



INTERNATIONAL UNION  
OF RAILWAYS

# Questions and Answers

## Tender “Traindy on web”

### Reference: RSF-Traindy-Web-2024

Date of publication of the present Questions and Answers document: 16/04/2024

#### Document history

Revision	Date	Description
1	05/04/2024	First publication, questions and answers from n°01 to n°04.
2	16/04/2024	Second publication, question and answer n°05.

#### Questions and answers – Tender “Traindy on web” reference RSF-Traindy-Web-2024

N°	Question	Answer
Q1	How many existing (and potential) users of TrainDy are there?	10 users
Q2	How many users are likely to be running calculations at the same time? This has important implications on the service hosting requirements, particularly without knowing how computationally demanding the calculations are at this stage?	3 users/at the same time
Q3	Could you provide us with a copy of the current Application now to help us understand its functions and computer performance requirements?	Consult the attached documents.

Q4	<p>You mention that that the “Application hosting is to be included by the Tenderer in its proposal.” Application hosting would be a recurring cost for the lifetime of the Application – what time period should be assumed in the tender?</p>	<p>One year of hosting is to be included in the breakdown of the price of the proposal. Hosting costs for the years following the first year are to be indicated separately in the proposal offer, as the hosting would indeed be a recurring cost for the lifetime of the Application. There is no specific time period for hosting after the end of the first year as the lifetime perspective of this project is long term.</p>
Q5	<p>Concerning the items "Maintenance contract with different service levels, price differentiation, and one-year included maintenance" and "Invoicing annually" of the tender document, is this an invoice for each user, or to UIC?</p>	<p>The invoicing will be to UIC.</p>

# TrainDy GUI

## Technical Guide

*Updated to 20/03/2009*

*Version: 1.0.5*

*This software has been developed by:*

*UniComp Informatica s.r.l*

**UNICOMP**

---

## Index

1. Overview.....	4
2. Appearance.....	4
2.1 Run Mode .....	4
2.1.1 Database Tree.....	4
2.1.2 Project Tree .....	4
2.1.3 Data-Entry Panel.....	5
2.2 Results Mode .....	5
2.2.1 Results Tree .....	5
2.2.2 Graph Data Panel.....	5
2.2.3 Export Data Panel .....	5
2.2.4 Graph Panels.....	5
2.2.5 Legend Panel.....	5
3. Packages and Classes.....	6
3.1 com.traindy.main .....	6
3.1.1 Main.....	6
3.1.2 TrainDy.....	6
3.2 com.traindy.beans.....	7
3.2.1 _TrainDyBean .....	7
3.2.2 _TrainDyField.....	7
3.2.3 other Classes.....	8
3.3 com.traindy.elements .....	8
3.3.1 _TrainDyElement .....	8
3.3.2 TrainDyElementsManager .....	9
3.3.3 other Classes.....	9
3.4 com.traindy.trees .....	10
3.4.1 DataBaseTree.....	10
3.4.2 ProjectTree .....	11
3.4.3 ResultTree.....	11

3.4.4	TrainDyTreeCellRenderer .....	12
3.4.5	TrainDyProjectTreeCellRenderer.....	12
3.5	com.traindy.tablemodels .....	12
3.5.1	_TrainDyTableModel .....	12
3.5.2	_TrainDyCellRenderer.....	13
3.5.3	SimpleTableModel.....	13
3.5.4	BlockFrictionLawsTableModel and BlockFrictionLawsCellRenderer .....	13
3.5.5	ConfigurationTableModel and ConfigurationCellRenderer .....	13
3.5.6	ManoeuvreTableModel and ManoeuvreCellRenderer .....	13
3.5.7	TrackTableModel .....	13
3.6	com.traindy.components .....	14
3.6.1	JIntegerTextField .....	14
3.6.2	JDoubleTextField .....	14
3.7	com.traindy.threads .....	14
3.7.1	CommandLineDaemonThread.....	14
3.7.2	MessageListenerDaemonThread.....	14
3.8	com.traindy.util .....	15
3.8.1	FileSystemUtility .....	15
3.8.2	GraphManager.....	15
3.8.3	LanguageManager .....	15
3.8.4	LanguageUtility .....	15
3.8.5	SwingUtility.....	16
3.8.6	GenericUtility.....	16
4.	Properties .....	17
4.1	_TrainDy.properties.....	17
4.2	_TrainDyLabels_<language code>.properties .....	17
4.3	_TrainDyMessages_<language code>.properties .....	18
4.4	<element name>_<language code>.properties .....	19
4.5	<element name><tab number>.properties.....	19
4.6	<element name><tab number>_<language code>.properties .....	20

# 1. Overview

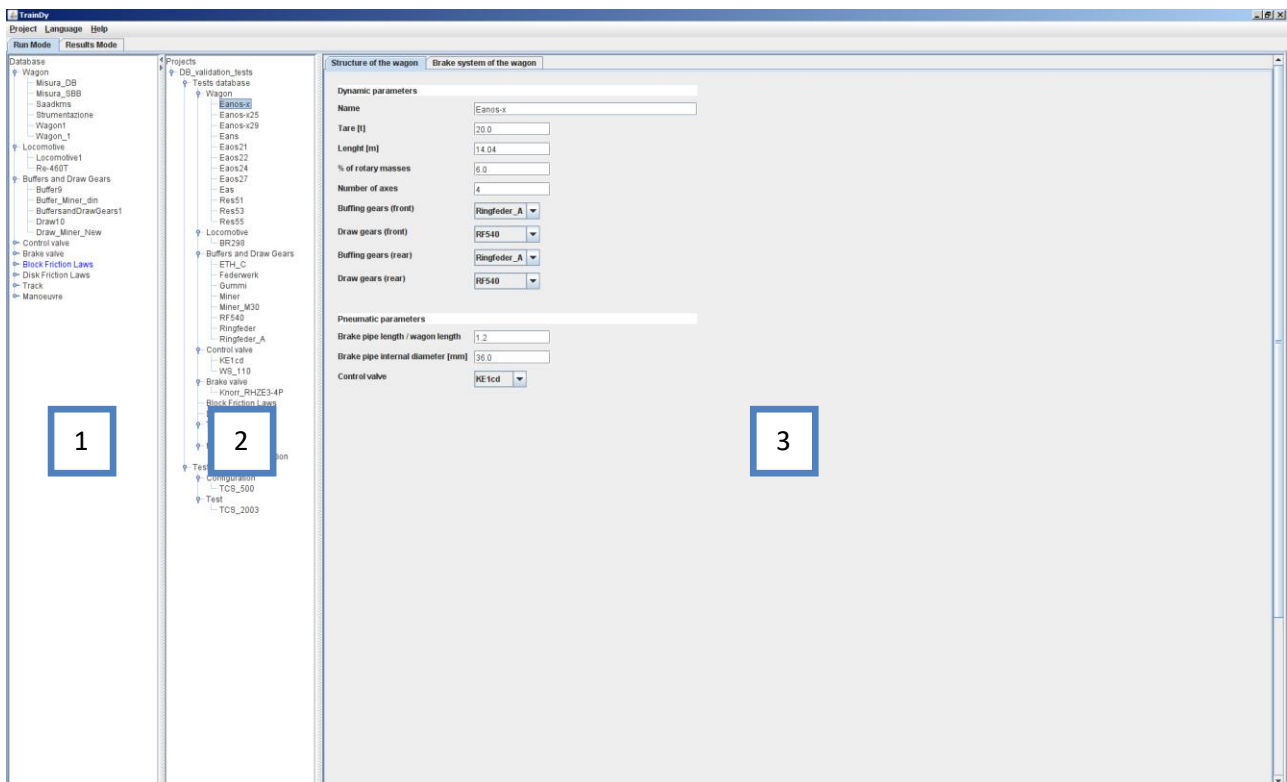
“TrainDyGUI” is a Multilanguage Graphical User Interface, realized in Java SE 6 using the Swing API.

The GUI has two main purposes:

- allows the user to easily manage all the information needed as input to the TrainDy Matlab application and to run simulations (Run Mode);
- show one or more charts related to a TrainDy Matlab simulation result, according to the user settings (Results Mode);

The “Results Mode” section, make use of the JFreeChart library (<http://www.jfree.org/jfreechart/>).

# 2. Appearance



## 2.1 Run Mode

### 2.1.1 Database Tree

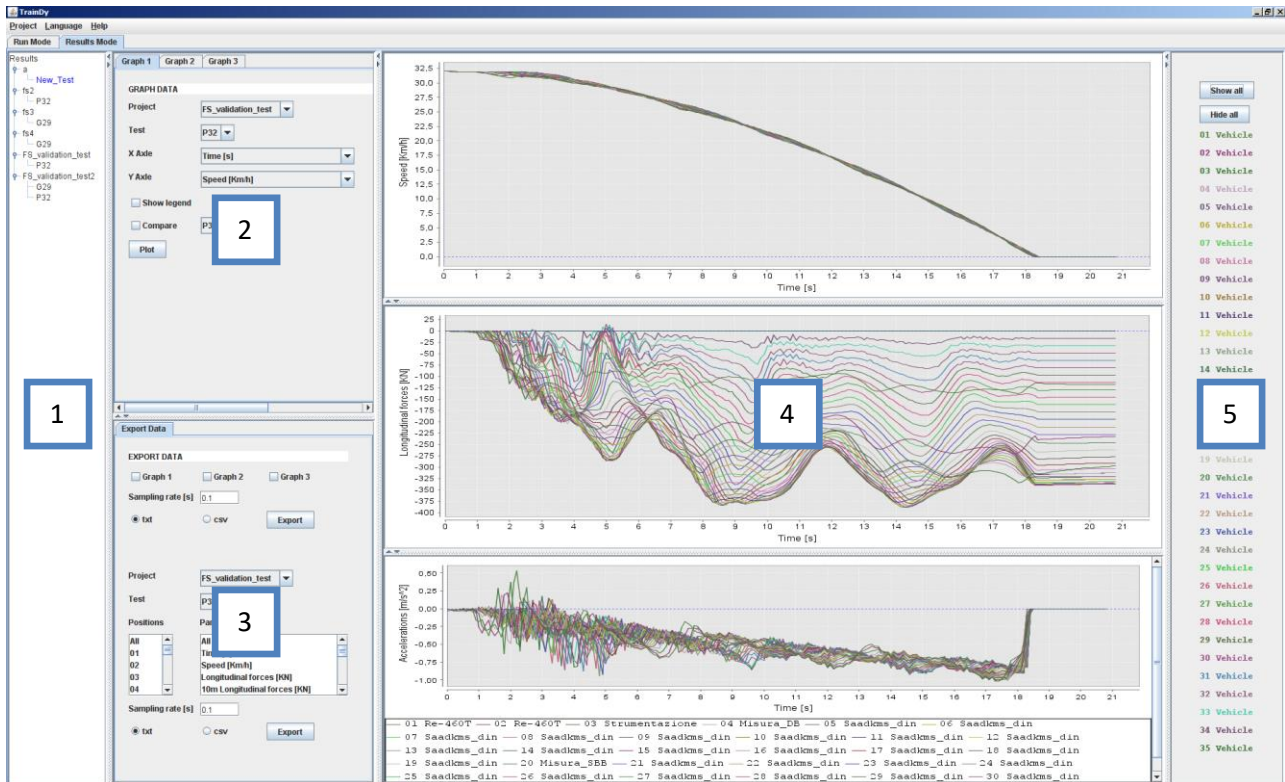
- Read-only menu tree showing pre-loaded database elements

### 2.1.2 Project Tree

- Menu tree showing current working projects;

## 2.1.3 Data-Entry Panel

- Main data-entry area: every click on an element of Database or Project Tree will load the corresponding labels, fields and values in this section, organized in one or more tabs;



## 2.2 Results Mode

### 2.2.1 Results Tree

- Read-only menu tree showing projects and tests that have simulation results;

### 2.2.2 Graph Data Panel

- Data selection section, manage the Graph Panels;

### 2.2.3 Export Data Panel

- Data selection source for exporting simulation values in text files;

### 2.2.4 Graph Panels

- Contain one or more charts, according to Graph Data Panel parameters;

### 2.2.5 Legend Panel

- Shows the list of elements represented in charts and allows to show or hide them (one by one or all); when only one chart is drawn, the list is composed by corresponding locomotive and wagon names, otherwise by repeating the generic word "Vehicle";

## 3. Packages and Classes

The TrainDyGUI application is composed by several packages, according to the meaning and the usage of the contained Java Classes. There are also several Properties files, used for base configuration parameters and for Multilanguage management.

### 3.1 *com.traindy.main*

The com.traindy.main package contains the Classes used during the start-up of the application.

#### 3.1.1 Main

- Is the only executable Class (a “public static void main” method is declared);
- Acquires launch arguments (default language, debug on/off) and draws the main JFrame window, that will contain the TrainDy instance;

#### 3.1.2 TrainDy

- Detect the running Operating System; gathered information will be used in the Matlab input files;
- Act as top-level objects container: it create an instance for every main JSplitPane, JScrollPane or JTree used in the application; some pane/tree-related management methods are also declared here, to allow calling from underlying objects;
- Set the font size, according to screen resolution and base values read from the “\_TrainDy.properties” configuration file;
- Set several String variables with the full path of working directories, according to the base path read from the “\_TrainDy.properties” configuration file;
- Create an instance of a private LanguageManager object and a private TrainDyElementManager object, declaring also some corresponding management methods: getters/setters, language switching, load/unload elements and so on;
- Set height and width of JSplitPanels and JScrollPane, according to percentage settings reported by the “\_TrainDy.properties” configuration file; set also some other layout parameters like orientation, expandability, etc.;
- Implements methods needed to change panes content, called after clicking on an item of the menu trees;
- Thru the ActionListener methods, manage the menu bar and the corresponding items (“New Project”, “Open Project”, “Change language” and so on);



## **3.2 *com.traindy.beans***

The `com.traindy.beans` package contains all the Classes that represent the inner components of data entry elements, in a Java bean-like standard. Every bean contains a series of private variables (data entry fields), with corresponding getter/setter methods, and a collection of `_TrainDyField` instances, that describe checks and translations, according to their properties file (more on this later). Classes are organized in one base Class named `_TrainDyBean`, and a series of derived Classes, one for each element/tab pair.

For example, in the `TrainDyGUI` the Locomotive data entry element is represented by two separate tabs, so two related bean Classes are defined: `Locomotive1Bean` and `Locomotive2Bean`.

### **3.2.1 `_TrainDyBean`**

- Is the base Class of every other `com.traindy.beans` Class, except `_TrainDyField`;
- Declares references to its graphic element container (`JComponent`) and to its logical element container (`_TrainDyElement`);
- Implements the interface `Cloneable`, to allow beans cloning during cut/copy/paste operations;
- Declares a collection of `_TrainDyField` objects, loading their characteristics (type, length, mandatory or not, value checks, label in various languages, etc.) from the corresponding Properties file; implements also collection's getter/setter methods and `_TrainDyField`'s wrapper getter/setter methods;
- Implements two generic methods to get or set bean properties thru reflection;
- Implements a method that checks every bean's property value, according to the information contained in the corresponding properties file;

### **3.2.2 `_TrainDyField`**

- Contains a reference to the `TrainDy` instance, to gain access to the main `LanguageManager` instance;
- Maps every information contained in a row (that means a single bean's field) of the bean configuration properties files, like `Locomotive1.properties`, `Wagon2.properties`, etc.;
- Implements several methods to retrieve the label of the field, that return different Strings formats depending on the usage of the label itself (HTML for error messages or plain text for column headers);
- Implements a generalized method to perform checks on the actual value of the field, according to rules described in the corresponding bean configuration properties file;

### 3.2.3 other Classes

- Several objects are declared, one for every key of the corresponding bean configuration properties file;
- In the constructor method, all the information about bean's fields are read from the corresponding properties file and stored in collections, for real-time usage;
- The constructor method is also the place where all the default values of bean's fields are set;
- Field variables are divided in two groups: labels and real fields; the main difference is in the declaration of getter/setter methods: real fields follow a get/set naming convention, labels a ret/put convention, so they are managed in a different way during object clone operations;

## 3.3 *com.traindy.elements*

The `com.traindy.elements` package contains all the Classes that represent the graphical part of a data entry element. The main base Class, `_TrainDyElement`, extends the swing Class `JPanel`, so every derived Class have a place to append its own `JLabel`, `JTextField`, `JComboBox`, etc.

There is also another base Class named `TrainDyElementManager`, that act as a collector for every instance of `_TrainDyElement` Object loaded in the `TrainDyGUI`, performing common operations such as load/save projects, data export to TrainDy Matlab file format, checks that involve more `_TrainDyElement` instances, etc.

### 3.3.1 `_TrainDyElement`

- Is the base Class of every other `com.traindy.elements` Class, except `TrainDyElementsManager`;
- Contains several methods to manage Tab names (including translations) and instances of `_TrainDyBean` derived objects that compose the `_TrainDyElement`;
- Store `_TrainDyElement` dedicated data, such as the name provided by the user for this instance, the index of the last visited tab, the currently displayed language, a flag to determine if the panel is in "read only" mode, etc.;
- Declare the abstract method "drawPane", that every derived class must implement by adding its own `java.swing` data entry Objects to the main `JPanel`;
- Declare base methods to find the Class type of a `java.swing` Object or its instance, given its name and a `Container` Object instance;
- Declare several generalized methods to store data entry value into corresponding `_TrainDyBean` variables;

- Implement the Cloneable interface, to clone instances during cut/copy/paste operations and PropertyChangeListener, FocusListener, ChangeListener interfaces, used to manage basic data entry features, such as real-time validation, fields enable/disable and so on;

### 3.3.2 TrainDyElementsManager

- The core of the TrainDyElementsManager class is a \_TrainDyElement collection, that contains every \_TrainDyElement instantiated at runtime; it is implemented by a nested HashMap, which keys are the Project name, the type of the element and the Element name;
- Every element of the nested HashMap can be stored, retrieved or manipulated by calling the corresponding public methods; basically are declared create/get/put/remove methods, but there are also other methods to rename elements, to translate them and to test their existence;
- Contains the implementation of methods dedicated to load a project, copy all the elements of a project (from file system or from a previously loaded project) to a new one, remove a Project Node from the Project Tree, delete a Project from the file system;
- Declare some methods to save projects, that mainly perform two tasks: they create, in separate directories, all the text files needed by the TrainDyGUI application (they map 1-to-1 all the fields contained in the \_TrainDyBean instances of a \_TrainDyElement) and all the text files needed by the TrainDy Matlab simulation software (created by a different data aggregation algorithm, according to the TrainDy Matlab software specification);

### 3.3.3 other Classes

- As stated in the com.traindy.beans package, several objects are declared, one for every node contained in the Project Tree section of the \_TrainDyLabels configuration properties file;
- Element Classes represent the graphical aspect of a data entry element, by organizing all the information related to the layout of its data entry fields;
- All the data entry fields are contained in one or more JTabbedPane objects, according to their properties configuration files;
- Every element Class extends the \_TrainDyElement base Class, inheriting its common methods;
- Elements that have data entry tables and/or “Advanced options” buttons in their graphical layout, have to declare corresponding objects as instance variables;
-

- The “setAllElements” method is an overload of the corresponding superclass method: every element Class must implement it by defining the structure of data entry field names, according to its properties configuration files;
- The “drawPane” method is the main element Class method: it performs a loop on every element’s JTabbedPane, invoking the “makePanel” method to create its content;
- The “makePanel” method builds a JPanel object by invoking SwingUtility methods to add all data entry fields as defined in the corresponding properties configuration file;
- Other methods perform custom management of JTable objects, such as implementation of “Add row” / “Remove row” functionalities, and “Advanced Options” show/hide fields toggle;

### **3.4 *com.traindy.trees***

The com.traindy.trees package contains all the Classes involved in the Menu tree management. There are three separate main Classes, each of them contains an instance of a JTree object, to graphically represent menu data. In this package are also contained a couple of Classes that extends the DefaultTreeCellRenderer, because TrainDy trees nodes need some custom management.

#### **3.4.1 DataBaseTree**

- Represents a read-only tree: contained data are loaded from the default DataBase directory at the startup of the TrainDyGUI and do not change until the next reload;
- Declare an HashMap instance, that contains every element loaded in the tree, as well as all its related management methods (get, put, exists, translate, etc.);
- Left-clicking on a DataBaseTree leaf node, will load corresponding element data in read-only mode inside the Data Entry Panel; this is realized by implementing the PropertyChangeListener interface;
- Right-clicking on a DataBaseTree leaf node, will show a context menu containing the “Copy” feature, that allow the user to select nodes that have to be copied in the ProjectTree; this is realized by implementing the ActionListener interface;
- The “Copy” feature is implemented by declaring an HashMap that will contain the user selected elements; the ProjectTree will perform reading and cloning operations of the elements contained in this HashMap during the “Paste” feature;
- Perform a custom management during the “Locomotive” or “Wagon” copy feature, by copying also each related sub-element;

### 3.4.2 ProjectTree

- Is the main menu tree: contains all the working projects data, loaded by the user thru the “Open Project” menu bar feature; projects may also be copied, closed or deleted thru corresponding features;
- Declare an HashMap instance, that contains every element loaded in the tree, as well as all its related management methods (get, put, exists, translate, etc.);
- Left-clicking on a ProjectTree leaf node, will load corresponding element data in update mode inside the Data Entry Panel; this is realized by implementing the PropertyChangeListener interface;
- Right-clicking on a ProjectTree leaf node, will show a context menu containing the “Cut” and “Copy” features, that allow the user to perform common node management operations; this is realized by implementing the ActionListener interface;
- The “Cut” and “Copy” features are implemented by declaring an HashMap that will contain the user selected elements; during the “Paste” feature, will be performed reading and cloning operations of the elements contained in this HashMap;
- Perform a custom management during the “Locomotive” or “Wagon” copy feature, by copying also each related sub-element;
- Right-clicking on a element node, will show a context menu containing the “New” and “Paste” features, that will allow the user to create new elements or paste previous copied ones;
- Right-clicking on a project node, will show a context menu containing the “Save”, “Copy”, “Close” and “Delete” project features, that allow the user to perform common operations on a loaded project;
- Double-clicking on a ProjectTree leaf node allow the user to rename it, according to some naming convention rules (no blank spaces, no special characters other than minus and underscore sign, no reserved names, etc.);
- The renaming of a node is implemented by several custom methods, needed to perform naming convention checks;
- Every node change (name or data) reports same effects on the HashMap managed by the TrainDyElementsManager Class;

### 3.4.3 ResultTree

- Is the only menu tree of the Results Mode: contains references to all the TrainDy Matlab simulation results, in read-only mode;

- The features of the ResultTree were significantly reduced since the beginning of the project, so left-clicking or double-clicking on nodes has no visible effect other than expanding or collapsing the menu tree;
- Right-clicking on a ResultTree leaf node, will show a context menu containing the “Delete Result” feature, that allow the user to delete the Matlab simulation output files from disk;

#### **3.4.4 TrainDyTreeCellRenderer**

- Extends DefaultTreeCellRenderer and it is applied in the DataBaseTree and in the ResultTree;
- Implements some “mouse-over” graphical features, like colour change on the active node;

#### **3.4.5 TrainDyProjectTreeCellRenderer**

- Extends DefaultTreeCellRenderer and it is applied in the ProjectTree;
- Implements some “mouse-over” graphical features, like colour change on the active node, as well as some custom management on the node name editing;

### ***3.5 com.traindy.tablemodels***

The com.traindy.tablemodels package contains all the Classes extending the DefaultTableModel java swing object. They are used to graphically render all the data tables of the GUI application and to manage some basic functions like adding/removing rows, enabling/disabling cells and so on. There is one generalized class (SimpleTableModel), used for simple tables, and several Element-coupled classes, to handle more complex situations; all of them extends the \_TrainDyTableModel class, that implements several common functionalities.

#### **3.5.1 \_TrainDyTableModel**

- Implements a base structure and several methods that represent a common base for every other TableModel declared in this package;
- Some methods offer base functionalities on table rows, like inserting, removing and counting;
- Some methods set or retrieve the “editable” cell property, used to enable or disable cell updating;
- The set of a table cell value is also implemented by corresponding methods, that also take care of some cases that require custom management (Friction laws, etc.);

### **3.5.2 \_TrainDyCellRenderer**

- Set some default table cell colour values, used in all TrainDy data tables, such as enabled/disabled background colour or special cell colour;
- Overrides the getTableCellRendererComponent method, by adding some custom management for cells containing Boolean or numeric Double values;
- Set the colour of a table cell, according to its “editable” property;

### **3.5.3 SimpleTableModel**

- Represents the standard data table in the TrainDy GUI application, it is used in most cases;
- Has a default minimum row count of 2 and no custom managements;

### **3.5.4 BlockFrictionLawsTableModel and BlockFrictionLawsCellRenderer**

- Implement the custom management of the Block Friction Laws element data table;
- Has a default minimum row count of 2, first row and first column are coloured as special cells;

### **3.5.5 ConfigurationTableModel and ConfigurationCellRenderer**

- Implement the custom management of the Configuration element data table;
- Has a default minimum row count of 20, several columns are coloured as special cells;
- Row values are copied from “Locomotive” or “Wagon” elements, according to the selection of the element’s name in the first cell;
- When adding new elements or editing mass/length values, methods “calcTotMass” and “calcTotLen” automatically perform a calculation of the total mass/length;

### **3.5.6 ManoeuvreTableModel and ManoeuvreCellRenderer**

- Implement the custom management of the Manoeuvre element data table;
- Has a default minimum row count of 1;
- For every row, cell editing is enabled/disabled according to the selection of the checkboxes in the row itself;

### **3.5.7 TrackTableModel**

- Implement the custom management of the Track element data table;
- Has a default minimum row count of 1;

- For every row, the “Curvature Radium” cell editing is enabled/disabled according to the value of the “Section Type” cell;

### **3.6 *com.traindy.components***

The `com.traindy.components` package contains some TrainDy-custom data entry objects, derived from the Java Swing `JTextField` object. They implements some basic formal checks that are common to all similar data entry fields.

#### **3.6.1 `JIntegerTextField`**

- Is an improved version of the `JTextField`, used to acquire integer numeric data values;
- After the user sets a value, some checks are performed to insure that the user typed a correct integer value; if correct, the value is definitively set, otherwise, the cell content is set with the previous value;

#### **3.6.2 `JDoubleTextField`**

- Is an improved version of the `JTextField`, used to acquire double-precision numeric data values;
- After the user sets a value, some checks are performed to insure that the user typed a correct double-precision value; if correct, the value is definitively set, otherwise, the cell content is set with the previous value;

### **3.7 *com.traindy.threads***

The `com.traindy.threads` package contains all the Classes involved in managing external system calls in a multi-thread environment. In that way, the user may go on working with the GUI while other tasks are running. The TrainDy GUI performs system calls to start the Matlab simulation and to retrieve results data from the Matlab simulation.

#### **3.7.1 `CommandLineDaemonThread`**

- Implements a system call with the command line contained in the “command” parameter;
- Store the thread state (text message) in the `GenericUtility` Class; only one thread at time is allowed;

#### **3.7.2 `MessageListenerDaemonThread`**

- Wait until the end of a system call launched by a `CommandLineDaemonThread` object, by periodically checking the text message stored in the `GenericUtility` Class;



- If the system call ends successfully, updates the Results Tree object;

### **3.8 *com.traindy.util***

The `com.traindy.util` package contains several utility Classes used in the TrainDy GUI project. Every method that performs common actions or base actions is defined inside an utility Class of this package, to easily allow their reuse.

#### **3.8.1 FileSystemUtility**

- Contains all the methods involved in file system management;
- Some methods perform common operations on directories, such as creating/deleting, reading all files names in a String Array, copying all the directory content in a new one;
- Other methods perform file reading, writing and deleting operations;

#### **3.8.2 GraphManager**

- Declares some methods used as interface between the TrainDy GUI data structure and the JFreeChart library;
- Methods of this class are mainly used in the Results Mode, where, for example, the GUI needs to retrieve wagons or parameters lists to perform system calls and read Matlab simulation results;
- The main method is the “createResultsChart”, that perform all the operations needed to draw the chart in the ChartPanel: it retrieves and aggregates data in the JFreeChart structures, dynamically create the content of the legend box and finally calls JFreeChart APIs;

#### **3.8.3 LanguageManager**

- Implements all methods needed to show the TrainDyGUI labels in the proper user language;
- Labels and error messages are loaded during the start-up phase and stored into internal structures, to ease the retrieve of information;
- When the user choose to change the GUI language at runtime, current LanguageManger instance is replaced by a new one, that will contain labels and error messages of the chosen language;

#### **3.8.4 LanguageUtility**

- In the LanguageUtility are defined some language-related constants, such as language internal code and language internal descriptions (short and long);

- This Class is mainly used by the LanguageManager;

### **3.8.5 SwingUtility**

- Collect all Java Swing-related methods, used during the graphical layout creation process;
- Several methods contain the common set of instructions needed to create a simple Java Swing data field and to assign its value, such as “createLabel”, “createTextField”, “createButton” and so on;
- Other methods contain the common set of instructions needed to create an instance of a TrainDyGUI data entry table (see com.traindy.tablemodels package), by defining their row count, column sizes, cell starting values, etc.;
- Some methods are related to TrainDyGUI JTree objects (see com.traindy.trees package), they are mainly used to find a node in a specific JTree, to retrieve a JTree branch, etc.;
- The graphical layout creation process uses some SwingUtility specific methods whose name starts with the “add” prefix (“addHeaderRow”, “addParameterRow”, “addEmptyRow”, etc.); they perform a sequence of “createXXX” methods, to create objects according to the purpose of the method itself;

### **3.8.6 GenericUtility**

- Contains all the generic-purpose methods, that do not fit into another specific utility class;
- Several methods involve array management, such as “isIn”, that look for a specific element into an array of elements of the same type, “arrayJoin”, that join two arrays in a third one, “mySplit”, that splits a constant-separated string into an array, etc.
- Some methods define the rules used during number formatting, providing specific functionalities such as rounding and so on;

## 4. Properties

Properties files are text files containing a list of key/value pairs, expressed in the following format:

```
<key> = <value>
```

In the TrainDy GUI application, you never have to add or remove keys in any of the properties files, but you may change their values for your own purpose.

Some properties file names contains the “language code” value; in these cases, files are duplicated for every TrainDy GUI managed language (language codes are defined in the LanguageUtility Class, variable LANG\_SHORT\_DESC), so the application may load them when needed, according to user’s language choice. Every value of these files must be translated in the corresponding language, or leaved in English language by default.

Due to the fact that every key can have only one value, a separator is needed for values representing arrays, so the value can be split by the software that process it.

In the TrainDy GUI application, the default separator are “;” (semicolon) for values containing coded data or “;\_” (underscore, semicolon, underscore) for values containing normal text.

E.g.

```
name = 30;Y;S
GUI_MAIN_TABS = Run Mode;_Results Mode
```

### 4.1 *\_TrainDy.properties*

The *\_TrainDy.properties* is the main configuration file. It contains the global TrainDy GUI configuration settings, like working directory names and paths, screen resolution, default JPanel dimensions, etc.

Its structure is divided into the following sections:

- Standard parameters
  - Contains several properties related to directory/element names; these are reserved names, so there is no need to modify them until the next major version change;
- Custom parameters
  - Contains information typically modified during the TrainDy GUI setup process; you may modify values if you manually move the application without using the installation software;
- Default panel dimension
  - The default dimension of every TrainDy GUI panel is customizable by changing these values; basically, you have to express the width and height of the panel in percent of the total screen width/height (see the file content to find more details);

### 4.2 *\_TrainDyLabels\_<language code>.properties*

(e.g. *\_TrainDyLabels\_en.properties*)

The `_TrainDyLabels` properties file contains all the labels used in the main TrainDy GUI Application, independently from elements, that have their own label-related properties file.

In this file you will find every label used in:

- Menu Bar (Project, Language, Help and their sub-elements)
- Base elements of menu trees (Project Tree, Database Tree, Results Tree)
  - Please notice that key names of this section start with a fixed prefix followed by an incremental number; if you plan to add the management of new elements, you have to add new keys in this section, according to that naming convention;
- Button labels (e.g. Add/Remove row, Advanced Options, Run, etc.)
- Value lists (e.g. brake regime, shoes type, etc.)
  - Please notice that value lists key names start with the dollar sign prefix (“\$”), so the application can easily identify them;
- Graph parameters, used in the Results Mode
  - As stated in the file itself, only the `$GP_LABELS` property value may be translated; `$GP_NAMES` and `$GP_FILES` must be leaved “as is”

### ***4.3 `_TrainDyMessages_<language code>.properties`***

(e.g. `_TrainDyMessages_en.properties`)

The `_TrainDyMessages` properties file contains all the messages that the TrainDy GUI application show to the user. They are divided in four types, according to the purpose and severity of the message:

- Action, when the message is a question that typically require an answer or intervention by the user;
- Information, generic information message, such as release version and so on;
- Warning, non-blocking error messages;
- Error, blocking error messages;

Every key in this file is composed according to the following rules:

```
<Message type><Section number><Message number>[_<Incremental number>]
```

where:

- “Message type” is the starting letter of the message type: A, I, W or E;
- “Section number” represents the scope of the message (e.g. GraphData, LegendPanel, Test, etc.);
- “Message number” is the unique number of the message, inside its section;
- (Optional) “Incremental number” is used when the message has to be split in more than one sentence (e.g. “Project”<project name>”loaded”)

The corresponding value is the label of the message, translated in the proper language

#### **4.4 *<element name>\_<language code>.properties***

(e.g. Wagon\_en.properties)

The element base properties files contains the list of its tab (JTabbedPane Java Swing Object) labels. Every key in this file is composed according to the following rules:

TAB\_<Incremental number>

where:

- “Incremental number” is the number representing the order of the tab when it has to be shown in the TrainDy GUI;

The corresponding value is the title of the tab, translated in the proper language

#### **4.5 *<element name><tab number>.properties***

(e.g. Wagon1.properties)

The element numbered properties files contains the list of data entry fields composing the tab whose “tab number” is referring to. The key represents the data entry field internal name, while the value is formed by a series of seven codes separated with the semicolon:

<Field size>;<Mandatory>;<Data Type>;<Operator>;<Operand1>;<Operand2>;<Val.List>

Each code contribute to provide all the information needed to dynamically create the corresponding data entry field at runtime. If a code is not necessary, may be left empty.

Codes are compiled according to the following rules:

- “Field size”, is the width of the data field, expressed in number of characters (not 100% accurate, strongly font-dependent);
- “Mandatory”, to determine if the data entry field is mandatory or not, must start with the capital “Y” (yes) or “N” (no); if the letter is followed by a data entry field name inside square brackets, it means that the field will become mandatory only if the “depending” field inside square brackets is compiled, otherwise it is not required;
- “Data Type”, represents the data entry field data type; may assume the following values:
  - “S”, string, plain text
  - “I”, numeric integer
  - “D”, numeric decimal, with double precision

- “B”, Boolean variable, yes/no, true/false

If the letter is followed by a number inside square brackets, it means that the data entry field is a table data column, and that the number is the number of mandatory rows that the user have to fulfil for that table.

- “Operator”, it is used only for numeric data types to check the correctness of their value; may assume the following values:
  - “GT”, greater than
  - “GE”, greater or equal than
  - “LT”, lesser than
  - “LE”, lesser or equal than
  - “BW”, between (bounds included, two operands required)
- “Operand1”, “Operand2”, numbers used when “Operator” is compiled, they represent the term of comparison between the data entry field value, ruled by the operator.
- “Values List”, if compiled, means that the data entry field may only assume a value that is in that list; it may be:
  - An element internal name (e.g. BuffersDrawGears), the list will be built at runtime using all the items of that element that are loaded in the same project;
  - Two element internal names, separated by a “+” sign (e.g. BlockFrictionLaws+DiskFrictionLaws), the list will be built at runtime using all the items of that two elements that are loaded in the same project;
  - A “value list” key defined in the `_TrainDyLabels_<language code>.properties` configuration file (e.g. \$SHOES\_TYPE), the list will be built performing a split of the value of that key;

#### **4.6 <element name><tab number>\_<language code>.properties**

(e.g. Wagon1\_en.properties)

The element numbered and language-coded properties files contains the list of data entry fields composing the tab whose “tab number” is referring to. The key represents the data entry field internal name, while the corresponding value is the name shown in the TrainDy GUI, translated in the proper language.




# User Guide

**This software has been developed by:**

*Faiveley Transport Italia* 

**In collaboration with:**

*University of Rome "Tor Vergata"* 

Updated to 04/04/2024

Version: 1.4.6

# INDEX

<b>INDEX</b>	<b>2</b>
<b>INSTALLATION</b>	<b>4</b>
<b>NEW VERSION CHECKING</b>	<b>5</b>
<b>WHAT'S NEW</b>	<b>7</b>
<b>SOFTWARE GUIDE</b>	<b>8</b>
<b>RUN MODE</b>	<b>8</b>
VEHICLE	10
BUFFERS AND DRAW GEARS	25
CONTROL VALVE	28
BRAKE VALVE	38
BLOCK FRICTION LAW	41
DISK FRICTION LAW	42
TRACK	43
MANOEUVRE	44
CONFIGURATION	46
TEST	50
<b>RESULT MODE</b>	<b>51</b>
AVAILABLE GRAPHS	52
EXPORT DATA	53
<b>TECHNICAL GUIDE</b>	<b>54</b>
<b>PNEUMATIC MODEL OF THE TRAIN BRAKE</b>	<b>54</b>
BRAKE PIPE	55
MASTER BRAKE VALVE	56
CONTROL VALVE	58



ACCELERATION CHAMBERS	60
AUXILIARY RESERVOIR	61
<b>BRAKE SYSTEM MODELS</b>	<b>62</b>
COMPUTATION OF BRAKED FORCES IN A BRAKE BLOCK SYSTEM AND COMPARISON WITH UIC 544-1	65
BRAKE CONTROLS	69
FRICTION LAWS	70
<b>BUFFERS AND DRAW GEARS IMPLEMENTATION</b>	<b>78</b>

# INSTALLATION

*TrainDy* is a *portable* software, i.e., it does not need installation, just copy the package provided by UIC in a folder of your computer.

In order to run/execute *TrainDy* for the first time, two steps must be performed:

1. Upgrade the java machine with the latest version (JDK 11 or above)
2. Install the MCRInstaller.exe corresponding to the Matlab version used to create the *TrainDy.exe* (currently the R2022b)

When a new version of *TrainDy* is released, it is likely that a new MCRInstaller.exe must be installed. Mathworks provides MCRInstaller.exe for different Matlab releases on the internet.

To launch *TrainDy*, double click on *TrainDyGui.bat*.

## NEW VERSION CHECKING

In order to check the consistency of the new version with the certified results, two tools working from MATLAB Command Window are made available: `automatic_version_verification` and `analisi_risultati_GUI`. These two files must be in the same folder of the file `elenco_TDED.txt`.

The first tool performs the computations using both the old TrainDy input (stored in the Results folder) and the new TrainDy input (stored in the Projects folder); the second tool automatically updates the file `risultati.xls` where the TrainDy results are compared with the experimental results.

Pre-requisites for the tools are that the folder `Projects`, `DB validation tests`, `FS validation tests` and `SNCF validation tests` must be in the same folder. Moreover, must be in the same folder also the files: `TrainDyToGUI.exe`, `tRTUf2011_0.exe` and `tRTUf2011_1.exe`. Lastly, the first row of the file `elenco_TDET.txt` must be customized according to the User folder structure.

In order to generate the mat file (i.e. the file containing the results of the simulation) by employing the new version and by using the old inputs (i.e. the inputs stored in the folders `DB validation tests`, ...) and the new inputs (i.e. the inputs stored in the `Projects` folder), the User has to run in the MATLAB Command Window the file `automatic_version_verification`, by simply typing it and pressing return. This will generate the mat files finishing by "one" if they are computed using the new inputs (stored in the `Projects` folder), otherwise without any particular label if they are computed using the old input structure: this allows the User to check if there are some differences. This tool will also generate two figures displaying the time evolution of longitudinal forces computed by using the old input and the new input so that the User can interactively check the differences.

More interesting is to launch, in a manner similar to before the file `analisi_risultati_GUI`: this will update the files `risultati.xls` in the subfolders of `DB validation tests`, ... where the User will find also the file `copia di risultati.xls`. The former contains a comparison among the newer simulation results and

the experimental results. The latter contains the counterpart but referred to the previously validated *TrainDy* version.

## WHAT'S NEW

The new version of *TrainDy* corrects some bugs and modifies some inputs of reference projects. The accuracy of the results has been checked comparing the previous results with the current results. Moreover, this version introduces some enhancements:

- A new model of friction coefficient to handle LL brake blocks
- A new braking model for wagons to make the brake cylinder pressure behaviour dependent on the wagon mass. This model is advantageous when the wagons employ K blocks, but it is not limited to this application.

# SOFTWARE GUIDE

The TrainDy graphical user interface (G.U.I.) is divided in two sections: Run Mode and Results Mode. In the Run Mode, the User can define all the inputs for the specific simulation (train configuration, manoeuvre, track, etc...) and then the User can run the simulation by simply pressing the run button (the executable file "TrainDy.exe" is launched). In the Result Mode, the User can view the results dealing with the already performed simulations.

In order to perform a simulation, a train, an operational rule for each locomotive and a track must be defined; the results of the simulations are stored in .mat files in the corresponding Results folder. The GUI extracts all the data the User can display by an executable file called "DisplyAscii.exe", more details about it are in the section "Result Mode".

The software has four main menus: Project, Language, Extra and Help. The Extra menu allows the user launching the FDyn application to update the friction characteristics of a wagon in agreement with the stopping distances of the UIC 544-1: only cast iron shoes are currently supported.

## RUN MODE

The section Run Mode is characterized by two columns on the left and a main window on the right. In the first column there is the general database, from which the User can copy the elements into the defining projects on the second column. Each project is divided in two sections: Test database and Test definition as showed in Fig. 1, on the right

The first section contains all the elements used to define the train configurations, the tracks and the manoeuvres. The second section of the project (Test definition) contains all the data regarding the train configuration along with the manoeuvre performed by each locomotive, the Track used in the simulation and the test general data (starting speed and initial pressure in brake pipe).

By clicking with the mouse right button on a project title (see Fig. 1 on the left), the User can save, copy , close or delete the project, more options about projects are in the menu Project (see Fig. 1

on the centre) on the top of the GUI in which you can create a new project, copy a project, open a saved project and save all the projects.

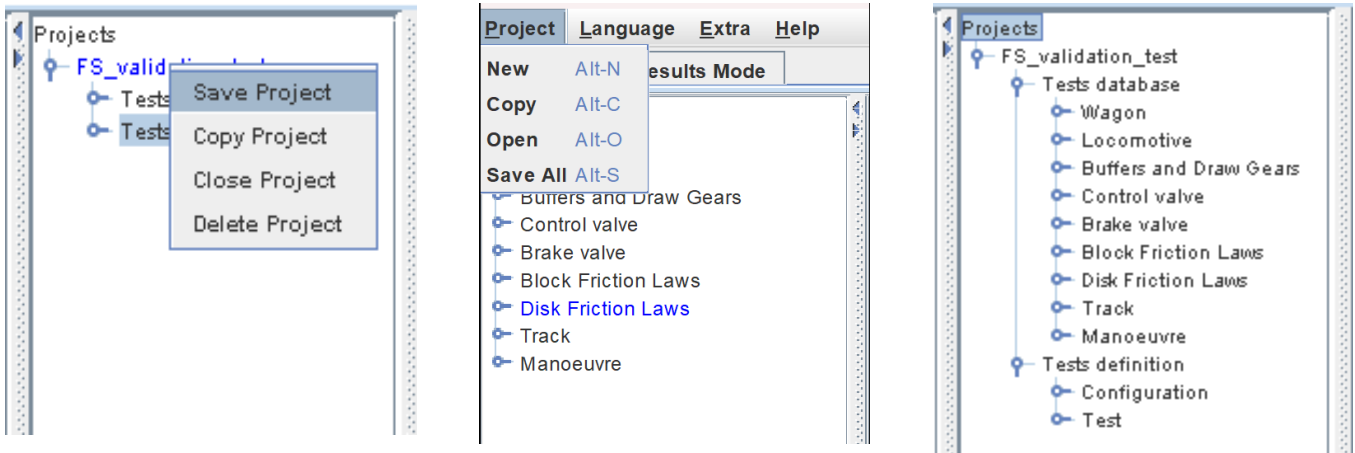


Fig. 1 Project options

In order to run a simulation, a specific test must be defined (see the last element of the tree on the right part of Fig. 1). By the test window (see Fig. 2), the User can choose the train configuration, the track, the starting speed and the initial pressure of the air in the brake pipe. By clicking on the Run button the simulation is performed (see picture below).

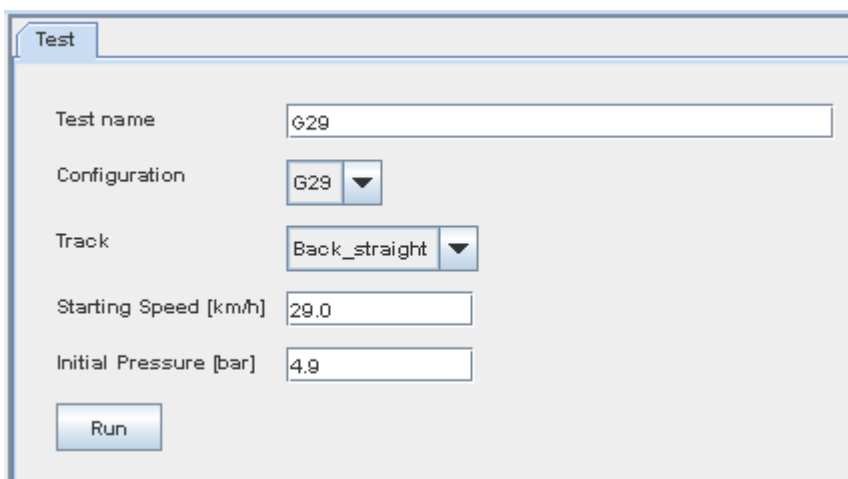


Fig. 2 Test definition window

When the Run button is pressed the GUI performs three steps:

- The GUI checks the input data and if there are some incompatibilities it returns an error window.
- The GUI writes in the **results folder** the files used as input by TrainDy.exe.
- The GUI launches TrainDy.exe file. The results of the simulation are stored in a .mat file with the name of the test in the corresponding folder “Results”.

Below are described the elements of the tree “tests database”.

## VEHICLE

The parameters used to define a wagon (or locomotive) are divided in two parts, the first part contains the “**Structure of the wagon (or locomotive)**” and the second the “**Brake system of the wagon (or locomotive)**” (More details about the procedure used for the braking force computation are in the section “BRAKE SYSTEM MODELS”).

Below the parameters used in these two sections are described:

- **Structure of the wagon (or locomotive)** (data about the general layout of the wagon)
  - **Dynamic parameters** (Fig. 3)
    - *Name*: Name of the wagon (or locomotive)
    - *Tare (mass) [t]*: Tare of the wagon (or locomotive)
    - *Length [m]*: Distance between buffers
    - *% of rotary masses*: This parameter is referred to the tare, it puts a surplus of mass in the model in way to consider the effects of the rotary inertia.
    - *Number of axles*: It contains the number of axles of the wagon (or locomotive): this parameters is not actually directly used by TrainDy, but in the future it will be.
    - *Concentrate pressure loss factor of hose couplings (only locomotive)*: The locomotive on front and rear is characterized by an hose coupling with a specific loss factor (the loss factors for the other vehicle are defined in the general data of the configuration window)
    - *Buffing gears (front)*: Model of buffers mounted in front of the wagon see Fig. 3



- *Draw gear (front)*: Model of draw gear mounted in front of the wagon see Fig. 3
- *Buffing gears (rear)*: As above, but referred to rear see Fig. 3
- *Draw gear (rear)*: As above, but referred to rear see Fig. 3

Dynamic parameters	
Name	Misura_DB
Tare [t]	68.0
Length [m]	22.0
% of rotary masses	4.0
Number of axes	4
Buffing gears (front)	Buffer_Miner_din ▼
Draw gears (front)	Draw_Miner_New ▼
Buffing gears (rear)	Buffer_Miner_din ▼
Draw gears (rear)	Draw_Miner_New ▼

Fig. 3 Wagon dynamic parameters

#### ➤ Pneumatic parameters

- *Brake pipe length / wagon (or locomotive) length*: This ratio defines the brake pipe length by the wagon length, often the brake pipe of the wagon (or locomotive) is not straight so its length is bigger than the length of the wagon (or locomotive) by the specified ratio.
- *Brake pipe internal diameter [mm]*: The most used values are 27.3 mm (1") and 36 (1"1/4).
- *Control valve*: Model of control valve (distributor) mounted on the wagon (or locomotive)
- *Driver's Brake valve (only for locomotive)*: Type of driver's brake valve used (if the check box is not selected the brake valve does not work).

**Pneumatic parameters**

Brake pipe length / wagon length

Brake pipe internal diameter [m]

Control valve

Driver's Brake valve

Brake valve activated

Fig. 4 Pneumatic parameters

- **Brake system of the wagon or locomotive (braking equipment)**

The wagon can brake with two different braking systems (also at the same time): Block brake and disk brake. The locomotive can brake with three different brake systems, the same two brake systems of the wagon and the electrodynamic brake.

When a block brake and a disk brake system work at the same time, the total brake force is computed as sum of the two braking forces. The inputs used to compute the two braking forces are independent each other, each braking system is characterized from a specific braked weight or physical parameters.

➤ **Block brake** (the general inputs below are the same for wagons or locomotives see Fig. 6):

- *Contribution %*: The actual brake force  $F_f$  computed with the block brake system (see Eq. 12) is scaled ( $F_{fs}$ ) by the contribution  $C_b$  (defined in this field) as showed in the Eq. 1

$$F_{fs} = C_b F_f$$

**Eq. 1 Braking force scaled**

- *Number of bring shoes*: Total number of bring shoes. The number does not depend from the shoes type ( $B_g$ ,  $B_{gu}$ ) because the number of shoes in each bring shoe changes automatically in function of the type. The total surface for each bring shoe in function of the type is specified in the next table.

- *Type of shoes:* The type of shoes implemented are Bg and Bgu, the main differences rely on the contact area among wheel and shoe as described in the table below.

Shoe type	Surface [mm <sup>2</sup> ]
Bg	25600
Bgu	40000

- *Friction Law (advanced option):* The analytical friction laws implemented in the TrainDy software are:
  - Karwatzki
  - BZA (only with Bg shoes) This friction law can not be used with the auto continuous brake system
  - OSS

In the menu it appears the other friction laws defined in the branch Block Friction Laws. The block friction law is defined in terms of speed and specific pressure among wheel and shoe.

An example of a block friction law defined by the user is in Fig. 5.

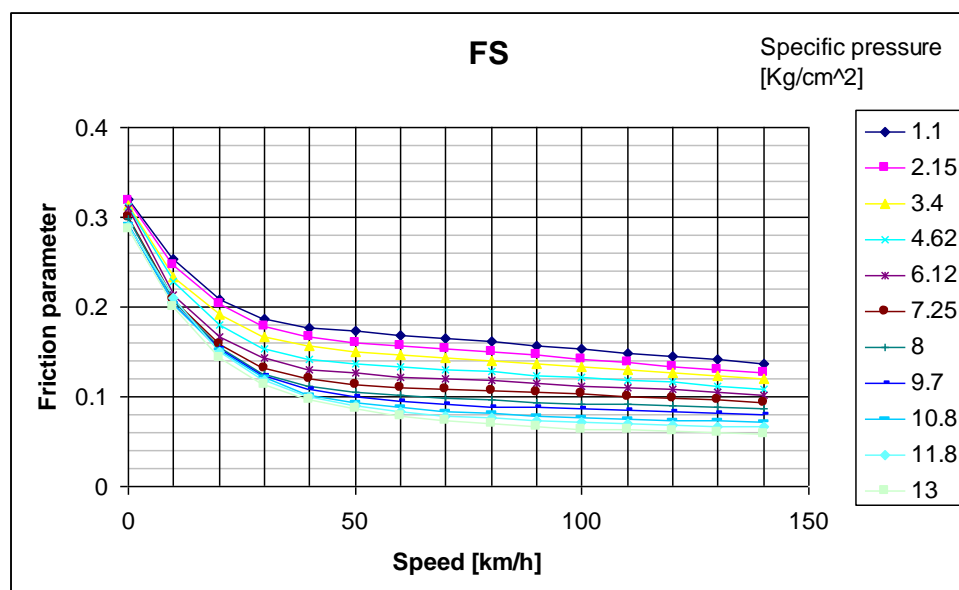


Fig. 5 Example of a block friction law defined by a look – up table

<input checked="" type="checkbox"/> Block brake	contribution:	100.0
	Number of bring shoes	16
	Type of shoes	Bgu ▼
<b>Advanced Options</b>		
	Friction law	Karwatzki ▼
	Rigging Efficiency	0.83
	Ff [kN]	1.5
	Fr [kN]	2.0

Fig. 6 Block brake

- *Rigging efficiency (advanced option)*: The rigging efficiency takes into account the energy dispersion in the rigging system. Generally the rigging efficiency is 0.83 (see UIC 544-1)
- *Ff [kN] (advanced option)*: Ff is the counteracting force applied by the spring on the brake cylinder. (Generally 1.5 kN see UIC 544-1)
- *Fr [kN] (advanced option)*: Fr is the counteracting force applied by the spring on the brake blocks. (Generally 2 kN see UIC 544-1)

The parameters that define the brake block force are different for wagons and locomotives, particularly for wagons they can be set in three ways (for locomotives in two ways):

**Definition of the brake block force (only for wagons):**

- Empty-load system with the brake cylinder characteristics
  - *Rigging ratio*: The rigging ration defines the brake cylinder force amplification.
  - *Cylinder section [dm<sup>2</sup>]*: Cross section of the brake cylinder

- *Inversion Mass [t]*: If the total mass of the wagon is less then (not equal) the inversion mass the system brake with the empty pressure.
- *Empty pressure [Bar]*: Pressure in the brake cylinder corresponding at the empty conditions (almost no load carried on by the wagon).

Empty load system with brake cylinder characteristics

Rigging ratio	<input type="text" value="12.67"/>
Cylinder section [dm <sup>2</sup> ]	<input type="text" value="12.95"/>
Inversion Mass [t]	<input type="text" value="30"/>
Empty pressure [bar]	<input type="text" value="1.25"/>

- Empty load system with the braked weight.
  - *Braked weight load [t]*: It is the braked weight of the wagon in load condition
  - *Changing weight [t]*: If the wagon has a total mass bigger than the changing weight, the wagon brakes with the load braked weight, if the total mass is lower then the changing weight, then the wagon brakes with braked weight empty.
  - *Braked weight empty [t]*: It is the braked weight of the wagon in empty conditions.

Empty load system with brake weight characteristics

Braked weight load [t]	<input type="text" value="55.0"/>
Changing weight [t]	<input type="text" value="40.0"/>
Braked weight empty [t]	<input type="text" value="30.0"/>

More information about TrainDy methods used to compute the braking force from the braked weight are in the section of the technical guide “TrainDy brake system model”.

- Auto continuous system

With this system, the braked weight can be imposed in terms of the total mass of the wagon (the braked weight is defined as a percentage of the wagon mass). The relationship between the % of braked mass and the wagon mass is defined by a sequence of points.

Auto continuous system

Total mass of the wagon [t]	% mass braked
17.5	101
51	100
64	82

**Definition of the brake block force** (only for Locomotives):

- Using the brake cylinder characteristics
  - *Rigging ratio*: The rigging ration defines the brake cylinder force amplification.
  - *Cylinder section [dm<sup>2</sup>]*: Section of the brake cylinder

System with brake cylinder characteristics

**Rigging ratio**

**Cylinder section [dm<sup>2</sup>]**

- Using the braked weight
  - *Brake weight [t]*: It is the braked weight painted on the locomotive

System with brake weight characteristics

**Braked weight [t]**

More information about the method used to compute the braking force from the braked weight are in the section “TrainDy brake system model” in the technical guide.

➤ **Disk brake:**

(the general input data below are the same for wagons or locomotives see Fig. 7)

- *Contribution %*: The actual brake force computed with the disk brake system (see Eq. 7) is scaled by the contribution, defined in this section, as showed in the Eq. 1.
- *Friction coefficient (Advanced option)*: A constant friction coefficient or a friction law can be selected, in the friction laws menu, all defined disk friction laws (.dfl) appear. For disk brakes the friction law is defined in terms of speed multiplied by the ratio between disk braking radius ad wheel radius.

The screenshot shows a configuration window for disk brake parameters. It includes a checked checkbox for 'Disk brake', a text input for 'contribution %' set to 100.0, an 'Advanced Options' button, and two radio button options: 'Constant friction coefficient' (selected) with a value of 0.35, and 'Friction law' with an empty dropdown menu.

**Fig. 7 Disk brake general parameters**

As for the block brake, the brake force can be defined in three ways for the wagons and in two ways for locomotives.

**Definition of the brake disk force (only for wagons):**

- Empty load system with brake cylinder characteristics Fig. 9

In Fig. 8 is showed the structure of a disk brake system calliper, to reproduce this system the parameters in Fig. 9 must be defined.

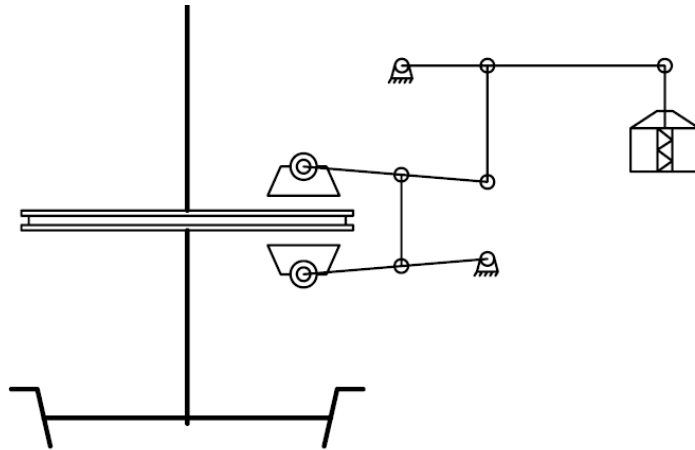


Fig. 8 Scheme of a disk brake system calliper

● Empty load system with brake cylinder characteristics

Cylinder section [dm <sup>2</sup> ]	16.0
Inversion Mass [t]	44.0
Empty pressure [bar]	2.0
First Rigging ratio	3.0
Second Rigging ratio	3.0
First Efficiency	0.9
Second Efficiency	0.9
Counteracting Force [kN]	1.2
Disk braking Radius [m]	1.4
Wheel Radius [m]	1.4

Fig. 9 Input data for disk brake specified with a model more closet o the reality.

- *Cylinder section [dm<sup>2</sup>]*: brake cylinder cross section
- *Inversion mass [t]*: If the wagon has a total mass bigger or equal than the changing weight the wagon brakes in load condition, if the total mass is lower then the changing weight the wagon brakes with empty conditions.
- *Empty pressure [bar]*: Pressure in brake cylinder when the wagon brakes in empty condition.



As showed in Fig. 8 the braking system is composed by two serial cinematic mechanisms, the first near the brake cylinder and the second near the calliper. If the brake cylinder and the brig shoes are joined by only one cinematic reduction, the second rigging ration and second rigging efficiency assume the value 1.

- *First rigging ratio*: The rigging ration defines the brake cylinder force amplification.
  - *Second rigging ratio*: The rigging ration defines the brake cylinder force amplification.
  - *First rigging efficiency*: Rigging efficiency of the first rigging ratio
  - *Second rigging efficiency*: Rigging efficiency of the first rigging ratio
  - *Counteracting Force [KN]*: counteracting force applied by the spring on the brake cylinder.
  - *Disk braking radius [m]*: Disk radius at the shoes – disk contact point
  - *Wheel radius [m]*: Wheel radius in the wheel – rail contact point
- Empty load system with the braked weight.
    - *Braked weight load [t]*: It is the braked weight of the wagon in load condition
    - *Changing weight [t]*: If the wagon has a total mass bigger then the changing weight the wagon brakes with braked weight load, if the total mass is lower then the changing weight the wagon brakes with braked weight empty.
    - *Braked weight empty [t]*: It is the braked weight of the wagon in empty condition.

Empty load system with braked weight characteristics

Braked weight load [t]	<input type="text" value="55.0"/>
Changing weight [t]	<input type="text" value="44.0"/>
Braked weight empty [t]	<input type="text" value="25.0"/>

In this case if the friction law is used there aren't information about the disk braking radius and the wheel radius so the ratio (braking radius/wheel radius) is imposed to 247/470.

- Auto continuous system

With this system it can be imposed a braked weight as function of the total mass of the wagon (the braked weight is defined as a percentage of the wagon mass). The relationship among the % of braked weight and the wagon mass is defined by a sequence of points.

**Auto continuous system**

Total mass of the wagon [t]	% mass braked
10	100
100	100

Alternatively, the user can impose the maximum air pressure in the brake cylinder, the corresponding air pressure for application stroke and the in-shot function as a function of the total wagon mass. This modification applies to the wagon and not the Control Valve, even if the parameters refer to the Control Valve's characteristics. The reason is that the user defines the gross mass vs maximum pressure characteristics for wagons, not control valves.

**Auto continuous system with brake cylinder characteristics**

Total mass of	Brake cyl.	Pressure for AS	Pressure for IS
0	0	0	0
0	0	0	0

**Rigging ratio**

**Rigging Efficiency**

**Cylinder section [dm<sup>2</sup>]**

**Definition of the brake disk force (only for locomotives):**

- System with brake cylinder characteristics
  - *Cylinder section [dm<sup>2</sup>]*: brake cylinder cross section

- *First rigging ratio*: The rigging ration defines the brake cylinder force amplification.
- *Second rigging ratio*: The rigging ration defines the brake cylinder force amplification.
- *First rigging efficiency*: Rigging efficiency of the first rigging ratio
- *Second rigging efficiency*: Rigging efficiency of the first rigging ratio
- *Counteracting Force [kN]*: counteracting force applied by the spring on the brake cylinder.
- *Disk braking radius [m]*: Disk radius at the shoes – disk contact point
- *Wheel radius [m]*: Wheel radius in the wheel – rail contact point

System with brake cylinder characteristics

Cylinder section [dm <sup>2</sup> ]	16.0
First Rigging ratio	3.0
Second Rigging ratio	3.0
First Efficiency	0.9
Second Efficiency	0.9
Counteracting Force [kN]	0.0
Disk braking Radius [m]	1.0
Wheel Radius [m]	1.4

- System with the braked weight characteristics.
  - *Braked weight*: To set the brake force, the model uses the Braked Weight of the loco.

System with brake weight characteristics

Braked weight [t]	70
-------------------	----

➤ **Electrodynamics characteristics** (only for locomotive):

- **Brake** (Fig. 10)

The electrodynamics brake system imposes the brake force as function of the speed and as function of the time. The relationships among brake force

and speed (electrodynamics brake characteristics) and among the brake force and the time of application (usually an increasing function) can be defined by a set of points in Fig. 10. The brake force value assumed during the braking is the minimum (as absolute value) between the two characteristics and it is function of the train speed.

**Braking**

**Electrodynamics brake characteristic**

Speed [Km/h]	Brake force [KN]
0	0
2	100
45	100

**Increasing function**

Time [sec]	Brake force [KN]
0	0
5	100

**Fig. 10 Definition of the electrodynamic brake.**

- **Traction** (Fig. 11)

The locomotive traction characteristics can be defined by two tables as for the braking: the tables are shown in the screenshot below and define the traction force – speed characteristic and the traction force - time characteristic. The value assumed during the simulation (as for the braking) is function of the speed and it is the minimum (as absolute value) between the two characteristics. Moreover it's possible define a gradient for traction insertion and removal; these gradients are used each time the loco percentage of application (in Manoeuvre tab) is changed: when it is increased (e.g. from 0 to 100%) the traction insertion gradient is used, on the contrary, when it is decreased (e.g. from 100% to 0%) the traction removal gradient is used. The (traction/removal) gradient has the priority on the force vs time characteristics, so that if the User wants reproduce a

particular Force vs Time evolution, the gradients shall be set to zero. The same gradients are used both for traction and for braking.

**Traction**

**Electrodynamics traction characteristics**

Speed [Km/h]	Traction force [kN]
0	200
2	200
45	75

Add Row Remove Row

**Increasing function**

Time [sec]	Traction force [kN]
0	0
5	100

Add Row Remove Row

Traction insertion gradient [kN/s]

Traction removal gradient [kN/s]

Fig. 11 Definition of the traction.

The braking and traction characteristics defined in Fig. 10 and Fig. 11 are showed in Fig. 12. In way to clarify how the software works (with the exception of gradients) an example is described.

If a locomotive works in traction with a speed of 20 km/h after 10 seconds from the traction start, the values computed following the Time – Traction force characteristic is 200 kN and following the Speed – Traction force characteristic is 150 kN; the value used in the simulation is the minimum value (150 kN). Instead, after 5 seconds if the speed is 20 km/h, the value used is 100 kN.

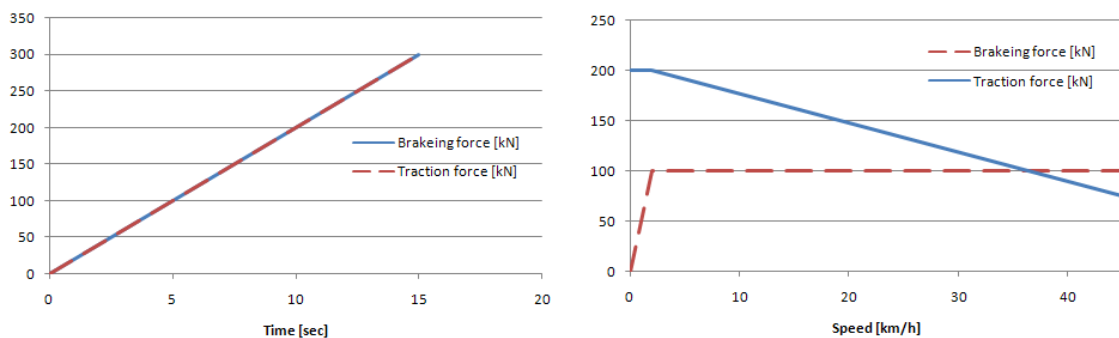


Fig. 12 Braking and traction force characteristic

If there aren't information about traction or electrodynamic brake you can put all zeros in the electrodynamic characteristics and in the increasing function, by this way the system disable the traction or the braking. Moreover, consider that by using the relationship among the traction force (or the braking force) and time the User can replicate every desired (or measured) time dependent force at locomotive.

## BUFFERS AND DRAW GEARS

In this section the parameters that TrainDy uses to define the characteristics of buffers and draw gears are described (for more details see “BUFFERS AND DRAW GEARS IMPLEMENTATION”).

The parameter used are:

- *Load limiting velocity [m/s]*: When the relative speed, in load conditions, is upper than load limiting velocity the model uses the load force – stroke characteristic. Load condition for buffers happens when the relative displacement between buffers decrease, the opposite for draw gears.
- *Unload limiting velocity [m/s]*: When the relative speed, in unload conditions, is upper then unload limiting velocity the model uses the unload stroke – force characteristic
- *Load and unload force – displacement (stroke) characteristics*: In the table below are defined the stroke – force load and unload characteristics

In order to define a buffer (or draw gear) it is necessary to input two force - stroke characteristics (Load and unload force - stroke characteristics), the load limit speed and unload limit speed, or the load characteristic, the limiting speeds and a “damping coefficient”. The unload characteristic is computed by the relation below:

$$F\_unload = F\_load \cdot \left(1 - \frac{\%Damping}{100}\right)$$

It is now possible to add a viscous damping to model, in a simplified way, the hydraulic buffers. Moreover, it is possible to define that the force-displacement characteristic is referred to a central coupler by ticking the appropriate box

An example of central coupler is showed in Fig. 13:

**Buffers and draw gears characteristics**

Name:

Load limiting velocity [m/s]:

Unload limiting velocity [m/s]:

Damping coefficient:

Viscous damping coefficient [kN s/m]:

---

**Load and unload characteristics**

Central Coupler

Stroke [mm]	Load [kN]	Unload [kN]
0	34	0
2	50	10
10	85	20
20	135	35
30	180	55
40	240	75
50	320	90
60	405	130
70	530	180
80	680	270
90	870	390
100	1,120	640
108.6	1,329	1,329

**Load and unload characteristics**

Fig. 13 Example of central coupler.

When the load and unload characteristics are defined the User can view them in a graph by a clicking on the button refresh graph. In this graph the load and unload characteristics are painted by straight lines to avoid particular behaviours in which the gradient changes the sign. In TrainDy



software the load and unload characteristics are computed by a piecewise third order polynomial function.

The points that define the load and unload characteristics ought be defined in way to have a limited force step between two consecutive definition points. To verify if the points chosen reproduce the real behaviour, an external tool can be used. The tool is called “Verify\_bgdg.exe”, it is located in the TrainDy main folder and it is automatically installed during the TrainDy setup. By a double click on it the buffers and draw gears equivalent characteristics used during the last simulation are plotted. The behaviours on the graphs are defined by piecewise third order polynomial functions. When the tool is launched a window for each different coupling (buffers or draw gears) is plotted. Each window is made by three graphs as showed in Fig. 14; the upper graph is the equivalent characteristic of the coupled buffers or draw gears and the other two graphs are the force stroke characteristics of the rear buffer (or draw gear) and the front buffer (or draw gear).

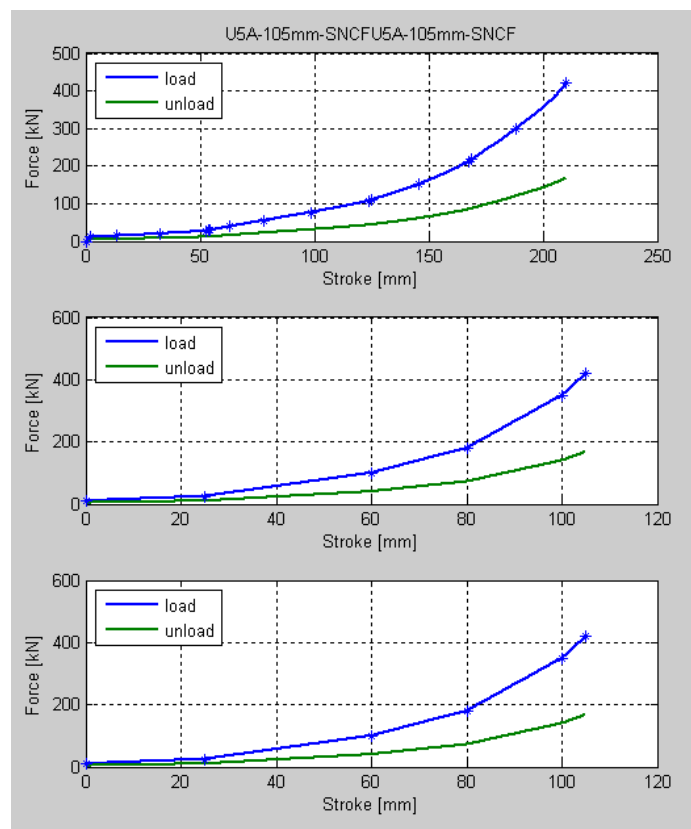


Fig. 14 Output of the tool “Verify\_bgdg.exe”

## CONTROL VALVE

In this section the parameters used to define the control valve behaviour are described (More details about the pneumatic are in the section “PNEUMATIC MODEL OF THE TRAIN BRAKE”), these parameters are:

### LIMITING CURVE

- *Application stroke data* (see Fig. 15)
  - a) *Pressure in brake cylinder*: Pressure of the application stroke (constant)
  - b) *Time*: Minimum time interval during which the application stroke pressure is maintained
  - c) *p general pipe*: Leap of pressure in general brake pipe, referred to the maximum pressure, corresponding at the end of application stroke in brake cylinder

During the building-up of pressure in brake cylinder (starting from 0 bar) it is necessary to define the preliminary raising of pressure in BC without considering the transfer function and the brake type behaviour. This phase is characterized by the application stroke and the following in-shot function. In order to simulate the application stroke phase, it is essential to input the pressure in BC reached at activation of control valve (this value will be maintained until the end of application stroke, when the brake shoe gets in contact with the wheel). The starting of the application stroke is defined by a decreasing leap in general brake pipe from the maximum value (*p general pipe*). The end of application stroke is controlled by the time interval (*Time*) and by the BP pressure (*p general pipe*). In the model, the User defines the minimum time interval of application stroke (*Time*) and the leap of pressure in BP (*p general pipe*). Obviously, the actual duration of this phase will be the maximum time among the time interval (*Time*) and the time to reach the pressure in BP (*p general pipe*).

An example is showed in Fig. 15, the brake cylinder described by the blue line, maintains the pressure of the application stroke for the minimum time interval imposed (*Time*). On the contrary, the end of the application stroke for the red curve is defined by the (*p general pipe*).

The parameters used in the test in Fig. 15 are:

Application stroke data	
Pressure in brake cylinder [bar]	<input type="text" value="0.35"/>
time [s]	<input type="text" value="0.6"/>
p general pipe [bar]	<input type="text" value="0.3"/>

- **Inshot function data** (The inshot function is a pressure trend imposed by the distributor, no depending from the brake regime)
  - d) *Pressure in brake cylinder*: Pressure in brake cylinder corresponding at the end of the inshot function
  - e) *Time*: Time difference between the end of the inshot function and the end of the application stroke.

After the application stroke phase the pressure in BC rises linearly according to the inshot function until “Pressure in brake cylinder” in a time interval equal to “Time”.

Fig. 15 shows an example of the influence of the inshot function parameters, the parameters used in the example are :

Inshot function data	
Pressure in brake cylinder [bar]	<input type="text" value="0.55"/>
Time [s]	<input type="text" value="0.6"/>

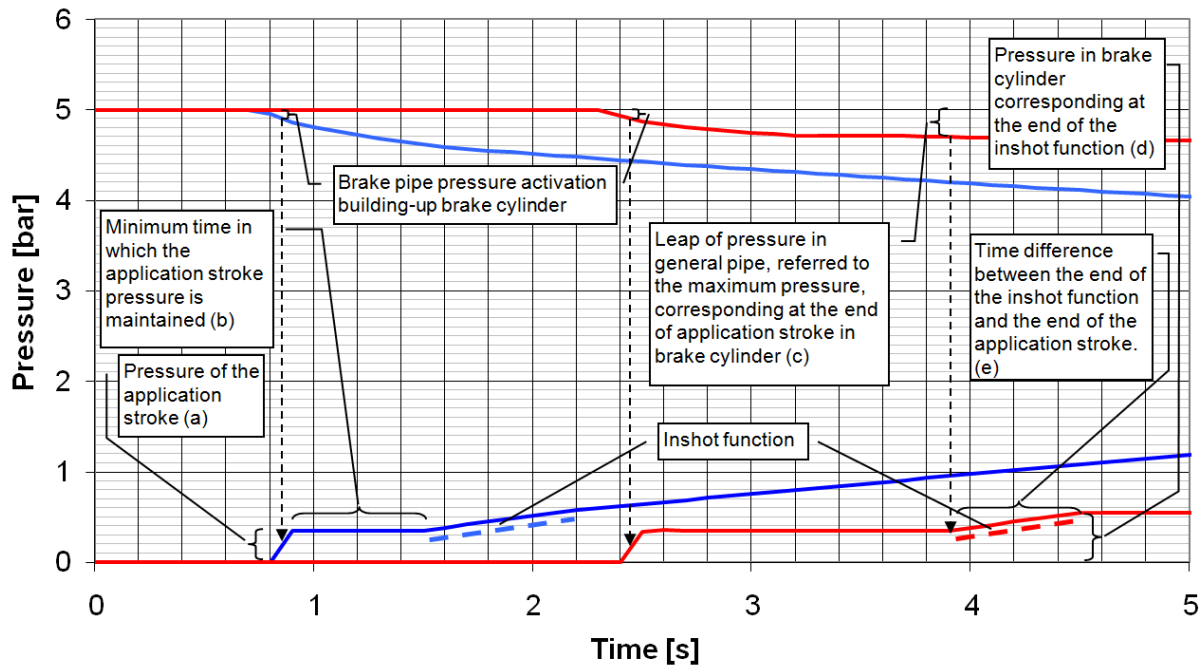


Fig. 15 Description of the application stroke data, and inshot function data

- **Braking timing P and G**

- *Time at 95% Pmax*: Time to reach the 95% of Pmax
- *Time at Pmax*: Time to reach the Pmax

In order to define the brake limiting curve it is necessary to set the time to reach the 95% of maximum pressure and the time to reach the maximum pressure. The shape of limiting curve is calculated starting from the end of the in-shot function data (with 3 points a parabolic curve is obtained).

Fig. 16 shows the effect of the limiting curve parameters. It is extracted by a simulation in G regime and the blue curve is fully bypassed by the limiting curve. The parameters used are written below and they differ from the case of Fig. 15.

The red curve is not influenced by the limiting curve because the pressure in general brake pipe decreases slowly so the pressure in brake cylinder is imposed by the transfer function.

**Braking timing**

**P regime**

At 95% Pmax [s]

At Pmax [s]

**G regime**

At 95% Pmax [s]

At Pmax [s]

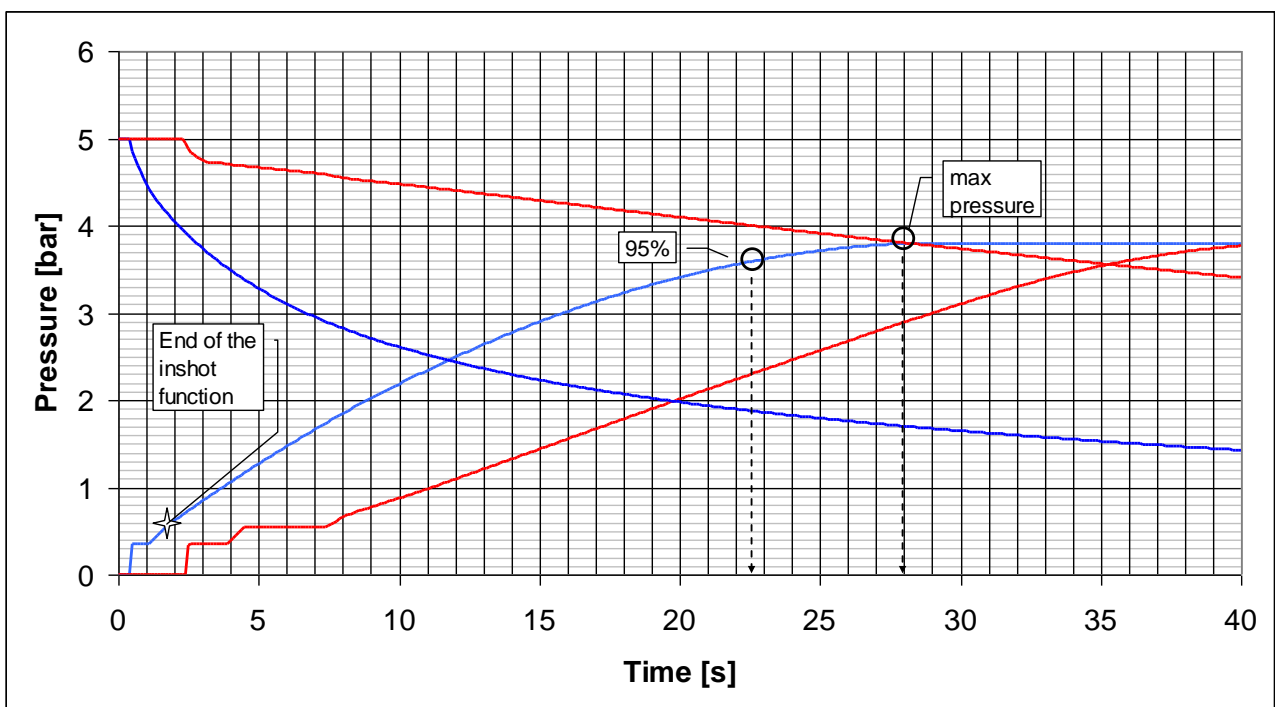


Fig. 16 Effect of the limiting curve

Also for releasing is defined a limiting curve as showed in Fig. 17.

- **Releasing timing P and G**
  - *Time at 110% Pmin*: Time to reach the 110% of Pmin
  - *Time at Pmin*: Time to reach the Pmin

**Releasing timing**

**P regime**

At 110% Pmin [s]

At Pmin [s]

**G regime**

At 110% Pmin [s]

At Pmin [s]

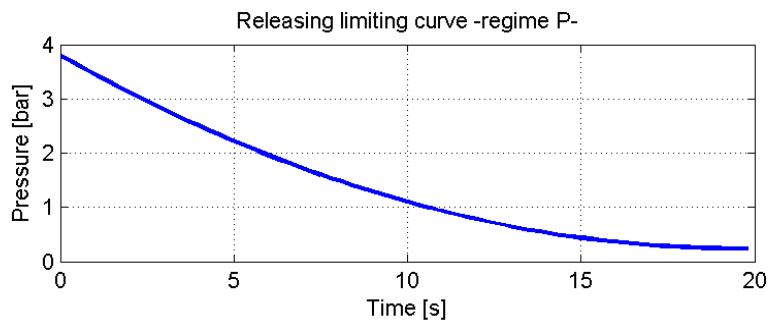


Fig. 17 Releasing limiting curve

## TRANSFER FUNCTION

- **Braking transfer function**

- pGP; pBC: The model takes as input points the pressure in general pipe and the corresponding pressure in brake cylinder, that allow building the braking transfer function. The data of pressure in brake cylinder must be referred to maximum nominal pressure of 3.8 bar.

The transfer function defines the relationship among the brake cylinder pressure and the brake pipe pressure and it is applied after the inshot function. The shape of the function that imposes the relationship is defined by a series of points (“pressure in general brake pipe”, corresponding “pressure in brake cylinder”) as showed in Fig. 18. Knowing the pressure in the general brake pipe, using the transfer function, the model imposes the pressure in brake cylinders.

Analyzing the red curves in Fig. 16 and using the transfer function in Fig. 18, when the pressure in brake pipe reaches 4 bars the transfer function imposes 2.3 bar in brake cylinder.

The values used for the transfer function in Fig. 16 and Fig. 18 are written below.

TRANSFER FUNCTION	
Braking (Referred to 3.8 bar)	
Pressure in Brake pipe [bar]	Pressure in Brake cylinder [bar]
3.42	3.8
4.7	0.55
Releasing (Referred to 3.8 bar)	
Pressure in Brake pipe [bar]	Pressure in Brake cylinder [bar]
3.75	3.8
4.7	0.23
4.85	0.23

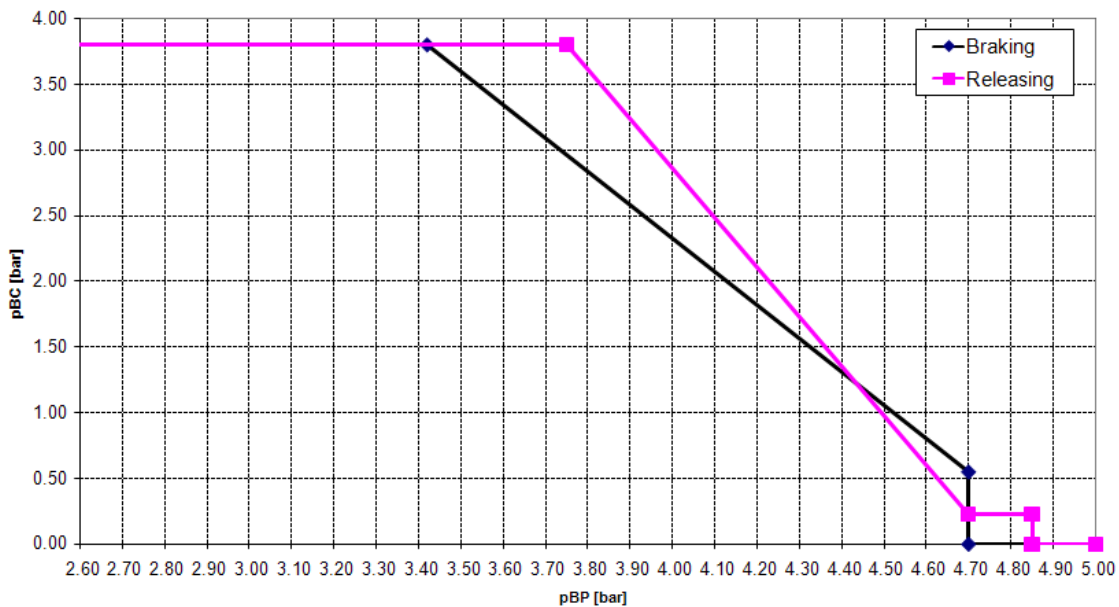
Advanced Options

Also for releasing, it is defined a transfer function:

- **Releasing transfer function**

pGP; pBC: The model takes as input points the pressure in general pipe and the corresponding pressure in brake cylinder, that allow building the releasing transfer function. The data of pressure in brake cylinder must be referred to maximum nominal pressure of 3.8

### Matching releasing/braking function



bar.

Fig. 18 Braking transfer function

- **Advanced features**

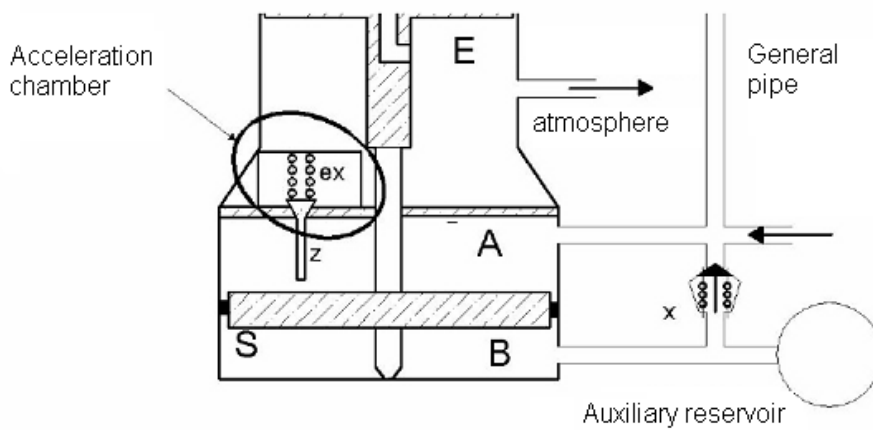
During a braking, the accelerating chambers of the control valves are modelled as lateral small volumes connected to the brake pipe by means of nozzles of equivalent constant diameters. The pressure of the air in these volumes changes according to the inflow flux mass, holding constant, for hypothesis, the temperature of the air in the volume of the accelerating chamber. In this case, the upstream pressure of the nozzle is the pressure in the brake pipe and the downstream pressure is the pressure of the air in the small volume; after a certain time period, the pressure of the air in the small volume is bigger than the pressure in the brake pipe and the air goes back, from the small volume to the brake pipe. Both the nozzle diameter and the capacity of the lateral volume are tuned on experimental measurements. Fortunately, this tuning occurs only one time and it neither depends on the manoeuvre (emergency or service braking) nor on the train length.

- *$\Delta p$  Brake pipe pressure activation building-up brake cylinder*: Leap in general brake pipe, referred to the maximum pressure, corresponding at the start of the application stroke. This pressure is necessary to win the static friction.
- *Stroke of brake cylinders [mm]*: This parameter is used only during the releasing in way to compute the decreasing pressure in the auxiliary reservoir.
- *Volume of acceleration chamber [L]*: this data is used to calculate the contribution of acceleration chambers on brake pipe pressure drop.
- *Diameter of acceleration chamber [mm]*: this parameter sets the orifice diameter. The orifice joints the volume of acceleration chamber with brake pipe: this data controls the air flow between brake pipe and acceleration chamber.
- *DP Brake pipe pressure for activation of acceleration chamber [bar]*: in the model of acceleration chamber, its opening is controlled by the reduction of pressure in brake pipe, imposed in this field.
- *Volume of auxiliary reservoir [L]*: This parameter defines the volume of the auxiliary reservoir, it is useful to compute the pressure in the reservoir during the braking because this value will be used during the releasing phase.
- *Diameter of auxiliary reservoir [mm]*: This is the equivalent diameter of the nozzle supposed placed between the general brake pipe and the auxiliary reservoir (see Fig. 31).
- *dP imposed by check valve AR-BP [bar]*: This is the surplus of pressure that the brake pipe must have to win the static force of the auxiliary reservoir refilling valve.



- *Pressure on running of auxiliary reservoir [bar]:* This is the initial pressure of the auxiliary reservoir.

Advanced Options	
dP Brake pipe pressure activation building-up brake cylinder [bar]	0.18
Stroke of brake cylinders [mm]	150.0
Volume of acceleration chamber [L]	0.8
Diameter of acceleration chamber [mm]	4.0
dP Brake pipe pressure for activation of acceleration chamber [bar]	0.09
Minimum pressure closing of acceleration chamber [bar]	0.0
Volume of auxiliary reservoir [L] (not useful for braking)	125.0
Diameter of auxiliary reservoir [mm] (not useful for braking)	10.0
dP imposed by check valve AR-BP [bar] (not useful for braking)	0.1
Pressure on running of auxiliary reservoir [bar] (not useful for braking)	4.8

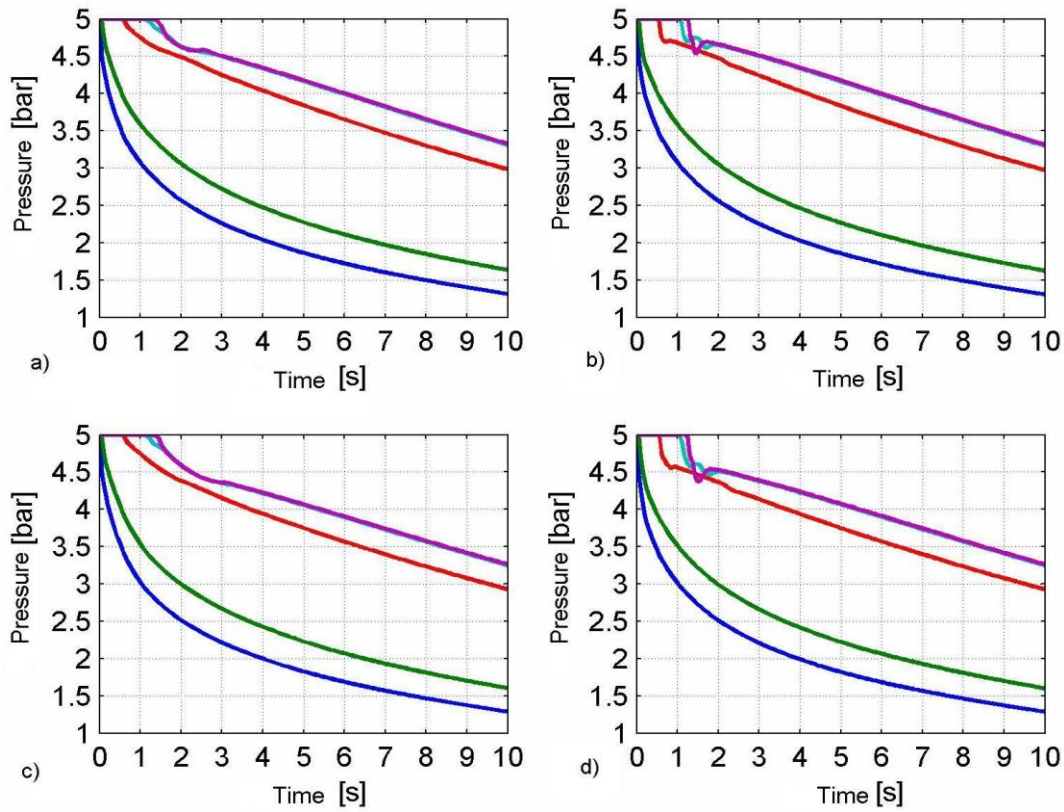


**Fig. 19 Simple diagram of the acceleration chamber in the control valve**

The effects of the volume and the equivalent diameter between the acceleration chamber and the general pipe can be seen in the graphs in Fig. 20.

The diameter and the volume of the accelerating chamber. change only the first part of the emptying of the general brake pipe. Increasing these values, especially the equivalent diameter between the general pipe and the accelerating chamber, the pressure decreases quickly particularly for the wagons far from the master brake valve.

The wagon measured in Fig. 20 are: 1, 2, 10, 18 and 21 (last wagon)



**Fig. 20** Effects of the acceleration chamber's volume and diameter. a) reference conditions, b) Increasing only the equivalent diameter between acceleration chamber and the general pipe, c) Increasing only the volume of the acceleration chamber, d) Increasing of each characteristics volume and diameter.

the values used in the tests in Fig. 20 are written in the table below:

Test	Volume [l]	Diameter [mm]
a)	0.9	3
b)	0.9	7
c)	1.5	3
d)	1.5	7

In Fig. 22 are showed the effects of the parameter “Brake pipe pressure for activation of acceleration chamber”. In the test 2 when the pressure in brake pipe decreases of 0.2 bar the acceleration chambers works and the pressure has a fast decrease. In the test 1 the acceleration chamber starts early because they need a decrease of only 0.08 bar in brake pipe to start.

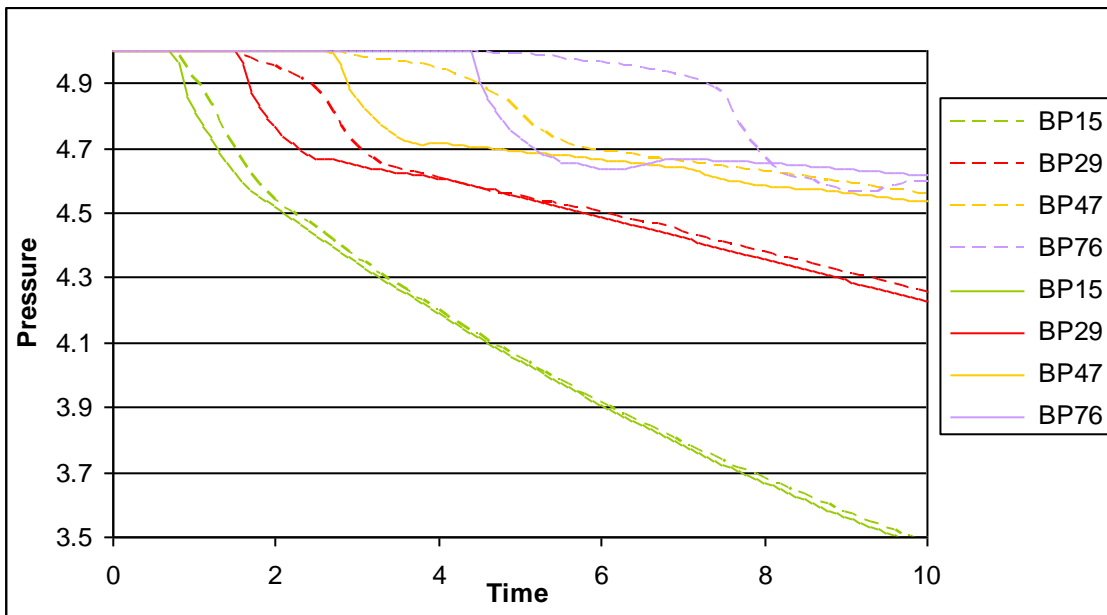


Fig. 21 Effects of the parameter “Brake pipe pressure for activation of the acceleration chambers”,  
 — Test 1, --- Test 2

$\Delta p$  decreasing in B.P. for  
 activation of acceleration  
 chambers

Test 1	0.08 bar
Test 2	0.2 bar

## BRAKE VALVE

The driver's brake valve is modelled as a nozzle with an equivalent diameter, tuned on experimental data. During a braking, the upstream pressure of the nozzle is the pressure of the brake pipe, while its downstream pressure is either the atmospheric pressure, for an emergency braking, or, for a service braking, the pressure of the pilot chamber of the driver's brake valve. In the latter case, the downstream pressure is time-variable, according to a law provided by the constructor of the valve (Fig. 22 shows an example, with different target pressures).

On the other hand, during a releasing, the upstream pressure of the nozzle is the pressure of the pilot chamber, yet provided by the valve constructor, whereas the downstream pressure is the pressure of the brake pipe. Since the real pneumatic circuits used for the three previous manoeuvres are different, there are also three equivalent diameters of the nozzle that simulates the same driver's brake valve. Fortunately these diameters need to be determined only one time and they do not depend on the length of the train.

For each manoeuvre an equivalent diameter must be imposed as an input and in the advanced options there is the possibility to choose a fixed  $C_q$  value (flow coefficient) or the Perry's law.

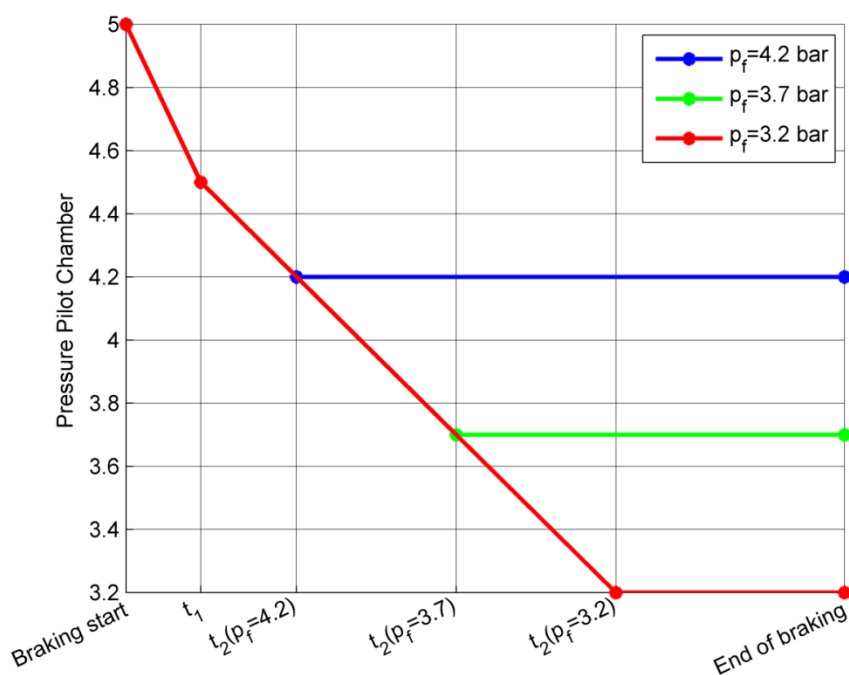


Fig. 22 Time evolution of the counter-pressure in the pilot chamber for a service braking

### Emergency

- *Diameter of equivalent orifice [mm]*: Discharge diameter

In the emergency brakes the counter pressure corresponds to the atmosphere pressure.

- *Flow coefficient  $C_q$  – Perry’s law (Advanced option)*: A constant flow coefficient or the Perry’ law can be chosen

EMERGENCY BRAKING

Diameter of equivalent orifice [mm]

Flow coefficient  $C_q$

Use Perry's law

### Service

- *Diameter of equivalent orifice [mm]*: For service brakes the counter pressure (pressure in pilot chamber) is defined by two time intervals, that simulates the behaviour of the driver’s brake valve during a service brake
- *Flow coefficient  $C_q$  – Perry’s law (Advanced option)*: A constant flow coefficient or the Perry’ law can be chosen
- *Time to achieve a drop of 1.5 bar in counter-pressure shape (from 5 bar to 3.5 bar)[s] (Advanced option)*: Time of first decreasing in counter-pressure shape (from 5 bar to 4.5 bar) [s] (Advanced option)

SERVICE BRAKING

Diameter of equivalent orifice [mm]

Flow coefficient  $C_q$

Use Perry's law

Time to achieve a drop of 1.5 bar in counter-pressure shape (from 5 bar to 3.5 bar)[s]

Time of first decreasing in counter-pressure shape (from 5 bar to 4.5 bar) [s]

## Releasing

- *Diameter of equivalent orifice [mm]*: The shape of the counter pressure (pressure in pilot chamber) is a straight line defined by the Time to achieve an increasing of 1.5 bar in counter-pressure shape (from 3.5 bar to 5 bar) [s]
- *Flow coefficient Cq – Perry's law (Advanced option)*: A constant flow coefficient or the Perry' law can be chosen
- *Time to achieve a increasing of 1.5 bar in counter-pressure shape (from 3.5 bar to 5 bar) [s] (Advanced option)*

**RELEASING**

Diameter of equivalent orifice [mm]

Flow coefficient Cq

Use Perry's law

Time to achieve an increasing of 1.5 bar in counter-pressure shape (from 3.5 bar to 5 bar) [s]

## BLOCK FRICTION LAW

New friction laws can be defined by look – up tables, for example in Fig. 23 (screenshot from the GUI) you can define the friction coefficient as function of the speed (first row) and the specific pressure between the shoes and the wheel (first column).

Analytical block friction laws are defined into the software as default, in particular, these friction laws are: Karwatzki, BZA, OSS and others (more explanations about them are in the section “FRICTION LAWS”).

Specific pressure [N/m <sup>2</sup> ]	Speed [Km/h]											
	0	0	10	20	30	40	50	60	70	80	90	100
1.1	0.32	0.254	0.209	0.186	0.177	0.173	0.169	0.165	0.161	0.157	0.153	0.153
2.15	0.318	0.246	0.203	0.178	0.166	0.16	0.157	0.153	0.15	0.146	0.142	0.142
3.4	0.314	0.234	0.191	0.167	0.156	0.15	0.146	0.143	0.14	0.136	0.133	0.133
4.62	0.312	0.228	0.18	0.153	0.141	0.136	0.133	0.13	0.128	0.124	0.122	0.122
6.12	0.31	0.214	0.166	0.143	0.13	0.126	0.122	0.12	0.118	0.115	0.112	0.112
7.25	0.3	0.208	0.158	0.132	0.12	0.113	0.11	0.108	0.107	0.105	0.103	0.103
8	0.298	0.207	0.155	0.125	0.112	0.105	0.102	0.099	0.096	0.094	0.092	0.092
9.7	0.292	0.204	0.153	0.123	0.108	0.1	0.095	0.091	0.089	0.088	0.086	0.086
10.8	0.292	0.203	0.152	0.122	0.102	0.094	0.088	0.084	0.082	0.079	0.077	0.077
11.8	0.29	0.21	0.15	0.118	0.1	0.09	0.083	0.079	0.076	0.074	0.072	0.072
13	0.286	0.2	0.144	0.114	0.096	0.086	0.078	0.073	0.07	0.067	0.064	0.064

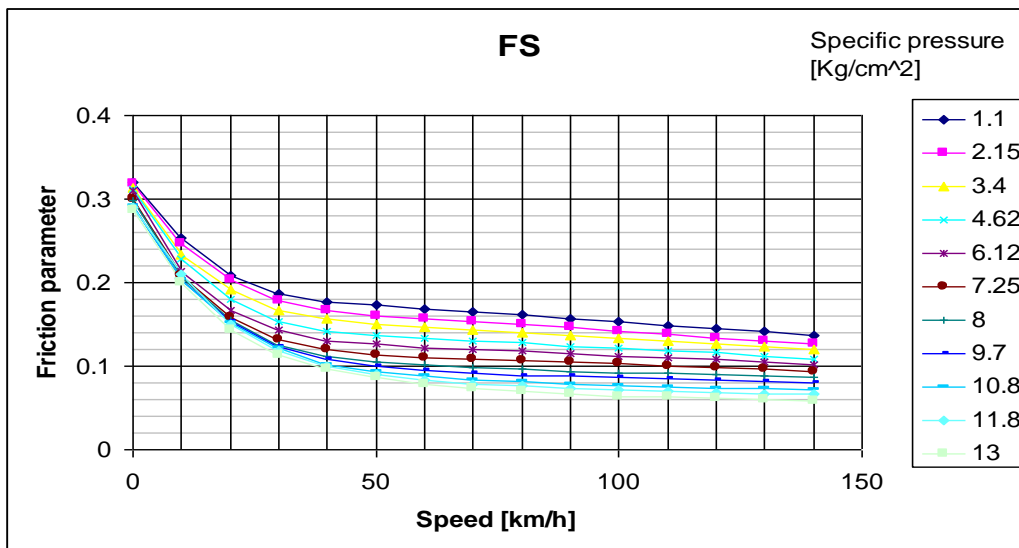


Fig. 23 Example of a look – up table that define a block friction law

## DISK FRICTION LAW

New disk friction laws can be defined by the table below (screenshot from the GU). In the first column there is the product written below, and in the second column there is the friction coefficient.

$$\frac{\text{Speed}[Km/h] \cdot (\text{shoes\_disk\_contact\_radius})}{\text{Wheel\_radius}}$$

V*Rm/Rr	Friction coefficient
0	0.42
20	0.39
50	0.37
100	0.35

**Add Row**      **Remove Row**



## TRACK

In order to define a track, the User can compose different sections of back straights and curves. The back straight is defined by the length and the up-hill. The curve is defined by length, curvature radius, up-hill and elevation. The last column of the table (see picture blow) contains the length of the parabolic section that joins the actual section with the previous section. The cell of the parabolic length in the first row is locked because there is anything to join with a previous section.

The track must always start with a straight section even if it is of a small length (e.g. 0.1 m).

Section Type	Length [m]	Curvature radium [m]	Slope [1:1000]	Elevation [cm]	Parabolic length [m]
Back straight	100	0	0	0	0
Curve	400	1,000	2	2	200
Back straight	2,000	0	4	0	200

### Convention on the signs:

- Curvature radium: Right curve +, Left curve –
- Slope: Up-hill +, Down-hill –
- Elevation: elevation of the left rail +, right rail –

## MANOEUVRE

By the table in Fig. 24 can be defined a manoeuvre for a locomotive, the manoeuvre file is a database element because it can be applied to a generic locomotive in the train configuration.

In Fig. 24 there is a manoeuvre example, in which there are three phases: after 40 meters from the beginning locomotive applies traction until 50 km/h then a pneumatic and electrodynamic brake are performed for 20 seconds.

In the first column of the table the control parameter to use can be selected, there are five types of possible controls: Time, Speed, Distance, Pressure in BP and Pressure in BC. In the second column the control parameter must be defined; for example in the second row of Fig. 24 the speed is set to 50, it means that the simulation stops when the locomotive speed reaches the speed of 50 km/h; in the last column Time is set to 20 s, it means that the duration of the sub-manoeuvre is 20 s; .

In the columns from 3 to 6, the brake system applied can be selected; if any check box is selected (for example in the first row) the locomotive does not perform any manoeuvre and it is in “running condition”. The last column is used to define the vehicle where the pressure in BP or BC is checked: in this case the sub manoeuvre ends when the pressure in column 2 (“Control”) on the vehicle reported in column 7 (“Vehicle”) is reached.

Manoeuvre Data									
Type	Control	pneumatic brake	e.d. brake	e.p. brake	traction	Pressure [bar]	Delay [s]	% of application	Vehicle
Distance [m]	40	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0	0	0
Speed [km/h]	50	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0	0	100	0
Time [sec]	20	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0	100	0

**Fig. 24** Manoeuvre definition

If the pneumatic brake is selected, the pressure (target pressure) must be selected (0 bar for an emergency brake). The electro-dynamic (e.d.) brake cannot be selected at the same time of the traction, if the electro-dynamic brake or the traction is selected, the last two columns must be

defined. Providing a the delay, the User can apply the e.d. brake or traction after a fixed time. With the % of application the actual e.d. traction or braking force are scaled by that factor.

In Fig. 25 and Fig. 26 the effects of the manoeuvre defined in Fig. 24 are showed, the speed and distance reference is taken on the first vehicle (Locomotive).

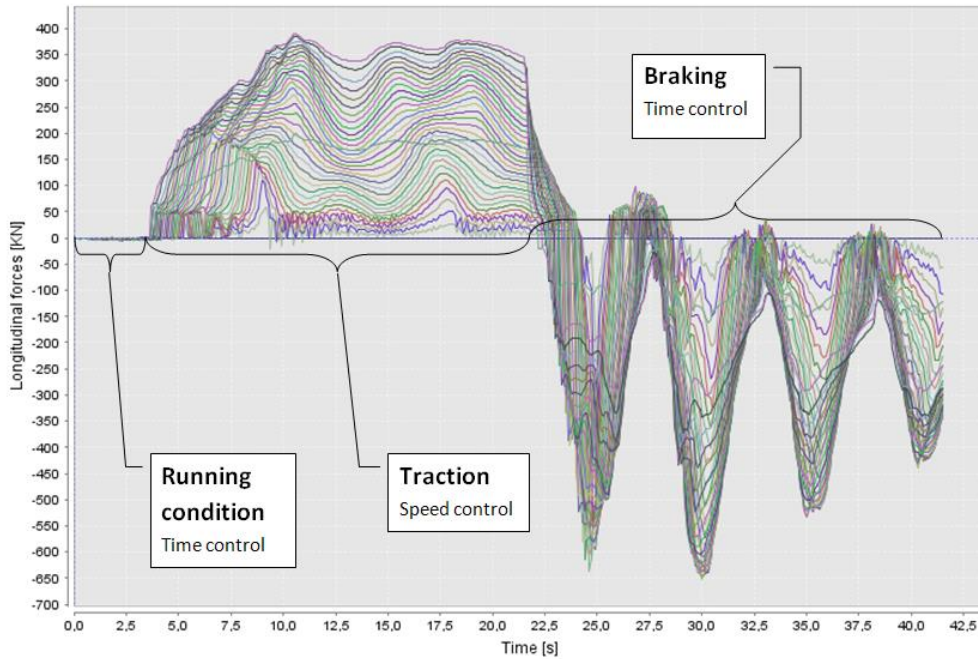


Fig. 25 Longitudinal forces (manoeuvre effects)

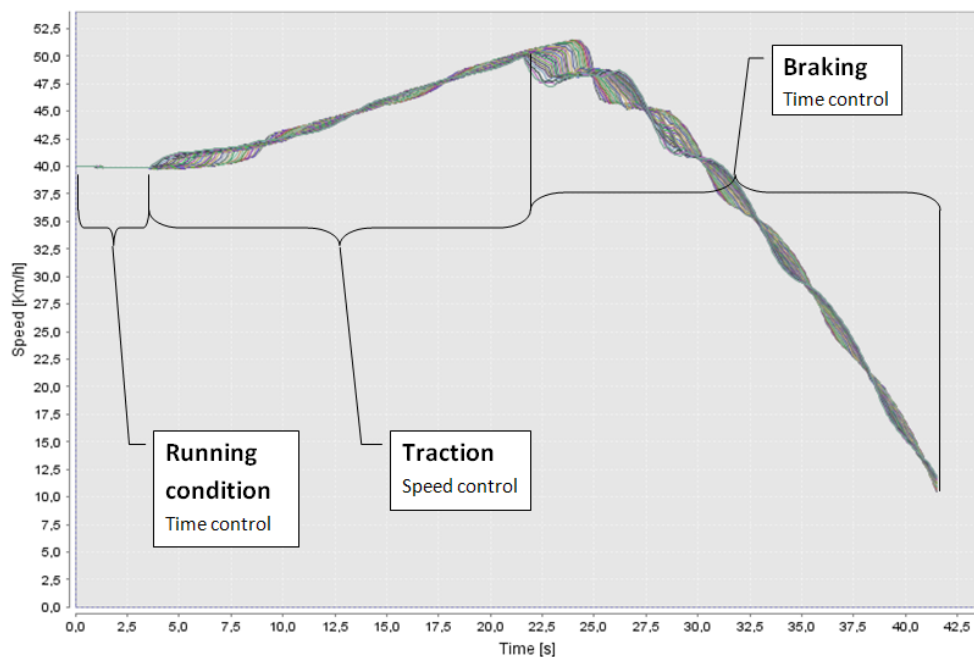


Fig. 26 Speed (manoeuvre effects)

## CONFIGURATION

The configuration window has two tabs: in the first tab there is the train configuration, in the second, the generic data.

### TRAIN CONFIGURATION

In the configuration window, the train composition and some parameters of each vehicle can be selected.

The columns are painted with different colours, the white columns contain values that must be defined to perform a simulation, these columns are:

- *Type*: In this column the name of the locomotive or the wagon can be chosen
- *Manoeuvre*: Only for the locomotives the User can associate a manoeuvre
- *Load [t]*: Load carried on each wagon, for the locomotive the load is 0.
- *GP coupling status*: If the check box is selected the general pipe (hose coupling) on the rear of the selected vehicle is uncoupled, it means that the general pipe is divided into two independent parts.

The grey columns contain values that can not be changed but they are showed only for information purpose. These columns are:

- *Position*: position of the vehicles in the train.
- *Train Length [m]*: in this column there is the sum of the vehicles length, the last cell contains the length of the train.
- *Tare [t]*: the values in this column are taken from the wagons and locomotive windows, and they are not changeable.
- *Train mass [t]*: The masses of each vehicle are summed; the last cell contains the total mass of the train.

The blue columns contain, as default values, the values defined in the wagons or locomotives project database, in these columns the values can be changed vehicle by vehicle without modifying the database: by this way the User does not have to duplicate the elements. The values changed in these columns have the priority on the values defined in the project database. The blue columns are:

- *Wagon length [m]*: Length between buffers for each wagon or locomotive
- *Brake pipe length [m]*: This parameter is computed by the wagon length and the ratio brake pipe length / vehicle length defined in the wagons and locomotives windows; it can be changed vehicle by vehicle in the configuration table.
- *F<sub>k</sub> [KN]*: by this values, the sum of the maximum contact forces between shoes (or disk) and wheel can be defined, for each vehicle,. The braking force is computed scaling the maximum contact force according to the pressure in the brake cylinder.
- *Brake cylinder experimental target pressure [bar]*: This is the real target pressure measured in the brake cylinders. The control valves in a train often have different behaviours, so the experimental target pressures in the brake cylinders are not the same. By this column this phenomena can be reproduced.
- *Brake cylinder nominal target pressure [bar]*: This is the nominal pressure in brake cylinder that realizes the braked weight defined in the brake system. If the brake system is defined by the brake cylinder characteristics then this pressure is not useful. If the experimental brake cylinder pressure is bigger (lower) than the nominal pressure, the braked weight applied is bigger (lower) than the nominal braked weight defined.
- *Control valve*: in this column the control valve for the vehicle can be selected.
- *Control valve status*: If the check box is selected the control valve is working unless the vehicle does not brake pneumatically.
- *Brake regime*: For each vehicle a different braking regime can be selected (G or P); on the top of the window there is the possibility to choose the brake regime and apply it to all the vehicles of the configuration, in this case all the changes done on the brake regime and on the columns filling tile 95% and 100% will be lost.
- *Filling time 95% - 100% [s]*: in these columns the shape of the limiting curve can be changed. The limiting curve is a third order polynomial function defined by three points: the end of the inshot function, the time to reach the 95% of the maximum experimental pressure end the time to reach the 100% of the maximum experimental pressure.
- *Contribution block*: In this column a contribution for the brake block system of each vehicle can be defined, if the value is not 100% the actual braking force is scaled. The block brake, the disk brake and the electrodynamics brake can be used at the same time. In this case the contribution of each braking system can be very useful the nominal braking force. Each braking system can assume a percentage of application more or less than 100%

- *Contribution disk*: In this column the contribution for the disk brake system of each vehicle can be defined, if the value is not 100% the actual braking force is scaled.
- *Gap [mm]*: A positive or negative value can be set for this parameter, a positive value defines the total distance between the buffers on the rear of the vehicle selected and the front buffers of the subsequent vehicle (in this case the draw gear is coupled without preload). A negative value defines the preload of the system and this value corresponds to the relative approach of the draw gears when the buffers are coupled. More details about the implementation of the gap are in the Technical Guide. This data does not come from the database of the project, but it is defined 0 mm as default (More information about this parameter are in the section “BUFFERS AND DRAW GEARS IMPLEMENTATION”).
- *Buffers and draw gears*: In these columns the front and rear buffers of each wagon can be selected.
- *Friction law*: This information comes from the wagon or locomotive project database. This column is referred only to the block friction law. A different friction law for each vehicle can be selected. If it is not selected a block friction law, then the box is filled with “not applicable” and it means that the box can not be used.

In the lower left part of the configuration table there is a button called “reload all”, using it all the configuration is refilled with the values used in the project database. If you want to reload locally (a row) the project database data you have to select again the type of vehicle.

## GENERIC DATA

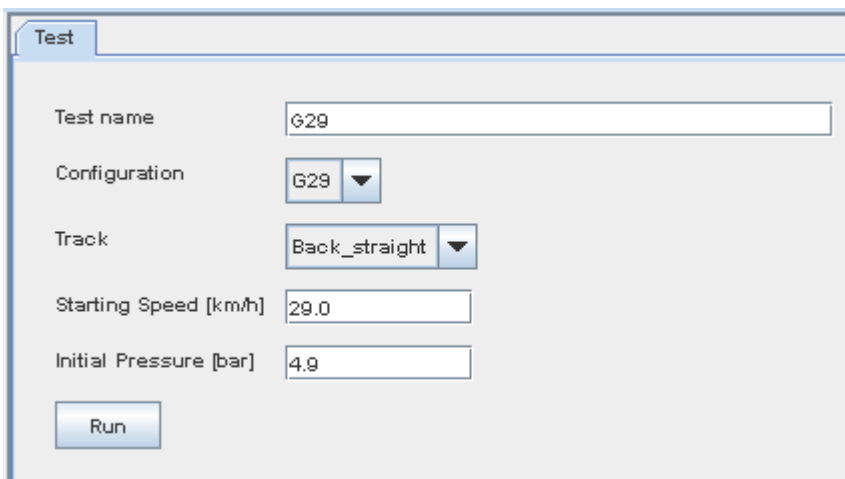
In this section the generic data of the simulation are defined, the parameters to set are:

- *Internal hose coupling diameter [mm]*:
- *Length of hose couplings [m]*:
- *Concentrate pressure loss factor of hose couplings*: this parameter is used for all the vehicles except for the locomotives (the concentrate pressure for the front and rear hose coupling of the locomotive is defined in the locomotive window).
- *Sampling rate data writing [s] (Advanced option)*: Here the sampling rate can be defined, the proposed value is 0.1 seconds.
- *Environmental temperature [K] (Advanced option)*: Atmospheric temperature
- *Roughness of brake pipe [mm] (Advanced option)*: Usually for steel pipe 0.0046 mm

Train Configuration	Generic Data
Internal hose couplings diameter [mm]	<input type="text" value="28.1"/>
Length of hose couplings [m]	<input type="text" value="1.4"/>
Concentrated pressure loss factor of hose couplings	<input type="text" value="2.4"/>
<input type="button" value="Advanced Options"/>	
Sampling rate data writing [s]	<input type="text" value="0.1"/>
Environmental temperature [K]	<input type="text" value="293.15"/>
Roughness of brake pipe [mm]	<input type="text" value="0.046"/>

## TEST

By the test window the simulation can be performed. The configuration of the train that the User wants to simulate must be already defined, along with the track, the starting speed and the initial pressure. By clicking the run button the GUI checks the inputs and runs the simulation. If there are some incompatibilities in the provided input data a warning window appears and the simulation is stopped.



The screenshot shows a window titled "Test" with the following fields and controls:

- Test name:
- Configuration:  ▼
- Track:  ▼
- Starting Speed [km/h]:
- Initial Pressure [bar]:
- Run:



## RESULT MODE

In the result mode the results can be analyzed and exported, this mode extracts the data from the [test name].mat file, in the folder “Results”, using the extractor software DisplyAscii.exe.

The parameters stored in the [test name].mat file are:

- Time [s]
- Speed [km/h]
- Longitudinal Forces [kN]
- 10m Longitudinal Forces [kN]
- Braking Forces [kN]
- Locomotives traction forces [kN]
- Accelerations [m/s<sup>2</sup>]
- Relative displacements [m]
- Relative velocity [m/s]
- Instantaneous friction coefficient
- Brake force divided weight force
- Brake cylinder pressure [bar]
- Brake pipe pressure [bar]
- Speed of the air in BP [m/s]
- Flow mass coefficient [kg/s]
- Braking Energy [kJ]

In order to know when the [test name].mat file is generated and what was the general conditions of the simulation (date, configuration, manoeuvres, tests) a .txt file called [test name]\_info.txt is generated. This file is located near the [test name].mat file.

## AVAILABLE GRAPHS

In the upper part of the first column there are the graph data to set. A maximum of three graphs can be showed, the settings of each graph are in three different tabs as showed in Fig. 27.

If the compare option is selected, a test (of the same project) can be compared in the same graph window. The test compared must have the same number of vehicle of the master test plotted, otherwise an error message occurs.

Only the tests of the same project (with the same number of vehicles) can be showed at the same time in the three graph windows.

When a graph is displayed, in the right column the list of the vehicles used in the simulation appears; by clicking on the vehicle the corresponding data can be showed or hided. If more then one graph is showed or the compare option is active, in the list doesn't appear the name of the vehicle but only the position because the configurations could be different (the check is done only on the number of elements). In order to know the corresponding vehicle name for each data in each graph the legend must be used.

More options on the graph visualization are selectable with a right click on the graph. Also a zoom on the graph can be performed selecting a square on it.

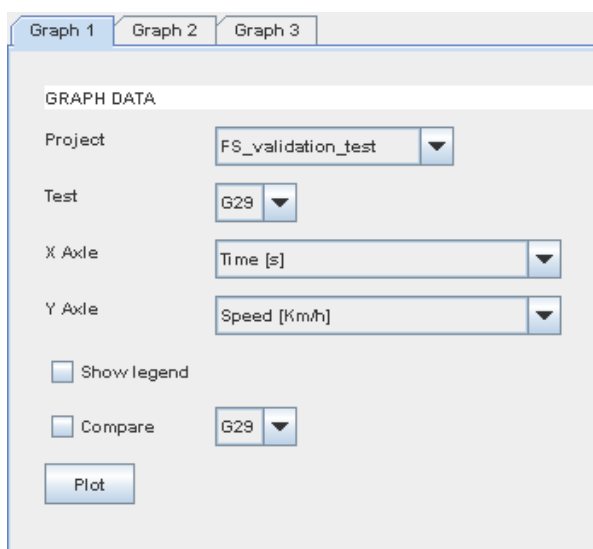


Fig. 27 Graph settings

## EXPORT DATA

The simulation data can be exported in two ways:

- Export the plot data
- Export simulation data

If one or more graph are set the data (contained in the graph) can be exported choosing only the sampling rate, the file extension and the folder in which the User wants save the data (see Fig. 28).

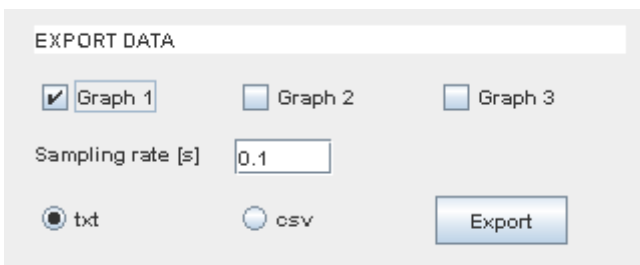


Fig. 28 Graph exporting data

A full customized exportation can be set in the lower part of the first column (see Fig. 29), in this case the positions and the parameters can be selected too.

The sampling rate selected must be greater then or equal to the sampling rate used during the simulation and only multiple of the simulation value can be set.

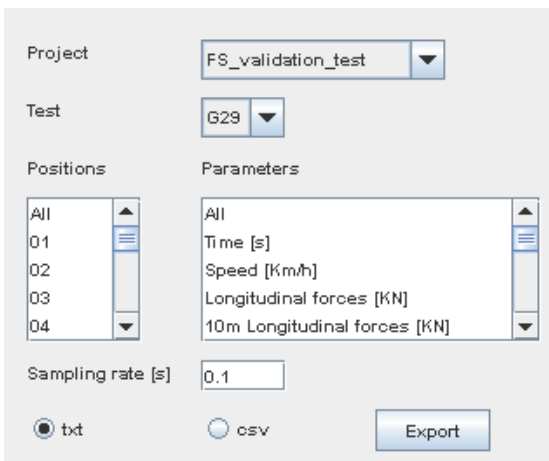


Fig. 29 Export settings

# TECHNICAL GUIDE

In this section the physical equation used to simulate the pneumatic system, the brake models and the buffers and draw gears behaviour are described.

## PNEUMATIC MODEL OF THE TRAIN BRAKE

Fig. 30 shows the basic structure of the pneumatic brake system and in Fig. 31 it is shown the structure of the model implemented. The brake components modelled in TrainDy software, in order to simulate the real brake system behaviour, are:

- Brake pipe
- Brake valve
- Control valve
- Acceleration chambers
- Auxiliary reservoir
- Brake cylinders

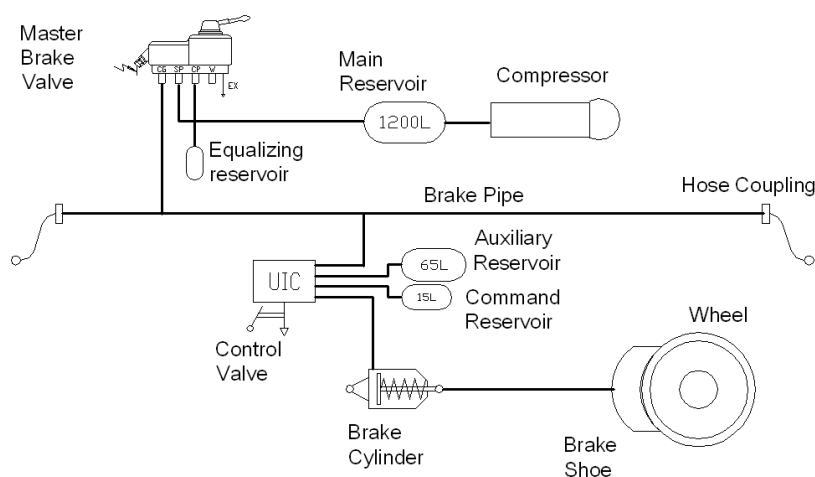


Fig. 30 Basic structure of the pneumatic brake system

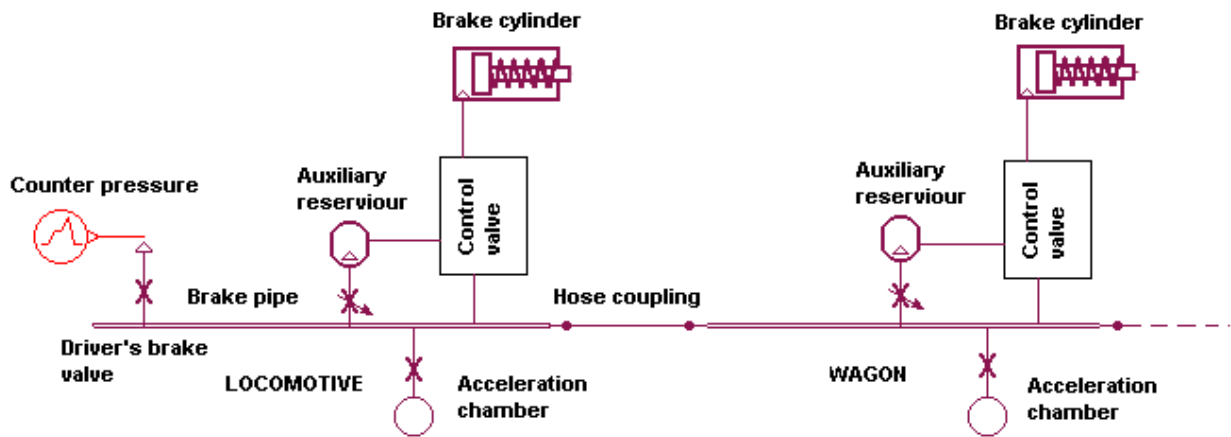


Fig. 31 Structure of the implemented model

Below it will be described the TrainDy models used to emulate these components.

## BRAKE PIPE

The main brake pipe is modelled by a circular pipe with variable cross section (quasi mono-dimensional model) from which air can be blown in or spilled out at the ends or at the side wall of each control volume. This allows for example the simulation of “end of train devices” or distributed braking devices, both for long trains with more than one loco, see Fig. 30. The pipe has a constant diameter within each vehicle and a diameter reduction, that simulates the hose couplings between two consecutive vehicles.

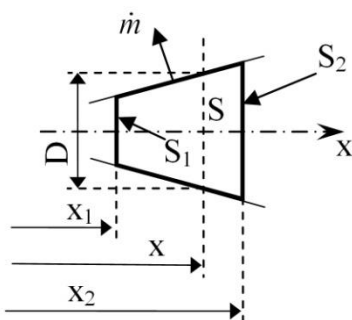


Fig. 32 Variable cross section pipe

From the conservation of mass and energy and the balance of momentum within the above hypotheses, the governing equations become:

$$\begin{cases} \frac{\partial \rho}{\partial t} + u \frac{\partial \rho}{\partial x} + \frac{\rho}{S} \frac{\partial (uS)}{\partial x} = -\frac{\dot{m}}{Sdx} \\ \frac{\partial u}{\partial t} + \frac{1}{\rho} \frac{\partial p}{\partial x} + u \frac{\partial u}{\partial x} = \frac{\tau}{D} + \frac{u}{\rho} \frac{\dot{m}}{Sdx} \\ \frac{\partial q}{\partial t} + u \left( \frac{\partial q}{\partial x} + r \frac{\partial T}{\partial x} \right) + r \frac{T}{\rho S} \frac{\partial (\rho u S)}{\partial x} = 4 \frac{\phi_T}{\rho D} - \frac{\tau u}{D} - \frac{\dot{m}}{Sdx} \frac{1}{\rho} \left[ (c_v + r) T_l + \frac{1}{2} u_l^2 - q \right] \end{cases}$$

**Eq. 2 Conservation of mass and energy and the balance of momentum into the brake pipe**

Where  $\rho$  is the density,  $u$  axial velocity,  $p$  pressure,  $T$  temperature and all of them must be considered as mean values on the general cross-section  $S$  of diameter  $D$  and abscissa  $x$ ;  $q$  is the

specific energy,  $c_v$  specific heat at constant volume,  $\tau = -\text{sgn}(u) \cdot \left( f + K \cdot \frac{D}{dx} \right) \cdot \frac{u^2}{2}$  takes into account the dissipative sources (there,  $f$  is the distributed coefficient of pressure loss,  $K$  concentrated coefficient of pressure loss and  $\text{sgn}(\cdot)$  sign function);  $\phi_T$  is the exchanged thermal flux,  $r$  gas constant,  $\dot{m}$  in-flow or out-flow mass flux; and, finally, subscript  $l$  refers to lateral quantities, which has to be computed by imposing the right boundary conditions.

The integration of Eq. 2 needs both suitable boundary conditions and initial conditions for all the components (accelerating chambers, master brake valve and auxiliary reservoirs) since all of them determine some of the source terms of Eq. 2. In the pneumatic module of TrainDy, the “real” pneumatic braking system of a train of Fig. 30 is modelled as a variable cross-section pipe with some lateral nozzles, as reported in Fig. 31.

## MASTER BRAKE VALVE

The master brake valve is modelled as a nozzle with an equivalent diameter, tuned on experimental data. During braking, the upstream pressure of the nozzle is the pressure of the brake pipe, while its downstream pressure is either the atmospheric pressure, for an emergency braking, or, for a service braking, the pressure of the pilot chamber of the master brake valve. In

the latter case, the downstream pressure is time-variable, according to a law provided by the constructor of the valve. On the other hand, during a releasing, the upstream pressure of the nozzle is the pressure of the pilot chamber, yet provided by the valve constructor, whereas the downstream pressure is the pressure of the brake pipe (Fig. 32 shows an example).

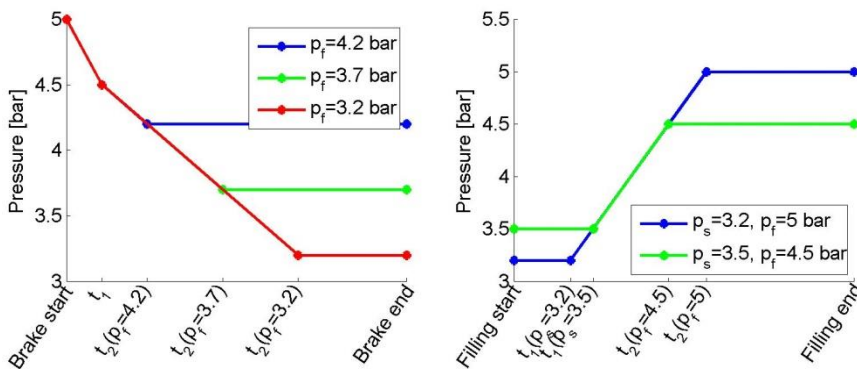


Fig. 33 Time evolution of the counter-pressure in the pilot chamber for a service braking.

Since the real pneumatic circuits used for the three previous manoeuvres are different, there are also three equivalent diameters of the nozzle that simulates the same master brake valve; fortunately these diameters need to be determined only once and they do not depend on the length of the train. As well known, the pressure drop close to the master brake valve depends much from the equivalent nozzle diameter and by this way it is possible to tune it easily. All the above considerations along with the classical equations of a nozzle make it easy to compute the mass flux terms of Eq. 2, corresponding to the sections  $S(x)$  where the master brake valves are located.

The determination of the mass flux  $\dot{m}$ , flowing through the equivalent cross-section, is obtained using the relation Eq. 3:

$$\dot{m} = C_q \cdot S_{DBV} \cdot C_m \cdot \frac{P_u}{\sqrt{T_u}},$$

Eq. 3 Mass flux equation used for the master brake valve

where SDBV is the equivalent nozzle cross-section, whereas the flow coefficient  $C_q$  and the mass flow parameter  $C_m$  are given, respectively by:

$$C_q = 0.8414 - 0.1002 \cdot P_r + 0.8415 \cdot (P_r)^2 - 3.9 \cdot (P_r)^3 + 4.6001 \cdot (P_r)^4 - 1.6827 \cdot (P_r)^5$$

$$C_m = \begin{cases} \sqrt{\frac{2\gamma}{r(\gamma-1)}} \sqrt{(P_r)^{2/\gamma} - (P_r)^{\gamma+1/\gamma}} & \text{(sub-sonic regime)} \\ \sqrt{\frac{\gamma}{r} \left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}}} & \text{(sonic regime)} \end{cases}$$

$P_d$  and  $P_u$  are, respectively, the downstream and upstream pressures for the nozzle,  $P_r = P_d/P_u$ ,  $T_u$  is the upstream temperature and  $\gamma$  is the ratio of the constant pressure and volume specific heats that for air is 1.4. Then the lateral flux speed used in Eq. 2 is given by:

$$u_l = \frac{\dot{m}}{\rho_d \cdot S_{DBV}}$$

#### Eq. 4 Flux speed

where  $\rho_d$  is the downstream air density in the equivalent model.

## CONTROL VALVE

To simulate the distributor valve behaviour, two functions are introduced in the model: the transfer function (see the example in Fig. 34) and the limiting curve (see the example in Fig. 35). These functions and so the distributor valve behaviour are defined by the fiche UIC 540-O and 547-O.

Once computed the pressure in the main brake pipe, the transfer function of the distributor valve together with the limiting curve of the braking cylinder are used in order to compute the time evolution of the pressure in the braking cylinder. In the very beginning of the braking phase the



filling of the braking cylinders is performed only mathematically because, according to the fiche UIC 540-O, it does not depend on the “braking position” (this first behaviour is called “inshot function”); then the pressure in the braking cylinders is controlled in time and depends not only on the pressure in the brake pipe but also on the limiting curve. Practically this implies that the pressure in the braking cylinder is, respectively, the minimum or the maximum between the limiting curve and the transfer function for braking or releasing manoeuvres.

The transfer function and the limiting curves are different for braking and releasing as showed in the figures below. The transfer function is defined a sequence of straight lines instead the limiting curve is defined by a third order polynomial function so to fix the shape must be defined three points, for braking these points are:

- The starting point (end of the “inshot function”: initial filling behaviour imposed, see UIC 540-O).
- Time to reach the 95% of the maximum pressure.
- Time to reach the 100% of maximum pressure.

For releasing the three points are:

- The starting point (3.8 bars)
- Time to reach the 110% of the minimum pressure
- Time to reach the minimum pressure

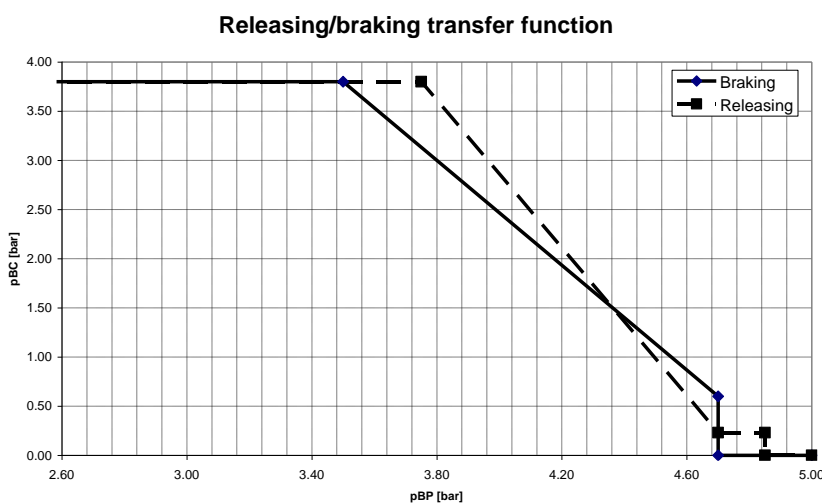


Fig. 34 Releasing and braking transfer functions

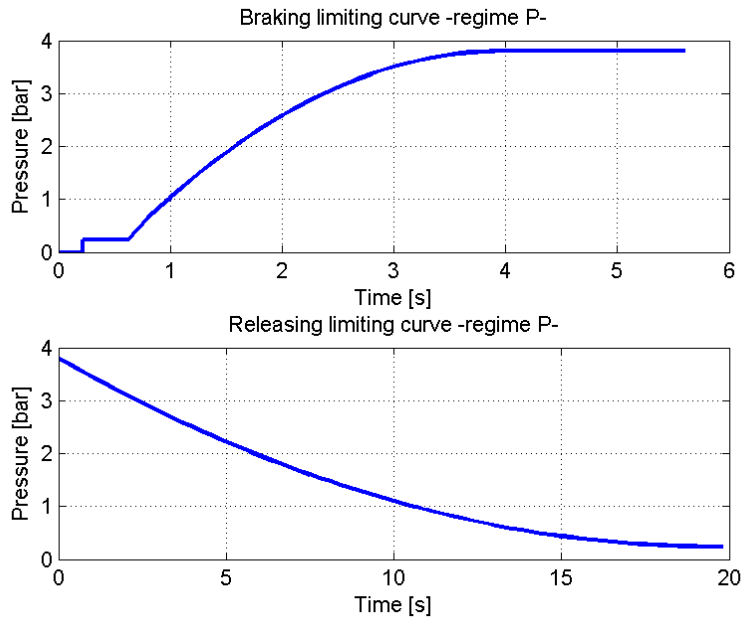


Fig. 35 Releasing and braking limiting curves in P regime

## ACCELERATION CHAMBERS

During a braking, the accelerating chambers of the control valves are modelled as lateral small volumes connected to the brake pipe by means of nozzles of equivalent constant diameters. The pressure of the air in these volumes changes according to the inflow flux mass, holding constant, for hypothesis, the temperature of the air in the volume of the accelerating chamber:

$$\frac{dP_{AC}}{dt} = \frac{\dot{m}}{m_{AC}} P_{AC}$$

Where:

$\frac{dP_{AC}}{dt}$  = pressure gradient in the AC

$\dot{m}$  = inflow flux mass

$m_{AC}$  = Air mass in the AC

$P_{AC}$  = Pressure in the AC

In this case, the upstream pressure of the nozzle is the pressure in the brake pipe and the downstream pressure is the pressure of the air in the small volume; after a certain time period, the pressure of the air in the small volume is bigger than the pressure in the brake pipe and the air goes back, from the small volume to the brake pipe. By this way, using equations Eq. 3 and Eq. 4 adapted to the accelerating chambers, it is easy to compute the mass flux terms of Eq. 2 corresponding to the sections  $S(x)$  where the accelerating chambers are located. Both the nozzle diameter and the capacity of the lateral volume are tuned on experimental measurements; it is worth to underline that this tuning occurs only one time and it neither depends on the manoeuvre (emergency or service braking) nor on the train length.

## AUXILIARY RESERVOIR

The auxiliary reservoirs are modelled as lateral big volumes connected to the brake pipe by means of a nozzle with variable diameter and are communicating with the brake pipe only during releasing. From the modelling point of view the auxiliary reservoirs are similar to the accelerating chambers except for the fact that their initial pressure depends on the previous braking manoeuvres, because the air in the auxiliary reservoirs has been used to fill the braking cylinders.

## BRAKE SYSTEM MODELS

Different brake systems are implemented in TrainDy software, these are:

- **Block brake system:** it is applied to wagons and locos and it can use Bg and Bgu shoes type. The forces applied from the shoes to the wheel are computed by the fiche UIC 544-1 and the braked forces by a friction law. The friction law implemented in TrainDy are described in the section “*Friction Laws*”. The procedure implemented in TrainDy, based on UIC 544-1 fiche, is described in the section “*Computation of braked forces in a brake block system and comparison with UIC 544-1*”.
- **Disk brake system:** it can be applied to wagons and locos. The shoes – disk forces are computed from the braked weight by the Eq. 5 (UIC 544-1). The friction law is constant and it can be set in the GUI.

$$B[t] = h \cdot F_b [kN] \cdot \mu_m \frac{r_m}{r_r} = h \cdot F_c$$

### Eq. 5 Determination of braked weight from disk brake characteristics

Where:

$h$ : Conversion factor 1.18 [t/kN]

$F_c$ : Brake force [kN]

$F_b$ : Total shoes - disks forces [kN]

$\mu_m$ : Friction coefficient (it can be considered as a constant) 0.35

$r_m$ : Disk radius measured from the shoes

$r_r$ : Wheel radius measured from the wheel rail contact point

The  $F_b$  value can be computed by the Eq. 6

$$F_b = P[\text{bar}] \cdot S \cdot i \cdot \eta$$

**Eq. 6 Computation of total shoes - disk forces**

The braking force  $F_f$  is computed by the disk – shoes friction coefficient as showed in the eq\_\_\_\_.

$$F_c = \mu_m \frac{r_m}{r_r} F_b$$

**Eq. 7 Braking force**

Where: *Pressure in the brake cylinders*

$P[\text{bar}]$ :

$S [\text{dm}^2]$ : *Total section of brake cylinders*

$\eta$ : *Rigging efficiency*

$i$ : *Rigging ratio*

TrainDy needs all the values above in way to compute the braking force of each vehicle, however if you know the braked weight you can compute the X value in the Eq. 8 and by it the braking forces can be computed by the Eq. 9.

$$X = S \cdot i \cdot \eta \cdot \mu_m \frac{r_m}{r_r} = \frac{B}{h \cdot P_{load}}$$

**Eq. 8 Correlation factor computed from the braked weight and load pressure**

$$F_c [\text{kN}] = X \cdot P [\text{bar}]$$

**Eq. 9 Computation of the brake force**

The maximum value of the brake forces, for a block brake or a disk brake system, can be function of the vehicle total mass, by an empty load system or by an auto continuous system. These systems are described in the section “*Brake controls*”.

- **Electrodynamic brake system:** It can be applied only to the locomotives and it computes the brake forces as function of time and speed.

In order to compute the brake force it is necessary do define some inputs in the model, these are:

- Electrodynamic brake characteristic: it is a Brake force – speed function defined by the loco technical guide, in Fig. 36 it is showed an example.

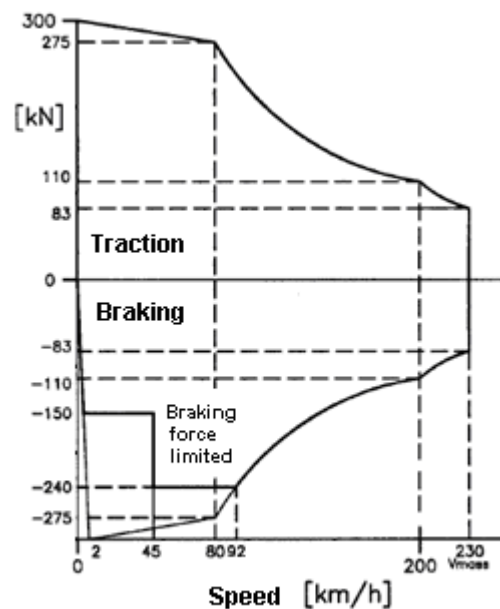
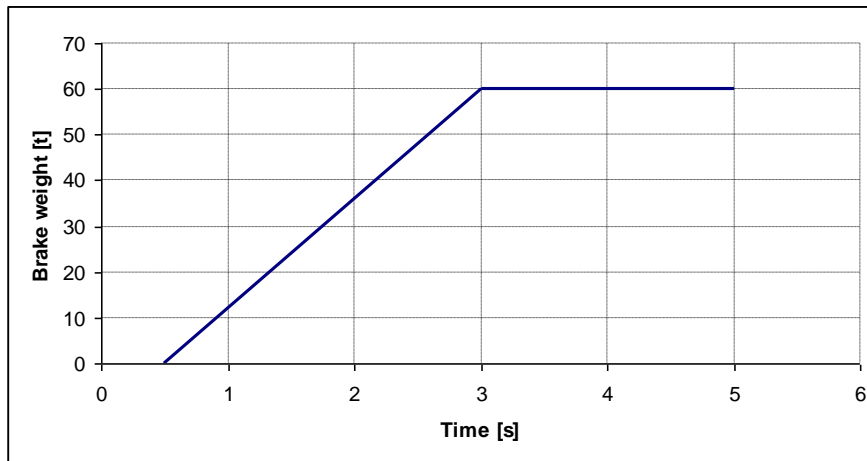


Fig. 36 Electrodynamic brake - traction characteristic

- Increasing function: this is a time – electrodynamic force; the braking (traction) force applied is the minimum (in module) force between this function and the speed – force function.
- Insertion and removal gradients in kN/s: the same values are used both for traction and for braking. If these values are set (different from zero), the Force vs Time function is skipped.



## COMPUTATION OF BRAKED FORCES IN A BRAKE BLOCK SYSTEM AND COMPARISON WITH UIC 544-1

*TrainDy* uses the UIC 544-1 as reference to compute the normal force acting between wheel and shoe during a braking. According to this standard, the total normal force is computed as follows:

$$\sum F_{dyn} = (F_t \cdot i_G - i^* \cdot F_R) \cdot \eta_{dyn}$$

**Eq. 10** Total normal force acting among wheel and shoe

$$F_t = p_{BC} \cdot S_{BC} - F_F$$

**Eq. 11** Force applied by the brake cylinder

where:

$p_{BC}$  is the pressure in the brake cylinder

$S_{BC}$  is the transversal section of the brake cylinder

$F_F$  is the brake rigging return force (usually 1.5 kN)

$F_R$  is the counteracting force of the brake rigging regulator (usually 2 kN) -slack adjuster-

$i_G$  is the total multiplication ratio of the brake rigging

$i^*$  is the multiplication ratio after the central rigging (4 for two-axle vehicles and 8 for bogie vehicles)

$\eta_{dyn}$  is the main efficiency of the rigging (for standard rigging 0.83).

The braking force is computed by the friction coefficient  $\mu$  (see FRICTION LAWS) and the total wheel shoes contact forces as showed in the Eq. 12

$$F_f = \mu \sum F_{dyn}$$

#### Eq. 12 Braking force computation

*TrainDy* needs to know all above parameters for an appropriate computation. Sometimes it can happen that there are some unknowns, e.g. it is known only the braked weight or  $\sum F_{dyn}$ ; in order to provide a computation also for these conditions, a non-linear procedure has been set. This procedure determines the total multiplication ratio of the rigging ratio considering  $S_{BC} = 707 \text{ cm}^2$  for two axle vehicles and  $S_{BC} = 1295 \text{ cm}^2$  for bogie vehicles (according to UIC 544-1).

The relation between the total normal force and the braked weight is defined in the UIC 544-1 by the Eq. 13.

$$B[t] = \frac{K[-] \cdot \sum F_{din} [kN]}{9,81[m/s^2]}$$

#### Eq. 13 Relation between the braked weight and the total normal force acting between wheel and shoe

Where K is a correlation factor defined by the Eq. 14.

$$K = a_0 + a_1 \cdot \frac{F_{dyn}}{N_{bs}} [kN] + a_2 \cdot \left( \frac{F_{dyn}}{N_{bs}} \right)^2 [kN] + a_3 \cdot \left( \frac{F_{dyn}}{N_{bs}} \right)^3 [kN]$$

#### Eq. 14 Correlation factor

$$F_{dyn} = \frac{\sum F_{din}}{N_{bs}}$$

Where:



	$a_0$	$a_1$	$a_2$	$a_3$
$K_{Bg}$	2.145	$-5.38 \cdot 10^{-2}$	$7.8 \cdot 10^{-4}$	$-5.36 \cdot 10^{-6}$
$K_{Bgu}$	2.137	$-5.14 \cdot 10^{-2}$	$8.32 \cdot 10^{-4}$	$-6.04 \cdot 10^{-6}$

$K_{Bg}$  e  $K_{Bgu}$  referred respectively to Bg and Bgu shoes.

By the Eq. 10, Eq. 11, Eq. 13, Eq. 14, it is possible to define a non linear relationship among the braked weight and the rigging ratio.

The procedure used to compute the normal brake force ( $\sum F_{dyn}$ ) from the braked weight, solving the non linear equation, is not approximate. This can be demonstrated in a simple way:

- Using general physical parameters of a block brake system, the brake force and the braked weight are computed by the above equation.

The parameters in the Tab. 1 are taken from the fiche 5-441 except for the Test 4 in which the brake cylinder cross section is chosen far (about the double) from the standard value in way to have an idea of the maximum error that could be occur.

The *TrainDy* function used to compute the braked weight and the normal brake force is:

```
[B,k,Ft,SFd] = UIC_BW(Ff,Fr,I,na,nbs,Pbrake,rtim,S,typeSh)
```

The meaning of the input and output parameters is:

```
B = braked weight
k = UIC corrector factor
Ft = Force applied by the brake cylinder
SFd = Brake force

Ff, Fr = counteracting forces
I = rigging ratio
na = number of axes
nbs = number of bring shoes (na*4)
Pbrake = braking target pressure
rtim = rigging efficiency
S = brake cylinder cross section
```

typeSh = type of shoes

	Test 1	Test 2	Test 3	Test 4
Shoe	Bg	Bg	Bgu	Bgu
N° axle	2	4	4	2
$\eta_{dyn}$	0.83	0.83	0.83	0.83
ig	11.14	11.76	12.68	6
SBC [cm <sup>2</sup> ]	707	1295	1295	1500
P [bar]	1.5	3.8	1.3	3.8
$\sum F_{dyn}$ [KN]	<b>77.54</b>	<b>452.41</b>	<b>148.11</b>	<b>269.75</b>
B [t]	13.37	51.93	26.08	30.75

**Tab. 1 Direct computation of the brake force and braked weight from physical block brake system parameters**

- Using the braked weight, computed in the Tab. 1, the normal brake force is computed by the *TrainDy* non linear procedure, then this force is compared with the brake force computed in the table above with the formulation described in UIC 544-1.

The function used in *TrainDy* code is “BWc2”:

[Bcomp, k, I, S, SFd] = BWc2 (B, typeSh, na, nbs, Ff, Fr, Pbrake)

The meaning of the input and output parameters is:

Bcomp = Braked weight computed by the k and SFd  
k = UIC corrector factor (from the non linear procedure)  
I = rigging ratio (from the non linear procedure)  
S = brake cylinder cross section (imposed in function of na)  
SFd = Brake force (from the non linear procedure)

B = braked weight  
typeSh = type of shoes  
na = number of axes  
nbs = number of bring shoes (na\*4)  
Ff, Fr = counteracting forces  
Pbrake = braking target pressure  
(The rigging efficiency is fixed to 0.83)

	Test 1	Test 2	Test 3	Test 4
Shoe	Bg	Bg	Bgu	Bgu
N° axle	2	4	4	2
P [bar]	1.5	3.8	1.3	3.8
B [t]	13.37	51.93	26.08	30.75
$\sum F_{dyn}$ [KN]	<b>77.54</b>	<b>452.41</b>	<b>148.11</b>	<b>269.75</b>

**Tab. 2 Non linear computation of the brake force from the braked weight**

As showed in the above table, *TrainDy* is cable to compute the braking force using different type of inputs: the brake parameters (*direct* computation) or the braked weight (*non linear* computation). However, in each case, the type and number of shoes, the target pressure and the  $F_f$ ,  $F_r$  parameters must be known.

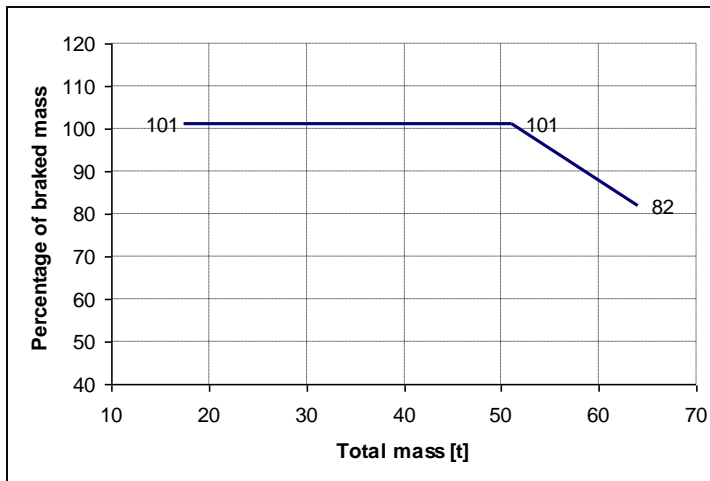
## BRAKE CONTROLS

In this section, the systems implemented in *TrainDy* that define the braked weight as function of the wagon mass are described. These systems can be applied only at the wagons because the mass of the loco is constant.

- *Empty-load system*: This system is characterized by two brake forces and a changing mass. If the vehicle total mass is bigger or equal then the changing mass, it brakes with the bigger brake force; instead, if the vehicle total mass is lower then changing mass the vehicle brakes with the lower brake force.

To define the upper and lower brake force you have to set the braked weight or the empty load pressure along with the brake system characteristics, the cylinder section and the rigging ratio.

- *Auto continuous system*: This system requires a relationship among total mass and percentage of braked mass as showed in Fig. 37. It can be set by a series of points that define a series of straight lines.



**Fig. 37 Auto continuous system characteristic**

When a block brake system is used, it must be defined a friction law between the shoe and the wheel. The friction coefficient depends from speed, specific pressure and atmospheric conditions. The friction laws implemented in the software as default are described in the section “*Friction laws*”.

## FRICION LAWS

The friction law implemented in the software as default are :

- *Karwatzki*: it is an analytical law, it takes in account only the block wheel force and the relative speed. The Eq. 15 describes the Karwatzky friction law painted in Fig. 38.

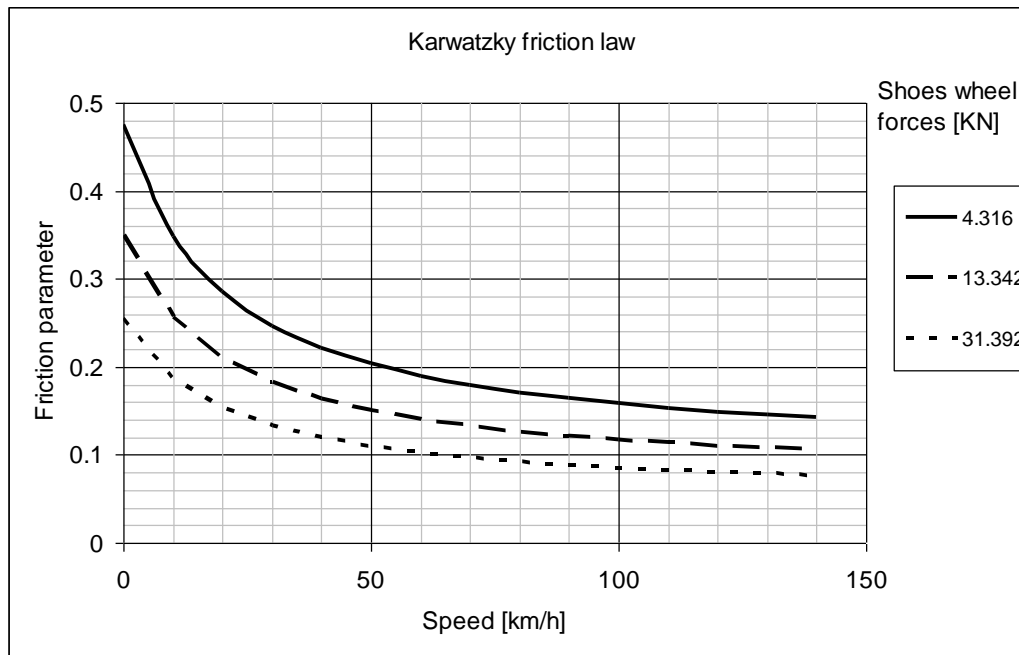


Fig. 38 Karwatzki friction law

$$\mu(V, F_k) = 0.6 \cdot \frac{\frac{16}{80} F_k + 100}{\frac{80}{g} F_k + 100} \cdot \frac{V + 100}{5V + 100}$$

Eq. 15 Karwatzki friction law

Where:

$F_k$  [kN]: is the total block – wheel force

$V$  [km/h]: speed

$g = 9.81 \text{ m/s}^2$

- **BZA**: it is a polynomial friction law, the coefficient of the polynomial depends from the initial speed, the current speed and the load position (i.e. the mass carried on the wagon).

This friction law is characterized by three limitations:

- It is not possible to compute the friction law with speed bigger than 60 km/h
- This law is able to emulate only the Bg shoes friction behaviour.
- It can not be used an auto continuous brake system (it works only in empty or load condition)

$$\mu(V_a, V, Load) = a \cdot \left(\frac{V}{100}\right)^3 + b \cdot \left(\frac{V}{100}\right)^2 + c \cdot \left(\frac{V}{100}\right) + d + e \cdot \left(\frac{100}{V}\right)$$

**Eq. 16 BZA friction law**

Where:

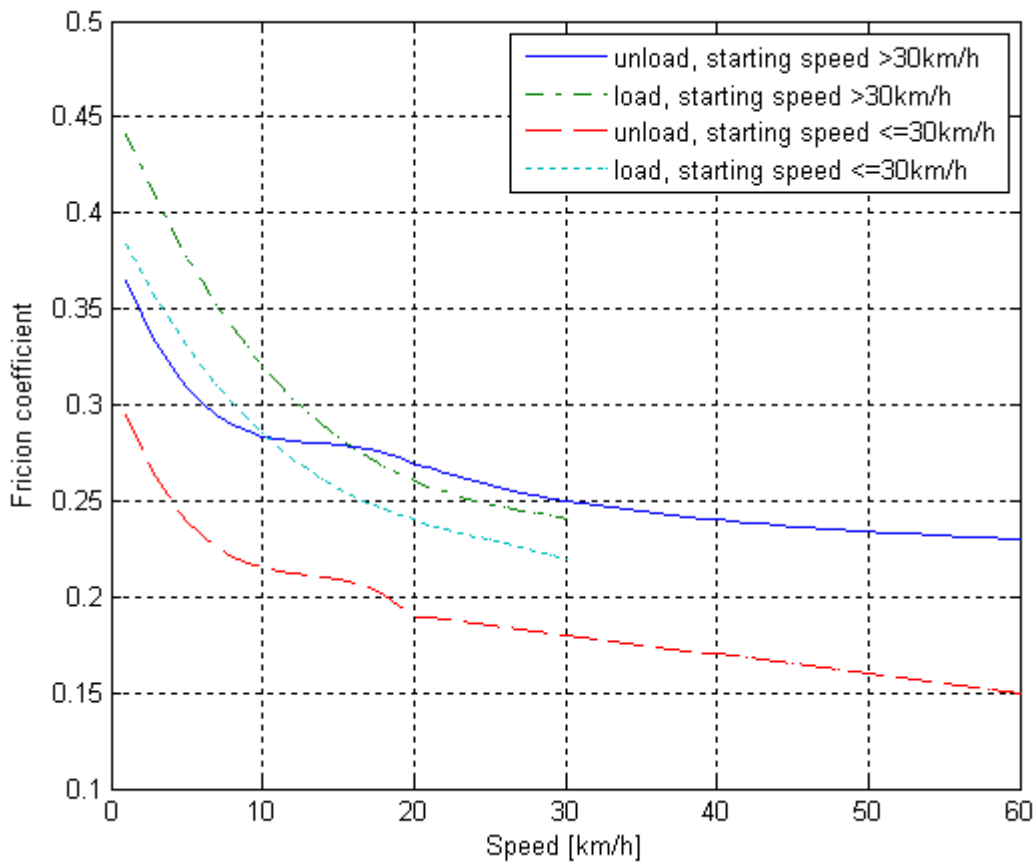
$V_a$  : initial speed [Km/h]

$V$  : current speed [Km/h]

Load: state of the wagon control valve

$a, b, c, d, e$ : Polinomial coefficients

$V_a$	$V$	Load	$a$	$b$	$c$	$d$	$e$
$\leq 30$	$\leq 30$	empty	-6.67	6	-1.93	0.46	0
		load	-7.5	5.75	-1.65	0.4	0
$30 < V_a \leq 60$	$\leq 20$	empty	-36.67	15.33	-2.18	0.385	0
	$20 < V_a \leq 60$		0	0	0	0.21	0.012
	$\leq 20$	load	-43.33	16.67	-2.23	0.315	0
	$20 < V_a \leq 60$		0	0	-0.1	0.21	0



- *OSShd/UIC*: this is an analytical friction law, the friction coefficient is computed in function of the speed and the specific pressure.

$$\mu(V, p_k) = 0.49 \cdot \frac{\frac{875}{g} p_k + 100}{\frac{2860}{g} p_k + 100} \cdot \frac{\frac{10}{3.6} V + 100}{\frac{35}{3.6} V + 100}$$

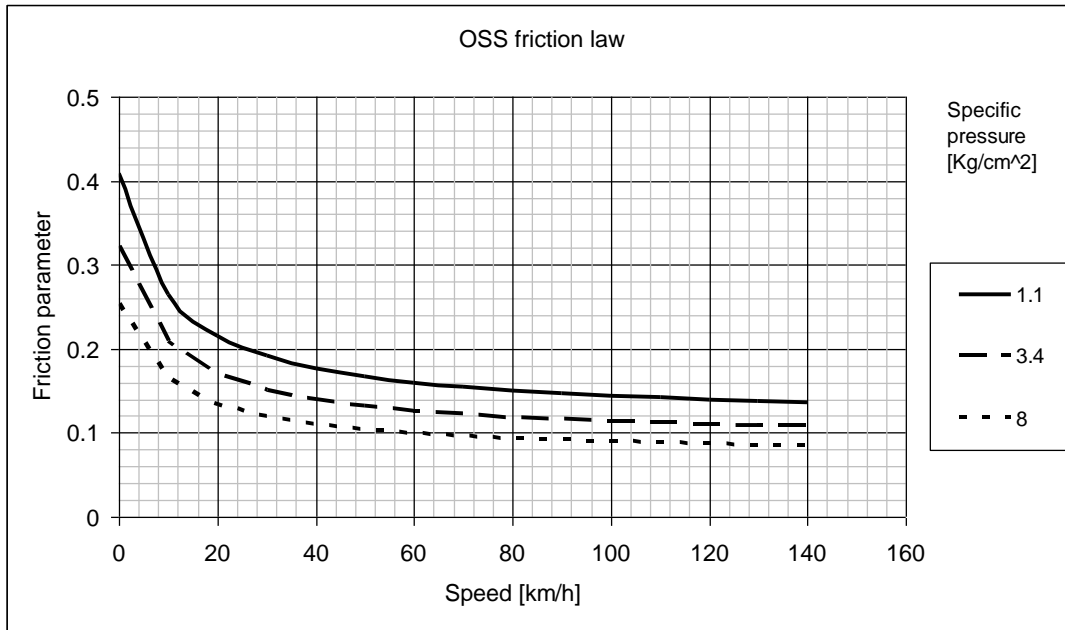
Eq. 17 *OSShd/UIC* friction law

Where:

$p_k$  [N/mm<sup>2</sup>]: specific pressure between shoe and wheel

$V$  [km/h]: speed

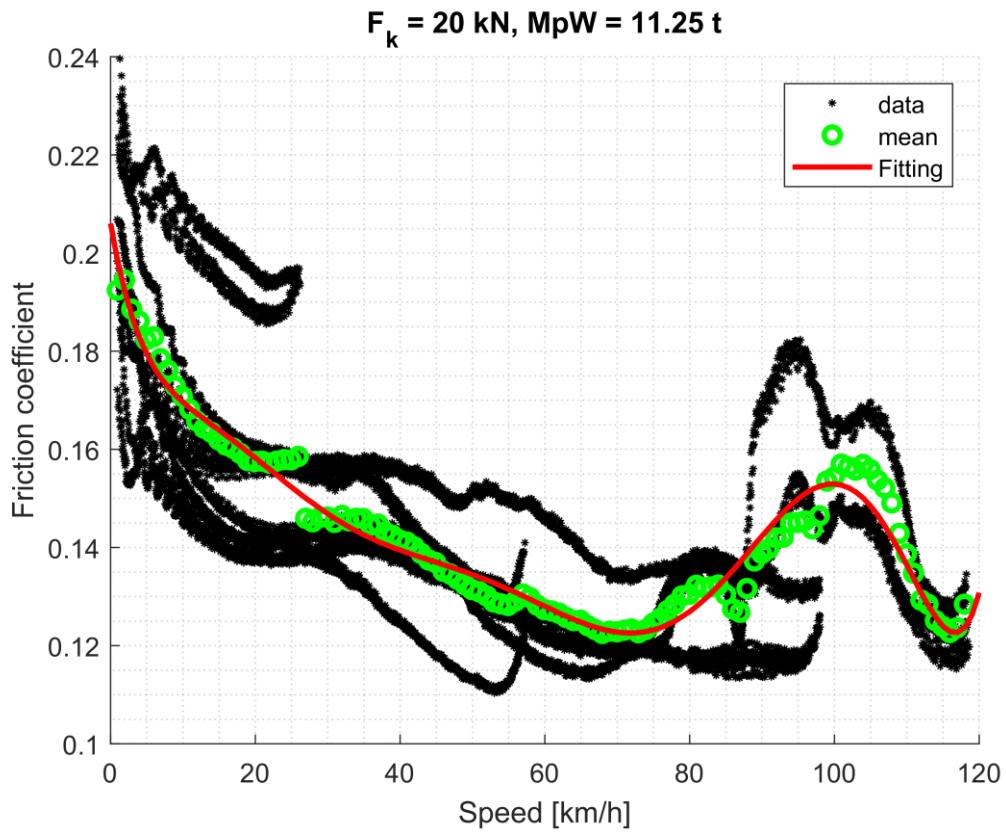
$g = 9.81$  m/s<sup>2</sup>



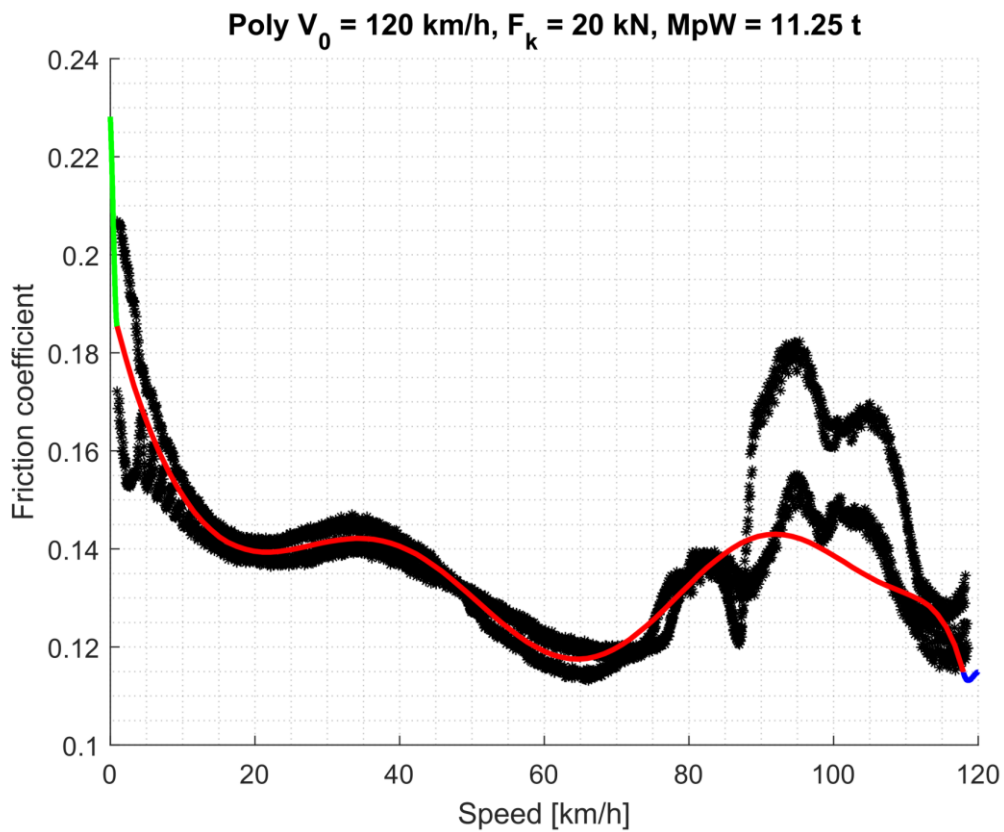
- *LL friction coefficient*: For a more detailed description, see the paper with DOI [10.2174/18744478018120100114](https://doi.org/10.2174/18744478018120100114). There are two methods: LLMethod1 and LLMethod2. The first is computationally quicker but less accurate than the second. The first considers an unique set of polynomial functions from 120 km/ to zero speed, whereas the second employs different polynomial functions for the following starting speeds of braking: 30, 60, 100 and 120 km/h. Both consider the effect of normal force and speed on the friction coefficient.

The following figure shows the friction coefficient evolution for a normal force between the shoe and wheel of 20 kN (corresponding to a mass per wheel of 11.25 t) in agreement with LLMethod1. The figure reports the measurements (data), too. The solid red line reports the weighted data fitting.



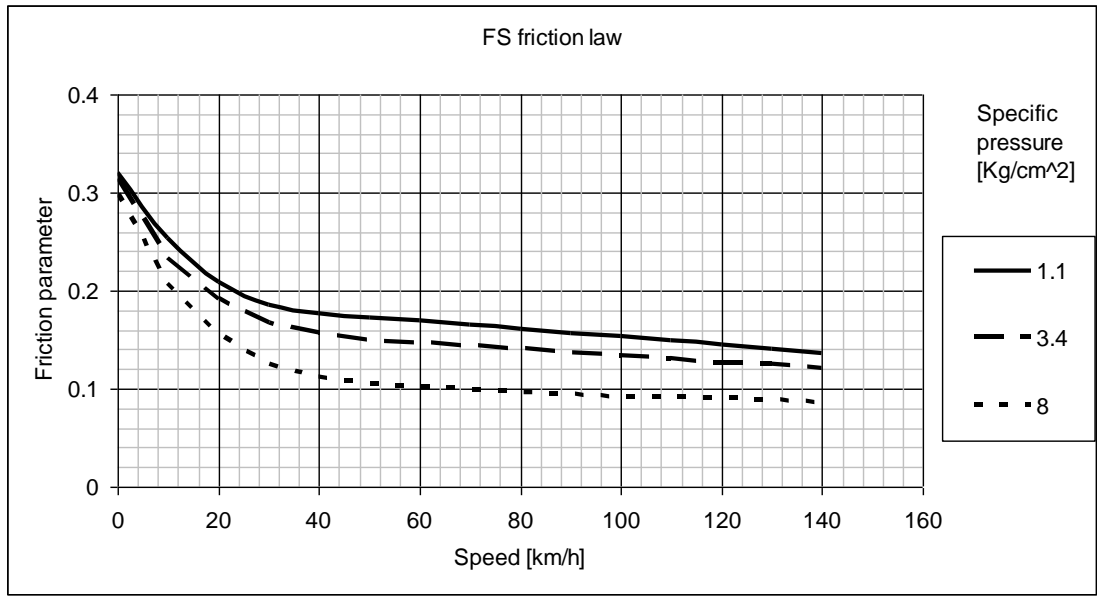


The following figure shows the friction coefficient evolution for a normal force between the shoe and wheel of 20 kN (corresponding, in this case, to a mass per wheel of 11.25 t) in agreement with LLMethod2. The figure shows only the experimental test data with a starting speed of 120 km/h. The extrapolated values are reported in green and blue for low and high speed, respectively.



When the friction law employs the LLMethod2, fixed the wagon mass, a double interpolation is applied: one on the initial braking speed and another (typical also for other experimental friction laws, such as FS) on the normal force. The LLMethod1 only requires the second interpolation. See [10.2174/18744478018120100114](https://doi.org/10.2174/18744478018120100114) for more details.

- *FS*: This friction law came from a look-up table, the friction law is a function of the specific pressure and the speed as showed below.



## BUFFERS AND DRAW GEARS IMPLEMENTATION

In its basic modelling, a train is regarded as series of masses connected by non linear springs, these springs have a behaviour like in Fig. 39.

For buffers, when the approaching speed of two consecutive wagons is more than  $v_{load}$ , the exchanged force follows the loading force-stroke characteristic, instead, when the buffers are compressed, but the wagons are moving away from each other with a speed greater than  $v_{un-load}$  (in absolute value) the exchanged force follows the un-loading force-stroke characteristic. When the relative speed of two consecutive wagons is in the interval between the loading and un-loading speed, the exchanged force is computed by the following formula painted in Fig. 40:

$$F(x, v) = c(v) \cdot F_{un-load}(x) + [1 - c(v)] \cdot F_{load}(x)$$

To the previous force the factor  $c_{damp}^{visc} \cdot v_{rel}$  is added in order to simulate the effect of a viscous damping. When two viscous buffers are coupled, the equivalent viscous damping is computed as the series scheme, when only one coupler has a viscous damping, this is assumed as the viscous damping of the equivalent coupling.

Similar considerations can be done for the draw gears. The force-stroke characteristic can be provided in two different ways:

- a) giving the damping and some points of the loading curve
- b) providing points both for loading and unloading curve

in both cases, the limiting speeds of loading and unloading must be given. TrainDy uses a cubic piecewise interpolation of the input data, which assures slope continuity, in order to improve numerical integration.

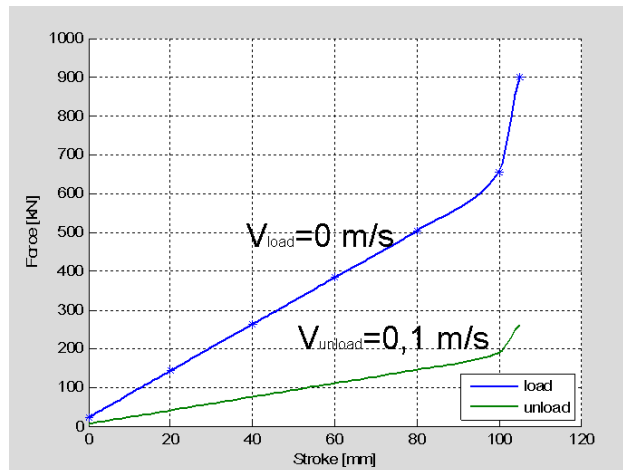


Fig. 39 Generic buffers characteristics

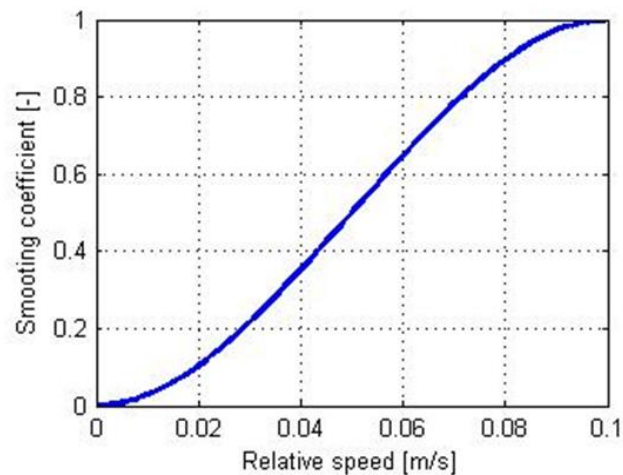
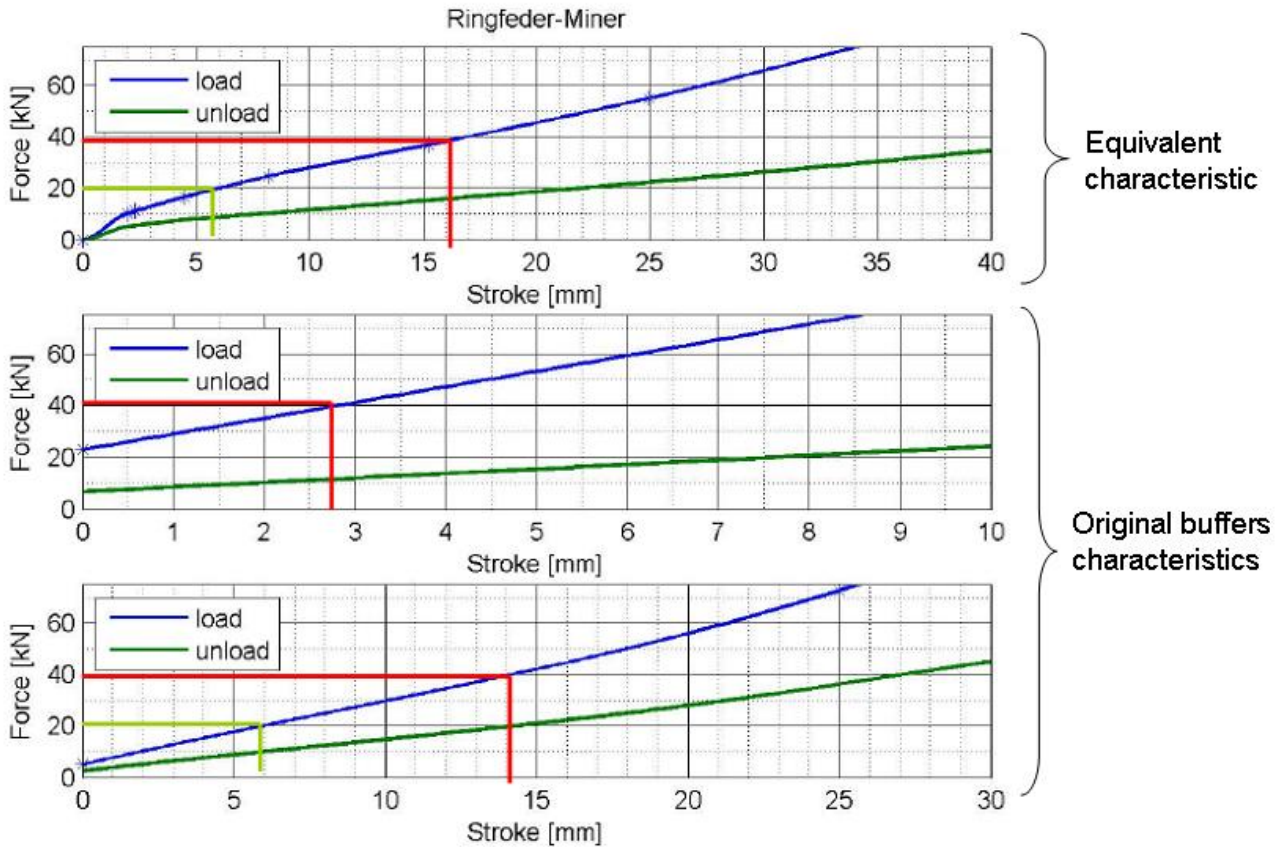


Fig. 40 Smoothing function

The train model uses an equivalent Force-stroke characteristic computed from the Buffers/Draw gear characteristics of two consecutive wagons. The equivalent characteristic of two coupled buffers or draw gears is determined imposing that at each force the equivalent stroke is the sum of the stroke of each Buffers/Draw gear. As a matter of fact, the buffers of two consecutive vehicle, for example, are mounted according to a “spring series scheme”: this means that at the interface they exchange the same force (for mechanical equilibrium) and the corresponding displacement of the couple of buffers is the sum of the displacements of the two buffers, simply. Fig. 41 shows graphically the described method:., considering a compression force of 20 kN, the input buffers characteristics return displacements of 0 mm and of 6 mm, so at 20 kN the equivalent characteristic (of the coupled buffers) has a relative approach of (0 + 6 mm).

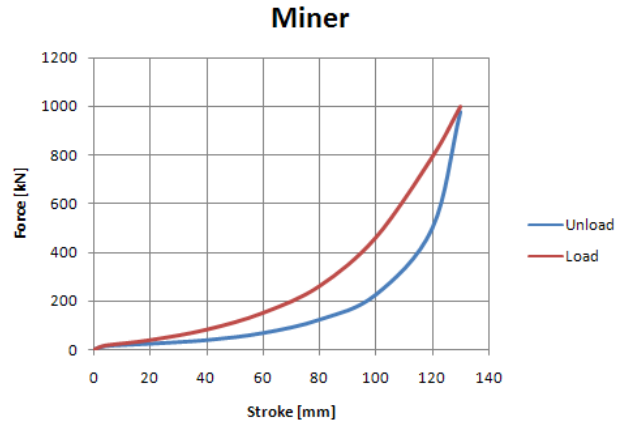
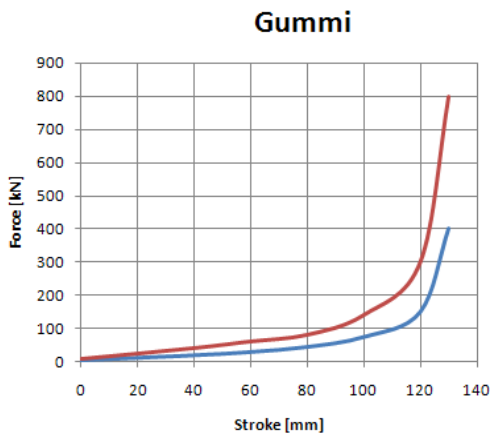
Considering a compression force of 40 kN the input buffers characteristics return displacements of about 2.8 mm and 14 mm, consequently, the equivalent characteristic has 16.8 mm of displacement with a compression force of 40 kN.



**Fig. 41** Equivalent characteristic computation

By the tool "Verify\_bgdg.exe", described in the Software Guide (Buffers and draw gear section), the equivalent characteristics used can be shown.

When the equivalent characteristics of the buffers and draw gears are computed the gap or the preload are considered. In Fig. 42 is showed an example of equivalent characteristics for buffers and draw gears and in Fig. 43 is plotted the equivalent coupling characteristic with no gap (considering that there are two buffers and one draw gear).



a)

(b)

Fig. 42 Equivalent characteristics for buffers (b) and draw gears (a)

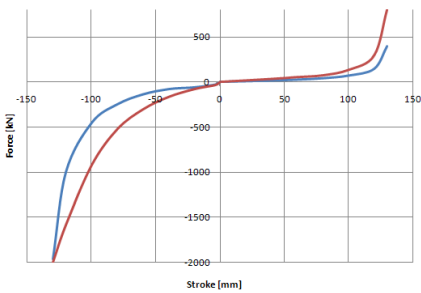


Fig. 43 Equivalent coupling characteristic with 0 mm of Gap

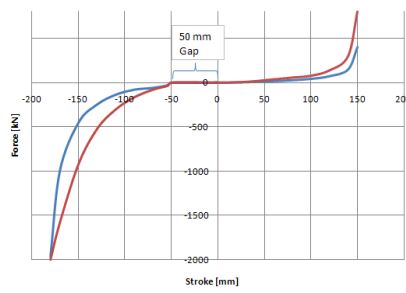


Fig. 44 Equivalent coupling characteristic with 50 mm of Gap

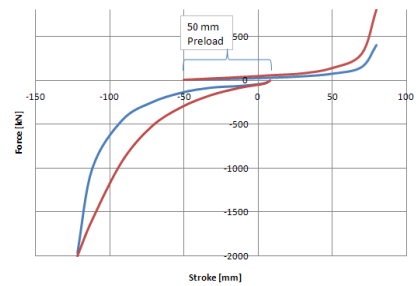


Fig. 45 Equivalent coupling characteristic with -50 mm of Gap

The negative stroke represents compression and the positive the traction. When a positive gap is set the equivalent coupling characteristic is modified by a section without force as showed in Fig. 44. Using a gap, the polynomial functions that define the force - stroke characteristics doesn't change but they are translated along the stroke axis.

If a negative gap (preload) is set, the equivalent characteristic is modified as showed in Fig. 45. In this case, there is a zone where there is compression and traction at the same time, the resulting force is the sum of these two forces.

If the relative approach of the draw gears is not known and the preload is known, the corresponding relative draw gear preload gap can be determined by the equivalent draw gear characteristic.

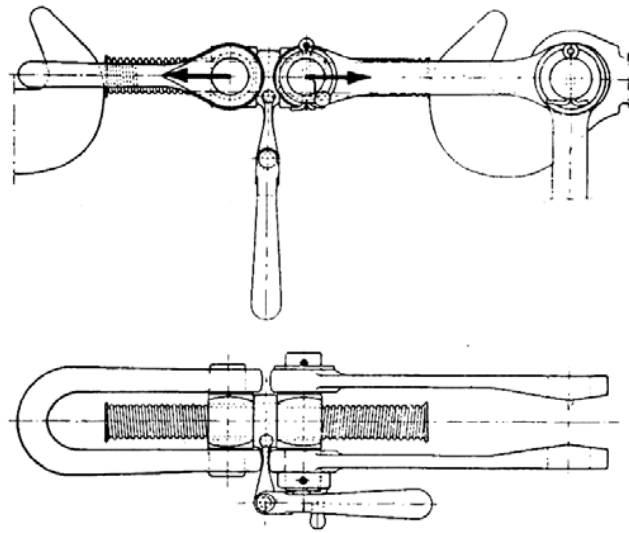


Fig. 46 Generic scheme of a coupling preload system