The Modeling of Energy Flows in Railway Networks using XML-Infrastructure Data

Andreas Heckmann* and Sebastian Streit◇
German Aerospace Center (DLR)
* Institute of System Dynamics and Control, Oberpfaffenhofen, D-82234 Wessling
◇ Institute of Vehicle Concepts, Pfaffenwaldring 38-40, D-70569 Stuttgart

Abstract

This paper introduces a new Modelica package called RailwaySystem Library that provides the capabilities of simulating the energy flow in electrical railway networks on which a fleet of railway vehicles is running. The focus of the library is set upon the interaction of the vehicle and its energy infrastructure, so that energy management aspects may be investigated from a holistic point of view taking the vehicle and the energy supply by the electric power grid into account. However this intention substantially relies on the provision of reliable data of the infrastructure, on the railway network and its power grid. To this purpose the library refers to an open XML-based data format dedicated to railway IT applications. Furthermore, the library is supposed to be used together with arbitrary component libraries to model the energy subsystems such as the vehicle or the power station.

1 Introduction

As public transport in general, railway transport as well has to cope with increasing demands on the reduction of energy consumption and CO_2 emission. This fact motivates activities of the DLR project Next Generation Train [1] regarding energy management in railway vehicles and recently led to the implementation of the Modelica RailwaySystem Library. This package provides the capabilities of simulating the energy flow in electrical railway networks on which railway vehicles are running.

From the modeling point of view two specific problems had to be taken into account. Railway vehicles may be interpreted as energy sources or sinks that are moving in an inhomogeneous network, see e.g. [2]. The network consists of catenaries or third rails that are supplied by power stations and may or may not be separated in isolated sections. Depending on the number and the instantaneous position and running state of the vehicles different types of flows may occur in parallel: energy may flow from power station to vehicle, or vice versa or from one vehicle to another vehicle.

As a second important aspect, the evaluation of the energy consumption of a vehicle is of course a function of the track characteristics such as length, slope, radius, positions of power station etc. so that data on the infrastructure topology and properties are required [3]. To this purpose, the library provides access to external infrastructure data, that are filed using the railway markup language railML®. This is a XML-based data format, advanced by the railML.org initiative [4] and licensed under creative commons conditions (CC By 2.0) [5].

The initial implementation in this paper is dedicated to consider energy consumption due to conduction losses, traction and auxiliary systems such as heating, ventilating and air-conditioning systems in DC urban-railway-networks. cp. e.g. [6]. However the simulation framework of the Railway System library does not introduce any restrictions on the modeling of the energy subsystems and is open for further extensions. In particular, the Railway System Library is supposed to be used together with component libraries such as the AlternativeVehicles [7] or the PowerTrain Library [8].

2 RailML® Data Interface

The non-profit railML.org initiative [4] is a consortium of railway companies, software and consulting firms, and academic institutions, that jointly define and advance a common data standard to be used in different railway simulation tools, see e.g. [9]. The addressed fields of applications are rather comprehensive and among others concern operation planning of rolling stock, resource planning of railroads, design of timeta-

bles, event and delay handling. As a consequence, the XML data standard railML®contains subschemas for three main areas: infrastructure, timetable, and rolling stock.

Compared to the scope of the railML.org initiative the piece of work to be presented here only covers specific aspects since it is focused on energy consumption. The initial implementation only considers data regarding track topology and geometry. The following section of an railML®file that specifies a track section of 2.7 km length is supposed to serve as an illustrative example:

```
<track id="t4">
   <trackTopology>
      <trackBegin pos="0" id="b4">
         <macroscopicNode ocpRef="PS3" />
      </trackBegin>
      <trackEnd pos="2700" id="e4">
          <macroscopicNode ocpRef="PS5"/>
      </trackEnd>
   </trackTopology>
   <trackElements>
     <radiusChanges>
        <radiusChange id="rC4"
                    pos="0" radius="900"/>
        <radiusChange id="rC5"
                    pos="400" radius="0"/>
     </radiusChanges>
     <gradientChanges>
        <gradientChange id="gC4"
                      pos="0" slope="5" />
        <gradientChange id="gC42"
                    pos="1000" slope="0"/>
      </gradientChanges>
    </trackElements>
</track>
```

The data set above specifies that the track starts as a curve with 900 m radius which changes to a straight track after 400 m. The gradient at the beginning of the track section is 5 per mill and changes after 1 km to be horizontally aligned. Note that every data element is specified by a XML-file-wide unique identifier *id*, which is required for later referencing that element.

Each Modelica model that wants to access railML®data has to contain an instance of the Modelica railML®class, see Fig. 1, and has to provide an XML file name as parameter. The railML®instance manages an external object that contains a Document Object Model (DOM) [10] tree of the XML data. The railML®instance may be addressed by the inner/outer mechanism so that other model components may easily acquire information from the railML®data.

During initialization of the Modelica model the XML-file is read using the XML parser library *expat* [11] published under MIT license [12] together with the wrapper *scew* [13] available under LGPL license [14]. Both open source tools are written in C and therefore may easily be compiled and bound together with translated Modelica code by every Modelica simulation environment such as OpenModelica, Dymola or SimulationX. As well from the legal point of view these two libraries may be distributed with a Modelica library as long as they are delivered as a self-contained code library which is not mixed up with other C-code.

During initialization a railML®object is instantiated and the DOM tree is built up by the two parser tools. In addition every track element found in the railML®data is assigned to an integer index number and a mapping of each index number to the XML-wide unique track *id* is organized.

3 Specific Modeling Issues

Fig. 1 presents a trivial network in order to give a first impression of the main modeling components of the RailwaySystem Library that are shortly introduced now.

3.1 Connectors

The library defines the following three connectors. The first one is an aggregation of the 3D-mechanical connector *frame* and the electrical connector *pin* and is tailored to connect catenary sections. The following

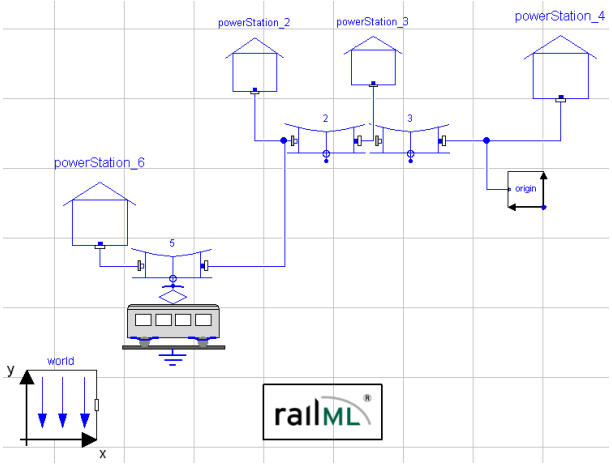

Figure 1: Diagram layer of a trivial network with 3 tracks, 4 power stations and 1 vehicle.

DOI 10.3384/ecp12076125

presentation will reveal that the capabilities of the 3D multibody framework are hardly exploited. Neither force or torques balances nor rotations are so far involved in the modeling approach of the library. However future applications may also consider longitudinal dynamics of train sets e.g. during braking or driving up-hill scenarios. In view of such use cases the 3D-mechanical connector *frame* are employed as described below:

```
connector frame_pin
  "supposed to connect catenary
     sections mechanically and
       electrically"
  import Modelica.Mechanics.
       MultiBody.Interfaces.Frame;
  import Modelica.Electrical.
       Analog.Interfaces.Pin;
  Frame frame;
  Pin pin;
end frame_pin;
```

Two other connectors are defined in order to provide the capability of attaching vehicles to the catenary. These connectors only differ in the prefix of the local position variable *s*, which is an output quantity on the vehicle side, while it is an input variable from the point of view of the catenary.

```
connector slidingContact_a
  "catenary side of catenary-
          pantograph connection"
  extends RailwaySystem.
         Interfaces.frame_pin;
  input Real s "local position"
end slidingContact_a ;
```

```
connector slidingContact_b
  "pantograph side of catenary-
          pantograph connection"
  extends RailwaySystem.
         Interfaces.frame_pin;
  output Real s "local position"
end slidingContact_b ;
```

In particular the definition and the purpose of the variable *s* is further motivated and explained in the following three sections.

3.2 Vehicle

The *Vehicle* model is a base class and supposed to be extended in order to characterize the energy system of a railway vehicle. The energy system itself may be arbitrary complex and may be modelled using components from the Standard Modelica library together with components from commercial libraries such as the AlternativeVehicles [7] or the PowerTrain Library [8].

Important parameters of the *Vehicle* model are *tracksToPass* and *tracksPassOver*:

- The parameter *tracksToPass*, e.g. *tracksToPass*= {5,2,3} in Fig. 1, is an integer vector containing the indices of the tracks the vehicle is supposed to run on. The order of the indices corresponds to the sequence of the tracks.
- The real vector *tracksPassOver*, e.g. *tracksPassOver*={0, 3200, 6800, 9800} in Fig. 1, specifies points on the path of the vehicle, at which one track is left and the following is entered.

Important transient variables of the vehicle model are *trackIndicator* and the real quantities *s* and *S*:

- The boolean vector *trackIndicator* is of the same length as tracksToPass. One and only one element of *trackIndicator* is true, namely the element that is associated to the track the vehicle is currently running on.
- The variable *s* defines a specific point on the track, the vehicle is currently running on. It is a local, track-specific variable on contrary to *S*.
- The variable *S* is a global vehicle-path-specific quantity. In the present implementation *S* is predefined as a function of time, so that motion of the vehicle along its path is preset e.g. as a result of the timetable. Alternatively it is also possible to give the velocity profile as a function of *S* and evaluate $S = S(t)$ accordingly.

The following table again summarizes the important variables explained above:

parameters	
tracksToPass	track indices
tracksPassOver	specific path points
transient variables	
s	local track position
S	global path position
trackIndicator	boolean track switch

From the purely structural point of view a vehicle instance is connected to all catenary sections that are listed in the parameter vector *tracksToPass* using the

sliding contact connector classes, see Sec. 3.1. However by employing ideal closing switches from the Standard Modelica.Electrical library it is guaranteed that only that electrical connection is closed to which the corresponding value of *trackIndicator* is set to true.

In order to access the railML®data and provide information on the e.g. the current gradient, the current values of *s* and *trackIndicator* together with *tracksToPass* are interpreted and passed to appropriate C-functions that extract data from the DOM tree.

3.3 Catenary

The *Catenary* model represents a track segment parametrized with the local length coordinate *s*. Its geometrical and electrical properties such as radius, gradient and electric resistance vary as a function of *s*. The integer parameter *ID* specifies the index to access the RailML database so that the necessary information on the track segment given by the RailML database may be acquired.

A sliding contact connector serves as an interface between vehicle and track and the current value of *s* denotes the current local position of the vehicle. Since the *Vehicle* instance as well as the *Catenary* instance both rely on the value *s*, the definition of the two connectors *slidingContact_a* and *slidingContact_b* in Sec. 3.1 considers the exchange of this variable.

The two other *frame_pin* connectors are supposed to connect different catenary sections. Future versions of the RailwaySystem Library will include the capability to automatically instantiate and connect all track sections found in the railML®database, so that the modeling of a complex network structure is substantially facilitated. So far the network structure is to be built up by manually instantiate and connect *Catenary* objects.

Note that variants of the *Catenary* model class are available that consider more than one vehicle running along the same track.

3.4 PowerStation and Origin

The *PowerStation* model is used to introduce transformer substations along the track that serve as voltage supply sources.

The railML®data only contains relative information like track lengths specifying the distances to travel form one point to another. In order to be able to set up an at least schematic animation of the traveling vehicles one absolute position has to be defined. This is done by the *Origin* model class.

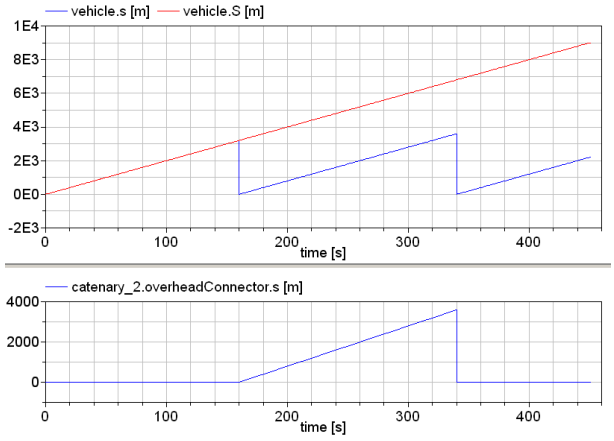

Figure 2: Plot of the variables *s* and *S* as a function of time.

3.5 Exemplary Simulation Sequence

In order to present the general simulation set-up of the RailwaySystem Library the simulation sequence of the trivial network shown in Fig. 1 will now be explained. The considered vehicle parameters are:

- *tracksToPass*= {5,2,3},
- *tracksPassOver*={0, 3200, 6800, 9800},
- $S = 20\ \text{m/s} \cdot t$

where *tracksToPass* and *S* are specified by user input, while the values of *tracksPassOver* are generated by a function that gains information from the railML®object during initialization.

According to the upper plot of Fig. 2, the vehicle leaves the first catenary section after 160 s, and the second after 340 s. This corresponds to the length of 3200 m and 3600 m of the first ($ID = 5$) and the second catenary ($ID = 2$) and the constant velocity of 20 m/s. The plot below shows the value of *s* seen from the *catenary_2* point of view. As long as the vehicle is not running on this catenary or track section, respectively, *s* is set to zero.

Fig. 3 demonstrates that the value of the first element of the vector *trackIndicater* is set to true as long as the vehicle is moving along the catenary specified by the first element of *trackToPass*. This applies for the second and the third element of *trackIndicater* accordingly.

The vector *trackIndicater* controls a vector of electrical switches so that the vehicle energy system is linked to that catenary or track section only, the vehicle is currently running on.

In summary, it is the general idea of the simulation set-up that the vehicle instance gathers all information.

DOI 10.3384/ecp12076125

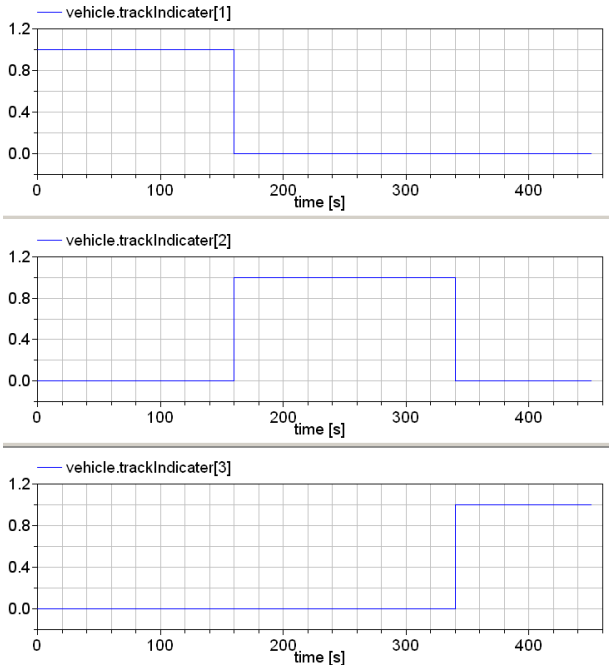

Figure 3: Plot of the boolean variable vehicle.trackIndicater.

The vehicle "knows" where, on which track or catenary section it is currently running and it is enabled to access the railML®data to acquire infrastructure information accordingly. The vehicle hooks itself up to the current catenary section in order to manage its own energy supply.

4 Application Example

The example model in Fig. 4 presents a small DC-powered urban light-rail network supplied by six power stations where two vehicles are running on six tracks.

Today power stations in DC-powered light-rail networks use rectifiers to provide a load-dependent control of the DC voltage. That means within the valid limits the output voltage is freely adjustable [2]. Therefore it is feasible to model the power stations as constant voltage sources. The nominal voltage used in this case is 750 V which is typical for DC-powered urban light-rail networks.

At a given load the traction current of the railway vehicle depends on the input voltage of the vehicle and thus of the specific position on the track [6]. For this reason the voltage drop alongside of the catenary has to be considered. Taking the resistance load per length into consideration, the voltage drop along the catenary can be calculated according to the length between the feeding point at the traction substation and the pantograph of the vehicle.

The basis for the calculation of the energy flow within the given network is the simulation of the vehicle trajectory. Based on the available nominal power of the vehicle and the tractive force at starting the tractive force-to-velocity characteristic is calculated. Thus the maximum available traction force can be calculated depending on the actual velocity. Depending on basic parameters of the vehicle e.g. weight, rolling resistance or aerodynamic resistance the driving resistance of the vehicle can also be calculated for each given velocity.

In addition specific parameters of the track resulting from curves, gradients or tunnel (provided via RailML data) lead to additional resisting forces that need to be considered. The movement of the vehicle is simulated by applying all resulting forces to a point mass. From the movement of this mass all necessary data for calculation of the electric network can be derived. The calculated mechanical power is used to derive the electrical power consumption of the vehicle. This power consumption is needed to calculate the traction current during simulation of the electrical network. The vehicle trajectory is performed for a predetermined velocity profile corresponding to the tracks to pass.

To simulate the energy flow within this network the vehicles are modeled as variable current sources using the actual required electric power consumption as input. In this way it is possible to calculate the resulting current sharing based on the electric power re-

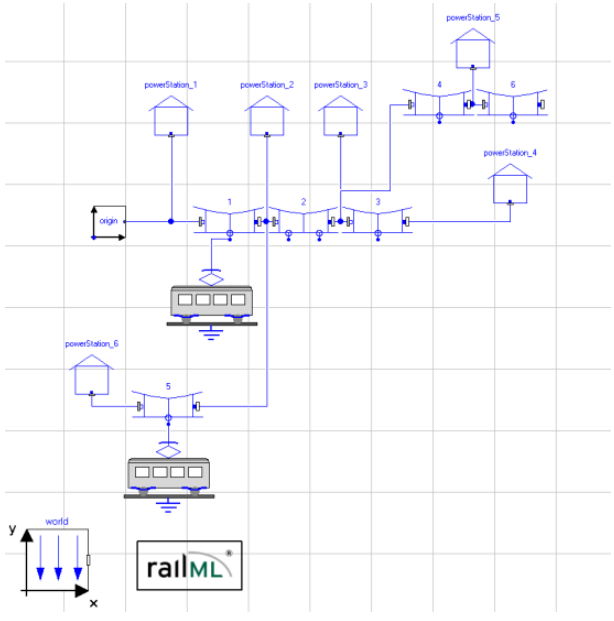

Figure 4: Diagram layer of the network.

quirement of the vehicles, the supply voltage and the voltage drop alongside of the catenaries.

As an example Fig. 5 shows the resulting voltage at the pantographs as well as the traction currents of two vehicles on their pass through section 2 of the exemplary network shown in Fig. 4. The first vehicle (red curves in Fig. 5) enters this section at about $t = 115$ s. It enters the section at a velocity of 40 km/h. Since the scheduled velocity in this section is 80 km/h the vehicle accelerates until it reaches the allowed velocity at about $t = 132$ s. This acceleration is associated to high traction forces resulting in a strongly increasing traction current.

Due to the increasing current the voltage at the pantograph drops during the acceleration to about 700 V. At approximately $t = 183$ s the traction current is again significantly increasing. At this specific point the gradient of the track changes from 0 to 30‰. This gradient abruptly increases the resisting force. To keep the scheduled velocity the traction force has to be increased as well resulting in a higher required mechanical and consequently electrical power consumption of the vehicle.

The traction current remains high until the gradient changes again from 30 to 0‰at approximately $t = 257$ s. Fig. 5 presents the voltage at the pantograph to jump up to 650 V during this passage.

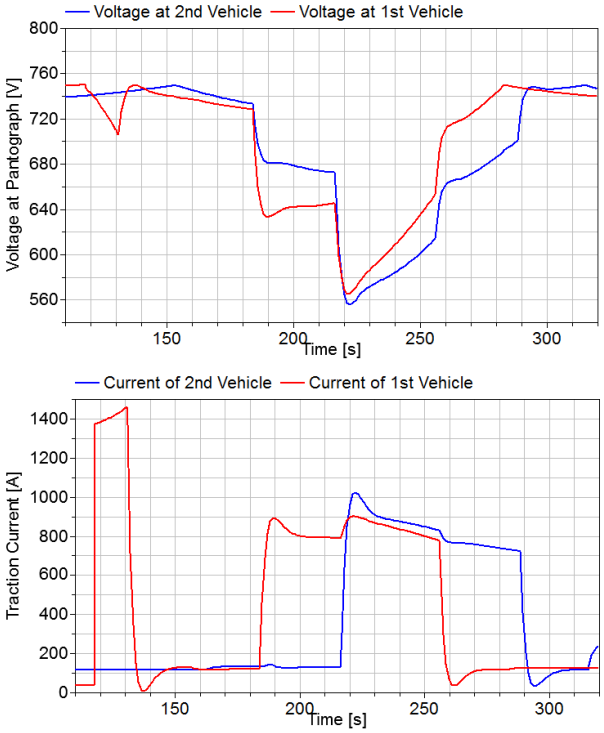

Figure 5: Simulation result of the voltages of both vehicles as a function of time.

The second vehicle (blue curves in Fig. 5) enters this section at about $t = 30$ s later already at a velocity of 80 km/h so that there is no further acceleration needed. When the second vehicle approaches the gradient change its traction current increases for the same reasons as mentioned before. At this point in time both vehicles have a high power consumption that leads to an additional voltage drop in the whole section. The voltage at the pantographs of both vehicles then drops significantly under 600 V which is critical since the minimum permitted voltage in DC-powered light rail networks with a nominal voltage of 750 V is 500 V.

To investigate the influence of energy storage devices as part of an energy management of railway networks a basic model of an electric double layer capacitor a so called Super Cap was also included within the vehicle model.

The Figures 6 and 7 each compare two different scenarios for the usage of these Super Caps. Fig. 6 shows the voltage drop at the pantograph of one vehicle passing the same section as shown in Fig. 5 as well as the state of charge of the Super Cap for two different cases.

Initially the vehicle is at rest and then accelerates up to 80 km/h. It stops at the end of the section. The Fig. 6 demonstrates the influence on the voltage drop during acceleration if state of charge is at 100% at starting time. In the sequence the voltage does not drop until

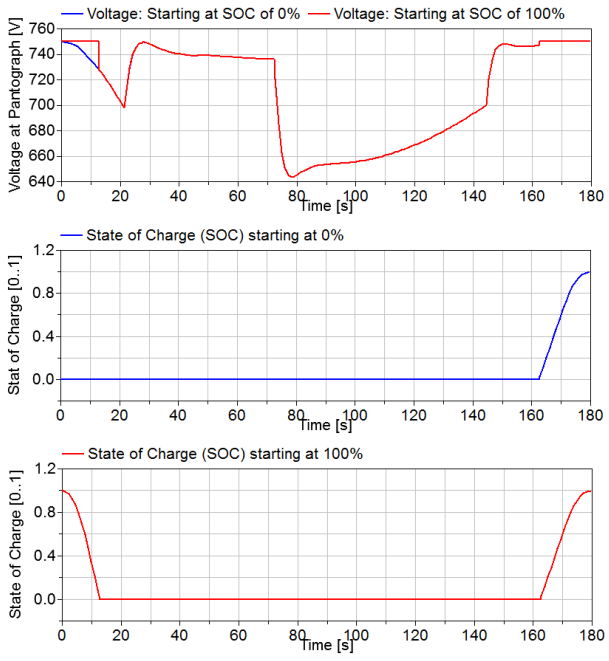

Figure 6: Simulation results presuming two different initial states of the Super Cap.

DOI 10.3384/ecp12076125

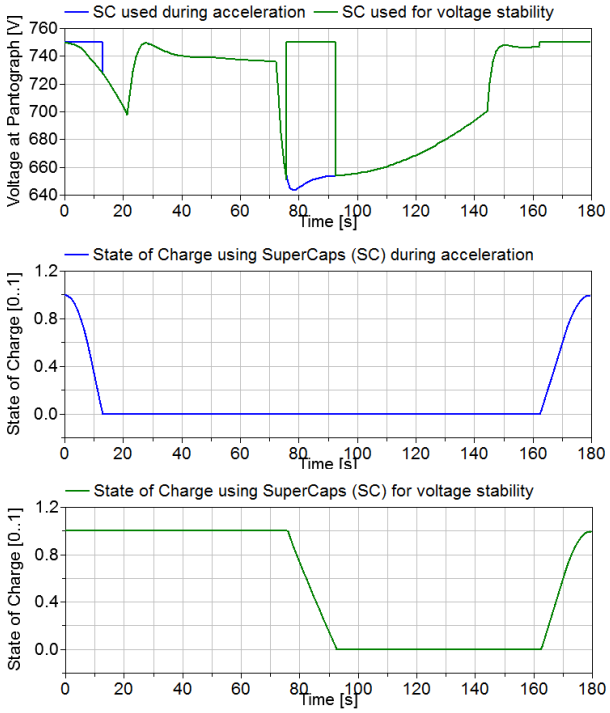

Figure 7: Simulation results associated to two different regimes for the use of Super Caps in operation.

the Super Cap is fully discharged at about $t = 17$ s. Until that point in time the full traction power is provided by the energy storage and no current is flowing between substation and vehicle.

When the vehicle starts braking at the end of the section, the Super Cap is fully charged and the stored energy can be used for the next run. Considering the same vehicle trajectory as in Fig. 6 Fig. 7 delineates another exemplary regime to employ an energy storage device. The Super Cap may not only be used to provide energy for acceleration but may also be exploited in order to ensure voltage stability. Fig. 7 presents the Super Cap not to be discharged until the voltage drops below 650 V. As soon as the voltage drops below this threshold the required electrical power is provided by the energy storage and for about 17 s the traction current is fully supplied through the Super Cap. This way

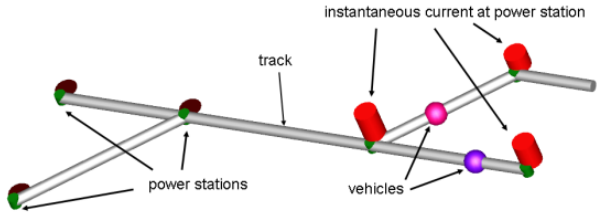

Figure 8: Schematic animation of the simulation.

a temporary voltage drop below critical values can be avoided.

Fig. 8 depicts an animation of the scenario when one vehicle is running along Track 3, while the other is moving between Power Station 3 and 5 at Track 4. The height of the red bars positioned close to each power station illustrate the instantaneous current provided by the associated power station.

The given examples have shown in principle that future investigations of energy flows within complex electric networks including the consideration of energy storage devices can be done using standardized data sets in Modelica.

5 Conclusions and Outlook

In the course of the DLR project Next Generation Train the RailwaySystem Library will be used in order to evaluate the energy reduction potential of an energy managément system that takes the vehicle energy system and the power supply infrastructure into account.

Future versions of the RailwaySystem Library will include the capability to automatically instantiate and connect all track sections found in the railML®database, so that the modeling of a complex network structure is substantially facilitated.

References

[1] J. Winter, E. Mittelbach, and J. Schykowski, editors. *RTR Special - Next Generation Train*. Eurailpress, DVV Media Group, 2011.

[2] H. Biesenack, G. George, G. Hofmann, A. Schmieder, E. Braun, K. Girbert, R.C. Klinge, R. Puschmann, S. Röhlig, E. Schlechter, E. Schneider, A. Stephan, and G. Zimmert. *Energieversorgung elektrischer Bahnen*. Teubner Verlag, Wiesbaden, 2006.

[3] D. Hürlimann. *Objektorientierte Modellierung von Infrastrukturelementen und Betriebsvorgängen im Eisenbahnwesen*. PhD thesis, ETH Zürich, 2001.

[4] RailML®. http://www.railml.org/web/. [Online; accessed 20-February-2012].

[5] Creative Commons License. http://en.wikipedia.org/wiki/Creative_Commons_license. [Online; accessed 15-May-2012].

[6] S. Röhlig. *Beschreibung und Berechnung der Bahnbelastung von Gleichstrom-Nahverkehrsbahnen*. PhD thesis, TU Dresden, 1992.

[7] Th. Braig, H. Dittus, J. Ungethüm, and T. Engelhardt. The Modelica library AlternativeVehicles for vehicle system simulation. In *21. Symposium Simulationstechnik, SIM 2011*, Winterthur, Suisse, Sept., 7. - 9. 2011.

[8] J. Tobolář, M. Otter, and T. Bünte. Modelling of Vehicle Powertrains with the Modelica PowerTrain Library. In *Systemanalyse in der Kfz-Antriebstechnik IV*, pages 204–216. expert Verlag, 2007.

[9] A. Nash and D. Hürlimann. Railroad simulation using OpenTrack. In J. Allan, C.A. Brebbia, R.J. Hill, G. Sciutto, and S. Sone, editors, *Computers in Railways IX*. WIT Press, 2004.

[10] Document Object Model (DOM). http://en.wikipedia.org/wiki/Document_object_model. [Online; accessed 15-May-2012].

[11] expat. The expat XML parser. expat.sourceforge.net/. [Online; accessed 16-March-2012].

[12] MIT License. http://en.wikipedia.org/wiki/MIT_License. [Online; accessed 15-May-2012].

[13] scew. Simple C expat wrapper. www.nongnu.org/scew/. [Online; accessed 16-March-2012].

[14] LGPL License. http://en.wikipedia.org/wiki/LGPL_license. [Online; accessed 15-May-2012].

DOI 10.3384/ecp12076125