

ABS

**The laboratory
Anti-lock Braking System**

User's Manual



inteco@inteco.com.pl

CONTENTS

- 1. INTRODUCTION AND GENERAL DESCRIPTION 3**
 - 1.1. Product overview 4
 - 1.2. Requirements 4
- 2. INSTALLATION OF THE SOFTWARE..... 5**
- 3. STARTING, TESTING AND STOPPING PROCEDURES..... 5**
 - 3.1. Starting procedure 5
 - 3.2. Testing..... 6
 - 3.3. Stopping procedure 7
- 4. ABS CONTROL WINDOW 8**
 - 4.1. Tools..... 8
 - 4.2. RTWT Device Driver 13
 - 4.3. Simulation model 14
 - 4.4. Demo Controller 15
- 5. PROTOTYPING YOUR OWN CONTROLLER IN THE RTWT ENVIRONMENT 21**
 - 5.1. Creating a model 22
 - 5.2. Code generation and the build process 23
- 6. MATHEMATICAL MODEL..... 26**
 - 6.1. Equations of motion 28
 - 6.2. Identification..... 30
 - 6.2.1. Geometrical issues 30
 - 6.2.2. Deriving of the torque M_g 30
 - 6.2.3. Derivations of the moments of inertia..... 31
 - 6.2.4. Deriving the friction coefficients in the bearings 32
 - 6.2.5. Identification of the friction coefficient and the braking torque..... 33
- 7. SIMULATION OF BRAKING 36**

1. Introduction and general description

The anti-lock braking system (ABS) in cars were implemented in late 70's. The main objective of the control system is to prevent of wheel-lock while braking. Usually we are interested in the tire slip on each of the four wheels in the car. Only longitudinal motion is considered. Nevertheless differences between the front and the back wheels and between the right and the left wheels, we consider only a simplified model of the quarter of a car.

ABS are designed to optimize braking effectiveness while maintaining car controllability. The performance of ABS can be demonstrated in our lab-set by simulation for various road condition (wet asphalt) and transition between such conditions (e.g., when emergency braking occurs and the road switches from dry to wet or vice versa).

ABS is driven by the powerful flat DC motor. There are two identical encoders measuring the rotational angles of two wheels and the deviation angle of the balance lever of the car wheel. The encoders measure movements with a high resolution equal to 4096 pulses per revolution.

The power interface amplifies the control signals which are transmitted from the PC to the DC motor. It also converts the encoders pulse signals to the digital 16-bit form to be read by the PC. The PC equipped with the RT-DAC/PCI-D multipurpose digital I/O board communicates with the power interface. The whole logic necessary to activate and read the encoder signals and to generate the appropriate sequence of pulses of PWM to control the DC motor is configured in the Xilinx[®] chip of the RT-DAC/PCI-D board. All functions of the board are accessed from the ABS Toolbox which operates directly in the MATLAB[®]/Simulink[®] and the RTWT toolbox environment.

At the beginning of an experiment the wheel animating the relative car-road motion is accelerated to a given threshold velocity. The wheel accelerates following the rotational motion of the "car-road" wheel. Next, the DC drive is switched off to enable free motion of both wheels.

In fact, there are two threshold velocities during the acceleration of the wheels. The first threshold velocity is introduced to the power interface to prevent a large starting current value. At the beginning the DC motor is supplied in series by a starting resistor from the power interface. If the first threshold velocity of the motor is reached then the starting resistor is switched off and the DC motor is supplied directly from the power interface. If the second threshold velocity (defined in the program and of course with the higher value than the first one) is reached then the power interface is switched off and the braking phase begins.

To understand the braking process one has to know relations between the vehicle tire and road during braking. The objective of ABS is to control wheel slip to maximize the coefficient of friction between the tire and road for any given road surface while the car is controllable.

If this wheel becomes motionless it means that it remains in slip motion (the car velocity is not equal to zero) or is absolutely stopped (the car velocity is equal to zero). In the first case the ABS algorithm has to unlocked the wheel to stop its slipping. The wheel starts to rotate and after a short time period it is stopped again. This process is repeated as long as the car velocity achieves zero value (the wheel animating the road motion related to car is stopped). If the braking time period is too long the wheel remains locked in slip motion and the car starts an uncontrollable motion. If this period is too short the wheel remains too long in the rolling stage and the brake distance is also enlarged.

1.1. Product overview

ABS is delivered in the mounted form. The mounting frame makes a support of the system. It is ready to operate after installation of its software.

KEY FEATURES

- Double-wheel laboratory model of ABS equipped with powerful flat DC motor.
- Two high-resolution measuring encoders.
- Full integration with MATLAB®/ Simulink®. Operation in real-time in Windows® XP/W7 32bit.
- Complete model of the car-road relations.
- Library of pre-programmed braking control algorithms familiarize the user with ABS technique in a fast way.
- Rapid prototyping of real-time control algorithms. No C-code programming.
- Ideal illustration of nonlinear control algorithms.

KEY BENEFITS

- Enables to provide laboratory tests of the antilock braking system in the car velocity range from 0 to 50 km/h.
- Accelerates implementation of new slip control algorithms.
- Demonstrates the slip control under different road conditions.

SETUP COMPONENTS

Hardware

- mechanical unit (frame, double wheel and DC flat motor),
- interface and Power Interface Unit
- I/O RT-DAC/PCI board (the PWM control logic is stored in a XILINX chip)

Software

- ABS Control/Simulation Toolbox operating in MATLAB®/Simulink® environment

Manuals

- *Installation Manual*
- *User's Manual*

1.2. Requirements

- Pentium or AMD based personal computer
- Microsoft Windows XP/W7 32bit
- MATLAB version R2008a/b, R2009a/b, R2010a/b or R2011a with Simulink, RTW and RTWT toolboxes (not included),

2. Installation of the software

Put the installation disc into your computer and follow the installation procedure step by step (see *Installation Manual*).

3. Starting, testing and stopping procedures

➔ The experiments and corresponding to them measurements have been conducted by the use of the standard INTECO system. Every new system manufactured and developed by INTECO can be slightly different to the standard. It explains why a user can obtain results that are not identical to these given in the manual.

3.1. Starting procedure

Combine the acquisition board, Power Interface and ABS together. In the WINDOWS environment invoke MATLAB by double clicking on the MATLAB icon. The MATLAB command window opens. Then simply type:

abs_main

MATLAB brings up the ABS *ABS Control Window* (see Fig. 3.1). Pushbuttons indicate an action that executes callback routines when the user selects item.

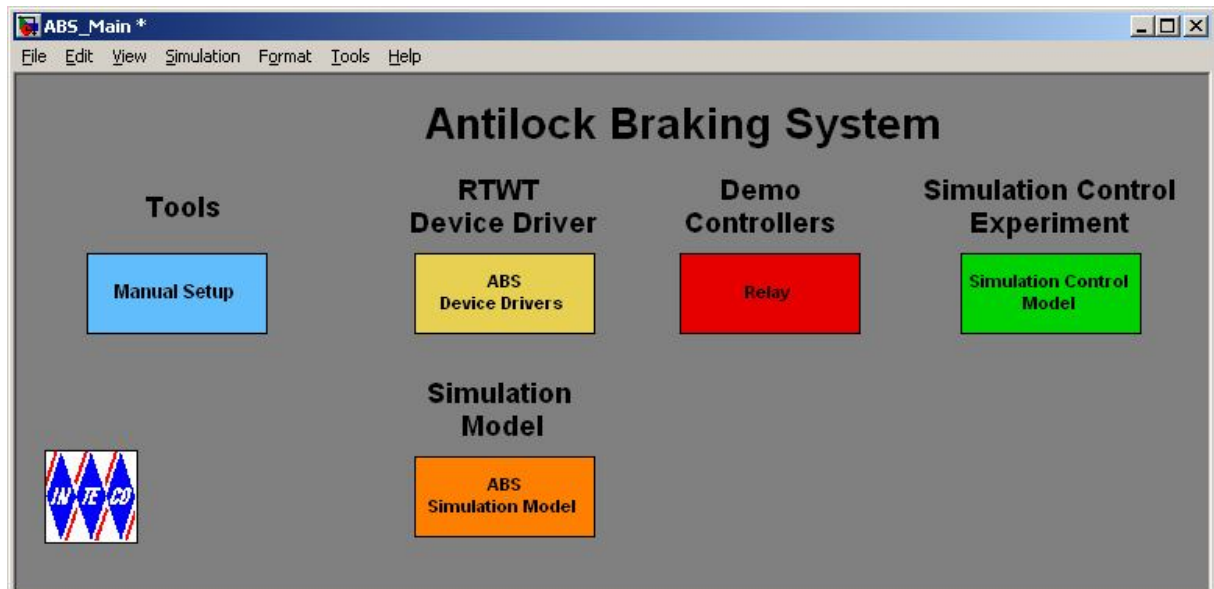


Fig. 3.1 The *ABS Control Window*

ABS Control Window contains testing tools, drivers, models and demo applications. You can see a number of pushbuttons ready to use.

3.2. Testing

Double click the *Manual Setup* button to check connections of all components of the system. The screen given in Fig. 3.2 opens. In the left upper corner box there are two text information fields: *Base Address* and *Logic Version*. Non zero data in these fields indicates that the RT-DAC/PCI-D board has been correctly detected by the system. We can notice that the measurement data in this window are equal to zero.

Now the following test is recommended. Change slightly the position of the *Acceleration* slider. The DC motor turns around. Click *Stop* button. You can notice that new measurement data have appeared (see Fig. 3.3). If the new data are not visible it means that the system is assembled in a wrong way. In such a case please come back to the *Installation Manual* and check the connections again.

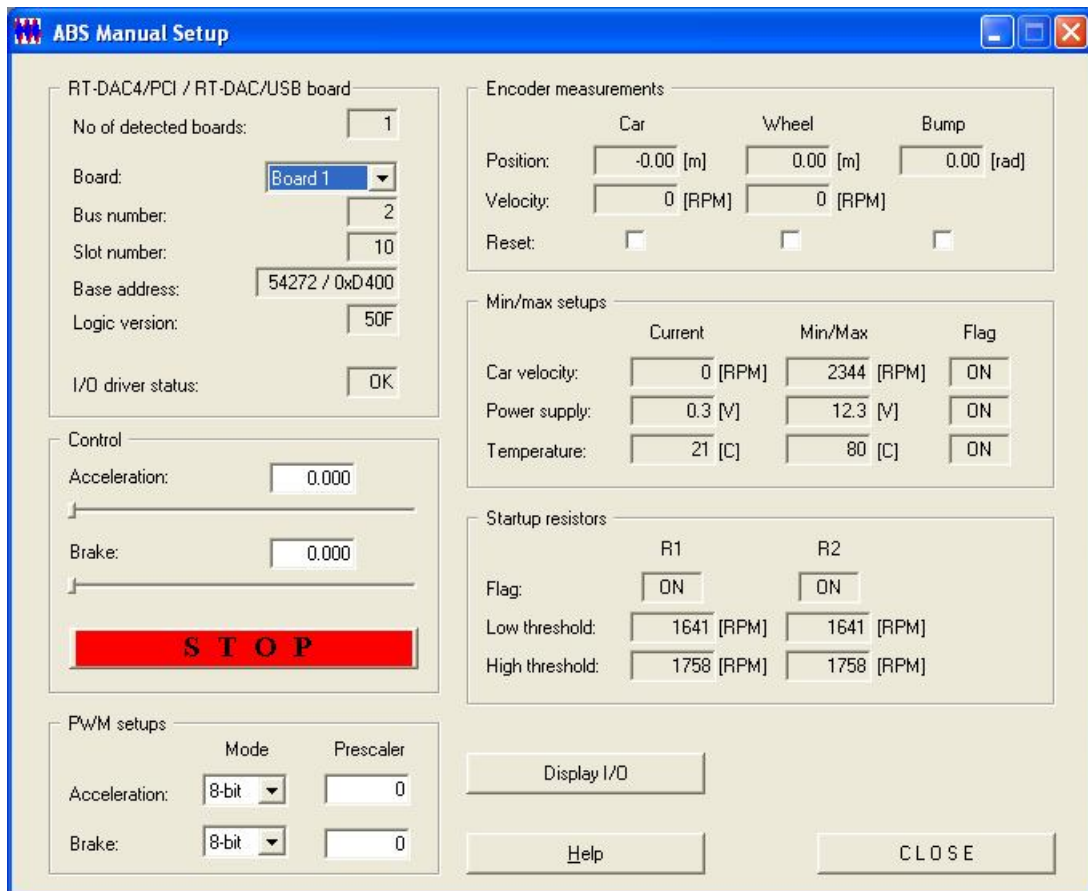


Fig. 3.2 Primary window in the testing procedure

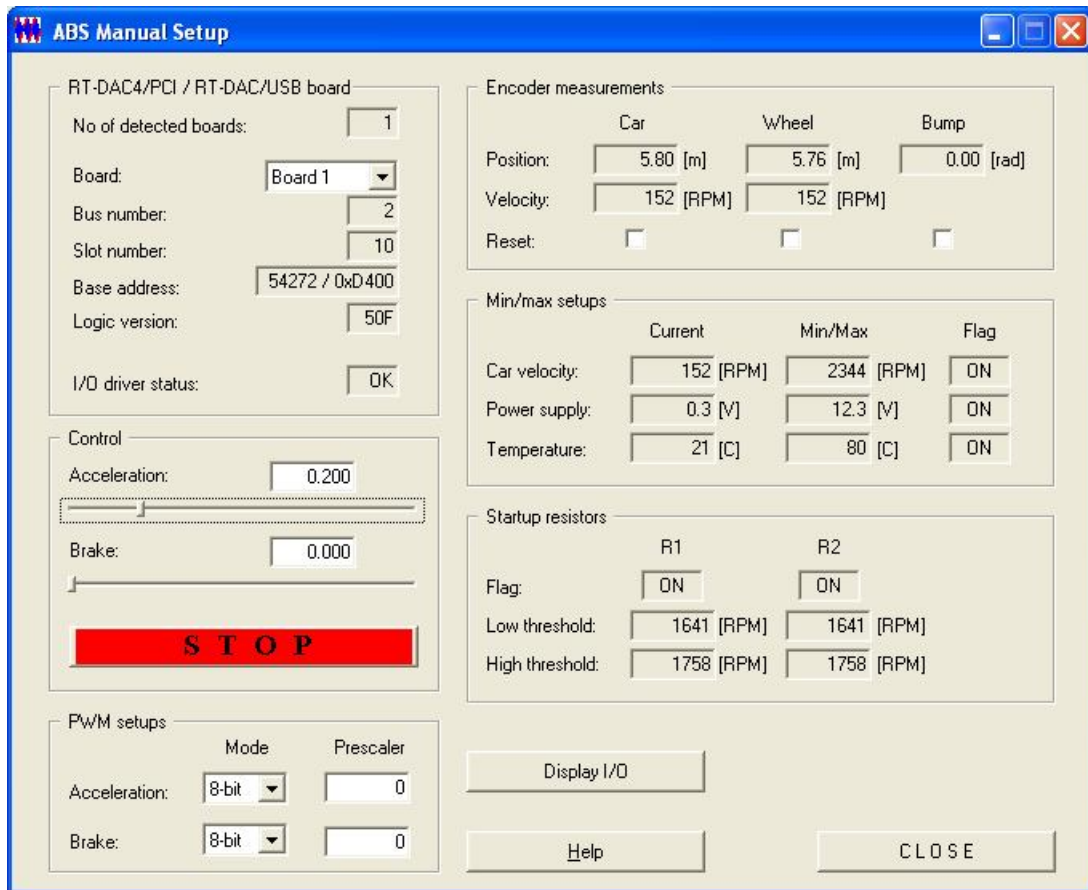


Fig. 3.3 Secondary window in the testing procedure

➡ **Remember *Car* in Fig. 3.2 and Fig. 3.3 denotes the lower wheel of the system. *Wheel* denotes the upper wheel.**

3.3. Stopping procedure

The system is equipped with the hardware emergency stop pushbutton. It cuts off the control transfer signal to the system drive. The pushbutton does not terminate the real-time process running in the background. Therefore to stop the task you have to use *Simulation/Stop* from the pull-down menus in the Simulink model window.

4. ABS Control Window

The user has a rapid access to all basic functions of ABS from the *ABS Control Window*. It includes tools, drivers, models and application examples.

The *ABS Control Window* shown in Fig. 3.1 contains:

- Tools
- RTWT Device Driver
- Simulation model
- Demo Controllers
- Simulation Control Experiment

4.1. Tools

ABS Manual Setup

The *ABS Manual Setup* program gives access to the basic parameters of the laboratory Anti-Lock Braking System setup. The program is used to watch: data transferred from the RT-DAC board and signals measured at the ABS system. Besides the status signals and flags are shown. Moreover the control signals: the acceleration and braking DC drives controls may be set.

Double click the *Manual Setup* button and the screen given in Fig. 4.1 opens.

The application contains six frames:

- RT-DAC/PCI board
- Control
- PWM setups
- Encoder measurements
- Min/max setups
- Startup resistors

The *RT-DAC/PCI / RT-DAC/USB* board frame shows the main parameters of the PCI board. In the *Control* frame the new values of the control signals are set. The *PWM setups* defines the mode and the frequency of the PWM control waves. One can read the positions and velocities of the car disk and the wheel disk and the angular positions of the bumper at the *Encoder measurements* frame. One can also reset the encoders. The *Min/max setups* frame gives the current value of the car velocity, the power supply voltage and the power supply temperature, the maximum value of the car velocity, the minimum value of the power supply voltage and the maximum value of the power interface temperature. If the current values exceed the minimum or maximum setups the respective flags are set. The *Startup resistors* frame shows the flags and threshold levels of the startup resistors.

The *Display I/O* pushbutton activates the window that displays the contents of all RT-DAC/PCI input registers. This option is not active for the RT-DAC/USB interface. All given in the *ABS Manual Setup* program are updated 10 times per second.

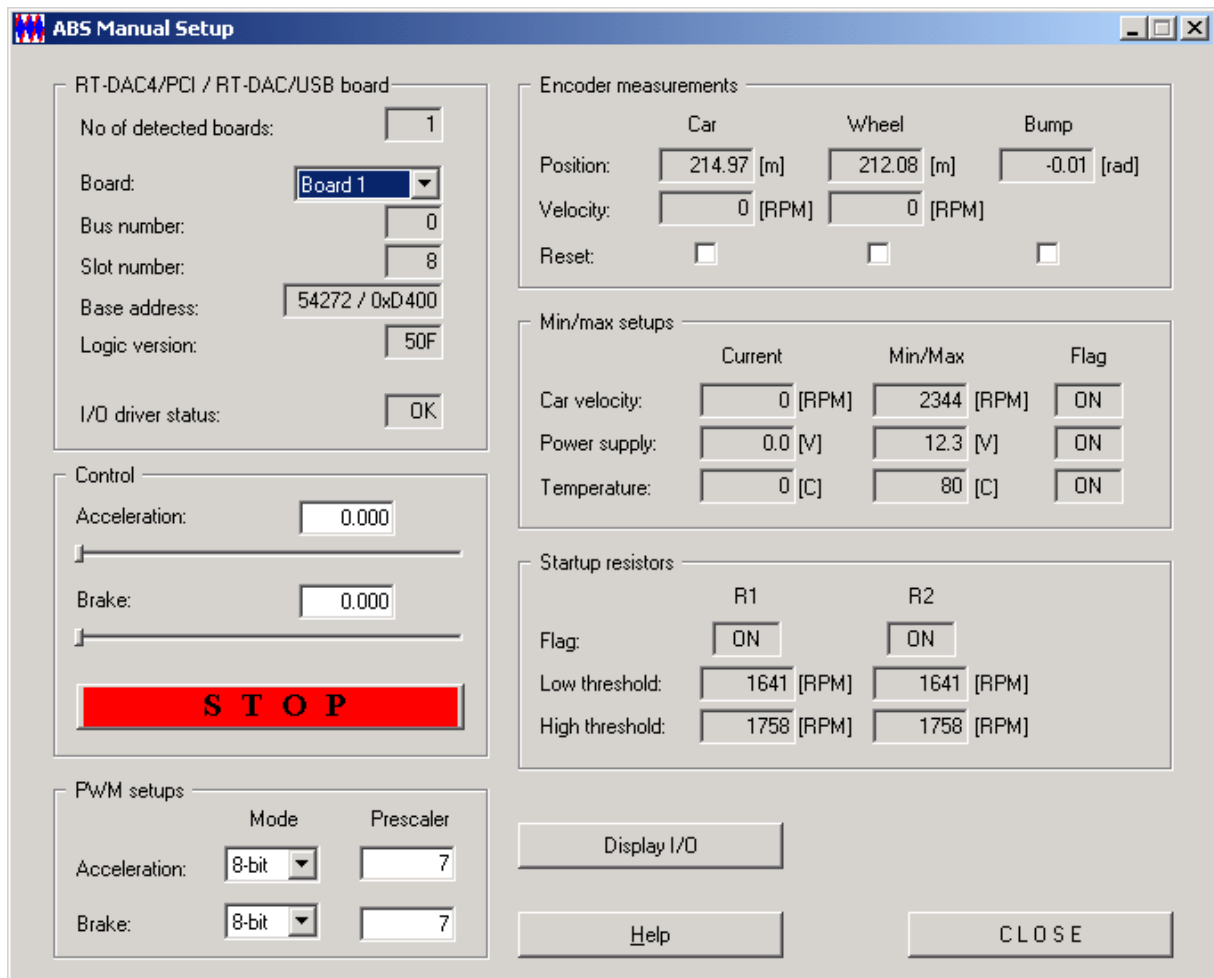


Fig. 4.1 The *Manual Setup* window

RT-DAC/PCI / RT-DAC/USB board

The frame contains the parameters of the RT-DAC/PCI or RT-DAC/USB boards detected by the computer. With respect to the interface board applied to control the ABS system the program operates with RT-DAC/PCI or RT-DAC/USB boards.

No of detected boards

The number of detected RT-DAC/PCI or RT-DAC/USB boards. The number equal to zero means that none I/O device has been detected by the software. If more then one board is detected the *Board* list must be used to select the board that communicates with the program.

Board

The list applied to select the board currently used by the program. The list contains a single entry for each RT-DAC/PCI or RT-DAC/USB boards installed in the computer. A new selection executed at the list automatically changes values of the remaining parameters within the frame. If more then one I/O device is detected the selection at the list must point to the board applied to control the ABS system. Otherwise the program is not able to operate in a proper way.

Bus number

The number of the PCI bus where the current RT-DAC/PCI board is plugged-in. The parameter may be useful to distinguish boards, when more than one board is used and the computer system contains more than a single PCI bus. This parameter is only active if the RT-DAC/PCI boards are applied.

Slot number

The number of the PCI slot where the current RT-DAC/PCI board is plugged-in. The parameter may be useful to distinguish boards, when more than one board is used. This parameter is only active if the RT-DAC/PCI boards are applied.

Base address

The base address of the current RT-DAC/PCI board. The RT-DAC/PCI board occupies 256 bytes of the I/O address space of the microprocessor. The base address is equal to the beginning of the occupied I/O range. The I/O space is assigned to the board by the computer system and may be different among computers. This parameter is only active if the RT-DAC/PCI boards are applied. The base address is given in the decimal and hexadecimal forms.

Logic version

The number of the configuration logic of the on-board FPGA chip. A logic version corresponds to the configuration of the RT-DAC/PCI board defined by this logic and depends on the version of the ABS model.

I/O driver status

The status of the driver that allows the access to the I/O address space of the microprocessor. The status has to be OK string. In other case the ABS software HAS TO BE REINSTALLED. This parameter is only active if the RT-DAC/PCI boards are applied.

Control

The frame allows to set the control signals: the acceleration and the braking DC drives controls.

Acceleration, Brake

The control signals of the DC drives may be set by entering a new value into the corresponding edit field or by dragging the corresponding slider. There are two flat DC drives: the bigger drive devoted to accelerate the bottom wheel and the small one devoted to braking of the upper wheel. The control signals for both DC drives operate in the range from 0.0 to +1.0. The control equal to 0.0 corresponds to the lack of force. The control equal to +1.0 corresponds to the maximum control.

If a new control value is entered in an edit field the corresponding slider changes respectively its value. Dragging the slider results in changing the value of the corresponding control signal and in changing the value in the respective edit field.

S T O P

The pushbutton is applied to switch off all the control signals. When it is pressed all the control values are set to zero.

PWM setups

The acceleration and braking DC drives are controlled by the PWM waves. The resolution and frequency of the PWM waves may be set in the PWM setups frame. The resolution is selected from the Mode list. The Prescaler values are responsible for the PWM frequencies.

Mode

The list is applied to select 8-bit or 12-bit PWM resolution. The 8-bit mode allows PWM to operate in high speed and the 12-bit mode allows PWM to achieve high accuracy of the output.

Prescaler

In the 12-bit mode a single PWM period contains 4095 impulses of the input prescaler frequency. A number from 0 to 4095 defines the time period when the PWM signal is set to '1'. In the 8-bit mode a PWM period contains 255 impulses of the input prescaler frequency. A number from 0 to 255 defines the time period when the PWM signal is set to '1'. The input (base) frequency of the PWM channels is set by default to 40MHz. This frequency is divided by the counter (called prescaler), which creates the PWM base period and the period of the „H” state. The valid prescaler value is a number taken from the range [0 - 65535].

The frequency of the PWM waves is calculated according to the formulas:

$$f_{PWM} = \frac{40MHz}{(prescaler + 1) * 255} \quad \text{for 8-bit mode and}$$

$$f_{PWM} = \frac{40MHz}{(prescaler + 1) * 4095} \quad \text{for 12-bit mode.}$$

Encoder measurements

With respect to the ABS version there are two or three encoders mounted in the system. Two encoders are obligatory and are applied to measure the angles of the car and wheel disks. The angles are converted by the RT-DAC board to obtain the distance (position) of the car and wheel disks. As well the RT-DAC board to calculate the velocities of the disks applies these encoder signals. The third, optional, encoder is applied to measure the angle of the wheel disk (the upper disk) over the car disk (the bottom disk). This measurement is called „bump”. There is possible to reset all the encoders.

Car, Wheel, Bump

In the Car, Wheel and Bump columns are given the positions, velocities and reset signals of the respective encoders.

Position

The positions (distances) of the car and wheel disks and the angle of the wheel shaft are shown. The distances are given in meters. The angle of the shaft is given in radians.

Velocity

The velocities of the car and wheel disks are displayed. The velocities are given in RPMs.

Reset

The checkboxes are applied to generate the reset signals for the encoders. If a checkbox is marked the corresponding encoder remains in the origin, zero state. If a checkbox is unmarked the RT-DAC board calculates the position and velocity values.

Min/max setups

The RT-DAC board automatically measures the velocity of the car disk, the power supply voltage and the temperature inside the power interface. If the velocity of the car disk is greater than the given value or the power supply voltage is less than the minimum voltage or the temperature is greater than maximum temperature then the respective flags are set. If any flag is set the control signals for the acceleration DC drive and for the braking drive are automatically set to zero by the RT-DAC board. If the car velocity flag is set it remains in this state until the car wheel stops. The power supply flag remains set until the power supply voltage is below the minimum level. The temperature flag remains set until the temperature of the power interface is higher than the maximum temperature.

Current, Min/Max, Flag

The columns displays the current values, the maximum or minimum values and the flag state of the car wheel velocity, the power supply voltage and the power interface temperature.

Car velocity

The current car velocity, the maximum velocity value and the car velocity flag are shown.

Power supply

The current power supply voltage, the maximum voltage value and the power supply flag are shown.

Temperature

The current power interface temperature, the maximum temperature value and the temperature flag are presented.

Startup resistors

During the first acceleration phase the flat acceleration DC drive consumes high current. To limit the current consumption the power interface contains two resistors connected in serial to the DC drive at the beginning of the motion.

R1, R2

The columns present the flags and threshold levels of the resistors.

Flag

If a flag is set the corresponding resistor is connected to the drive.

Low threshold

The low threshold velocity level of the car disk. If the car disk velocity goes below the low threshold the corresponding resistor flag is set.

High threshold

The high threshold velocity level of the car disk. If the car disk velocity goes above the high threshold the corresponding resistor flag is cleared.

Display I/O

The Display I/O window presents the contents of all input registers of the RT-DAC/PCI board. The board contains 256 bytes of input registers organized as 64 32-bit words. The values of the registers are given in hexadecimal form. The snapshot of the Display I/O window may be useful for service purposes.

4.2. RTWT Device Driver

The driver is a software go-between for the real ABS system MATLAB environment and the RT-DAC/PCI-D acquisition board. This driver serves to the control and measurement signals. Click the *ABS Device Driver* button and the driver window opens (Fig. 4.2)

ABS Device Driver

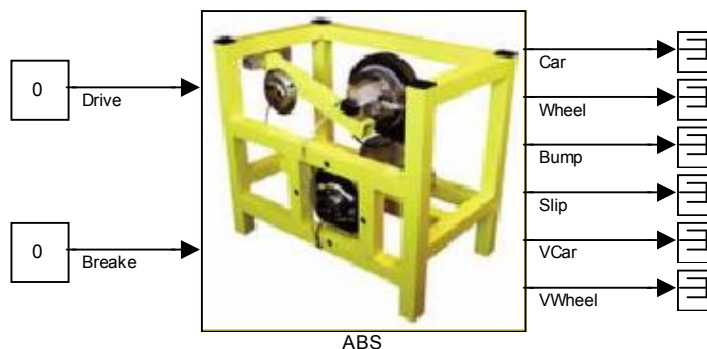


Fig. 4.2 ABS Device Driver

The driver has two inputs: PWM input -*Drive* (DC motor control) and *Brake* (brake control). There are 6 outputs of the driver: Car position (*Car*), Wheel position (*Wheel*), Bump, Car Velocity (*Vcar*), Wheel Velocity (*VWheel*) and *Slip*.

When one wants to build ones own application one can copy this driver to a new model.



Do not do changes inside the original driver. They can be done only inside its copy!!!

4.3. Simulation model

The simulation model of the ABS is accessible from the *ABS Simulation Model* button. Its external view (see Fig. 4.3) is very similar as the model given in the *ABS Device Driver*. Notice, that model has only one input denoted *Brake*. If we use the simulation model it is not necessary to accelerate the system. We set non zero initial velocities and begin the braking procedure. If we compare the simulation model to the *ABS Device Driver* we notice that four outputs are added to the model. These outputs are described in the section 6. The simulation model is used for many purposes: identification, design of controllers, etc. The interior of the simulation model is given in Fig. 4.4. There are two integrators introduced to obtain angular positions from angular velocities. Two gain blocks are used to calculate the appropriate units.

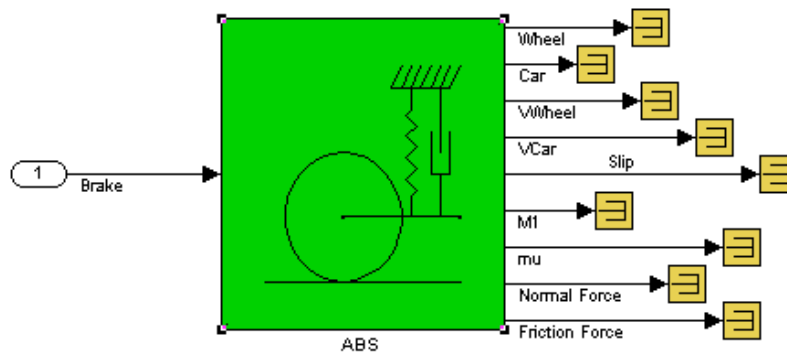


Fig. 4.3 *ABS Simulation Model*

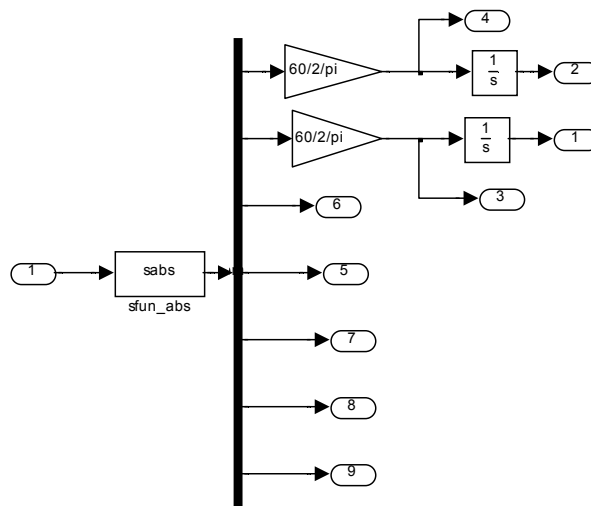


Fig. 4.4 Interior of *ABS Simulation Model*

The simulation model is running in the normal simulation mode.



Set solver options to: *Fixed step* and *Fixed-step size* equal to 0.01. The Runge-Kutta solver is recommended.

In the mask presented in Fig. 4.5 you can set all coefficients of the model and introduce initial conditions for the model state variables. See section 6 for details.

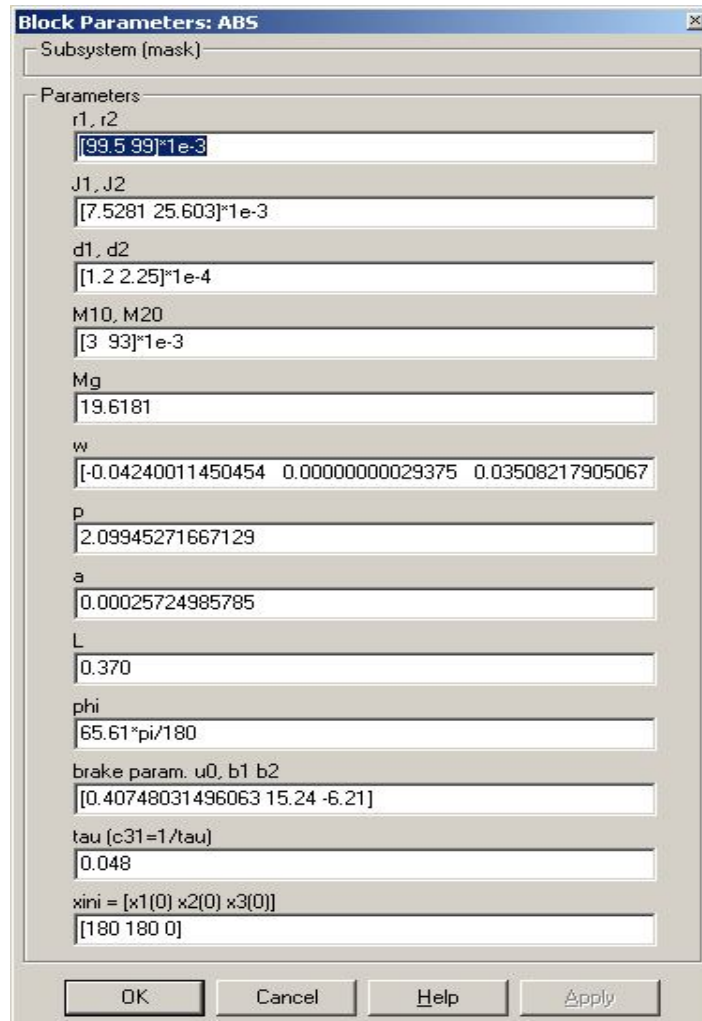


Fig. 4.5 Mask of the simulation model

4.4. Demo Controller

In this column examples of control systems are given. These demos can be used to familiarise the user with the ABS operation and allow to create the user-defined control systems. The examples must be rebuilt before using.

Due to similarity of the examples we focus our attention on one of them. After clicking on the *Relay* button the model shown in

Fig. 4.6 appears. Notice that this model looks like a typical Simulink model. The device driver is applied in the same way as other blocks from the Simulink library. The only

difference is that the model is used by Real Time Windows Target (RTWT) to create the executable library, which runs in the real-time mode.

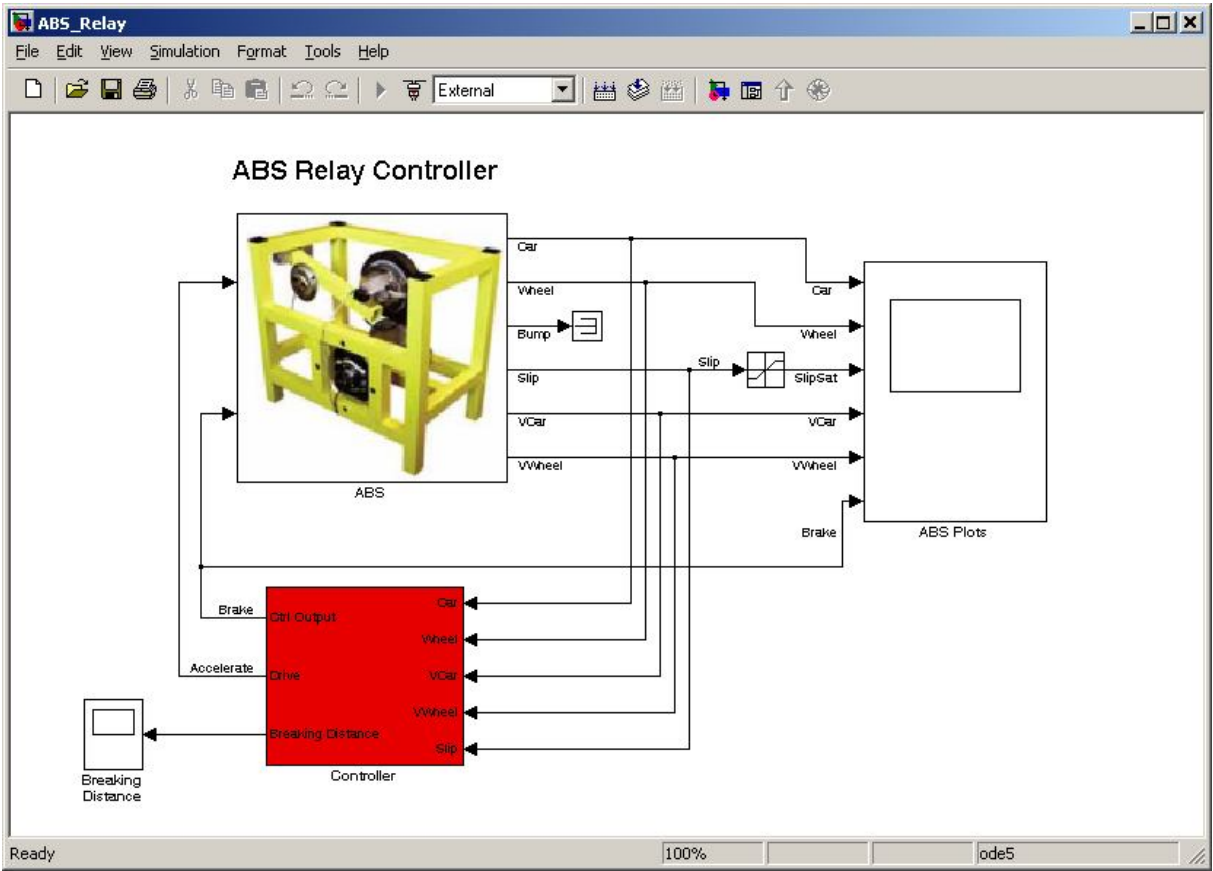


Fig. 4.6 Control system with the relay controller

Click *Controller* block. The interior of the *Controller* block shown in Fig. 4.7 appears. Pay attention that some blocks in Fig. 4.7 are the *Enable* type. These are: *Acceleration* block, *ABS Controller* block, *Calculate braking distance* and *Stop experiment* block.

How the experiment is performed? At the beginning the *Acceleration* block is enabled. The DC motor is controlled and the wheel is accelerated until the angular velocity of the wheel reaches the velocity limit. The proper logic is placed in the *Acceleration* block. If the *Acceleration* block halts then it enables three next blocks: the *ABS Controller* block, *Calculate braking distance* and *Stop experiment* block. The names of the blocks correspond to the functions realized by them. The braking action runs until the car is stopped. The relay controller is applied. The interior of the *ABS Controller* block is given in Fig. 4.8.

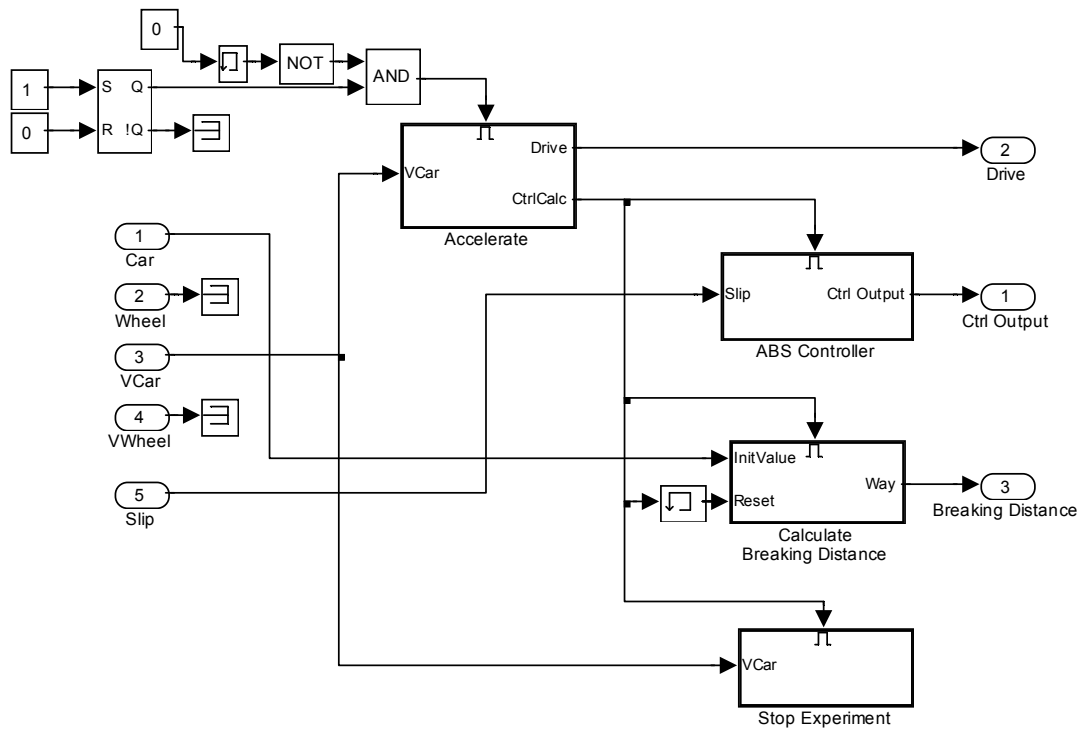


Fig. 4.7 Interior of the *Controller* block

We are ready now to return to the model window (see Fig. 4.6) and click the *Controller* block. The mask shown in Fig. 4.9 opens.

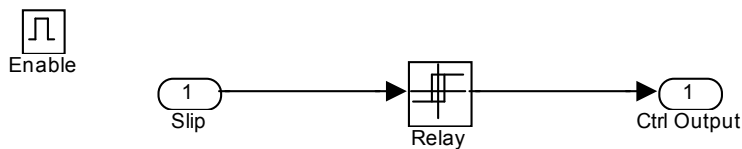


Fig. 4.8 *ABS Controller* block

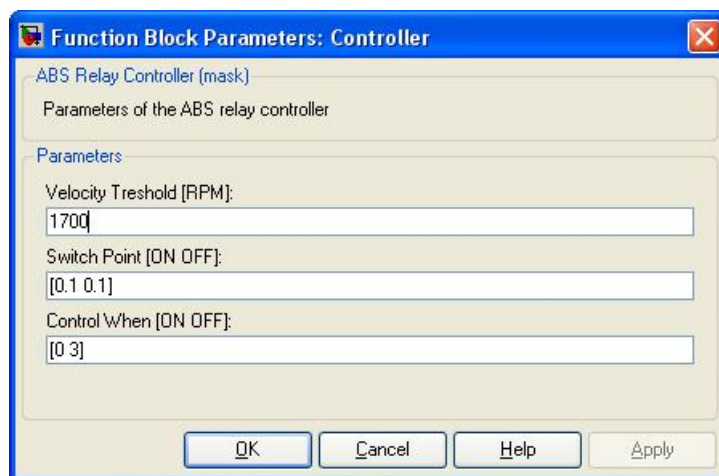


Fig. 4.9 Parameters of the relay controller

Velocity Treshold [RPM] equal to 1700 means that if velocity reaches 1700 RPM the wheel acceleration is stopped and the braking of the wheel starts.

Switch point [ON OFF] equal to [0.1 0.1] means: if the slip is greater than 0.1 the relay turns on the brake, and if the slip is less than 0.1 the relay turns off the brake.

Control When [ON OFF] equal to [0 1] means the control is equal to 0 when Switch Point is ON and the control is equal to 1 when Switch Point is OFF.

Then, choose the *Tools* pull-down menu in the Simulink model window. The pop-up menu provide a choice between predefined items. Choose the *RTW Build* item. A successful compilation and linking processes are finished with the following message:

```
Model ABS_Relay.rtd successfully created
### Successful completion of Real-Time Workshop build procedure for model:
ABS_Relay
```

If an error occurs then the message corresponding to this error is displayed in the MATLAB command window.

Now, we return to the model window and click the *Simulation/Connect to Target* option. Next, click the *Simulation/Start real-time code* item.

The system starts and the wheel accelerates. When the wheel velocity reaches upper limit (1700 rpm) the acceleration stops and the braking begins. This is visible in the plots in the scope. The experiment stops after approximately 10 seconds.

Results displayed in the scope are stored as the structure with time in the variable *ABSHistory1* (it has been set in the option of the scope *ABS plots*). In order to plot the results we can write in *Matlab Command Window*:

```
t=ABSHistory1.time;
vwheel=ABSHistory1.signals(5).values;
slip=ABSHistory1.signals(3).values;
plot(t,vwheel);grid;title('Wheel velocity');xlabel('time [sec]');
plot(t,slip);grid;title('Slip');xlabel('time [sec]');
```

All experimental results related to acceleration and braking are shown in Fig. 4.10.

```
a = 1250; b = 1950;
plot(t(a:b),vwheel(a:b));grid;title('Wheel velocity - braking time');xlabel('time [sec]');
plot(t(a:b),slip(a:b));grid;title('Slip - braking time');xlabel('time [sec]');
```

The braking phase of the experiment 1 in the zoomed form is shown in Fig. 4.11.

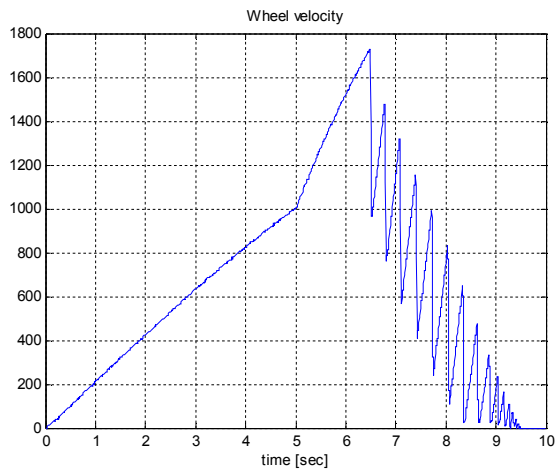


Fig. 4.10 Relay controller. Experiment 1

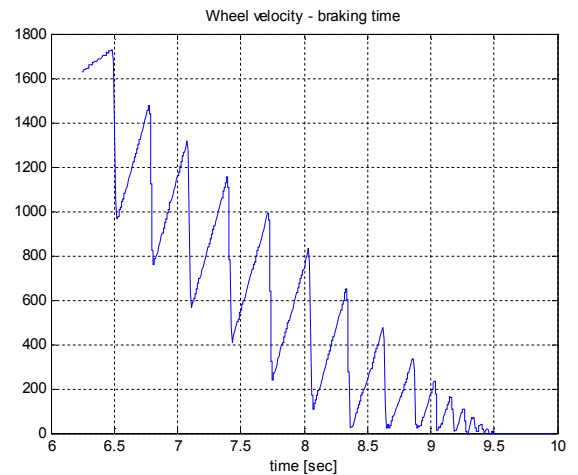
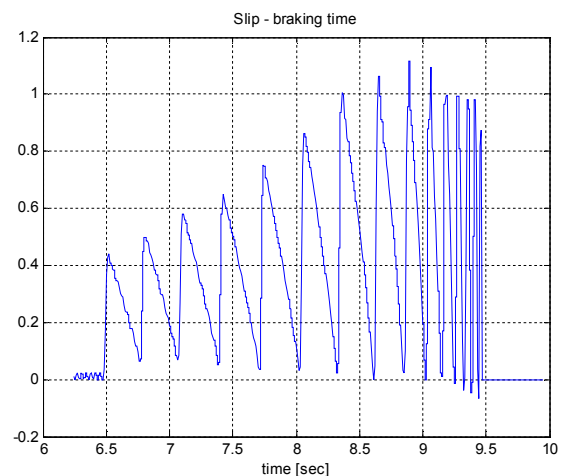
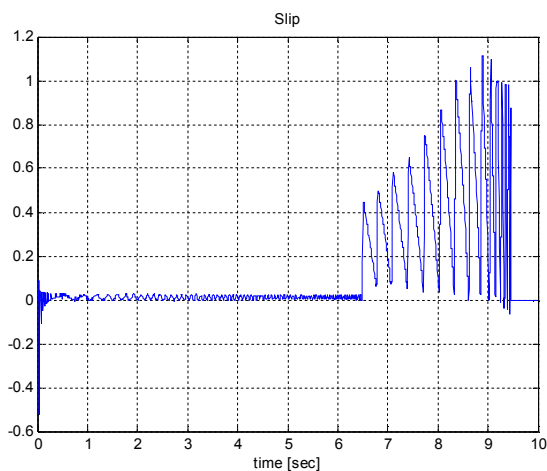


Fig. 4.11 Zoomed braking phase of experiment 1



The braking distance, read out from the *Braking Distance* scope is equal to 26.5 m. We can notice that negative slip values may appear in the plot. If small velocities are observed the numerical data are not accurate.

Note that the slip often reaches zero (see Fig. 4.11). It means that in these moments the braking is off and the brake is released. In this case the braking distance is long and it shows that the applied control algorithm does not fit well to the braking task.

In this demo one can try to adjust the parameters of the control algorithm. Click *Controller* block and set *Switch point* to [0.7 0.7] (see Fig. 4.9). Next return to the model window and click the *Simulation/Connect to Target* option. Next, click the *Simulation/Start real-time code* item.

Acceleration and braking experimental results are shown in Fig. 4.12. The braking phase of the experiment in the zoomed form is given in Fig. 4.12.

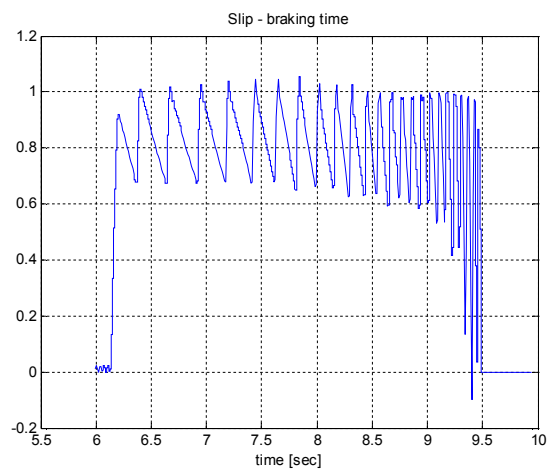
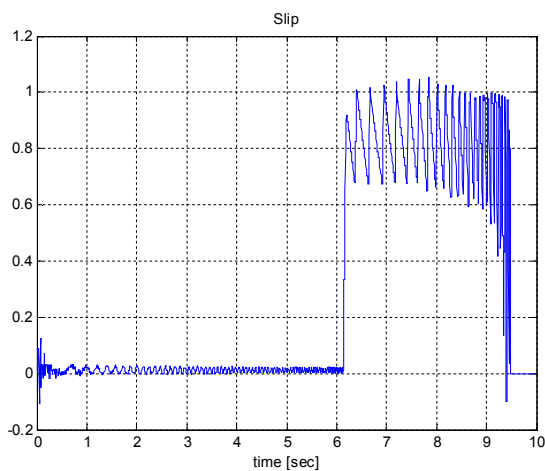
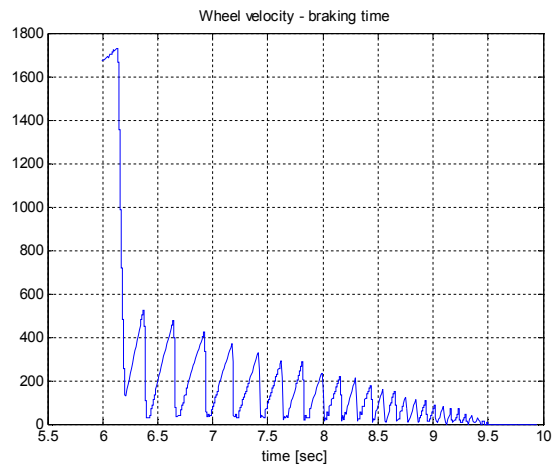
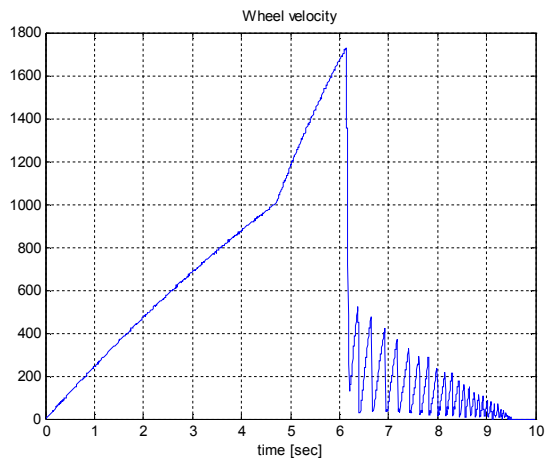


Fig. 4.12 Relay controller. Experiment 2

Fig. 4.13 Zoomed braking phase. Experiment 2

Comparing the experimental results of Experiment 1 to Experiment 2 we can notice the similar car velocity plots and very different slip plots.

The braking distance read from the *Breaking Distance* scope is equal to 22.8 m. This result is better than that obtained in the previous experiment. Note that the slip is equal to zero or negative only at the end of the braking zone when the wheel velocity is very small.

5. Prototyping your own controller in the RTWT environment

In this section the process of building of your own control system is described. The *Real Time Windows Target* (RTWT) toolbox is used. An example how to use the ABS software is shown in section 4.4. In this section we give indications how to proceed in the RTWT environment. We assume a user is familiarised with Matlab, Simulink and RTW/RTWT toolboxes.



Before start, test your MATLAB configuration by building and running an example of real-time application. Real-time Windows Target Toolbox includes the `rtvdp.mdl` model. Running this model will test the installation by running Real-Time Workshop, Real-Time Windows Target, and the Real-Time Windows Target kernel.

In the MATLAB window, type

`rtvdp`

Next, build and run the real-time model.

To build the system that operates in the real-time mode the user has to:

- create a Simulink model of the control system which consists of the *ABS system Device Driver* and other blocks chosen from the Simulink library,
- built the executable file under RTWT (see the pop-up menu in Fig. 5.1)

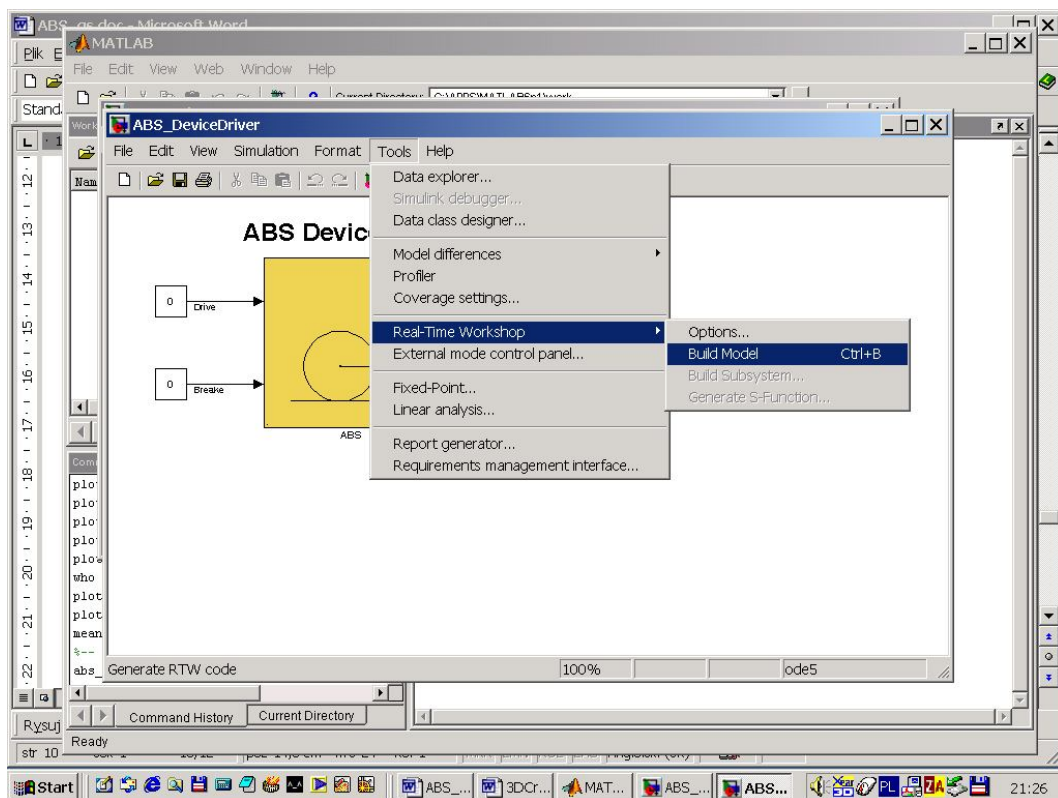


Fig. 5.1 Creating the executable file under RTWT

- click *Connect to target* and start the real-time code from the *Simulation/Start real-time code* pull-down menu.

5.1. Creating a model

The simplest way to create a Simulink model of the control system is to use one of the models included in the *ABS Control Window* as a template. For example, click on the *Relay* button and save it as *MySystem.mdl* name. The *MySystem* Simulink model is shown in Fig. 5.2.

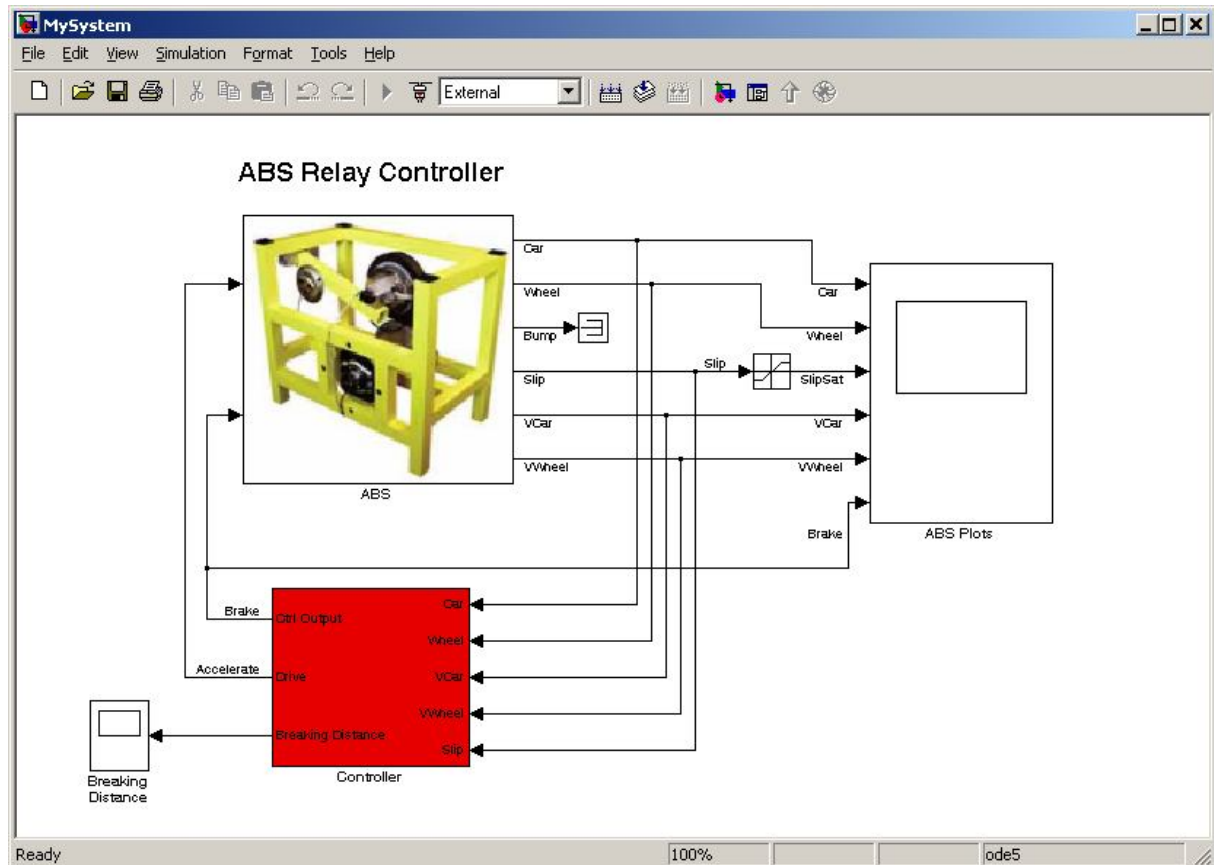


Fig. 5.2 The *MySystem* Simulink model

Now, you can modify the model. You have absolute freedom to develop your own controller. Remember to leave the *ABS driver* block in the window.

Though it is not obligatory, we recommend you to leave the scope block. You need a scope to watch how the system runs. Other blocks remaining in the window are not necessary for our new project.

Creating your own model on the basis of an old example ensures that all-internal options of the model are set properly. These options are required to proceed with compiling and linking in a proper way. To put the *ABS system Device Driver* into the real-time code a special make-file is required. This file is included into the *ABS system software*.

You can apply almost all of the blocks from the Simulink library. However, some of them cannot be used (see *RTW references manual*).

The scope block properties are important for an appropriate data acquisition and watching how the system runs.

The *Scope* block properties are defined in the *Scope* property window (see Fig. 5.3). This window opens after the selection of the *Scope/Properties* tab. You can gather measurement data to the *Matlab Workspace* marking the *Save data to workspace* checkbox. The data is placed under *Variable name*. The variable format can be set as *structure* or *matrix*. The default *Sampling Decimation* parameter value is set to 1. This means that each measured point is plotted and saved. Often we choose the *Decimation* parameter value equal to 5 or 10. This is a good choice to get enough points to describe the signal behaviour and simultaneously to save the computer memory.

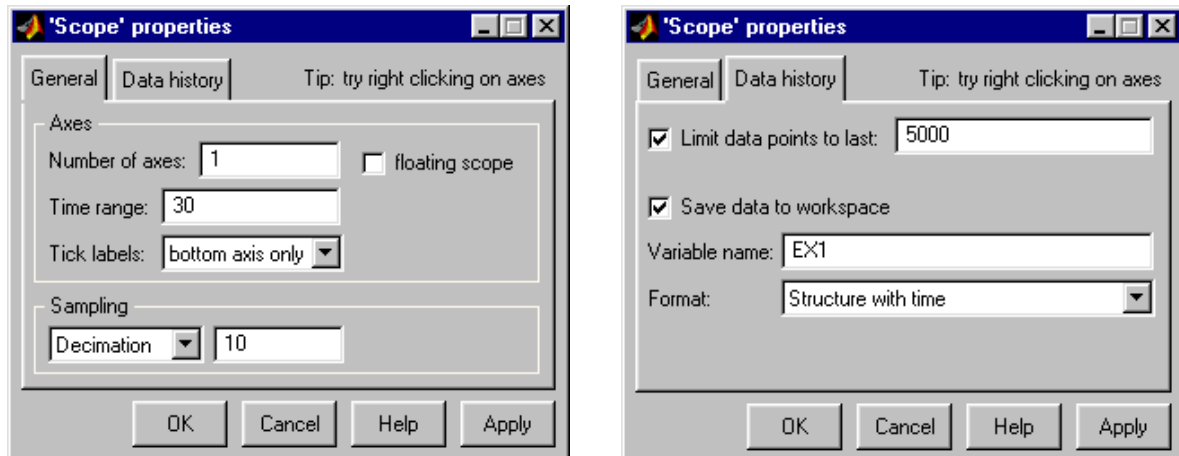


Fig. 5.3 Setting the parameters of the *Scope* block

5.2. Code generation and the build process

Once a model of the system has been designed the code for real-time mode can be generated, compiled, linked and downloaded into the processor.

The code is generated by the use of Target Language Compiler (TLC) (see description of the Simulink Target Language). The make-file is used to build and download object files to the target hardware automatically.

First, you have to specify the simulation parameters of your Simulink model in the *Configuration parameters* dialog box. The *RTW* page appears when you select the *RTW* tab (Fig. 5.4). The *RTW* page allows you to set the real-time build options and then to start the building process of the executable file.

The system target file name is *rtwin.tlc*. It manages the code generation process. It is ready to be used by the built-in Open Watcom compiler.

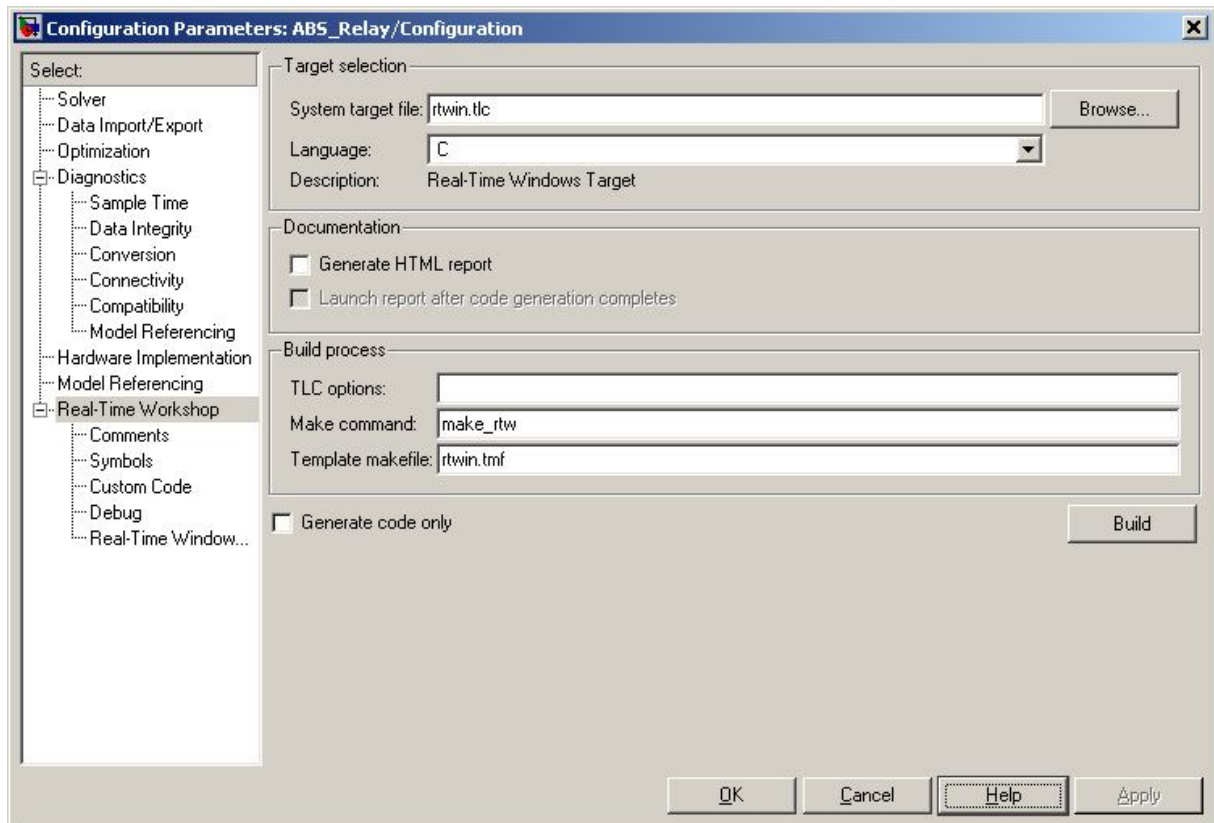



Fig. 5.4 RTW tab of the *Simulation parameters*

There is additional option *Generate code only* which have to be properly marked as shown in Fig. 5.4. If marked, a code is generated but compilation is not performed.

The *Solver* dialog box appears when you select the *Solver* tab (Fig. 5.5). The *Solver* tab allows you to set the simulation parameters. Several parameters and options are available in the window. The *Fixed-step size* editable text box is set to 0.01 (this is the sampling period in seconds).

 **The *Fixed-step* solver is obligatory for real-time applications. If you use an arbitrary block from the discrete Simulink library remember that different sampling periods must have a common divider.**

The *Start time* has to be set to 0. The solver has to be selected. In our example the fifth-order integration method – *ode5* is chosen.

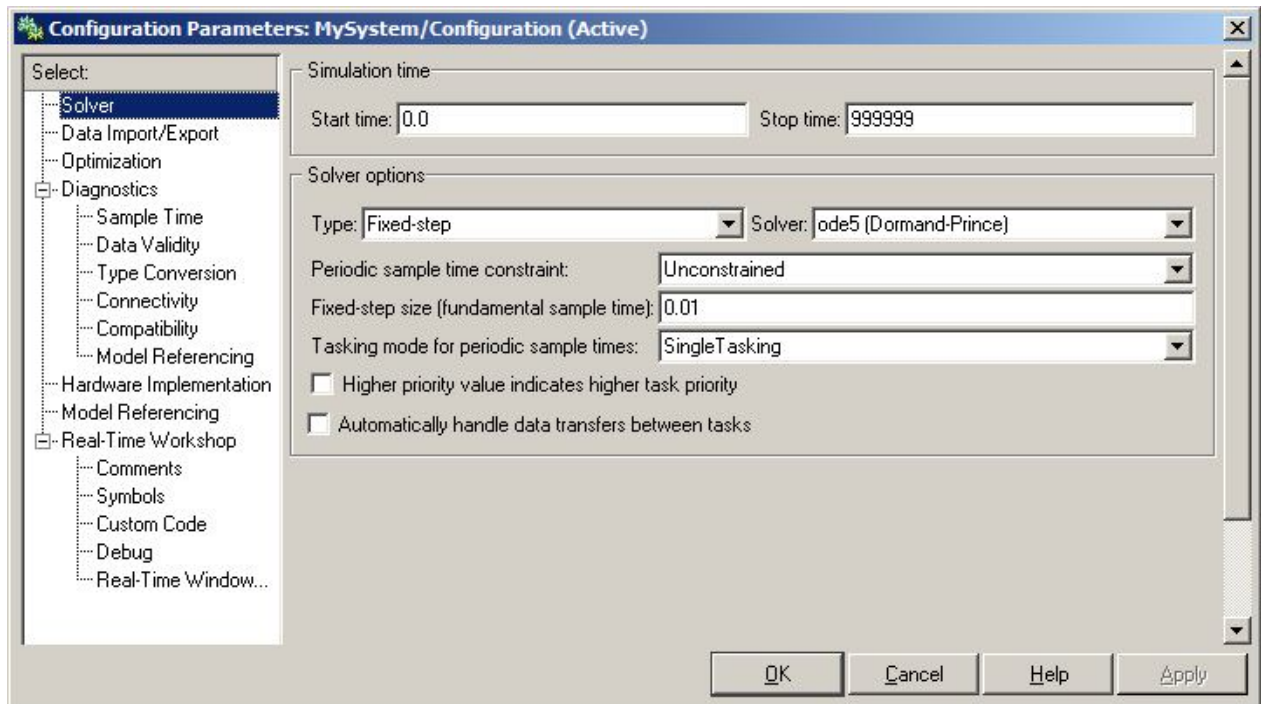


Fig. 5.5 Solver tab in the Configuration parameters

As it was mentioned above the third party compiler is not requested. The built-in Open Watcom compiler is used to creating real-time executable code for RTWT.

If all parameters are set properly you can start the executable building process. For this purpose press the *Build* push button on the *RTW* tab (Fig. 5.4). Successful compilation and linking processes generate the following message:

```
Model MySystem.rtd successfully created
#### Successful completion of Real-Time Workshop build procedure for model: MySystem
```

Otherwise, an error message is displayed in the MATLAB command window.

6. Mathematical model

Consider the system depicted in Fig. 6.1.

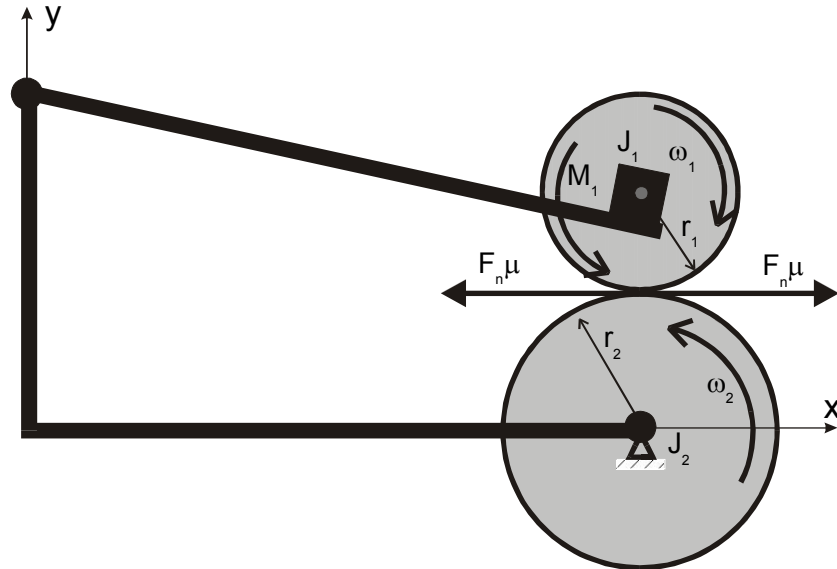


Fig. 6.1 Schematic diagram of ABS

There are two rolling wheels. The lower *car-road wheel* animating relative road motion and the upper *car wheel* permanently remaining in a rolling contact with the lower wheel. The car-road wheel has a smooth surface which can be covered by a given material to animate a surface of the road.

There are three angles measured by three encoders. The accuracies of measurements are $2\pi/4096 = 0.0015$ [rad]. The two first are rotational angles of the wheels. The third measurement is the angle between subsequent balance lever angular positions. This one might be used in more complex models of ABS. The wheel angular velocities are not measured. They have to be observed. The simplest Euler formula is used. The sample time is defined as 0.5 ms.

The upper wheel is equipped in the disk brake system connected via hydraulic coupling to the brake lever which by the tight side and tightening pulley is driven by the small DC motor. The lower wheel is coupled to the big flat DC motor which is power supplied to accelerate the wheel. During the braking phase the power supply of the DC motor is switched off. Both big and small DC motors are controlled by PWM (pulse-width modulation) signals of the 3.5 kHz frequency. The varying pulse-width is the control variable.

We define the car velocity to be equivalent to the angular velocity of the lower wheel multiplied by the radius of this wheel. We define the angular velocity of the wheel to be equivalent to the angular velocity of the upper wheel.

The state variables and parameters of the model are given in Table1.

Table 1. Parameters of the model

Name	Description	Units
x_1	angular velocity of the upper wheel	rad/s
x_2	angular velocity of the lower wheel	rad/s
M_1	braking torque	Nm
r_1	radius of the upper wheel	m
r_2	radius of the lower wheel	m
J_1	moment of inertia of the upper wheel	kgm ²
J_2	moment of inertia of the lower wheel	kgm ²
d_1	viscous friction coefficient of the upper wheel	kgm ² /s
d_2	viscous friction coefficient of the lower wheel	kgm ² /s
F_n	total force generated by the upper wheel and pressing on the lower wheel	N
$\mu(\lambda)$	friction coefficient between wheels	
λ	slip – the relative difference of the wheel velocities	
F_n	normal force – the upper wheel acting on the lower wheel	N
$\mu(\lambda)$	friction coefficient between the wheels	
λ	slip – the relative difference of the wheels velocities	
M_{10}	static friction of the upper wheel	Nm
M_{20}	static friction of the lower wheel	Nm
M_g	gravitational and shock absorber torques acting on the balance lever	Nm
L	distance between the contact point of the wheels and the rotational axis of the balance lever (see Fig. 6.2)	m
φ	angle between the normal in the contact point and the line L	°
u	control of the brake	

We assume that the friction force is proportional to the normal pressing force F_n . $\mu(\lambda)$ is the proportionality coefficient.

We introduce auxiliary variables:

$$s = \text{sgn}(r_2 x_2 - r_1 x_1), \quad (6.1)$$

$$s_1 = \text{sgn}(x_1), \quad (6.2)$$

$$s_2 = \text{sgn}(x_2), \quad (6.3)$$

We denote λ as the slip (the relative difference of the wheels velocities) in the following way

$$\lambda = \begin{cases} \frac{r_2 x_2 - r_1 x_1}{r_2 x_2}, & r_2 x_2 \geq r_1 x_1, x_1 \geq 0, x_2 \geq 0, \\ \frac{r_1 x_1 - r_2 x_2}{r_1 x_1}, & r_2 x_2 < r_1 x_1, x_1 \geq 0, x_2 \geq 0, \\ \frac{r_2 x_2 - r_1 x_1}{r_2 x_2}, & r_2 x_2 < r_1 x_1, x_1 < 0, x_2 < 0, \\ \frac{r_1 x_1 - r_2 x_2}{r_1 x_1}, & r_2 x_2 \geq r_1 x_1, x_1 < 0, x_2 < 0, \\ 1, & x_1 < 0, x_2 \geq 0, \\ 1, & x_1 \geq 0, x_2 < 0. \end{cases} \quad (6.4)$$

6.1. Equations of motion

There are three torques acting on the upper wheel: the braking torque M_1 , the friction torque in the upper bearing and the friction torque among the wheels. There are two torques acting on the lower wheel: the friction torque in the lower bearing and the friction torque among the wheels. Besides these we have two forces acting on the lower wheel: the gravity force of the upper wheel and the pressing force of the shock absorber.

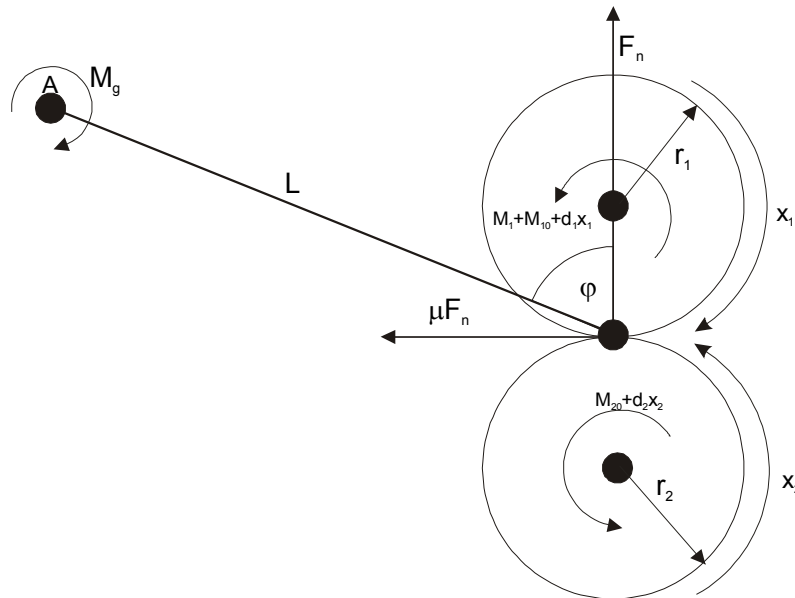


Fig. 6.2 Auxiliary diagram to develop the model

We can write the following motion equation for the upper wheel

$$J_1 \dot{x}_1 = F_n r_1 s \mu(\lambda) - d_1 x_1 - s_1 M_{10} - s_1 M_1. \quad (6.1.1)$$

We can write the following motion equation for the lower wheel

$$J_2 \dot{x}_2 = -F_n r_2 s \mu(\lambda) - d_2 x_2 - s_2 M_{20}. \quad (6.1.2)$$

To derive the normal force F_n we can write the sum of torques corresponding to the point A (see Fig. 6.2).

$$F_n L (\sin \varphi - s \mu(\lambda) \cos \varphi) = M_g + s_1 M_1 + s_1 M_{10} + d_1 x_1 \quad (6.1.3)$$

$$F_n = \frac{M_g + s_1 M_1 + s_1 M_{10} + d_1 x_1}{L (\sin \varphi - s \mu(\lambda) \cos \varphi)}. \quad (6.1.4)$$

Putting formula (6.1.4) to formulas (6.1.1) and (6.1.2) we obtain

$$J_1 \dot{x}_1 = \frac{M_g + s_1 M_1 + s_1 M_{10} + d_1 x_1}{L (\sin \varphi - s \mu(\lambda) \cos \varphi)} r_1 s \mu(\lambda) - s_1 M_1 - d_1 x_1 - s_1 M_{10}, \quad (6.1.5)$$

$$J_2 \dot{x}_2 = -\frac{M_g + s_1 M_1 + s_1 M_{10} + d_1 x_1}{L (\sin \varphi - s \mu(\lambda) \cos \varphi)} r_2 s \mu(\lambda) - d_2 x_2 - s_2 M_{20}. \quad (6.1.6)$$

In both equations we have the common factor

$$S(\lambda) = \frac{s \mu(\lambda)}{L (\sin \varphi - s \mu(\lambda) \cos \varphi)}. \quad (6.1.7)$$

$\mu(\lambda)$ can be approximated by the following formula

$$\mu(\lambda) = \frac{w_4 \lambda^p}{a + \lambda^p} + w_3 \lambda^3 + w_2 \lambda^2 + w_1 \lambda. \quad (6.1.8)$$

It will be explained later how to derive $\mu(\lambda)$ and $S(\lambda)$ on the basis of experiments.

We denote

$$c_{11} = \frac{r_1 d_1}{J_1}, c_{12} = \frac{(s_1 M_{10} + M_g) r_1}{J_1}, c_{13} = -\frac{d_1}{J_1}, c_{14} = -\frac{s_1 M_{10}}{J_1}, c_{15} = \frac{r_1}{J_1}, c_{16} = -\frac{1}{J_1},$$

$$c_{21} = -\frac{r_2 d_1}{J_2}, c_{22} = -\frac{(s_1 M_{10} + M_g) r_2}{J_2}, c_{23} = -\frac{d_2}{J_2}, c_{24} = -\frac{s_2 M_{20}}{J_2}, c_{25} = -\frac{r_2}{J_2}.$$

The equations (6.1.1) and (6.1.2) have the forms:

$$\dot{x}_1 = S(\lambda)(c_{11} x_1 + c_{12}) + c_{13} x_1 + c_{14} + (c_{15} S(\lambda) + c_{16}) s_1 M_1, \quad (6.1.9)$$

$$\dot{x}_2 = S(\lambda)(c_{21} x_1 + c_{22}) + c_{23} x_2 + c_{24} + c_{25} S(x_1, x_2) s_1 M_1, \quad (6.1.10)$$

The driving system of the brake is described by the following equation

$$\dot{M}_1 = c_{31}(b(u) - M_1) \quad (6.1.11)$$

Constant c_{31} is equal to 20.37 [1/s]. Function $b(u)$ (see formula (6.1.11)) may be approximated by the formula:

$$b(u) = \begin{cases} b_1 u + b_2, & u \geq u_0 \\ 0, & u < u_0 \end{cases}$$

$$b_1 = 15.24, \quad b_2 = -6.21, \quad u_0 = 0.415.$$

6.2. Identification

6.2.1. Geometrical issues

At the beginning the distance L , angle ϕ and radii of the wheels (see Fig. 2) are measured. The following results are obtained (see Tab. 2.).

Table 2. Geometrical parameters

r_1 [m]	r_2 [m]	ϕ [°]	L [m]
0.0995	0.099	65.61	0.370

6.2.2. Deriving of the torque M_g

From the formula (6.1.4) for $x_1 = 0$ we obtain $M_g = F_n L \sin \phi$. F_n is measured by a dynamometer or weighing conveyor (see Fig. 6.3)

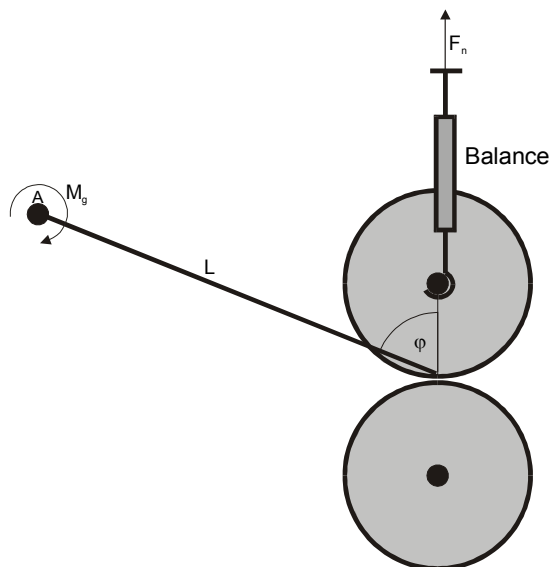


Fig. 6.3 Deriving of M_g

We obtain $F_n = 58.214$ N and we derive $M_g = F_n L \sin \varphi = 19.62$ N.

6.2.3. Derivations of the moments of inertia

J_1 and J_2 are derived on the basis of the upper and lower wheels velocities measured for a constant driving torque applied to the considered wheel (see Fig. 6.4).

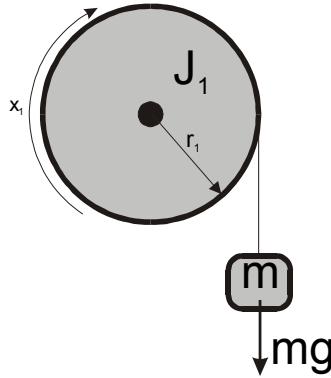


Fig. 6.4. Measurement of the moment of inertia

From formula (6.1) we obtain

$$J_1 \dot{x}_1 = -d_1 x_1 - M_{10} + mgr_1. \quad (6.2.1)$$

For a small x_1 the first term of the right hand side can be neglected.

$$J_1 \dot{x}_1 = -M_{10} + mgr_1, \quad x_1(0) = 0. \quad (6.2.2)$$

The solution of the above equation has the form

$$x_1(t) = \frac{mgr_1 - M_{10}}{J_1} t. \quad (6.2.3)$$

After few seconds the load is released and formula (2.3.1) becomes

$$J_1 \dot{x}_1 = -M_{10}, \quad x_1(0) = x_{10}. \quad (6.2.4)$$

$$x_1(t) = x_{10} - \frac{M_{10}}{J_1} t. \quad (6.2.5)$$

The slope of line (6.2.3) is $a_1 = \frac{mgr_1 - M_{10}}{J_1}$. Respectively the slope of line (6.2.5) is

$$a_2 = \frac{-M_{10}}{J_1}.$$

Hence

$$a_1 - a_2 = \frac{mgr_1 - M_{10}}{J_1} + \frac{M_{10}}{J_1} = \frac{mgr_1}{J_1},$$

The moment of inertia is

$$J_1 = \frac{mgr_1}{a_1 - a_2}.$$

The moment of inertia of the lower wheel is obtained. The results are given in Table 3.

Table 3. Moments of inertia

J_1 [kgm ²]	J_2 [kgm ²]
$7.53 \cdot 10^{-3}$	$25.60 \cdot 10^{-3}$

6.2.4. Deriving the friction coefficients in the bearings

Viscous friction coefficients in the bearings and static frictions are derived on the basis of the following equations

$$J_1 \dot{x}_1 = -d_1 x_1 - M_{10}$$

$$J_2 \dot{x}_2 = -d_2 x_2 - M_{20}.$$

The coefficients are obtained by the *mean square error* method. Results are presented in Table 4 and Fig. 6.5.

Table 4. Measurement results of friction coefficients

	Upper wheel		Lower wheel	
	d_1 [kgm ² /s]	M_{10} [Nm]	d_2 [kgm ² /s]	M_{20} [Nm]
1	1.3436e-004	0.0022	2.1528e-004	0.0920
2	1.1978e-004	0.0025	2.1354e-004	0.0935
3	1.0209e-004	0.0050	2.1521e-004	0.0920
Mean value	1.1874e-004	0.0032	2.1468e-004	0.0925

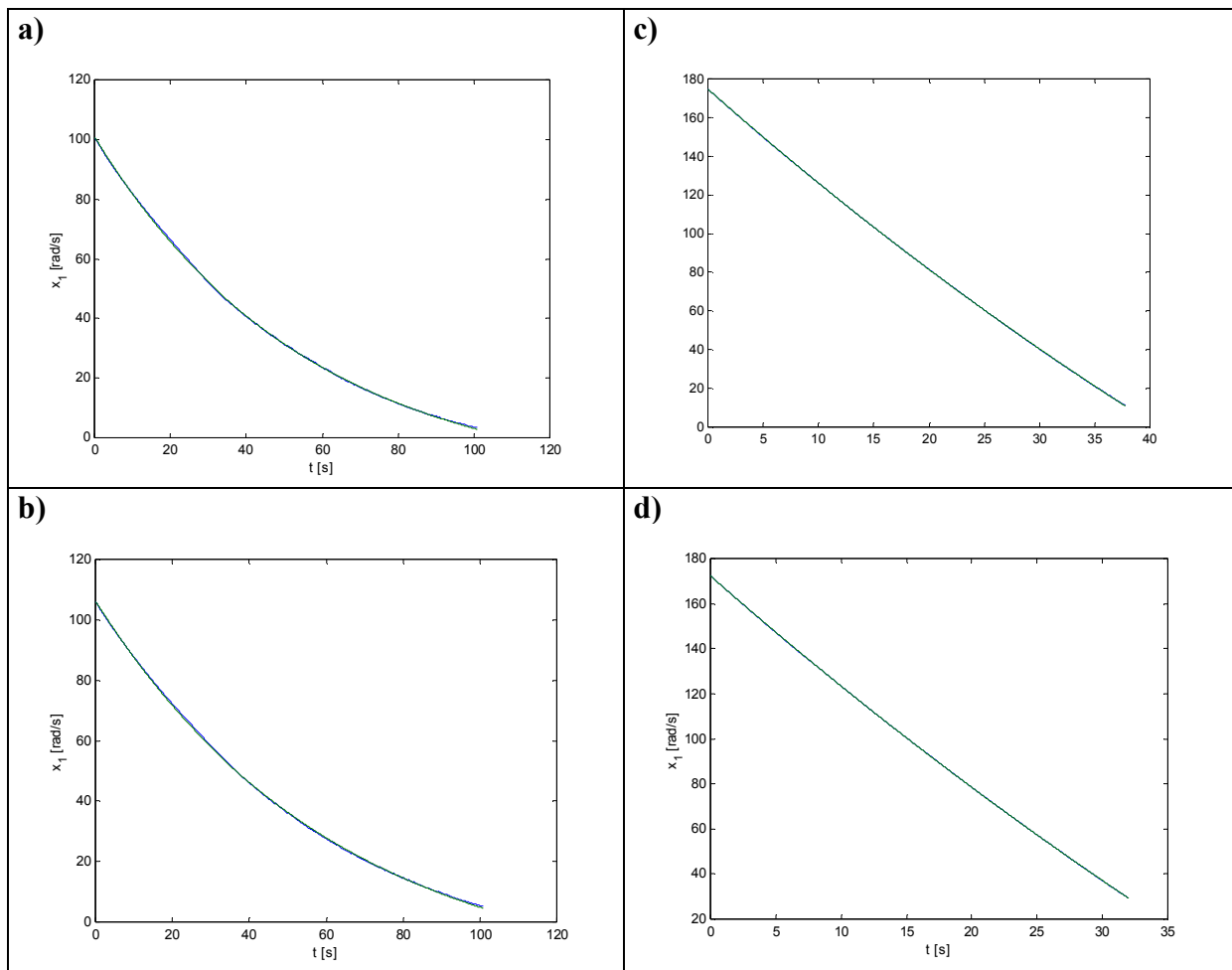


Fig. 6.5 Experimental results. a), b) run of the upper wheel, c), d) run of the lower wheel. Blue – real time experiment, green – simulated model.

6.2.5. Identification of the friction coefficient and the braking torque

From formulas (6.1) and (6.2) we have

$$M_1 = - \left[\frac{r_2}{r_1} (J_2 \dot{x}_2 + d_2 x_2 + M_{20}) + J_1 \dot{x}_1 + d_1 x_1 + M_{10} \right] \quad (6.2.6)$$

The braking torque versus velocity x_1 and control u is derived from formula (6.2.6).

From formulas (6.1.2) and (6.1.4) we obtain

$$F_n = \frac{M_g + M_1 + M_{10} + d_1 x_1}{L(\sin \phi - \mu(\lambda) \cos \phi)} = - \frac{J_2 \dot{x}_2 + d_2 x_2 + M_{20}}{\mu(\lambda) r_2}. \quad (6.2.7)$$

$$\mu = \frac{\frac{L \sin \phi}{r_2} (J_2 \dot{x}_2 + d_2 x_2 + M_{20})}{(J_2 \dot{x}_2 + d_2 x_2 + M_{20}) \left(\frac{L \cos \phi}{r_2} + \frac{r_2}{r_1} \right) + J_1 \dot{x}_1 - M_g} \quad (6.2.8)$$

From formula (6.2.8) the friction coefficient versus the slip is calculated. The accelerations in formulas (6.2.6) and (6.2.7) are not measured. They are observed on the basis of the measured velocities.

A number of experiments have been performed. In every experiment the wheels are accelerated to reach the angular velocity about 180 rad/s. Then, the brake procedure begins. The small DC motor responsible for braking obtains a jump of control of a magnitude between 0 and 1. The wheel velocities are registered during the brake procedure. On the basis of formulas (6.2.6) and (6.2.7) the mean square braking torque and the friction coefficient corresponding to the upper wheel are derived. The results of experiments are presented in Fig. 6.6

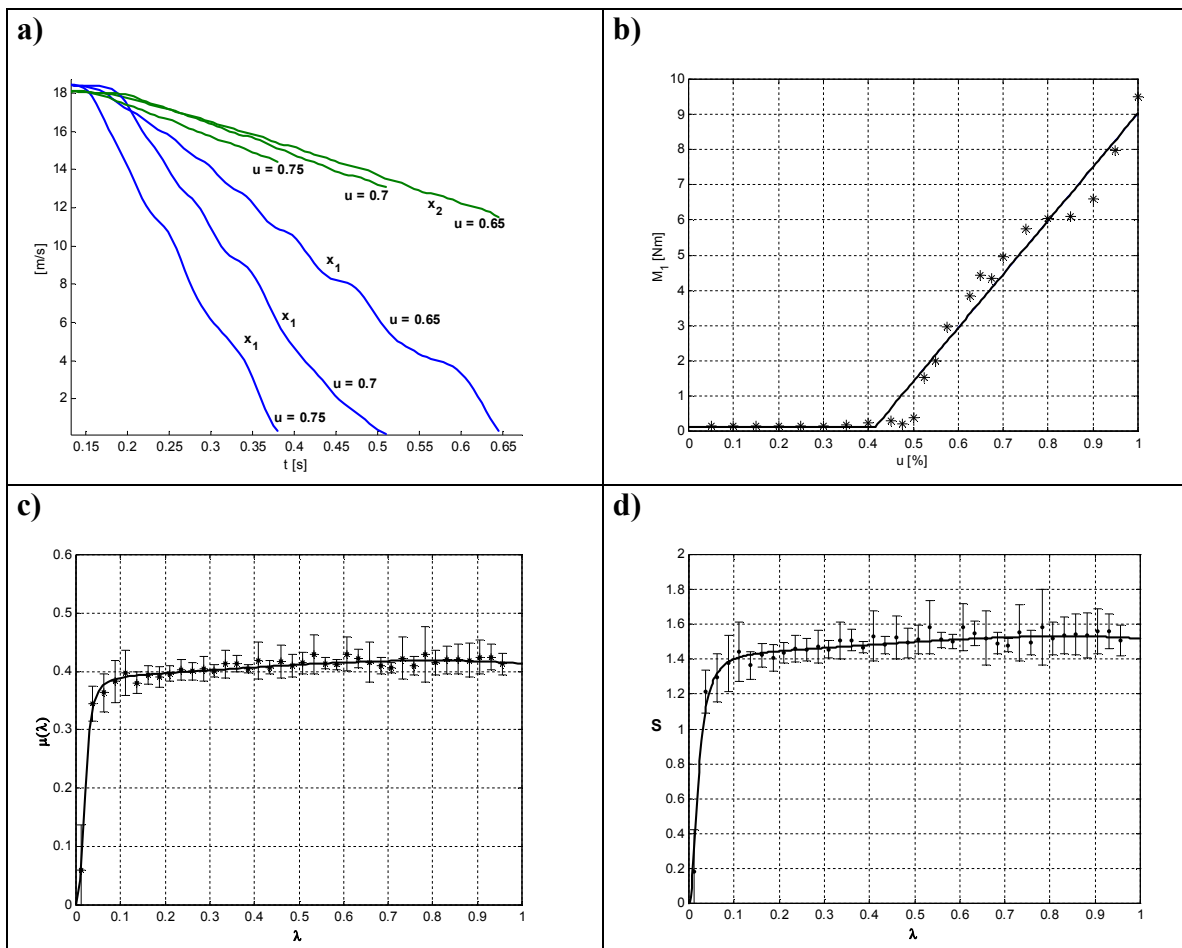


Fig. 6.6. Results of experiments. a) Velocities registered under three different controls, b) static characteristic of the brake (formula 1.11), c) friction coefficient vs. slip, d) function denoted by S.

Constant c_{31} is equal to 10.37 [1/s]. Function $b(u)$ (see. formula (6.1.11)) can be approximated by the formula:

$$b(u) = \begin{cases} 15.24u - 6.21, & u \geq 0.415 \\ 0.12, & u < 0.415 \end{cases}$$

The model coefficients have the following values:

$$\begin{aligned} c_{11} &= 0.00158605757097, & c_{12} &= 2.593351896228796e+002, \\ c_{11} = c_{13} &= 0.01594027709515, & c_{14} &= 0.39850692737875, \\ c_{15} &= 13.21714642472868, & c_{16} &= 132.8356424595848, \\ c_{21} &= 0.000464008124048, & c_{22} &= 75.86965129086435, \\ c_{23} &= 0.00878803265242, & c_{24} &= 3.63238682966840, \\ c_{25} &= 3.86673436706636, & c_{31} &= 20.37, \\ w_1 &= -0.04240011450454, & w_2 &= 0.00000000029375, \\ w_3 &= 0.03508217905067, & w_4 &= 0.40662691102315, \\ a &= 0.00025724985785, & p &= 2.09945271667129. \end{aligned}$$

7. Simulation of braking

The ABS dynamical model has been created in Simulink (see the green box in Fig. 7.1)

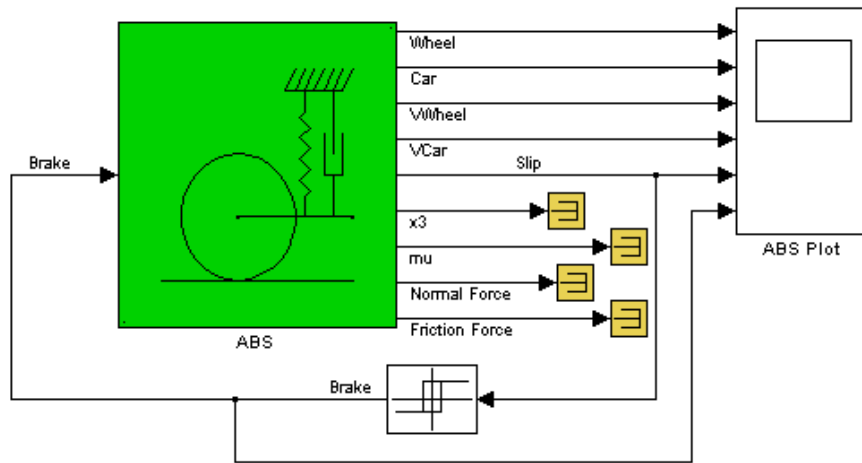


Fig. 7.1 ABS model

The ABS block has one input u and nine outputs x . The brake control is the input variable. The output variables are visible in Fig. 7.1. These are: $Wheel$, Car , $VWheel = x_1$, $VCar = x_2$, $Slip = \lambda$, $M_1 = x_3$, $mu = \mu$, $Normal Force = F_n$, $Friction Force = \mu F_n$. In fact, the model is the S-function denoted $sfun_abs$ (see Fig. 7.2). We can see the interior obtained after looking under mask (the left hand side picture) and the parameters of the S-function (the right hand side picture). A user can define 19 parameters of the S-function from r_1 to x_{ini} .

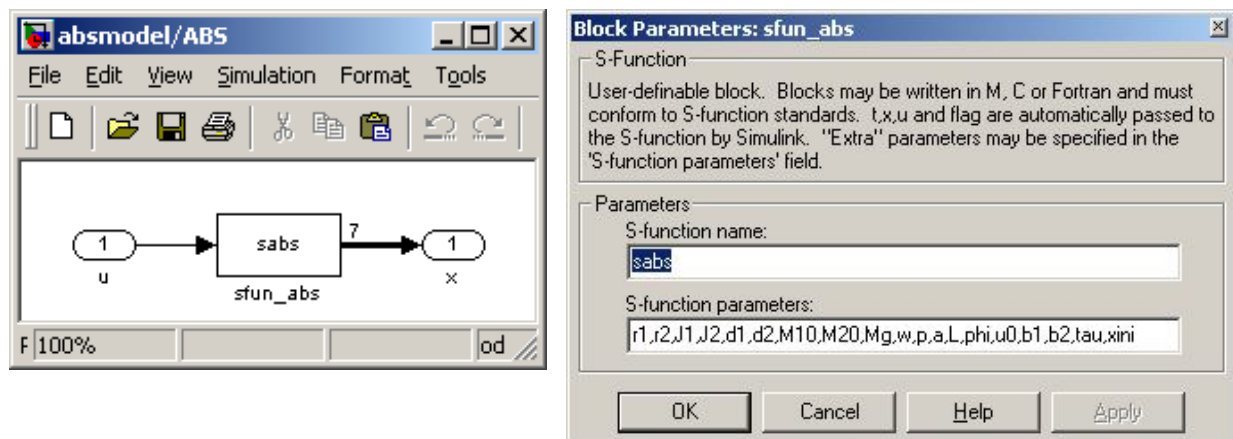


Fig. 7.2 ABS model as the S-function in Simulink

If one clicks on the green box the following mask appears (see Fig. 7.3).

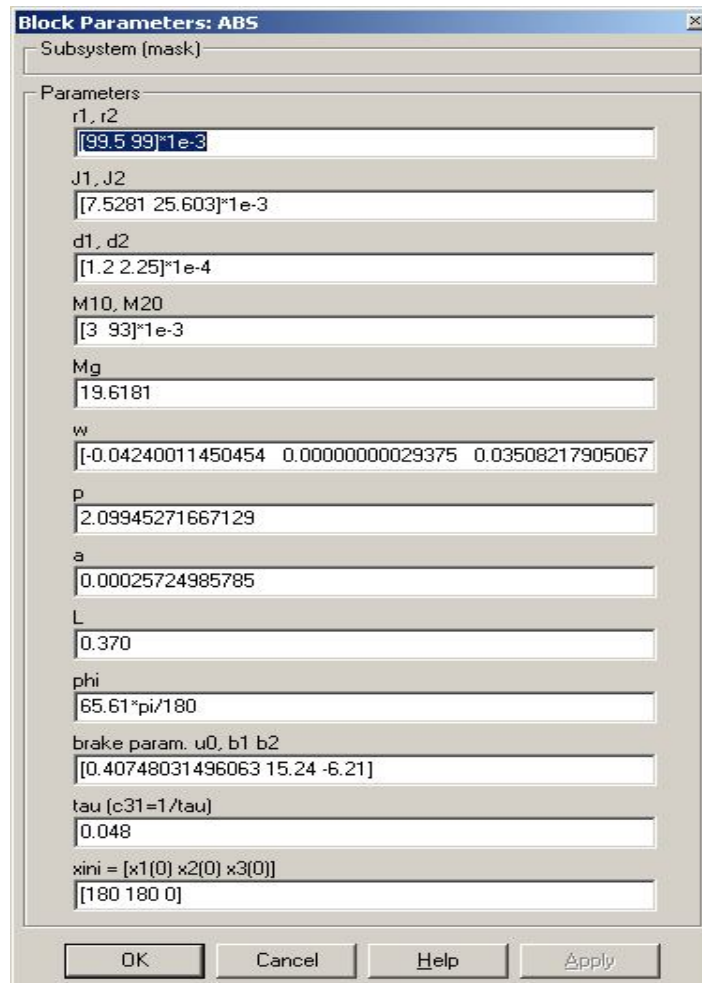


Fig. 7.3. Mask of the ABS model

We can perform the simulation of the model corresponding to the parameters given in the mask. The following graphical data are obtained in ABS scope and x_1 - x_2 plane (see Fig. 7.4).

The X Y Plot shows the state variables x_1 vs. x_2 . In the ABS scope one can see time responses of different variables related to the input brake control (the last diagram). We can verify the accuracy of the model investigating how close these responses are to the real system responses.

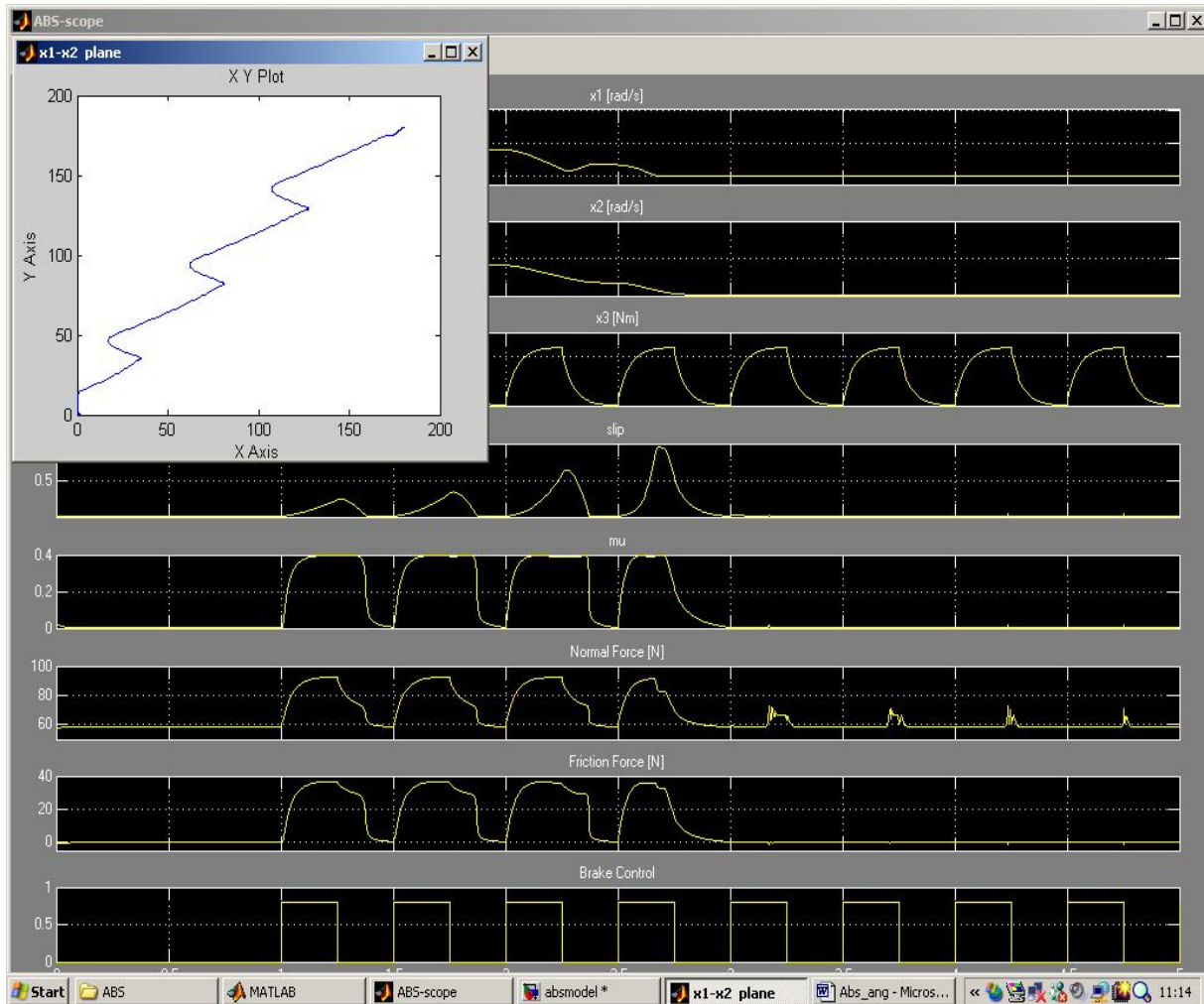


Fig. 7.4. Simulation results in the graphical presentation

Using the model given in Fig. 7.1 with a relay feedback we obtain simulation results. We perform simulation 1 identical to the experiment 1 (see Fig. 4.11). In this experiment the slip defined in the relay block is equal to 0.1. We simulate only the braking phase. Therefore the initial car velocity is assumed to be equal to 1800 rpm.

The zoomed braking phase of simulation 1 is given in Fig. 7.5. One can notice the similar braking time (approx. 1.5 seconds) in both: real-time experiment 1 (Fig. 4.11) and simulation 1 (Fig. 7.5). The experimental and simulated slips correspond also one to another.

Simulation 2 is performed for the slip defined in the relay block equal to 0.7. The zoomed braking phase of simulation 2 is given in Fig. 7.6. We can notice similarities between the real-time experiment 2 (see Fig. 4.13) and simulation 2 (see Fig. 7.6). As in the first example we simulate only the braking phase of control.

In fact a slip cannot be negative. This is guaranteed due to formula (6.4). However the slips in the real-time experiments calculated from the velocities obtain negative values. We must remember that the car and wheel velocities are not measured. They are only observed. Large velocity errors are introduced if the velocities are low.

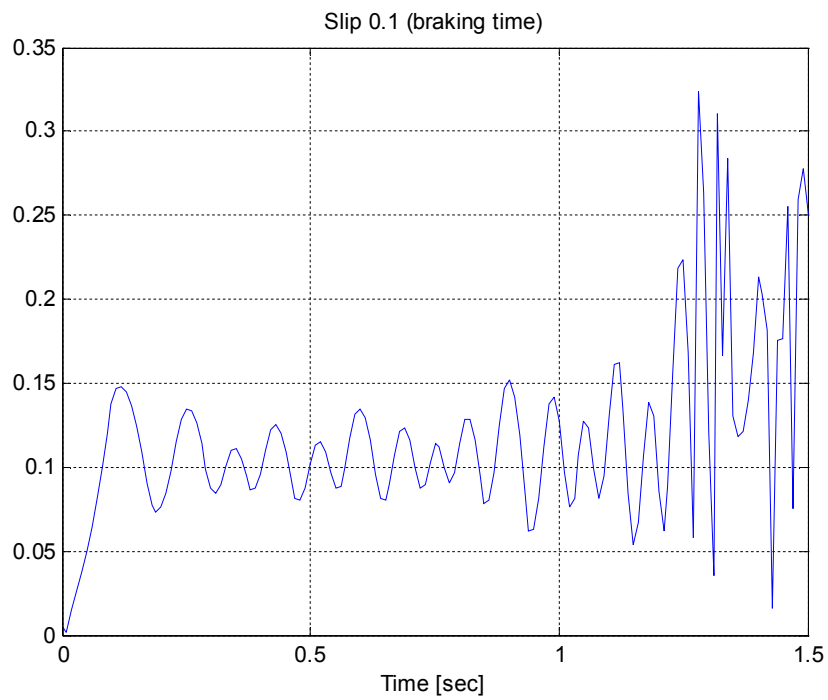
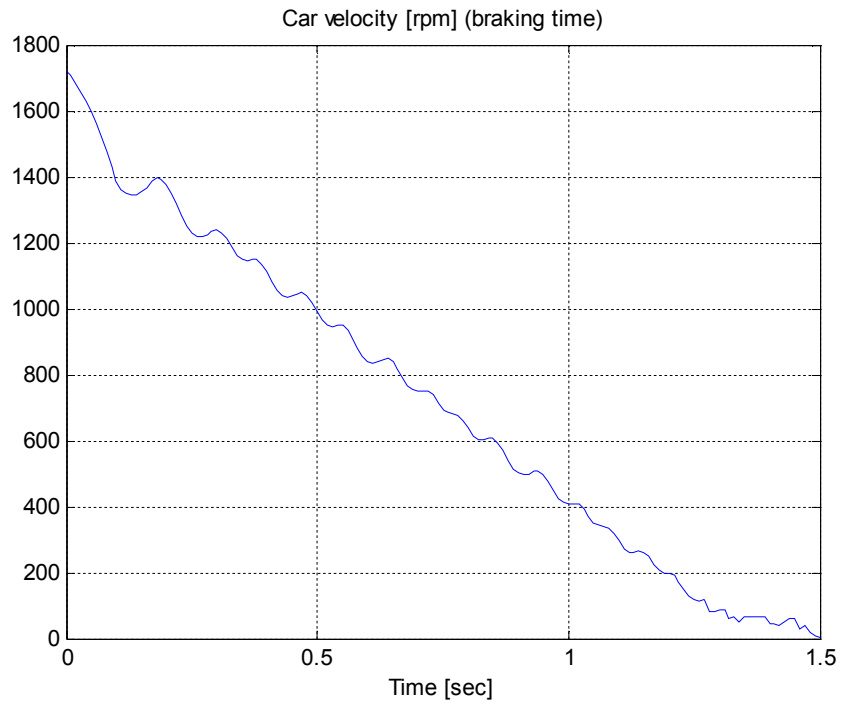


Fig. 7.5 Zoomed braking phase of simulation 1 (slip = 0.1)

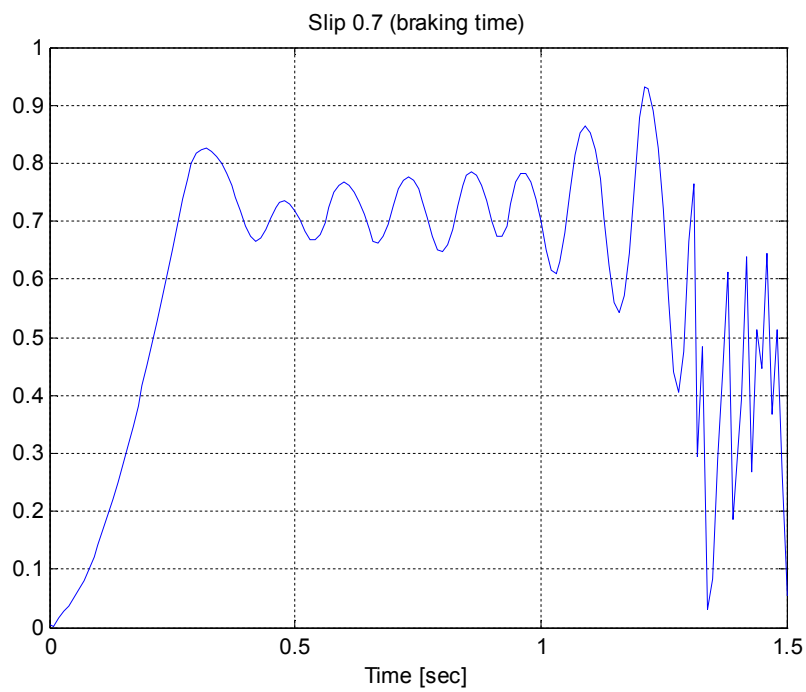
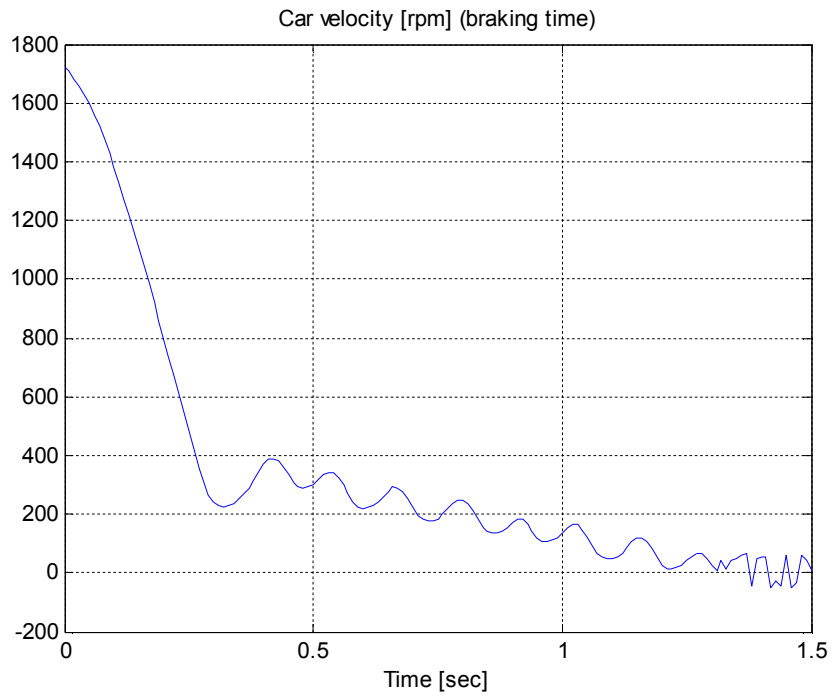


Fig. 7.6. Zoomed braking phase of simulation 2 (slip = 0.7)