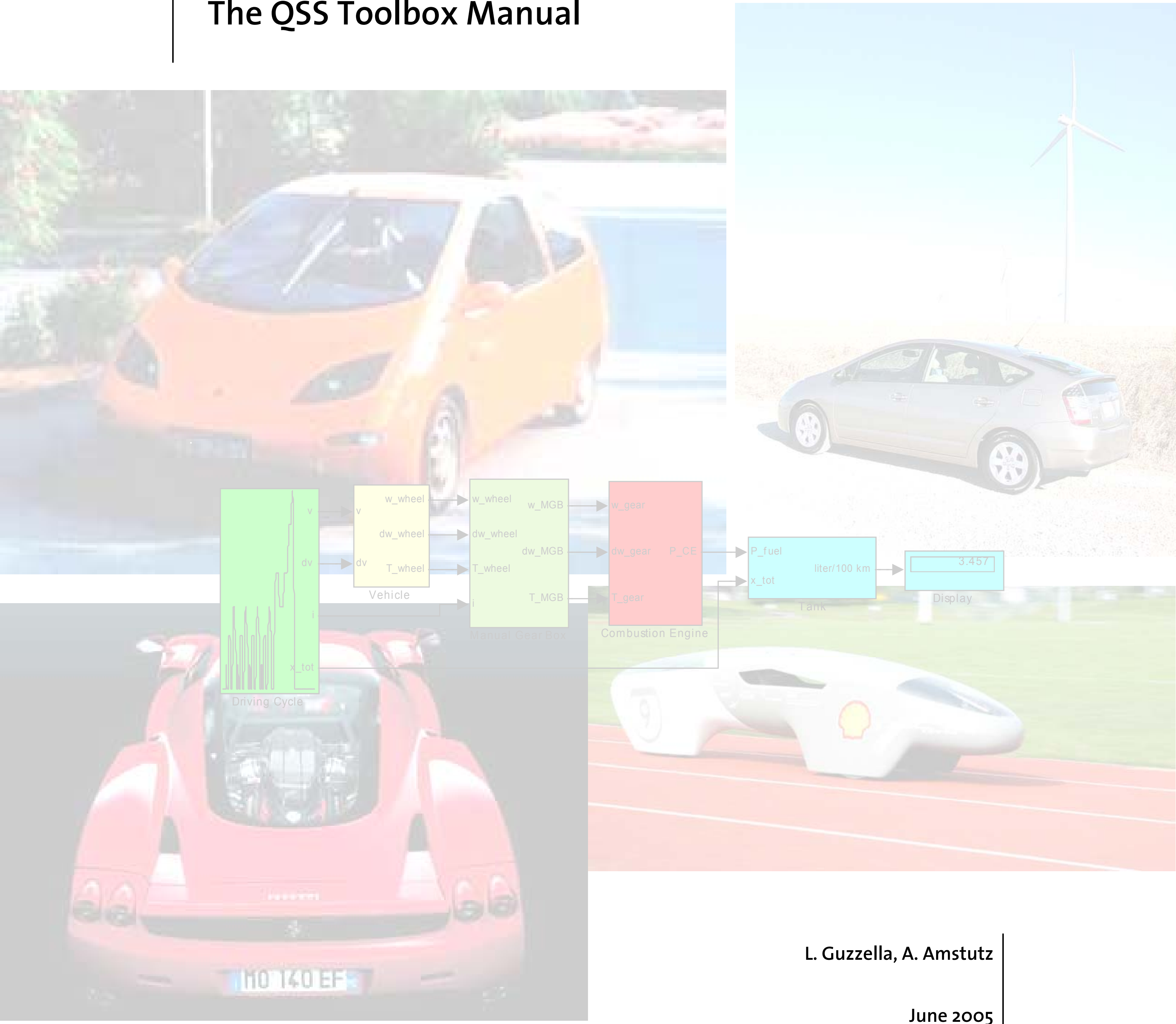
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für Mess- und Regeltechnik
Measurement and Control Laboratory

The QSS Toolbox Manual



L. Guzzella, A. Amstutz

June 2005

Preface

This text describes the design and use of the second version of the QuasiStatic Simulation Toolbox (QSS TB) which permits a fast and simple estimation of the fuel consumption for many powertrain systems. This toolbox is used in ongoing research projects and in the exercises to the lecture "Vehicle Propulsion Systems."

This is the manual's second version. The authors welcome any notifications of errors (in content or terminology), suggestions for improvements or expansion, etc. (send an email to: `guzzella@imrt.mavt.ethz.ch`).

The QSS TB is provided to students and other interested persons under the following conditions only:

1. all rights to the program remain with the authors
2. the authors decline any liability related to the use of the QSS toolbox.

This document is not entirely self-explanatory; at certain points, the consultation of the script to the lecture "Vehicle Propulsion Systems" is necessary for comprehension.

A special thanks to Brigitte Rohrbach for the translation of this manual!

Contents

List of Symbols

Latin

a	Acceleration	[m/s^2]
A	Surface	[m^2]
B	Bore	[m]
d	Diameter	[m]
e	Efficiency	[-]
g	Gravitation constant	[$kg*m/s^2$]
h	Integration step size	[s]
H_u	Lower heating value	[J/kg]
k	Computational step	[-]
m	Mass	[kg]
r	Radius	[m]
v	Speed	[m/s]
V	Fuel consumption	[kg/s]
V_d	Displaced volume	[m^3]
P	Power	[W]
S	Stroke	[m]
T	Torque	[Nm]
x	Distance	[m]

Greek

η	Efficiency	[-]
ρ	Density	[kg/m^3]
Θ	Inertia	[$kg*m^2$]
ω	Rotational speed	[rad/s]

Acronyms

BT	Battery
CE	Combustion Engine
CVT	Continuously Variable Transmission
EG	Electric Generator
EM	Electric Motor
FC	Fuel Cell
MGB	Manual Gear Box
SC	Supercapacitor

Indexes

f	Vehicle
L	Air
φ	Fuel

1 Introduction

1.1 Goals and limitations

Prerequisites
The following two conditions must be met for a user to work efficiently with the QSS toolbox (QSS TB):

- Users must be familiar with Matlab/Simulink™, on which this toolbox is based. A familiarity with file handling, the execution of a simulation, and the presentation of results are minimum prerequisites.
- Users must have a basic understanding of the physics and the design of powertrain systems.

What the QSS TB can do
The QSS TB makes it possible for powertrain systems to be designed quickly and in a flexible manner and to calculate easily the fuel consumption of such systems. The QSS TB contains examples of a number of elements. Chapter 2 contains detailed descriptions of all these elements.
Users with a good grasp of the "philosophy" behind the QSS TB (Sec. 1.2), who know the general structure of a QSS TB program (Sec. 3.1) and who have read the commentary in Section 3.2 below are well prepared, once they have gained a little experience, to readily design and include new elements of their own.
The most efficient use of the QSS TB can be made once users fully understand the techniques required (i.e., the optimization routines) to integrate the toolbox with other programs. This allows a smooth integration with the functionality of Matlab™ and all its other toolboxes.
Due to the extremely short CPU time it requires (i.e., on a regular PC, a speedup factor of 100 to 1000 for a conventional powertrain), a QSS model is ideally suited for the optimization of the fuel consumption under various control strategies.

What the QSS TB cannot do
The quasistatic approach obviously is not suitable for the capture of dynamic phenomena, i.e. those adequately described by differential equations. There are numerical approaches better suited for the efficient solution of those problems. Typical examples of such problems are drivetrain ringing phenomena or the analysis of state events such as stickslip effects.
In principle, the QSS TB could be used for the calculation of vehicle pollutant emissions as well. However, due to the current lack of reliable and easily acquired data describing the physics with sufficient accuracy, this second version of the toolbox does not include these aspects.

1.2 The quasistatic approach

Since the basic principles of the QSS are discussed in detail in the script to the lecture "Vehicle Propulsion Systems," our primary intention here is to show how the elements of the QSS TB can be linked.
The key idea behind the QSS TB is to reverse the usual cause-and-effect relationships of dynamic systems. Rather than calculating speeds from given forces, based upon given speeds (at discrete times), the toolbox calculates accelerations and determines the necessary forces.

Vehicle in the plane – traditional approach:

System: $m \cdot \dot{v}_f(t) = F_a(t) - m_f \cdot g \cdot c_r - \frac{1}{2} \cdot \rho_L \cdot c_w \cdot A_f \cdot v_f^2(t)$

Cause: force $F_a(t)$

Effect: vehicle speed $v_f(t)$

Vehicle in the plane – QSS approach:

System: $m \cdot \dot{v}_f(t) = F_a(t) - m_f \cdot g \cdot c_r - \frac{1}{2} \cdot \rho_L \cdot c_w \cdot A_f \cdot v_f^2(t)$

Cause: $v_f(k \cdot h)$, i.e., speed given at certain times

Effects:
1. Mean speed $v_f(t) = (v_f(k \cdot h + h) + v_f(k \cdot h))/2$, $\forall t \in [k \cdot h, k \cdot h + h)$
2. Acceleration $\dot{v}_f(t) = (v_f(k \cdot h + h) - v_f(k \cdot h))/h$, $\forall t \in [k \cdot h, k \cdot h + h)$
3. Driving force $F_a(t)$ (constant in the interval $\forall t \in [k \cdot h, k \cdot h + h)$!)

The symbol "*h*" representing the step size of the QSS simulation, is protected and cannot be redefined. If the accuracy is to be increased, the step size *h* may be decreased. However, not only will the calculations take longer, but there are more efficient ways to solve these problems numerically.
On the basis of given values for speed and force, fuel consumption is calculated using so-called engine maps. Surprisingly enough, this approach is valid for a number of different structures and elements, such as IC engines, electric motors, intermediate storage devices, drivetrains, etc. It even permits the integration of control systems in that these forces are assigned to the various drive elements in accordance with the internal control logic. A few examples are shown below.

1.3 An introductory example

Each program within the QSS TB consists basically of an .mdl file (Simulink) where the system model is described (which contains the description of the drive assembly).
It is always good programming practice not to enter any numerical values for the parameters directly in the .mdl file (even though that would easily be possible); rather, all parameter values can be easily entered in each block of the model through an intuitive user-interface (so called “mask”).
The units and the remarks of the parameters are normally specified within the mask environment; since the number of calculation steps is relatively small[1], it is generally possible to do without normalizing them but rather to work with SI units directly.

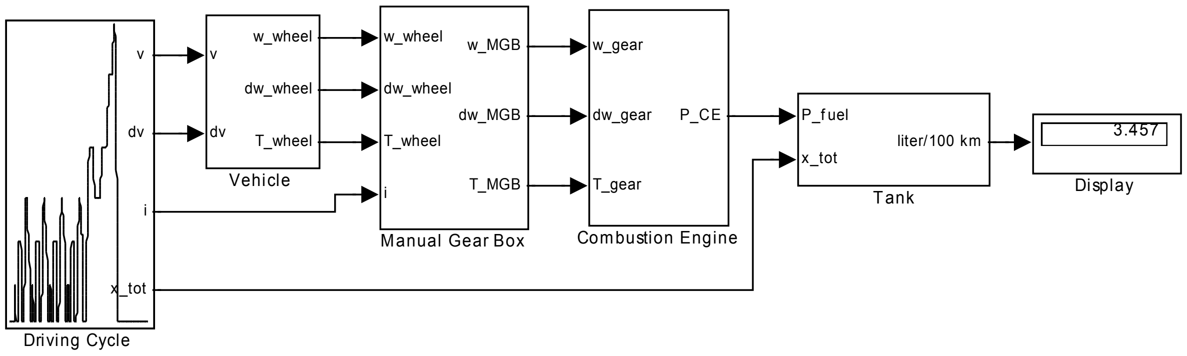
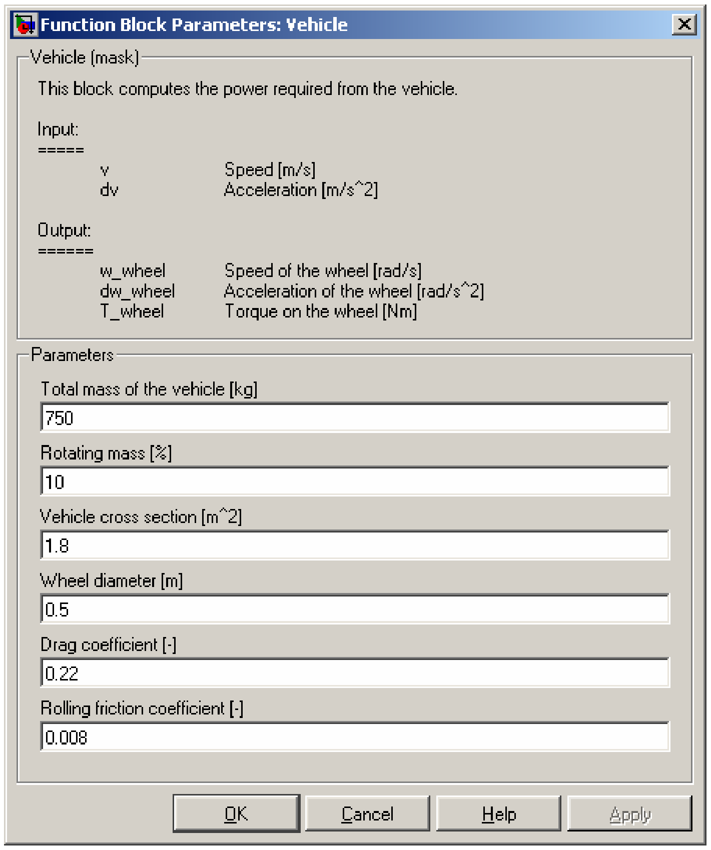


Fig. 1.3.1 The .mdl file of a classical IC engine drive in the QSS TB form

First a drive cycle has to be chosen (EU cycle and FTP are typical candidates; however, any cycles designed by users may of course be used as well). Within the function block "Vehicle" the torque at the wheel (i.e. resistances plus acceleration) is calculated. While this torque is determined on the basis of the gear selected, in the case of automatic transmissions it may also be determined online by a control system. The transmission converts it to a demanded net engine torque. From the total torque, which consists of net torque plus acceleration torque, the IC engine then "produces" an amount of fuel consumption. Within the block called "Tank" those consumption data are summarized and converted to a consumption figure in liters per 100 km.

The detailed explanation of the various blocks in Fig. 1.3.1 will be given in Chapter 2.
To accede the mask in order to change the parameter values, simply double-click on the corresponding Simulink block. An environment similar to the one shown as an example in Fig. 1.3.2 allows the user to enter the desired values.

[1]This is no longer the case if the program contains a large number of implicit equations.



Function Block Parameters: Vehicle

Vehicle (mask)

This block computes the power required from the vehicle.

```
Input:
=====
        v               Speed [m/s]
        dv              Acceleration [m/s^2]

Output:
======
        w_wheel         Speed of the wheel [rad/s]
        dw_wheel        Acceleration of the wheel [rad/s^2]
        T_wheel         Torque on the wheel [Nm]
```

Parameters

Total mass of the vehicle [kg]	750
Rotating mass [%]	10
Vehicle cross section [m^2]	1.8
Wheel diameter [m]	0.5
Drag coefficient [-]	0.22
Rolling friction coefficient [-]	0.008

Fig. 1.3.2 The user interface (mask) of the "Vehicle" block

Referring to the example of Fig. 1.3.1, the following parameter values have to be entered in the mask of the corresponding block:

Driving Cycle:

Parameter	**Value**
Choose a cycle	Europe NEDC
Step size [s]	1
Enable automatic simulation stop	ON

Vehicle:

Parameter	**Value**
Total mass of the vehicle [kg]	750
Rotating mass [%]	5
Vehicle cross section [m^2]	1.8
Wheel diameter [m]	0.5
Drag coefficient [-]	0.22
Rolling friction coefficient [-]	0.008

Manual Gear Box:

Parameter	**Value**
1. gear [-]	15.174
2. gear [-]	8.338
3. gear [-]	5.378
4. gear [-]	3.937
5. gear [-]	2.748
Differential gear [-]	1
Efficiency [-]	0.98
Idling losses (friction) [W]	300
Minimum wheel speed [rad/s]	1

Combustion Engine:

Parameter	**Value**
Engine type	Otto
Displacement [l]	0.708
Engine scaling factor [-]	1
Engine inertia [kg*m^2]	0.05
Engine speed at idle [rad/s]	105
Engine power at idle [W]	2600
Power required by auxiliaries [W]	300
Enable fuel cutoff	ON
Engine torque at fuel cutoff [Nm]	5
Engine power at fuel cutoff [W]	0

Tank:

Parameter	**Value**
Fuel	Gasoline
Include cold start losses	ON

The exact significance of each parameter is discussed in more detail in Chapter 2. Notice, however, that even for this relatively simple example a significant number of parameters must be known.

Figs. 1.3.3 through 1.3.5 show a few typical results of these calculations. It may not be immediately apparent just how easily so-called "what-if" questions for parameter variations can be answered (e.g., concerning sensitivities). Even questions regarding structural variations (e.g., standard shift vs. CVT) can be investigated with just slight changes within the .mdl file.

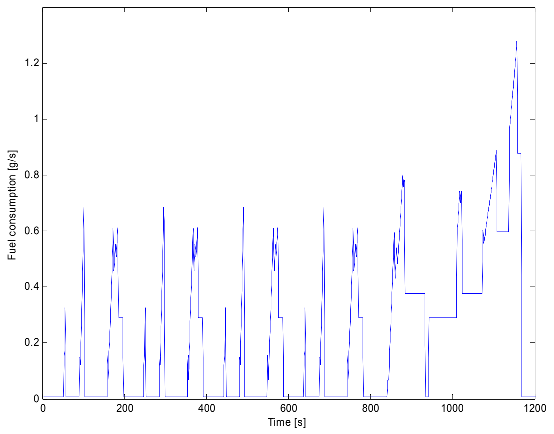


Fig. 1.3.3 Fuel consumption of the IC engine [g/s]

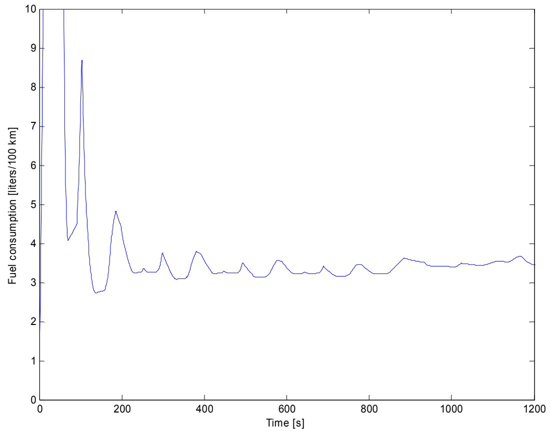


Fig. 1.3.4 Fuel consumption [liters/100 km]

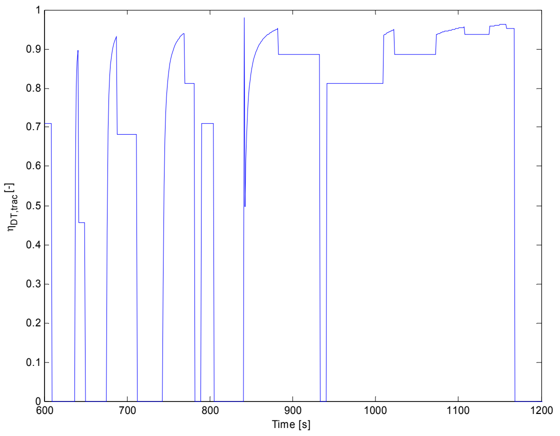


Fig. 1.3.5 Efficiency of the drivetrain during the last city and extra-urban portions (only traction phase considered)

2. The Elements of the QSS Toolbox Library

The various elements of the QSS TB library are grouped according to their function as depicted in Fig. 2.0.1. To open a group of elements, double-click on the corresponding block.

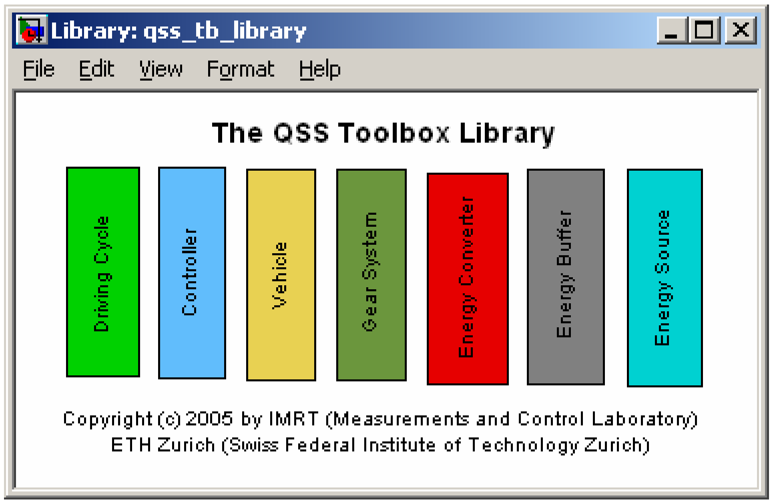

Library: qss_tb_library
File Edit View Format Help
The QSS Toolbox Library
Driving Cycle
Controller
Vehicle
Gear System
Energy Converter
Energy Buffer
Energy Source
Copyright (c) 2005 by IMRT (Measurements and Control Laboratory)
ETH Zurich (Swiss Federal Institute of Technology Zurich)

Fig. 2.0.1 The top level of the QSS TB Library

In the following sections each group of elements of the QSS TB library will be analyzed in detail.

2.1 Driving Cycle

A comparison of the fuel consumption data of various drive systems only makes sense if those data are acquired during a normalized test drive, a so-called test cycle. Typically, Europeans use the EU cycle, whereas the U.S. use the FTP cycle. It is generally known that these cycles tend to be on the "soft" side, i.e., the data for acceleration and top speeds frequently are exceeded in daily driving. The use of proprietary test cycles may thus make more sense.

A cycle is defined by at least two vectors:

1. a time vector (in which the time spans usually are equal), and
2. a vehicle speed vector $v_f(k \cdot h)$.

The acceleration is calculated from the vehicle speed on the basis of the following equation:

$$a_f(k \cdot h) = \frac{v_f(k \cdot h + h) - v_f(k \cdot h)}{h}, \quad k = 1, \ldots k_{\max} - 1, \qquad a_f(k_{\max}) = 0, \tag{2.1.1}$$

The acceleration here is treated "predictively" whereas it may also be treated "retrospectively," as follows:

$$\tilde{a}_f(1) = 0, \qquad \tilde{a}_f(k \cdot h) = \frac{v_f(k \cdot h) - v_f(k \cdot h - h)}{h}, \quad k = 2, \ldots k_{\max} \tag{2.1.2}$$

which leads to less realistic values.

The EU cycle defines the gear shift points for manual shift gear boxes, while the FTP cycle does not have fixed gear shift points. This amounts to a total of three vectors which in the QSS TB must be defined in three .mat files, e.g.,

1. T_Z.mat = time vector[2]
2. V_Z.mat = speed vector
3. G_ Z.mat = gear vector.

In the QSS TB package the most frequently used driving cycles are already defined, i.e. the corresponding time, speed, and gear vectors of cycles such as EU or FTP are already saved in the folder "*\Data\DrivingCycles*". Therefore the user may choose to work with the desired driving cycle.

Of course there is the possibility to define own cycles; this would require the definition of two or three vectors of equal length. The gear values would not need to be predefined. A CVT may be represented by taking a conventional transmission (cf. Sec. 2.4 below) whose gear ratio is given by a control unit (Sec. 2.7 below).

Please refer to the Chapter 4 "Customizing the QSS Toolbox Library" for further information.

[2] This text uses explicit names for the parameters, if at all possible those used in the .m file of Sec. 1.2 above. Obviously, these names may be chosen at will or redefined, as long as they are used consistently. The only exception is the step size *h* which may not be changed.

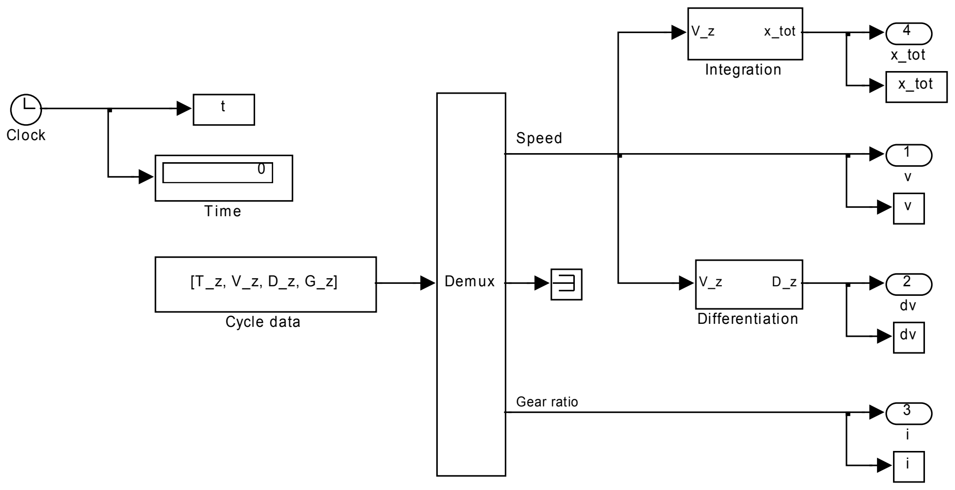
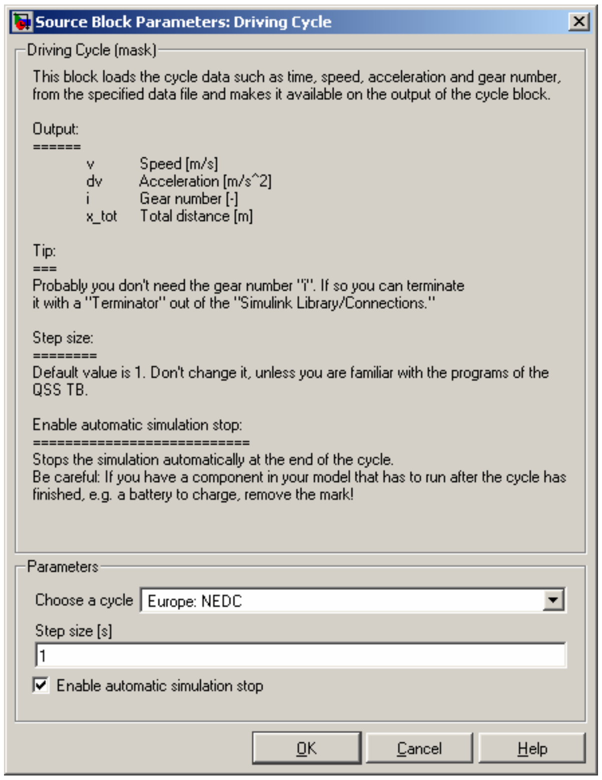


Fig. 2.1.1 **Schematic representation of the block "Driving Cycle" with given gear values**



This block loads the cycle data such as time, speed, acceleration and gear number, from the specified data file and makes it available on the output of the cycle block.

Output:
======

v	Speed [m/s]
dv	Acceleration [m/s^2]
i	Gear number [-]
x_tot	Total distance [m]

Tip:
===
Probably you don't need the gear number "i". If so you can terminate it with a "Terminator" out of the "Simulink Library/Connections."

Step size:
========
Default value is 1. Don't change it, unless you are familiar with the programs of the QSS TB.

Enable automatic simulation stop:
===========================
Stops the simulation automatically at the end of the cycle.
Be careful: If you have a component in your model that has to run after the cycle has finished, e.g. a battery to charge, remove the mark!

Parameters

Choose a cycle: Europe: NEDC

Step size [s]: 1

[✓] Enable automatic simulation stop

Fig. 2.1.2 **The user interface (mask) of the "Driving Cycle" block**

2.2 Vehicle

The total running resistance may be divided in four groups:

- Aerodynamic losses
- Rolling resistance losses
- Vehicle acceleration resistance losses (inertia)
- External resistance losses such as those arising from climbing.

In the QSS TB, the block "Vehicle" shows the first three of these effects.

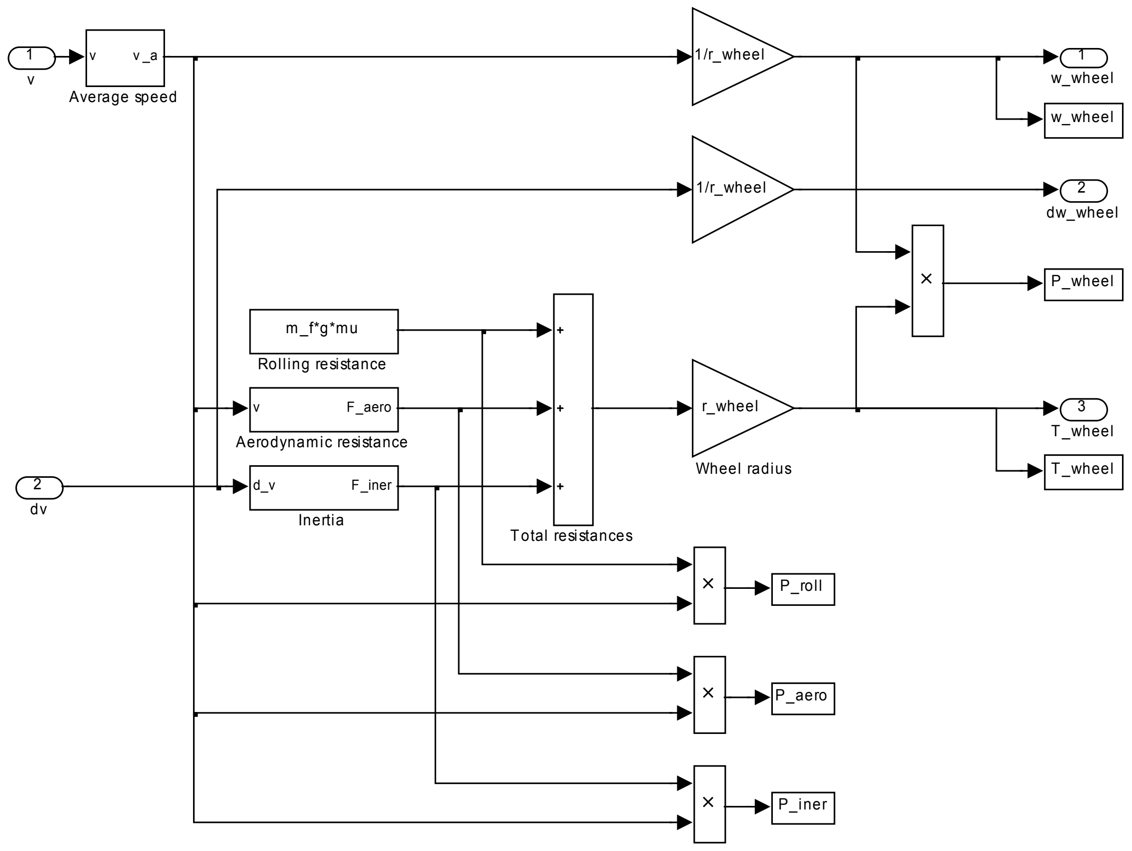
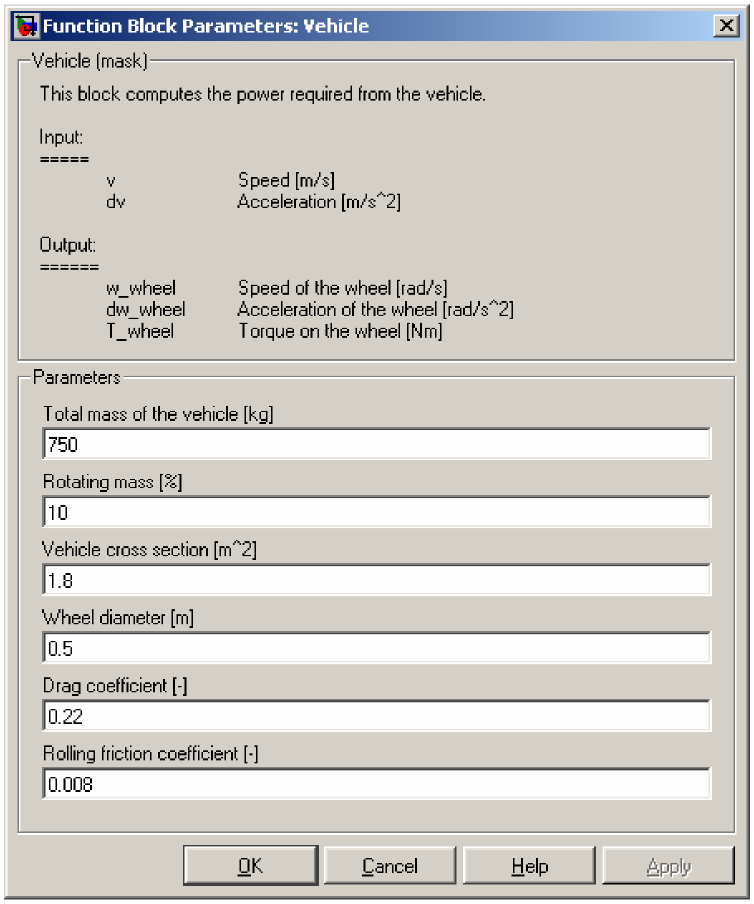


Fig. 2.2.1 Block "Vehicle" – first level

The vehicle inertia reflects the mass of the vehicle (without wheels) plus the wheel inertia data (parameter "Rotating mass [%]" in the mask of Fig. 2.2.2 below), as follows:

$$m_f + 4 \cdot \Theta_{wheel} / r_{wheel}^2 \tag{2.2.1}$$

Notice that any additional inertia, such as engine, flywheel, etc., are not included in this block. They will be represented in their own respective blocks later on.


Function Block Parameters: Vehicle

Vehicle (mask)

This block computes the power required from the vehicle.

```
Input:
=====
        v           Speed [m/s]
        dv          Acceleration [m/s^2]

Output:
======
        w_wheel     Speed of the wheel [rad/s]
        dw_wheel    Acceleration of the wheel [rad/s^2]
        T_wheel     Torque on the wheel [Nm]
```

Parameters

Total mass of the vehicle [kg]	750
Rotating mass [%]	10
Vehicle cross section [m^2]	1.8
Wheel diameter [m]	0.5
Drag coefficient [-]	0.22
Rolling friction coefficient [-]	0.008

Fig. 2.2.2 The user interface (mask) of the "Vehicle" block

2.3 Energy Converter

There are three types of energy converters included in the QSS TB library:

1. Combustion Engine
2. Electric Motor and Generator
3. Fuel Cell

Notice that only the combustion engine and the fuel cell can be directly connected to an energy source (fuel tank), i.e. they can directly "produce" a fuel consumption.

2.3.1 Combustion Engine

In the sense of this toolbox the power produced from an internal combustion engine can be modeled following two different approaches:

- Computing the fuel consumption from a consumption map
- Approximating the efficiency by means of the Willans approximation

2.3.1.1 Based on consumption map

According to this approach the internal combustion engine essentially produces a fuel consumption from a torque and a rotational speed demand (cf. program map block "Engine consumption map" in Fig. 2.3.1). The engine torque consists of the net torque plus the engine acceleration torque, whereby a few special engine operating cases must be accounted for:

- idle
- fuel cutoff
- overload and overspeed.

These special cases are described below. Fig. 2.3.1 shows an overview of the QSS TB model of the IC engine.

Accounting for engine inertia

The engine inertia is accounted for by adding an auxiliary torque that is proportional to the engine inertia and the engine acceleration (see Fig. 2.3.1). Certain delay situations, i.e. "flywheel effects", may even lead to negative torques.

Fuel cutoff

As soon as the total load falls below a certain limit (parameter "*T_CE_cutoff*"), the engine power no longer is determined by the engine map, but rather by a fixed parameter ("*P_CE_cutoff*"), which usually is zero. This so-called engine cutoff function is standard in modern fuel injection systems. The user may choose to enable or disenable this function.

Overload and overspeed detection

Obviously, there are limits to the torque/rotational speed pairs the engine can deliver. The block shown in Fig. 2.3.1, which is shown in detail in Fig. 2.3.2, considers these two limiting conditions. As long as the values for rotational speed and torque remain below their maximum values (torque depending on rotational speed), a positive differential value is passed on to the saturation blocks. They are defined such that only negative values may pass, while positive values are limited to zero. As soon as the limits are exceeded, i.e., the output values of the saturation blocks are nonzero the "Stop" blocks stop the simulation.

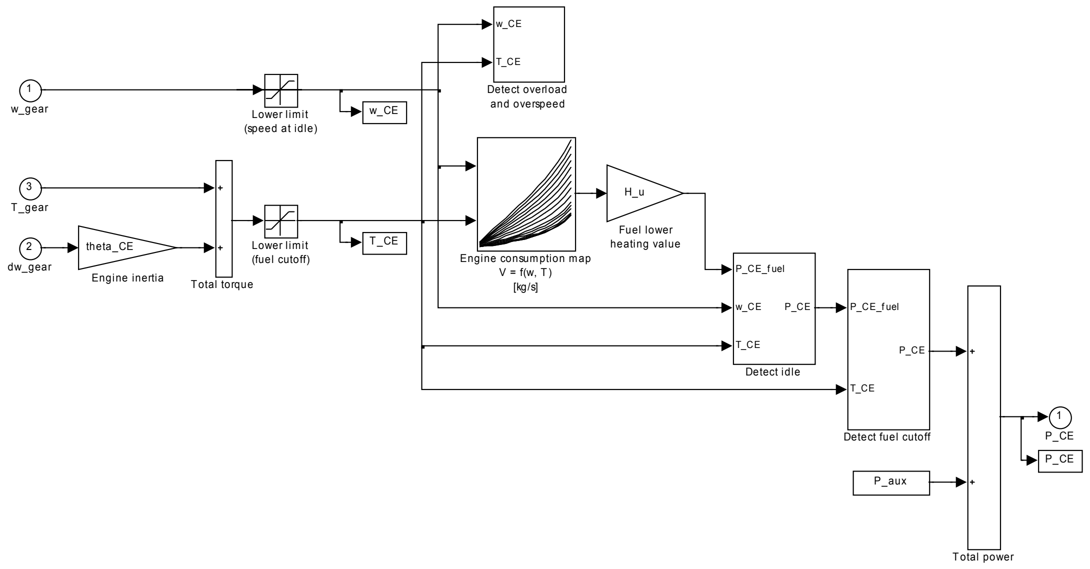


Fig. 2.3.1 Top view of the model "Combustion Engine" based on a consumption map

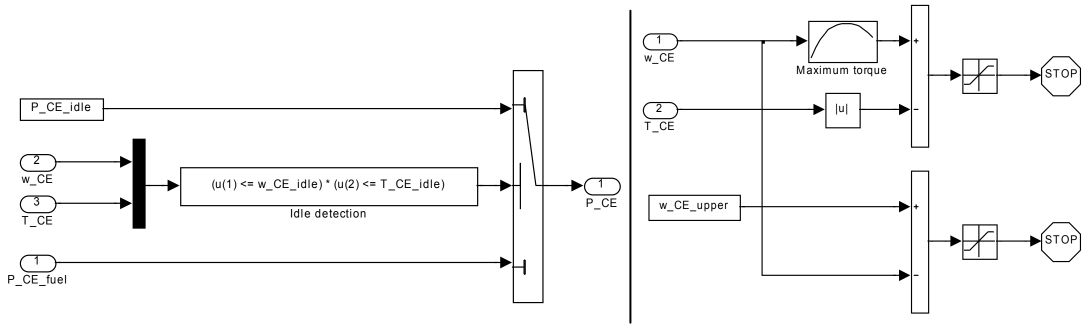


Fig. 2.3.2 Left: idle detection; right: overload and overspeed detection

Consumption at idle

Idle speed is detected as soon as the demanded rotational speed of the engine falls below the limit "*w_CE_idle*" and simultaneously the demanded torque value drops below the limit "*T_CE_idle*" (typically zero, cf. Fig. 2.3.2, left). In this case, the engine power value is set to "*P_CE_idle*". Fig. 2.3.1 shows that the overrun fuel cutoff is given a higher priority, which means that as soon as the torque value at idle speed falls below the fuel cutoff limit, the fuel cutoff is activated.

Consumption maps

For the reasons addressed in depth in the lecture "Vehicle Propulsion Systems", the QSS TB uses two-dimensional fuel consumption maps*:*

$$V_{CE} = f(\omega_{CE}, T_{CE}) \tag{2.3.1}$$

The block "Engine consumption map" needs three parameters:

- a vector 1 x n, termed "*w_CE_row"* containing the rotational speed support points[3]
- a vector m x 1, termed "*T_CE_col* "containing the torque support points
- a map n x m termed "*V_CE_map"* representing the fuel use data (in kg/s) of the combination of torque and rotational speed under consideration.

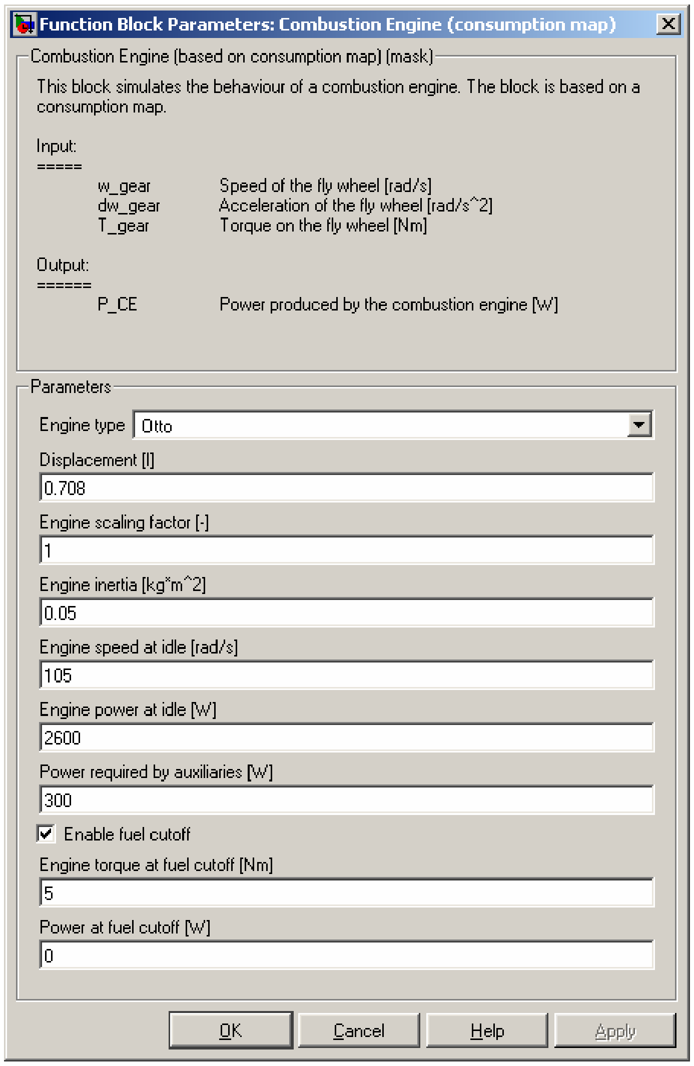
The data structure of the fuel consumption map is shown in Table 2.3.1. Due to the fact that in the two-dimensional map block Simulink demands independent variables to be strictly monotonously increasing vectors, this table must be read from top left to bottom right.

Table 2.3.1 Design of the fuel consumption map of the IC engine, $V_{CE}(\omega_{CE},T_{CE})$ [kg/s]

$V_{CE}(\omega_{\min},T_{\min})$	……	$V_{CE}(\omega_{\min},T_{\max})$
……		……
……		……
……		……
$V_{CE}(\omega_{\max},T_{\min})$	……	$V_{CE}(\omega_{\max},T_{\max})$

Power required by auxiliaries
Auxiliaries such as air conditioning and light cause increased fuel consumption due to their power demands. This fact is taken into account by the parameter "*P_aux"* in Fig. 2.3.1 above.

[3]These values do not have to be equidistant, but they must increase strictly monotonously.



Function Block Parameters: Combustion Engine (consumption map)

Combustion Engine (based on consumption map) (mask)

This block simulates the behaviour of a combustion engine. The block is based on a consumption map.

```
Input:
=====
        w_gear          Speed of the fly wheel [rad/s]
        dw_gear         Acceleration of the fly wheel [rad/s^2]
        T_gear          Torque on the fly wheel [Nm]

Output:
======
        P_CE            Power produced by the combustion engine [W]
```

Parameters

Engine type	Otto
Displacement [l]	0.708
Engine scaling factor [-]	1
Engine inertia [kg*m^2]	0.05
Engine speed at idle [rad/s]	105
Engine power at idle [W]	2600
Power required by auxiliaries [W]	300
Enable fuel cutoff	✓
Engine torque at fuel cutoff [Nm]	5
Power at fuel cutoff [W]	0

Fig. 2.3.3 The user interface (mask) of the "Combustion Engine" block based on consumption map

2.3.1.2 Based on the Willans approximation

A useful simplification of the engine's torque and efficiency characteristics that approximates real engine behavior astonishingly well can often be made (please refer to [1] for further details):

$$p_{me} = e \cdot p_{m\varphi} - p_{me0} \qquad (2.3.2)$$

where p_{me} is the brake mean effective pressure, i.e. a normalized formulation of the engine's torque

$$p_{me} = \frac{T_{CE} \cdot 4\pi}{V_d} \qquad (2.3.3)$$

and $p_{m\varphi}$ is defined as the fuel mean effective pressure

$$p_{m\varphi} = \frac{m_\varphi \cdot H_u}{V_d} \qquad (2.3.4)$$

The brake mean effective pressure is the pressure that has to act on the piston during one full expansion stroke to produce the same amount of work as the real engine does in two engine revolution (a four-stroke engine is assumed here); the fuel mean effective pressure is that brake mean effective pressure that an engine with an efficiency of 1 would produce with the fuel mass m_φ burnt per engine cycle.

Therefore the engine's efficiency can be written as

$$\eta_{CE} = \frac{p_{me}}{p_{m\varphi}} \tag{2.3.5}$$

In Eq. (2.3.2) the coefficient e represents the thermodynamic properties of the engine (related to the indicated mean effective pressure) and p_{me0} represents the engine's losses (due to friction and gas exchange)

$$p_{me0} = p_{me0g} + p_{me0f} \tag{2.3.6}$$

The friction mean effective pressure is calculated using the ETH friction model described in [1]

$$p_{me0f} = k_1 \cdot \left(k_2 + k_3 \cdot S^2 \cdot \omega_{CE}^2\right) \cdot \Pi_{\max} \cdot \sqrt{\frac{k_4}{B}} \tag{2.3.7}$$

whose parameters are listed in Table 2.3.2 below.

Table 2.3.2 Parameters of the ETH friction model according to [1]

Parameter	**Otto**	**Diesel**
k_1	$1.44 * 10^5$ [Pa]	$1.44 * 10^5$ [Pa]
k_2	0.46 [-]	0.50 [-]
k_3	$9.1 * 10^{-4}$ [s^2/m^2]	$1.1 * 10^{-3}$ [s^2/m^2]
k_4	0.075 [m]	0.075 [m]

Finally, the power produced from the combustion engine is

$$P_{CE} = \eta_{CE} \cdot P_{gear} \tag{2.3.8}$$

where the power delivered from the gear box is defined as:

$$P_{gear} = T_{gear} \cdot \omega_{gear} \tag{2.3.9}$$

Fig. 2.3.4 shows an overview of the QSS TB model of the IC engine based on the Willans approximation. Notice the similarity to Fig. 2.3.1.

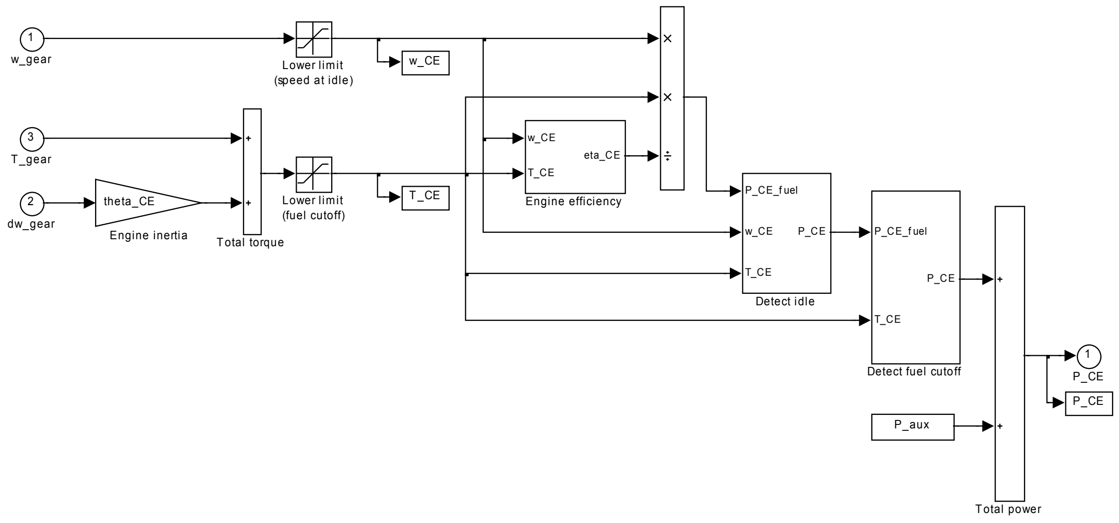


Fig. 2.3.4 Top view of the model "Combustion Engine" based on the Willans approximation

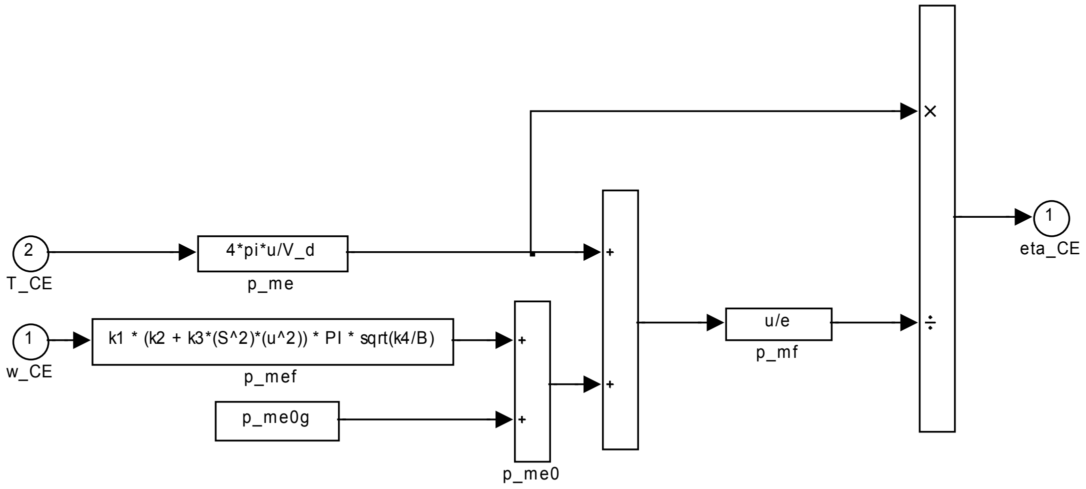
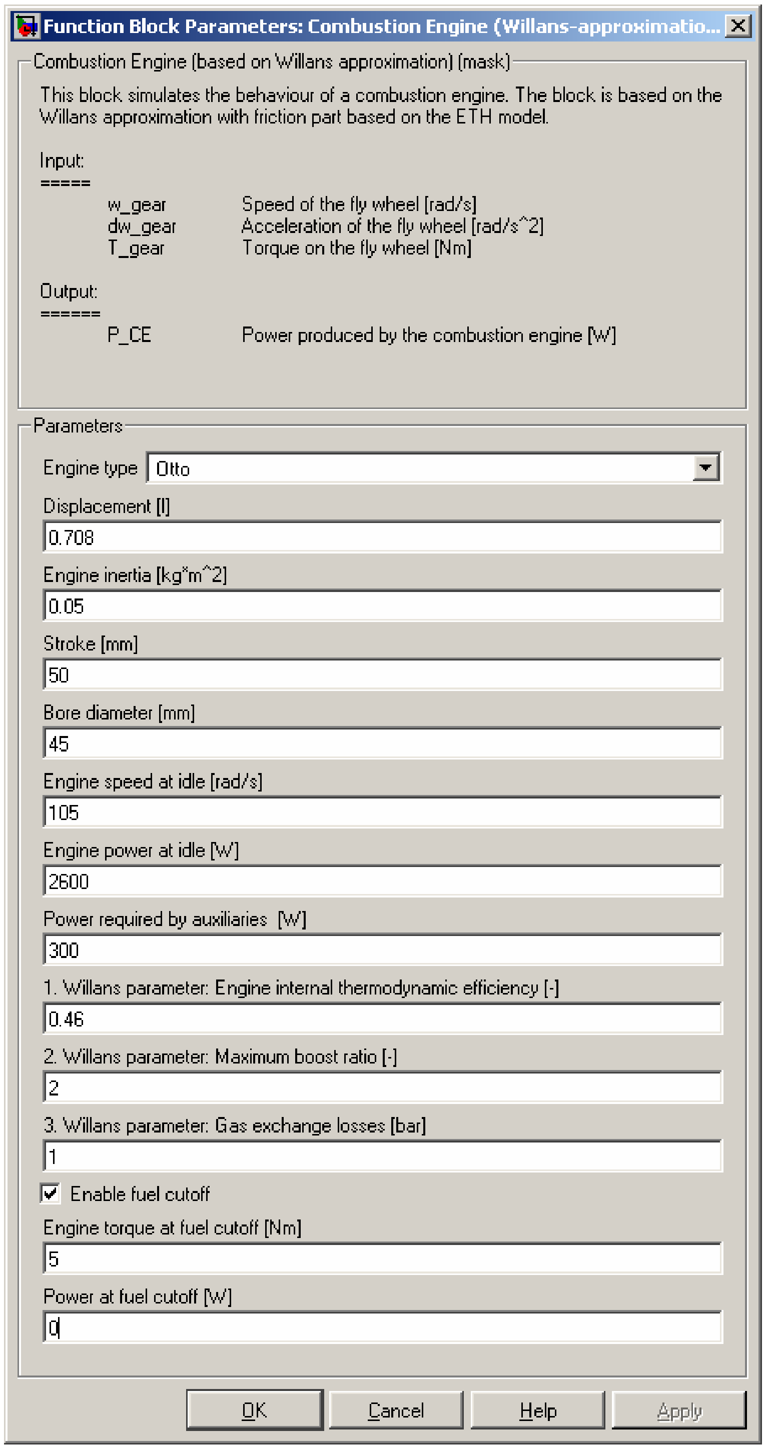


Fig. 2.3.5 Computation of the engine's efficiency according to the Willans approach


Function Block Parameters: Combustion Engine (Willans-approximatio...

Combustion Engine (based on Willans approximation) (mask)

This block simulates the behaviour of a combustion engine. The block is based on the Willans approximation with friction part based on the ETH model.

```
Input:
=====
         w_gear          Speed of the fly wheel [rad/s]
         dw_gear         Acceleration of the fly wheel [rad/s^2]
         T_gear          Torque on the fly wheel [Nm]

Output:
======
         P_CE            Power produced by the combustion engine [W]
```

Parameters

Engine type	Otto
Displacement [l]	0.708
Engine inertia [kg*m^2]	0.05
Stroke [mm]	50
Bore diameter [mm]	45
Engine speed at idle [rad/s]	105
Engine power at idle [W]	2600
Power required by auxiliaries [W]	300
1. Willans parameter: Engine internal thermodynamic efficiency [-]	0.46
2. Willans parameter: Maximum boost ratio [-]	2
3. Willans parameter: Gas exchange losses [bar]	1
Enable fuel cutoff	✓
Engine torque at fuel cutoff [Nm]	5
Power at fuel cutoff [W]	0

Fig. 2.3.6 The user interface (mask) of the "Combustion Engine" block based on the Willans approximation

2.3.2 Electric Motor and Generator

The models for electric motors are similar to those of IC engines, i.e., their inputs consist of the demanded torque values at the rotational speed being considered. The output is the required electric power.
In addition, just about every electric motor can be used in either two or four quadrants, as long as a suitable power amplifier is available. Here, the power amplifier is not modeled since its losses are included in the overall motor losses. As does the IC engine, an electric motor obviously requires a maximum torque switch as well as a maximum rotational speed switch, leading to the top level of the model as shown in Fig. 2.3.7:

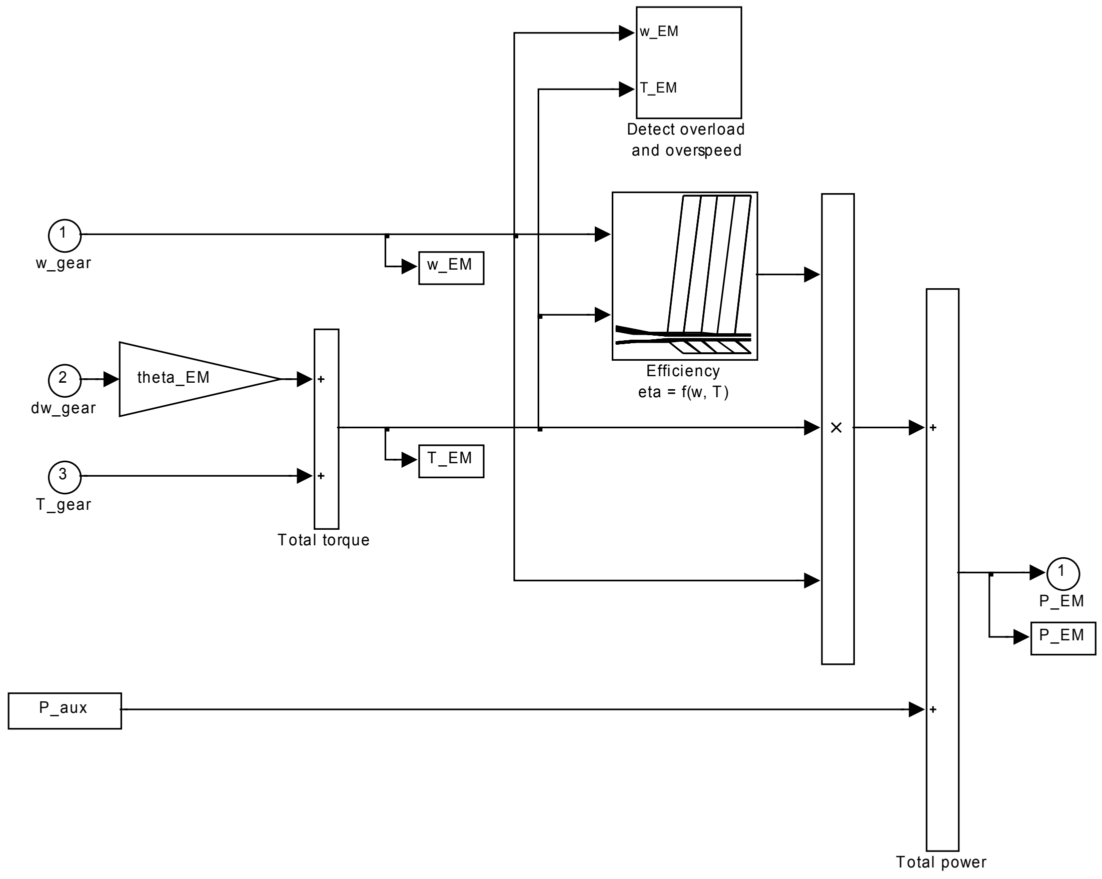


Fig. 2.3.7 QSS TB model of an electric motor in two-quadrant operation

The characteristics of electric motors may be entered in the model directly since they do not contain any singularities. The electric power required within the first quadrant ($\omega_{EM} > 0$ and $T_{EM} > 0$) can thus be expressed as follows:

$$P_{EM} = \omega_{EM} \cdot T_{EM} \cdot \frac{1}{\eta_{EM}(\omega_{EM}, T_{EM})} \tag{2.3.10}$$

or, in the second quadrant[4] ($\omega_{EM} > 0$ and $T_{EM} < 0$),

$$P_{EM} = \omega_{EM} \cdot T_{EM} \cdot \eta_{EM}(\omega_{EM}, T_{EM}) \tag{2.3.11}$$

respectively.

To avoid having to keep distinguishing between these two cases, equations 2.3.10 and 2.3.11 can be combined into a single efficiency map as shown in Fig. 2.3.8, with Eq. 2.3.11 being used for both quadrants.

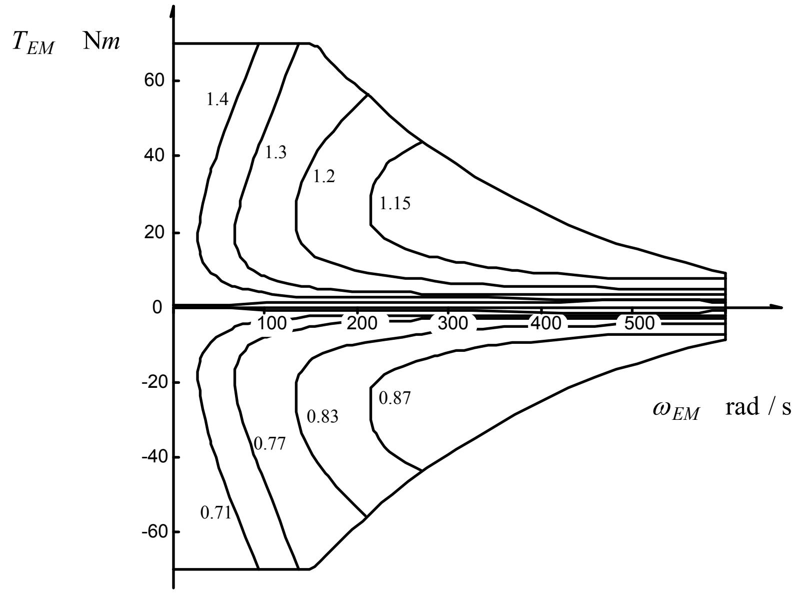


Fig. 2.3.8 Efficiency map of QSS TB electric motor in two-quadrant mode

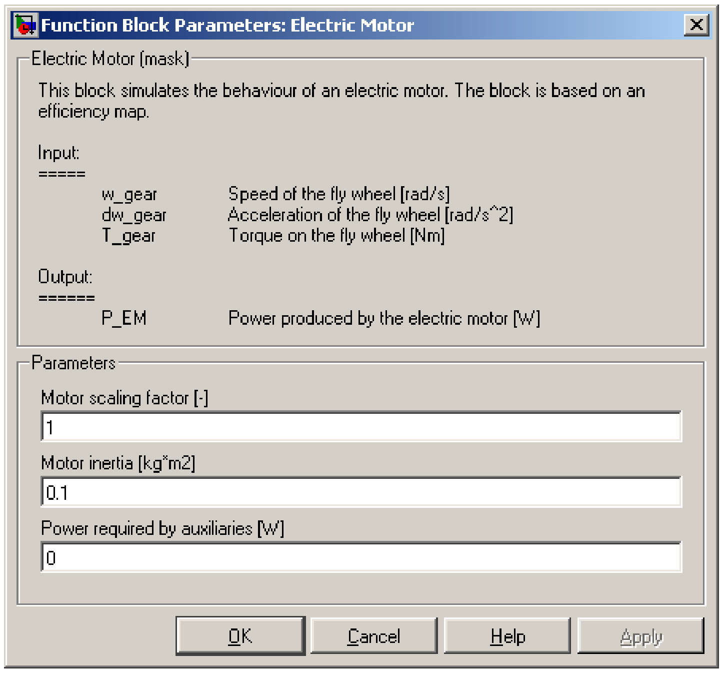
The design and data structure of this program map is identical to the one in IC engines, even though physically they have different meanings, i.e., the block "Efficiency" in Fig. 2.3.5 contains

- a 1 x n vector "*w_EM_row*" containing the support points of the rotational speed of the electric motor
- an m x 1 vector "*T_EM_col*" containing the support points of the motor torque
- an efficiency map n x m "*eta_EM_map*" representing the efficiencies of the relative combination of torque and rotational speed being considered.

An electric generator is described simply by the lower half of the second quadrant of the motor map shown in Fig. 2.3.8. With one exception, the remainder is unchanged. The exception is that since electric generators can be used only within the context of serial hybrid structure, i.e., in connection with a prime mover (e.g., IC engine, gas turbine, etc.), the sole function of the generator is to generate electric power from mechanical energy. Thus the inertia of the generator can always be included in the prime mover elements and therefore the block "Electric Generator" does not require an acceleration input of its own.

[4]Since driving in reverse is meaningless for the calculation of average fuel consumption, the third and fourth quadrants are not considered here.

Electric motors are scaled using a "mean effective pressure" formulation similar to the one common for IC engines or by simply scaling the support vectors and assuming the motor efficiencies to remain unaffected by that operation.


Function Block Parameters: Electric Motor

Electric Motor (mask)

This block simulates the behaviour of an electric motor. The block is based on an efficiency map.

```
Input:
=====
        w_gear          Speed of the fly wheel [rad/s]
        dw_gear         Acceleration of the fly wheel [rad/s^2]
        T_gear          Torque on the fly wheel [Nm]

Output:
======
        P_EM            Power produced by the electric motor [W]
```

Parameters

Motor scaling factor [-]: 1

Motor inertia [kg*m2]: 0.1

Power required by auxiliaries [W]: 0

Fig. 2.3.9 The user interface (mask) of the “Electric Motor” block

2.3.3 Fuel Cell

A fuel cell consists of a complex group of systems, as schematically shown in Fig. 2.3.10. They all must be powered dynamically which means that the drive system must be able to frequently switch from idle to the equivalent of a wide open throttle and back in order to keep up with the constantly changing states of operation. The dynamics and the duration depend on the size of the fuel cell, the drive cycle, and the configuration of the drive system.

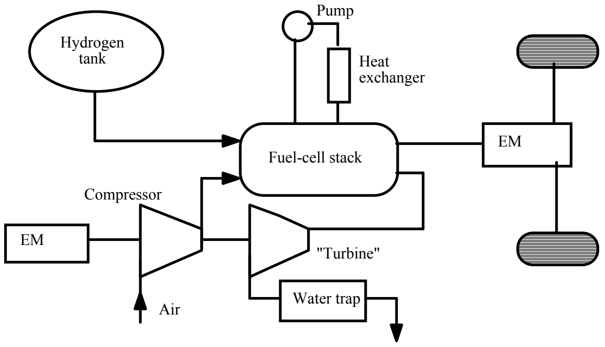


Fig. 2.3.10 Configuration of a fuel cell drive system

The energy conversion in a fuel cell depends on its electrodes being supplied with hydrogen and air or oxygen. The supply of the former is ensured if hydrogen is furnished from a tank within the vehicle as the fuel cell simply "draws" the amount it requires. The supply with air is more complicated. Under dynamic conditions the amount and pressure of the air depend on the load at the fuel cell. Depending upon response times, air mass flows, and charge-air pressures, the compressor may require an amount of electric power that severely affects the total efficiency of the drive system.

Modeling of a fuel cell
Just as in a battery, the voltage in a fuel cells decreases under load conditions. The maximum power usually is reached at around 0.5 to 0.6 V single cell voltage. The voltage drop is a function of the current and the internal resistance of the cell (on either side of the electric motor), of the kinetics of the electrode (primarily on the oxygen side), of limitations present in the gas supply, and of the water discharge on the oxygen side of the fuel cell. A voltage level that is practically utilizable is reached by stacking fuel cells in a serial configuration. Such a fuel cell stack, however, must take into account the temperature and humidity control of the reaction gases and it must enable the controlled transport of the heat, water, and electrical energy thus generated (cf. Fig. 2.3.11).

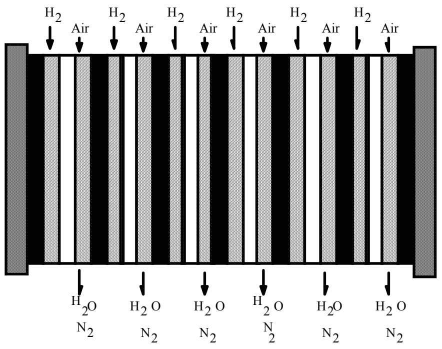


Fig. 2.3.11 Diagram of a fuel cell stack

Flow diagram and model
Fig. 2.3.12 shows the simplified diagram of an ideal fuel cell. The voltage drop in the cell is represented by a resistor

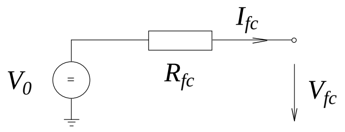


Fig. 2.3.12 Simplified electric model of a fuel cell

Since the resistance R_{fc} is a function of the area F_{fc} of the exchange membrane described above, the current is usually normalized to F_{fc}, i.e.

$$i_{fc} = \frac{I_{fc}}{F_{fc}} \tag{2.3.12}$$

yielding, in a first approximation, the following voltage for the fuel cell:

$$V_{fc} = V_0 - R_{fc} \cdot F_{fc} \cdot i_{fc} = V_0 - \tilde{R}_{fc} \cdot i_{fc} \tag{2.3.13}$$

With reference to Fig. 2.3.14 below, the variable V_0 represents the voltage at which the ideal characteristic intersects the abscissa at $i_{fc} = 0$, rather than the theoretical voltage with the fuel cell. (For a detailed explanation, please refer to the lecture "Vehicle Propulsion Systems."). The resistance $\tilde{R}_{fc}$ $[\Omega \cdot \mathrm{m}^2]$ is constant for any one fuel cell type and thus no longer a function of the surface area of the cell membrane. As described above, a so-called stack (cf. Fig. 2.3.13) is obtained when a number of fuel cells are serially linked. The stack voltage is

$$V_{fc,stack} = N \cdot V_{fc} \tag{2.3.14}$$

If several stacks are joined in parallel, the model described above would still be valid. However, the increase of the fuel cell area would lead to an increase in current.

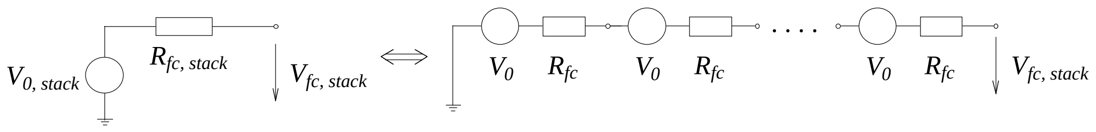


Fig. 2.3.13 Simplified diagram of a fuel cell stack

The model of a fuel cell stack in the QSS TB assumes that a power amplifier adjusts the voltage level automatically. The current level I_{fc} within the fuel cell can thus be represented by the following equation:

$$I_{fc} = \frac{P_{ex} - P_{vc}}{V_{fc}}, \tag{2.3.15}$$

where the variable P_{ex} represents the current demanded and P_{vc} the amount of the losses within the internal condenser. However, since the voltage in the denominator of (2.3.15) is a function of the fuel cell current (see eq. (2.3.13)), eq. (2.3.15) results in an implicit equation.

For the approach shown in eq. (2.3.13), this loop may be resolved. However, since at small current densities especially, the relationship $V_{fc}(i_{fc})$ is extremely nonlinear and significantly deviates from those results, the QSS TB model does not attempt to resolve it. Instead, an additional delay is introduced (cf. Fig. 2.3.15) which breaks that implicit loop.

A fuel cell produces significant amounts of heat as well:

$$P_{loss} = \left(V_{th} - V_{fc}\right) \cdot N \cdot I_{fc} \tag{2.3.16}$$

which must be dealt with by providing for an appropriate coolant system.

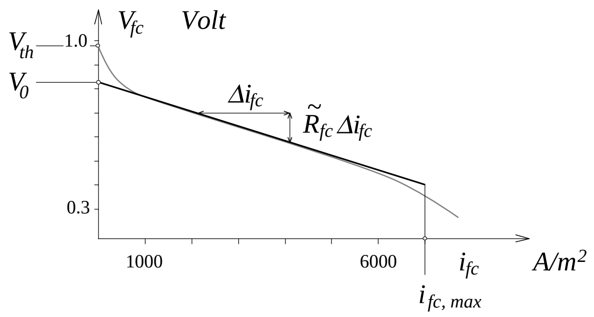


Fig. 2.3.14 Voltage characteristics of a fuel cell

A number of parameter values typically seen in such a fuel cell model are shown in Fig. 2.3.17. Figure 2.3.16 finally shows the model for the intake air compressor.

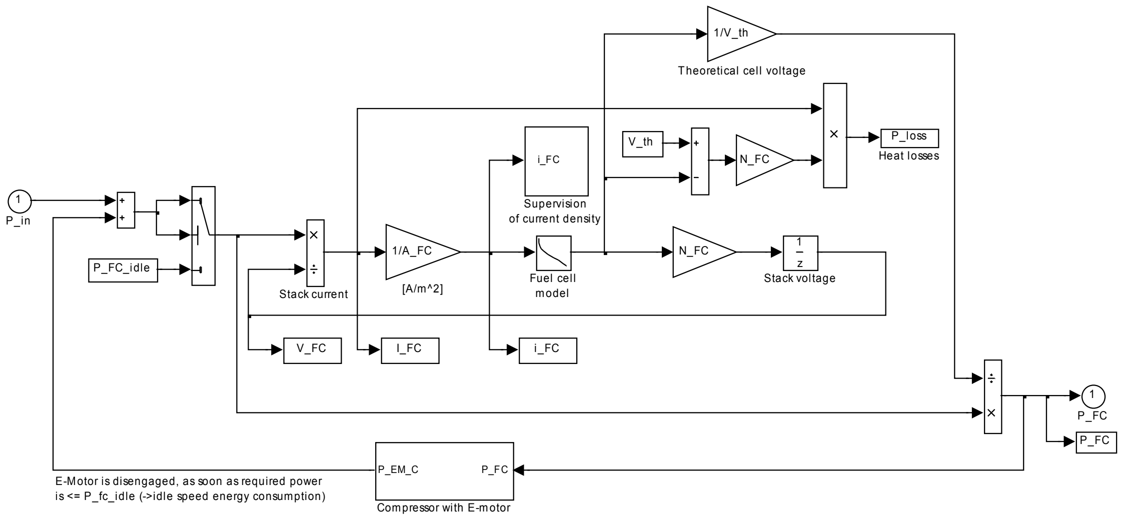


Fig. 2.3.15 QSS TB model of a fuel cell

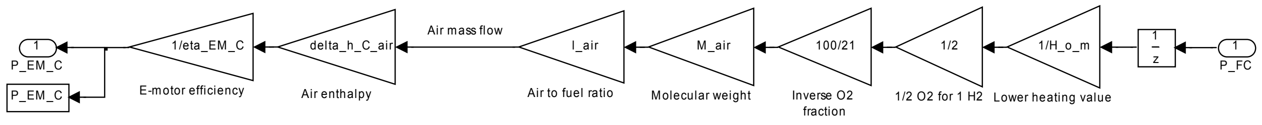
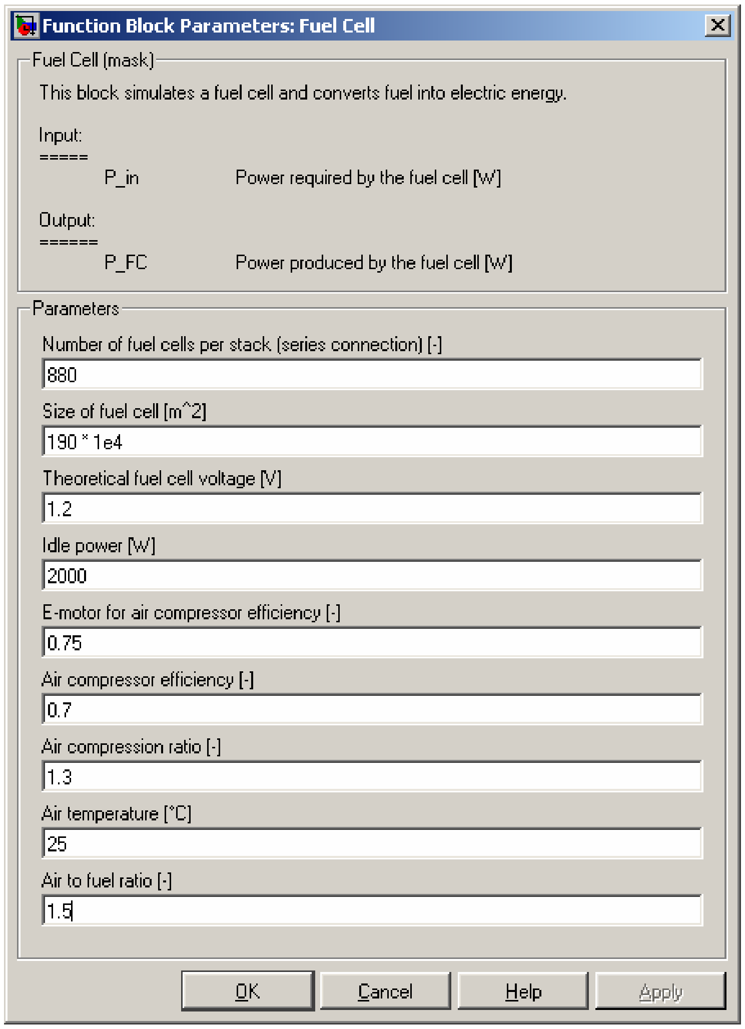


Fig. 2.3.16 Model of the compressor



Function Block Parameters: Fuel Cell

Fuel Cell (mask)

This block simulates a fuel cell and converts fuel into electric energy.

Input:
=====
P_in Power required by the fuel cell [W]

Output:
======
P_FC Power produced by the fuel cell [W]

Parameters

Number of fuel cells per stack (series connection) [-]	880
Size of fuel cell [m^2]	190 * 1e4
Theoretical fuel cell voltage [V]	1.2
Idle power [W]	2000
E-motor for air compressor efficiency [-]	0.75
Air compressor efficiency [-]	0.7
Air compression ratio [-]	1.3
Air temperature [°C]	25
Air to fuel ratio [-]	1.5

Fig. 2.3.17 The user interface (mask) of the "Fuel Cell" block

2.4 Gear System

The transmission of mechanical work between different levels of torque or rotational speed, respectively, is possible due to the following three building blocks:

- simple transmission (i.e., fixed relationships between torque levels or rotational speed levels)
- manual gear box (i.e., a finite number of relationships between torque levels or rotational speed levels)
- CVT (i.e., continuously variable relationships between torque or rotational speed levels, respectively).

Manual gear boxes and CVTs can be controlled either by the drive cycle block, i.e. the gear ratios are defined a priori, or by an online controller.

2.4.1 Simple Transmission

As shown in Fig. 2.4.1 for the simplest case, the kinematical relationships are assumed to be ideal, i.e., there are no inertia effects by backlashes.

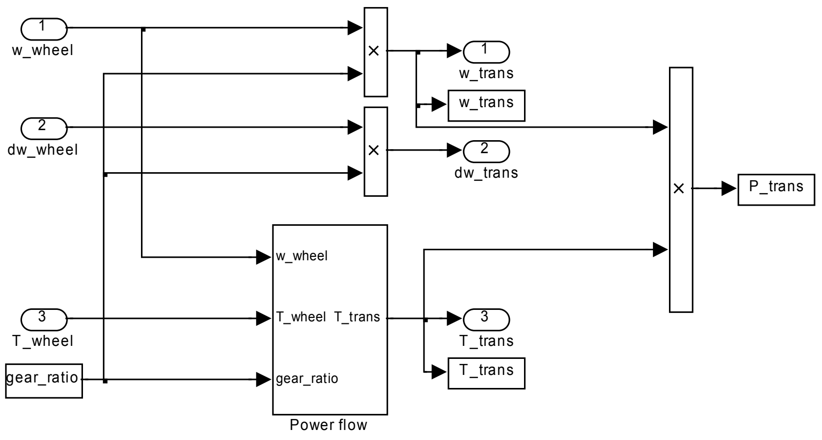


Fig. 2.4.1 Top level of the block "Simple Transmission", showing kinematical relationships on top and torque at the bottom

In all three types of gearbox systems, all losses are described by an affine relationship

$$P_{out} = e \cdot P_{in} - P_0 \tag{2.4.1}$$

where P_{out} and P_{in} represent the power leaving and entering the system, respectively. Since it is possible that the flow is reversed (e.g., between the two phases "drive" and "fuel cutoff"), equation 2.4.1 has to be interpreted case by case. Fig. 2.4.2 shows the solution presented by the QSS TB for the simplest case. Equation 2.4.1 is solved within the third level, once with the driving element being the torque on the engine side (i.e., force flowing from the engine to the wheel), and once with the driving element being the torque on the side of the wheel. The relationships resulting therefrom are shown in Fig. 2.4.3, whereby it is to be noted that in the case of the power direction wheel-to-engine, the idle speed losses $\tilde{P}_0$ are negative, i.e., the relationship $\tilde{P}_0 = -P_0$ holds (P_0 represents the idle speed loss in the "normal" case of the power flowing in the engine-to-wheel direction).

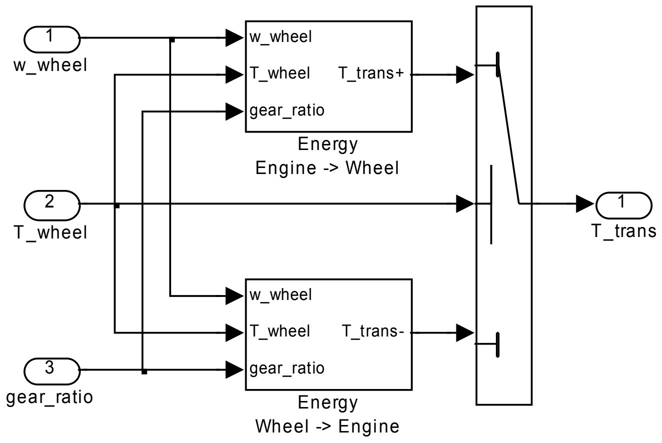


Fig. 2.4.2 Differentiation according to flow direction in the block "Power flow"

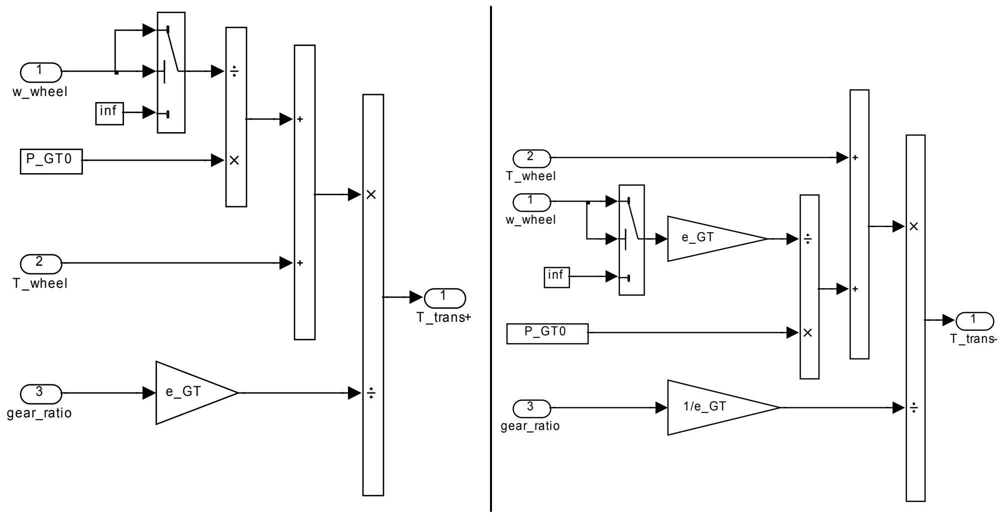
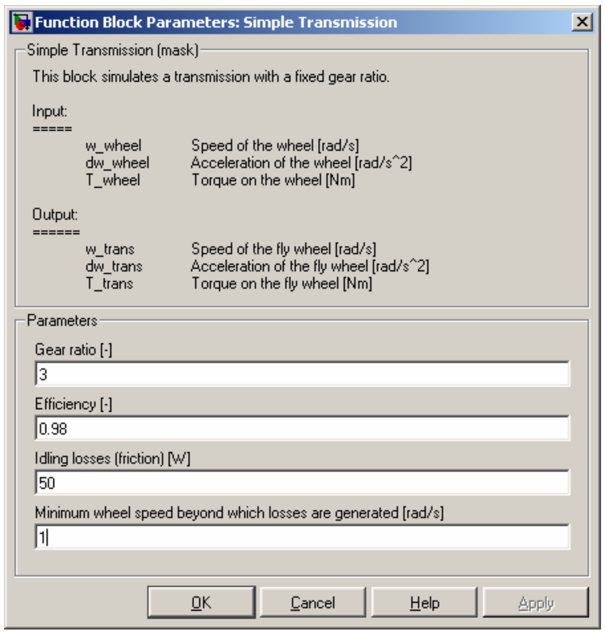


Fig. 2.4.3 Torque calculation in the engine: a) representing the flow engine-to-wheel, b) representing the flow wheel-to-engine



Function Block Parameters: Simple Transmission

Simple Transmission (mask)

This block simulates a transmission with a fixed gear ratio.

```
Input:
=====
        w_wheel         Speed of the wheel [rad/s]
        dw_wheel        Acceleration of the wheel [rad/s^2]
        T_wheel         Torque on the wheel [Nm]

Output:
======
        w_trans         Speed of the fly wheel [rad/s]
        dw_trans        Acceleration of the fly wheel [rad/s^2]
        T_trans         Torque on the fly wheel [Nm]
```

Parameters

Gear ratio [-]: 3

Efficiency [-]: 0.98

Idling losses (friction) [W]: 50

Minimum wheel speed beyond which losses are generated [rad/s]: 1

Fig. 2.4.4 The user interface (mask) of the "Simple Transmission" block

2.4.2 Manual Gear Box

The basic modeling approach of the "Manual Gear Box" block is the same as the "Simple Transmission" block of Fig. 2.4.1 above; the only difference is that in the former case the user can specify the fixed gear ratios of the gear box (a 5-gear shift manual gear box is assumed here).

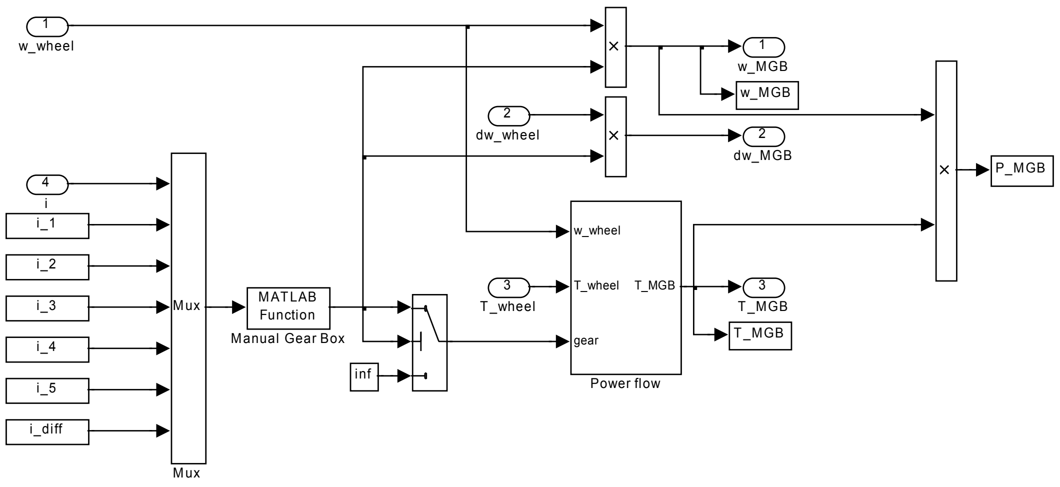
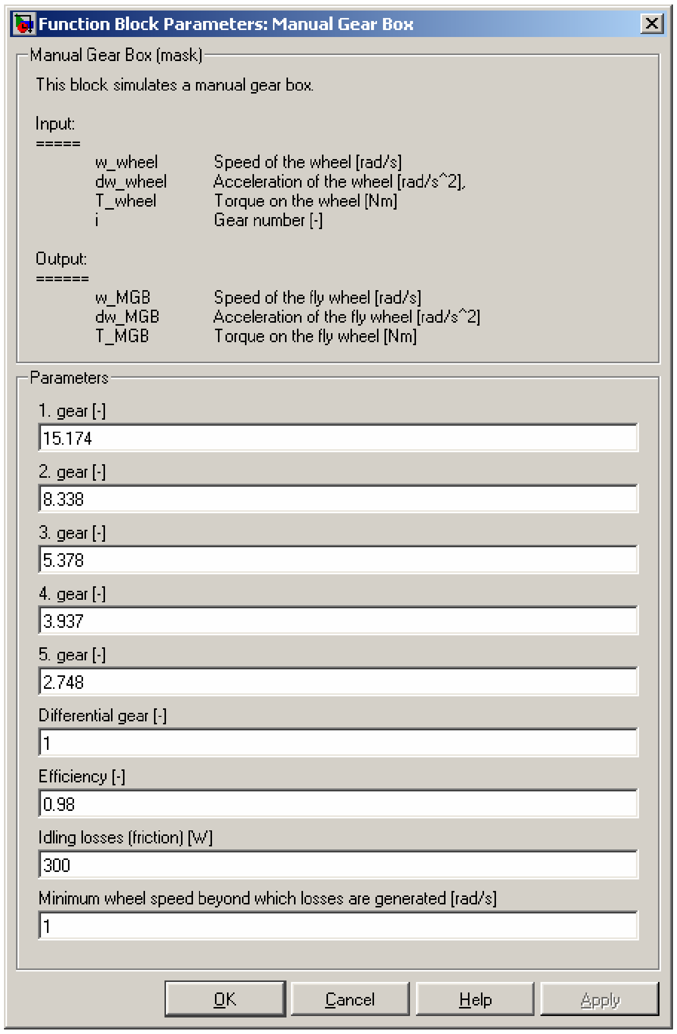


Fig. 2.4.5 Top level of the block "Manual Gear Box": notice the similarity with Fig. 2.4.1 and the Matlab function to compute the actual gear ratio

Therefore an additional input, i.e. the gear number generated by the “Driving Cycle” block (see section 2.1 above) is needed. An m-function named “*manual_gearbox*” located in the folder “*\Functions”* computes the gear ratio at the actual cycle time (see Fig. 2.4.5).


Function Block Parameters: Manual Gear Box

Manual Gear Box (mask)

This block simulates a manual gear box.

```
Input:
=====
        w_wheel         Speed of the wheel [rad/s]
        dw_wheel        Acceleration of the wheel [rad/s^2],
        T_wheel         Torque on the wheel [Nm]
        i               Gear number [-]

Output:
======
        w_MGB           Speed of the fly wheel [rad/s]
        dw_MGB          Acceleration of the fly wheel [rad/s^2]
        T_MGB           Torque on the fly wheel [Nm]
```

Parameters

1. gear [-]	15.174
2. gear [-]	8.338
3. gear [-]	5.378
4. gear [-]	3.937
5. gear [-]	2.748
Differential gear [-]	1
Efficiency [-]	0.98
Idling losses (friction) [W]	300
Minimum wheel speed beyond which losses are generated [rad/s]	1

Fig. 2.4.6 The user interface (mask) of the “Manual Gear Box” block

2.4.3 Continuously Variable Transmission (CVT)

The physics of CVTs in the "forward" formulation are somewhat more complicated. In a QSS formulation, however, the structure shown in Fig. 2.4.7 arises. This rather surprisingly straightforward model structure is a consequence of the QSS approach.
The online control of CVTs demands that it be "predictive". One of the ways to achieve this is to transmit the desired speed generated by the block "Driving Cycle" directly to the control unit, but to delay its transmission to the block "Vehicle" by one step (this means that the unit delay blocks in Simulink are set to a sample time equal to the step size *h*).
This would be easy to implement in practice, i.e., with a drive-by-wire engine, the step size (being the artificial delay) would be on the order of one tenth of a second.

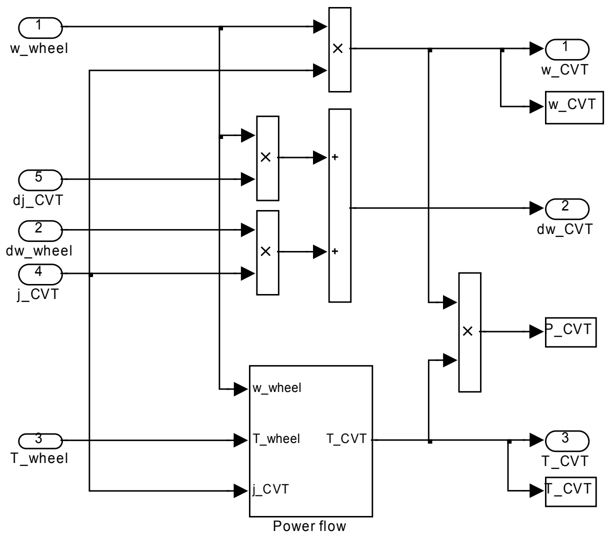
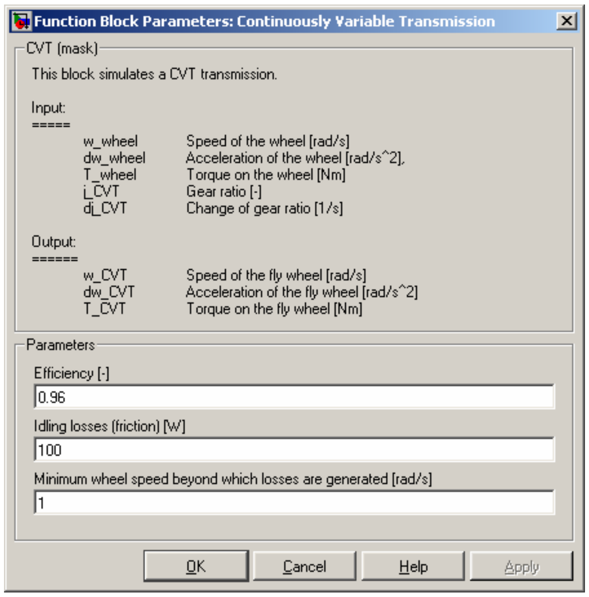


Fig. 2.4.7 **Top level CVT block, the remaining blocks being identical to those of Figs. 2.4.2 and 2.4.3**

Comments:
So far, the parameters of the gearbox efficiencies *e* and P_0 have been modeled as being invariable, in particular, as being independent of the gear ratio. However, it would be quite simple to extend the system to consider such variations as well.
The gear systems discussed thus far have all been mechanical. The QSS TB has not been including any hydrodynamic torque converters, as they are not candidates for use with highly economic drive systems. However, it would be easy enough to derive the respective models.


Function Block Parameters: Continuously Variable Transmission

CVT (mask)

This block simulates a CVT transmission.

```
Input:
=====
        w_wheel      Speed of the wheel [rad/s]
        dw_wheel     Acceleration of the wheel [rad/s^2],
        T_wheel      Torque on the wheel [Nm]
        i_CVT        Gear ratio [-]
        di_CVT       Change of gear ratio [1/s]

Output:
======
        w_CVT        Speed of the fly wheel [rad/s]
        dw_CVT       Acceleration of the fly wheel [rad/s^2]
        T_CVT        Torque on the fly wheel [Nm]
```

Parameters

Efficiency [-]: 0.96

Idling losses (friction) [W]: 100

Minimum wheel speed beyond which losses are generated [rad/s]: 1

Fig. 2.4.8 **The user interface (mask) of the "CVT" block**

2.5 Energy Buffer

There are two types of energy buffers included in the QSS TB library:

1. Battery
2. Supercapacitor

2.5.1 Battery

The models of batteries are designed in accordance with the approach introduced in the lecture "Vehicle Propulsion Systems", i.e., the electric power "*P_BT*" flowing into the battery (negative when loading, positive during discharge) represents the input, while the actual battery charge "*Q_BT*" represents the output. Fig. 2.5.1 shows the top level of the battery model. Beyond the battery model proper (gray shading), this block calculates the amount of electric power (in kWh per 100 km) drawn up to a certain instance ("*Q_BT_IC*" representing the initial charge of the battery).

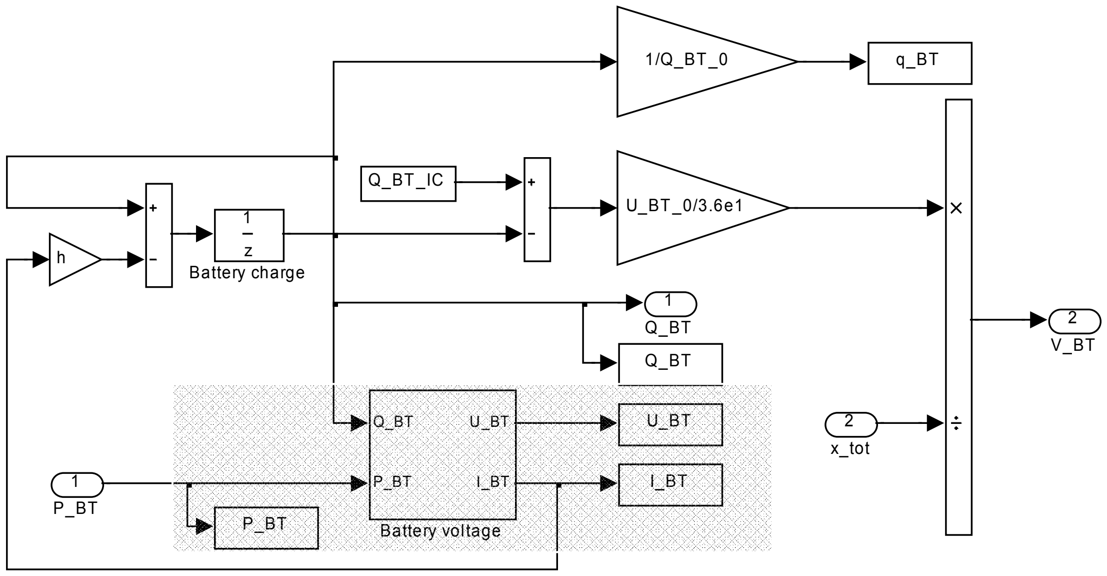


Fig. 2.5.1 Top level of the battery model, calculation of power drawn in kWh / 100 km

The basic idea behind the battery model is to integrate the power (in accordance with the QSS approach) after calculating it from the known power and actual voltage of the battery, in order to then be able to calculate the actual battery charge therefrom. The voltage depends on the battery charge, which is easy to determine, and from the current charging or discharging the battery which – unfortunately – is not yet known. This second dependency thus leads to an implicit equation for the battery power. Although the solver in Simulink frequently is able to solve such equations, it is advantageous to first obtain a closed-form solution to this implicit loop. This is the case especially whenever the battery is being utilized in an infeasible operating mode. These derivations are shown below.

First, the modes "Charging," "Discharging" and "Idle" must be distinguished. Whereas the parameters usually vary in the first two modes, in "neutral" the terminals just show the idle voltage of the battery.

As shown in Fig. 2.5.2, it is mandatory to keep close tabs on the battery since neither extremely high charging nor discharging loads are permissible (details are shown below). The block "Supervision" is designed to assume this supervising function.

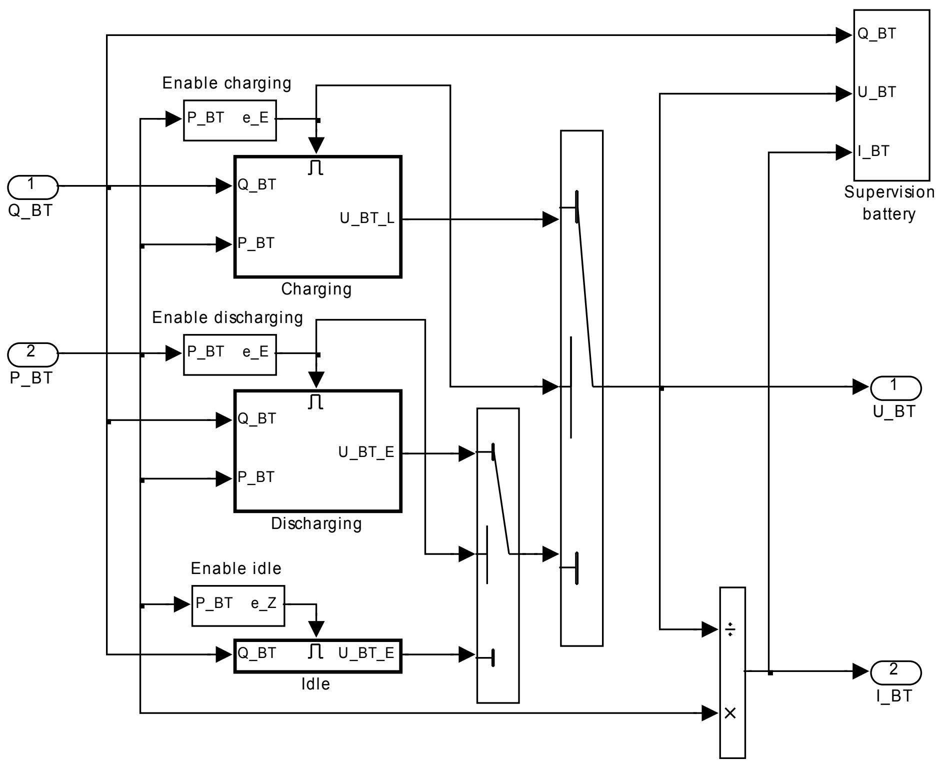


Fig. 2.5.2 Distinction among the cases "Charging", "Discharging," and "Idle"

The equations for the mode "Charging" are the following:

Charge rate:
$$q(k \cdot h) = \frac{Q(k \cdot h)}{Q_0}, \quad Q(k \cdot h) = Q(0) + \sum_k i(k \cdot h) \cdot h \tag{2.5.1}$$
(Q_0 is the nominal capacity, $Q(0)$ the charge at $t = 0$)

C rate:
$$c(k \cdot h) = \frac{i(k \cdot h)}{i_0}, \quad i_0 = \frac{Q_0}{1\ h} \tag{2.5.2}$$
(i_0 fully charges an empty battery in one hour)

They allow the use of the typical approach (affine in the charge ratio) to calculate the battery load

$$u_{BL}(k \cdot h) = u_{1L}(c(k \cdot h)) \cdot q(k \cdot h) + u_{0L}(c(k \cdot h)) \tag{2.5.3}$$

whereby the two weights u_{1L} and u_{0L} depend on the C rate as well

$$u_{1L}(c(k \cdot h)) = c_{L4} \cdot c(k \cdot h) + c_{L3}, \quad u_{0L}(c(k \cdot h)) = c_{L2} \cdot c(k \cdot h) + c_{L1}. \tag{2.5.4}$$

The equations for the mode "Discharging" are similar, except for the parameters in Eq. (2.5.4) where obviously the C rate is negative

$$u_1(c(k \cdot h)) = c_{E4} \cdot c(k \cdot h) + c_{E3}, \quad u_0(c(k \cdot h)) = c_{E2} \cdot c(k \cdot h) + c_{E1}. \tag{2.5.5}$$

The eight coefficients in Eqs. (2.5.4) and (2.5.5) may not be chosen entirely freely. For continuity reasons, they must fulfill the following conditions:

$$c_{E1} = c_{L1}, \qquad c_{E3} = c_{L3}. \tag{2.5.6}$$

Based upon the power (Eq. (2.5.3)) and the voltage (Eq. (2.5.4)) of the battery, the value of the current flow in or out of the battery, respectively, can be eliminated, which permits the presentation of the voltage as a function of power only. For the charging mode the following relation is obtained:

$$u_{BE}(k \cdot h) = \frac{1}{2} \cdot \left(c_{E3} \cdot q(k \cdot h) + c_{E1} + \sqrt{\left(c_{E3} \cdot q(k \cdot h) + c_{E1}\right)^2 + 4 \cdot \left(c_{E4} \cdot q(k \cdot h) + c_{E2}\right) \cdot P_{BL}(k \cdot h) / i_0} \right) \tag{2.5.7}$$

The ambiguity in the solution of the quadratic equation obtained can be resolved using physical arguments.

The solution for the discharging mode is obtained following the same approach. These two equations are then incorporated in the programs of the blocks "Charging" and "Discharging", respectively.
The maximum storage capacity of a battery sets physical limitations to the amount of positive power (in the "Charging" case) that may enter the battery. For reasons of simplicity, it is assumed that a constant maximum current cannot be exceeded, or else the simulation is terminated.
In the opposite case ("Discharging"), the amount of current a battery can deliver is limited by the internal resistance that is always present. As stated above, the power delivered by a battery is given by

$$P_{BE}(k \cdot h) = u_{BE}(k \cdot h) \cdot i(k \cdot h). \tag{2.5.8}$$

Rewriting Eqs. (2.5.3) and (2.5.4), we obtain

$$u_{BE}(k \cdot h) = \left(c_{E4} \cdot q(k \cdot h) + c_{E2}\right) \cdot \frac{i(k \cdot h)}{i_0} + \left(c_{E3} \cdot q(k \cdot h) + c_{E1}\right) = R_{Ei}(k \cdot h) \cdot i(k \cdot h) + u_{Ei}(k \cdot h) \tag{2.5.9}$$

which permits an interpretation that includes both internal resistance and internal voltage (obviously, both values depend on the charge or discharge status of the battery, respectively). The battery power thus can be determined as follows:

$$P_{BE}(k \cdot h) = u_{BE}(k \cdot h) \cdot \frac{u_{BE}(k \cdot h) - u_{Ei}(k \cdot h)}{R_{Ei}(k \cdot h)}. \tag{2.5.10}$$

Deriving Eq. (2.5.10) from the voltage u_{BE}, we state that the maximum of the amount of discharge power (i.e., the minimum level of P_{BE}) is delivered at a terminal voltage that is one-half the amount of the internal voltage:

$$u^*_{BE} = \frac{1}{2} \cdot u_{Ei} \quad \Rightarrow \quad P^*_{BE} = -\frac{1}{4} \cdot \frac{u^2_{Ei}}{R_{Ei}}. \tag{2.5.11}$$

Therefore, it is clear that the battery is never able to deliver more than P^*_{BE}. The model verifies this basic limitation by terminating the simulation as soon as the value of P^*_{BE} is reached.

The last point to be addressed is the question of the efficiency of a battery. It is not an obvious quantity, especially it may be defined locally or „globally“ (over an entire cycle, i.e., the total power drawn from the battery during a test drive divided by the energy required to recharge it). In the local approach, it is assumed that within one time step *h* the battery is charged by a constant current, only to be discharged in the next step by the same current:

$$\eta_{BT} = \frac{u_{BE}\cdot(-I_E)\cdot h}{u_{BL}\cdot I_L\cdot h} = \frac{u_{BE}}{u_{BL}} = \frac{u_{1E}\cdot q + u_{0E}}{u_{1L}\cdot q + u_{0L}} = \frac{(-c_{E4}\cdot c + c_{E3})\cdot q + (-c_{E2}\cdot c + c_{E1})}{(c_{L4}\cdot c + c_{L3})\cdot q + (c_{L2}\cdot c + c_{L1})}\,. \tag{2.5.12}$$

The local efficiency thus is a function of the current charging or discharging the battery, respectively, and of the actual battery charge level $\eta_{BT}(c, q)$. The example depicted in Fig. 2.5.3 shows such an efficiency map for a good battery and how strongly the efficiency depends on the C rate (or internal resistance) rather than on the weak relationship to the charge level.

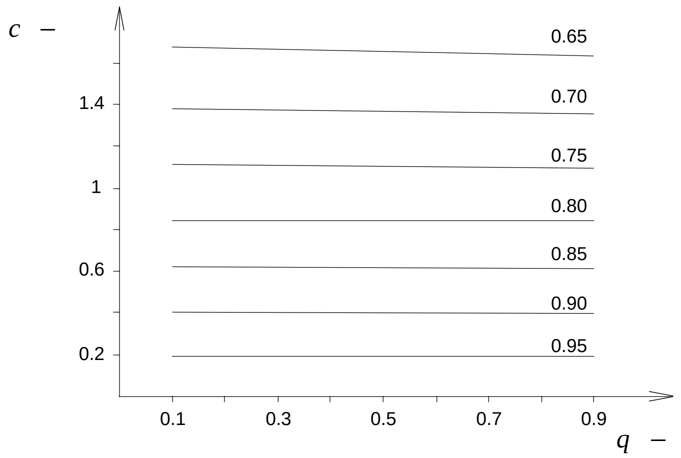
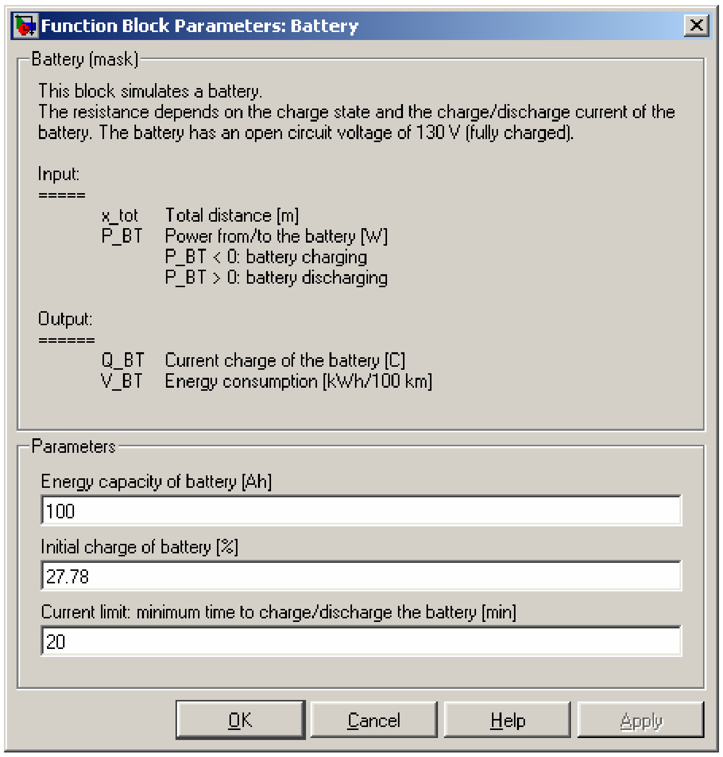


Fig. 2.5.3 Local efficiency map of a battery; charge level 10% – 90%; C rate 0 – 200%

Comments:
The model described above is valid only for the normal operating range of the battery, i.e. between approximately 10% and 90% of its nominal capacity. Outside of this range, rather strong nonlinear effects may be observed. However, it is a well-known fact that batteries should not be operated in those outside ranges due to serious effects on their longevity, efficiency, etc.
During the vehicle acceleration phase, batteries are subject to rather brief, but high peaks of energy demands which either overload them or are the reason for a vehicle being equipped with unnecessarily large batteries. This situation may be avoided by the installation of so-called supercapacitors, which are capacitors with very large capacity (cf. Sec. 2.5.2 below).
The electronic power system controls the amounts of the current running between the battery and other electrical components as well as among those components themselves. The details of these losses are not being considered or modeled separately; rather they can be ascribed to the losses of the motors or generators.



Battery (mask)

This block simulates a battery.
The resistance depends on the charge state and the charge/discharge current of the battery. The battery has an open circuit voltage of 130 V (fully charged).

```
Input:
=====
        x_tot    Total distance [m]
        P_BT     Power from/to the battery [W]
                 P_BT < 0: battery charging
                 P_BT > 0: battery discharging

Output:
======
        Q_BT     Current charge of the battery [C]
        V_BT     Energy consumption [kWh/100 km]
```

Parameters

Energy capacity of battery [Ah]: 100

Initial charge of battery [%]: 27.78

Current limit: minimum time to charge/discharge the battery [min]: 20

Fig. 2.5.4 The user interface (mask) of the “Battery” block

2.5.2 *Supercapacitor*

Supercapacitors (or “supercaps”) can store energy in electrostatic fields and are able to attain very high power densities (realistically around 1 kW per kg). Although their energy densities are moderate (i.e., 2–5 Wh/kg), their high power density still makes them ideally suited to function as electrical "peak shavers." If they are used together with batteries, they can thus contribute to a lower battery weight and an efficient use of energy.
The simplified model used in the QSS TB is shown in Fig. 2.5.5. The electrical power output, as demanded by a controller, is converted using the voltage U, resulting in a corresponding current flow I_{SC}.

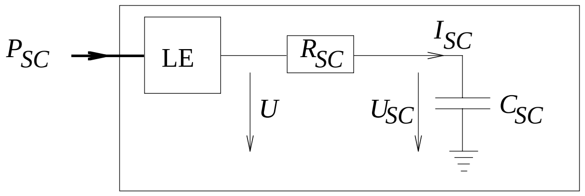


Fig. 2.5.5 Schematic flow diagram of the model of the "supercap" (LE = power amplifier)

The charge Q_{SC} stored within the "supercap" is related to the inner voltage U_{SC} as follows:

$$U_{SC} = \frac{1}{C_{SC}} \cdot Q_{SC} \qquad (2.5.13)$$

According to Kirchhoff's rule the value P_{SC} is obtained as follows:

$$P = \left(R_{SC} \cdot I_{SC} + \frac{1}{C_{SC}} \cdot Q_{SC} \right) \cdot I_{SC} \qquad (2.5.14)$$

The power output is prescribed by the controller, while the charge Q_{SC} is defined by the memory of the "supercap" (the charge being a state variable). The only remaining unknown is the current flow, which is calculated as follows:

$$\dot{Q}_{SC} = I_{SC} = \left(\frac{-Q_{SC}}{C_{SC}} \pm \sqrt{\left(\frac{Q_{SC}}{C_{SC}}\right)^2 + 4 \cdot P \cdot R_{SC}} \right) \Big/ \left(2 \cdot R_{SC}\right) \qquad (2.5.15)$$

whereby physical arguments preclude the negative sign before a square root.

Fig. 2.5.6 shows how this relationship is presented in the QSS TB. The block "Supervision" tests whether the "supercap" is being used within its permitted operating range, i.e., analogously to batteries, the amount of power drawn from a "supercap" cannot exceed the amount lost due to internal resistance. Details are shown in Fig. 2.5.7 below.

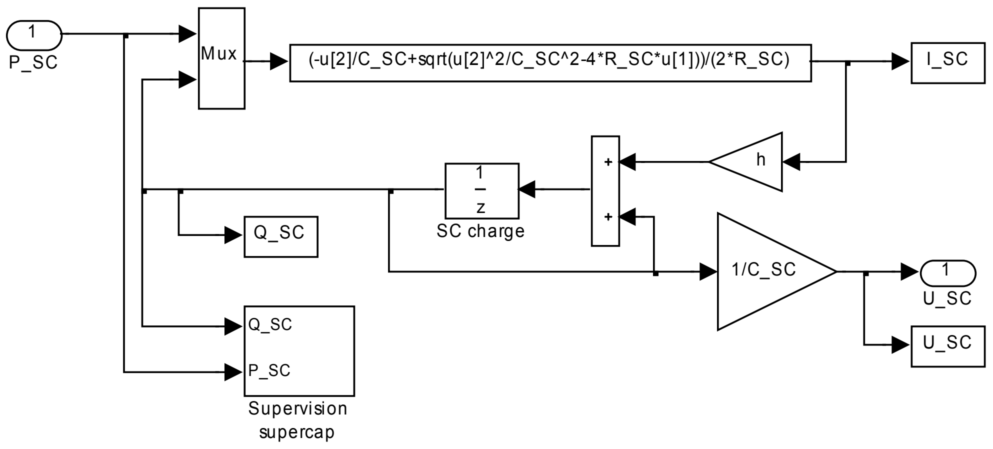


Fig. 2.5.6 **Top level of the hierarchy within the "Supercap" model of the QSS TB**

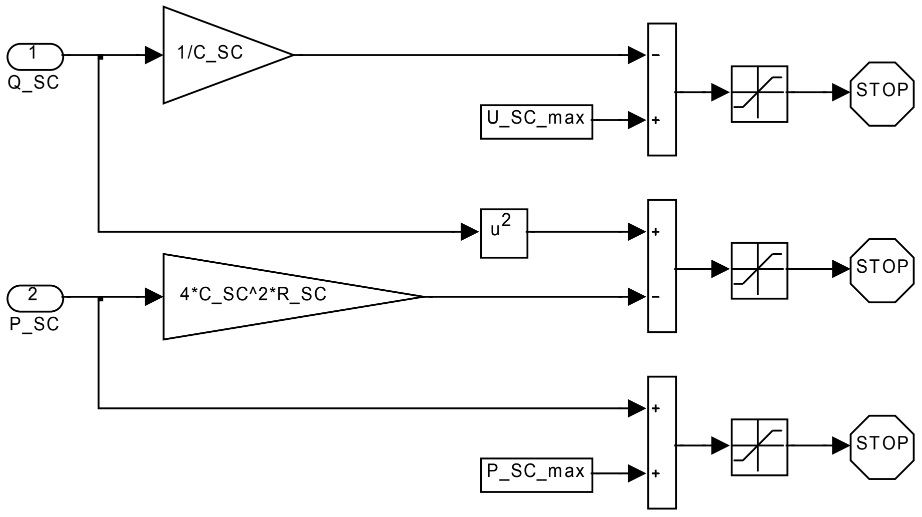
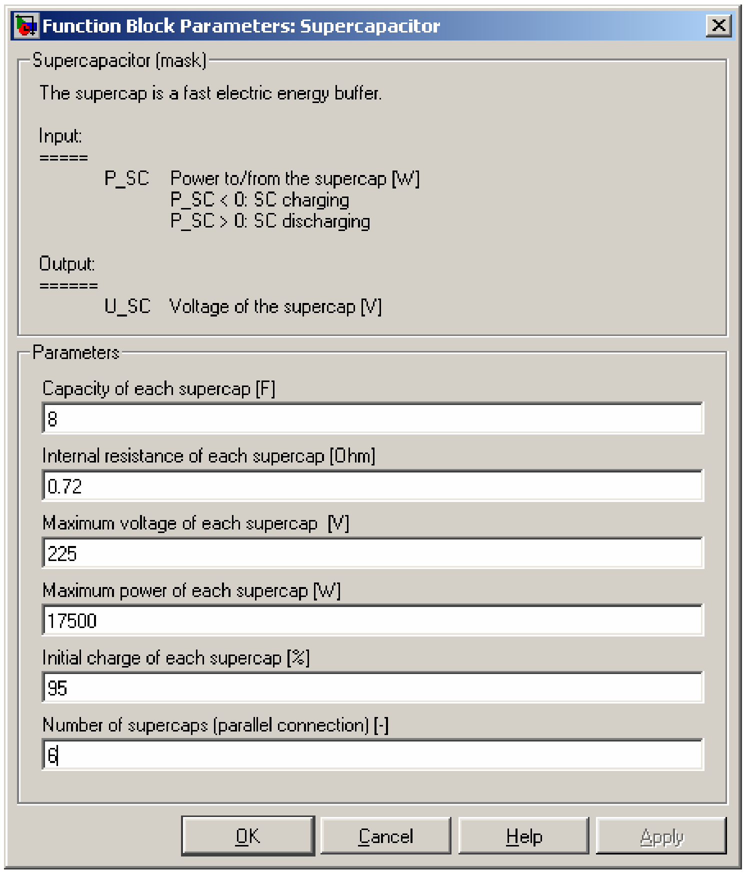


Fig. 2.5.7 Block „Supervision“



Supercapacitor (mask)

The supercap is a fast electric energy buffer.

```
Input:
=====
          P_SC    Power to/from the supercap [W]
                  P_SC < 0: SC charging
                  P_SC > 0: SC discharging

Output:
======
          U_SC    Voltage of the supercap [V]
```

Parameters

Capacity of each supercap [F]	8
Internal resistance of each supercap [Ohm]	0.72
Maximum voltage of each supercap [V]	225
Maximum power of each supercap [W]	17500
Initial charge of each supercap [%]	95
Number of supercaps (parallel connection) [-]	6

Fig. 2.5.8 The user interface (mask) of the “Supercapacitor” block

2.6 Energy Source

The required power of the combustion engine or the fuel cell is divided by the lower heating value of the corresponding fuel to compute the resulting actual fuel mass flow. Then the fuel mass flow is integrated in order to obtain the consumed fuel mass to complete the chosen driving cycle.

The total fuel mass is finally divided by the total distance and transformed in liters/100 km.

Any losses due to cold starts or ancillary devices are best considered via global factors. The losses due to cold starts are typically accounted for by multiplying the total fuel used by a factor of 1.15:

$$k_{cs} = 1.15 \tag{2.6.1}$$

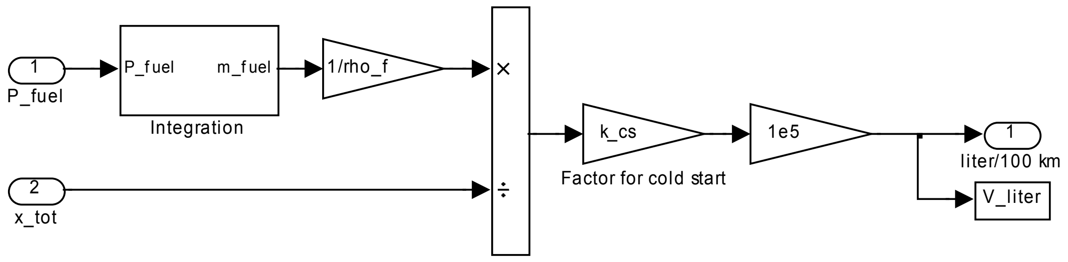
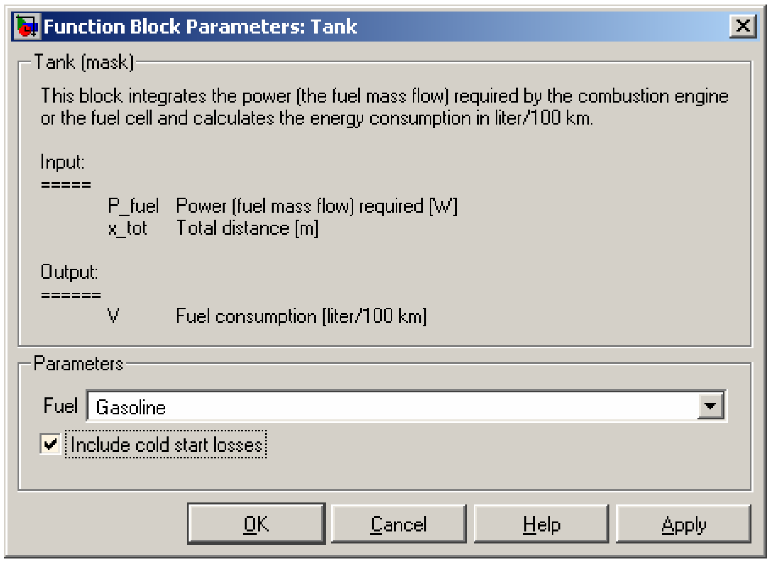


Fig. 2.6.1 Top level of the "Tank" model



Tank (mask)

This block integrates the power (the fuel mass flow) required by the combustion engine or the fuel cell and calculates the energy consumption in liter/100 km.

Input:
=====
P_fuel Power (fuel mass flow) required [W]
x_tot Total distance [m]

Output:
======
V Fuel consumption [liter/100 km]

Parameters

Fuel: Gasoline

[✓] Include cold start losses

Fig. 2.6.2 The user interface (mask) of the "Tank" block

2.7 Controller

Control systems in the QSS TB can be simple or quite complex. One example of the former is a system for which it is assumed that the demanded engine load is always present (cf. Sec. 1.3 above), whereas the control system of the series hybrid system described in Sec. 3.1 below is an example of a more complex system. The purpose of control systems in general is to process system data, external inputs (e.g. during simulations), and cycle data (i.e., control instructions set a priori), generating input data for the various system modules (see Fig. 2.7.1) such as engine speed, acceleration, or torque.

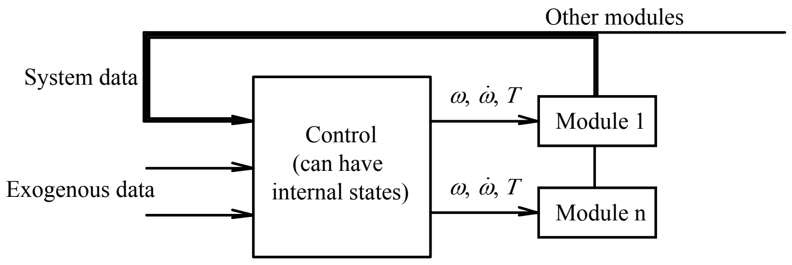


Fig. 2.7.1 Diagram of a basic control system

A control system in which input changes immediately affect the output (i.e., no storage device in-between) frequently contains implicit loops. Generally, the built-in Simulink solver is able to deal with those. However, whenever possible, it is preferable to try to avoid such loops.

Another reason why control systems sometimes are provided with a "memory" is the presence of state automata, or when the control system must be enabled to act preemptively. For instance, external data of a drive cycle must be detected and processed before they influence the vehicle.

With reference to Sec. 2.4.3 above, Fig. 2.7.2 shows an example of a control system for a CVT. The output data from a block "Driving Cycle" are provided to the "real" vehicle with a delay of one step size while they simultaneously are being processed within the control system without any delay whatsoever. The vehicle model contained in the control system computes the required drive power for any point in time. The controller determines the rotational speed at which that power is best generated by the engine, taking into account such goals as minimum consumption, running smoothness, as well as gear ratio limitations of the CVT.
A real CVT control system could look similar. The controller inputs would include such data as rotational speeds of the wheels and the engine, and the driver's torque demand as expressed by the gas pedal position. The output would include the change in gear ratio, whereas the gear ratio itself would be its integral value.

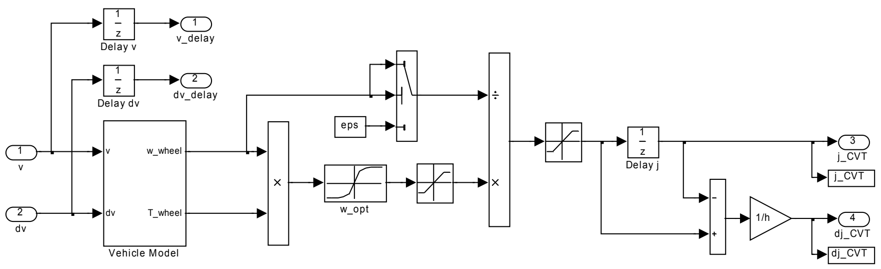


Fig. 2.7.2 Control system of a CVT with delays, vehicle model, and CVT gear ratio optimization

The engine map for the EU cycle shown in Fig. 2.7.3 takes the vehicle example from Sec. 1.3, but includes a CVT and the control system from Fig. 2.7.2. It shows two different cases: One is a CVT with a realistic gear ratio band of 1:5 (with the characteristic data encircled), whereas the other is a CVT with an unrealistic ratio of 1:7.5 (data marked with green "+" signs). The red curve designates the operating points the control system considers to be optimum (cf. block "*w_opt*" in Fig. 2.7.2 above).
The deviations of the operating points calculated from those on the optimum curve are explained as follows:

- The bandwidth is limited, which means that the optimum is not attainable for all points in the cycle. (The "+" case shows that these deviations do not occur when the spread is sufficiently large.)
- Losses within the CVT must be compensated by additional engine power.
- The calculation of the actual load included the vehicle, but none of the inertia within the powertrain or engine for which the additional accelerating power is required to overcome it.

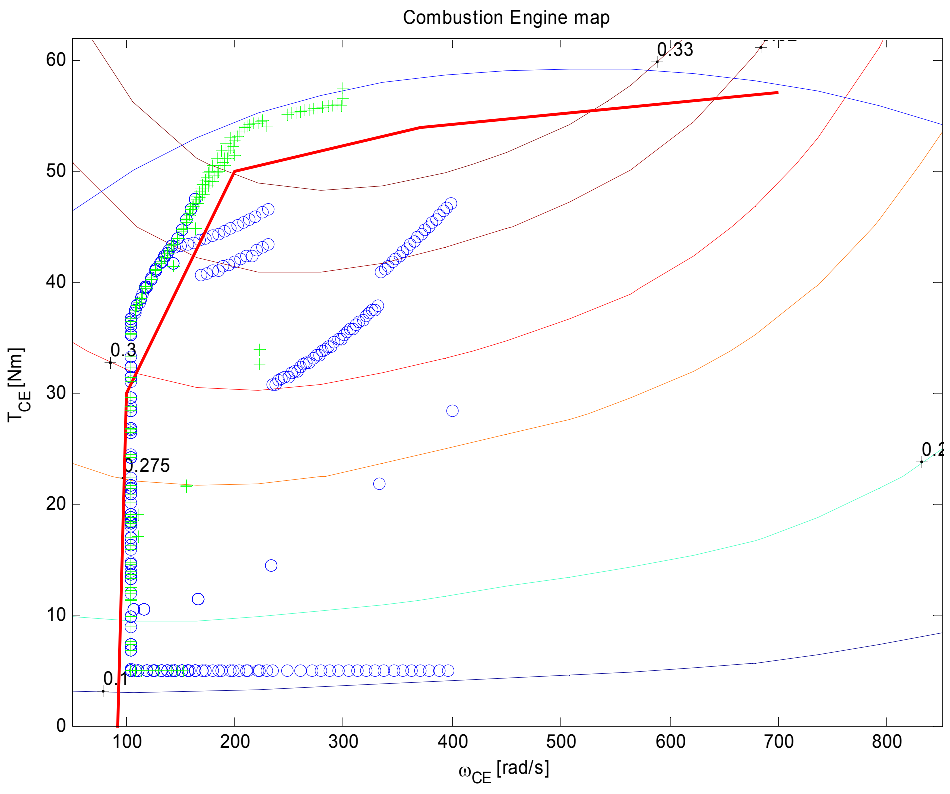
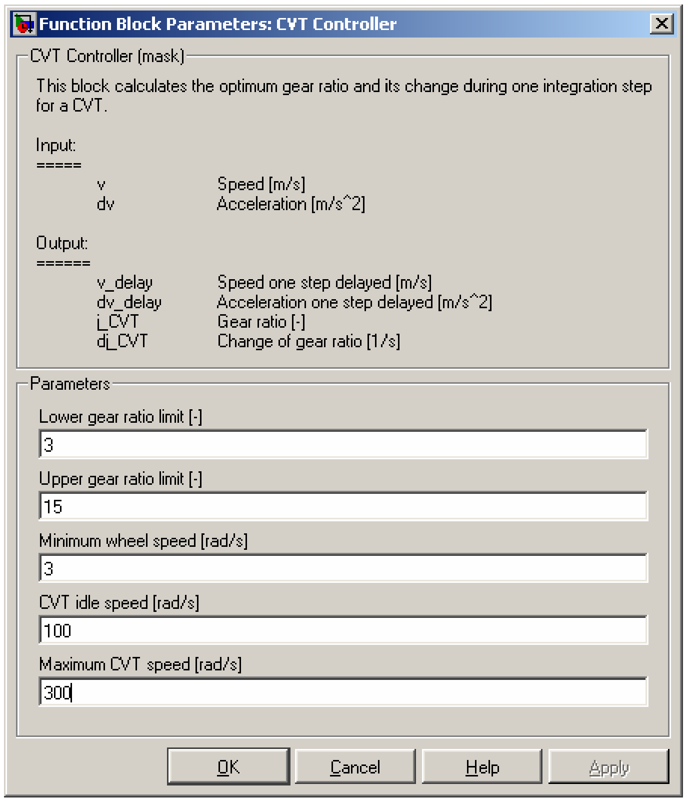


Fig. 2.7.3 Program map of a CVT drive system; CVT gear ratios: blue o = 1:5, green + = 1:7.5

The fuel consumption data for the EU cycle for the case of the gear ratio bandwidth 1:7.5 are about 8% lower than those measured at the 1:5 bandwidth, which is a noticeable improvement.



CVT Controller (mask)

This block calculates the optimum gear ratio and its change during one integration step for a CVT.

```
Input:
=====
        v               Speed [m/s]
        dv              Acceleration [m/s^2]

Output:
======
        v_delay         Speed one step delayed [m/s]
        dv_delay        Acceleration one step delayed [m/s^2]
        i_CVT           Gear ratio [-]
        di_CVT          Change of gear ratio [1/s]
```

Parameters

Lower gear ratio limit [-]	3
Upper gear ratio limit [-]	15
Minimum wheel speed [rad/s]	3
CVT idle speed [rad/s]	100
Maximum CVT speed [rad/s]	300

Fig. 2.7.4 The user interface (mask) of the "CVT Controller" block

3 Applications

3.1 Example "Series Hybrid Vehicle"

This example is based only on those elements that are included in the QSS TB. Our goal is to design a series hybrid vehicle that runs the internal combustion engine only at its good operating states. In particular, part-load losses are to be avoided[5]. As the vehicle is able to produce its own electricity (via its so-called "range extender" group), the battery capacity can be designed much smaller than it would have to be in a purely electric vehicle, which of course results in a lower battery weight.
The top level of the model is shown in Fig. 3.1.1, where the elements introduced above are evident in the arrangement suitable for the type of vehicle to be designed. The linkages of the electrical components are especially noteworthy as the sum of the three electric power systems (i.e., starter motor, generator, and battery) must always amount to zero. The power system which takes care of the necessary adjustments of the voltage level and of the control of the power flow is assumed to be ideal and is thus neglected here.

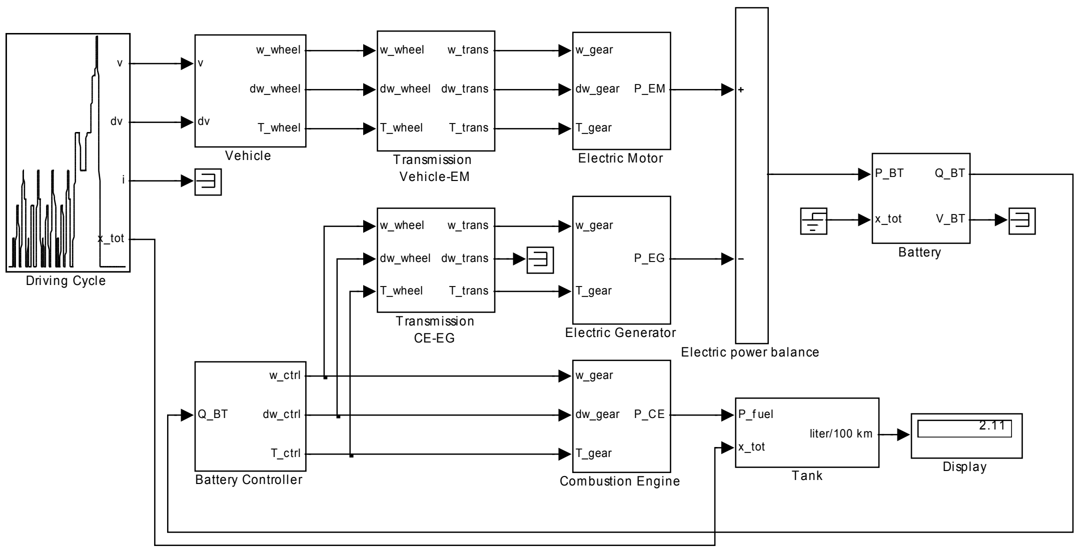


Fig. 3.1.1 Top level of the QSS TB model series hybrid vehicle

The various elements of this model are scaled-down versions of the basic elements. The following points are worth mentioning:

- The "range extender" block is designed for a minimum of 20 kW, which allows the vehicle to reach a (theoretical) top speed of approximately 140 km/h.
- The internal combustion engine is assumed to be one-half the size of the basic engine introduced in the QSS TB (i.e. two cylinders of approximately 180 cc each). Its fuel consumption program map therefore is scaled down accordingly (i.e., one-half the consumption at each point of the program map p_{me} / c_m).

[5]With a suitable design, it is possible for series hybrid vehicles to have extremely low pollution emissions; e.g. the California EZEV limits.

- The electric generator is assumed to be twice the size of the one provided with the QSS-TB. Its efficiency is assumed to remain the same.
- The electric motor is scaled up by a factor of 3.5 (in both quadrants), which yields a maximum torque level of 250 Nm and a maximum power output of approximately 37 kW. The efficiency levels remain the same as those of the basic motor.
- The electric motor is coupled to the wheel at a fixed gear ratio. The gear ratio selected of 3.5 allows a top speed of around 150 km/h. Due to its large maximum torque at small rotating speeds, at low speeds (up to around 60 km/h) the vehicle has rather good acceleration reserves.
- The batteries can deliver approximately 9 kWh and are charged to about 30% capacity at the start. If smaller batteries were to be installed, which would be possible considering the requirements of the EU cycle, the result would be unacceptably high voltages[6] during the acceleration phases at the end of the cycle.
- If we assume an energy density of 70 Wh/kg, which is a good value nowadays, a battery of around 130 kg would be required. Adding the electric power system, the two electric motors, etc. would add a net weight of 200 kg (i.e., additional weight minus the weight savings due to a smaller IC engine and drivetrain).
- The controller is rather straightforward: The main goal is to run the vehicle in such a way that at the end of an EU cycle the battery shows the same charge level as it did at the beginning (the so-called "autarky" or self-sufficiency level). The charge period is designed to be during the high-load phase near the end of the EU cycle.
- The actual battery charge level serves as input to the controller which detects any drop below a certain level. Whenever the controller is activated, it first increases the idle speed of the IC engine to the maximum charging level and then gradually adds the electric generator torque. The deactivation occurs equally gently.

Some of the results of the calculations described above are depicted in Figs. 3.1.2 through 3.1.5. As mentioned earlier, the electric motor shows rather significant torque reserves, particularly at low rotating speeds (see Fig. 3.1.2). However, the top speed cannot be raised much beyond the top of the one shown for the EU cycle. These characteristics of the vehicle can be improved by changes in the transmission ratio or, more simply, by using a two-gear gearbox system.

[6]Of course, those voltages could be attenuated by electrical "peak shavers" such as the "supercaps" described earlier.

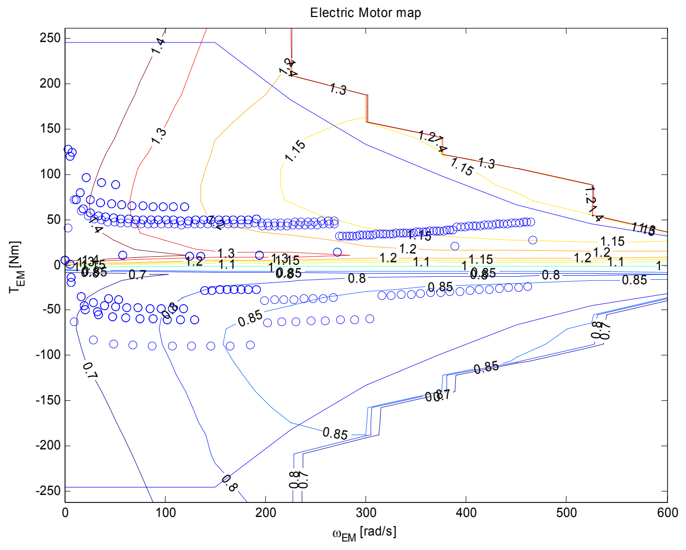


Fig. 3.1.2 Operating points of the electric motor in the EU cycle

Driving Cycle:

Parameter	Value
Choose a cycle	Europe NEDC
Step size [s]	1
Enable automatic simulation stop	ON

Vehicle:

Parameter	Value
Total mass of the vehicle [kg]	750+200
Rotating mass [%]	5
Vehicle cross section [m^2]	2.0
Wheel diameter [m]	0.5
Drag coefficient [-]	0.22
Rolling friction coefficient [-]	0.008

Transmission Vehicle-EM:

Parameter	Value
Differential gear [-]	3.5
Efficiency [-]	0.98
Idling losses (friction) [W]	300
Minimum wheel speed [rad/s]	1

Transmission CE-EG:

Parameter	Value
Gear ratio [-]	0.75
Efficiency [-]	0.98
Idling losses (friction) [W]	200
Minimum wheel speed [rad/s]	1

Electric Motor:

Parameter	**Value**
Motor scaling factor [-]	3.5
Motor inertia [kg*m^2]	0.1
Auxiliaries power [W]	0

Electric Generator:

Parameter	**Value**
Generator scaling factor [-]	2

Combustion Engine:

Parameter	**Value**
Engine type	Otto
Displacement [l]	0.708
Engine scaling factor [-]	0.5
Engine inertia [kg*m^2]	0.05
Engine speed at idle [rad/s]	105
Engine power at idle [W]	0
Power required by auxiliaries [W]	0
Enable fuel cutoff	OFF
Engine torque at fuel cutoff [Nm]	0
Engine power at fuel cutoff [W]	0

Battery:

Parameter	**Value**
Energy capacity of battery [Ah]	100
Initial charge of battery [%]	27.78
Minimum time to charge/discharge [min]	20

Tank:

Parameter	**Value**
Fuel	Gasoline
Include cold start losses	OFF

The battery variables during the EU cycle are shown in Fig. 3.1.3 a)–d) below. Since the latter part of the EU cycle is very demanding, Fig. 3.1.3.b) clearly shows the drop in battery voltage.

Fig. 3.1.4 shows the operating points of the generator from its start at idle (i.e., without torque or current output) and then charged. The operating points are all found to be near its peak efficiency of at least 85%.

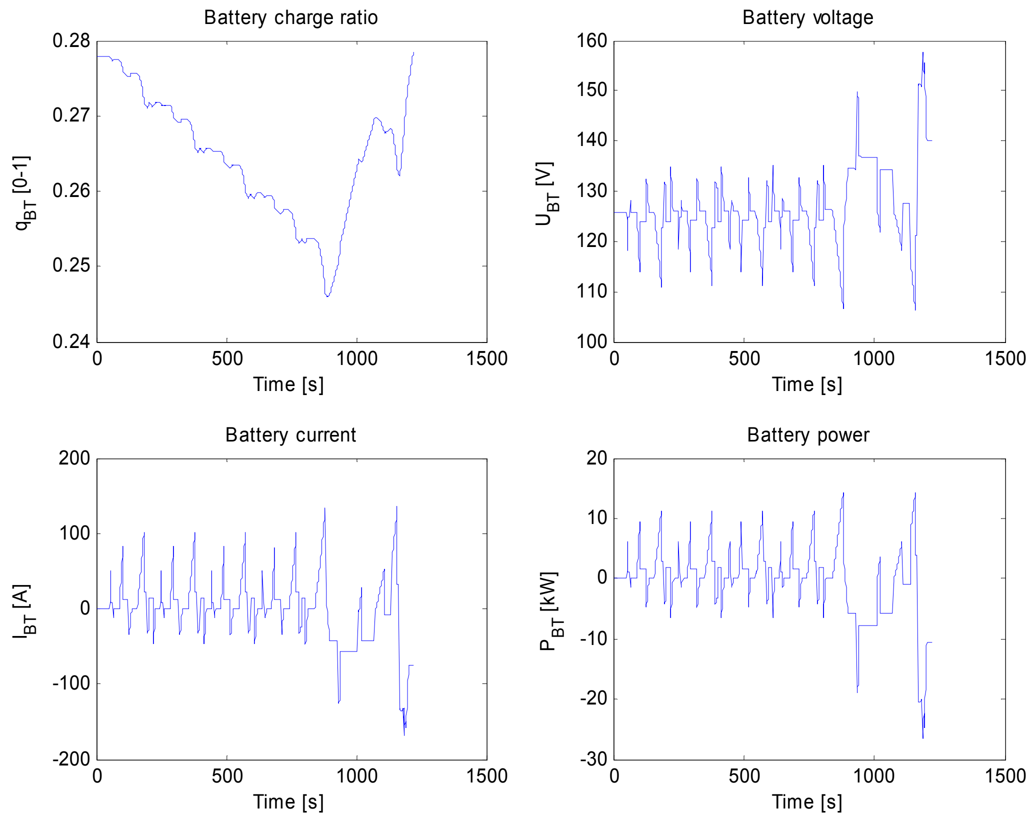


Fig. 3.1.3 a) – d) Battery: a) charge ratio, b) voltage, c) current and d) power

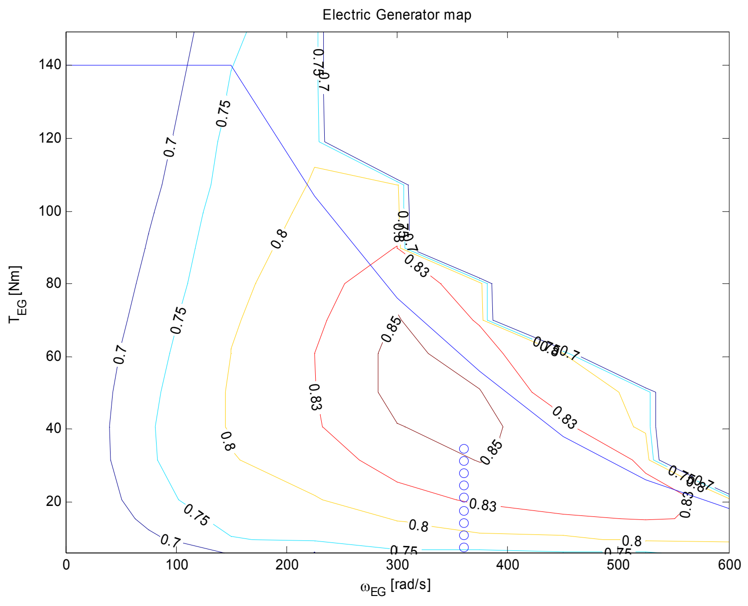


Fig. 3.1.4 Operating points of the generator

The efficiency of an IC engine during the charging phase is shown as well. During acceleration, it runs up only its inertia and that of the electric generator, but even those few peaks are within an acceptable efficiency level of about 20%.

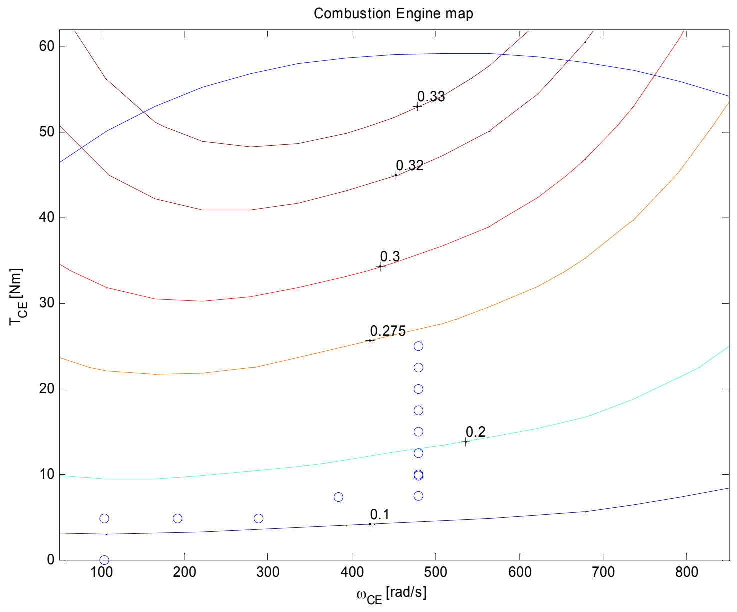


Fig. 3.1.5 Operating points of the internal combustion engine

A comparison between Figs. 3.1.4 and 3.1.5 clearly shows the torque reserves of the generator as opposed to a nearly maximum torque of the IC engine when both are run at the maximum output of slightly over 20 kW at the rotating speed selected. If a higher output level of the range extender group was desired, a different transmission ratio would have to be used which in turn would not be tolerable for the batteries chosen. In real situations demanding higher outputs flowing from the generator to the electric motor, such as in climbing for extended times, either a gear box would have to be included in the "range extender group", or the output of the IC engine would have to be throttled.

3.2 Adding new elements

New elements can be added easily if a few points are considered along the way. Using its inputs any new element must recalculate its outputs once every cycle. As Simulink does not show the sequence in which it recalculates the various blocks, it is important to ensure that these blocks do not contain any implicit conditions whenever these outputs are recalculated.

If any "dynamic" elements are to be considered, such as those necessary for integrators, they should be included in time-discrete "delay" blocks only. Within those blocks, their step size must be set to h , and they may not be termed differently.

Implicit functions are allowed, but they usually cause noticeably longer computation times. Besides, the Simulink solver may have trouble solving implicit equations due to the fact that they can represent discontinuities within the model equations. It is thus preferable to solve any implicit equations prior to adding those elements. If that cannot be done, the inclusion of a "delay" block may break up the implicit loop (cf. the description of the battery model in Sec. 2.5.1 above).

Besides the fact that the general rules for Simulink programming and generally good programming techniques should be followed, there is nothing new required for the addition of new elements.

3.3 Embedding of QSS-TB programs in Matlab

One of the main strengths of the QSS TB is the ease with which all of the models developed can be included in other Matlab programs, thus allowing to take advantage of the full range of functions that the multitude of Matlab toolboxes offers. One example of such an inclusion in the Optimization Toolbox is shown below (Sec. 3.4).

The "*sim*" instruction is the key to the inclusion of any QSS TB program. The model to be simulated may be called up simply with the instruction:

```
>> sim('model_name');
```

where the single quote signs around the name of the model from the QSS TB to be simulated are a Matlab rule. Simulink then refers to the model parameters defined within the Matlab workspace and to the initial conditions and the simulation parameters (e.g., accuracy demanded, time to end, etc.) defined within the QSS TB model to run the computation of the model. A number of control parameters as well as the desired output may be changed at the time the model is called up since the instruction "*sim*" is a "regular" Simulink instruction. The instructions "*help sim*" and "*help simset*" provide further details.

If the transmission of the results from Simulink to Matlab does not run smoothly, this may be due to the distinctions Matlab makes among the areas where its variables are saved. There are two ways to avoid this problem: Either the "*sim*" instruction is written to include specific instructions for the output, or so-called "global" variables are introduced with the instruction "*global*".

Calling up the QSS TB is comparable to calling up any other Matlab function, except of course that the CPU time of an entire cycle simulation is longer than that of a simple function. In any case, the full range of possibilities Matlab offers in signal processing, the visualization of data, etc. can be used with this toolbox.

3.4 Example "Optimal Transmission Design"

The following optimization problem is to serve as an example for the embedding of a QSS program in a Matlab routine:

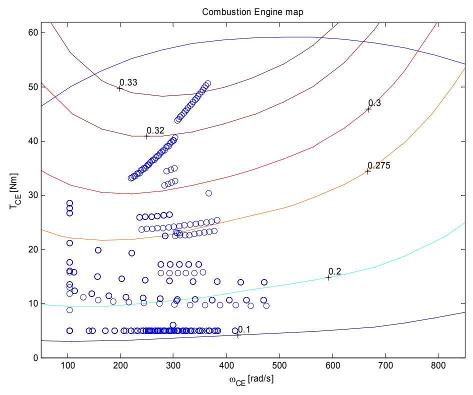
How should a standard (manual) transmission be designed for optimum fuel consumption in the EU cycle?

Given that the vehicle under consideration must be able to drive the EU cycle, we want to find the gear ratios that lead to the lowest fuel consumption possible under the conditions defined by the EU legislation (the EU cycle prescribes which gear is to be engaged for all times in the cycle). Obviously, a more useful problem would consider as additional parameters the time instances for the gear changes, etc., but it would also be too complex for this brief example.
For this example, the vehicle, the engine, and all the other elements except for the gearbox system are those considered in Sec. 1.3 above and in the lecture "Vehicle Engine Systems". Table 3.4.1 shows the initial and the optimized gear ratios that resulted from these calculations.

Table 3.4.1 Gear ratios – original and optimized values

Original values	Optimized values
$i_1 = 15.7140$	$i_1 = 15.534$
$i_2 = 8.3380$	$i_2 = 10.597$
$i_3 = 5.3780$	$i_3 = 2.7029$
$i_4 = 3.9370$	$i_4 = 2.5746$
$i_5 = 2.7480$	$i_5 = 2.4089$

The resulting fuel savings are approximately 0.323 liters per 100 km which, based on the initial consumption values of 3.457 l/100 km, indicate an improvement of slightly over 9%. As expected, the concomitant changes in torque and rotating speed are shown in Fig. 3.4.1, i.e. torque increases, while speed decreases somewhat. The drivability of a vehicle with these values would not likely to be acceptable since it would have a relatively weak acceleration performance.



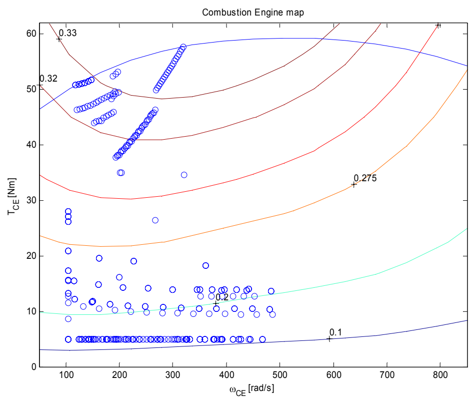


Fig. 3.4.1 Distribution of the operating points compared – a) original powertrain system b) optimized version

The primary purpose of this optimization is to include the Simulink model, i.e., the QSS TB based thereon which is used to compute the fuel consumption, as a subroutine in an existing optimization routine to compute its optimization criterion.
This is a fully numerical optimization, which means that the gradients of the optimization criterion are not available in a closed form. Instead, the optimization algorithm itself must numerically approximate the gradients of the optimization criterion. For this example, the routine "*fminsearch*" from the Optimization toolbox is used. Table 3.4.2 shows a relevant excerpt from its User's Guide.

Table 3.4.2 Excerpt from the help text concerning the optimization routine used

```
FMINSEARCH Multidimensional unconstrained nonlinear minimization (Nelder-Mead).
    X = FMINSEARCH(FUN,X0) starts at X0 and attempts to find a local minimizer
    X of the function FUN. FUN accepts input X and returns a scalar function
    value F evaluated at X. X0 can be a scalar, vector or matrix.

    X = FMINSEARCH(FUN,X0,OPTIONS)  minimizes with the default optimization
    parameters replaced by values in the structure OPTIONS, created
    with the OPTIMSET function.  See OPTIMSET for details.  FMINSEARCH uses
    these options: Display, TolX, TolFun, MaxFunEvals, MaxIter, FunValCheck,
    and OutputFcn.
```

To call "*fminsearch*" thus needs two things:

- the name of the .m file containing the computation of the optimization criterion, e.g., "*OptiGear*"
- an estimate of the optimum gear ratios, termed "*i_guess*", for instance, with:

```
i_guess = [i_1g; i_2g; i_3g; i_4g; i_5g;]
```

(where "*i_1g*" … "*i_5g*" are the guessed values of the gear ratios that have to be entered by the user prior to starting the optimization) and:

```
i_opt = fminsearch('OptiGear',i_guess).
```

Table 3.4.3 shows the computation of the optimization criterion within "*OptigGear*".
The routine "*OptiGear*" receives the actual value for the gear ratio in the vector "*gear_result*".
Any additional parameters that are subject to being treated or changed must be predefined as "*global*" (a procedure copied from the "common" structure of the FORTRAN programming language).

The computation itself, i.e., the quasistatic simulation of the vehicle model, is started by the instruction "*sim*". In the simplest case in which all starting conditions, end time, etc. are given within the QSS program, "*sim*" only requires the name of the Simulink file to be run.
The output of the subroutine "*OptiGear*" represents the total fuel consumption of the cycle.

In case the cycle is not computed all the way through, i.e., the actual number of computation steps required to complete the cycle is less than the given value "*N_sim*", the computation yields a high consumption value by design to ensure that the constraints are duly considered. However, this is possible only if the Simulink program proper also contains the mechanisms necessary to recognize overload and overspeed situations and to react with a simulation abort, if necessary. The QSS TB elements of course do fulfill these conditions.

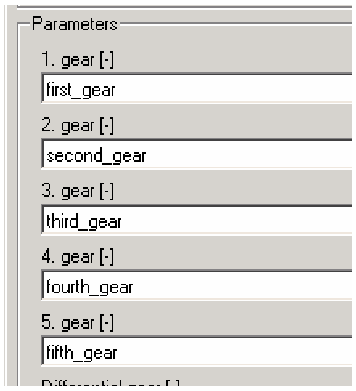
Additionally, a plausibility check to ensure that the computed gear ratios are "sorted" (in the sense that the inequality "*first gear > second gear > third gear > fourth gear > fifth gear"* holds) is done; if the above mentioned inequality is not satisfied, by design the computation yields too high a consumption value.

Table 3.4.3 Computation of the optimization criterion for the numerical optimization

```
function V_result = OptiGear(x);

global t
global N_sim
global w_CE T_CE
global gear_result
global V_liter
global first_gear second_gear third_gear fourth_gear fifth_gear

% Set actual gear ratios
% ----------------------
    first_gear      = x(1);
    second_gear     = x(2);
    third_gear      = x(3);
    fourth_gear     = x(4);
    fifth_gear      = x(5);

% Build matrix of gear ratio values
% ---------------------------------
    gear_result = [gear_result; x'];

% Do simulation
% -------------
    sim('qss_example_optigear');

% Consider last value of the computed fuel consumption vector
% -----------------------------------------------------------
    V_result = V_liter(max(size(t)));

% Check whether cycle could be finished exactly in N_sim computational
% steps;
% if cycle duration is less than N_sim, set fuel consumption to infinite
% ----------------------------------------------------------------------
    if (max(size(t)) < N_sim)
        V_result = Inf;
    end

% Plausibility check of the computed gear ratios
% ----------------------------------------------
    if (first_gear > second_gear) && (second_gear > third_gear) &&
(third_gear > fourth_gear) && (fourth_gear > fifth_gear)
        V_result = V_result;
    else
        V_result = Inf;
    end
```

In order to allow the optimization routine to "communicate" with the .mdl file and to run the simulation with the actual computed gear ratio values, the mask of the "Manual Gear Box" block (see Fig. 2.4.6) has to be slightly modified by entering the variable names as depicted in Fig. 3.4.2.

The variables "*first gear*" … "*fifth gear*" are computed at each computation step by the optimization routine (see Table 3.4.3) and then, in accordance with the modifications of Fig. 3.4.2, "sent" to the .mdl file in order to run the simulation.



Parameters

1. gear [-]

first_gear

2. gear [-]

second_gear

3. gear [-]

third_gear

4. gear [-]

fourth_gear

5. gear [-]

fifth_gear

Fig. 3.4.2 The modified entries for the mask of the "Manual Gear Box" block

The various gear ratios computed during the course of the optimization routine are shown in Fig. 3.4.3. The optimization process reaches a static state after approximately 900 iterations; the concomitant fuel consumption value of around 3.134 l/100 km should be near a local minimum as well.

One of the most interesting results of this simulation is that for the EU cycle a power train system requires no more than three gears. The third, fourth, and fifth gears are so close together that their combination into a single one does not significantly increase the consumption value: in fact, if the value for third gear is used for the fourth and fifth gears as well, the resulting consumption value for the whole cycle is slightly higher than the result obtained from the optimization algorithm (i.e., 3.134 vs. 3.171 l/100 km).

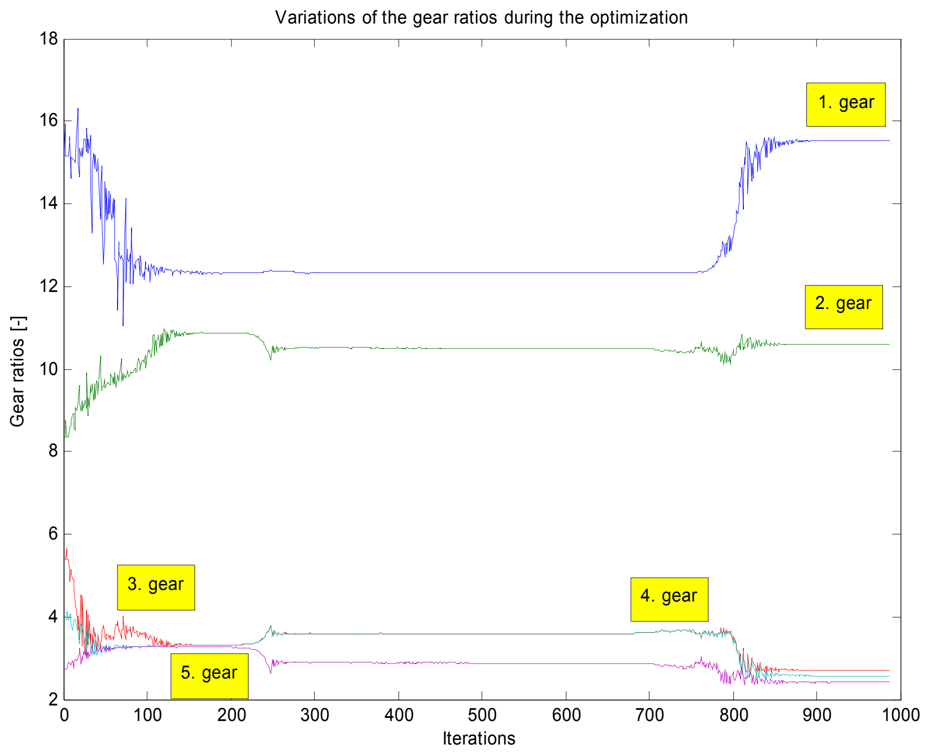


Fig. 3.4.3 The various gear ratios computed during the course of the optimization routine

4 Customizing the QSS Toolbox Library

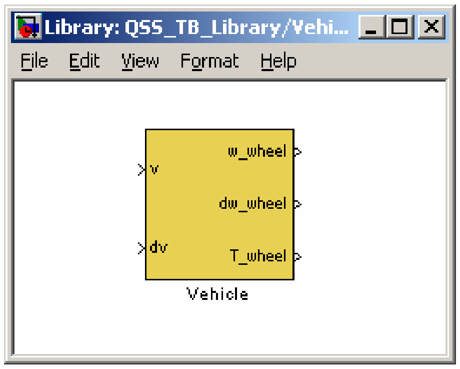
The QSS toolbox is a library of Matlab/Simulink block diagrams, analogous to the well known “Simulink Library” included in each Matlab release. The user can imagine each element of a library like a pre-defined object: to build a model it is simply necessary to drag the desired elements from the library to the (blank) .mdl file. Then by double-clicking on each element it is possible to enter the desired corresponding parameter values.

By default a library is locked, i.e. its elements cannot be customized without unlocking the library. Therefore, before starting to customize the QSS TB library it is necessary to unlock it:

1. Open the QSS toolbox library
2. Click on “Edit” in the menu bar
3. Click on “Unlock Library”

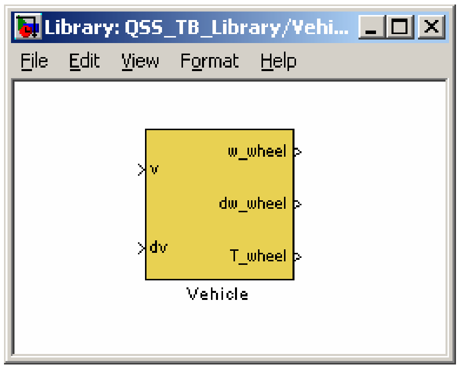
Now the user is ready to effect the desired changes within the QSS TB library. As an example, the “Vehicle” block is presented below.

Customizing the Simulink block diagram
Double-click on the “Vehicle” block in the QSS TB library. The following window appears:

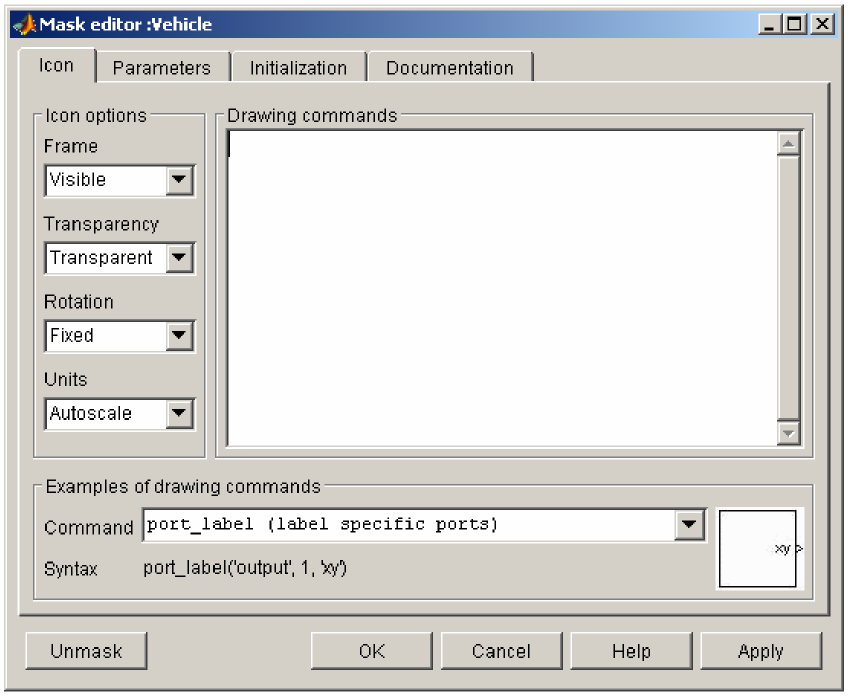


Next, right-click on the block and choose “Look Under Mask”. The complete block diagram of the “Vehicle” block (see Fig. 2.2.1) will appear.

Customizing the Mask
Double-click on the "Vehicle" block in the QSS TB library. The following window appears:

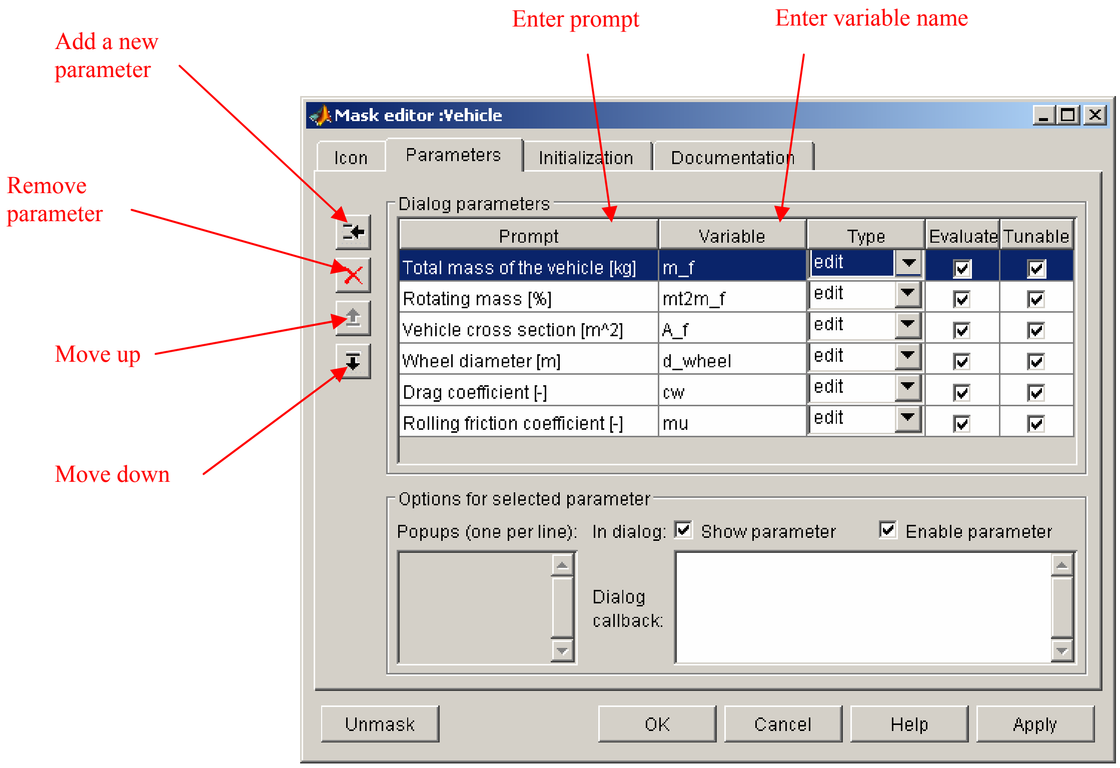


Next, right-click on the block and choose "Edit Mask". The Mask editor is started as follows:



Frame: Visible
Transparency: Transparent
Rotation: Fixed
Units: Autoscale
Command: port_label (label specific ports)
Syntax: port_label('output', 1, 'xy')

First, the procedure to change the model parameters is shown. By clicking on the layer "Parameters" a user interface to add, remove, or rename the parameters of the block will appear:



Dialog parameters

Prompt	Variable	Type	Evaluate	Tunable
Total mass of the vehicle [kg]	m_f	edit	✓	✓
Rotating mass [%]	mt2m_f	edit	✓	✓
Vehicle cross section [m^2]	A_f	edit	✓	✓
Wheel diameter [m]	d_wheel	edit	✓	✓
Drag coefficient [-]	cw	edit	✓	✓
Rolling friction coefficient [-]	mu	edit	✓	✓

Options for selected parameter

Popups (one per line):

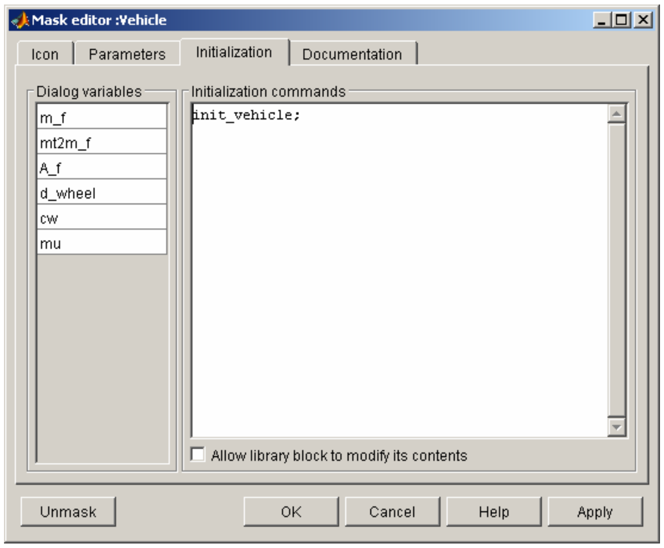
In dialog: ✓ Show parameter ✓ Enable parameter

Dialog callback:

Obviously, the chosen variable names should be the same as in the Simulink block diagram!

Next, by clicking on the layer "Initialization" it is possible to enter the commands that the "Vehicle" block needs prior to starting the complete simulation. In this case, all commands are saved in the m-file "*init_vehicle*". The file "*init_vehicle*" is located in the folder "*\Initialize*" as well as the others "*init*"-files needed by all others QSS-TB blocks.

The file "*init_vehicle*" (as well as the others "*init*"-files) must be included in the Matlab path!



m_f
mt2m_f
A_f
d_wheel
cw
mu

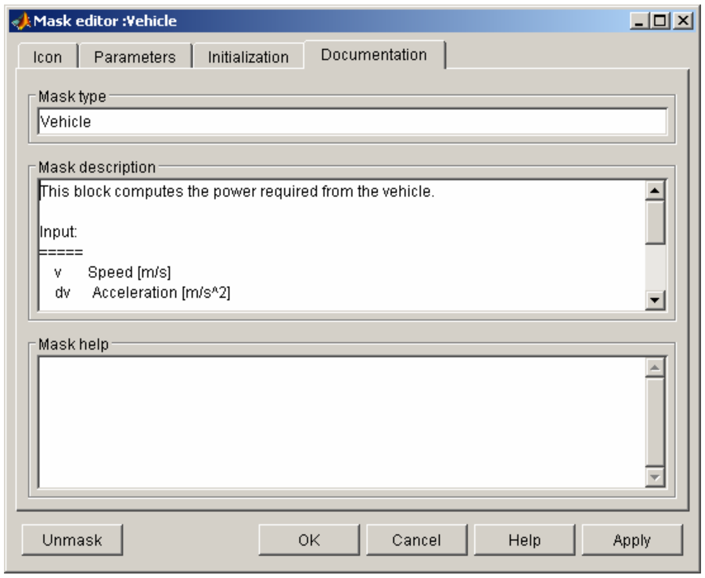
```
init_vehicle;
```

☐ Allow library block to modify its contents

For example, when using the block “Vehicle” the user has to enter the value for the parameter “*Wheel diameter [m]*” (*d_wheel*). The QSS TB, however, needs the wheel radius to perform its computations. Therefore in the file “*init_vehicle*” one can find following code line:

```
r_wheel = d_wheel / 2;
```

Finally, the user can change the name of the mask and write his/her own mask description. This is done by clicking on the layer “Documentation” and entering the desired entries in the corresponding spaces.



Vehicle

```
This block computes the power required from the vehicle.

Input:
=====
  v     Speed [m/s]
  dv    Acceleration [m/s^2]
```

Additionally the user can write his/her own html help pages, for example to give other users further information about the new customized version of the QSS TB library block “Vehicle”.

To create a reference between the customized html help file “*MyHelpFile.htm*” located at the Internet site “*www.MyWebSite.com*” and the mask, simply write the following command in the “Mask help”:

```
eval('web http://www.MyWebSite.com/MyHelpFile.htm');
```

5 References

[1] Guzzella L., Onder C. H.:
Introduction to Modeling and Control of Internal Combustion Engine Systems,
Springer Verlag, Berlin, 2004