Rail4Future

Project title:	**Resilient Digital Railway Systems to enhance performance**
Start date:	**01/04/2021**
Time duration:	**42 months**
Project number:	**882504**
Announcement:	**8. Ausschreibung COMET Projekte 2019**

Deliverable D1.3.5

Report – Implementation Strategies

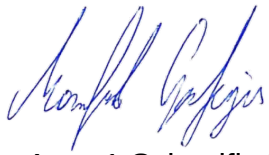
Due date	
Submission date	
Submitted by	TU Wien MIVP

Version	Date	Edited by	Description
0.1	13.05.2024	Ozan Kugu	Creation
0.2	06.06.2024	ViF	Additions (VTI Use Case) & Revision
0.3	12.06.2024	Ozan Kugu	Finalization
1.0	13.06.2024	Manfred Grafinger	Proofreading

Deliverable released

by Area 1 Manager
Dr. Michaela Haberler-Weber

Deliverable released



by Area 1 Scientific Lead
Dr. Manfred Grafinger

1 Executive Summary

In this deliverable, we show and describe strategies we followed to implement different railway models and data into the R4F Platform. During this implementation, we mention different open-source or commercial software tools, packages, libraries, file formats and interface standards, which we propose to use to standardize the simulation units of these models, and then run their simulation in the platform (see Deliverable D1.3.4). By this, we aim to reduce model and data complexity, to describe system structure, to increase tool-independence and file portability as mentioned in Deliverable D1.3.4. In general, this report provides information about how exactly we implemented the use cases mentioned above into the platform for the R4F project.

2 Table of Content

3 Abbreviations and Acronyms

Abbreviations / Acronyms	Description
R4F	Rail4Future
MBS	Multi body simulation
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
RLT	Residual Life Time
VTI	Vehicle Track Interaction
AIT	Austrian Institute of Technology
ViV	Virtual Vehicle Research GmbH
SSP	System Structure and Parametrization
GUI	Graphical User Interface
JSON	JavaScript Object Notation
JPG	Joint Photographic Experts Group
CSV	Comma-separated values
OS	Operating System
AI	Artificial Intelligence
ML	Machine Learning
PID	Proportional-Integral-Derivative
ÖBB	Österreichische Bundesbahnen (Austrian Federal Railways Holding Company)
AG	Aktiengesellschaft
AIT	Austrian Institute of Technology GmbH

4 Problem Description / Objectives

4.1 Problem Description

To be able to implement different use cases into a virtual platform, it is crucial to be familiar with simulation tools, simulation processes, file formats, programming languages, input parameterization, output generation, result validation, asset (model & data) integration, containerization and co-simulation techniques. Besides, some use cases take a relatively long time to finish their large-scale simulation in the platform. Thus, the simulation models of the use cases are to be reduced, trained by using an AI-based algorithm (for more details see Deliverable D1.3.3 and D1.3.4), and then to be adapted to the platform. In addition, the OS of the platform is a significant concern, because the simulation runs of models are OS-dependent. Furthermore, we need to connect the platform to a license server for some simulation tools, that allows us to run model simulations belonging to the use cases in the platform. Therefore, the necessity of commercial software licenses cannot be denied for the use case implementation. To overcome the limitations, we proposed a model standardization approach, where we standardize and containerize the simulation unit of different use case models incl. the AI-based simulation models with their data, and a model simulation approach, showing how to run simulations, how input parameterization, output generation and result validation of all the use cases are realized in the platform (see Deliverable D1.3.4).

4.2 Objectives

This deliverable aims to provide comprehensive insights about the railway use case implementation strategies to adapt these to the R4F Platform. Exact methods, software tools, packages and libraries used for this implementation task are defined and described depending in the use case. In this report, we mention five use cases:

- MBS model of a railway vehicle,
- ML-based surrogate model for the MBS model of the vehicle,
- Anti-slip traction and vehicle speed control co-simulation model of the vehicle,
- RLT calculation of a railway steel bridge,
- VTI (Vehicle Track Interaction) use case.

We should point out that we always tried to stay in the open-source level as much as possible for cost-saving. Surely, we needed to tend to the commercial side in some cases, which are mentioned in Description Section, while implementing the use cases into the platform as well.

5 Significance for the overall Project

This deliverable plays a crucial role in concretizing and virtualizing the previously defined use cases in the R4F Platform. This gives technical and valuable insights about the use case implementation in a railway digital twin platform characterizing the R4F project. Consequently, these are ready to be automatically integrated, then visualized and controlled in the platform by the users (incl. railway operators and infrastructure managers), who are official customers of the project.

6 Description

6.1 Use Case Implementation

6.1.1 MBS Model of a Railway Vehicle

This use case demonstrates the train ride of a Manchester Benchmark based virtual railway vehicle with two bogies (see [4]). The MBS vehicle model was provided from ViV and created with the commercial Simpack software tool from Dassault Systèmes. In Simpack, the model is interacting with a designed track to realize and perform simulation of the rail drive on track. As first step of the implementation of the model, input and output channels have to be defined and assigned to the model, because these are then to be used for input parameterization and output generation in the R4F Platform. In the meanwhile, the model needs to be configured so that right input and output parameters implemented in the model are addressed and then invoked. Besides, the Simpack tool with or without GUI is able to package the model into FMU, which the FMI standard provides as simulation unit and container [2]. For the R4F project, the MBS model is packed into FMU with the version of FMI 2.0 for Co-Simulation enabling to give inputs and outputs in boolean, float, integer, string and enumeration formats (see [5]). As a result, the MBS model is presented as an FMU file, which can then be brought into simulation as a black box with input and output connectors, coming from the predefined channels, in the platform.

6.1.2 ML-based Surrogate Model

The initial development of our ML-based surrogate model, described in Deliverable D1.3.3 and D1.1.5, utilized MATLAB, the commercial software tool from MathWorks, primarily because the measurement data we acquired were also in MATLAB format. However, recognizing the importance of programming language uniformity for seamless integration into our platform and enhanced accessibility for industry partners, we opted to transition the code to Python, an open-source language widely utilized in various domains. This decision aimed to facilitate easier deployment and accessibility for industry partners to incorporate the model into their systems. Through meticulous translation of the code and corresponding adjustments, the Python version of the model proved to be effective, demonstrating comparable performance to its MATLAB counterpart. Subsequently, the model was encapsulated into FMUs, which can then be successfully deployed within the pipeline for integration into the R4F Platform. This transition not only ensured compatibility and accessibility but also streamlined the integration process, aligning with our overarching objectives of fostering interoperability and facilitating widespread adoption within the industry.

After extracting the surrogate model as a Python code, the model is to be manually processed so that it can be packaged into the FMU format. First, we register necessary input and output channels as previously done in the MBS model use case. This enables us to give correspondent inputs and then to get relevant outputs in the platform. Besides, the functions implemented in the code are moved into a newly created Python class, from where the Python functions are invoked

for the ML-based MBS of the railway vehicle. In addition, another Python class is created, in which the entire simulation is performed by invoking the functions from the firstly created class, parameterizing inputs and generating outputs. Fig. 1 shows the structural overview, which is based on the class and function replacement, of the first and final versions of the ML-based surrogate model Python code.

Fig. 1: Python code versions of the ML-based Surrogate Model

After finishing the manual processing of the Python code, we packed it into the FMU by using the PythonFMU Python package [3]. As a result, we get an FMU file with the version of FMI 2.0 for Co-Simulation, by which we can give the name of the folder, containing all the CSV input and output files, as a string. This string is put into a JSON file, likely used for all the use cases of the R4F Project. As outputs, three MAT files and one diagram picture (JPG) with validation curves are generated as mentioned before.

6.1.3 Anti-Slip Traction and Vehicle Speed Control Co-Simulation

In this use case, we aim to control the traction of the MBS vehicle model by using a PID-based controller system. We previously discovered SIMAT in the Simpack documentation, which provides an interface between Simpack and MATLAB. For our case, we already have the MBS model as a Simpack file, can model a simple PID controller in MATLAB/Simulink and then connect these Simpack and Simulink models to each other by using SIMAT.

The PID-based Simulink model consists of two parts. In the first part, the actual slip of the MBS vehicle is approached to the desired slip, given by the user, to enhance the anti-slip traction control of the vehicle. The second part purposes to directly control the vehicle speed by co-accelerating and co-braking the vehicle. Besides, we implement input blocks into the Simulink model, which help us then to realize the input parameterization of the co-simulation model in the R4F Platform.

In addition, the Simpack MBS model has to be manually processed in Simpack, where we create four force elements (one per each wheelset) invoking the wheelsets of the vehicle, define use case specific input and output channels and then assign the input channels to the force elements.

After creating the PID model, we import a SIMAT component, which is directly connected to the MBS model, and then connect its input and output ports to those of the PID controller model in Simulink (see Fig. 2a). In this case, Simpack works as a server and MATLAB as a client. To perform a SIMAT co-simulation, first we initialize the co-simulation mode of the MBS model in Simpack. After that, we start the co-simulation in the Simulink model. As a result, we get outputs both as directly resulted from the MBS model in Simpack as well as directly resulted from the PID model in Simulink.

To completely adapt the co-simulation model to the R4F Platform, we prefer to use the FMI and SSP standards, because these help us to simulate the model tool-independently as one container in the platform. First, we package the Simpack MBS and Simulink PID models into two FMU files, which we then connect to each other in the commercial co-simulation tool Model.CONNECT provided from AVL List GmbH. After successfully testing the co-simulation in Model.CONNECT and comparing its results to the SIMAT's results, we package the entire model into SSP by using the Model.CONNECT tool. We have then a more descriptive and containerized version of the model. Besides, the SSP can be given as a black-box with inputs and outputs for its simulation in the platform as previously discovered. (see Fig. 2b)

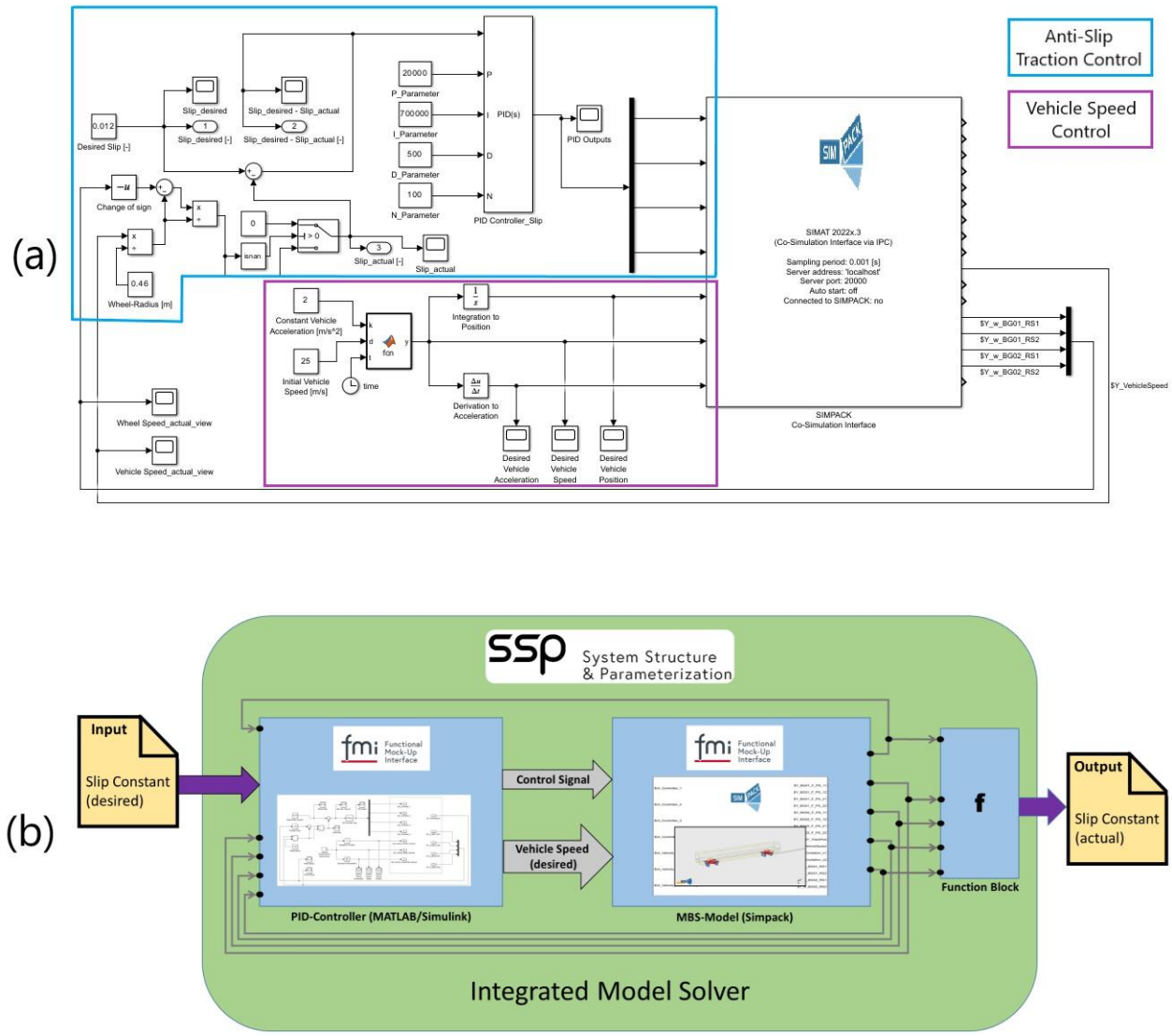


Fig. 2: Structure of the Implemented Anti-Slip Co-Simulation Model; (a) SIMAT; (b) FMI & SSP [6]

6.1.4 RLT Calculation of a Railway Steel Bridge

Like in the ML-based surrogate model subsection, there is a Python code for the RLT calculation of a railway steel bridge, provided from AIT, to be further processed for its integration in the platform. In the code, first, necessary calculation functions are separately implemented and then executed for the RLT calculation with the rainflow algorithm of a Python package. For this bridge use case, we adapted a dummy version of the bridge. Later, we successfully integrated three real versions of the use case such as Eschenau bridge, Mürzbrücke from ÖBB Infrastruktur AG, and Schellhammergasse bridge from Wiener Linien GmbH & Co. KG in the platform. For the manual adaptation of the Python code, we followed the same procedure as in the ML-based surrogate model use case. After that, we tested the RLT simulation with CSV inputs and outputs by executing a Python simulation code script (see [1]). Because of the relatively high data complexity of the inputs (especially the influence lines), we preferred to give folder name as a simple string in a

JSON file. The folder contains all the CSV input files incl. influence lines, load data, detail positions, detail categories and train data.

After the manual processing of the Python code, we package it into FMU like previously done for the ML-based surrogate model use case. As outputs, we get one CSV file incl. all the life time [year] and damage sum [1/year] results. Besides, one JSON file based on the R4F standards is generated after executing the FMU.

6.1.5 VTI Use Case

Ballasted tracks represent the most common used type of railway tracks. Vertical track stiffness and its variation as well as the variation of vertical track geometry along the track are one of the main sources for the dynamic wheel-rail contact forces. These dynamic forces are responsible for

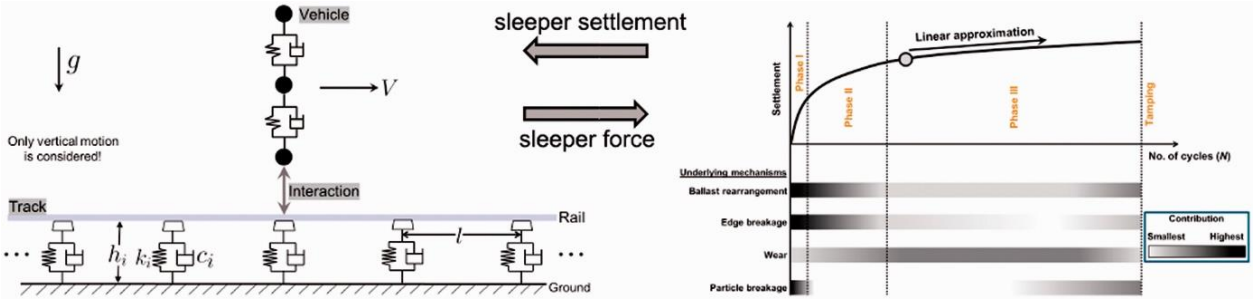


Fig. 3: The modelling approach. On the left side the Vehicle-Track-Interaction (VTI) model and on the right side the sleeper settlement model. [7]

the development of the track geometry. Thus, models predicting the evolution of vertical track irregularities are of high interest e.g. investigations regarding vehicle dynamics, to assess the track-friendliness of vehicles or for better designing and maintaining railway tracks. Therefore, the vehicle-track interaction (VTI) model was developed to study the long-term behavior of railway tracks. [7]

Fig. 3 shows the modelling approach. The vehicle and track system are shown in the left part. The track model considers the discrete support of the elastic rail, where each sleeper support can have its own stiffness, relative height and settlement behavior. The vehicle model represents an 8th of a car and considers only vertical dynamics. The initialization of the track model is done using the measured vertical static track deflection and track geometry. If these measurements are not available, assumptions need to be made. On this initialized track, a 1/8 vehicle runs with constant speed. After the vehicle pass, a load history is obtained for each sleeper due to dynamic interaction between vehicle and track. The peak sleeper force is used as an input in the right part of Fig. 3 to obtain the incremental sleeper settlement dependent on the number of vehicle passes (load cycles). This loop is repeated to obtain the track geometry evolution for the considered track and is visualized in a flowchart in Fig. 4. [7]

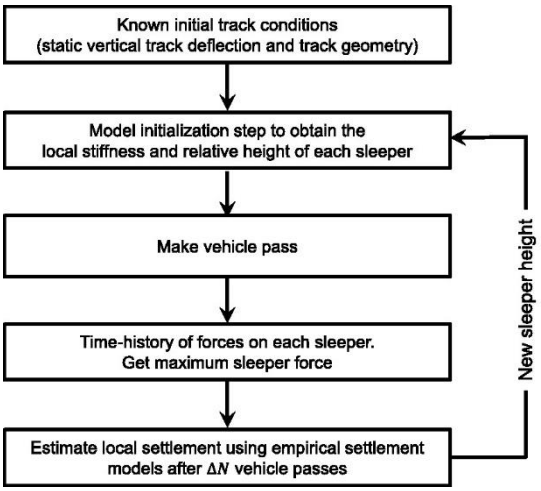


Fig. 4: Flowchart of the VTI model simulation methodology for calculating the evolution of track geometry. [7]

7 Conclusion

This report presented methods showing and describing how to implement all the above mentioned use cases into the R4F Platform. As methods, open-source or commercial software tools, packages, libraries, file formats and interface standards were named, which we preferred to use for the use case implementation incl. the standardization of the simulation units, model reduction and simplification mentioned in Deliverable D1.3.4.

In Deliverable D1.3.6 we will present deployment results of these use cases comprehensively as validated.

8 References

[1] Kugu, O., Zhou, S., Nowak, R., Müller, G., Reiterer, S. H., Meierhofer, A., ... & Grafinger, M. (2023, December). An FMI-and SSP-based Model Integration Methodology for a Digital Twin Platform of a Holistic Railway Infrastructure System. In Modelica Conferences (pp. 717-726). https://doi.org/10.3384/ecp204717

[2] Functional Mock-up Interface (FMI): https://fmi-standard.org/

[3] PythonFMU: https://pypi.org/project/pythonfmu/

[4] Iwnick, S. (1998). Manchester benchmarks for rail vehicle simulation. Vehicle System Dynamics, 30(3-4), 295-313. https://doi.org/10.1080/00423119808969454

[5] Documentation of FMI 2.0: Specification 2.0.4. pdf-Link: https://github.com/modelica/fmi-standard/releases/download/v2.0.4/FMI-Specification-2.0.4.pdf

[6] Zhou, S., Kugu, O., Reiterer, S., H., Wurth, L. & Grafinger, M. (2024), A Reinforcement-Learning-based Parameter Tuning Methodology for Traction Control in the Holistic Railway Digital Twin System. CIRP Design 2024. (paper accepted)

[7] Kumar, N., Kossmann, C., Scheriau, S. & Six, K. An efficient physical-based method for predicting the long-term evolution of vertical railway track geometries. Proc Institution Mech Eng Part F J Rail Rapid Transit 236, 447–465 (2021).