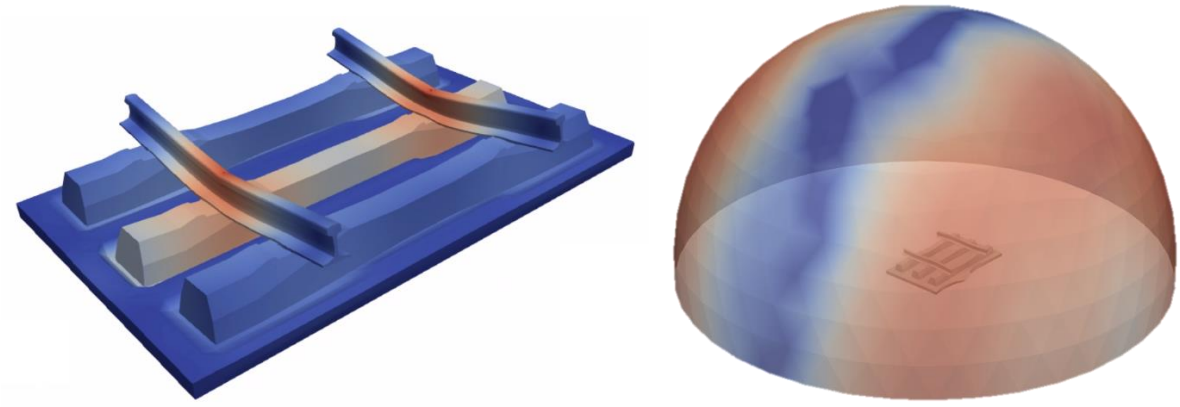
Rail Track Modelling Toolbox

Documentation

M. Ammann, J. Cugnoni, B. Morin, R. Nardin, B. Van Damme,



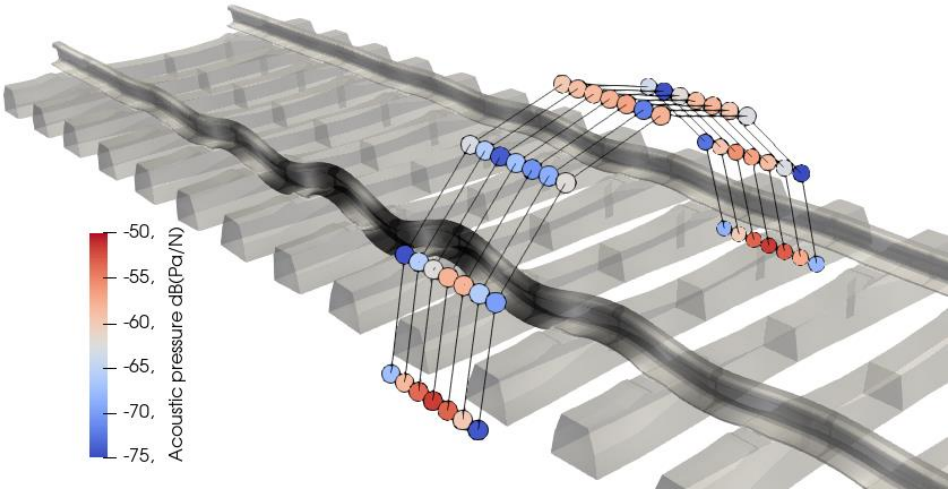


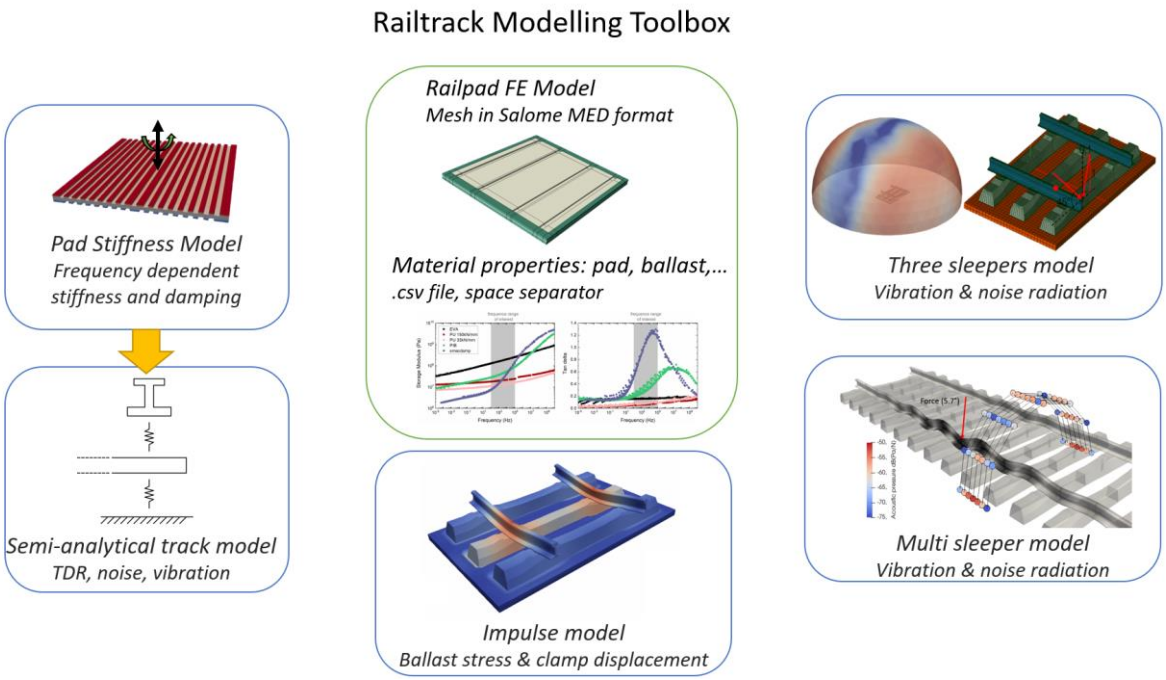
Table of contents

1. Introduction

The Rail Track Modelling Toolbox is an open source tool, available on GitHub (https://github.com/jcugnoni-heig/RailTrackModellingToolbox), which allows performing simulations of rail track components. The Toolbox includes five simulation models:

1. The **Three-Sleeper Model** performs vibro-acoustic simulations.
2. The **Multi-Sleeper Model** is an extension of the Three-Sleeper Model to longer tracks and performs vibro-acoustic simulations as well. It was developed later, and somewhat replaced it because it is faster and as much accurate. The user is therefore advised to use the Multi-Sleeper Model over the Three-Sleeper one.
3. The **Impulse Model** performs time-domain pass-by simulations on a three-sleeper-long rail track mesh.
4. The **Pad Stiffness Model** computes the frequency-dependent stiffness and damping ratio of rail pads. Its results are used as inputs of the Rail Track Semi-Analytical Model.
5. The **Rail Track Semi-Analytical Model** is the only model that is not finite element based. It uses the Pad Stiffness Model results to define spring-damper discrete elements connected to analytically-modelled rail beams.

They were all used in the context of the project "Novel Rail Pads for Improved Noise Reduction and Reduced Track Maintenance", supported by FOEN (Swiss Federal Office for Environment) and in partnership with SBB (Swiss Federal Railways) and Semperit AG. All models but the last one are finite element based and use the open source solver Code_Aster.



This document is a user and developer guide for the Rail Track Modelling Toolbox. It mainly focuses on the use of the Graphical User Interfaces (GUI) use and some specificities about the models. If the user needs more information about the Code_Aster functions that were used, and have a better

understanding of the solver itself, he or she should refer the official documentation on https://www.code-aster.org.

2. General information

2.1. About Code_Aster

Here are presented some aspects of the Code_Aster solver, which is used in this Modelling Toolbox. Firstly, Code_Aster is "unit-free". Thus, the user is responsible to use a consistent unit system. The unit system used in all the following models is as follows.

- N - Newton
- mm - millimeter
- t - ton
- s - second
- K – Kelvin

Secondly, a simulation in the Code_Aster sense is called a *job* here. Some simulations, in the Toolbox sense, are computed using several jobs in parallel. Each job run by Code_Aster is entirely defined by a single file having with the *.export* extension, by convention. It contains all information needed, such as CPU and memory allocation, running directory, input/output files.

In particular, export files contain the path of the command file (**.comm*): a script where all Code_Aster commands are coded. From materials assignment to matrices assembly and solving, again, all commands can be found in the Code_Aster documentation (www.code-aster.org). Note that when programming command files, all Aster concepts (objects) names as well as mesh groups can contain up to 8 characters only.

The role of the GUIs provided in the Modelling Toolbox is to manage files correctly based on inputs provided by the user. Then, they run the jobs properly, gather and post-process the results of all jobs in order to provide usable results. When a problem occurs, it is sometimes due to memory or paths issues, for instance, and this information is set in export files. The user and/or developer can refer to Code_Aster documentation [D1.02.05] to better understand export files.

Finally, the most convenient pre- and post-processing softwares to use alongside Code_Aster are Salome and ParaView, respectively. The software Salome-Meca, freely available on https://www.salome-platform.org/, includes Salome's pre-processing tools, Code_Aster's solver, and the post-processing and visualization software ParaView (called ParaVis in Salome-Meca). Salome-Meca is included in the Toolbox.

2.2. Materials modelling

The Table 1 shows the summary of the hard-coded properties. The remaining materials properties are chosen by the user through the GUIs.

Table 1: Summery of the hard coded properties.

Parts	Young Modulus [MPa]	Poisson's ratio [-]	Density [t/mm^3]
Rail - steel	210000	0.3	7.85e-09
Sleeper - concrete	46300	0.2	2.44e-09
Equivalent Ballast – wood beams	46.33	0.3	0.52e-09

2.2.1. Frequency-domain materials properties

To implement visco-elastic properties in the frequency domain, one should know the complex Young modulus of the material in function of the frequency. In those models, the hypothesis of a non-frequency dependent Poisson's ratio has been done.

Therefore, a frequency dependent material need:

1. The Storage modulus E' versus frequency
2. The tan(delta) versus frequency
3. The Poisson's ratio
4. The density

The complex Young Modulus E^* is defined by:

$$E^* = E' + i \cdot E'' \tag{1}$$

E' is the storage modulus and E'' the loss modulus.

Tan(delta) is define by:

$$\tan(delta) = \frac{E''}{E'} \tag{2}$$

To create a new frequency based material, please refer to the dedicated model instructions.

The properties are frequency dependent therefore each line contain the frequency and the associated property (E' or $\tan(delta)$) separated by a "tab" (\t).

For the PadStiffness model, after the calculation of all the frequencies required, a static analysis is performed. Therefore, one should add the quasi-static young modulus at the beginning of the E.csv file associated with 0 Hz frequency.

2.2.2. Time-domain materials properties

The Maxwell generalized model is used to implement the time based visco-elastic properties in the impulse model.

Therefore, a material needs:

1. The Young modulus – E
2. The shear modulus – G_i
3. The bulk modulus – K_i
4. The specific times – tau_i
5. The Poisson's ratio
6. The density

A four terms Maxwell model is used, therefore K_i is composed of K_0, K_1, K_2, K_3 and K_4, G_i is composed of G_0, G_1, G_2, G_3 and G_4 and tau_i is composed of tau_1, tau_2, tau_3, tau_4.

K_0 and G_0 correspond to the bulk and shear modulus at infinity (quasi-static) respectively. K_i and G_i are the bulk and shear modulus corresponding the "tau_i" specific time.

For more information about the material law: http://tfel.sourceforge.net/gallery.html#viscoelasticity

To create a new time based material, please refer to the dedicated model instructions.

2.3. Meshes

All meshes used are in *.MED* format (*Modélisation et Échanges de Données*). Meshes visualization or generation can be done in the Mesh module of Salome-Meca (file -> import -> med file). Mesh groups can then be displayed or modified using the tree-view.

2.3.1. Pad meshes

The mesh can be generated using Salome (see previous section) and the expected format is *.med*. Pads are used in all models. For most of them, a quadratic mesh is recommended, but if time and/or computational resources are an issue, linear meshes can work too (for instance for the three-sleeper model).

The orientation of the pad is shown on the Figure 1. The origin is centered on the bottom face of the rail's pad. The X axis is parallel to the rail, the Y axis is vertical pointing upward and the Z axis is parallel to the sleeper.

The dimension of the pads vary in function of the selected type of rail and sleepers. For the all the models except the Impulse model the following shape is required, corresponding to a 60E2 rail and a B91/60E2 sleeper:

The dimension of the rail's pad is the 162 x 150 x 7 mm:

- 162 mm in X-axis, parallel to the rail
- 7 mm in Y-axis, vertical
- 150 mm in Z-axis, parallel to the sleeper

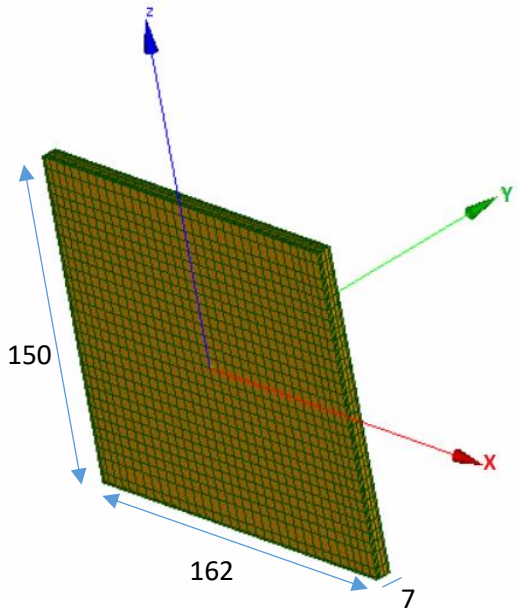


Figure 1: Dimension (mm) and orientation of the rail's pad mesh.

A pad mesh must contain some mesh groups. Although it is recommended to create all of them in order to make the mesh usable for every model, one can see which groups are necessary for each FE model in Figure 2. Here follows a quick description of each mesh group:

- *hard*: volume of the pad which will be attributed the material called "hard" (or "Material 1").

- *soft*: idem with material “soft” or “Material 2”. Note that none of these groups can be empty, their union must compose the whole pad mesh, and their intersection should be null. Simply assign the same material to both groups to create a single material pad.
- *petr*: the surface elements, which constitute the top face of the pad (Y+).
- *pets*: idem with bottom face (Y-).
- *pntr*: the nodes lying on the top face of the pad (Y+).
- *pnts*: the nodes lying on the bottom face of the pad (Y-).

		Model			
Group	**Type**	**Pad Stiffness**	**Impulse**	**Three-sleeper**	**Multi-sleeper**
hard	Volume	X	X	X	X
soft	Volume	X	X	X	X
top	Surface		X		
bot	Surface		X		
topn	Node		X	X	X
botn	Node		X	X	X

Figure 2: Mesh groups of the pads

A tool was developed to convert pad meshes from Abaqus *.inp* format to *.med* format. It is located in *src/ThreeSleeperModel/ConvertAbaqusPadMesh*. Firstly, consider using the script “changePath.sh” to automatically update the hard-coded paths in the export file “meshConvert.export” (or *setup.sh* in the main directory). Then, paste a copy of your *.inp* file in the same folder and rename it “Export_Aster.inp”. Finally, run the script “ConvertMesh.sh”. The converted mesh file is called “importedPad.mesh.med”, and it is highly recommended to import it in Salome and check if everything is right (groups, linear/quadratic, etc.).

2.3.2. Three-sleeper mesh description

The unit-cell mesh is used in the Three-sleeper Model. It is composed of two rails, three sleepers and a ballast layer. The 6 pads are added in a separate step after being selected by the user via the GUI. This mesh reproduces a real experimental set-up composed of B91 sleepers, 1.80m-long 60E1 rail segments, Vossloh W14 clamping systems and a ballast substitute of 10cm-thick wooden beams.

The previous version of the Impulse model was also using the unit-cell mesh. The new version is modular and allow the user to select which rail, sleeper, ballast and pads mesh to use. Then the assembly is made. Therefore the different meshes are presented in the following section with their respective groups and orientation.

An overview of the mesh is shown in Figure 3. All finite elements are 3D solids with linear interpolation functions for computational performances purposes. The ballast, sleepers and rails contain about 63000 nodes. The total number of nodes obviously depends on the pad mesh used. In general, the mesh of one rail pad contains about 10000 nodes. As for the USPs mesh, each USP contains 8000 nodes and 3 through-thickness elements.

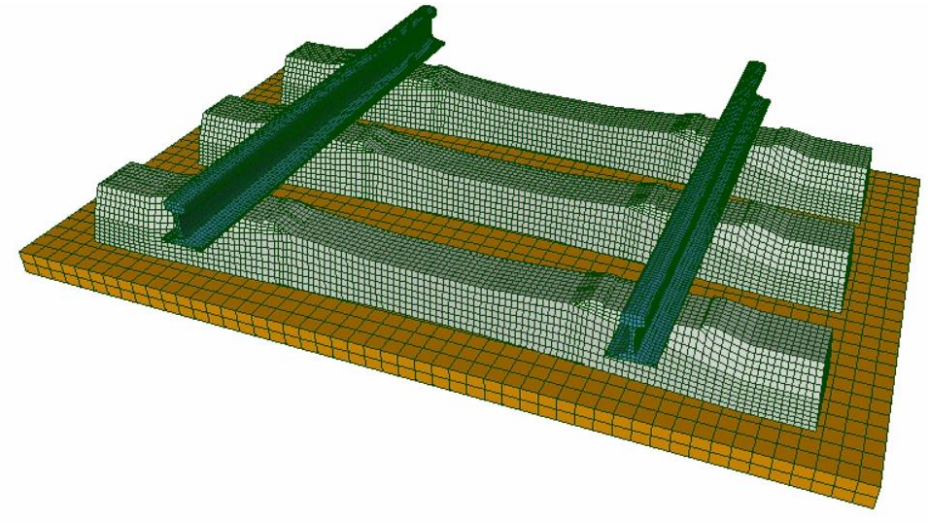


Figure 3: Unit-cell mesh overview.

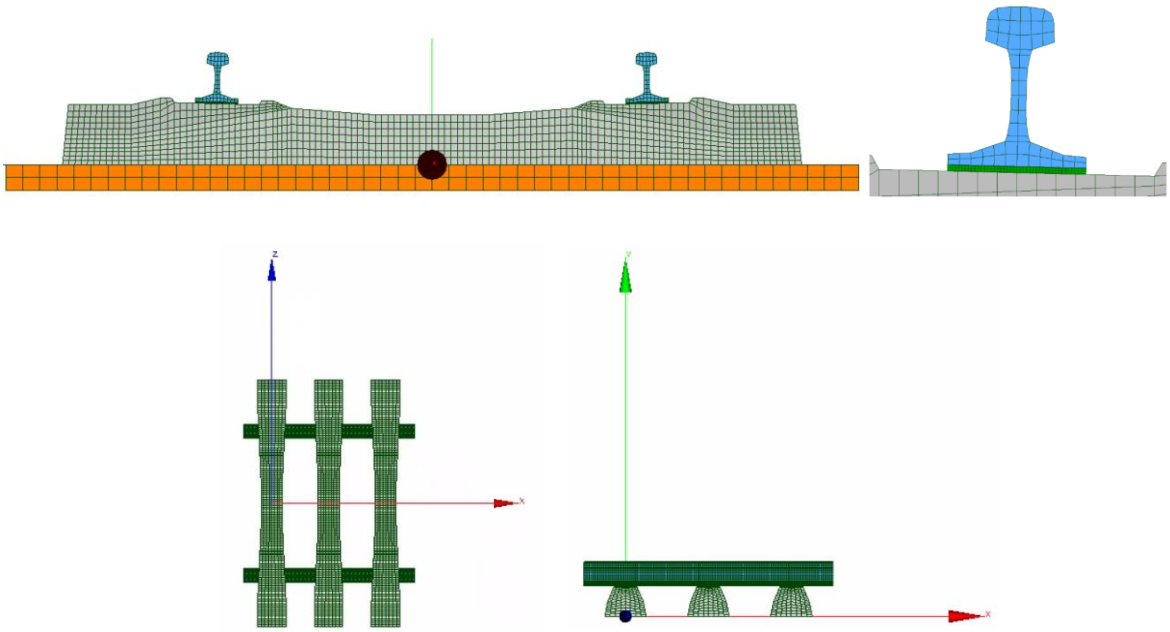


Figure 4: Orientation of the three sleepers mesh. X-axis parallel to the rails, Y-axis vertical and Z-axis parallel to the sleepers. The origin is on the middle of the bottom face of the first sleeper.

2.3.3. Rail meshes

The rail meshes, as all other meshes, have to be .med formatted. The origin of the reference system is in the middle of the bottom face of the rail. The X-axis is parallel to the rail, the Z-axis is perpendicular to the rail and the Y-axis is vertical pointing upward, see Figure 5.

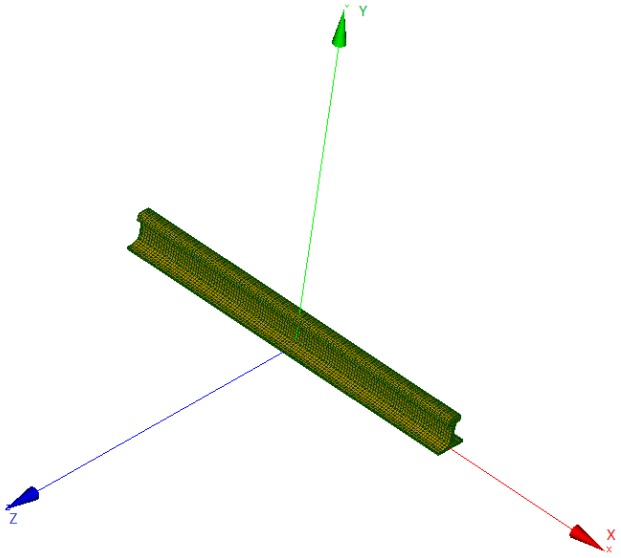


Figure 5: Rail mesh and reference system overview.

The rail mesh have to be 1.8 meter long, the shape vary in function of the type.

In order to make the unit-cell assembly correctly a number of nodes and elements groups are required:

- All the rail elements are placed in a group called: “raile” (for **RAIL E**lements).
- The nodes and elements that will be in contact with the pads. They are name rntp1, rntp2, rntp3, retp1, retp2 and retp3 (**R**ail **N**odes **T**o **P**ad **1** or **R**ail **E**lements **T**o **P**ad **1**). The Z-direction is used for numeration order (i.e. Z- : 1, O : 2 and Z+ : 3). See Figure 6.

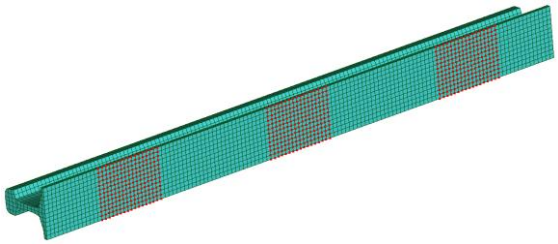


Figure 6: Rail nodes groups of the bottom surface.

- The user need to define the location where the force will be applied: “nodeF”. This group is located on the top of the rail in the middle, Figure 7.
- The clamps are not explicitly modelled and are replaced by springs and dashpot. Therefore the nodes where those springs are attached on the rail need to be specified. Each node, i.e. anchor, is on its own group. They are called: “rntsr1a”, “rntsr1b”, “rntsl1a”, “rntsl1b”, “rntsr2a”, “rntsr2b”, “rntsl2a”, “rntsl2b”, “rntsr3a”, “rntsr3b”, “rntsl3a” and “rntsl3b”. As one can see there are two springs per clamp. The numeration is the same as for the bottom side groups. The right side of the rail is in X+ and left in X-. The acronym stand for: **R**ail **N**ode **T**o **S**leeper **R**ight **1 a**. The final “a” or “b” follow the same logic as the 1, 2, 3, i.e. X-direction.
  In addition those groups, there are six similar groups: “rntsr1”, “rntsl1”, “rntsr2”, “rntsl2”, “rntsr3” and “rntsl3”. Only one of these groups is currently used in the model, but more can be used if necessary. The “rntsl2” group is used and allow, with a similar group on the sleeper side, to compute the pad compression.

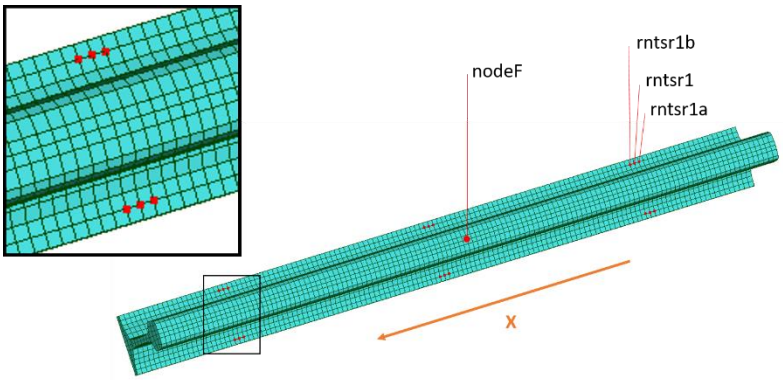


Figure 7: Rail nodes groups of the top surface.

2.3.4. Sleeper meshes

The sleeper mesh orientation is as follow: The origin is on the bottom face in the middle, the Z-axis is parallel to the sleeper, the X-axis perpendicular to the sleeper and the Y-axis vertical pointing upward, Figure 8.

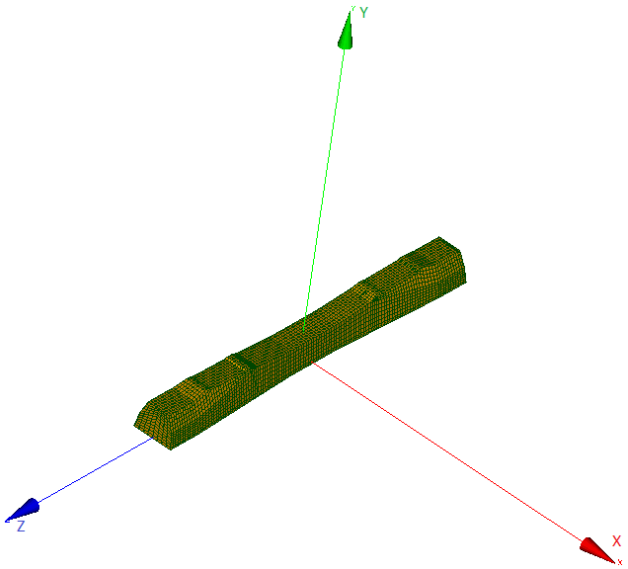


Figure 8: Sleeper mesh and orientation overview.

The following groups are required:

- All the elements of the sleeper have to be placed in a group called: “sleepere” (for **SLEEPER E**lements).
- All the nodes and elements of the bottom surface have to be in the groups called: “sntb” and “setb” (for **S**leeper **N**odes **T**o **B**allast and **S**leeper **E**lements **T**o **B**allast).
- The bottom face has a second node group that contain only one node: “botSlp”. This node is used for data extraction at the end of the simulation. See Figure 9 and Figure 15.

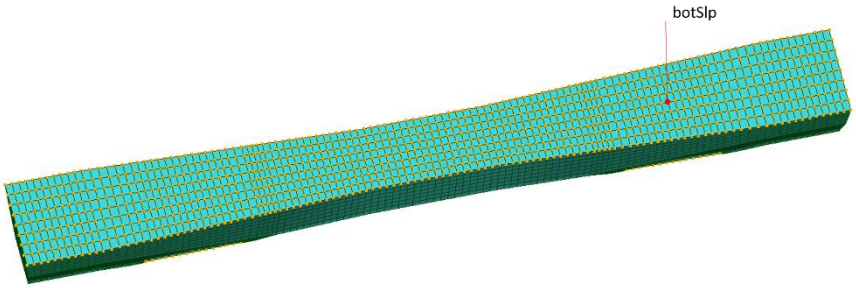


Figure 9: Sleeper's bottom face groups.

- The groups called: “sntpr”, “sntpl”, “setpr” and “setpl” (**S**leeper **N**odes **T**o **P**ad **R**ight, **S**leeper **N**ode **T**o **P**ad **L**eft, **S**leeper **E**lements **T**o **P**ad **R**ight and **S**leeper **E**lements **T**o **P**ad **L**eft) are used for the link with the pads. Right is the Z+ side and left Z- side.
- The node groups for the equivalent clamps, corresponding the to one on the rail are: “sntrlea”, “sntrleb”, “sntrlia”, “sntrlib”, “sntrrea”, “sntrreb”, “sntrria” and “sntrrib” (**S**leeper **N**ode **T**o **R**ail **R**ight **E**xterior **a** or **S**leeper **N**ode **T**o **R**ail **L**eft **I**nterior **b**). The final “a” or “b” follow the X-direction (i.e. X- : a, X+ : b), see Figure 10.
  The groups “sntrle”, “sntrli”, “sntrre” and “sntrri” are the counter part of the node groups from the rail to cumpute the pad compression. Currently only one of those groups is used, Figure 10.

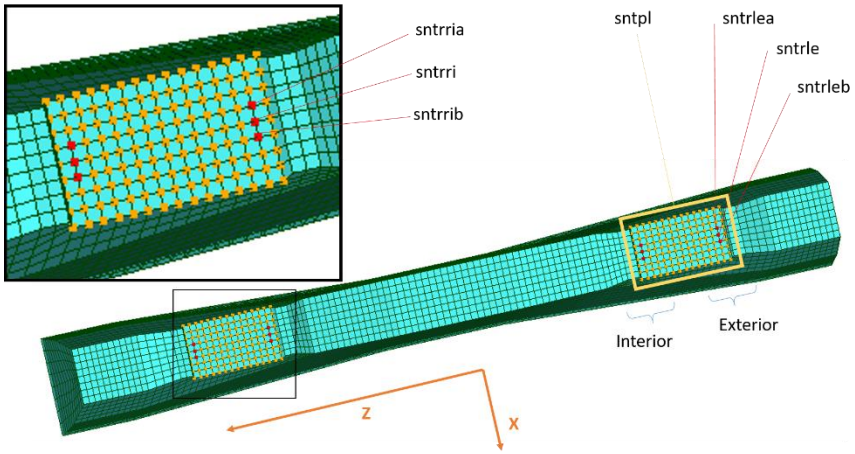


Figure 10: Sleeper's top face groups.

2.3.5. Clamps

The clamps are modelled as a pair of edges. Each edge as a stiffness and a damping coefficient. The Figure 11 shows those pair of edges on both side of the rail. Depending on the location of the node groups related to the equivalent clamps, the edges can be different.

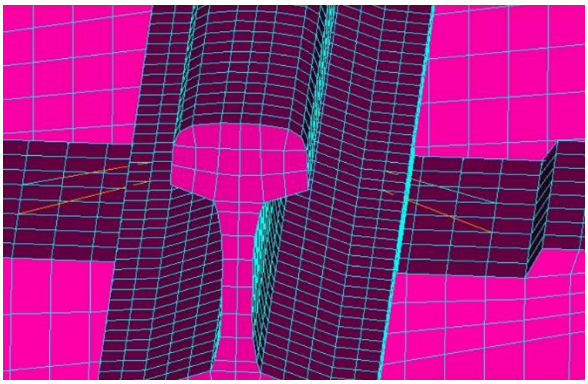


Figure 11: Illustration of the equivalent clamps.

2.3.6. Ballast meshes

The ballast mesh orientation is as follow: The origin is on the bottom face in the middle, the Z-axis is parallel to the sleeper, the X-axis perpendicular to the sleeper and the Y-axis vertical pointing upward, Figure 12.

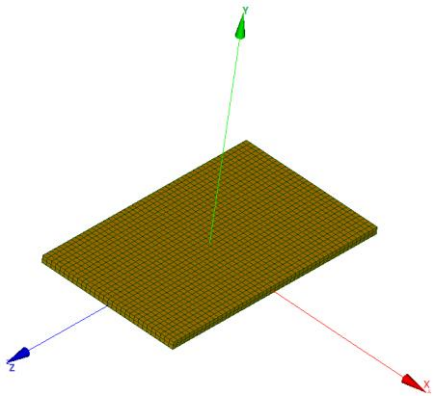


Figure 12: Ballast mesh and orientation overview.

Here are the groups required:

- All the elements of the ballast mesh have to be in the group called: “ballaste” (for **BALLAST E**lements).
- All the nodes and elements of the bottom face have to be in the groups called: “bnbot” and “bebot” (for **B**allast **N**odes **BOT**tom and for **B**allast **E**lements **BOT**tom).

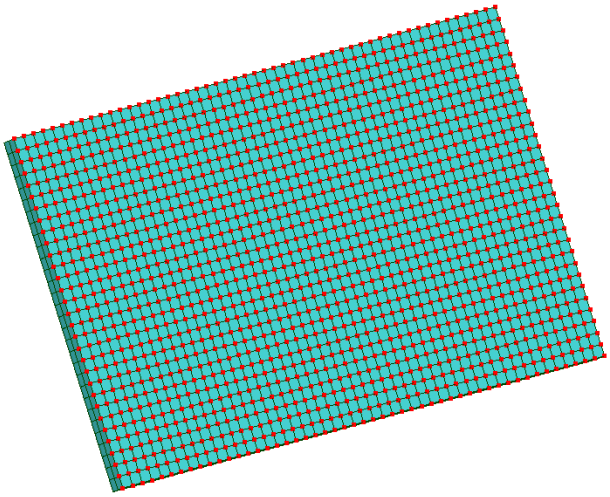


Figure 13: Ballast's bottom face nodes groups.

- There are three nodes and elements groups on the top face: “bnts1”, “bnts2”, “bnts3”, “bets1”, “bets2” and “bets3” (**B**allast **N**odes **T**o **S**leeper **1** or **B**allast **E**lements **T**o **S**leeper **1**). They are used for the contact between ballast and sleepers, see Figure 14.

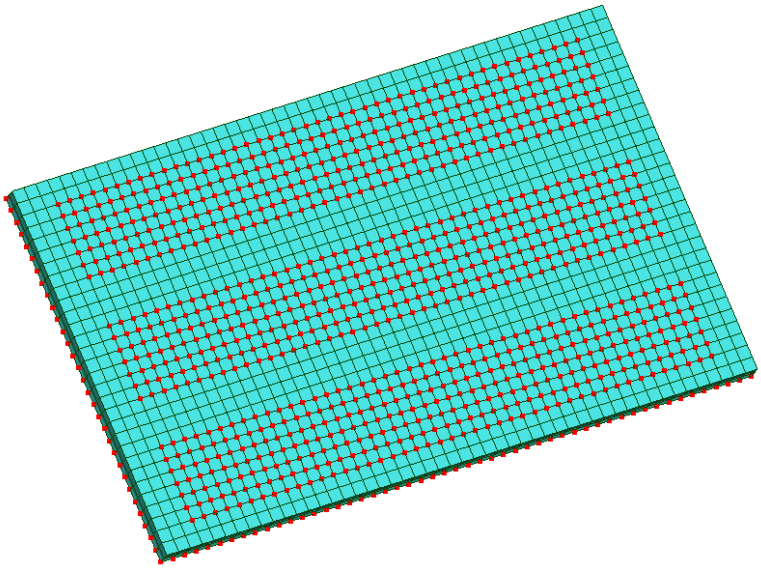


Figure 14: Ballast's top face nodes groups.

- The Figure 15, shows two groups of nodes: one of one node on the bottom of the sleeper and one of 7 nodes inside the ballast. Those groups are link to the extraction of data. The one on the sleeper has already been described and is called: “botSlp”. The second one is called: “balLoad” and, has one can see on the Figure 15, is located inside the ballast bellow the middle sleeper and aligned with the rail.

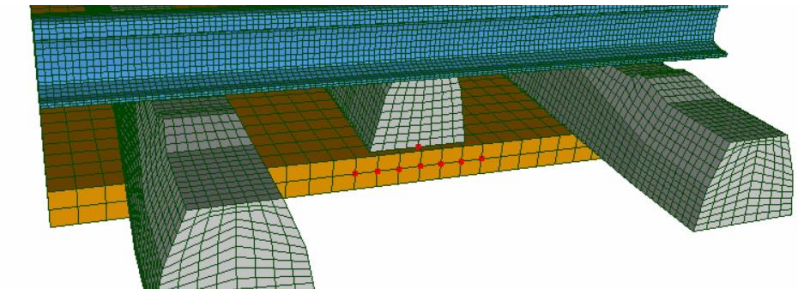


Figure 15: Mesh groups for the impulse model: seven nodes in the ballast to extract the stress and one node bellow the sleeper to estimate its displacement during a pass-by.

2.3.7. Three-sleeper full mesh generation (threeSleeperModel)

A tool was developed to insert six instances of a custom pad mesh into the three-sleeper mesh described above. It is located in *src/ThreeSleeperModel/ Generate3SleeperMesh*. Firstly, consider using the script "changePath.sh" to automatically update the hard-coded paths in the export file "gen3sleeper.export" (or *setup.sh* in the main directory). Then, paste a copy of your *.med* mesh file in the same folder and rename it "importedPad.mesh.med". Finally, run the script "GenerateMesh.sh". The generated three-sleeper mesh file is called "ThreeSleeperMesh.mesh.med", and it is highly recommended to import it in Salome and check if the pads were placed correctly between sleepers and rails.

2.4. Debugging

If a bug occurs, here are a few tips to help fixing it.

1. Close the GUI you are using; if the source code of the model contains a script called "clean.sh", run it in a new terminal. It will remove all unnecessary files or folders and allow restarting the GUI with a clean working directory. You can run *setup.sh* in the main directory as well.
2. Check all materials properties: some of them, such as Poisson's ratios, cannot take any value.
3. Look for information on the task manager if there is one, or on the terminal, which runs the GUI.
4. Check the memory and time limits set in the **.export* files. They must be adapted to the characteristics of the machine.
5. Analyze the steps of the simulation in the message files (usually in a folder folder called "Messages" in the source code of the model). Code_Aster writes down all the commands that were executed. If the error occurs in Code_Aster, it must be reported at the end of any of these files.

3. Three-sleeper Model

3.1. Description

The vibro-acoustic three-sleeper finite element model is a digital twin of the three-sleeper bench test (Figure 17). It allows performing harmonic simulations over a range of frequencies that usually goes from 300Hz to 1500Hz, and compute acoustic pressure fields using monopole superposition. It was mainly used to evaluate pad prototypes performances and to understand the influence of various physical parameters on rail tracks dynamics.

The model inputs are the force direction, a pad mesh, which can be bi-material, and frequency-dependent materials properties for the pads, the ballast, and the UnderSleeperPads (if activated). The main outputs are acceleration FRFs at different mesh points and the acoustic power FRF. Simulations workflow is illustrated in Figure 16, with computation time estimates from the machine used for development.

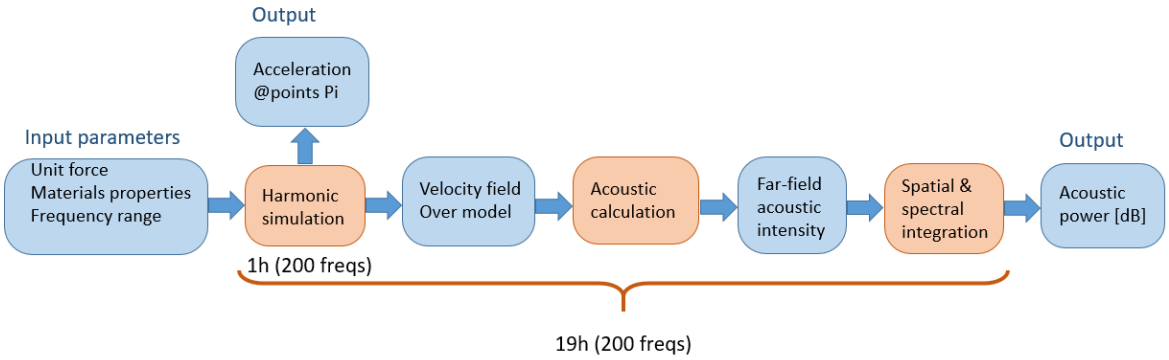


Figure 16: Three-sleeper model simulations workflow

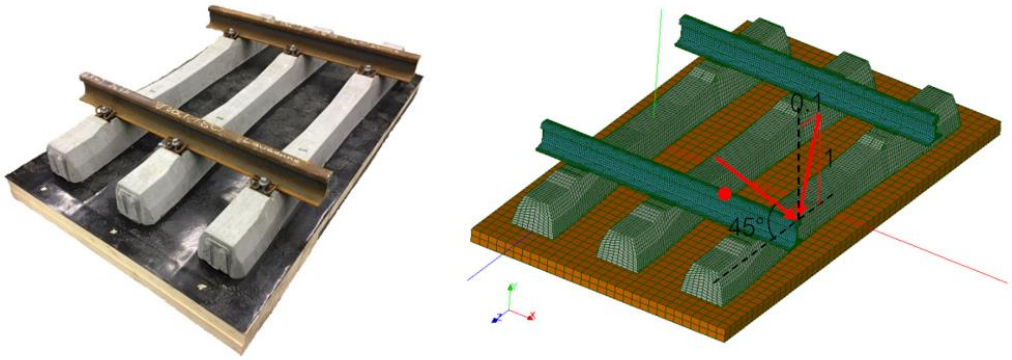


Figure 17 - three-sleeper bench test and FE model

3.2. Directories and files architecture

The three-sleeper model source code contains the following folders and files:

- *Materials_properties*: a folder which contains the files associated to the Young's modulus and damping coefficient of materials.
- *Meshes*: a folder where are located the 3-sleeper meshes, having different pad meshes. One can find also a mesh of the USPs and the acoustic mesh that is used.
- *Messages*: a folder where Code_Aster writes the message files from the last simulation execution. If a simulation fails, they provide an indication why.

- Eventually *base1*, a folder which contains some information about the last simulation FE mesh. It allows saving time by reusing some properties of the last mesh if it remains the same.
- The temporary folder *temp_parameters*, where are located one parameter file (*parameters_***.py*) per simulation defined. It is deleted once all simulations are done.
- The *.ui* and *.py* files generating the GUI.
- *clean.sh*, a file allowing to clean the working directory and restart to use the model and the GUI in case of eventual bugs.
- The files **.export*, in which one can adequately set some parameters such as memory and time limits. The **.astk* files, used to generate the **.export* files, are there as well.
- **.comm* files, *finalPostPro.py* and *RUN.SH*, which are executed by the GUI or by Code_Aster to run simulations properly.
- *Meshes_list.txt*, a file which allows defining, line by line, which 3-sleeper meshes are going to be available in the GUI.

Every time a simulation is run, all necessary files are copied to the main directory and the simulation is run from there. When it is over, all input and output files are copied to the directory specified by the user.

3.3. Define new simulation

Firstly, make sure the three-sleeper meshes needed are located in *./Meshes/*. To generate three-sleeper meshes, refer to 2.3.7 Three-sleeper full mesh generation. Then, type the name of your mesh in *./Meshes_list.txt* to allow your mesh to be used in the GUI.

1. Run the GUI. The main python script is *main.py*.
2. Start defining a new simulation by giving it a name and a directory to save the results in. Note that a folder with the name of the simulation will be created in the directory provided.
3. Choose whether the acoustic part must be computed (otherwise only the harmonic part is done, which is much quicker), whether the USPs (Under Sleeper Pads) must be included, and which load direction is required (45° or 10%=5.7°, see Figure 17).
4. Click on *Select frequencies and CPUs…* to define how many CPUs the simulation will use, and what the frequency bands are. Default ranges (300-1500Hz are proposed).
5. Select the mesh to use. This combo box is filled with respect to the content of the file *Meshes_list.txt* mentioned above
6. Since all pads are made of two regions, select the material to be applied to each region, namely *Material 1* and *Material 2*. You can either select a predefined material (EVA, PUsoft) or click on *Use custom…* and select the desired mechanical properties. Refer to 2.2.1Frequency-domain materials properties to know the formatting of *.csv* files.
7. Under the *Ballast properties* and the *USPs properties* tabs, define the ballast and the USPs properties (if the latter are activated) the same way it is done for a custom pad material. Note that the material corresponding to the wooden beams is proposed as ballast material by default.
8. Finally, click on *Add to list* to add this new simulation to the simulations list on the right.

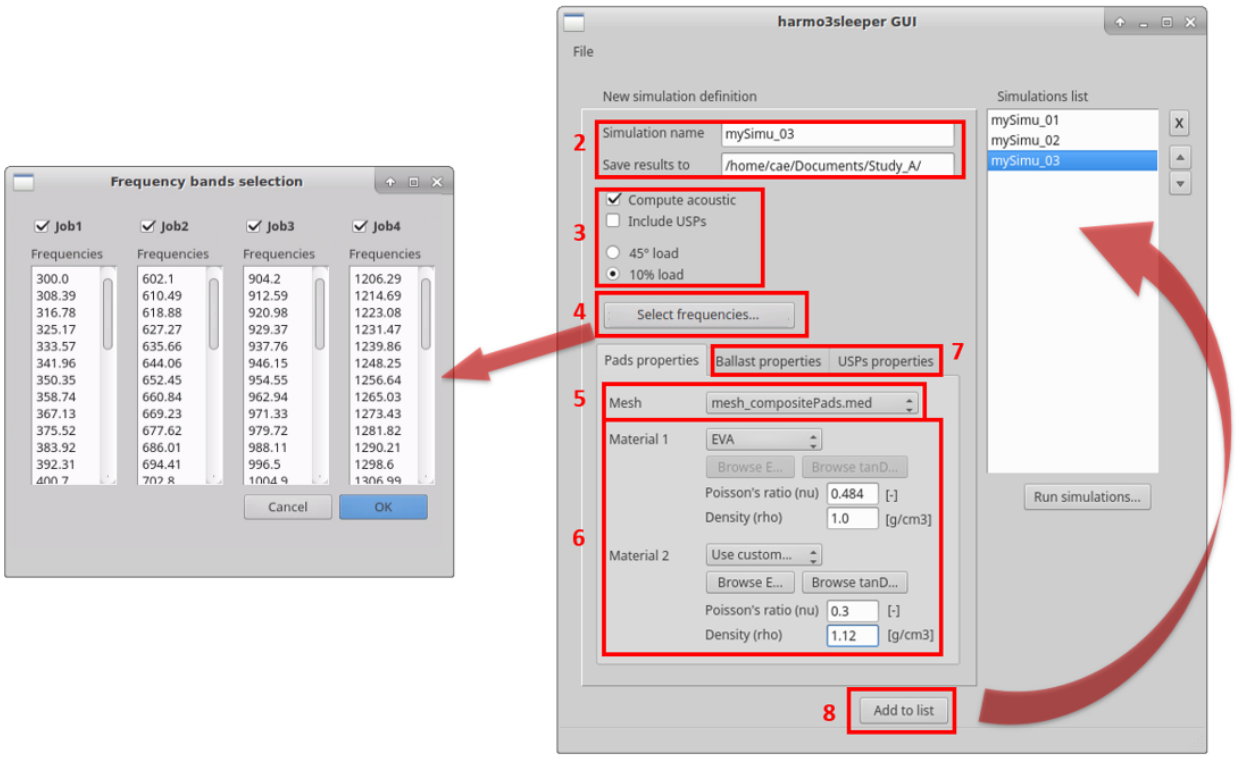


Figure 18: Three-Sleeper Model GUI

Important note: simulations parameters can be displayed by clicking on any simulation in *Simulations list*. However, simulations cannot be modified. The only way to edit some parameters is to select the desired simulation in the list, edit its parameters, and add the simulation to the list under a new name. Afterwards, the previous simulation can be deleted using the *X* button next to the list.

3.4. Run simulations

After defining the order of simulations (arrows), open the task manager by clicking on *Run simulations…*. Then click on *Run* and each simulation of the list will be run one after the other.

While the task manager gives real-time information on the execution, the main window of the GUI remains available to display the simulations parameters, if needed. However, to make any modifications on parameters or to create other simulations, you need to *Stop* the current simulation and *Close* the task manager before.

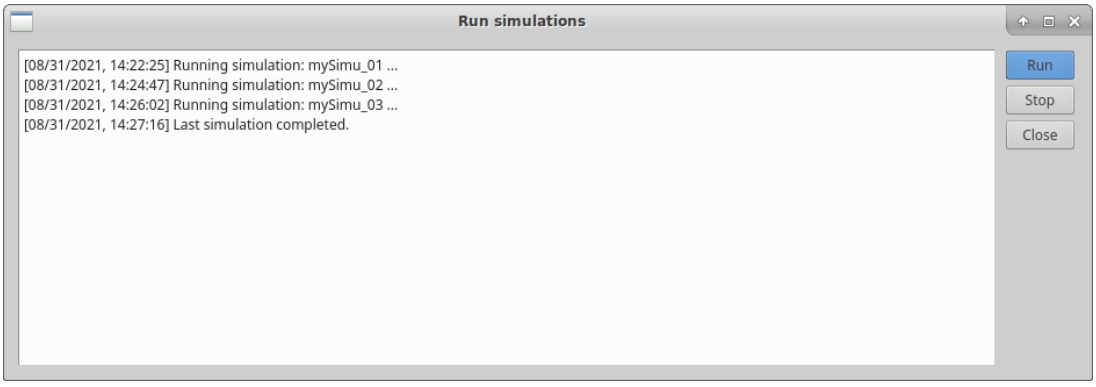


Figure 19: Three-Sleeper Model task manager

Note: once the process starts, various files and folders are created or moved to the main directory. For a standard usage, there is no need to know exactly how these files interact with each other. Anyway, at the end of each simulation execution, the main directory is cleaned and the results are saved to the desired location.

3.5. Access results

For each simulation, the result files that are created are the following:

- *resuHarm_b*.res.med*: up to 4 files, one per job (hence per frequency band), which allows visualizing the results of the harmonic part of a simulation in ParaView.
- *resuAcc_b*.res.med* (if the acoustic part was computed): up to 4 files, one per job, which allow visualizing the results of the acoustic part of a simulation in ParaView.

  One can typically display the deformed shape of the 3-sleeper unit cell at selected frequencies as well as the acoustic pressure or intensity field over the far-field hemisphere around the setup like in the figure below.

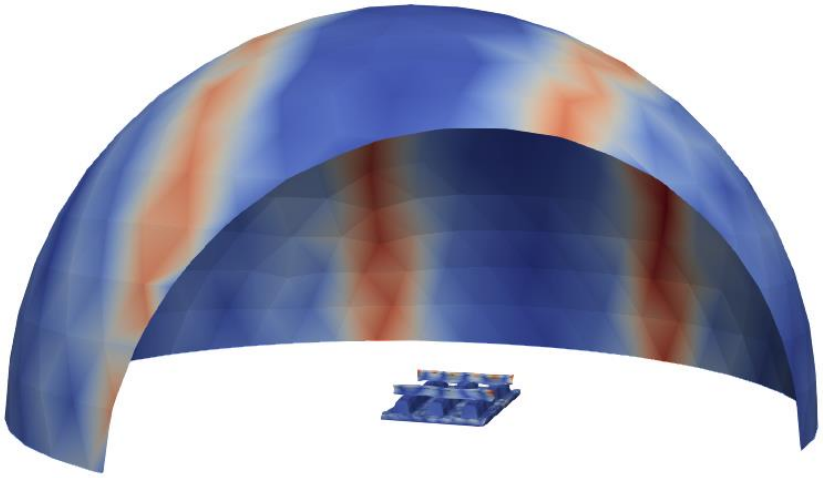
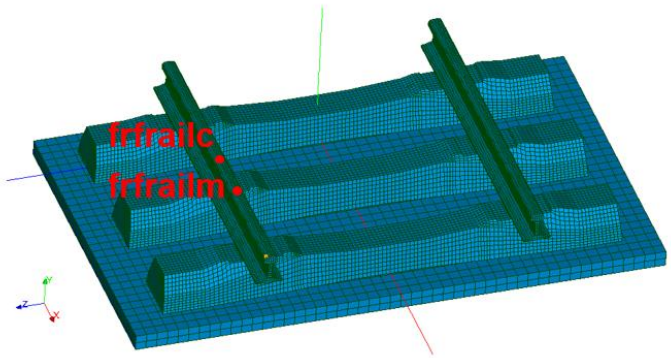


Figure 20: Three-Sleeper model results visualization

- *FRF_concat.txt*: a list of accelerances (in g/N) obtained in the vertical (Y) and lateral (Z) directions at the two points shown below, for each frequency calculated.



FREQ	frfrailmY	frfrailmZ	frfrailcY	frfrailcZ
300.0	0.0015197212636483336	0.003663115155144526	0.0035857813669170234	0.004270883243542107
318.6	0.0002461741424427647	0.0018847202721054784	0.0015704840620844457	0.0011035095874805654
333.26	0.0002349605487982603	0.0014048849977851012	0.000938065686448007	0.0005449148165104546
344.8	0.0003222879681624175 7	0.0012994561000965988	0.0006634621194853611	0.0003789624887527581

Figure 21: FRFs computation

- *press_concat.txt* (if the acoustic part was computed): the acoustic power spectrum values $P_i(f)$ (in W/N^2) obtained at each frequency. There is a column for the contribution of all radiating components together (ballast, sleepers and rails) and a column for each contribution. Note that the sum of all contributions is not equal to the total power because only the complex acoustic pressure can be directly summed, in opposition to the acoustic power. On top of the file is displayed the total acoustic power L_w (in dB(W/N^2)), computed as

$$L_w = 10 \cdot \log_{10}\left(\int_{f_{min}}^{f_{max}} P_{tot} \cdot \mathrm{d}f\right)$$

  using the trapezoidal rule for numerical integration.

```
Lw [dB] -37.6616922882

Acoustic power [W/N2]
Freq [Hz]       Total   Rails   Sleepers        Ballast
300.0   1.5155318533826064e-07  2.542237837193316e-08   5.0766605057365045e-08  4.896589199719414e-09
318.6   3.419593203227423e-08   3.3745180671137776e-09  1.4854193158459563e-08  1.755305704641109e-09
333.26  1.8652098988964488e-08  1.2689258087932248e-09  1.0064987178408878e-08  1.1797048034991775e-09
344.8   1.5492807466998943e-08  1.0039115232016272e-09  9.051781443373615e-09   1.0898717762607688e-09
```

Figure 22: Acoustic power results

- *allPressureDataPicked* (if the acoustic part was computed): it contains all acoustic pressure related data in a Python format. This data is available as well in the *resuAcc_b*.res.med* files, but the Python format allows carrying out further analyses more easily.

```
1  import pickle
2
3  file = '/home/cae/Documents/Study_A/mySimu_01/allPressureDataPickled'
4  pickle_in = open(file,"rb")
5  data = pickle.load(pickle_in)
```

Figure 23: Use of Pickle module to read acoustic data

A simple script like the one showed above generates the variable *data*. It is a list of dictionaries, each of them being related to an acoustic mesh node. Each dictionary contains the following keys and values:

  - *ID*: the ID of the acoustic mesh node
  - *coords*: an array with the X, Y and Z coordinates of the node
  - *freqs*: the list of simulated frequencies
  - *p_*_R* and *p_*_I*: some lists of real and imaginary pressure values, respectively, associated to the simulated frequencies (in the same order). The star character (*) here stands either for *total* (the sum of all pressure contributions), *rails*, *sleepers* or *ballast*.

- *parameters.py*: this Python script is read by various files and defines completely a simulation. It is firstly created in *./temp_parameters/*, a folder which contains the parameter files of all simulation that are about to be executed, and is removed as soon as the executions are over or if the GUI is closed. It remains useful even for a standard usage of the model to identify what were the materials properties, force applied, etc. of a completed simulation.

4. Multi-sleeper model

4.1. Description

The multi-sleeper model is an extension of the three-sleeper model to a much larger number of sleepers. It performs harmonic vibro-acoustic FE simulations as well, in order to make actual tracks dynamics predictions and evaluate the acoustic pressure at some locations as sensors would do. It uses three main techniques to reduce computation time: dynamic substructuring, modes pre-computation and reuse, and jobs parallelization.

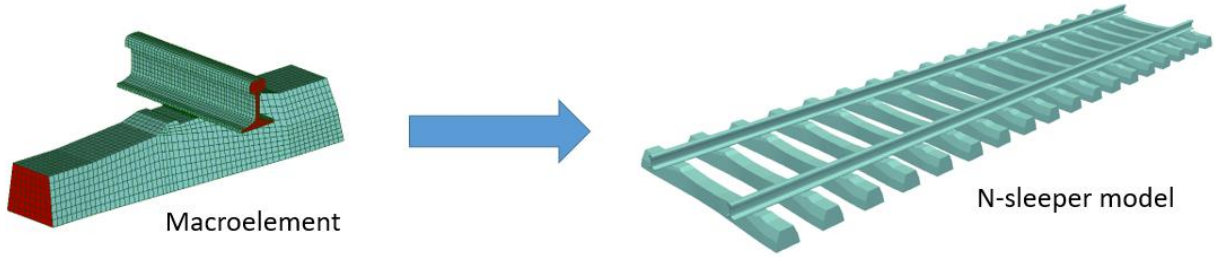


Figure 24 - multi-sleeper model generation using dynamic substructuring

A harmonic simulation, which takes an hour with the three-sleeper model, is reduced to less than five minutes with this model. As for the acoustic part, time is reduced from 19 hours to 5 hours with comparable simulations. These values were observed from the machine used during development (128GB of RAM, 24 CPUs). As mentioned in 1 Introduction,

4.2. Directories and files architecture

The diagram in Figure 25 summarizes the main steps in the simulations. They are performed in two phases. Every time a new macroelement mesh is used, Phase 1 needs to be run. A macroelement meshes, or *configurations*, can only differ from each other in terms of pad mesh and USP inclusion. Since computing macroelements modes is computationally expensive, the goal is to do it once per configuration and generate a *Base* (folder containing modal information) meant to be reused.

To summarize, Phase 1 runs a harmonic simulation of a three-sleeper-long track using dynamic substructuring (1a, 1b). The modes largest contribution and the modes shapes are saved in the Base folder defined by the user.

In Phase 2 (*i.e.* the main simulation with more sleepers), the Base is loaded and a modal basis is created using the *N* most contributing mode shapes (2a). The value of *N* is a user input. One can decide to use a few modes only to speed up simulations, eventually at the cost of accuracy. Updated assembled matrices are projected on the modal basis, such that updated modes are computed in generalized coordinates very quickly. Then the multi-sleeper model is generated using dynamic substructuring and the main harmonic simulation is run (2e).

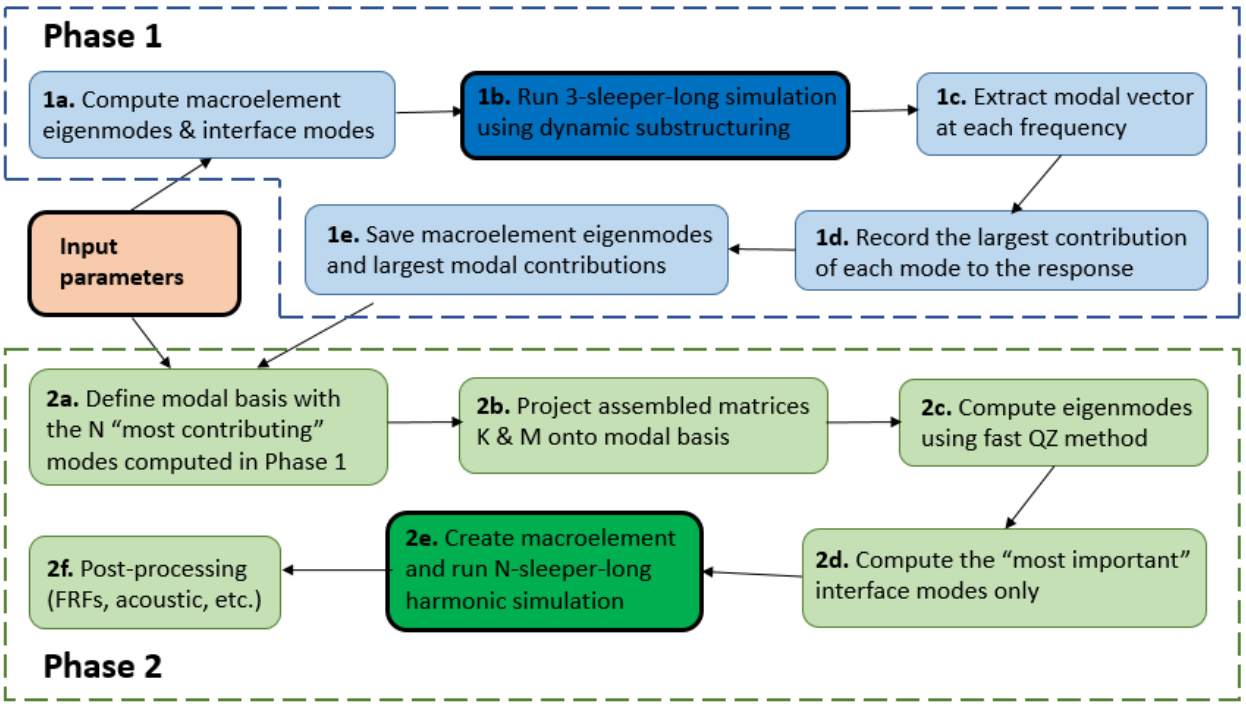


Figure 25 - multi-sleeper model simulations steps

The Multi-Sleeper Model main folder contains the following folders and files:

- *AsterFiles*: folder where all *.comm* and *.export* files are located. When for example the simulation *SimuX* is run, a folder *./SimuX_phase1/* is created if Phase 1 computation was planned, as well as a folder *[savingDirectory]/SimuX/*. All necessary files, including command files and export files, are copied to these folders where simulations are run.
- *Bases*: folder where some base folders can be saved. Note that it is a suggestion, bases can be saved anywhere. A base folder contains: the folder *base1_bi* where the mode shapes are stored, the script *modesNumber.py*, which allows passing the information to Phase 2 what was the number of eigenmodes and interface modes computed, and *modesMag[dir].txt* where modal contributions are stored as illustrated below.

	Mode number	Substructure 1	Substructure 2	Substructure 3	Substructure 4	Substructure 5	Substructure 6
Eigenmodes	1	5.22E-01	4.24E-01	2.67E-01	1.92E-01	6.78E-01	5.13E-01
	2	3.25E-01	2.04E-01	5.98E-01	5.62E-01	1.31E+00	1.04E+00
	3	2.60E-01	2.36E-01	1.41E-01	1.08E-01	3.10E-01	2.68E-01
	...	...	...	...	...	...	...
Rail interface	115	2.78E+00	9.95E-01	3.11E+00	7.76E-01	3.66E+00	7.15E-01
	116	1.45E+00	8.78E-01	1.02E+00	8.70E-01	1.21E+00	1.83E+00
	...	...	...	...	...	...	...
Sleeper interface	315	1.06E-01	1.10E-01	4.49E-02	4.67E-02	1.22E-01	1.22E-01
	316	3.40E-02	4.12E-02	6.36E-02	5.11E-02	1.27E-01	5.75E-02
	...	...	...	...	...	...	...

Figure 26: Largest modal vector components among the frequency spectrum

- *MaterialsProperties*: folder with *.csv* files containing materials storage modulus and damping ratio as explained in 2.2 Materials modelling.
- *Meshes*: folder storing pad meshes, acoustic meshes, as well as the meshes used for the portions of USP, sleeper and rail.
- *Messages*: folder where Code_Aster messages are written for every job run.

- *.py, .sh and .ui* files used to run the GUI and the model properly.

4.3. Define and run simulations

1. Run the GUI. The main script is *main.py.* Do not forget that **most widgets have tooltips explaining how to proceed**. Place your mouse on textboxes, buttons, etc. to display them.
2. Tab Base: if a Base was already created for the configuration (pad mesh, USP inclusion), one can select its folder. Otherwise, check "(Re)Run Phase 1" and create a new Base. Some default parameters are proposed. Note that Phase 1 simulations are always run using 4 jobs. So choose the number of CPUs accordingly to your machine. Finally, choose the frequency range which must be simulated. Cells can be copied from/to spreadsheet programs. The frequency list will be divided into the four jobs (*i.e.* frequency bands).
3. Tab Computing: select the computing and files parameters referring to the main simulation (*i.e.* Phase 2).

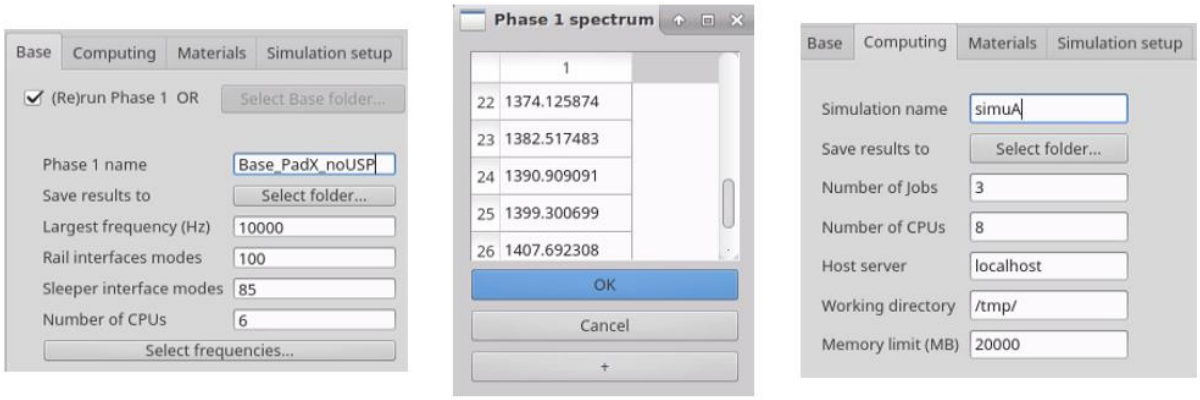


Figure 27: Multi-Sleeper Model GUI

4. Tab Materials: select the pad mesh as described in 2.3.1Pad meshes. Again, remember that the mesh selected must correspond to the one used when creating the Base. Only the materials properties can differ. Then select the .csv files for Young's modulus (storage) and damping for Materials 1 and 2, the ballast, and the USPs if included. Ballast properties allow computing discrete springs and dampers properties.
5. Tab Simulation setup: select the main parameters of the simulation. The number of modes to use, where to apply the force, the force direction (orientations correspond to Figure 17) and the frequencies to simulate (again, cells can be copied/pasted from/to spreadsheets). Post-processing consists, firstly, of velocity (m/(s.N)) or acceleration (g/N) FRFs on a set of nodes the desired sleepers. More details are explained in 4.4Access results. Secondly, it consists of the acoustic part (which can be turned on or off). If the mesh is 1D, acoustic pressure FRFs will be computed at each acoustic mesh point. If it is 2D, acoustic intensity will be integrated over the surface in order to compute the acoustic power spectrum FRF.
6. Finally, click on "Add simulation to list" to create a list of simulations and click on "Simulate all". The real time simulations progress is displayed in the terminal.

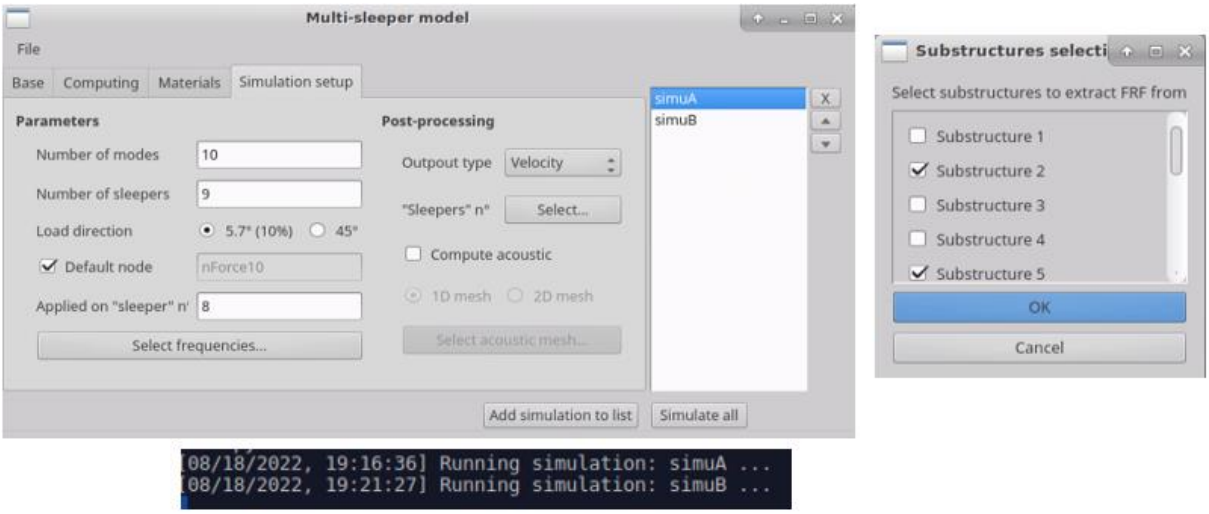


Figure 28: Multi-Sleeper Model GUI

4.4. Access results

Each simulation folder contains an *Inputs* folder, where all files necessary to run the simulation are copied, as well as an *Outputs* folder, which is detailed below. Finally, the simulation folder contains all *.export* and *.comm* files used, as well as a *.json* file containing all information about a simulation (files, parameters, etc.). The *Outputs* folder contains the following files:

- *skeleton.med*: MED file which can be used to visualize the assembled substructures. Like all MED files, it can be opened in ParaView.
- *resultsHarmonic.med:* MED file containing the velocity or acceleration field in the structure at all simulated frequencies.
- *modes_b1.med*: MED file with the macroelement eigenmodes, which were used in the simulation. From one job (frequency band) to another, they may be slightly different due to the frequency-dependence of some materials properties (pads, ballast). By default, the ones of the first job only are printed. This is valid for the other mode files described below.
- *modeRail*[i]*_b1.med*: MED file with the macroelement interface modes of the i^{th} rail interface (i=1,2), which were used in the simulation.
- *modeSlp_b1.med*: MED file with the macroelement interface modes of the sleeper interface, which were used in the simulation.
- *mesh.med*: mesh file of the macroelement, assembled with the pad mesh input.

  FRF.txt: text file with, all velocity or acceleration FRFs, column by column. FRFs are computed at all specified sleeper numbers, and for all groups of nodes specified in the variable *FRFgroups*, hard-coded in *phase2_runSimulation.comm* (*i.e.* the command file of the second phase). Note that each FRF corresponds to the FRF averaged over all nodes contained in the mesh group a mesh group, in order to avoid computing a FRF on a modal "node".

  The header of each column is *E[i]_[XY]_[dir]*, followed by the units. *E* just stands for *excited*, such that the FRF was computed on the group of nodes *[XY]_[dir]* of the i^{th} sleeper excited side. *XY* is either:
  - *Ra*: rail above; it is the rail head above the i^{th} sleeper.
  - *Rb*: rail between; it is the rail head between the i^{th} and the $(i+1)^{th}$ sleepers.
  - *Rf*: rail foot. It is the side of the rail foot between the i^{th} and the $(i+1)^{th}$ sleepers.

- *Sc*: ith sleeper center (top face).
- *Ss*: ith sleeper side (top face).
- *Ba*: ballast; It is on the bottom face of the ith sleeper, right below the pad.

Finally, *dir* is either the *Y* direction (vertical) or the *Z* direction (lateral).

- *acousticResults.txt*: if the acoustic computation was done, this text file is written and its content depends on the acoustic mesh type. If "2D" was selected, it provides the integration of the acoustic intensity over the surface, in other words, the acoustic power spectrum FRF. The format is exactly as explained in 3.5 Access results for the file *press_concat.txt*.

  If "1D" was selected, it provides the acoustic pressure FRF at each node of the acoustic mesh. For each node *N[i]*, there is also the total pressure, the pressure coming from the rails, the pressure coming from the sleepers, and the pressure coming from the ballast (which is zero since the latter was replaced by discrete elements).

- *acPressData_pythonPickle*: all acoustic pressure data gathered in a file, using the Pickle Python module. Refer to *allPressureDataPicked* in Section 3.5 Access results for more information.

5. Pad stiffness model

5.1. Aim of the model

The Pad Stiffness Model aim to estimate the stiffness of the input pad design. It is a digital twin of an experimental compression system.
After validation in static and low dynamic (10 and 20 Hz) compression with the experimental results, the model has been extended to provide the stiffness in various directions and with an extended frequency range.

5.2. Inputs

This model need three inputs:

1. The mesh of the pad.
2. The material properties (up to two materials per pad).
3. The frequencies to compute.

5.3. Outputs

In outputs, the user gets a CSV file containing the complex stiffness of the pad in four directions:

1. Compression.
2. Rotation of the upper face of the pad around the X-axis (parallel to the rail).
3. Rotation of the upper face of the pad around the Y-axis (parallel to the sleeper).
4. In shear, horizontal displacement of the upper face in the Y-axis (parallel to the sleeper).

For each frequency, there are the Real and Imaginary part of the stiffness in N/mm.
There is also a CSV file containing the static stiffness in compression of the pad.
The results are also available in .MED format (Salome_Meca format) if needed.

5.4. Boundary conditions and interactions

When the model is run through the GUI, a script is run that add a reference point (a 0 dimensional element) above the upper face. A solid link is made between the reference point and the upper face nodes. Thus, the reference point pilot the upper face.
There are many loading cases: compression, two rotations of the upper face and shear. For the three first loading cases, all the displacements except the vertical one are blocked (i.e. the upper face nodes can move only vertically). For the last case only lateral movement of the upper face nodes are allowed.
Depending on the loading case, a vertical downward force, a moment around the X-axis, a moment around the Y-axis or a shear force in the Y-axis is applied. The X-axis is parallel to the rail and the Y-axis is parallel to the sleeper.
In any cases, the bottom face is fixed in every direction.
The resulting displacement of the reference point is used to compute the stiffness in the specified directions.

5.5. Directories and files architecture

The Figure 29 shows the architecture of the Pad Stiffness model and its GUI.

The main folder, PadStiffness_GUI, contain the files related to the GUI and 4 main directories:

1. Frequencies
2. Materials
3. Meshes
4. Working directory

There is an additional folder, Example result, which shows the format of the result folder that is created during a simulation.

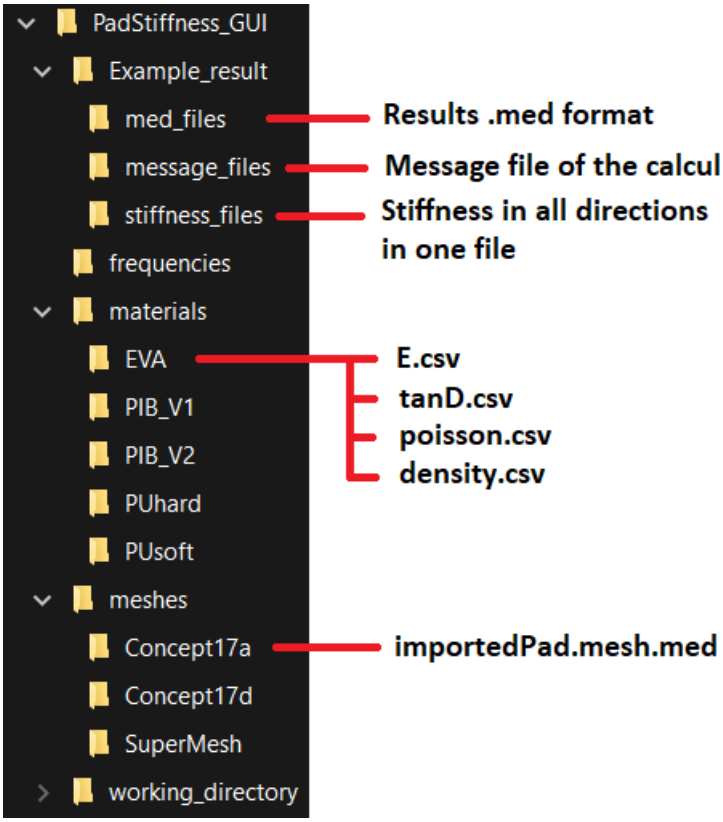


Figure 29: Pad Stiffness Model and GUI directories architecture.

The frequencies folder contains four text files, in which the list of frequencies is listed (one frequency per line). There are four files because the simulation is split in four batch of calculus, which run in parallel.

As one can see on the Figure 29, the materials folder contain the list of materials available. Each material folder contains the same four files, as described in the material section: 1) E.csv, 2) tanD.csv, 3) poisson.csv and 4) density.csv.

Meshes folder contain the list of pad mesh. If a new pad is added manually, it has to be named "importedPad.mesh.med" and the parent folder name will be the name of the mesh in the GUI.

Finally yet importantly, the working directory contain the files related to the calculation of the model, especially the .comm and .export files that define the model.

5.6. Define the frequencies

The main folder, PadStiffness_GUI, contain a folder called “frequencies”. In this folder, there are four files: “frequencies_b1.txt”, “frequencies_b2.txt”, “frequencies_b3.txt”, and “frequencies_b4.txt”.

Those files contain the list of frequencies that will be computed by the model. There are four files because the resolution is parallelized and four process run side by side.

If the user wants to change, remove or add frequencies for a new calculation, simply modify those files by hand.

5.7. Define a new simulation

To start the user interface, open a terminal in the PadStiffness_GUI folder and run the following command: “python pad_stiffness_model_gui.py”.

The Figure 30 shows the main window of the GUI. For a new simulation, those few steps are needed:

1. Give a name to your simulation
2. Enter the absolute path of the saving location (/home/user/Documents)
3. Select the mesh that you want (i.e. the pad).
4. Select the first material
5. Select the second material
6. Click on “Run Simulation”
7. Wait
8. Close the four calculation windows
9. Check the results (/home/user/Documents/MySimuation)

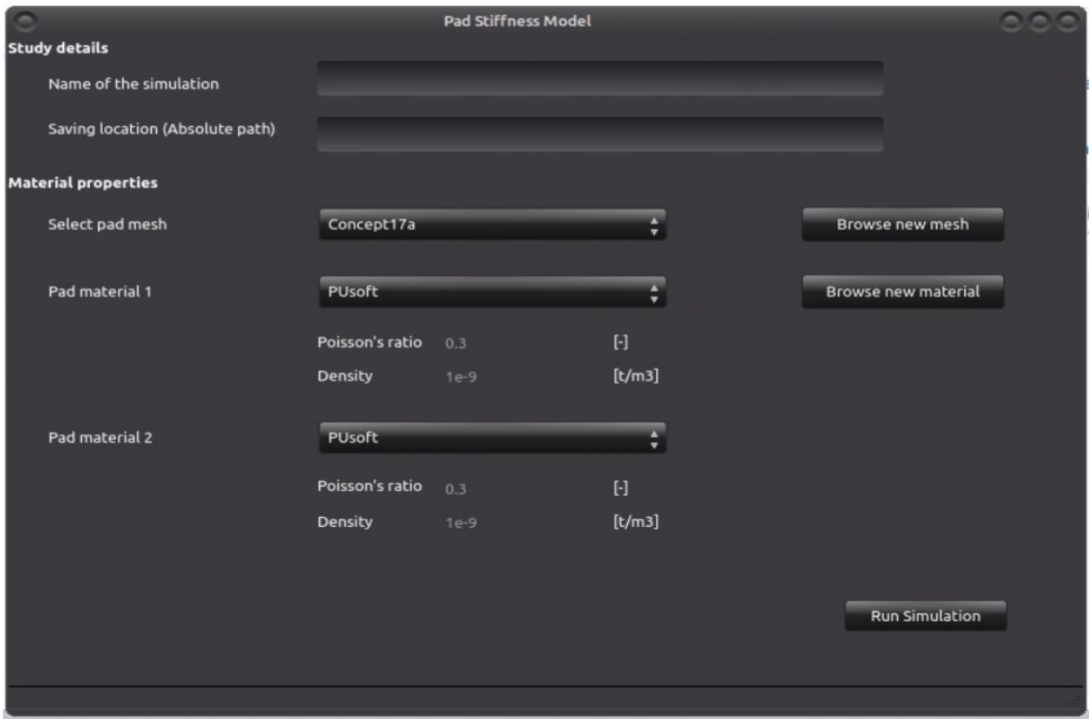


Figure 30: Main window of the padStiffness Model GUI.

5.8. Add a new mesh

5.8.1. Using the interface

The GUI allow the user to add a new rail's pad mesh and/or a new material.

The Figure 31 shows the window to add a new mesh. To add a new mesh there are three steps:

1. Name your pad
2. Select the file on your computer
3. Click on Add mesh

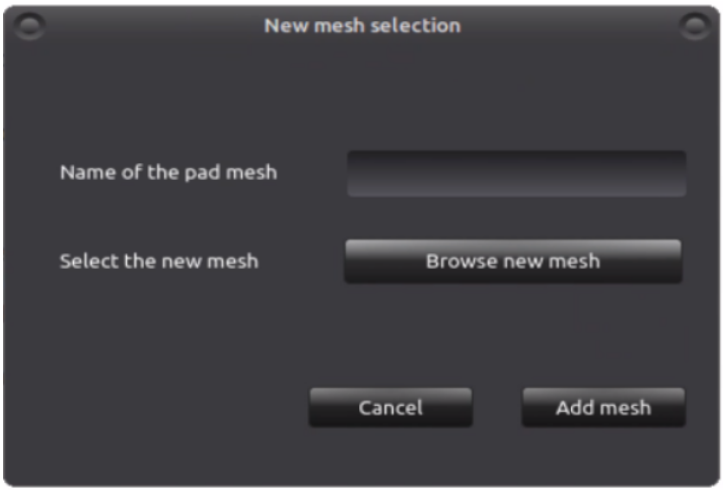


Figure 31: Add mesh window.

To know the specificities of the pad mesh, please refer to 2.3.1 Pad meshes.

5.8.2. Manually

To add a mesh manually, one need to create a folder with the mesh name inside the "meshes" folder. Then simply move the mesh file inside this new folder, the name of the mesh should be "importedPad.mesh.med".

For the exact format of this file and the mesh, refer to the dedicated section.

After re-starting the GUI, the new mesh should be present in the list.

5.9. Add a new material

5.9.1. Using the interface

The Figure 32 shows the window for adding a new material.

To add a new material six steps are needed:

1. Give a name to your material
2. Select the file containing the Young modulus (frequency dependent)
3. Select the file containing the tan(delta) (frequency dependent)
4. Enter the Poisson's Ratio ($0 < a \leq 0.495$)
5. Enter the density in [t/mm^3]
6. Click on Add material

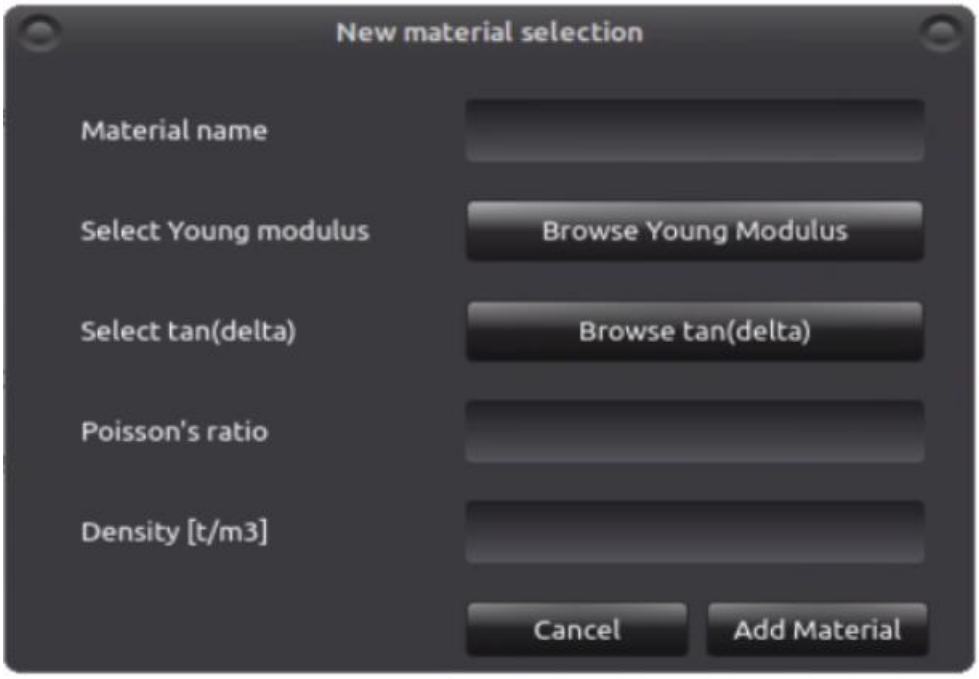


Figure 32: Add a new material window.

5.9.2. Manually

To add a material manually, one need to create a folder with the material name inside the "materials" folder. Then simply move the material files inside this new folder. The files needed for a material are:

1. E.csv
2. tanD.csv
3. poisson.csv
4. density.csv

For the exact format of those files, refer to the dedicated section.

After re-starting the GUI, the new material should be present in the list.

5.10. Access the results

Once the calculation is finished, the results files in med format are copied in the folder selected by the user (step 1 and 2 in the main window).

The file copied are the followings:

1. The .med files that can be open with salome-meca to check the results.
2. The stiffness calculated for each frequencies, one file that contain everything and four files corresponding to the four frequency batch.
3. The message file from the calculation.
4. A file containing a brief summary of the parameters of the study.

6. Impulse model

6.1. Aim of the model

The Impulse model is a variation of the three sleepers model, operating in the time domain and not the frequency domain. It aims to reproduce a pass-by of a bogie on a sleeper. The load of an axel is spread over three sleepers. The loaded sleeper holds around 55% of the load and around 22% on the direct neighbors. The other sleepers hold a tiny fraction of the load.

6.2. Inputs

This model need seven inputs:

1. The mesh of the rail.
2. The mesh of the pad.
3. The mesh of the sleeper.
4. The distance between sleepers.
5. The mesh of the ballast.
6. The material properties (for every meshes).
7. The clamps properties.

6.3. Outputs

In outputs, the user gets three CSV files containing:

1. The displacement of the sleepers bottom surface and of the upper and lower contact point of the clamp.
2. The stress invariants (Von Mises, Tresca…) in the ballast at specific locations.
3. The stress in the ballast at specific locations.

The results are also available in .MED format (Salome_Meca format) if needed.
In addition the mesh assembly is also available in .med format.

6.4. Boundary conditions and interactions

The boundary conditions and parts interactions are the same than the three sleepers model: Bottom surface of the ballast fixed and all parts glued together.
The difference is the loading conditions. For this model, the force is applied vertically downward on both rails. The force application location is on top of the rail on top of the middle sleeper (on node group “noeuForc”). The load vary with the time and have an “M” shape, see Figure 33. A weight of 22 tons per axle have been used to compute the maximum load applied, which is around 108 kN per rail.

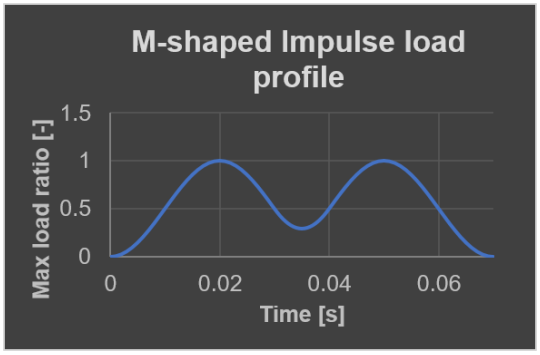


Figure 33: Load profile: ratio of the max load applied.

6.5. Directories and files architecture

The Figure 34 shows the architecture of the Impulse model and its GUI.

The main folder, Impulse_GUI, contain the files related to the GUI and 4 main directories:

1. Materials
2. Meshes
3. Working directory

There is an additional folder, Example result, which show the format of the result folder that is created during a simulation.

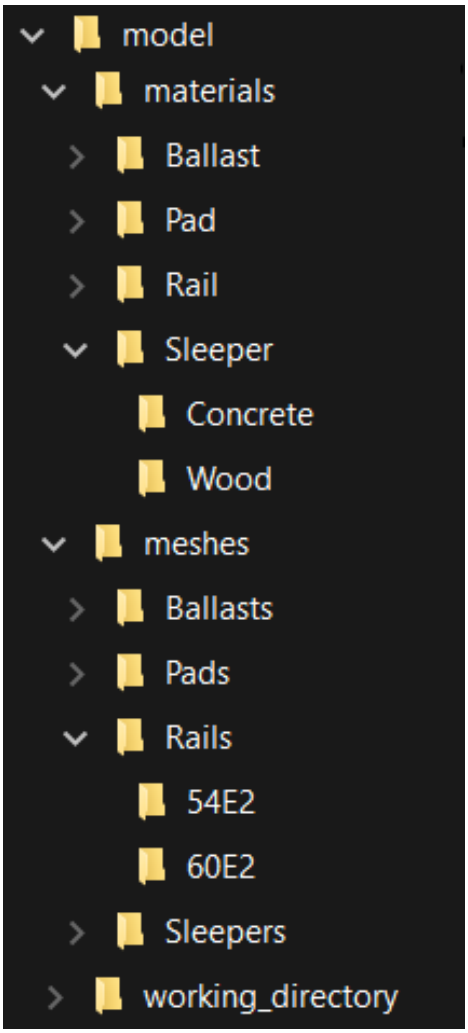


Figure 34: Impulse Model and GUI directories architecture.

As one can see on the Figure 34, the materials folder contain the list of materials available for each parst of the assembly. As an example the sleeper folder contain two materials: concrete and wood (those names are the material name in the GUI).

Those last folders for every type of parts except pads have to contain the following files:

1. sleeper_AmHyst.csv -> store the hysteretic damping properties.
2. sleeper_E.csv -> store the Young modulus.
3. sleeper_Nu.csv -> store the Poisson coefficient.
4. sleeper_Rho.csv -> store the density.

Of course “sleeper” should be replaced by the type of part: ballast, rail or sleeper.

For the pad properties, as they are visco-elastic material modelled as a Maxwell generalized series, each material folder contains the same four files, as presented in the dedicated section:

1) E.csv
2) G.csv
3) K.csv
4) tau.csv
5) poisson.csv
6) density.csv.

Meshes folder contain the list of mesh for every type of parts. For example, for the rail there are the 60E1 and 54E2 types. If a new mesh is added manually, it has to be named pad.med, rail.med, sleeper.med or ballast.med, depending on the part type, and the parent folder name will be the name of the mesh in the GUI.

Finally yet importantly, the working directory contain the files related to the calculation of the model, especially the .comm and .export files that define the model.

6.6. Define a new simulation

To start the user interface, open a terminal in the ImpulseAster_GUI folder and run the following command: “python impulse_model_gui.py”.

The Figure 35 shows the main window of the GUI. For a new simuation, those few steps are needed:

1. Give a name to your simulation
2. Enter the absolute path of the saving location (/home/user/Documents)
3. Select the mesh that you want for every type of parts.
4. Select the material that you want for every type of parts (two different materials for the pads).
5. Enter the distance between the sleepers*
6. Enter the value for the clamps (global reference system is used).
7. Click on “Run Simulation”
8. The meshes are assembled and a Salome window is open to visualized the final mesh**.
9. If everything is OK click on the corresponding button on the pop-up window.
10. Wait
11. Close the calculation window
12. Check the results (/home/user/Documents/MySimuation)

* Obviously the nodes and elements groups on the different meshes have to match the desire distance between the sleepers. The nominal value is 600mm.

** The Salome window is by default open without any module activated. Click on the “mesh” module and then on the “eye” icon in the object browser, see Figure 36.

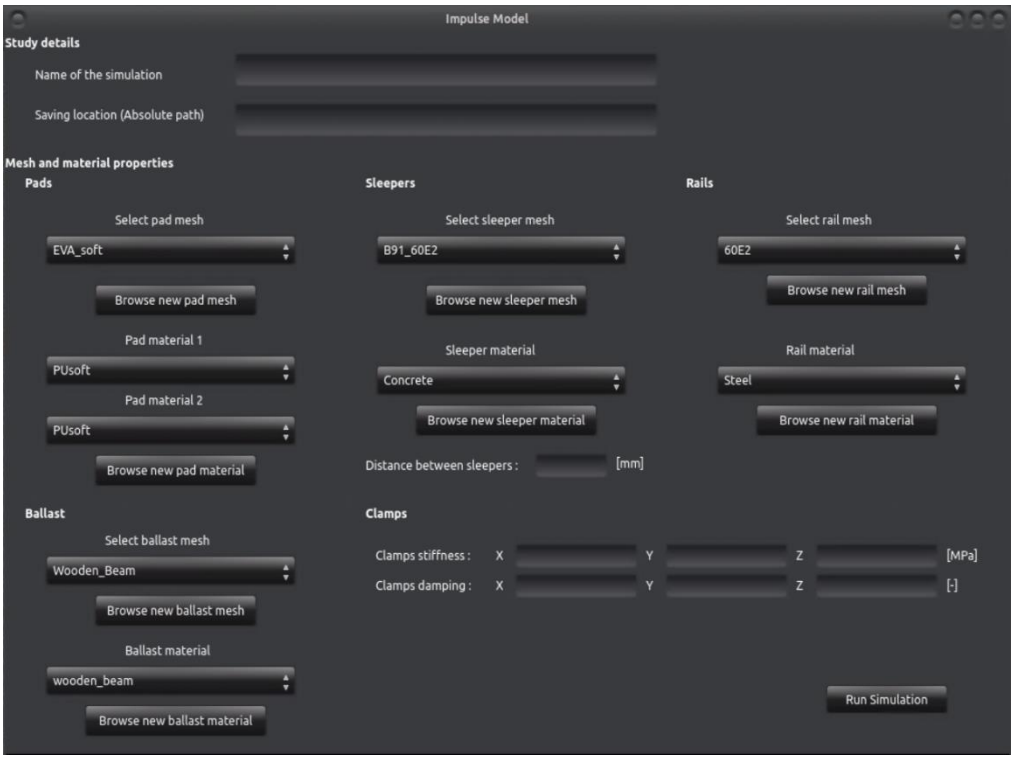


Figure 35: Main window of the Impulse Model GUI.

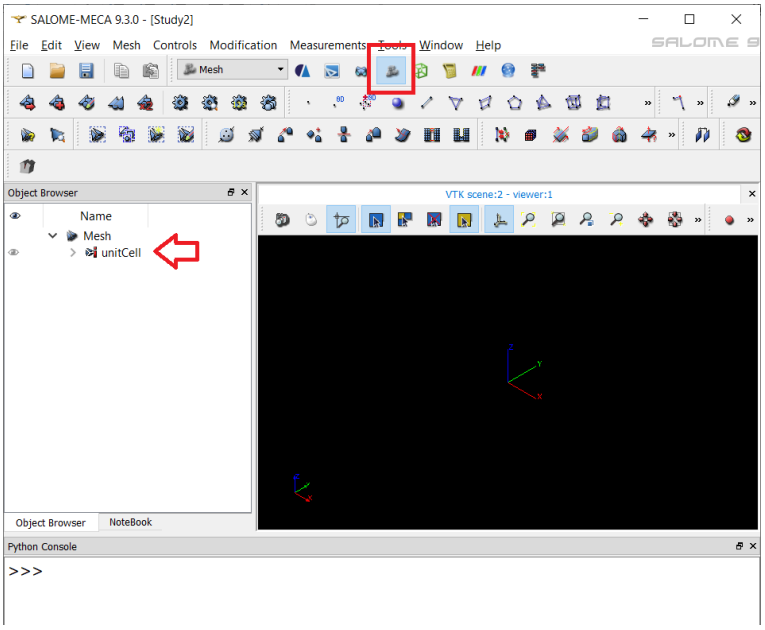


Figure 36: Salome platform: Red square is the mesh module button, red arrow is the current mesh imported.

The model can alternatively be run using a command line:

```
cae@mycomputer:~/Documents$ python Impulse_commande_line.py data.json
```

Data.json is a JSON file containing all the parameters to run the study, see the Figure 37.

Figure 37: Structure of the JSON input data file to run the impulse model via a command line.

The ”mesh verification” parameter is used to skip, if equal to “false”, the step opening the mesh in Salome and go directly to the next step.

6.7. Add a new mesh

6.7.1. Using the interface

The GUI allow the user to add a new mesh and/or a new material.

The Figure 31 shows the window to add a new mesh. To add a new mesh there are three steps:

1. Name your mesh
2. Select the file on your computer
3. Click on Add mesh

Figure 38: Add mesh window.

There are few things that need to be present on the mesh. To know the specificities of the pad mesh, please refer to the dedicated section.

6.7.2. Manually

To add a mesh manually, one need to create a folder with the mesh name inside the “meshes” folder, in the good part type. Then simply move the mesh file inside this new folder, the name of the mesh should be pad.med, rail.med, sleeper.med or ballast.med.

For the exact format of this file and the mesh, refer to the dedicated section.

After re-starting the GUI, the new mesh should be present in the list.

6.8. Add a new material

6.8.1. Using the interface

The Figure 32 shows the window for adding a new pad material.

To add a new material six steps are needed:

1. Give a name to your material
2. Select the file containing the Young modulus
3. Select the file containing the Bulk modulus
4. Select the file containing the Shear modulus
5. Select the file containing the time scale

6. Enter the Poisson's Ratio ($0 < a \leq 0.495$)
7. Enter the density in [t/mm^3]
8. Click on Add material

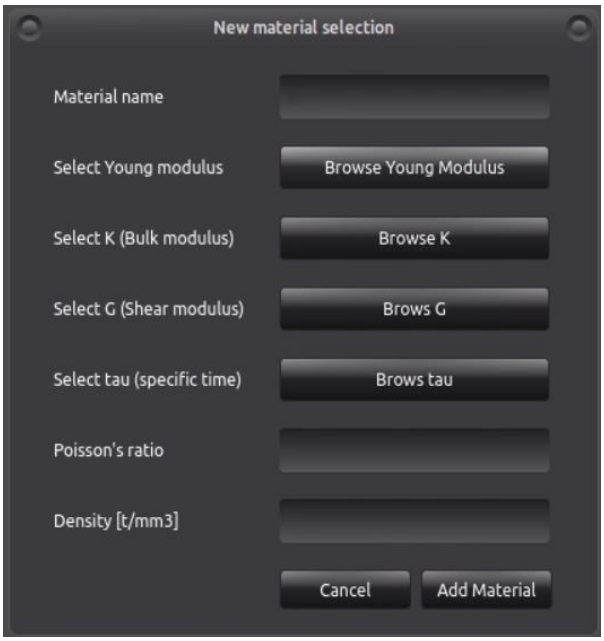


Figure 39: Add a new material window for pads.

For the other materials the steps are easier, see Figure 40. The user need to enter the following information:

1) Name of the material.
2) The young modulus.
3) The hysteretic damping coefficient.
4) The Poisson's ratio ($0 < a \leq 0.495$).
5) The density.

Then click on Add material and it's done.

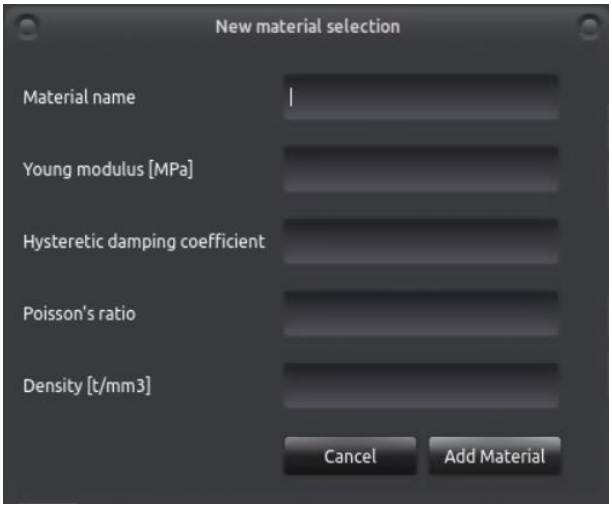


Figure 40: Add a new material window for rails, sleepers and ballasts.

6.8.2. Manually

To add a material manually, one need to create a folder with the material name inside the “materials” folder, under the good part type. Then simply move the material files inside this new folder. The files needed for a pad material are:

1. E.csv
2. K.csv
3. G.csv
4. tau.csv
5. poisson.csv
6. density.csv

And for the others parts type:

1. sleeper_AmHyst.csv
2. sleeper_E.csv
3. sleeper_Nu.csv
4. sleeper_Rho.csv

“sleeper” has to be replaced by the good part type.

For the exact format of those files, refer to the dedicated section.

After re-starting the GUI, the new material should be present in the list.

6.9. Access the results

Once the calculation are finished, the results file in med format are copied in the folder selected by the user (step 1 and 2 in the main window).

The file copied are the followings:

1. The .med file that can be open with salome-meca to check the results.
2. The Stresses calculated for each time steps.
3. The message file from the calculation.
4. A file containing a brief summary of the parameters of the study.

7. Material properties identification scripts

A python/Code_Aster script have been added to the tool box. This script aim to automatize the identification of the material properties of the sleeper model.

It requires as input the mesh of the sleeper, an initial guess for the material properties and the experimental Eigen frequencies, see Figure 41 and Figure 42. For the mesh orientation and other parameters, one can see the requirement of the impulse model.

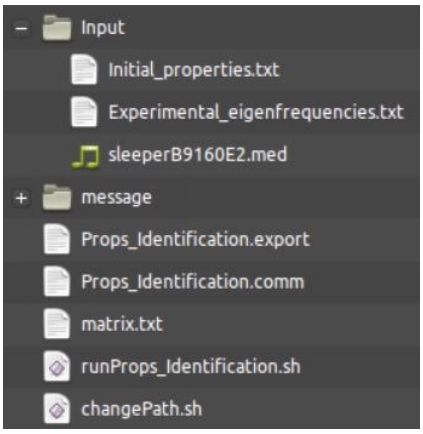


Figure 41: Files architecture for the properties identification folder.

a

```
1 E_L;90000.0
2 E_N;90000.0
3 E_T;90000.0
4 G_LN;37000.0
5 G_LT;37000.0
6 G_TN;37000.0
7 NU_LN;0.2
8 NU_LT;0.2
9 NU_TN;0.2
10 RHO;2.4356e-09
```

b

```
1 mode1;113
2 mode2;125
3 mode3;344
4 mode4;366
5 mode5;394
6 mode6;663
7 mode7;718
8 mode8;766
9 mode9;1040
10 mode10;1065
11 mode11;1125
12 mode12;1472
```

Figure 42: Example of input files content: a) Initial_properties.txt, b) Experimental_eigenfrequencie.txt.

There is no GUI for this script and the user need to run it with a terminal. To run the script, one have to open a terminal in the folder containing the “Props_Identification.comm” file. Then the following command can be used:

```
cae@mycomputer:~/Documents$ ./runProps_Identification.sh
```

Once the script end, there will be many files in the result folder. The first file to check is “results_summary.txt”.

This file contain all the main information about the identification process:

- Number of iterations,
- Experimental versus numerical Eigen frequencies,
- Time of calculation,
- Evolution of the properties,
- Evolution of the error.

If the identification hasn’t converge after the max iteration allowed, currently hard coded to 50, you can plot the error and see how far from the convergence the computation is or if it has diverge for any reason.

One can then re-run the script using a set of initial properties closer to the converged ones by looking the set of properties corresponding to the selected iteration number.

The script start with a sensitivity analysis, then use this analysis and the experimental Eigen frequencies to modify the properties and have a better match. If the match is not good enough a new iteration is done, and so on until the convergence.

8. Rail track semi-analytical model

Documentation to be written…