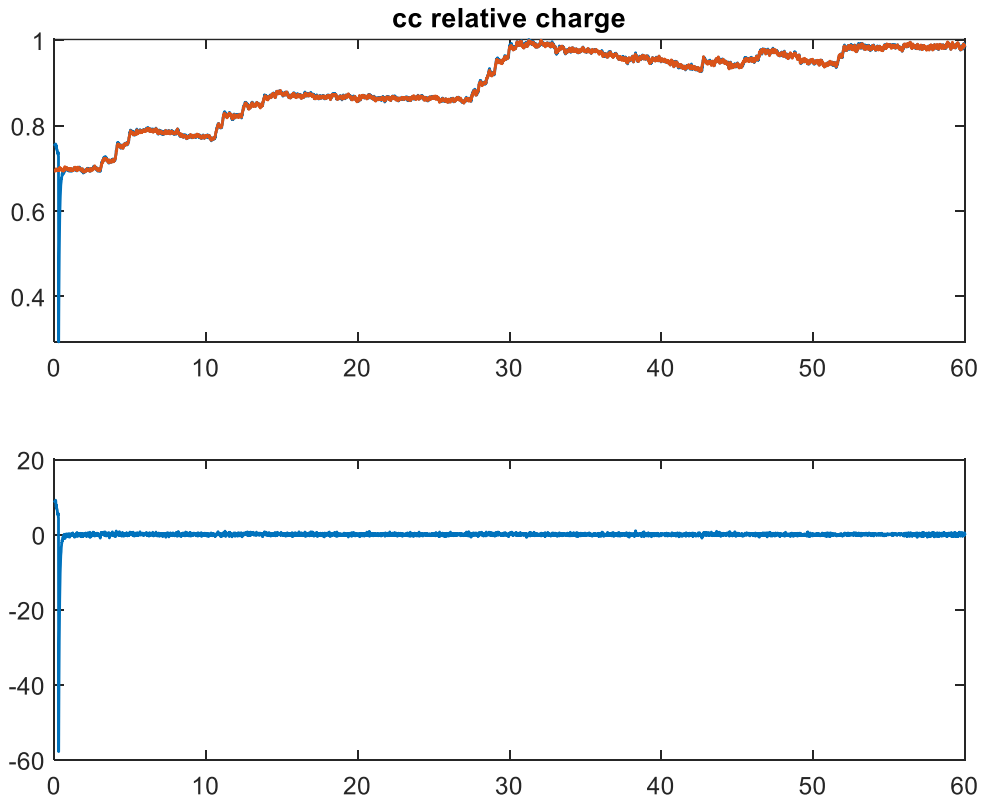


Figure 52, air assembly combustion chamber relative charge validation

The relative charge signal correctly captures the frequencies, the peaks, and the valleys of the recorded values while keeping the error in the reasonable range.

4.2 Charge support task

Charge modules require several external inputs from both engine sensors and other software modules. To make the simulation environment as independent as possible from the other ECU systems, external systems were considered beyond the charge calculations.

4.2.1 Exhaust system

Mass flow model

The subsystem assumes the mass conservation across every exhaust element, assigning to each device the same mass flow value according to the current engine operating mode. It also introduces the concept of the configuration array, a calibration data from which the software is configured for reproducing the exhaust sequence. Exhaust subsystems, in that way, are parametric towards the specific engine, letting the ECU supplier reusing the same software block in more than one implementation.

Figure 53 summarizes the software implementation.

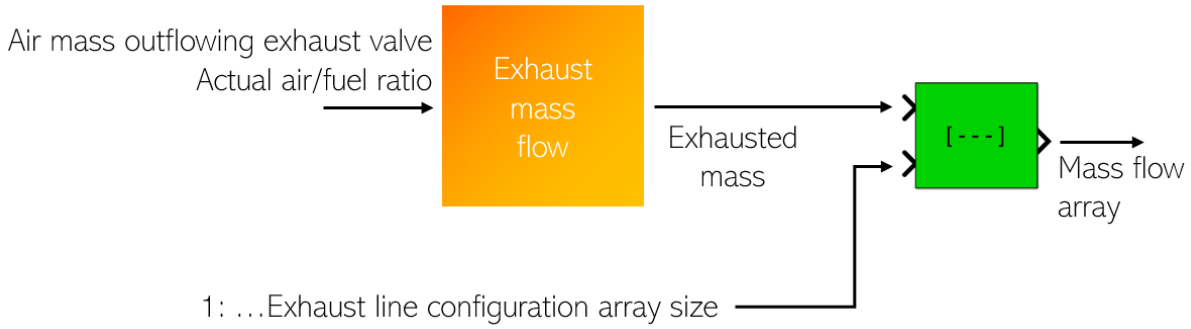


Figure 53, mass flow model blocks scheme

Hence, comparing the first element is sufficient for validating the module as the array contains the same signal at every index.

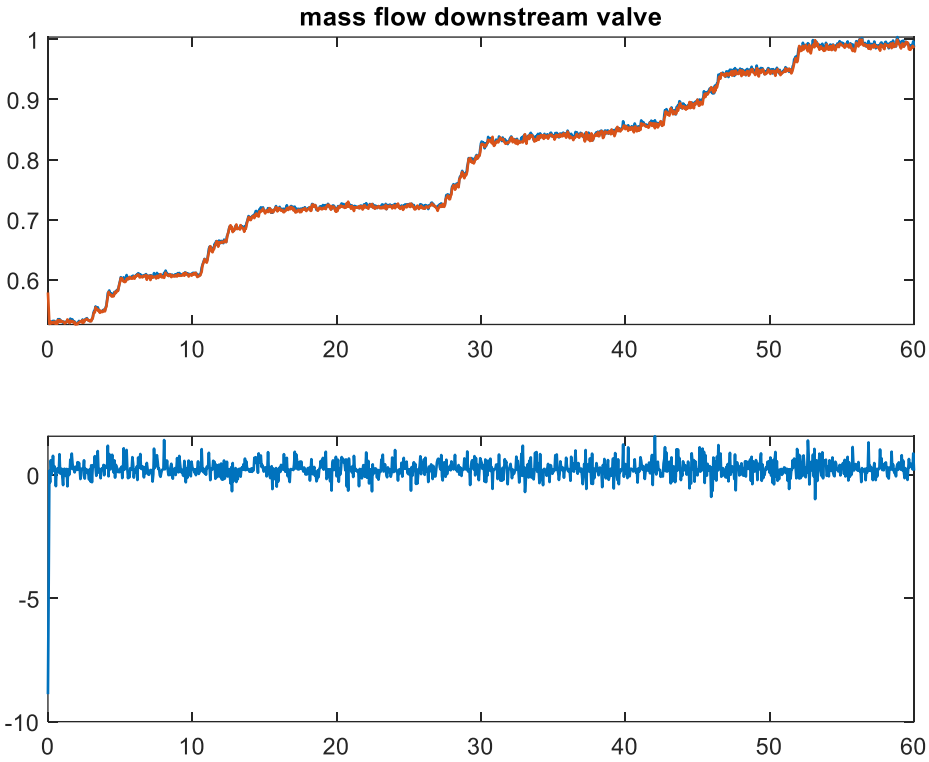


Figure 54, mass flow downstream exhaust valve validation

The comparison returns acceptable results even if the error shape is slightly different from what the air system produces. In particular, the error appears less influenced by the background noise, but the spikes are wider and higher. The different simulation timestep of the exhaust subsystems is the reason why the error shape is different, as it is five times larger than the air system one, giving longer error duration and coarser signal interpolations.

Pressure model

The subsystem models the pressures between the exhaust elements along the exhaust path from the tailpipe atmospheric pressure to the first catalyst value upstream.

Figure 55 shows the array filling operation through the iterative *for loop*. In this case, the loop performs one iteration for every exhaust path element.

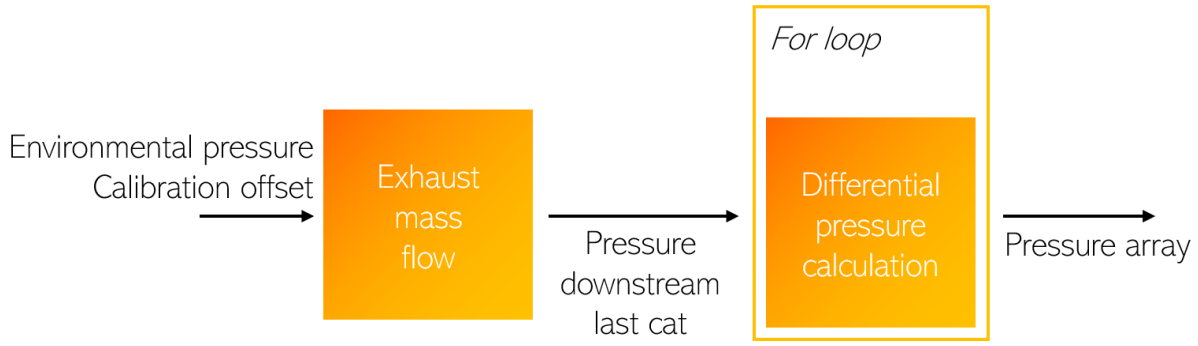


Figure 55, pressure model blocks scheme

Representing the same construct with Simulink block language is not so straightforward because of the different purposes that the iterator block has. For instance, a workaround is necessary for the following reasons:

- Unlike ECU programming language, a signal can't be written through different sources as the original logic implementation chooses from which block the signal comes. Simulink model must bring all the sources outside the computational blocks and decide from which taking the signal, after all the calculations are performed.
- Block activation or deactivation in Simulink must observe strict conditions (prohibited use of *From/GoTo* within deactivated blocks, single writing source even before the model run).

The applied solution consists of repeating the single computational unit as many times as the original iterative loop runs. The main disadvantage of this approach is the non-parametrization according to the calibration dataset. Changes in the topology of the exhaust path thus require simple model rearrangement.

Figure 56 shows the way in which the model is reconstructed. The pressure array is made by three elements. The last one, from which the calculation starts, is the measurement of the pressure downstream the last element. The second array element calculates the delta-pressure that the next exhaust device generates, which is added up to the measured tail pressure. The first array index value is computed in the same way.

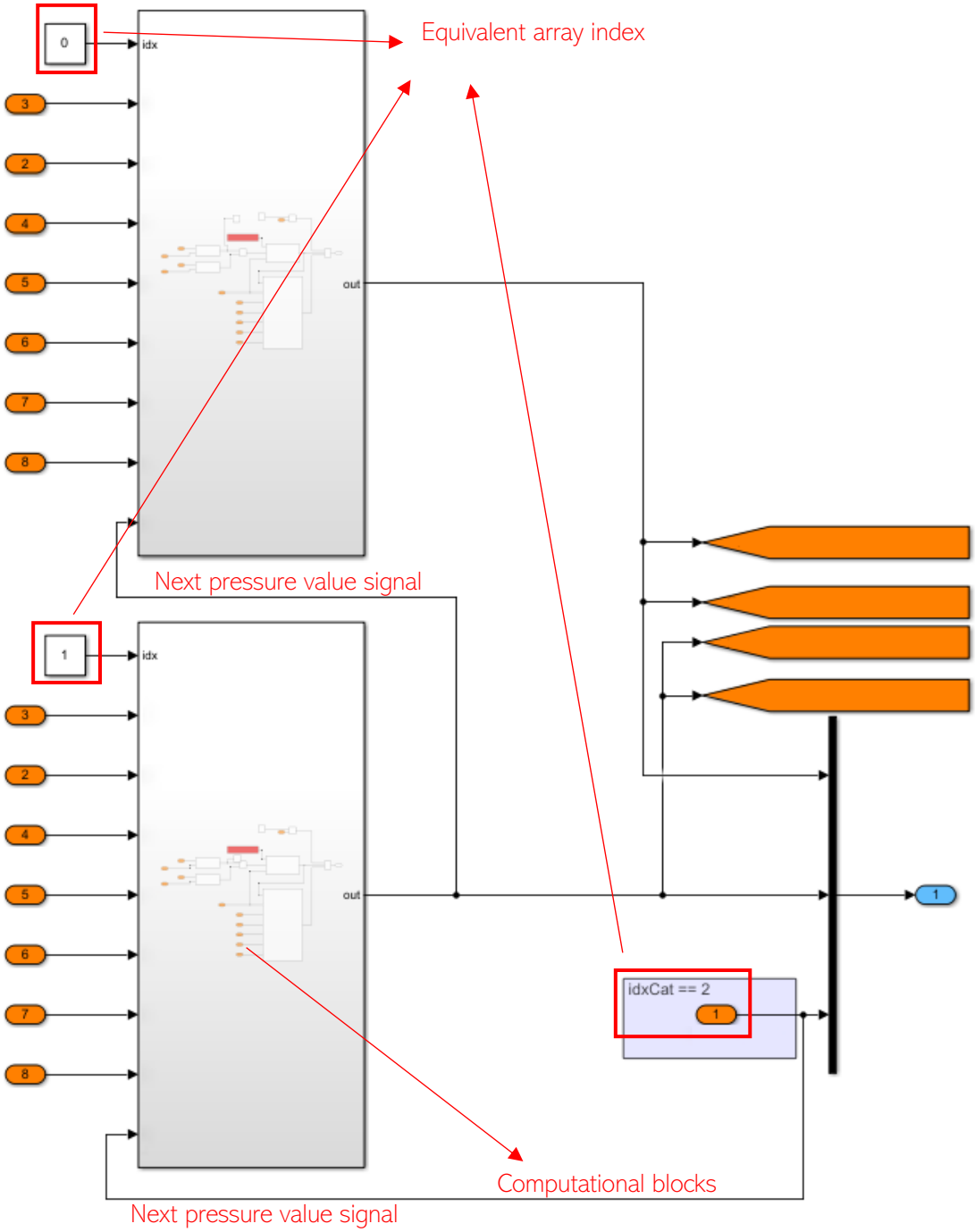


Figure 56, pressure model Simulink implementation

The inputs are the same for the two computational units, as the block depends on the downstream pressure only.

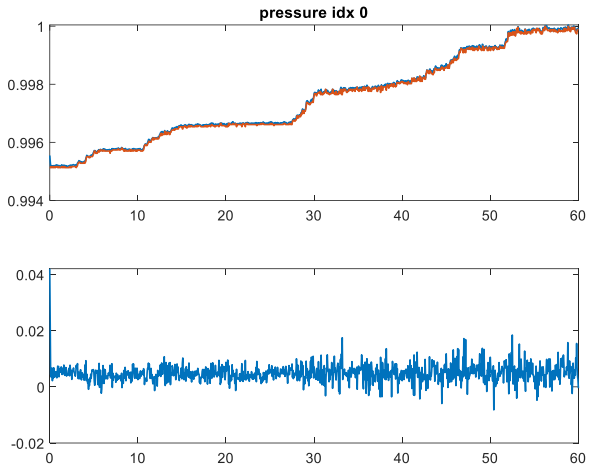


Figure 58, pressure ds second cat validation

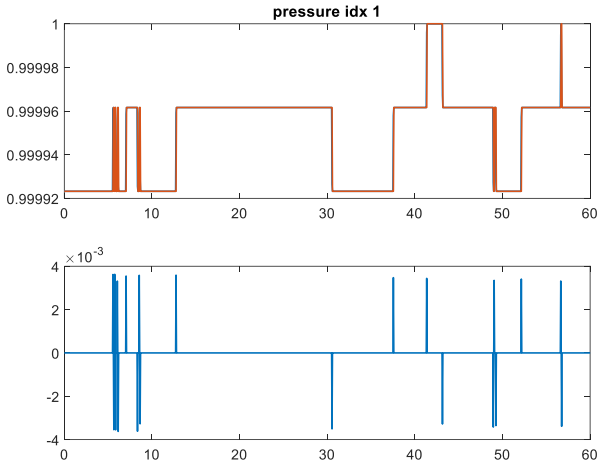


Figure 57, pressure ds first cat validation

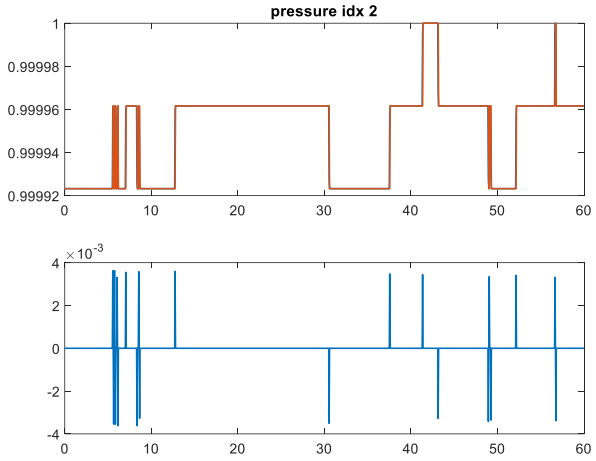


Figure 59, pressure us first cat validation

At the first index, the pressure plot shows a physical trend according to the sensor measurement on the engine run. The other two plots show a stairs trend. The phenomenon is due to a coarse signal resolution and rounding. The constant, on which the rounding is based, is a property of the software and it is imposed by the ECU supplier.

The Simulink representation is thus able to reproduce that behavior.

Temperature module

The temperature model follows the same iterative scheme of the previously implemented pressure block. In this case, the configuration array is larger. For instance, every catalyst is subdivided in four slices: input, output, and middle temperatures.

The for loop, as the ECU datasheet shows, is set up in a slightly different way. If the pressure model consisted of the same replicated blocks, the temperature computational unit is

specialized in three different thermal models: pipe, turbine, and catalyst. Every *for* iteration the configuration array activates the corresponding thermal model as *figure 60* shows.

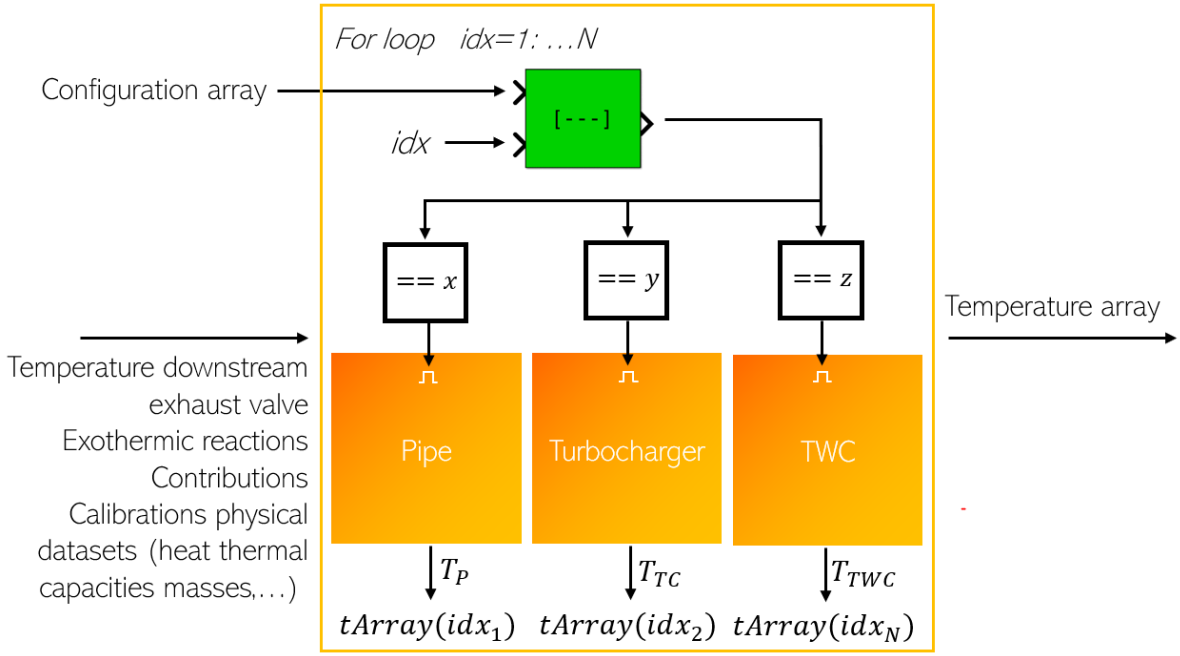


Figure 60, temperature model datasheet implementation

Simulink implementation does not consider the *for-iterator* block. The unit is added in the model for every element of the configuration array instead. The three thermal models (pipe, turbine, and catalyst) are always active at the same time and the selection takes place after the three delta-temperatures are computed. In that way, performing the array assignment outside the loop is possible in compliance with the Simulink restrictions, as *figure 61* shows.

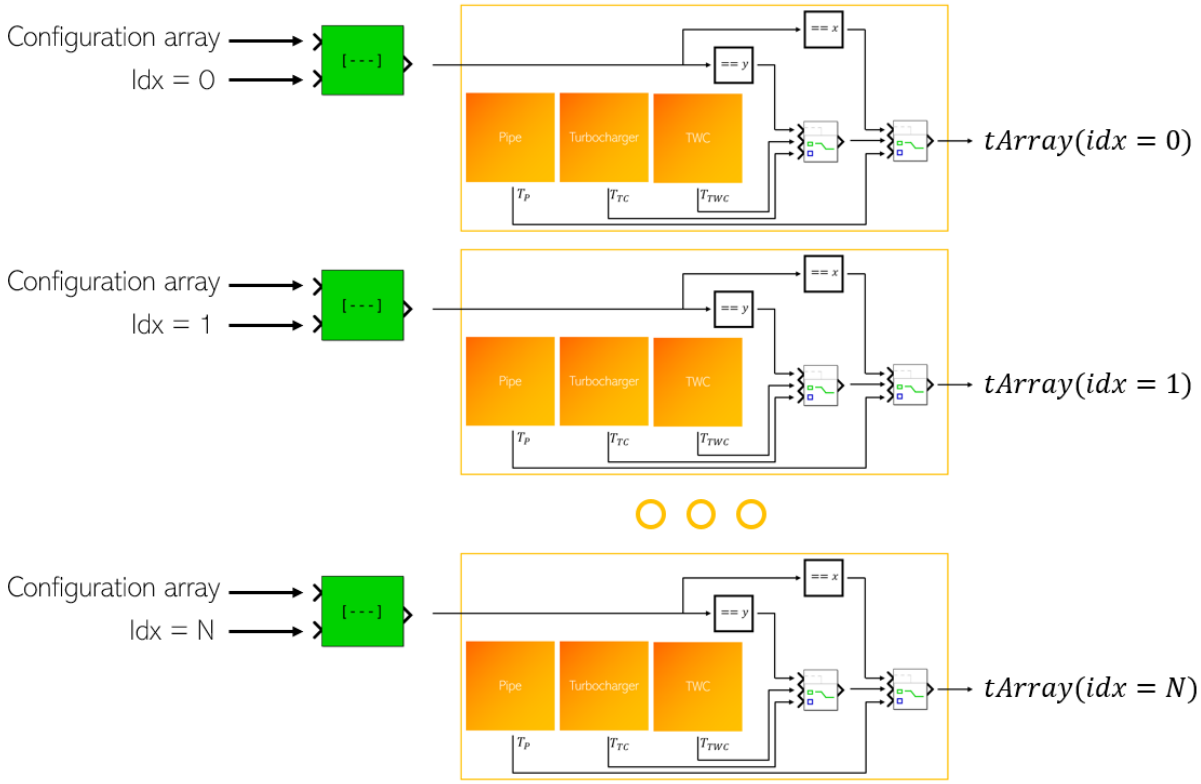


Figure 61, temperature model Simulink implementation

At the first running attempt, the previously shown *algebraic loop* management issue was found. The temperature calculation, indeed, works as the pressure model, adding a delta-temperature to the previously computed values. *Figure 62* highlights the error between the original Simulink implementation and the acquisition benchmark due to wrong loop management.

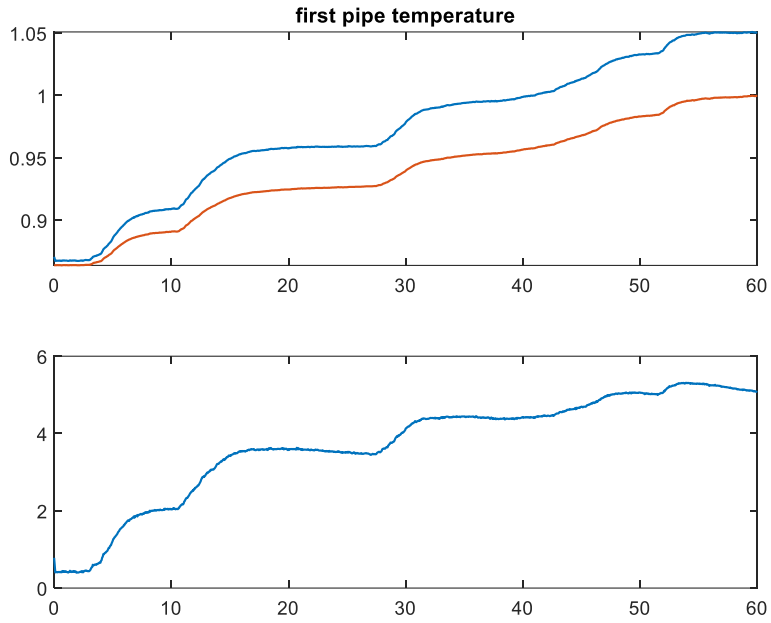


Figure 62, first pipe temperature trend without artificial delay

Another issue arose when the delay block was added into the feedback path. The default null delay initial value did not let the model to correctly reproduce the temperature trend. Even if the plots were completely stackable, the two lines was separated by a constant offset, as *figure 63* shows.

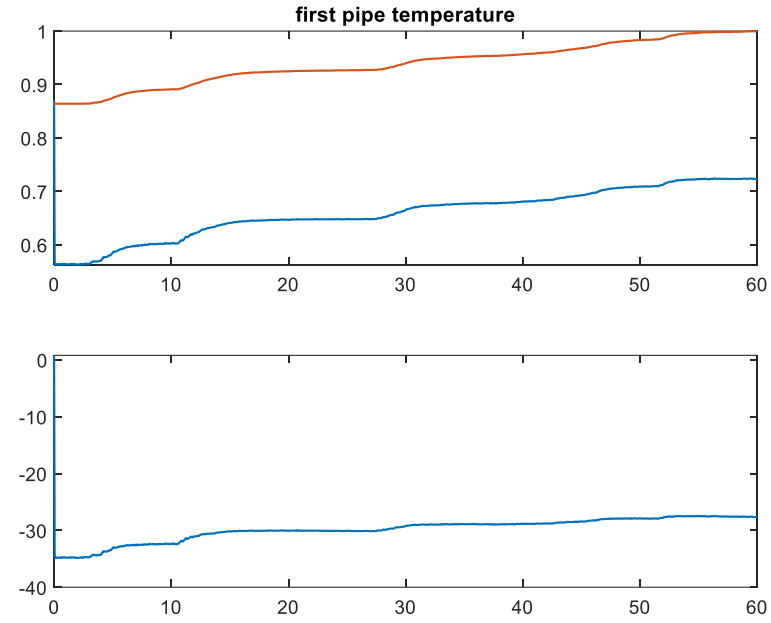


Figure 63, first pipe temperature without delay initialization

To overcome the issue, a dynamic delay block was built according to *figure 64* schematic, allowing the correct initialization of the delay, without inserting a validation value directly inside the delay configuration parameters.

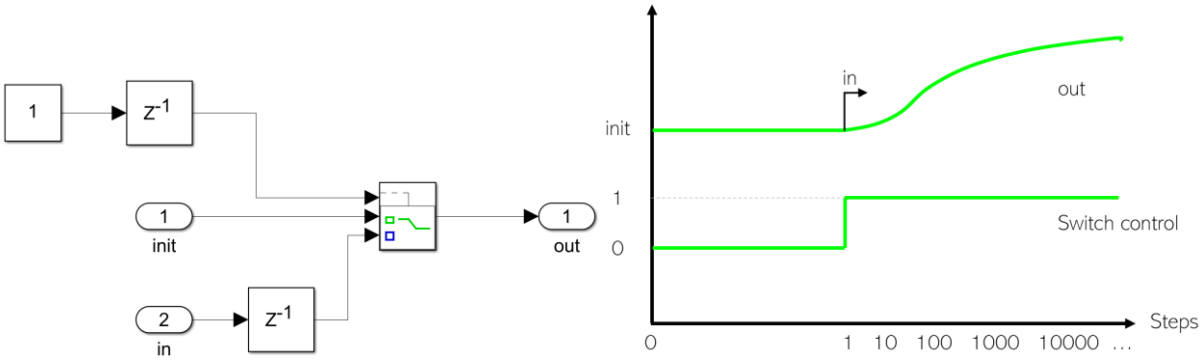


Figure 64, dynamic initialization delay block

The initial temperature is now provided with the block inputs for emulating every engine operating condition properly.

One validation scope for each thermal model (pipe: *figure 65*, turbine: *figure 66*, catalyst: *figure 67*) is reported.

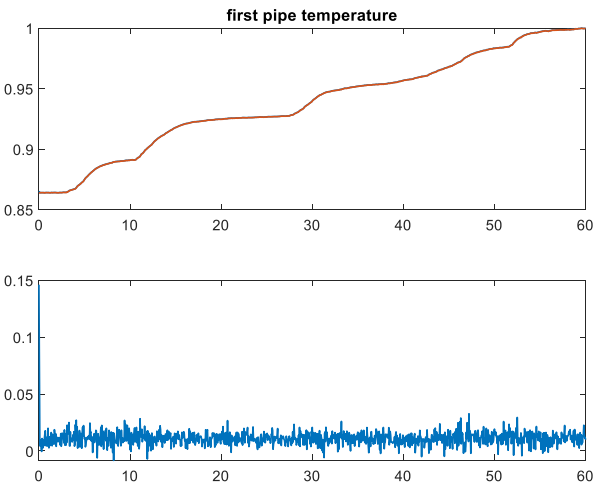


Figure 66, first pipe temperature validation

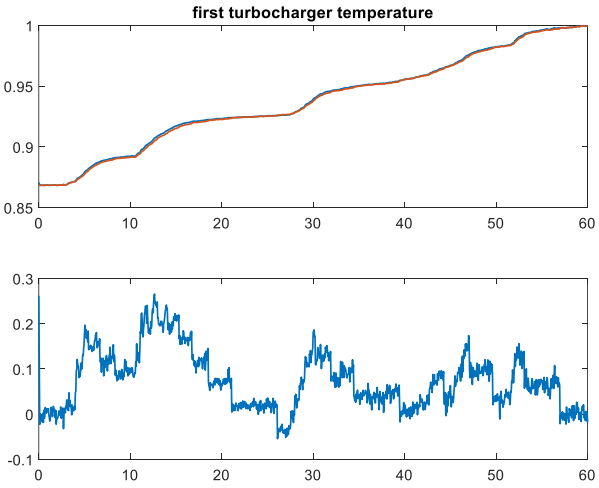


Figure 65, turbocharger temperature validation

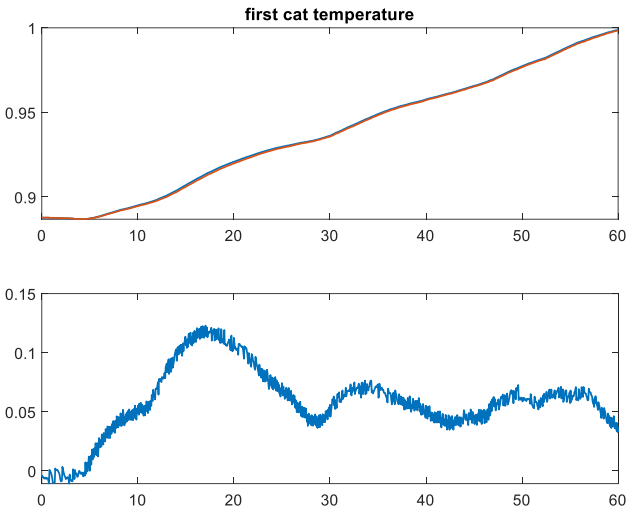


Figure 67, first catalyst temperature validation

Finally, the validation results are completely satisfactory as the error is below the 2% critical threshold, even if the model topology is different with respect to the original ECU software.

Exhaust assembly

The exhaust assembly is available in the library like the charge system block.

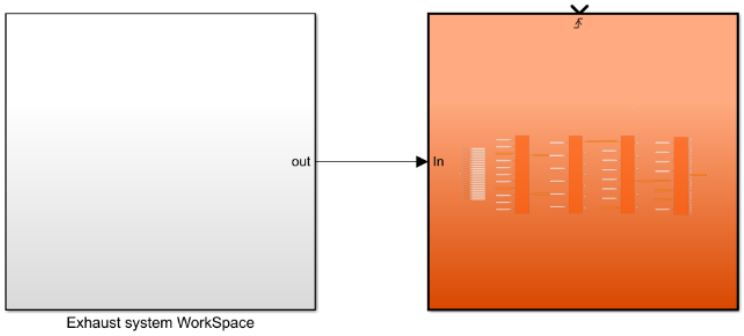


Figure 68, exhaust system Simulink implementation

In the exhaust case, the different timestep must be accounted for multiple systems running.

Considering that the exhaust timestep is multiple of the charge timestep, a possible workaround consists of triggering the assembly every delayed step. For this purpose an appropriate trigger block is added to the library.

Figure 69 shows the trigger implementation and its activity plot during the model execution.

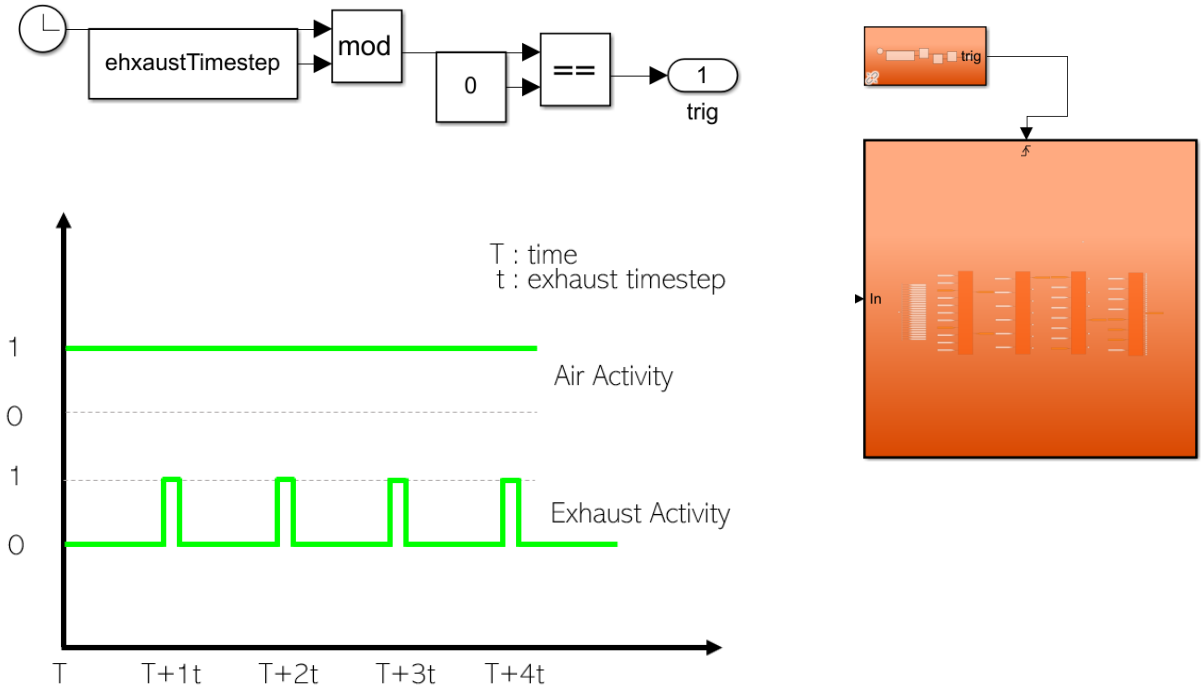


Figure 69, exhaust system triggering procedure

4.2.2 Charge control system

Throttle valve mass flow model

The subsystem calculates the air mass flow over the throttle blade through physical relations. It depends on the cross-section area of the opening, the pressure upstream, the air temperature, and the ratio between the downstream and the upstream pressures.

As the *figure 70* shows, the ECU software uses two lookup tables to resolve the mass flow equation instead of implementing the relation operation by operation. This usage is recurrent in the software and allows increasing the computational speed while lowering the ECU memory demand.

Hence, these types of calibration dataset must not be modified as they are not engine dependent.

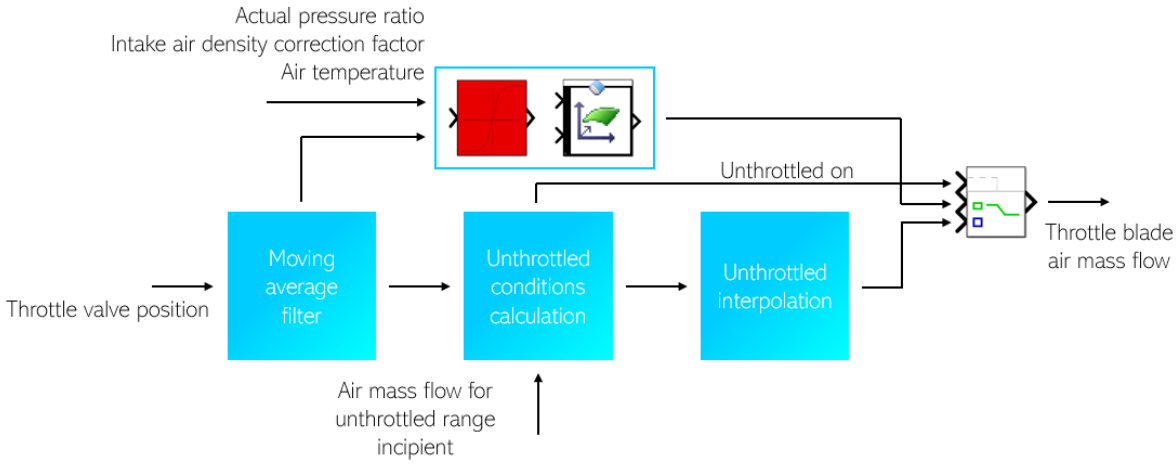


Figure 70, throttle valve mass flow model blocks scheme

Figure 71 shows the output validation. The error is below the 2% reference threshold.

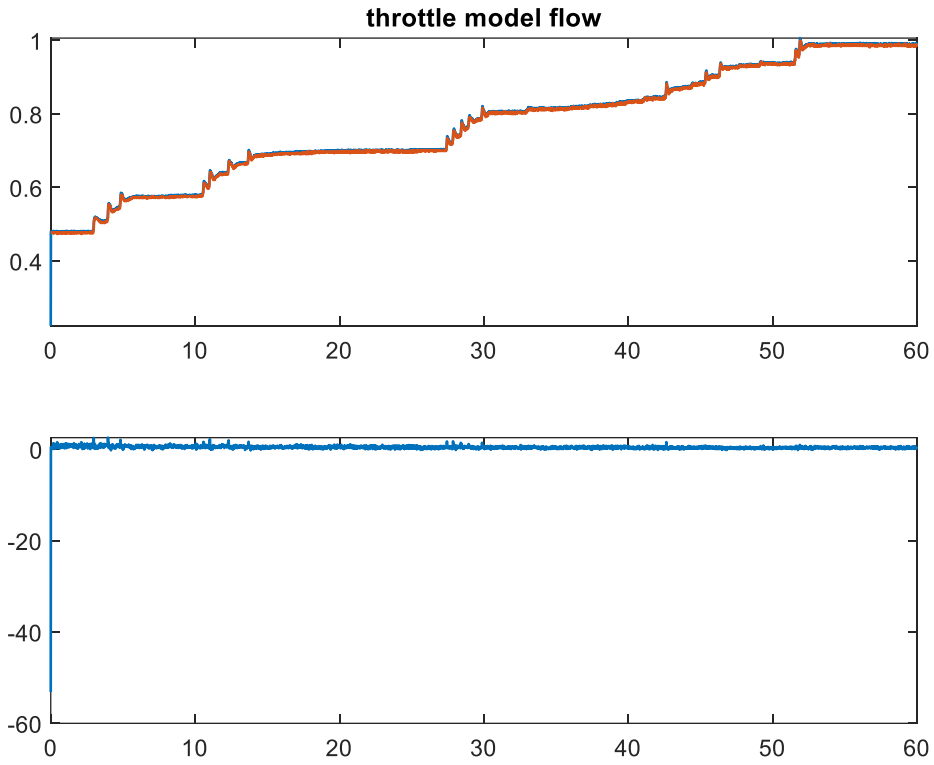


Figure 71, modeled throttle flow validation

Throttle valve mass flow setpoint calculation

The module is a simple wrapper between different relative charge calculation sources. The conversion takes place as illustrated in the right of *figure 72*. Involved signals are:

- Target air charge
- Air flow for unthrottled range incipient instability (pressure ratio = 0.95)
- Maximum air charge for unthrottled range instability (pressure ratio = 1)

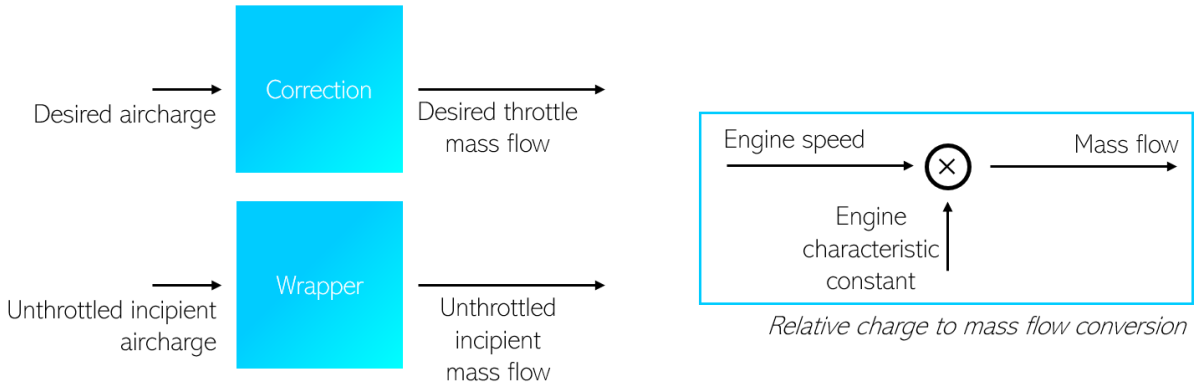


Figure 72, throttle mass flow setpoint blocks scheme and relative charge calculation

Validation plots are reported with *figure 73* and *figure 74*.

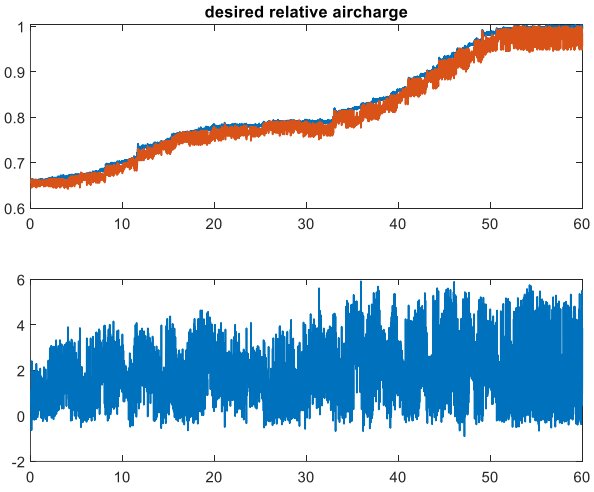


Figure 74, desired relative charge validation

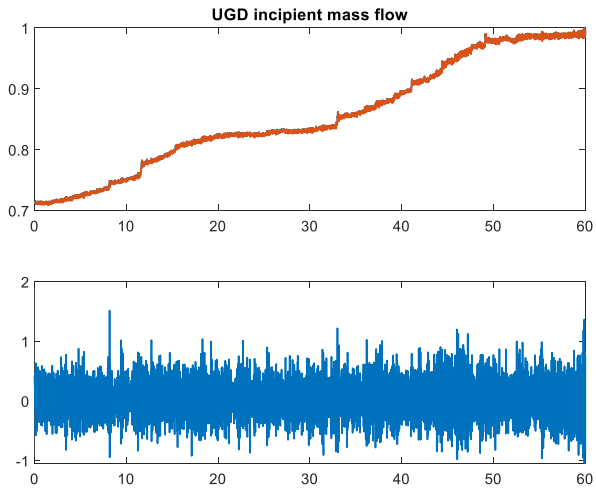


Figure 73, unthrottled mass flow validation

The model output trace of the desired relative air charge plot does not appear coherent with the validation data, as the error exceeds the 2% threshold. *Figure 75* reports the magnification of the signal.

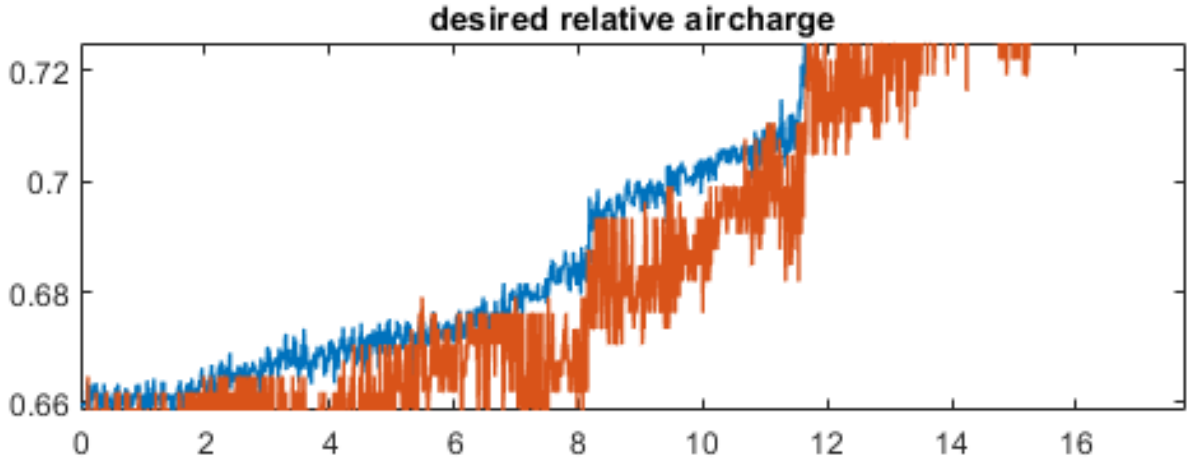


Figure 75, quantization magnification

The orange signal, generated by the engine control unit, shows a typical stairs effect due to coarse rounding offset. The quantization constant is characteristic of the ECU software and cannot be changed during this phase.

The validation is therefore considered coherent.

Throttle angle setpoint calculation

The model consists of a simple filters sequence that takes as input the target throttle angle and outputs a filtered actuator setpoint. There are two different paths, as *figure 76* shows, depending on the desired responsiveness.

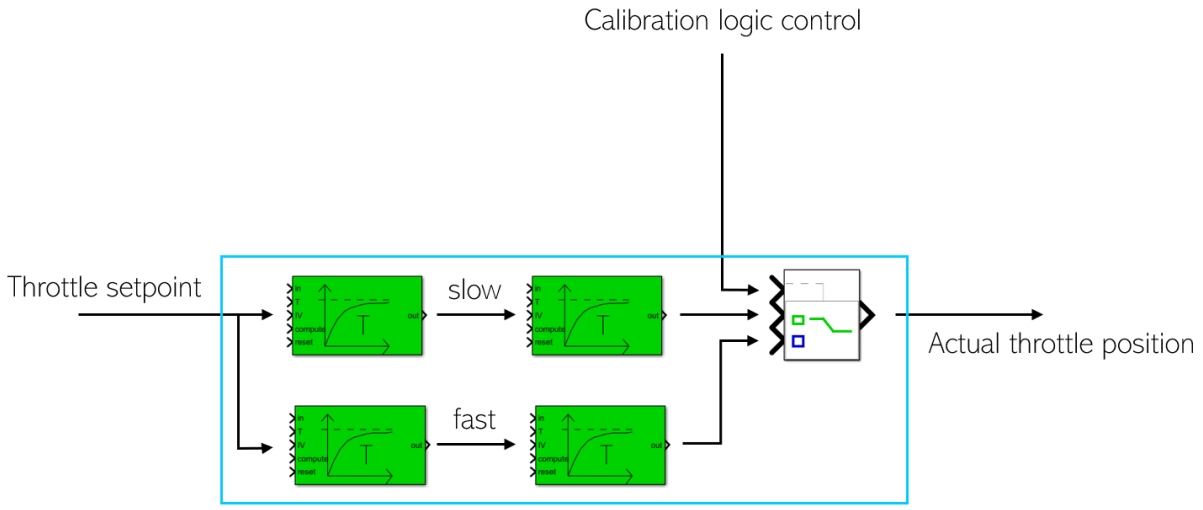


Figure 76, throttle angle setpoint blocks scheme

At the first running attempt, a long transient affected the first seconds of the run due to wrong filters initial conditions.

To reduce the transient entity, a delay-one-step block is used to reset the lowpass filters on-the-fly. The initial condition of the delay is *true*, while in the next simulation steps it outputs *false*. *Figure 77* shows the implementation of the workaround and the delay logic output depending on the simulation steps.

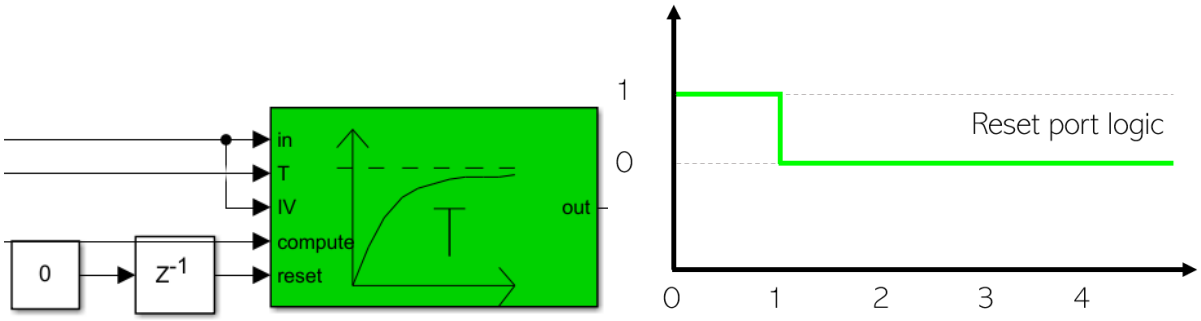


Figure 77, first-step-reset implementation

Figure 78 shows the output comparison before and after the fixing.

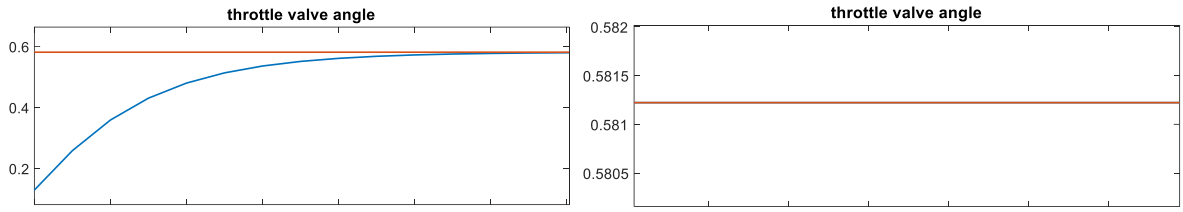


Figure 78, before and after the fixing

Nominal throttle angle calculation

The input of the previous subsystem, the nominal throttle angle, is calculated in this module with the conversion from the target throttle mass flow to the target blade setpoint. The module relies on the inverse throttle model.

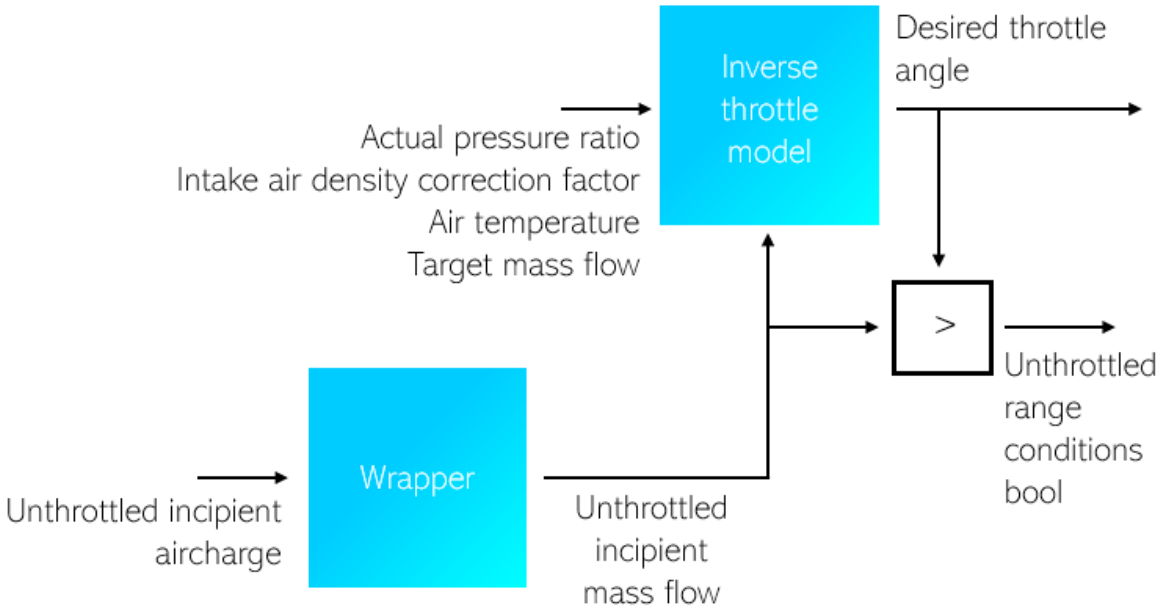


Figure 79, nominal throttle angle blocks scheme

The error the model makes is very weak ($|< 10^{-2}\%|$)

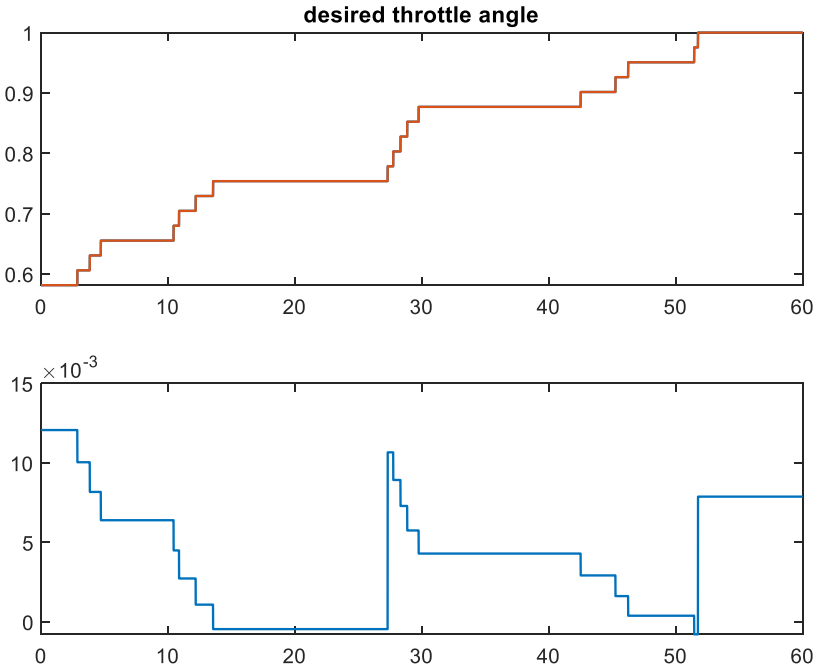


Figure 80, desired throttle angle blocks scheme

Throttle angle setpoint for physical actuator reliability

For increasing the lifespan of the throttle actuator, the desired throttle angle is smoothed by a moving average filter, active for small changes of the target load. If the difference

between the actual desired charge and its previous average value is smaller than a calibration threshold the blade angle does not change.

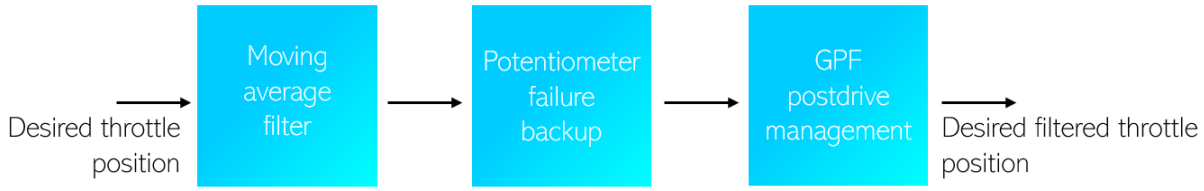


Figure 81, throttle setpoint for actuator reliability blocks scheme

The initial transient of the validation plot is due to null initial condition of the moving average filter. For instance, if the filter considers a four-element buffer, in the first simulation step, its values are null at every index. It needs four simulation steps to fill the array completely.

Charge control assembly

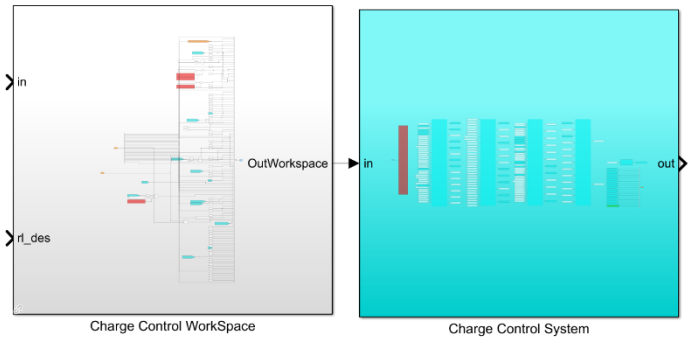


Figure 82, charge control Simulink implementation

The assembled block is therefore available in the Simulink library for smooth integration in the final model.

4.2.3 Charge set system

Desired charge to desired intake manifold pressure conversion

The module requires the target relative charge value from the torque structure, which provides the input according to the gas pedal setpoint. The output is the target intake manifold pressure.

Figure 83 shows the block structure and the inverse charge determination.

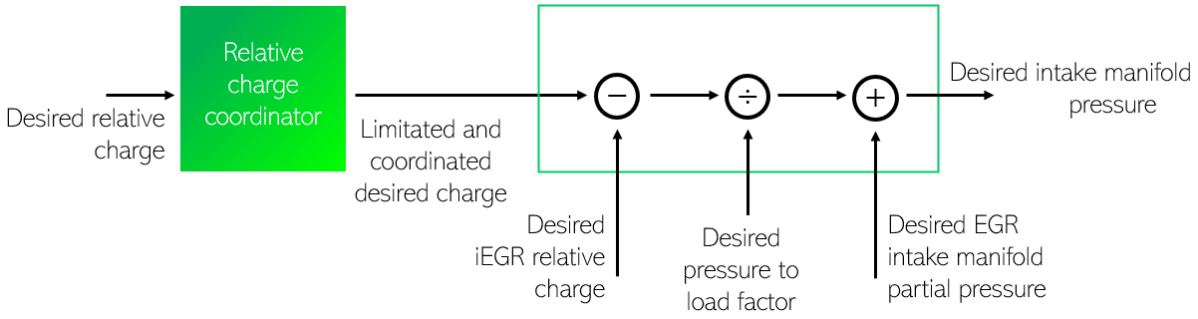


Figure 83, desired charge to desired pressure conversion blocks scheme

Figure 84 shows coherent validation results.

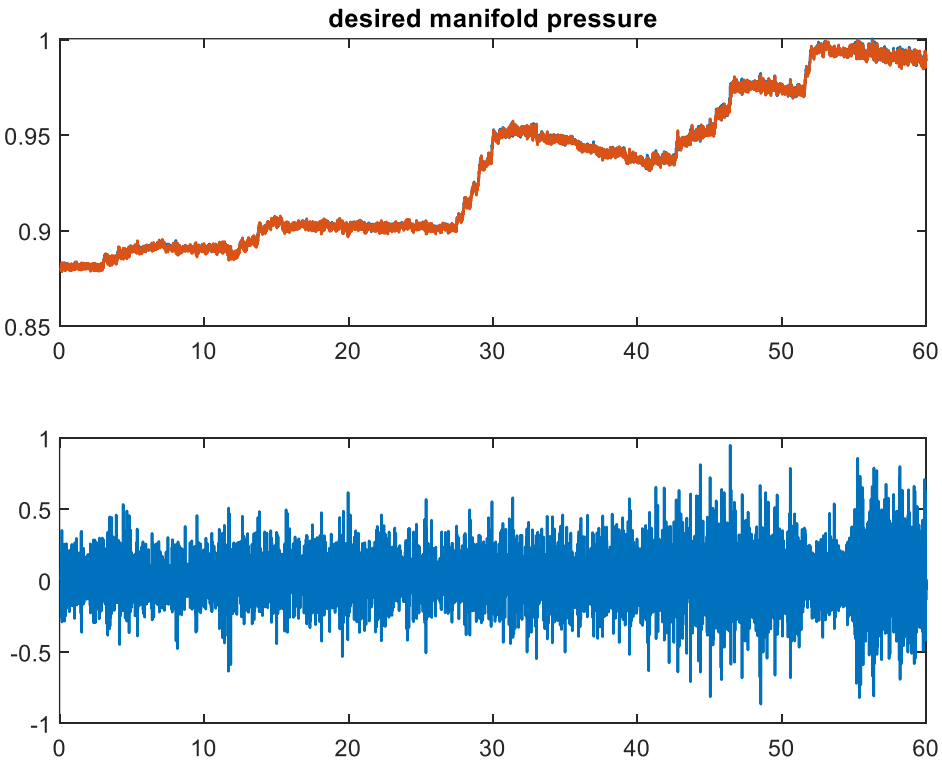


Figure 84, desired intake manifold pressure validation

Charge set assembly

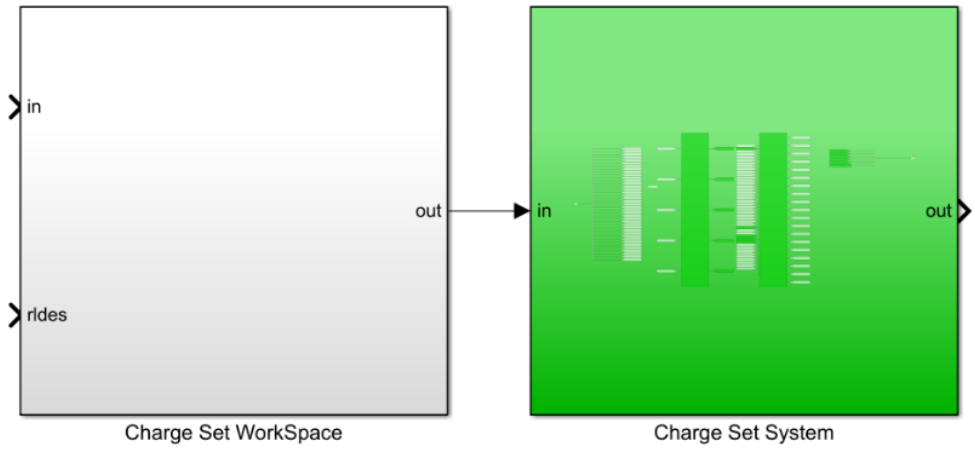


Figure 85, charge set Simulink implementation

Finally, the system is also provided in the form of connected block like the previously validated structures.

4.3 Boost task

Unlike the charge task, a validation dataset for the boost control was not completely available yet. This chapter will describe the modeling activity of the boost functions and the validation process when available.

4.3.1 Signal prioritization for boost pressure actuator

The module constitutes the boundary of the boost control software as it directly outputs the actuator setpoint. It considers all the different contributions, like the catalyst heating occurrence, the gearbox shifting requests, the launch-control and so on. Moreover, it chooses which path should be used between open-loop and closed-loop control. Finally, it limits the control signal according to the calibration offset.

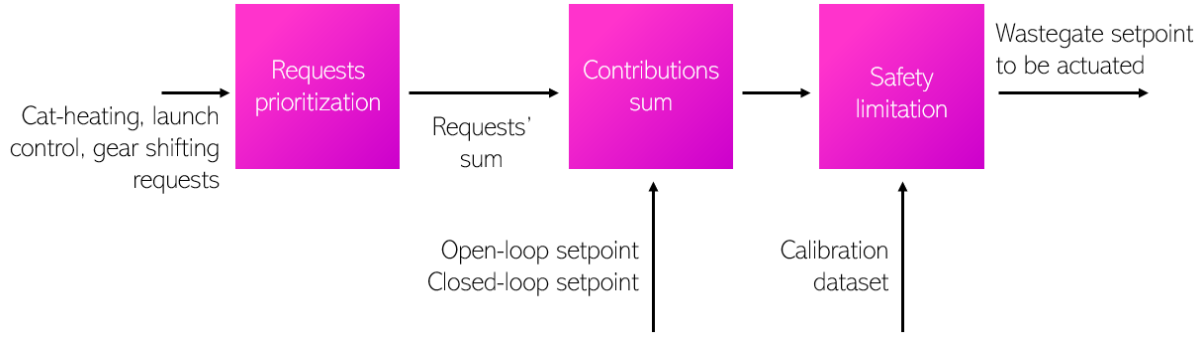


Figure 86, signal prioritization blocks scheme

4.3.2 Limitation of the maximum actuator position

The subsystem provides the maximum value for the actuator. It uses both model-based and map-based approaches. After the preliminary limitation according to the exhaust pressure, the operating point, and the standard conditions, the actuator signal is limited by the maximum allowed turbocharger speed.

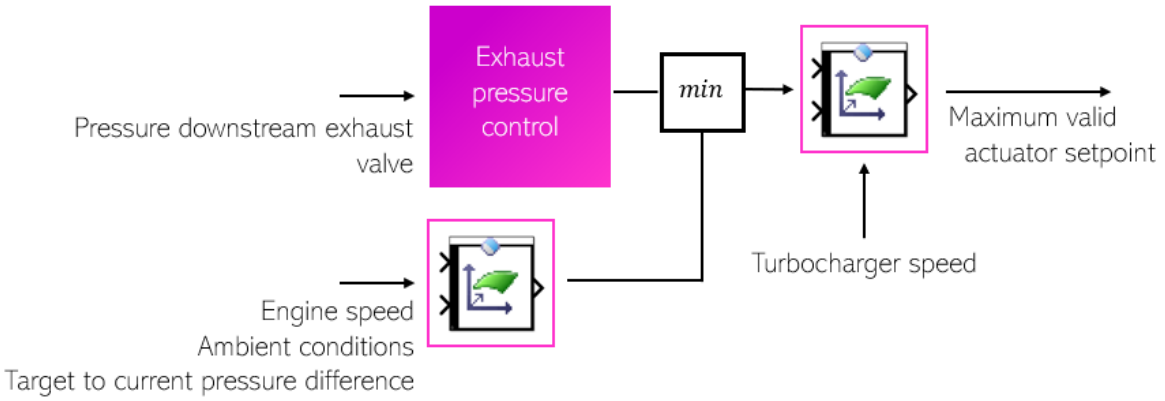


Figure 87, maximum actuator position blocks scheme

4.3.3 Actuator position forced release

To avoid rattling or unnecessary movement of the boost actuator, its position must be forced at certain engine operating points. According to vehicle and engine speed, load and injection control, the release logic is computed.

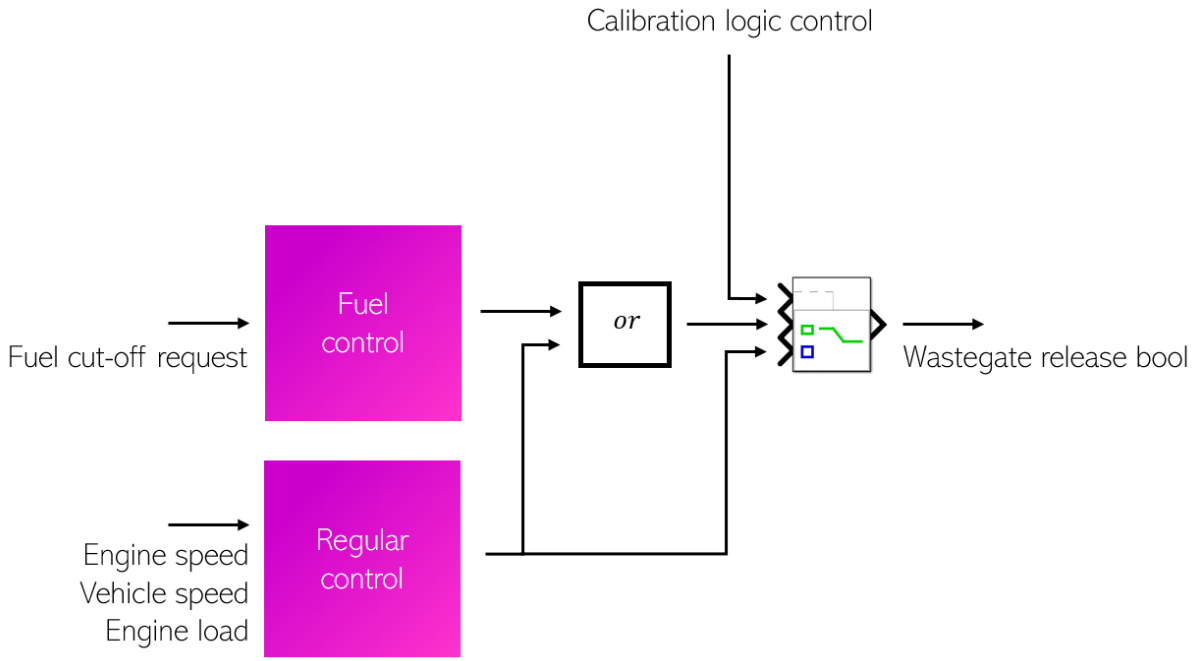


Figure 88, actuator position forced release blocks scheme

4.3.4 Model-based feedforward boost control

According to the actual and the desired engine setpoint, the module calculates the open-loop turbine actuator position through the physics-based approach. It consists of the physical desired mass flow wastegate equation provided with 4.3.4.1 formua.

$$\dot{m}_{TrbnDes} = \frac{\dot{m}_{CmprDes} \cdot c_{p\,Air} \cdot T_{CmprUs} \left(\left(\frac{p_{CmprDsDes}}{p_{CmprUs}} \right)^{\frac{k_{air}-1}{k_{air}}} - 1 \right)}{c_{p\,Exh} \cdot T_{TrbnUs} \cdot \left(1 - \left(\frac{p_{TrbnDs}}{p_{TrbnUs}} \right)^{\frac{k_{exh}-1}{k_{exh}}} \right)} \cdot \frac{1}{\eta_{Cmpr} \cdot \eta_{Trbn}} \qquad 4.3.4.1$$

Figure 89 shows the overall structure of the subsystem.

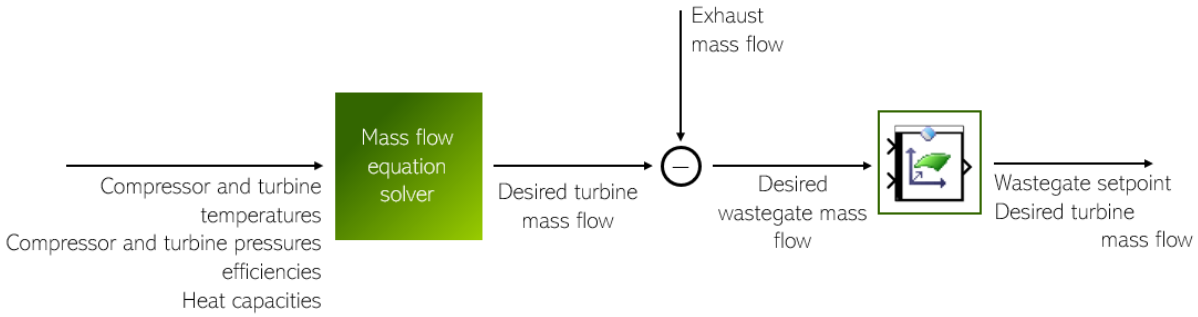


Figure 89, model-based feedforward blocks scheme

Figure 90 shows the validation plot of the signal, which is generally satisfactory apart from the two error peaks above the threshold due to sharp signal steps occurrences.

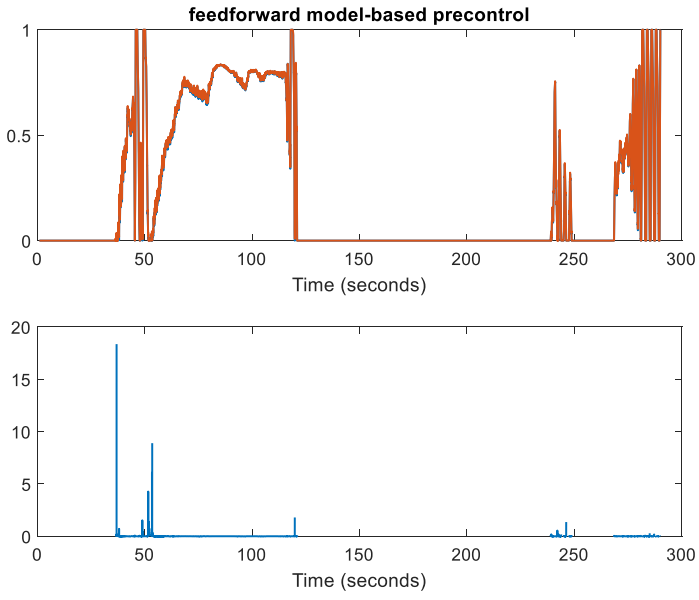


Figure 90, feedforward model-based precontrol validation

4.3.5 Map-based feedforward boost control

The module performs the same functions of the 4.3.4 subsystem through map-based approach. The simple structure consists of a lookup table that interpolates the actuator position depending on the engine speed and on the ratio between the desired intake manifold and the upstream compressor pressures. The prioritization module switches between map-based and model-based subsystems according to the calibration value.

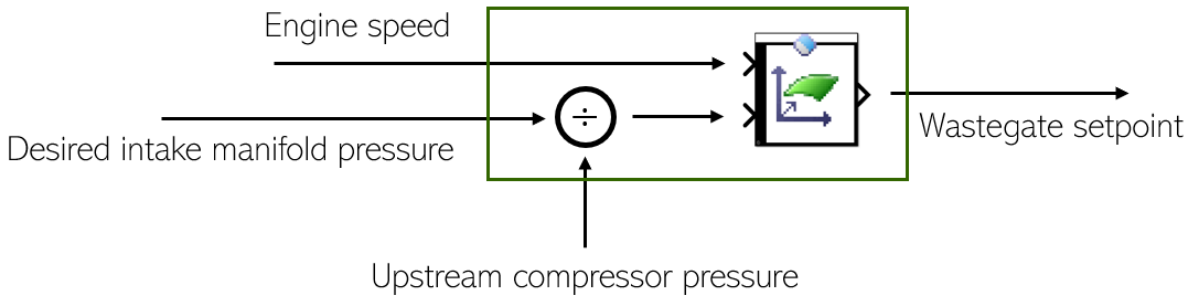


Figure 91, map-based feedforward blocks scheme

Validation plot is reported in *figure 92*. The error trace is generally satisfactory except for the final part of the acquisition, in which the numerical instability due to almost zero signal values makes the error higher.

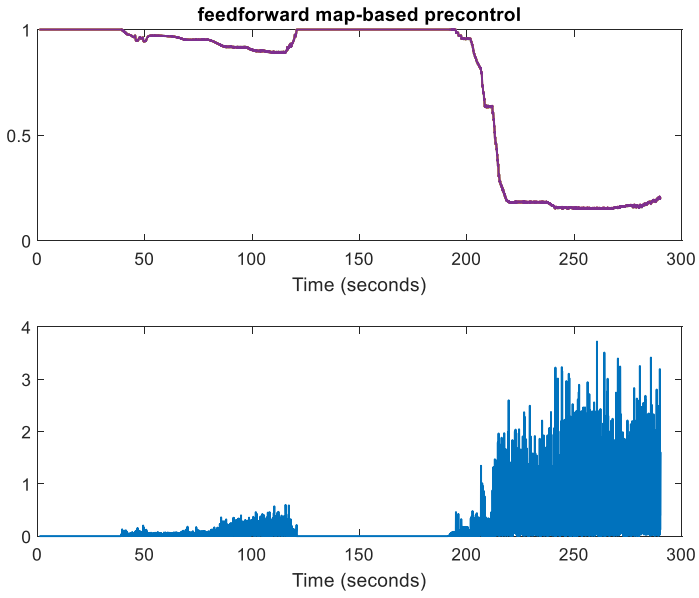


Figure 92, feedforward map-based precontrol validation

4.3.6 Throttled wastegate setpoint control

The module switches between look-up tables for calculating the actuator setpoint in throttled operations. The boost control requests to the compressor a pressure value greater than

atmospheric conditions even during throttled occurences. This software feature helps to reduce the torque build-up lag in turbocharged combustion engines.

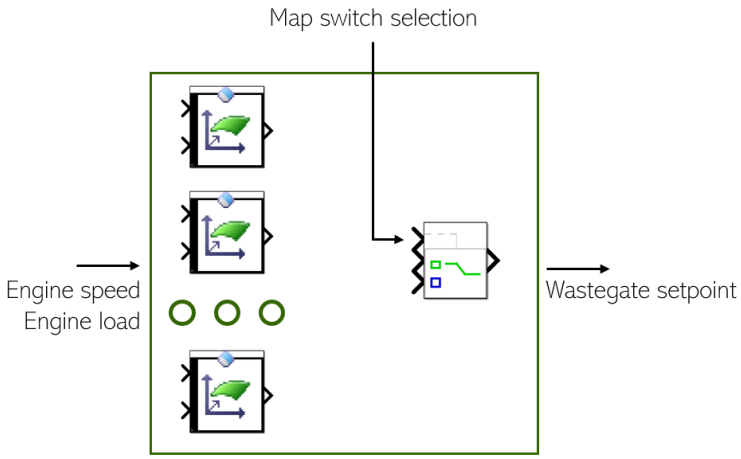


Figure 93, throttled wastegate setpoint calculation blocks scheme

4.3.7 Closed-loop controller

The second contribution to the actuator setpoint comes from the closed-loop control blocks. If the feedforward path is calibrated to reach the intake manifold target pressure, the closed-loop correction allows improving pressure build-up time, shaping the transient response, and adapting the control to engine aging.

The model is based upon a proportional-integral controller, which outputs a wastegate correction from the difference between target and measured pressure.

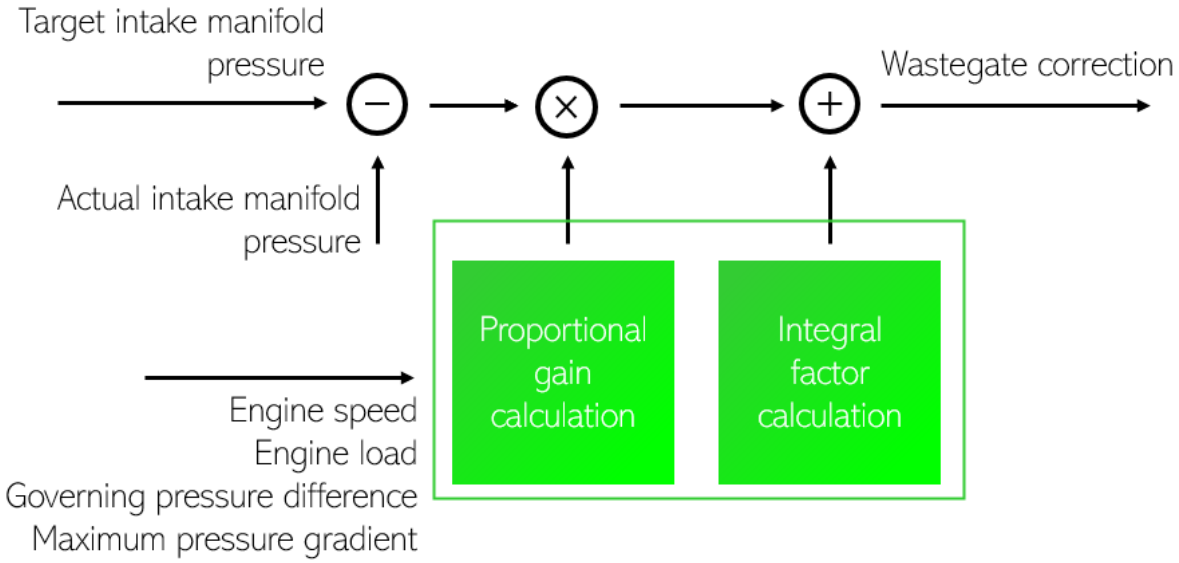


Figure 94, closed-loop controller blocks scheme

Figure 95 shows the validation report.

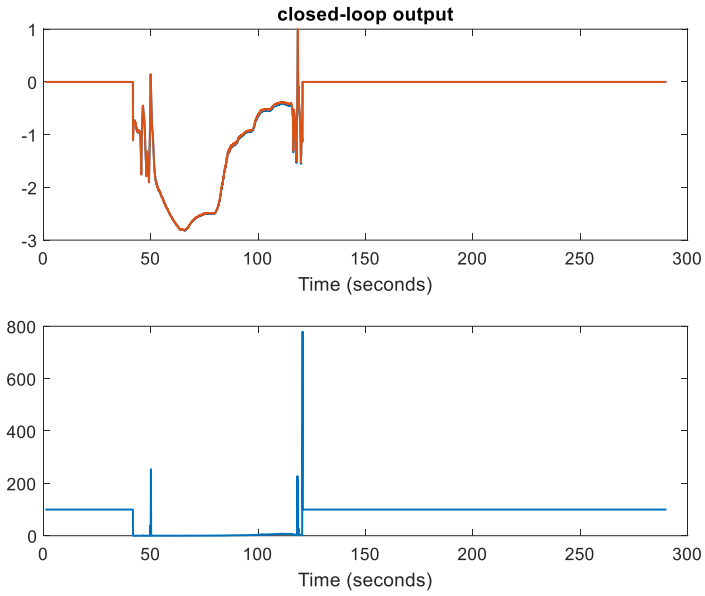


Figure 95, closed-loop correction validation

Here, large numerical instability impacts on the error trace when the signal tends to zero. Even small offset between the model and a null validation signal generates a huge error. In this case, when the validation dataset outputs null value, the model outputs -0.005, which can be considered a numerical zero even if the fictitious error value is close to 100%.

4.3.8 Charge control strategy selection

The subsystem allows choosing between the conventional pressure-based and the speed-based boost control according to the calibration dataset.

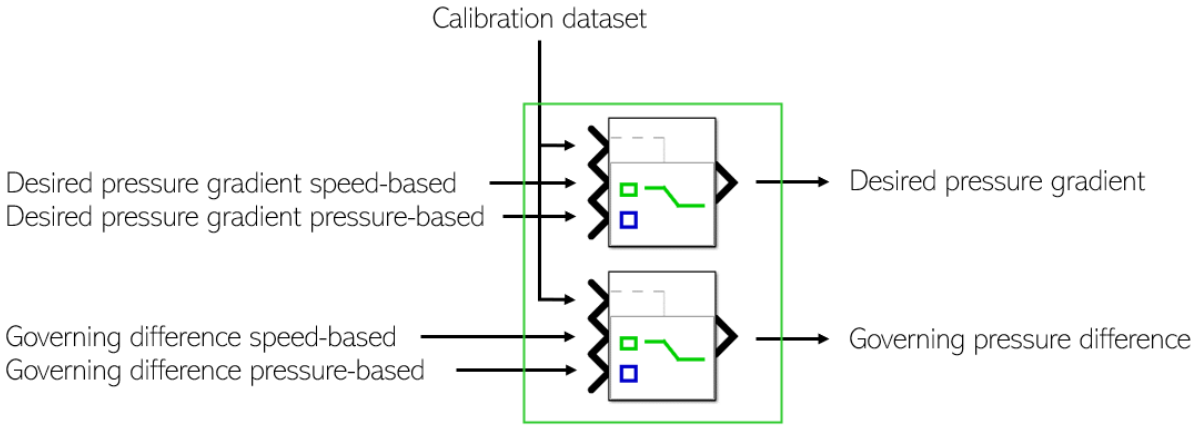


Figure 96, charge control strategy selection blocks scheme

4.3.9 Compressor model

The model produces physical quantities relative to the compressor device through both model-based and map-based approaches. It calculates the isentropic efficiency, the turbocharger speed, and the pressure loss across the intercooler device.

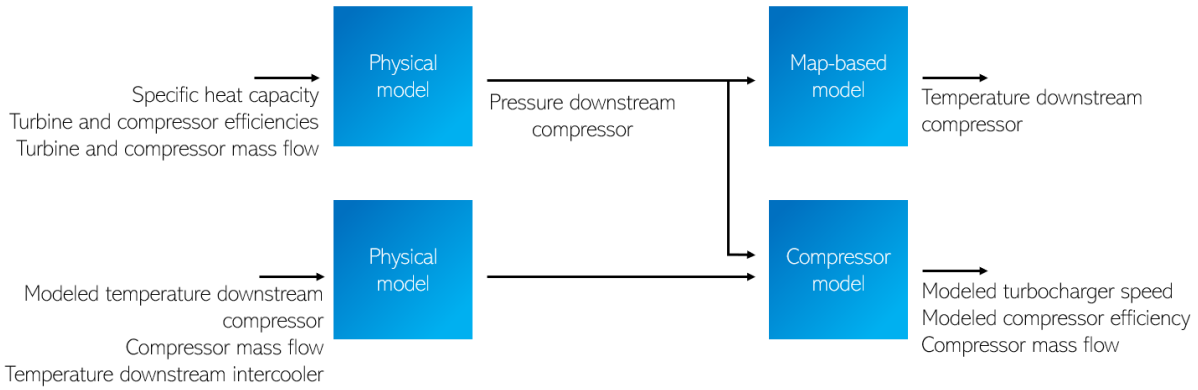


Figure 97, compressor model blocks scheme

4.3.10 Turbine model

The model calculates the isentropic efficiency of the turbine from its actual speed and its pressure through map-based approach. The maximum turbine safety factors and the temperature downstream are also computed.

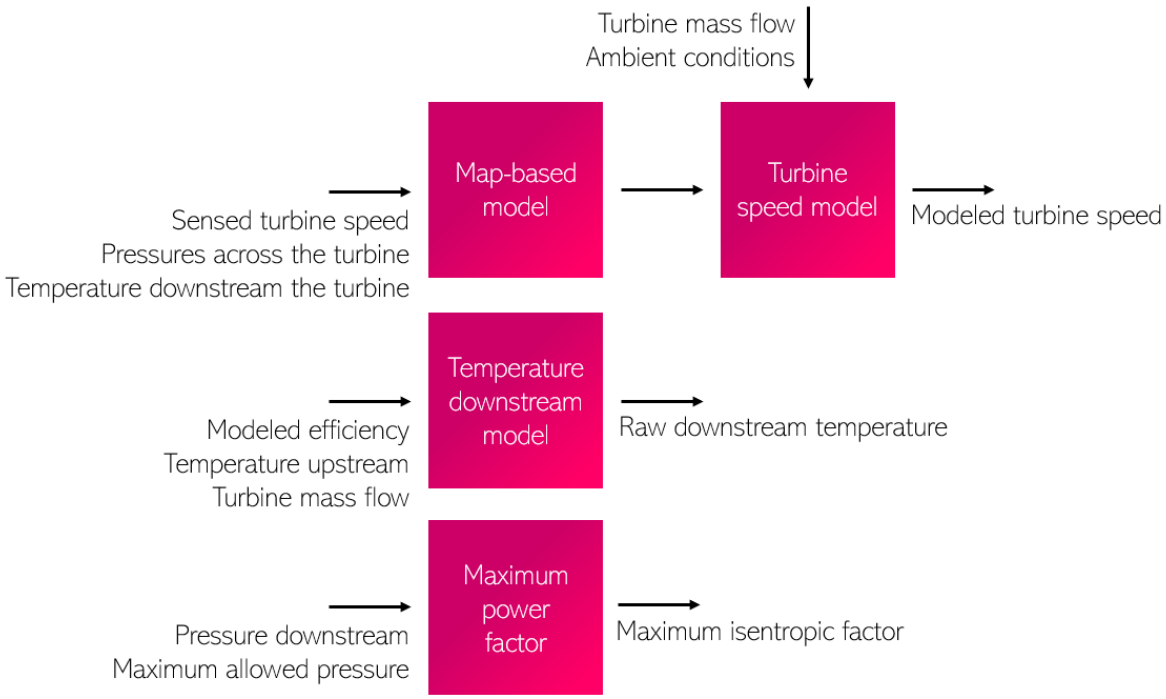


Figure 98, turbine model blocks scheme

4.3.11 Exhaust manifold pressure model

The model contains a physical representation to compute the exhaust gas back pressure. The inputs are the exhaust mass flow rate, the exhaust manifold temperature, the turbine downstream pressure, and the wastegate position.

The physical pressure module is codified in the reference paper as a *C++* function. Due to implementation difficulties, the model was made available in the library with two versions:

MATLAB code translation and block language interpretation. The two alternatives produce coherent outputs with given inputs.

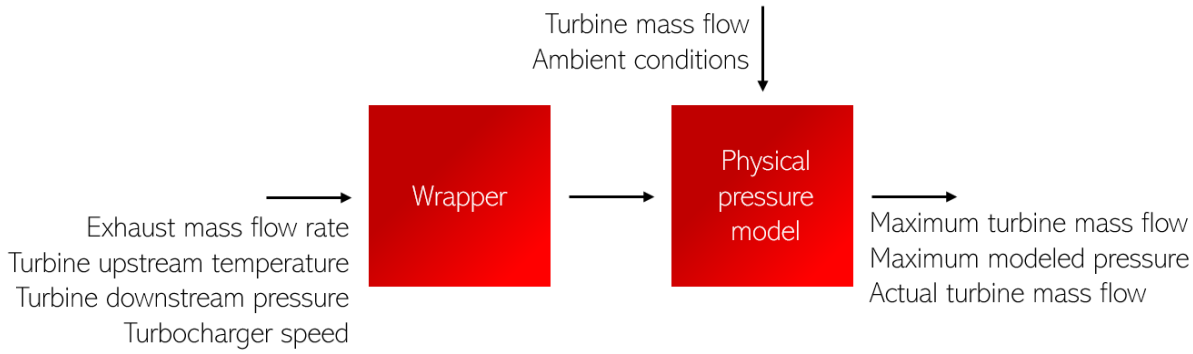


Figure 99, exhaust manifold pressure model blocks scheme

Figure 100 shows the validation report for the exhaust manifold pressure signal.

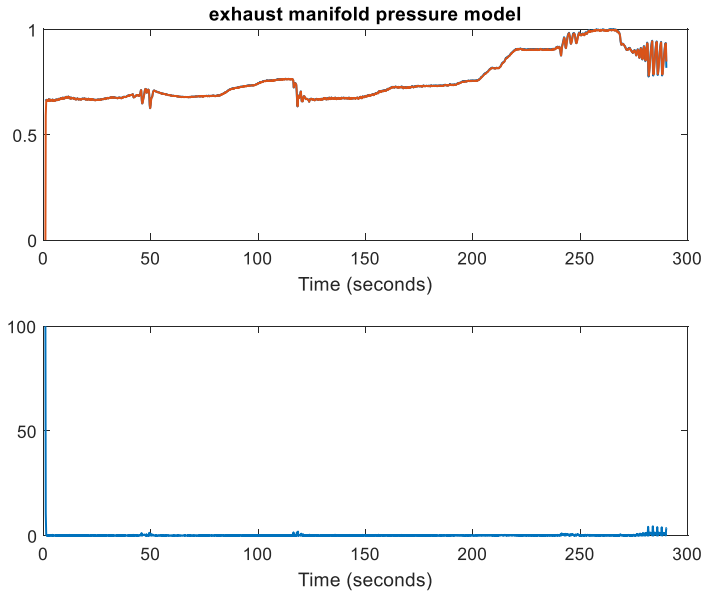


Figure 100, exhaust manifold pressure validation

Apart from the initial transient which the artificial *algebraic loop* initialization is responsible for, the validation error trace stays below the critical threshold.

5

Model-in-the-loop environment

The result of the previous activities is a Simulink block suite that comprehends several controls concerning all the engine air path, from the intake to the exhaust.

The main purpose of the library is providing an ECU emulation environment to virtually perform pre-calibration activities and to get faster and safer bench test campaigns.

Hence, the plant model is added constituting the model-in-the-loop environment. In particular, the Simulink logic acts on the virtual engine model, which reacts to the control requests providing in turn the necessaries inputs, as represented in *figure 101*.

For this purpose, the engine simulation is performed through GT-Power suite. The software allows fast Simulink integration and accurate engine performances simulations.

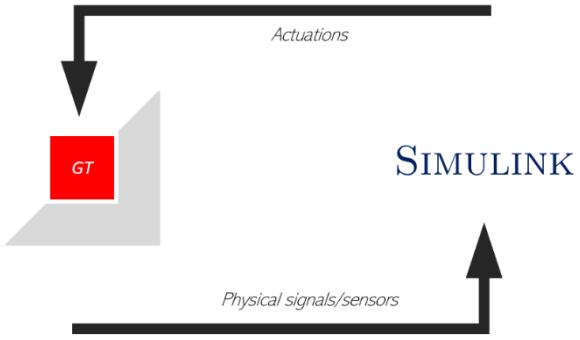


Figure 101, model-in-the-loop Simulink/GT-Power

5.1 GT-Power / Simulink set-up

GT-Power consists of a block-based drag-n-drop interface like Simulink and it is specifically designed to reproduce physical phenomena. The software is used in this activity to predict the combustion engine performances like power, torque, emissions, and fuel consumption.

5.1.1 Fast running model

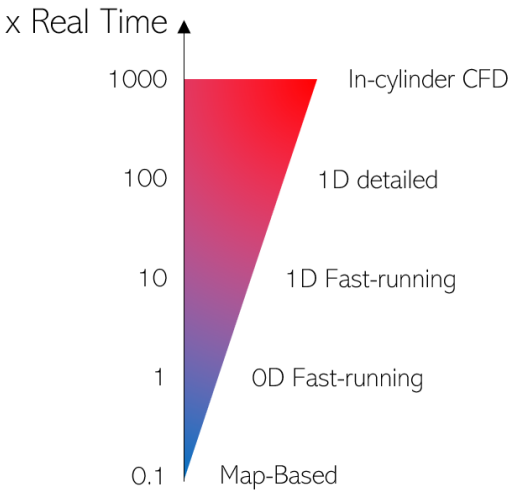


Figure 102, simulation types

GT-Power software covers a wide range of engine simulations complexity, from 1D detailed to map-based mean values models.

In-cylinder 3D flow simulations are, typically, computationally intensive and performed in an external software environment. Because of their complexity a single cylinder cycle is usually simulated and, therefore, the purpose of the model-in-the-loop simulation can't be achieved.

1D simulations aim to increase the computational efficiency by reproducing the engine systems through mono-dimensional flows while preserving the correlation with the CFD benchmark like in-cylinder pressure trace or knocking occurrences.

The simulation speed is affected by the timestep size and the number of calculations per each one. Typically, in the early engine design process, a detailed 1D model is compiled for accurately capturing as much as possible the physical bond. This model is precisely calibrated according to experimentally derived data for pressure losses in pipes, combustion phasing, and thermal models.

When the development phase proceeds and a large number of runs is needed for calibration purposes, a fast-running model is typically obtained from the detailed reference by:

- Reducing the number of calculations per timestep through greater pipe discretization length, lower volumes number, lower cylinder number (especially in vee engine configurations)
- Increasing the step size while respecting the Courant condition for appropriate convergence.

Both decreasing the calculations number and increasing the step size, give the so called Fast-Running-Model (FRM) that is still able to capture a coarse but realistic gas dynamic of the engine and a good correlation with experimental and detailed derived data.

Figure 103 expresses graphically the conversion from the detailed 1D to the fast-running model.

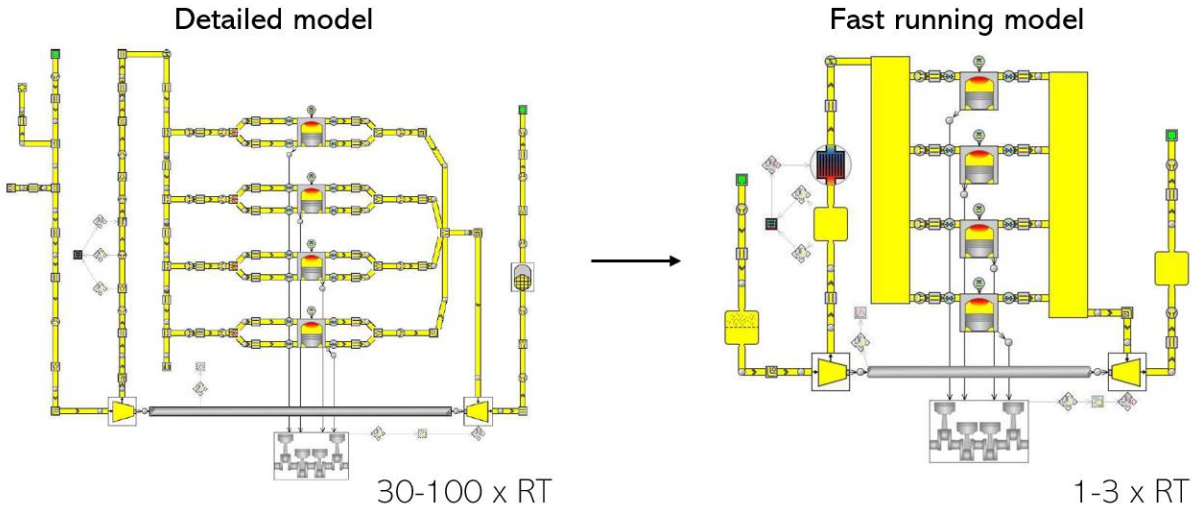


Figure 103, from detailed 1D model to fast-running-model

The simulation execution speed is measured as the ratio between the time that the software needs to complete the run and the simulated time.

A detailed engine simulation typically lasts from 30 to 100 times the simulated duration. A fast-running-model is typically 100 times faster while producing suitable results for calibration purposes.

The model which this dissertation will focus on is a fast-running-model while keeping some detailed characteristics such as the predictive turbulent combustion. The model is still able to respond to spark advance adjustments and give information about knocking occurrences and cycle to cycle variability.

On the other side, heat exchange models are substituted with imposed wall temperature like coolant and oil temperature and calibrated according to 1D detailed simulations.

5.1.2 GT/Simulink simulations setup

There are two ways in which GT-Power and Simulink can be linked: *GT as lead* or *Simulink as lead*. In the first case, GT solver manages the convergence criteria and the simulation running, while Simulink model is compiled through MATLAB coder and linked as external *C++* library.

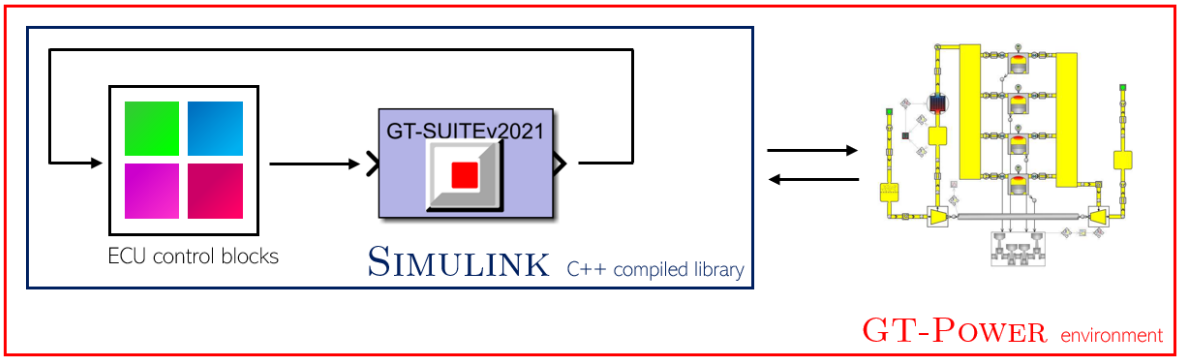


Figure 104, GT-Power as lead

In *Simulink as lead* case, the GT-Power model is linked into the ECU model as a compiled *S-Function*. Simulink imposes the simulation duration and the timestep.

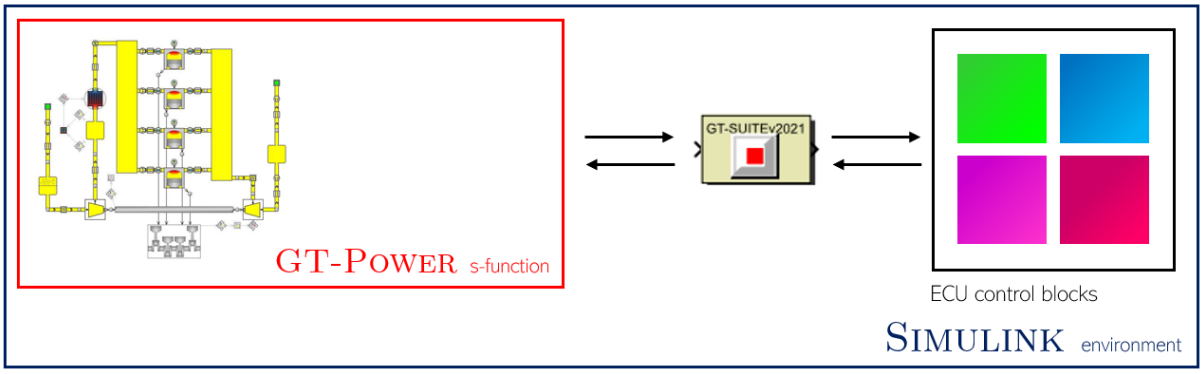


Figure 105, Simulink as lead

The current activity uses the second option as it is focused on the control tuning.

5.1.3 Simulink mask

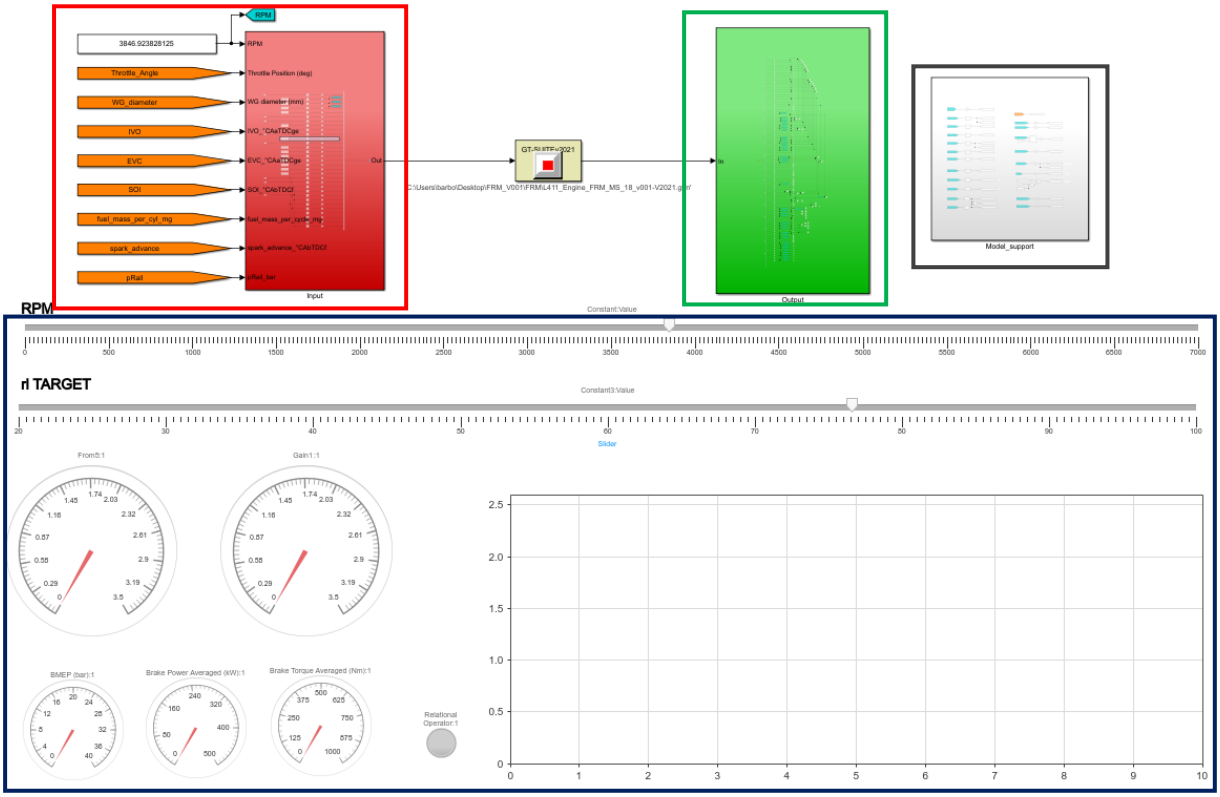


Figure 106, Simulink mask

Dashboard (blue square, *figure 106*)

The dashboard lets understanding how the model is behaving and interacting with imposed values.

The two bigger gauges measure the target and the actual intake manifold pressure, giving the estimation of the transient duration and the control targeting precision. The three smaller gauges display the break mean effective pressure, the brake torque, and the brake power, giving an overview of the actual engine performances. The chart on the right plots the trace of the target and the actual intake manifold pressure to appreciate the engine time response. In the middle of the dashboard, a lamp reports when the knock induction time integral is greater than 1, which means that knocking combustion is occurring.

The two upper sliders allow the user to impose the desired engine speed and load.

Inputs generation (red square, *figure 106*)

While boost control provides its actuator setpoint, all the other control actuations must be emulated within the Simulink interface to get the engine running properly.

The required values are:

- Throttle valve angle
- Intake valve opening angle
- Exhaust valve closing angle
- Start of Injection
- Fuel rail pressure
- Spark advance
- Fuel mass to be injected per cycle

The throttle valve control is implemented with a proportional-integral controller, as *figure 107* shows.

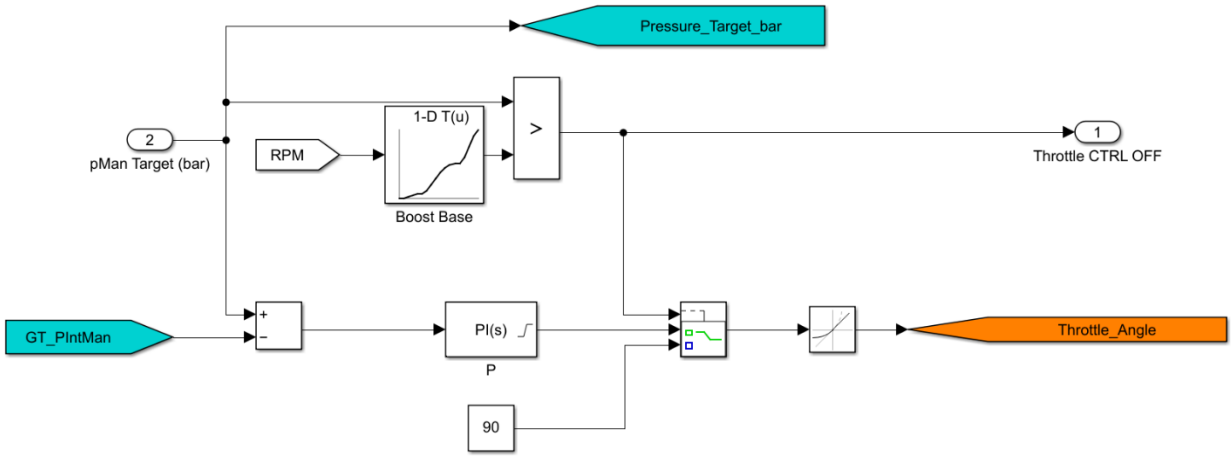


Figure 107, throttle control

Target intake manifold pressure greater than the base boost pressure means that the throttle valve should be completely opened allowing the control in boosted mode, the switch is set to *true*, and the maximum value of throttle is output. When the target intake manifold pressure is lower than the base boost pressure, the wastegate is completely opened and the control is performed through the throttle blade angle. In this case, the switch chooses the proportional-integral controller output, which is based upon the difference between the target and the actual intake manifold pressure.

For VVT phasing, intake and exhaust cam maps were provided, both depending on the engine speed and the actual relative charge. The maps were implemented with the lookup table Simulink block as the measuring unit is the same of GT (after top dead center gas exchange).

Figure 108, VVT setpoints

The rail fuel pressure actuation is computed like VVT maps, but the unit conversion must be added to be GT-Power compliant.

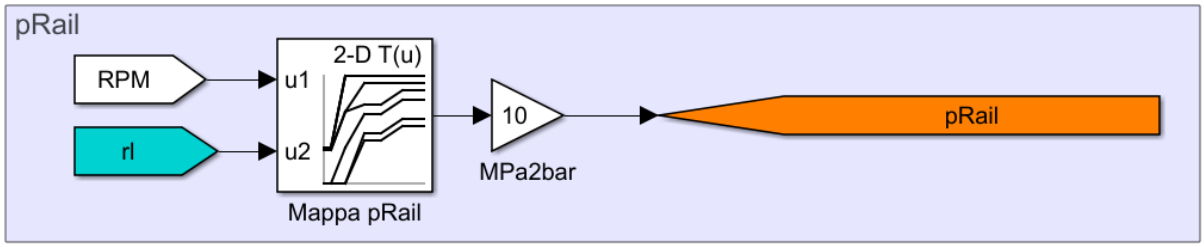


Figure 109, rail pressure setpoint

The start of injection angle is provided similarly, but conversion from angle before top dead center firing to angle after top dead center firing must be accounted for.

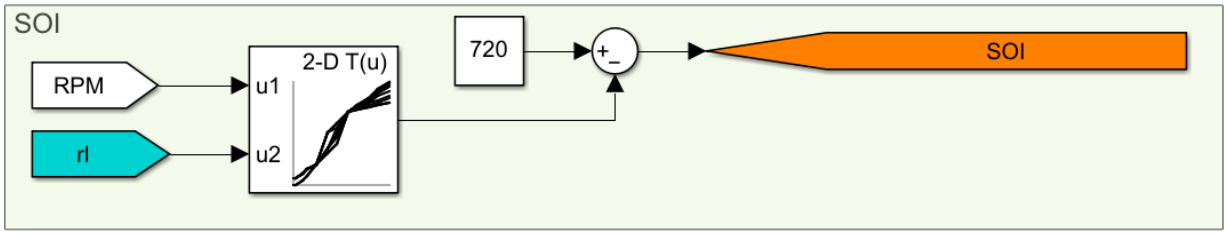


Figure 110, start of injection setpoint

In the same fashion, the spark advance is provided, and the map must be converted from crank degrees before top dead center firing to crank degrees after the top dead center firing. Moreover, the source map for the ignition crank angle derives from the provided base map after the subtraction by an offset map. This is due to the absence in the engine model of the MFB50 GT-Power follower control which accounts for knocking limits. The control scheme is not present due to fast-running-model conversion.

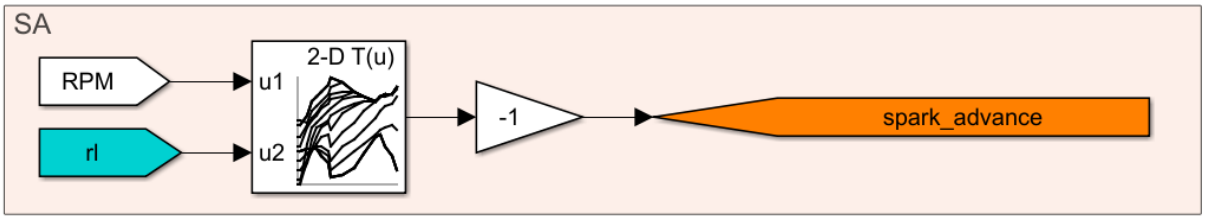


Figure 111, spark advance setpoint

The control scheme to calculate the fuel mass that must be injected per cycle is more sophisticated. It requires computing the relative charge and the lambda target map.

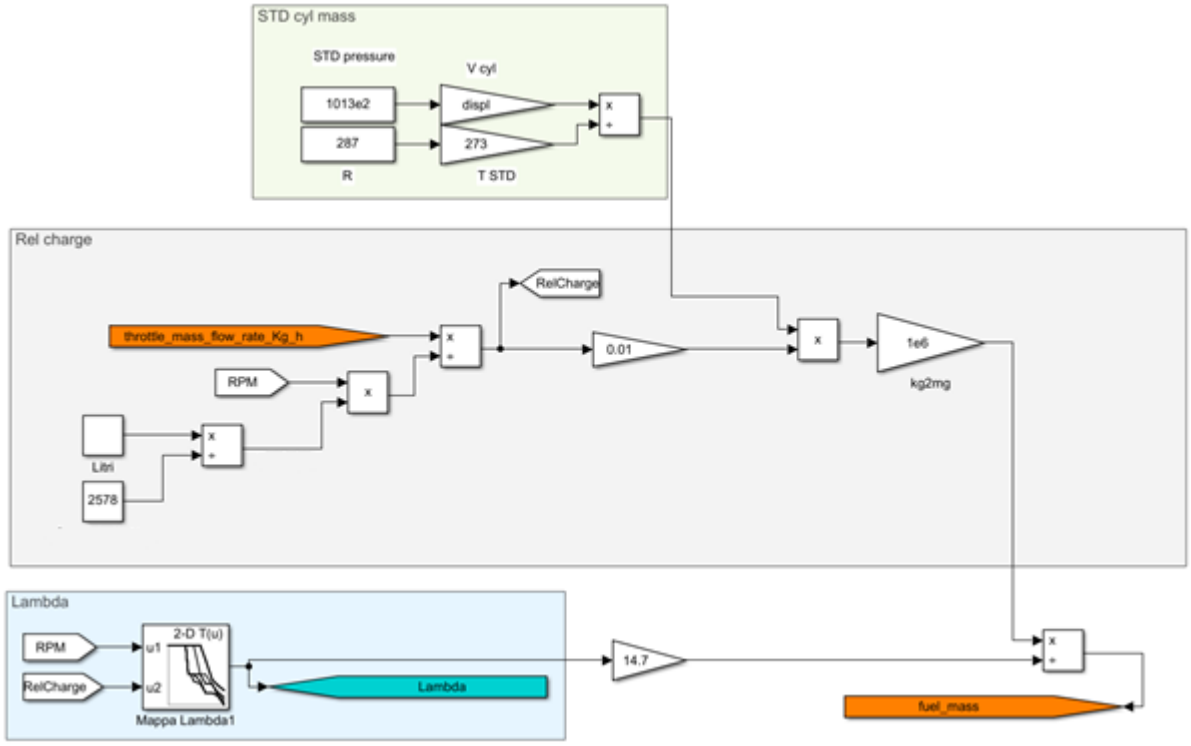


Figure 112, charge and fuel mass estimation

Starting from the middle of *figure 112*, the air mass flow is measured from the engine model at the equivalent throttle location. This value, divided by the multiplication of the actual speed by an engine constant, gives the relative charge. Its multiplication by the standard conditions air mass, according to perfect gas law, gives the actual air charge inside the cylinder.

Lambda value coming from the bottom lookup table, is defined by the *formula 5.1.3.1* which depends on engine speed and load.

$$\lambda = \frac{(air\ mass/fuel\ mass)_{actual}}{(air\ mass/fuel\ mass)_{stoichiometric}} \qquad 5.1.3.1$$

Hence, the definition can be manipulated as formula 5.1.3.2,

$$(air\ mass/fuel\ mass)_{stoichiometric} \cdot \lambda = (air\ mass/fuel\ mass)_{actual} \qquad 5.1.3.2$$

making the fuel mass quantity explicit.

$$fuel\ mass_{actual} = \frac{air\ mass_{actual}}{(air\ mass/fuel\ mass)_{stoichiometric} \cdot \lambda} \quad 5.1.3.3$$

Considering that for standard gasoline engine the air-fuel ratio is expressed as 5.1.3.4 shows,

$$(air\ mass/fuel\ mass)_{stoichiometric} = 14.7 \quad 5.1.3.4$$

the fuel to be injected in the cylinder per engine cycle is determined according to 5.1.3.5 formula.

$$fuel\ mass_{inj} = \frac{air\ mass_{actual}}{14.7 \cdot \lambda_{actual}} \quad 5.1.3.5$$

Outputs reading (green square, *figure 106*)

The output vector of the GT-Power engine model is decomposed through a *demux* Simulink block. Here, filtering and naming operations are performed.

Model support (black square, *figure 106*)

Control blocks request its inputs, some of them are physical sensors measurements and some others are derived signals. The first type can be directly measured by the engine model, the second must be calculated within the Simulink mask. The model support block is composed by spare engine software pieces and acts as a wrapper between external systems and the required signals.

5.2 Boost model-in-the-loop combined simulation

The combined simulation can be split in two phases, boost and charge control implementations as the systems are independent one from each other.

In the first phase the boost control is considered as the engine model needs the pressure actuator control for proper working.

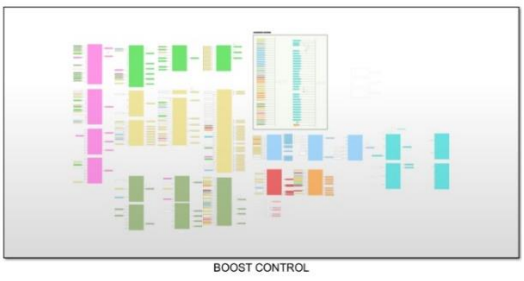


Figure 113, Simulink boost control implementation

5.2.1 Inputs generation

The first step to get the model running properly is the generation of the required boost inputs. There are three types of signals:

- Boolean quantities
- Externally calculated quantities
- Target quantities

Boolean quantities

Boolean quantities specify mainly the current engine operating mode and its topology.

For instance, the following table represents some of them.

Signal description
Variable geometry turbine boost control
Electrical wastegate valve available
Aging adaptation active
Launch control active
Half engine running active
Catalyst heating procedure active

Operation mode assignment is accomplished according to the simulation purpose. In this case, the aim is reproducing stationary and transient engine regular operations during warm conditions so, therefore, catalyst heating and launch control are deactivated.

The topology values are chosen according to GT-Power model and the actual engine configuration.

Externally calculated quantities

Boost control requires many signals coming from external systems. For instance, *p3_Averaged_bar* and *p4_Averaged_bar* are output from the GT-Power engine model. They are filtered and then sent to the boost control unit. Here, they are converted in compliancy with the ECU agreement and the difference is performed generating the desired signal as represented in *figure 114*.

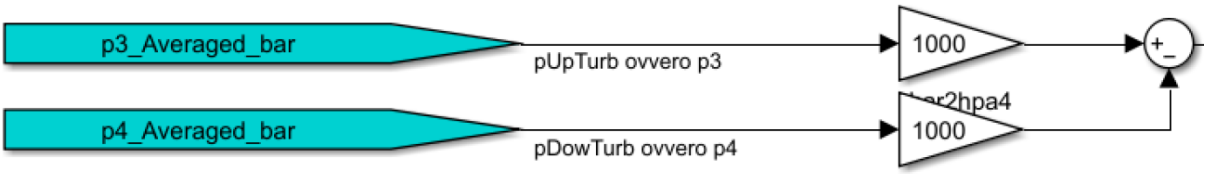


Figure 114, external calculated quantity

Another example of the procedure is the evaluation of the exhaust gasses heat specific capacity. In this case, a 2D interpolation map is used as the value depends on the current lambda setpoint and on the turbine exhaust gasses temperature upstream.

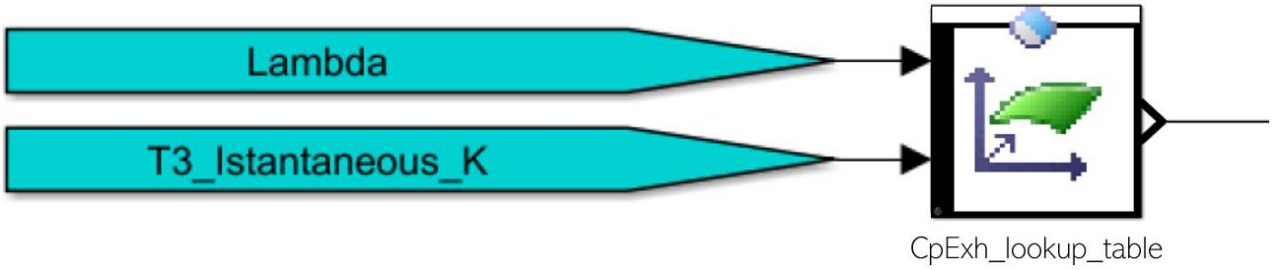


Figure 115, external calculated quantity

Target quantities

Some models rely on target quantities, which are values that the system should assume when the target conditions (engine speed and load) are achieved. Some of them are calculated with external models so they are not available yet and some others are based upon the calibration dataset, which isn't tuned yet. In particular, the required quantities are:

- The target throttle valve mass flow
- The target compressor efficiency
- The target exhaust mass flow
- The target relative charge

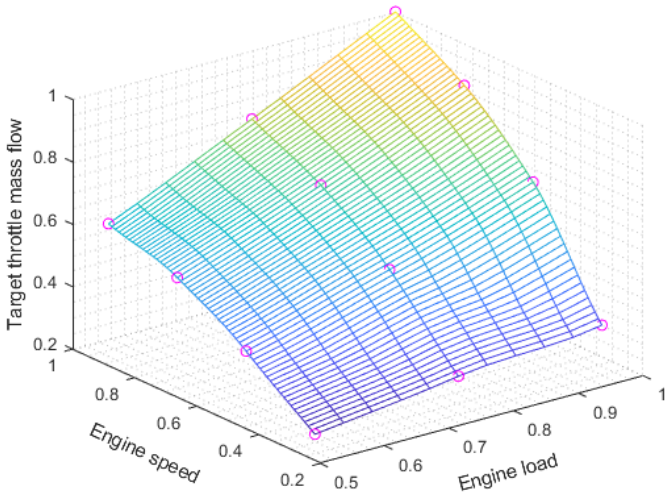
To overcome this issue, a Design-of-Experiments (DoE) was performed with the GT-Power integrated tool investigating the engine operating points in the boosted range. Very coarse discretization is chosen as more control models will be considered reliable as the activity proceeds.

The experiment consists of a full-factorial investigation according to the following grid which is filled with GT-Power measurements directly.

The grid is reported in the following table.

	× Maximum boost pressure			
× Maximum engine speed		0.5	0.75	1
	0.25	-	-	-
	0.5	-	-	-
	0.75	-	-	-
	1	-	-	-

The measured quantities are reported below.



	0.5	0.75	1
0.25	0.201171	0.262342	0.300901
0.5	0.365766	0.502072	0.657658
0.75	0.5	0.670721	0.865766
1	0.570721	0.781081	1

× max (*throttle flow*)

Figure 116, target throttle flow map

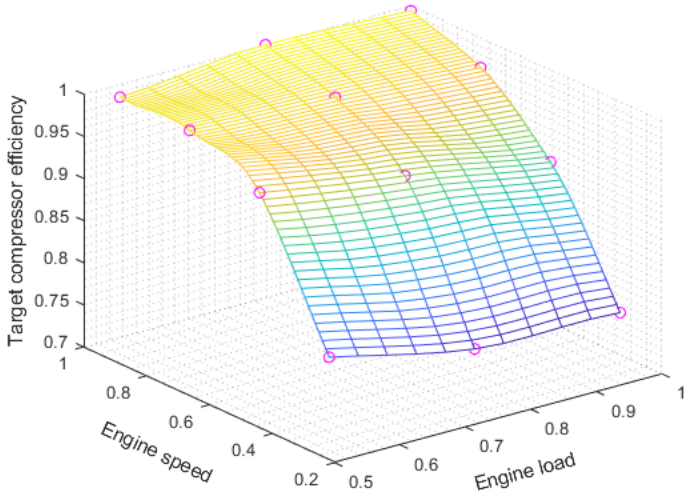


Figure 117, target compressor efficiency map

	0.5	0.75	1
0.25	0.7905	0.7535	0.7497
0.5	0.9476	0.9208	0.8902
0.75	0.9834	0.9757	0.9642
1	0.9847	1	0.9936

$\times \max(compressor\ efficiency)$

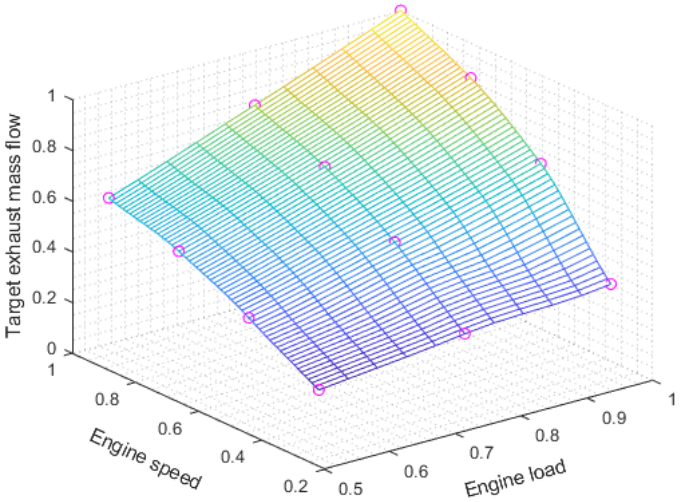


Figure 118, target exhaust mass flow map

	0.5	0.75	1
0.25	0.1979	0.2633	0.3029
0.5	0.3550	0.4963	0.6504
0.75	0.4900	0.6669	0.8608
1	0.5731	0.7825	1

$\times \max(exhaust\ flow)$

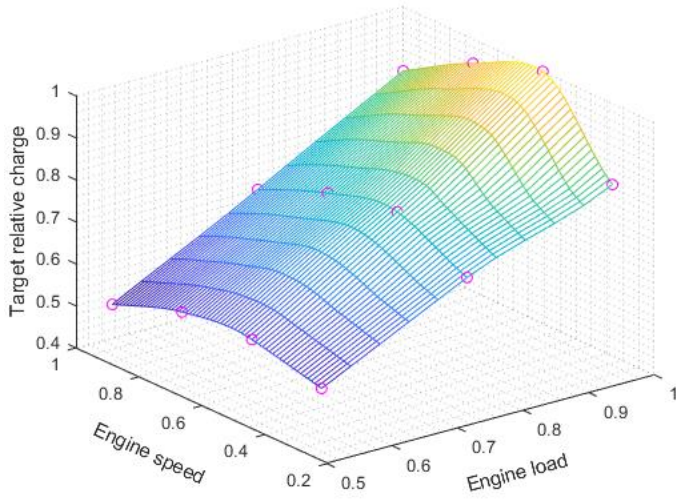


Figure 119, target relative charge map

	0.5	0.75	1
0.25	0.5098	0.6798	0.8072
0.5	0.5497	0.7594	1
0.75	0.5385	0.7276	0.9426
1	0.4801	0.6596	0.8481

$\times \max(target\ charge)$

5.2.2 Simulation run

When all the required inputs dependencies are satisfied, the simulation can run.

Preliminary operations

Even if Simulink can compile the model correctly, the boost control does not achieve properly the boost control due to blank untuned calibration dataset. In particular, the actuator was forced to keep the valve constantly opened for safety concerns during early bench testing.

The first calibration tuning consists of switching from the manually actuated valve to the automatic setpoint control via subsystem codeword. Every subsystem dataset contains a specific setting that typically switches between its signal sources. The value is a decimal-base number that is converted to binary form by *GETBIT* Simulink Block. The digit of the binary representation, at the index specified with the *bit* input port, can be equal to *false* or *true*. The switch, that the bit refers to, actuates the path according to the boolean calibration. *Figure 120* represents the structure.

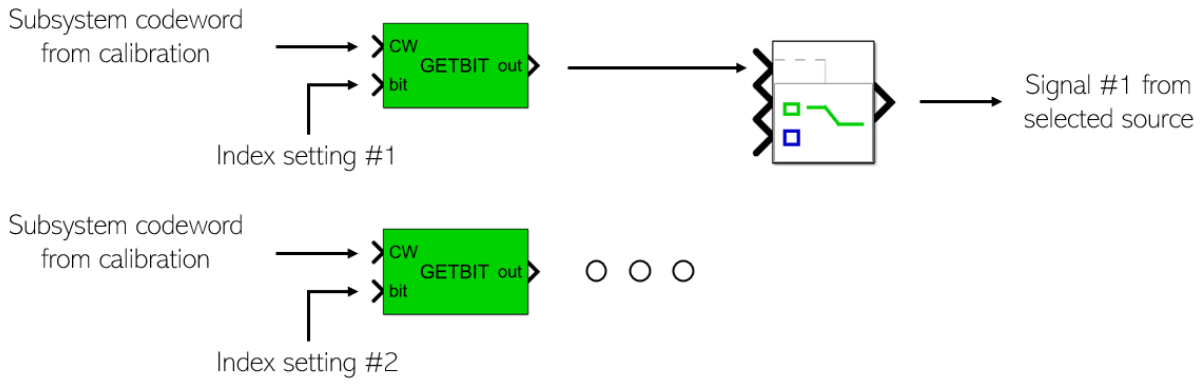


Figure 120, GETBIT logic control

For instance, the modification procedure for setting *false* at index 4 is represented in *figure 121*. The actual calibration is converted into its binary form, the value at the specific index is modified by the user and the binary number is then reconverted back into the decimal representation.

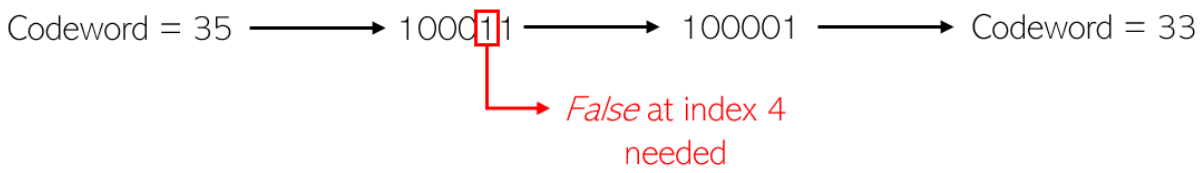


Figure 121, codeword tuning

Once the boost structure correctly controls the actuator, targeting the model in the boost region was not possible yet. The current physical feed-forward pre-control scheme presents a blank 2D map, which gives null setpoint for any inputs values.

To overcome the issue, an experiment was designed similarly to the previously calculated target quantities.

As the map depends on the actual transmission gear and the target wastegate mass flow, the following data was obtained.

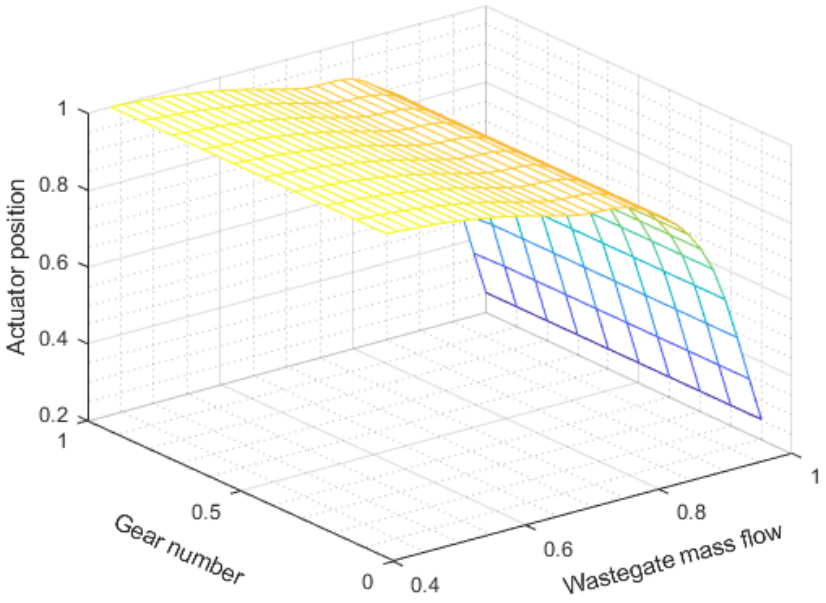


Figure 122, wastegate setpoint map

Other tunings were performed improving the model capabilities and the transient response.

For instance, the switch to the map-based feedforward boost precontrol is tested through codeword value. High boost pressure target was not achievable during the first test.

Another blank 2D lookup table was responsible for the behavior. For interfering as little as possible with the pre-calibration task, it was decided to relax the integral gain limitation of the closed-loop control to let the target being reached through deeper correction. *Figure 123* represents the location of the limiter.

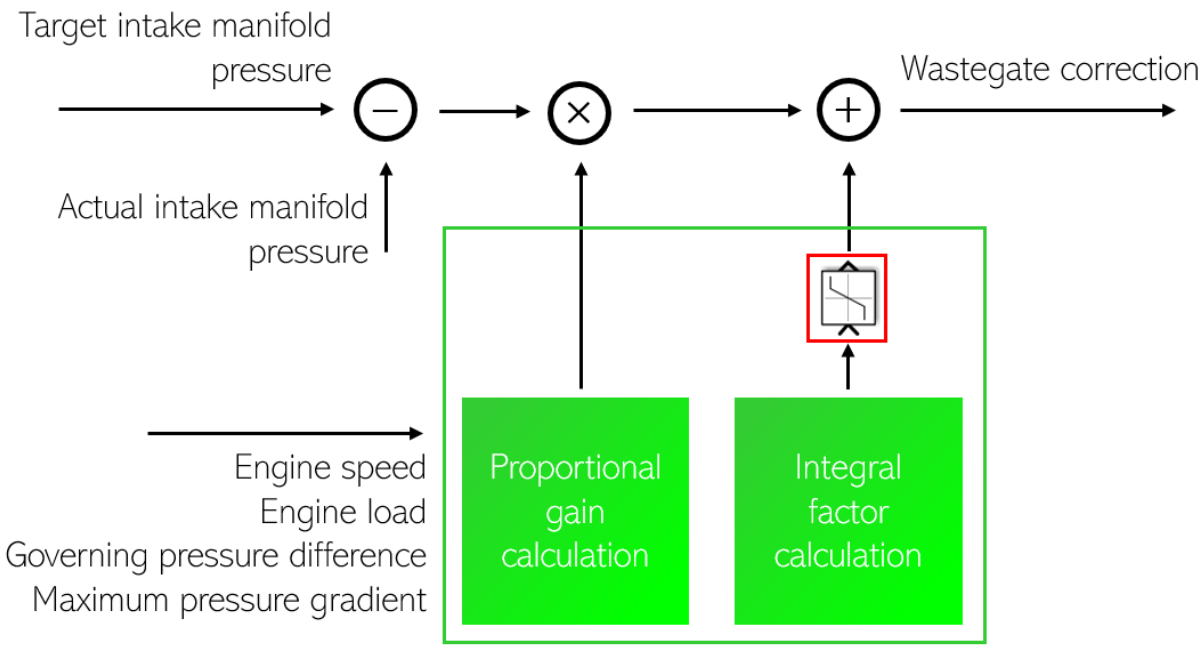


Figure 123, PI controller integral gain limitation

Finally, the simulation can run properly, enabling the possibility to achieve fully virtual pre-calibration of the boost engine control.

5.3 Charge model-in-the-loop combined simulation

The next step involves the integration of the charge control modules. The task has a different purpose than the previous one as the charge control does not contribute to the engine model actuations. All the inputs that GT-Power requires are, indeed, calculated by the provided base lookup tables.

In this case, the activity focuses on the module integration for comparing and cross correlating the charge outputs with the model computed physical quantities.

From this consideration, some assumptions can be made improving the simulation execution speed and the pre-calibration process.

Considering that the engine model will have the same response with the same inputs, it is possible to run a first simulation covering all the engine operating points the user is interested in, while recording the GT-Power model output through a *to workspace* Simulink block.

When the user needs testing different calibrations, he can comment out easily the GT-Power link from the mask while substituting its output with the recordings. The procedure is illustrated within *figure 124* and *figure 125*.

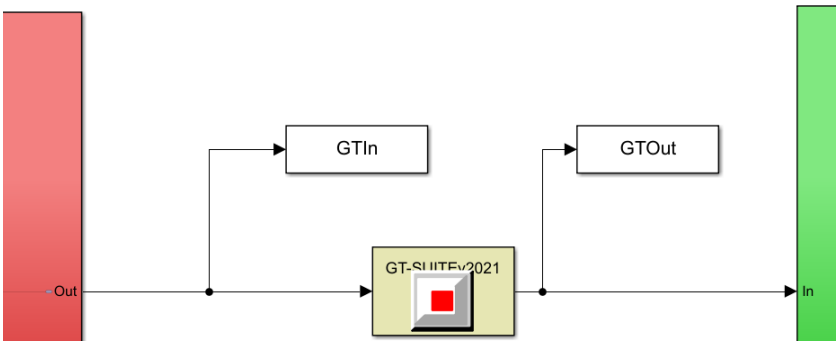


Figure 124, GT-Power recording

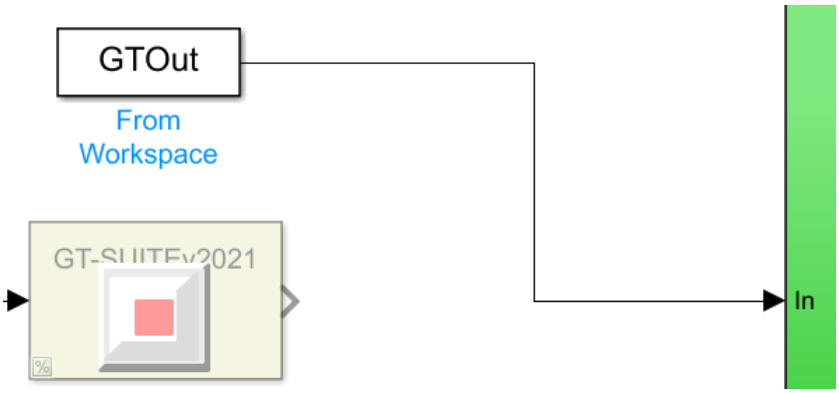


Figure 125, workaround implementation

There are two main benefits with the application of the workaround:

- The huge execution speed improvement
- The possibility to perform the pre-calibration task while the GT-Power software license is busy

The automatic model targets is implemented in the model with a *from workspace* Simulink block which substitutes the slider targets control. MATLAB workspace contains two structures for both engine speed and manifold pressure with two fields each one:

- *X* field, which contains the time vector
- *V* field, which contains the target data that must be actuated in the model at a given *X* time

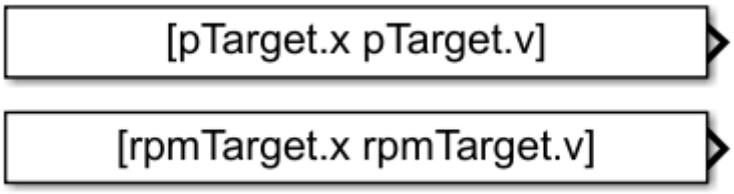


Figure 126, targets Simulink implementation

A example of the output from the *rpmTarget* block is reported in *figure 124*.

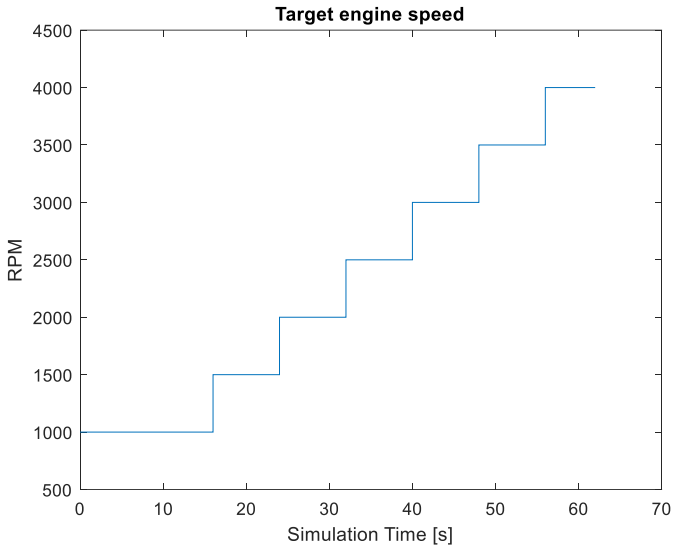


Figure 127, engine speed output example

rpmTarget.x array	*rpmTarget.v* array
0	1000
8	1000
16	1500
24	2000
32	2500
40	3000
48	3500
56	4000

The user can then save in the MATLAB workspace the timeseries of the required model signals similarly to the engine model output. The post-processing of the saved signals occurs with a specifically designed script that takes the mean value of the last 0.5 seconds of the considered signals for each speed and load target step.

The value is then recorded in a table close to its engine speed and manifold pressure target.

Finally, all the operating points where the engine does not reach the desired setpoint or where it exceeds the knocking limit are deleted.

5.3.1 Inputs generation

Boolean quantities

Even in the air system case many of the required inputs come in the form of boolean signals. A few examples are reported in the following table:

Signal description
Cut-off active
Exhaust flap control
Intake manifold pressure sensor fault
Anemometer fault
Exhaust pressure sensor fault
Starting procedures ended

The fuel cut-off occurrences and the sensor faults are not considered as a warm condition stable engine running is considered.

Externally calculated quantities

As a reference, the calculation of the ratio between the manifold pressure and the value upstream the throttle body is reported in *figure 128*:

Figure 128, external calculated quantity

Another example of a required input is the boolean value that is calculated in an external module, which is *true* when the engine speed is greater than a calibration threshold that, in this case, is equal to 5000RPM.

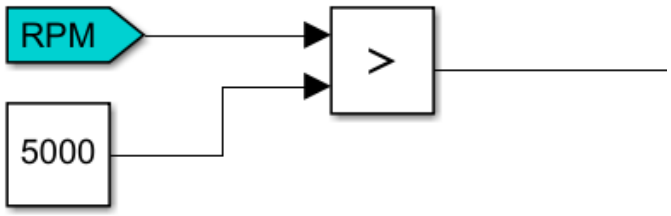


Figure 129, external calculated quantity

Target quantities

Air Module requires the desired throttle mass flow and the desired relative charge, according to the imposed intake manifold pressure target. Previously calculated lookup tables is reused after expanding them in all the engine operating range including the throttled sub-atmospheric manifold pressure region.

5.3.2 Simulation run

After the input selection, the simulation can run fine since the calibration dataset has already proved to be valid during the first benching activities, so the model can be used to perform tuning operations directly.

Intake manifold pressure model

The highest priority task that was done in the MiL environment is verifying the results of the *Manifold pressure model* physical subsystem, which depends on the main outputs of *Charge exchange thought manifold pressure* and of the *Intake Manifold temperature model*. The matching between the modeled intake manifold pressure and the engine measurement gives an estimation of the calibration datasets goodness.

Figure 130 reports the comparison between the measured and the modeled intake manifold pressure.

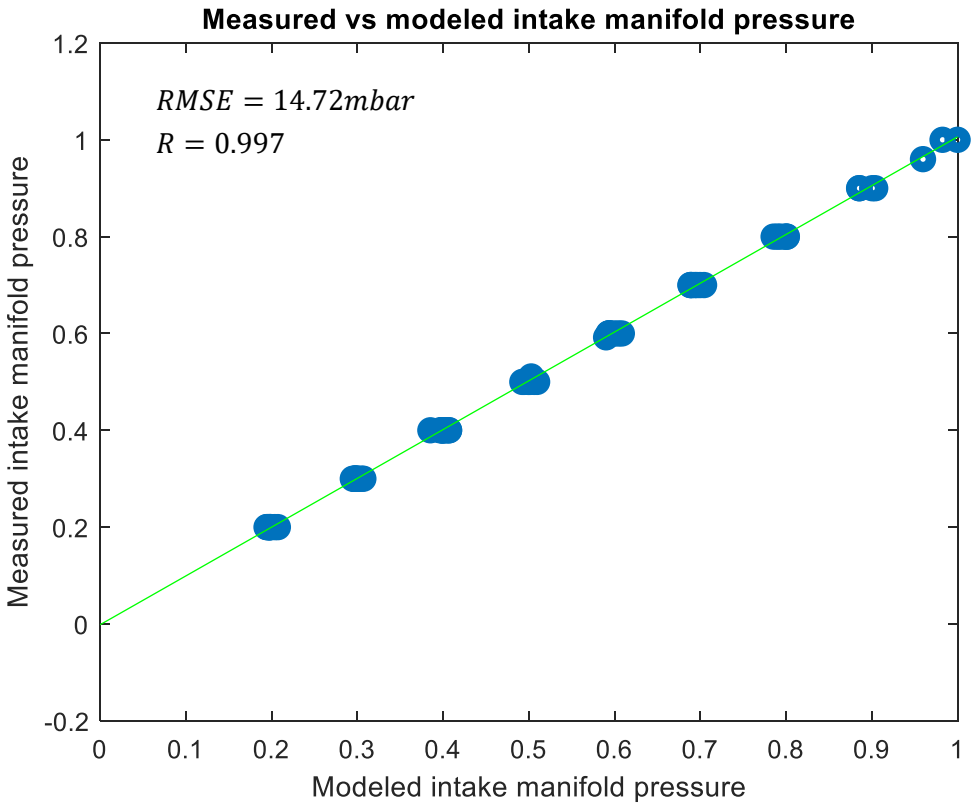


Figure 130, modeled and measured intake manifold pressure comparison

The comparison shows a good correlation coefficient (0.997) between the measured and the modeled intake manifold pressure, meaning that the subsystems which the pressure modeling depends on are well-tuned.

The root mean square error is a frequently used quantity representative of the difference between predicted and observed scatter data, measuring the accuracy of the estimation. The value is calculated according to *formula 5.3.2.1*

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_{estimated} - y_{measured})^2}{N}} \qquad 5.3.2.1$$

In this case, the RMSE is small enough for properly test cell runs.

The error can be evaluated from different perspectives, as dependent of the engine speed (*figure 131*) and of the load (*figure 132*).

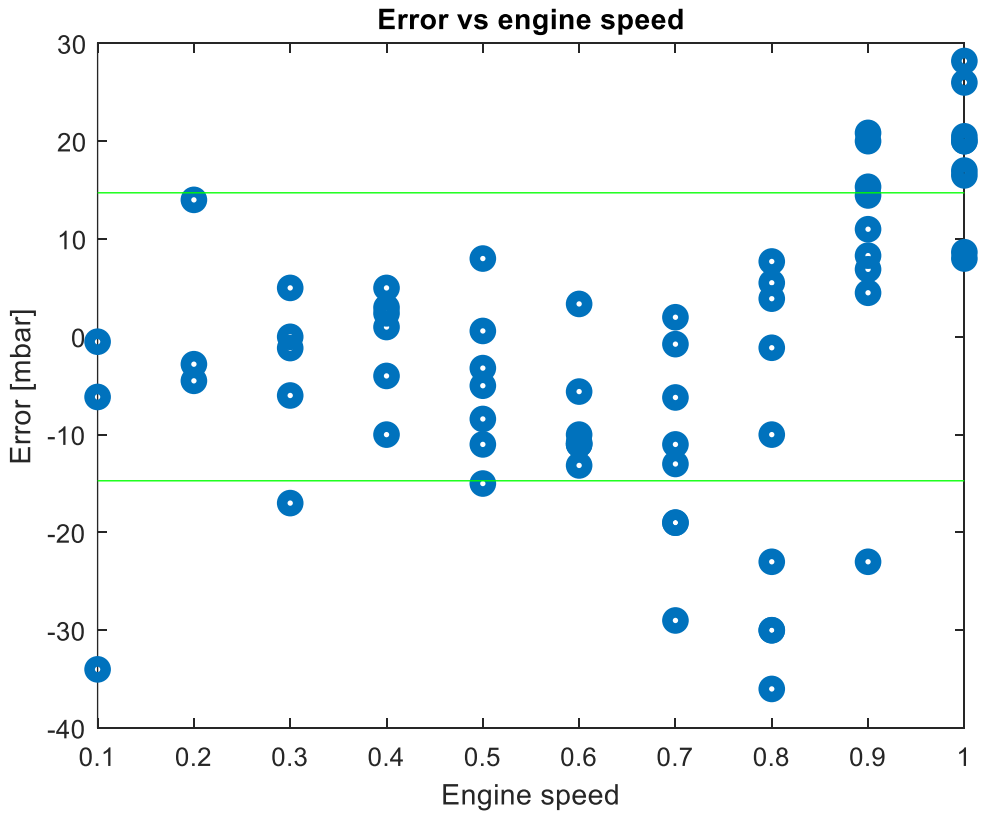


Figure 131, error as function of engine speed

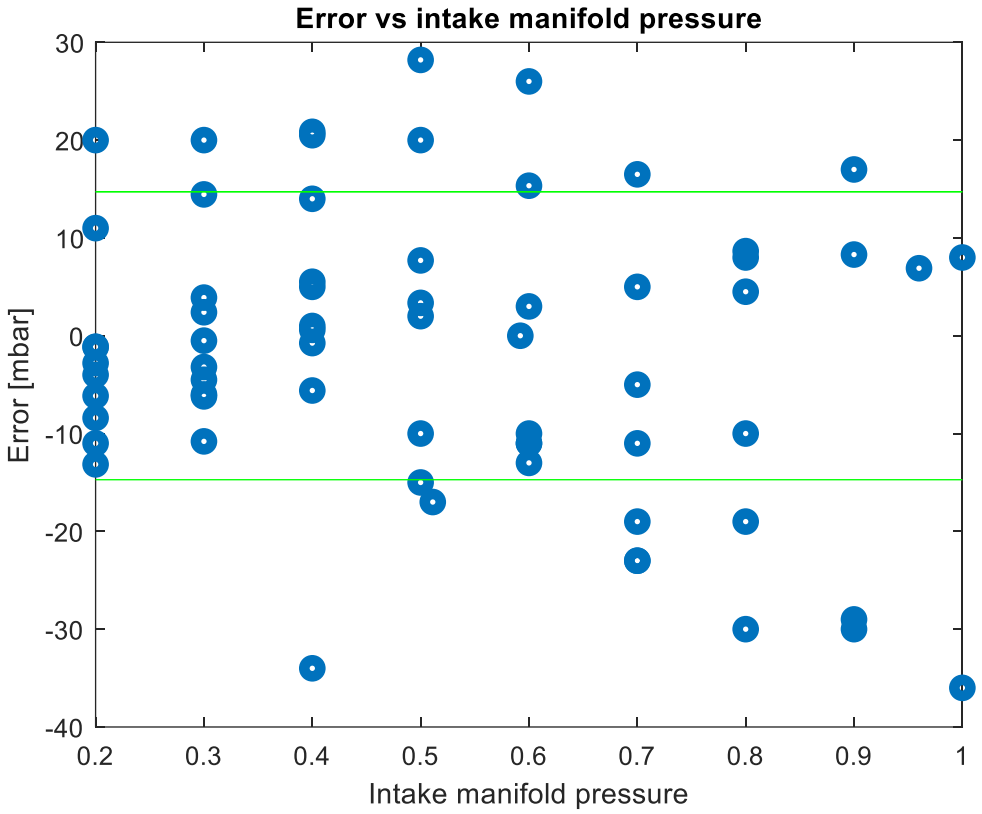


Figure 132, error as function of load

The scatter shows that the engine control unit software overestimates the pressure on light loads while underestimates it when heavy charge is requested.

In the case of speed dependent error, the scatter is denser at medium speeds while degrading when higher performances are reached.

In both cases, the error trend is influenced by the calibration dataset based on an older company engine, which provides lower performance targets.

Once the calibration dataset was tuned according to the model-in-the-loop simulation outcomes, a bench run data was recorded. In particular, the RMSE value obtained from the bench activities is close to the predicted value as the *figure 133* shows.

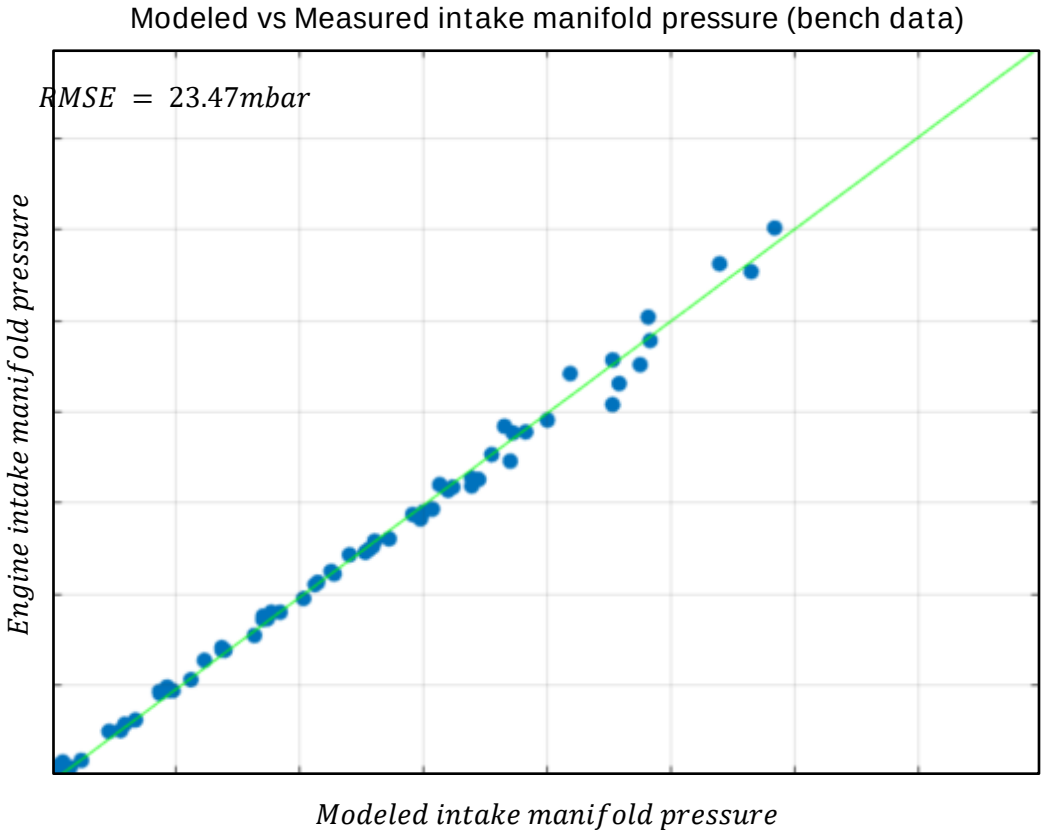


Figure 133, modeled versus measured intake manifold pressure (bench data)

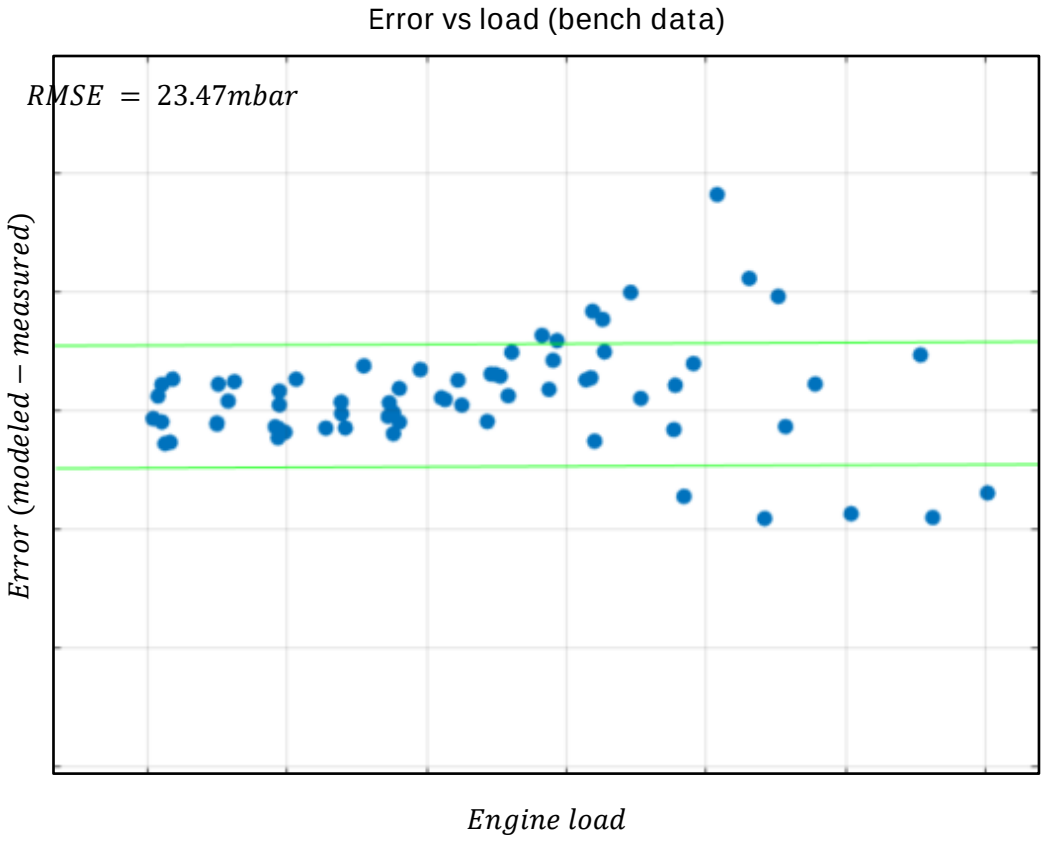


Figure 134, error versus engine load (bench data)

The load-based error distribution appears coherent with the combined simulation outcome. The scatter starts losing accuracy when high loads are requested.

Exhaust manifold pressure

The second main task of the charge combined simulation activity is the calibration of the *Pressure downstream valve model* and the matching between the computed exhaust manifold pressure provided with charge control and the engine model value.

The module relies completely on external system outputs. Therefore, individually tune the calibration dataset of the single module is possible with the maximum accuracy since the GT-Power signals are trusted.

Figure 135 shows qualitatively the base calibration performances on the already recorded experiment.

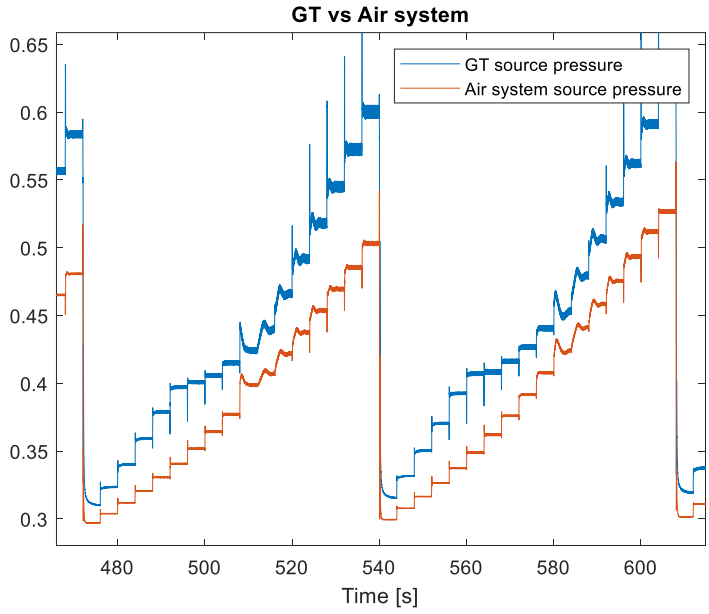


Figure 135, computed and measured exhaust pressure

Starting from the topology of the model, switching to the map-based logic is preferred in this phase. For the purpose, the first operation involves the module *codeword* as *figure 136* shows.

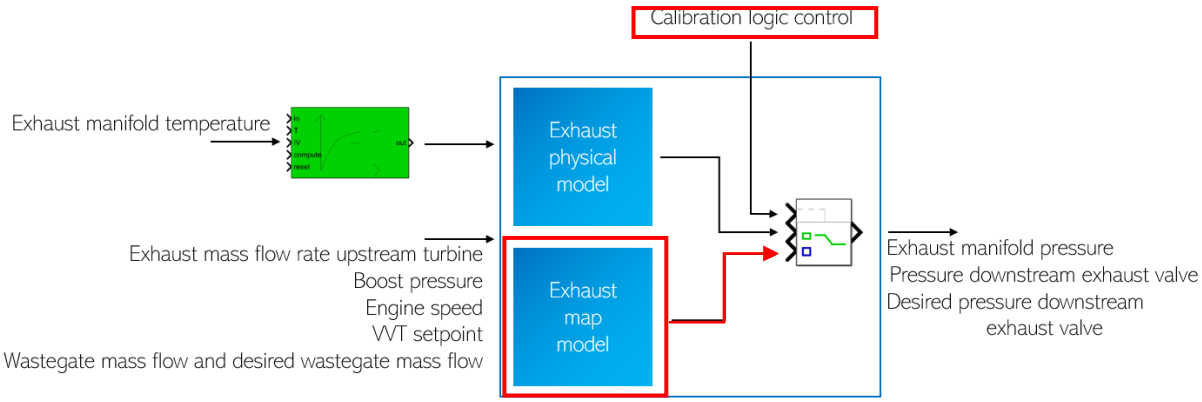


Figure 136, exhaust pressure model tuning

The next step calibrates a 2D map, which depends on the boost, the ambient pressure, and the turbine exhaust mass flow upstream.

During the simulation run the following signals are recorded and post-processed:

- Exhaust manifold pressure
- Boost pressure
- Exhaust mass flow
- Required signals to revert the exhaust pressure into the output of the map

To fit the data, the Model-Based Calibration MATLAB toolbox was used.

The fitting procedure consists of the data import, its fitting, and the extrapolation to the desired map boundaries, as *figure 137* highlights.

The importing activity chooses the variables to import from MATLAB workspace. In this case, a structure is provided by the simulation post-processing with three fields, the two map inputs and the value that the lookup table should respond.

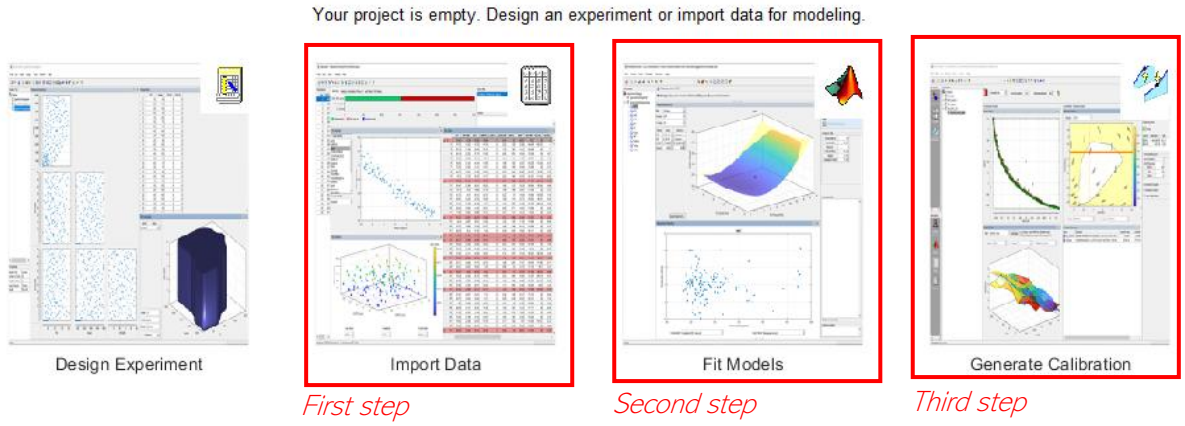


Figure 137, Model based calibration toolbox fitting steps

Three options are available for fitting the data:

- One-stage model fits a model to all the data in one process
- A two-stage model fits a model to data with a hierarchical structure. If the data contains variables that are fixed while varying others, this fitting procedure must be chosen
- The point-by-point model fits a model at each operating point. No predictions are available between operating points.

One stage model option is suitable for the tasks because it fits all the triplets (two inputs and one output) together in a single surface. The outcome of the fitting procedure is the interpolated plot and the validation report.

A random selection from the provided data tests the goodness of the operations. *Figure 138* shows the validation plot. On the horizontal axis there is the output of the fitted surface while on the vertical axis there are the residual values (the difference between the observed and the predicted data).

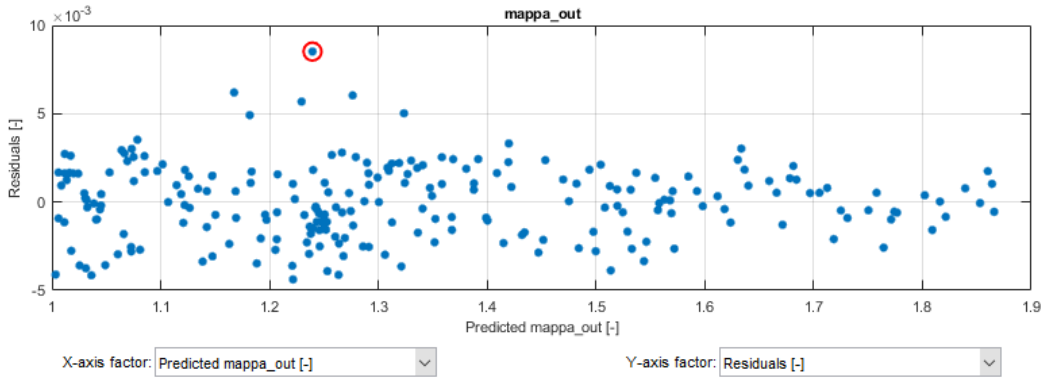


Figure 138, residuals plot

The interpolation of the random sorted values is completely satisfactory as long as the residuals stay below $10^{-2} mbar$ order of magnitude. The fitting process returns the *figure 139*, represented as contour scope.

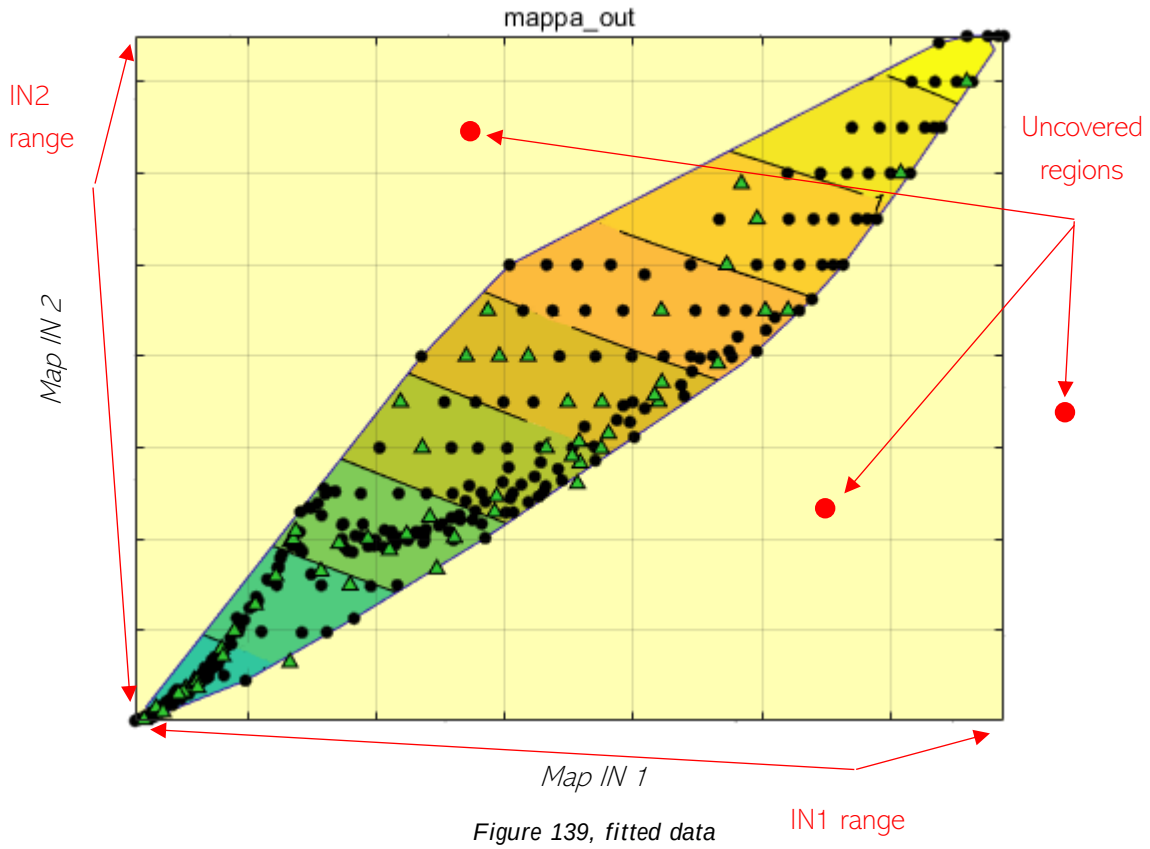


Figure 139, fitted data

The third step consists of the map filling further the experiment boundaries, according to the complete engine operating range.

The last toolbox option was used specifying the boundary of the required dataset via *IN1* and *IN2* vectors.

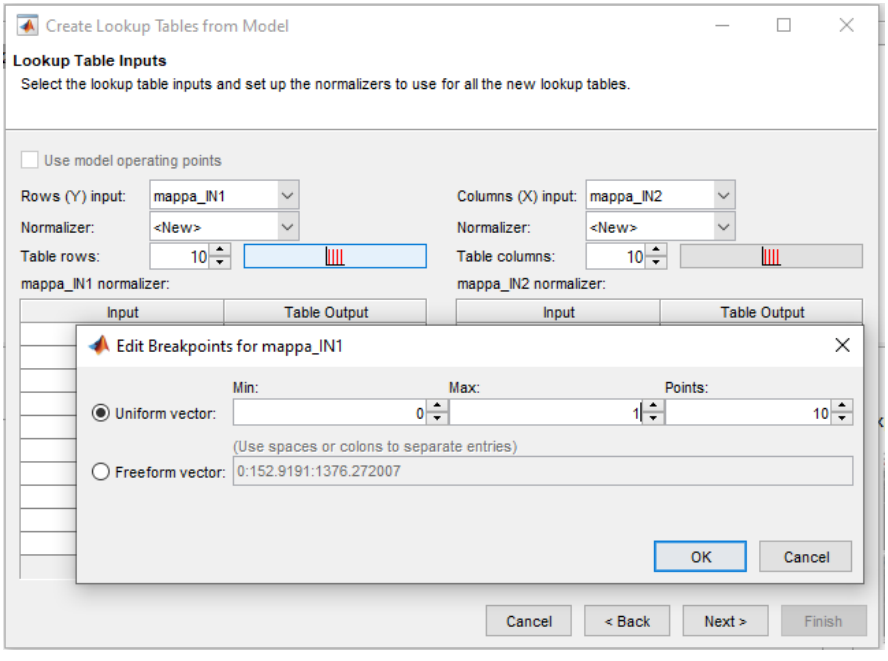


Figure 140, boundaries expansion

Figure 141 represents the obtained surface in which the experimental points are highlighted with a sphere. The other data is extrapolated.

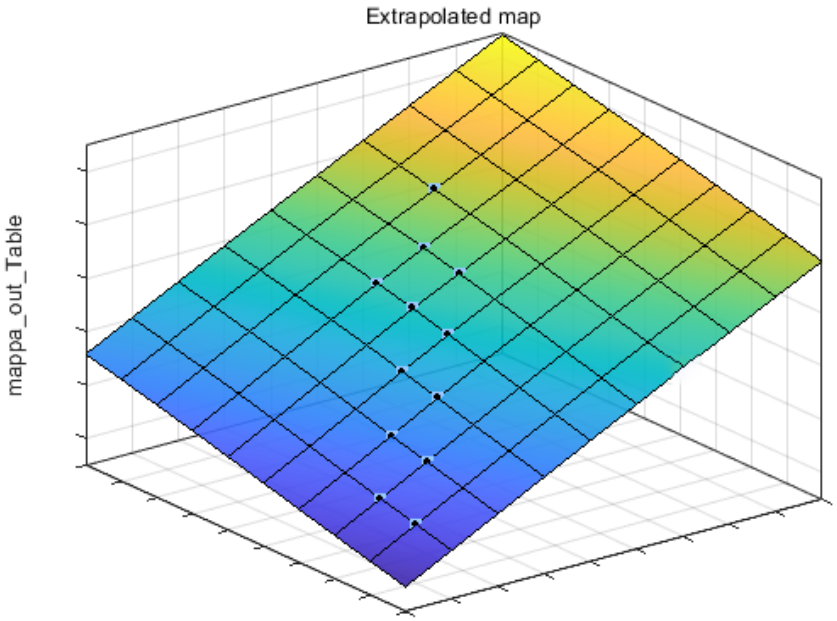


Figure 141, extrapolated data

After loading the obtained dataset in the workspace calibration structure, the simulation can run with substantial improvements of the exhaust manifold pressure calculation, as *figure 142* and *figure 143* show.

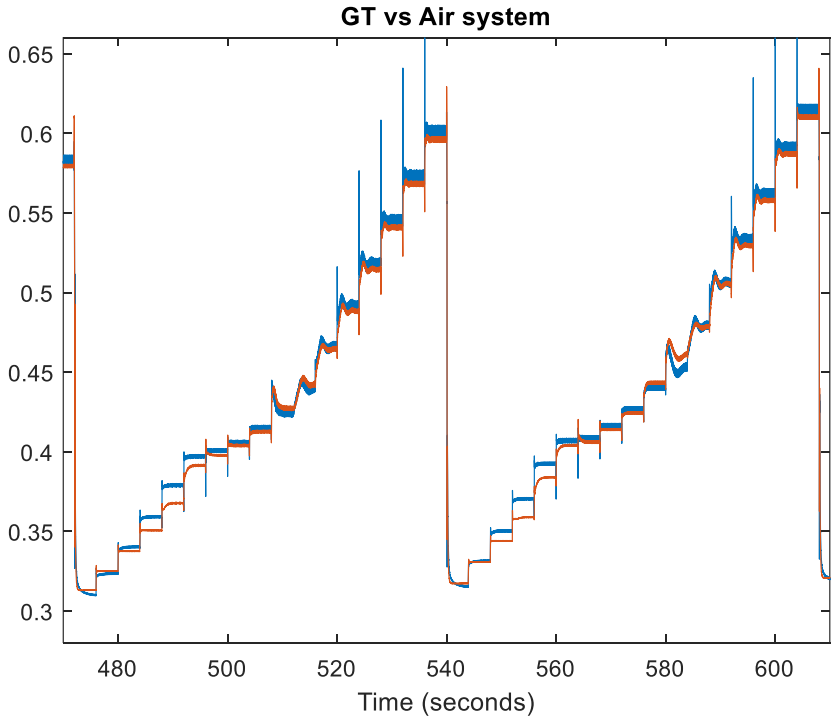


Figure 142, tuned model and measured exhaust pressure comparison

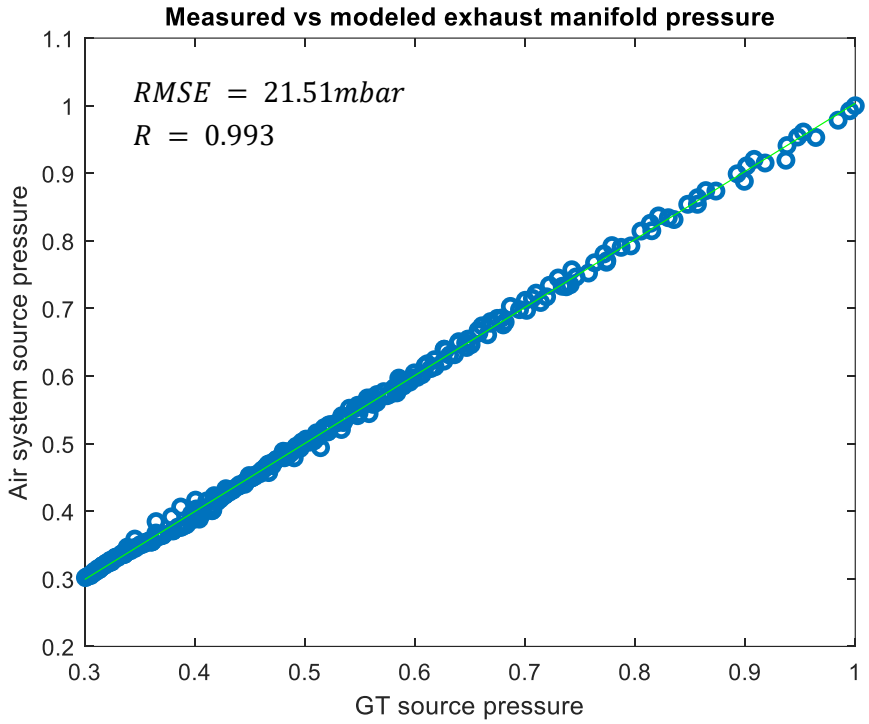


Figure 143, tuned model and measured exhaust pressure correlation

The determination of the exhaust manifold pressure improves significantly, even if the RMSE tolerance can be further lowered through advanced extrapolation settings and smarter grid selection like thickening the boundaries in the most frequently used operating points.

6

Final considerations

In the first phase of the activity, it was described the individual modules functions, their Simulink implementation, and their validation. In the second part of the dissertation the combined simulation environment was detailed with its settings and its structures. Finally, two examples about the possible usage of the model were given.

The potential of the model-in-the-loop testing approach was also confirmed during the test bench runs where the predictions about the behavior of the tuned calibration confirmed.

Yet, the possible applications of the model-in-the-loop testing are limitless. For instance, the systems that could be further integrated might be:

- Fuel injection system
- Variable valve timing system
- Spark advance system with closed loop knocking control
- Exhaust gas recirculation system

Over the engine control purpose, huge benefits can be obtained with hybrid powertrains simulation. The engine electrification, hence, adds infinite degrees of freedom to the power

output strategy, which must be coordinated and optimized according to the vehicle targets and performances.

Moreover, an entire driving cycle can be emulated with a little more effort, enabling optimization, for example, of driving mode maps, gearbox shifting, torque splitting between engine and electric drive and battery state-of-charge management. Therefore, GT-Power can also provide estimations about emissions and aftertreatments operations like catalyst heating or particulate filter purging policy, which are a particularly felt themes of the incoming Euro 7 regulations.

References

ECU supplier (2021), *ECU software reference paper*.

Automobili Lamborghini (2021), *Boost module ECU reference paper*.

POWERTECH Engineering (2021), *GT-Power training course* documentation.

B. Lumpp, M. Tanimou, M. McMackin, E. Bouillon et al. (2014), *Desktop Simulation and Calibration of Diesel Engine ECU Software using Software-in-the-Loop Methodology*. SAE TECHNICAL PAPER.

V. Jaikamal (2009), *Model-based ECU development – An integrated MiL-SiL-HiL Approach.* SAE INTERNATIONAL, ETAS Inc.

O. Philipp, M. Buhl, S. Diehl, M. Huber, S. Roehlich and J. Thalhauser (2005), *Engine ECU Function Development Using Software-in-the-Loop Methodology*. SAE TECHNICAL PAPER.

ECU supplier (2004), *Air path documentation*.

ECU supplier (2004), *Torque structure documentation*.

E. Rask, M. Sellnau (2004), *Simulation-Based Engine Calibration: Tools, Techniques, and Applications*. SAE TECHNICAL PAPER.

A. Caraceni, F. De Cristofaro, F. Ferrara and S. Scala (2003), *Benefits of Using a Real-Time Engine Model During Engine ECU Development*. SAE TECHNICAL PAPER.