NMPB08 – Railway Emission Model

Programmers Guide

Van Maercke Dirk
CSTB Grenoble
24/10/2011

Table of Contents

1. Getting started

1.1 Using a Railway Traffic element

The **RailwayEmissionNMPB08** library uses an internal structure to store railway traffic data, as defined by the user, and to construct an emission model defined in terms of equivalent source positions and sound powers .

In order to use the library, your application should include the appropriate header file:

```
#include "RailwayEmissionNMPB08.h"
```

The library uses a single database of trains, units and associated sources. In order to use the functionalities of the library, your application must fill the database from external storage:

```
bool ok = NMPB08_LoadRailwayDatabase ("../data/trains_sncf.xml") ;
```

In order to calculate the sound power emission associated with railway traffic, the application creates an internal data structure and fills in the appropriate traffic over a given period of time.

```
// prepare for calculation of equivalent emission levels for 12 hours period
void* tr = NMPB08_CreateRailwayTraffic (12.0) ;

// add 20 TGV-R trains at 300 km/h
NMPB08_AddRailwayTraffic (tr, "TGV-R", 20, 300.) ;

// add 10 TGV-A trains at 270 km/h
NMPB08_AddRailwayTraffic (tr, "TGV-A", 10, 270.) ;
```

In order to get the equivalent source model associated with the railway traffic, your application calls the *NMPB08_GetRailwayEmission* function. This function returns a pointer to the associated emission data.

```
// get the equivalent NMPB emission model
RailwayEmission const* emi = NMPB08_GetRailwayEmission (tr) ;
```

The RailwayEmission structure is maintained by the library ; your application should not delete or free the pointer returned by the *NMPB08_GetRailwayEmission* function.

When done with the railway traffic data, you application should clean up the internal data structures and free all allocated memory :

```
// cleanup
NMPB08_DeleteRailwayTraffic (tr) ;
```

The database, once loaded into memory, remains available for further use with other railway traffic data.

1.2 The equivalent source model

The equivalent source model is defined as :

```
struct RailwayEmission
{
       unsigned int  nbFreq ;
       double*       freq ;
       double        sin_h ;
       double        sin_v ;
       unsigned int  nbSources ;
       struct RailwayEquivalentSource
       {
          double      height ;
          bool        has_Lw ;
          double      Lw_dir[nbFreq] ;
       } source[nbSources] ;
} ;
```

Where:

nbFreq	the frequency range of the emission model. In the NMPB standard this is fixed to 18 third-octave bands from 100 to 5000 Hz.
freq	the centre frequencies of the 18 third-octave bands.
sin_h	the horizontal component of the direction of propagation (see figure 6 of NF S31-133).
sin_v	the vertical component of the direction of propagation (see figure 7 of NF S 31-133).
nbSources	number of equivalent source positions ; each position correspond to a specific height above the head of the track. The NMPB standard defines three allowable source heights : 0 cm, 50 cm and 4m above the head of the track.
source	data for each equivalent source position.
height	height of the equivalent source position.
has_Lw	true if there is a sound power associated with this source height.
Lw_dir	the sound power of the equivalent source as a function frequency.

Applications should apply the appropriate attenuations to the different sources and sum the resulting noise levels at the receiver.

1.3 Enumerating available trains and units

Railway traffic is defined in terms of trains and units. Each train and unit in the database has a unique identifier, i.e. a short text. Applications can enumerate the list of available trains

and units, e.g. to present these in the graphical interface. The library provides the following function :

```
NMPB08_EnumRailwayDatabase (EnumRailwayEntities enumProc,
                            unsigned int include_types = ENTITY_TRAFFIC,
                            void* userdata = 0) ;
```

Your application must declare a callback function that will be called for each entry in the database. This user-defined function should match the following type definition :

```
typedef bool (*EnumRailwayEntities)(RailwayEntity const& info, void* userdata);
```

Note that you can use the *userdata* argument to pass any data to the callback function, e.g. in order to create your own list of identifiers in application memory, to fill in a combo-box control, to set up a tree-view control,...

The base entity of the source database is the “unit of rolling stock”. In principle, railway traffic is defined in terms of units. As an enhanced feature, the database can also contain “train” records, where trains are defined in terms of a fixed composition of units (see annexe 1 for details). The second argument to the enumeration call allows for selecting specific records from the database:

```
ENTITY_UNIT         units only (this is the standard use, especially in the
                    case of strategic noise mapping)
ENTITY_TRAIN        trains only
ENTITY_PARTIAL      parts of units that occur as elements in train composition
                    but do not correspond to individual units of rolling stock
                    (extended use and special applications)
ENTITY_TRAFFIC      rolling stock (trains and units)
ENTITY_ALL          all records (trains, units and parts of units)
```

2. Advanced use

2.1 Directivity and apparent sound power

The NMPB standard defines horizontal and vertical directivity patters for railway sources. Directivity is built into the software library, i.e. is considered as part of the emission model. Therefore, the emission model does not return true sound power but rather “apparent” sound power radiated in a given direction.

The direction of radiation from the source to the receiver is set by calling:

```
int NMPB08_SetRailwayEmissionAngles (void* id, double sin_h, double sin_v) ;
```

This call should be made before accessing the emission values. Note that the calculation of apparent sound power in different directions can be called in a loop without changing the traffic state.

The horizontal and vertical angles are measured relative to the principal direction which is horizontal and perpendicular to the track. The angles are given as sinus values in order to avoid unnecessary conversions from coordinates to angles and vice-versa. A general method for calculating angle values is given in annexe 2 to this document. Note that sin_h = sin_v = 0 correspond to the principal direction.

2.2 Rail corrections

The NMPB method by itself does not define sound power calculations or corrections for rail types and mounting conditions. However, guidelines for producing strategic noise maps published by Sétra define a set of generic values that can be used together with the train database defined by RFF/SNCF.

The correction for rails can be set by calling :

```
unsigned int NMPB08_SetRailCorrection (void* id, NMPB_RailCorrectionType corr) ;
```

Where the NMPB_RailCorrectionType corresponds to a dB(A) weighted global correction value applied to the rolling noise source, i.e. on the equivalent sources at 0 and 50 cm above the head of the track. The correction is the same in all frequency bands.

Typical values, taken from the Sétra guidelines are given below:

Type de voie	Terme correctif
LRS sur traverses béton	référence
LRS sur traverses bois, mixtes ou métalliques	+ 3
Rails courts sur traverses béton	+ 3
Rails courts sur traverses bois, mixtes ou métalliques	+ 6

Type de voie	Terme correctif
LRS sur traverses béton	référence
Zone d'appareils de voie	+ 6
Ouvrages d'art métalliques	+ 5 à + 10

The following corrections are included as part of the library :

```
const NMPB_RailCorrectionType  RAIL_LONG_TRAVERSES_BETON  = 0.0 ;
const NMPB_RailCorrectionType  RAIL_LONG_TRAVERSES_AUTRE  = 3.0 ;
const NMPB_RailCorrectionType  RAIL_COURT_TRAVERSES_BETON = 3.0 ;
const NMPB_RailCorrectionType  RAIL_COURT_TRAVERSES_AUTRE = 6.0 ;
const NMPB_RailCorrectionType  RAIL_ZONE_APPAREILS_VOIE   = 6.0 ;
```

2.3 Interaction between train and nearby barrier

Multiple reflections between the train car's bodies and a barrier parallel to the track me largely reduce the screening effect of the barrier; see e.g. paragraph 7.4.6 of NF S 31-133. In principle, these effects can be simulated by means of image source and/or ray tracing techniques. In order to simplify implementation and to improve reproducibility, it has been decided to implement this feature as part of the railway source models as an "equivalent source" model; i.e. the library estimates the effect of the reflections between the car body and the barrier and modifies the sound power of the elementary sources accordingly.

In order to use this feature, the *NMPB08_GetRailwayEmission* function accepts, as its second argument, a pointer to a data structure describing the (simplified) geometry of the actual situation, including the source, the barrier and the receiver.

```
struct ScreenBodyInteraction
{
	int     maxInteractions ;
	double  dS ;
	double  hS ;
	double  dR ;
	double  hR ;
	double* alpha ;
} ;
```

Where :

- maxInteractions : the maximum number of interactions to be taken into account ; one interaction accounts for a double reflection, once on the inner side of the barrier, once on the car's body
- dS : the horizontal distance from the source to the barrier
- hS : the height of the barrier relative to the head of the track
- dR : the horizontal distance from the barrier to the receiver
- hR : height of the receiver relative to the head of the track
- alpha : absorption coefficients applicable to the inner side of the barrier (in 1/3 octave bands, from 100 till 5000 Hz)

To obtain the sound powers of the equivalent sources, use :

```
// fill in the nearby geometry record
ScreenBdoyInteraction interaction ;
interaction.maxInteractions = 3 :
interaction.dS = 3.0 ;
interaction.hS = 2.0 ;
interaction.dR = 7.5 ;
interaction.hR = 3.5 :
interaction.alpha = 0 ;
 // get the equivalent NMPB emission model
RailwayEmission const* emi = NMPB08_GetRailwayEmission (tr, &interaction) ;
```

Note that in case the second argument is a NULL pointer or if *interaction.maxInteractions* is equal to zero, no interactions will be accounted for and the original equivalent sources will be returned without modification.

Note that if *interaction.alpha* is a NULL pointer, the inner side of the barrier will be considered as perfectly reflecting. This will maximize the loss of efficiency of the barrier due to multiple reflections.

2.4 Error codes

Most of the functions in the RailwayEmissionNMPB08 library return error codes in case the function failed. The following codes are defined in the header file:

```
ERROR_XML_PARSER     The database file could not be parsed
ERROR_XML_SCHEMA     The XML file does not match the expected schema
ERROR_OPEN_FILE      The XML file does not exist or cannot be opened
ERROR_INVALID_ID     The identifier of a train, unit or source does not appears
                      in the database
ERROR_INVALID_HEIGHT An elementary source was defined at a height not supported
                      by the NMPB08 standard.
```

The last error code may be issued when a database is loaded that was not specifically designed to operate with the NMPB standard (e.g. one that was designed for use with the Harmonoise, Imagine or CNOSSOS-EU models). It occurs when the traffic data contains a train or unit that contains a source at a height that is not supported by the NMPB standard; e.g. an extra source at 2.5m height for traction noise.

2.5 Options

For specific application purposes, use of the following options may be required:

```
OPTION_TRACE_DEBUG    print out intermediate results to the console window
OPTION_NO_DIRECTIVITY ignore directivity in the calculation of sound power.
                      This option is useful for applications that pre-calculate
                      and cache the sound power per source line before starting
```

```
                        ray path detection. Such applications should create their
                        own functions to correct for directivity.
OPTION_MODIFY_HEIGHTS if this option is set, no INVALID_HEIGHT errors will be
                        issued. Instead, the sound power of the extra source will
                        be distributed over the 3 source heights defined by the
                        NMPB standard. This is non-standard behavior and should
                        not be used when strict compliance with existing
                        legislation is required.
```

Options are set or cleared by means of :

```
unsigned int NMPB08_SetRailwayOptions (void* id,
                                       unsigned int option, bool on_off) ;
```

2.6 Elementary source model

Within the database, trains and units are defined in terms elementary sources distributed along the units, at different heights. An application may need to locate each of the individual sources and identify the associated sound powers, e.g. in order to simulate by-pass measurements or to calculate more precisely L_{max}, SEL, TEL or $L_{eq,Tp}$ levels.

The elementary sources associated with a train or a unit can be retrieved by means of :

```
int NMPB08_EnumRailwaySources (EnumRailwaySources enumProc,
                               const char* train_or_unit,
                               void* userdata) ;
```

Your application must declare a callback function that will be called for each entry in the database. This user-defined function should match the following type definition :

```
typedef bool (*EnumRailwaySources) (RailwayElementarySource const& info,
                                    void* userdata) ;
```

Information about elementary sources includes:

```
struct RailwayElementarySource
{
      const char*    id ;         // unique identifier
      double         pos ;        // position relative to the head of the train
      double         height ;     // height relative to the head of the track
      double         vref ;       // reference speed for sound power
      double         slope ;      // speed dependency of the sound power
      int            hdir_model ;// horizontal directivity pattern
      int            vdir_model ;// vertical directivity pattern
      double const* Lw ;          // sound power spectrum at the reference speed
} ;
```

3. Worked out examples

3.1 Analysing the equivalent source model

In this example, we create a railway traffic and display the associated emission model

```
#include "RailwayEmissionNMPB08.h"
#include <stdio.h>

// forward declaration
void showEmission (RailwayEmission const* emi) ;

int _tmain(int argc, _TCHAR* argv[])
{
	// load the database
	const char* xml_file = "../data/example_train.xml" ;
	NMPB08_LoadRailwayDatabase (xml_file) ;

	// create traffic over a 12 hours period
	void* tr = NMPB08_CreateRailwayTraffic (12.0) ;

	// add 30 TGV-D trains at 270 km/h
	NMPB08_AddRailwayTraffic (tr, "TGV-D", 30, 270.) ;

	// get the equivalent NMPB emission model
	RailwayEmission const* emi = NMPB08_GetRailwayEmission (tr) ;

	// display the equivalent NMPB emission model
	showEmission (emi) ;

	// cleanup
	NMPB08_DeleteRailwayTraffic (tr) ;
	return 0;
}

// show emission model
void showEmission (RailwayEmission const* emi)
{
   if (emi)
   {
	printf ("nb sources = %d \n", emi->nbSources) ;
	printf ("height") ;
	for (unsigned int i = 0 ; i < emi->nbSources ; i++)
	{
		printf ("\t%6.2f", emi->source[i].height) ;
	}
	printf ("\n") ;
	printf ("active") ;
	for (unsigned int i = 0 ; i < emi->nbSources ; i++)
	{
		printf ("\t  %s", emi->source[i].has_Lw ? "YES" : "NO") ;
	}
	printf ("\n") ;
	for (unsigned int j = 0 ; j < emi->nbFreq ; j++)
	{
		printf ("%.0fHz", emi->freq[j]) ;
		for (unsigned int i = 0 ; i < emi->nbSources ; i++)
		{
```

```
                printf ("\t %6.1f", emi->source[i].Lw_dir[j]) ;
            }
            printf ("\n") ;
        }
    }
    else
    {
        printf (".error : invalid source height in database\n") ;
    }
}
```

The example code produces the following output :

```
height      0.00    0.50    4.00
active       YES     YES      NO
100Hz      -99.9    78.3   -99.9
125Hz      -99.9    76.8   -99.9
160Hz      -99.9    75.0   -99.9
200Hz      -99.9    74.8   -99.9
250Hz      -99.9    75.9   -99.9
315Hz      -99.9    75.5   -99.9
400Hz      -99.9    75.0   -99.9
500Hz      -99.9    74.3   -99.9
630Hz      -99.9    74.5   -99.9
800Hz       75.5   -99.9   -99.9
1000Hz      75.8   -99.9   -99.9
1250Hz      76.3   -99.9   -99.9
1600Hz      75.8   -99.9   -99.9
2000Hz      75.4   -99.9   -99.9
2500Hz      76.5   -99.9   -99.9
3200Hz      75.5   -99.9   -99.9
4000Hz      72.4   -99.9   -99.9
5000Hz      69.8   -99.9   -99.9
```

Note that in this simplified model the sound power due to rolling noise is split between the two lower source heights depending on frequency and that there is no equivalent source defined at the 4 m high position (in this case, has_Lw is set to false and the sound power levels to -99.9 dB).

3.2 Enumerating the contents of the database

In this example, we examine the contents of the database and list the available trains and units that can be used in the railway traffic model.

```
#include <stdio.h>
#include "RailwayEmissionNMPB08.h"

// callback function for enumerating units and trains
bool enumTrainsAndUnits (RailwayEntity const& info, void* userdata)
{
       printf ("%s [id] = %s \n", (info.type == ENTITY_TRAIN) ?
              "TRAIN" : "UNIT", info.id) ;
       printf (".name   : %s \n", info.name) ;
       printf (".vmax   : %.0f km/h \n", info.vmax) ;
```

```
        printf (".lenght : %.1f m \n", info.length) ;
        return true ;
}

int _tmain(int argc, _TCHAR* argv[])
{
        // load the database
        const char* xml_file = "../data/trains_sncf.xml" ;
        NMPB08_LoadRailwayDatabase (xml_file) ;

        // enumerate all trains and units
        NMPB08_EnumRailwayDatabase (enumTrainsAndUnits, ENTITY_ALL) ;
}
```

The example code produces the following output :

```
UNIT [id] = BB16000
.name   : BB 16000-BB 16100
.vmax   : 160 km/h
.lenght : 16.7 m
UNIT [id] = BB17000
.name   : BB 17000, BB 17100
.vmax   : 140 km/h
.lenght : 14.9 m
UNIT [id] = BB22200
.name   : BB22200-22300-22400
.vmax   : 160 km/h
.lenght : 17.5 m
UNIT [id] = CC72000
.name   : CC 72000-72100
.vmax   : 160 km/h
.lenght : 20.2 m
UNIT [id] = Fretdivers(G19)
.name   : Wagon de fret (divers) freinée fonte
.vmax   : 120 km/h
.lenght : 17.0 m
UNIT [id] = MI2N-Z22500
.name   : Z 22500, Z 22600 pentacaisses (MI2N)
.vmax   : 140 km/h
.lenght : 129.4 m
UNIT [id] = TGV-A
.name   : TGV 300,TGV 400 (TGV-A)
.vmax   : 300 km/h
.lenght : 237.6 m
UNIT [id] = TGV-D
.name   : TGV 200 (TGV Duplex)
.vmax   : 300 km/h
.lenght : 200.1 m
UNIT [id] = TGV-R
.name   : TGV 500,TGV 4500 (TGV-R,TGV Thalys PBA)
.vmax   : 300 km/h
.lenght : 200.1 m
UNIT [id] = TGV-TM
.name   : TGV 3200 (TGV-TMST Eurostar)
.vmax   : 300 km/h
.lenght : 393.7 m
UNIT [id] = V2N
.name   : Voiture V2N
.vmax   : 160 km/h
```

```
.lenght : 26.9 m
UNIT [id] = VB2N
.name   : Voiture VB2N
.vmax   : 120 km/h
.lenght : 24.8 m
UNIT [id] = VU-VTU-FF
.name   : Voiture VU, VTU freinée Fonte (CORAIL)
.vmax   : 200 km/h
.lenght : 26.4 m
UNIT [id] = WTREMIE_F76
.name   : Wagon tr├®mie F76
.vmax   : 120 km/h
.lenght : 13.4 m
UNIT [id] = X72500_Bi
.name   : X 72500 bicaisses (X-TER)
.vmax   : 160 km/h
.lenght : 53.0 m
UNIT [id] = X73500i
.name   : X 73500, 73600, 73700, 73800, 73900 (A-TER)
.vmax   : 140 km/h
.lenght : 28.9 m
UNIT [id] = Z2N-Z2N0500FC
.name   : Z 20500 pentacaisses freinée fonte (Z2N)
.vmax   : 140 km/h
.lenght : 129.4 m
UNIT [id] = Z2N-Z2N0500FF
.name   : Z 20500 pentacaisses freinée fonte (Z2N)
.vmax   : 140 km/h
.lenght : 129.4 m
UNIT [id] = Z5300
.name   : Z 5300, Z5400
.vmax   : 120 km/h
.lenght : 103.6 m
UNIT [id] = Z6400
.name   : Z 6400, Z6500
.vmax   : 120 km/h
.lenght : 92.4 m
UNIT [id] = Z8100
.name   : Z8100, Z8200
.vmax   : 140 km/h
.lenght : 92.4 m
```

3.3 Enumerating elementary sources

Within the database, trains are defined in terms of units and elementary sources distributed along the units, at different heights. An application may need to locate each of the individual sources and identify the associated sound powers, e.g. in order to simulate by-pass measurements or to calculate more precisely L_{max}, SEL, TEL or $L_{eq,Tp}$ levels.

In this example we examine the elementary sources associated with a detailed train defined in terms of three different units. The model uses 4 different elementary sources for rolling noise (rail and wheel), aerodynamic noise and engine noise.

```
#include <stdio.h>
```

```
#include "RailwayEmissionNMPB08.h"

// callback funtion for enumerating equivalent sources
bool enumSources (RailwayElementarySource const& info, void* userdata)
{
	printf (".source type = %-16.16s  d = %5.1f  h = %3.1f \n",
		info.id, info.pos, info.height) ;
	return true ;
}

int _tmain(int argc, _TCHAR* argv[])
{
	// load the database
	const char* xml_file = "../data/example_train.xml" ;
	NMPB08_LoadRailwayDatabase (xml_file) ;

	// list elementary sources for train TGV-D-SIM
	NMPB08_EnumRailwaySources (enumSources, "TGV-D-SIM", 0) ;
}
```

The example code produces the following output:

```
.source type = TGV-D-RAIL        d =   4.0  h = 0.0
.source type = TGV-D-RAIL        d =  17.0  h = 0.0
.source type = TGV-D-WHEEL       d =   4.0  h = 0.5
.source type = TGV-D-WHEEL       d =  17.0  h = 0.5
.source type = TGV-D-ENGINE      d =  10.0  h = 2.5
.source type = TGV-D-RAIL        d =  20.1  h = 0.0
.source type = TGV-D-RAIL        d =  40.1  h = 0.0
.source type = TGV-D-RAIL        d =  60.1  h = 0.0
.source type = TGV-D-RAIL        d =  80.1  h = 0.0
.source type = TGV-D-RAIL        d = 100.1  h = 0.0
.source type = TGV-D-RAIL        d = 120.1  h = 0.0
.source type = TGV-D-RAIL        d = 140.1  h = 0.0
.source type = TGV-D-RAIL        d = 160.1  h = 0.0
.source type = TGV-D-RAIL        d = 180.1  h = 0.0
.source type = TGV-D-WHEEL       d =  20.1  h = 0.5
.source type = TGV-D-WHEEL       d =  40.1  h = 0.5
.source type = TGV-D-WHEEL       d =  60.1  h = 0.5
.source type = TGV-D-WHEEL       d =  80.1  h = 0.5
.source type = TGV-D-WHEEL       d = 100.1  h = 0.5
.source type = TGV-D-WHEEL       d = 120.1  h = 0.5
.source type = TGV-D-WHEEL       d = 140.1  h = 0.5
.source type = TGV-D-WHEEL       d = 160.1  h = 0.5
.source type = TGV-D-WHEEL       d = 180.1  h = 0.5
.source type = TGV-D-RAIL        d = 183.2  h = 0.0
.source type = TGV-D-RAIL        d = 196.2  h = 0.0
.source type = TGV-D-WHEEL       d = 183.2  h = 0.5
.source type = TGV-D-WHEEL       d = 196.2  h = 0.5
.source type = TGV-D-AERO        d = 185.2  h = 4.0
```

Note that this model contains an equivalent source at 2.5 m height and therefore cannot be used with the NMPB standard. When using this train in a traffic record, an error will occur unless the MODIFY_HEIGHTS option is set.

3.4 Multiple reflections between train and nearby barrier

The file *TestRailwayInteraction.xml* contains a worked out example illustrating the effects of multiple reflections between the train and a nearby barrier. Note that this file is to be used with the *TestNMPB.exe* executable, as it relies on both the railway emission model and the propagation library.

In the example, the top of the barrier is located at 3m distance from and 2m above the head of the track. The receiver is located 3.5m above the head of the track, 25m away from the barrier. The maximum number of interactions has been set to 3 (but a simulation up to 10 interaction has also been carried out.

Note that the number of trains per hour has been set to 300000 so that the ratio Q/(1000*V) equals to 1, i.e. the sound power (per meter) of the equivalent line source equals the total power output of a single train. This allows to get results comparable to $L_{A,max}$ values at the receiver position.

Four situations are simulated in this file :

1) no barrier,
2) with the barrier in place, but ignoring all effects of train-barrier interaction,
3) a perfectly hard barrier, taking into account train-barrier interaction,
4) a barrier with an inner lining with an absorption coefficient α=0.5 at all frequencies.

The results of these simulations are summarised below :

	3 interactions	10 interactions
No barrier	96.1	96.1
Barrier, no interaction	76.9	76.9
Hard barrier	88.6	91.5
Soft barrier (α=0.5)	83.5	84.0

As can be seen from this simple test case :

1) ignoring multiple reflections between the train's body and the barrier can lead to an important overestimation of the efficiency of the barrier ;
2) when the car's bodies and the inner side of the barrier are both assimilated with perfectly flat, hard mirrors, the barrier effect is almost completely destroyed due to the multiple reflections, i.e. all sound energy not diffracted over the barrier is

accumulated without loss in between the train and the barrier and after multiple reflections ends up by being diffracted over the barrier ;

As can be seen from this example, the choice of taking into account or ignoring multiple reflections, can lead to differences up to 15 dB(A) in the estimated noise levels. As a consequence, it must be emphasised that train-barrier interaction should be used with care and preferably be fine-tuned by means of experimental data or numerical simulations.

As a (temporary) workaround we recommend:

1) To fix the number of interactions to 3 ;
2) To introduce realistic absorption coefficients for all barriers near trains and to avoid perfectly hard barriers (i.e. with $\alpha = 0$, which is the default if no absorption is specified) ; all real materials will absorb sound to some extent, especially in the high frequency range (i.e. due to roughness).
3) To avoid representing trains by means of perfectly hard, perfectly flat mirrors; e.g. by fixing an equivalent reflection coefficient R < 0.9 for all units.

Note: when using the TRACE_DEBUG option, calls to *NMPB08_GetRailwayEmission* will print out tables representing the relative weight of the real sources and their images, as a function of reflection order, source height and frequency.

E.g. for the perfectly hard barrier and a source at 0cm above the head of the track :

```
ScreenBodyInteraction correction table
source height : 0.00m
Freq     N=0     N=1     N=2     N=3     N=4     N=5     N=6     N=7     N=8
100     -12.1   -11.5   -14.7   -19.1   -23.9   -28.8   -34.4   -40.2   -46.1
125     -12.6   -11.3   -14.5   -18.8   -23.2   -27.9   -33.4   -39.1   -44.8
160     -13.2   -11.1   -14.4   -18.3   -22.4   -26.8   -32.1   -37.6   -43.3
200     -13.8   -11.5   -14.4   -17.8   -21.6   -25.7   -30.8   -36.2   -41.7
250     -14.4   -11.9   -14.3   -17.3   -20.7   -24.4   -29.4   -34.6   -40.0
315     -15.2   -12.5   -14.2   -16.7   -19.7   -23.1   -27.9   -33.1   -38.4
400     -16.2   -13.1   -14.1   -16.0   -18.8   -22.2   -26.8   -31.7   -36.9
500     -17.1   -13.7   -14.0   -15.4   -18.3   -21.2   -25.6   -30.3   -35.2
630     -18.1   -14.2   -13.8   -15.3   -17.6   -20.1   -24.1   -28.6   -33.6
800     -19.2   -14.8   -13.7   -15.2   -16.9   -18.9   -22.9   -27.4   -32.2
1000    -20.1   -15.4   -13.7   -15.0   -16.2   -18.2   -22.0   -26.2   -30.7
1250    -21.1   -16.0   -14.2   -14.9   -15.6   -17.5   -20.9   -24.8   -29.2
1600    -22.2   -16.9   -14.9   -14.7   -15.4   -16.6   -19.5   -23.6   -27.9
2000    -23.1   -17.7   -15.4   -14.5   -15.1   -15.7   -18.8   -22.6   -26.7
2500    -24.1   -18.7   -15.9   -14.3   -14.9   -15.2   -18.1   -21.5   -25.3
3200    -25.2   -19.8   -16.6   -14.7   -14.6   -14.8   -17.1   -20.1   -24.2
4000    -26.2   -20.7   -17.2   -15.2   -14.2   -14.4   -16.2   -19.3   -23.2
5000    -27.1   -21.7   -18.0   -15.8   -13.9   -13.9   -15.3   -18.6   -22.1
```

From these tables it is easily seen that in the high frequency range, the equivalent power level is dominated by higher order images that are much less subject to diffraction than the real source. At lower frequencies, these effects are rapidly compensated by retro-diffraction (i.e. the finite height of the barrier as a reflector, compared to the wavelength) and the model converges rapidly as the order of reflections is increased.

ANNEXE 1 : Example of train definition file

This is the contents of the “example_train.xml” file used in the worked out examples.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<train_data model="NMPB08">
	<!--  EXEMPLE 1   -->
	<!--  this is minimal information required for noise mapping  -->
	<!--  note that spectra can be defined in multiple ways...  -->
	<!--  elementary source at 0cm  -->
	<source id="TGV-D-0cm" type="rolling">
		<height>0.00</height>
		<vref>300.0</vref>
		<slope>30.0</slope>
		<Lw fmin="100" fmax="5000" bw="1/3oct">
			-99.9 -99.9 -99.9 -99.9 -99.9 -99.9 -99.9 -99.9 -99.9
			115.0 115.3 115.8 115.3 114.9 116.0 115.0 111.9 109.3
		</Lw>
		<hdir model="nmpb" />
		<vdir model="nmpb" />
	</source>
	<!--  elementary source at 50cm   -->
	<source id="TGV-D-50cm" type="rolling">
		<height>0.50</height>
		<vref>300.0</vref>
		<slope>30.0</slope>
		<Lw fmin="100" fmax="5000" bw="1/3oct">
			<value freq="100">117.8</value>
			<value freq="125">116.3</value>
			<value freq="160">114.5</value>
			<value freq="200">114.3</value>
			<value freq="250">115.4</value>
			<value freq="315">115.0</value>
			<value freq="400">114.5</value>
			<value freq="500">113.8</value>
			<value freq="630">114.0</value>
			<value freq="800">-99.9</value>
			<value freq="1000">-99.9</value>
			<value freq="1250">-99.9</value>
			<value freq="1600">-99.9</value>
			<value freq="2000">-99.9</value>
			<value freq="2500">-99.9</value>
			<value freq="3150">-99.9</value>
			<value freq="4000">-99.9</value>
			<value freq="5000">-99.9</value>
		</Lw>
		<hdir model="nmpb" />
		<vdir model="nmpb" />
	</source>
	<!--  TGV-D defined as a single traffic unit   -->
	<!--  note : cref is used as an equivalent reflection coefficient  -->
	<!--  in the calculation of train/barrier interaction -->
	<unit id="TGV-D">
```

```
        <name>TGV 200 (TGV Duplex)</name>
        <image file="tgv-duplex.jpg" />
        <length>200.1</length>
        <vmax>300.0</vmax>
        <cref>0.8</cref>
        <sources>
                <source ref="TGV-D-0cm" nb="13" pos="4.0" spacing="16" />
                <source ref="TGV-D-50cm" nb="13" pos="4.0" spacing="16" />
        </sources>
</unit>
<!--  EXAMPLE 2 -->
<!-- extended version, multiples units and multiple source heights -->
<!-- explicitly defined  extra features compared to example 1  -->
<!--   1) more source heights -->
<!--   2) specific source powers for locos and cars  -->
<!--   3) exact position of all sources along the train -->
<!-- Note that, in the NMPB model there is no source at 2.5m  -->
<!-- therefore  this train will not be available for standard -->
<!-- NMPB calculations !! -->
<!--  elementary source = radiation from rail  -->
<source id="TGV-D-RAIL" type="rolling">
        <height>0.00</height>
        <vref>300.0</vref>
        <slope>30.0</slope>
        <Lw fmin="100" fmax="5000" bw="1/3oct">
                97.8 96.3 94.5 94.3 95.4 95.0 94.5 93.8 94.0 115.0 115.3
                115.8 115.3 114.9 116.0 115.0 111.9 109.3
        </Lw>
        <hdir model="nmpb" />
        <vdir model="nmpb" />
</source>
<!-- elementary source = radiation from wheel -->
<source id="TGV-D-WHEEL" type="rolling">
        <height>0.50</height>
        <vref>300.0</vref>
        <slope>30.0</slope>
        <Lw fmin="100" fmax="5000" bw="1/3oct">
                117.8 116.3 114.5 114.3 115.4 115.0 114.5 113.8 114.0 95.0
                95.3 95.8 95.3 94.9 96.0 95.0 91.9 89.3
        </Lw>
        <hdir model="nmpb" />
        <vdir model="nmpb" />
</source>
<!--  traction noise at 2.5m  -->
<source id="TGV-D-ENGINE" type="traction">
        <height>2.50</height>
        <vref>300.0</vref>
        <slope>20.0</slope>
        <Lw fmin="100" fmax="5000" bw="1/3oct">
                -99.9 -99.9 -99.9 -99.9 -99.9 -99.9 -99.9 -99.9 -99.9 -99.9 -
                99.9 -99.9 -99.9 -99.9 -99.9 -99.9 -99.9 -99.9</Lw>
        <hdir model="nmpb" />
        <vdir model="omni" />
</source>
<!-- aerodynamic noise at 4m  -->
```

```
        <source id="TGV-D-AERO" type="aerodynamic">
                <height>4.00</height>
                <vref>300.0</vref>
                <slope>45.0</slope>
                <Lw fmin="100" fmax="5000" bw="1/3oct">
                        -99.9 -99.9 -99.9 -99.9 -99.9 -99.9 -99.9 -99.9 -99.9 -99.9 -
                        99.9 -99.9 -99.9 -99.9 -99.9 -99.9 -99.9 -99.9</Lw>
                <hdir model="omni" />
                <vdir model="hemi" />
        </source>
        <!--  partial units (not intended to be used as traffic units -->
        <unit id="TGV-D-HEAD" part="true">
                <name>TGV 200 - Motrice de tête</name>
                <length>20.0</length>
                <vmax>300.0</vmax>
                <cref>0.8</cref>
                <sources>
                        <source ref="TGV-D-RAIL" nb="2" pos="4.0" spacing="13" />
                        <source ref="TGV-D-WHEEL" nb="2" pos="4.0" spacing="13" />
                        <source ref="TGV-D-ENGINE" nb="1" pos="10.0" />
                </sources>
        </unit>
        <unit id="TGV-D-CARS" part="true">
                <name>TGV 200 - 8 voitures duplex</name>
                <length>160.2</length>
                <vmax>300.0</vmax>
                <cref>0.8</cref>
                <sources>
                        <source ref="TGV-D-RAIL" nb="9" pos="0.1" spacing="20.0" />
                        <source ref="TGV-D-WHEEL" nb="9" pos="0.1" spacing="20"/>
                </sources>
        </unit>
        <unit id="TGV-D-TAIL" part="true">
                <name>TGV 200 - Motrice de queue</name>
                <length>20.0</length>
                <vmax>300.0</vmax>
                <cref>0.8</cref>
                <sources>
                        <source ref="TGV-D-RAIL" nb="2" pos="3.0" spacing="13.0" />
                        <source ref="TGV-D-WHEEL" nb="2" pos="3.0" spacing="13" />
                        <source ref="TGV-D-AERO" nb="1" pos="5.0" />
                </sources>
        </unit>
        <!--  TGV-D-SIM is defined as a train composed of partial units  -->
        <train id="TGV-D-SIM">
                <name>TGV 200 (TGV Duplex)</name>
                <image file="tgv-duplex.jpg" />
                <units>
                        <unit ref="TGV-D-HEAD" nb="1" />
                        <unit ref="TGV-D-CARS" nb="1" />
                         <unit ref="TGV-D-TAIL" nb="1" />
                </units>
        </train>
</train_data>
```

ANNEXE 2 : Using local coordinate systems with directivity

Directivity of trains (and other sources) is defined in a local coordinate system attached to the train or unit. Assuming a flat track, the coordinate system associated with trains is constructed as follows:

- the X axis is aligned with the tracks,
- the Y axis is horizontal and perpendicular to the tracks,
- the Z axis is the usual upward oriented vertical line.

Directivity is usually expressed by angles:

- the horizontal angle is measured in the X-Y plane with zero on the Y axis,
- the vertical angle is measured with respect to the X-Y plane, positive towards the upwards oriented Z axis.

Note that both the horizontal and vertical angles equal zero along the Y axis.

When a railway line source is defined in the general coordinate system of the project, it takes some calculation to determine the correct angles of emission to be used in the directivity model, especially if the source line is not perfectly horizontal[1].

In this annex, a general method is proposed for converting from general coordinates to local angles to be fed into the sound power calculations as defined in the NMPB railway model.

We first define a 3-dimensional vector that can be used to represent coordinates in any coordinate system.

```
struct Vector
{
	double x ;
	double y ;
	double z ;
	Vector (double _x = 0, double _y = 0, double _z = 0)
	: x(_x), y(_y), z(_z) { }
} ;
```

Next, we define addition of vectors and multiplication of vectors with real numbers :

```
inline Vector operator+ (Vector const& v1, Vector const& v2)
{
	return Vector (v1.x + v2.x, v1.y + v2.y, v1.z + v2.z) ;
}

inline Vector operator- (Vector const& v1, Vector const& v2)
{
```

[1] In this case, there is no such thing as a « horizontal plane containing the source line », as assumed in the text of NF S 31-133 :2011, paragraphe 7.4.4.1.

```
        return Vector (v1.x - v2.x, v1.y - v2.y, v1.z - v2.z) ;
}

inline Vector operator* (Vector const& v1, double k)
{
        return Vector (k * v1.x, k * v1.y, k * v1.z) ;
}

inline Vector operator/ (Vector const& v1, double k)
{
        return Vector (v1.x/k, v1.y/k, v1.z/k) ;
}
```

As a metric space, the vector space has norm, dot product and cross product defined :

```
inline double dot (Vector const& v1, Vector const& v2)
{
        return v1.x * v2.x + v1.y * v2.y + v1.z * v2.z ;
}

inline double norm (Vector const& v)
{
        return sqrt (dot(v,v)) ;
}

inline Vector cross (Vector const& v1, Vector const& v2)
{
        Vector v3 ;
        v3.x = v1.y * v2.z - v1.z * v2.y ;
        v3.y = v1.z * v2.x - v1.x * v2.z ;
        v3.z = v1.x * v2.y - v1.y * v2.x ;
        return v3 ;
}

inline void normalize (Vector &v)
{
        double n = norm(v) ;
        if (n > 0)
        {
                v.x /= n ;
                v.y /= n ;
                v.z /= n ;
        }
}
```

We now have all the necessary machinery for coordinate transformations and calculation of angles. We assume that the actual source line is represented by two positions with coordinates p1 and p2. The directivity angles are calculated for an equivalent point source at the centre of the source segment.

```
void GetDirectivityAngles (Vector const& p1, Vector const& p2,
                           Vector const& rec,
                           double& sin_h, double& sin_v)
{
        // upwards oriented vector
        Vector z (0.0, 0.0, 1.0) ;
        // u-axis aligned with the segment
        Vector u = p2 - p1 ;
        normalize (u) ;
```

```
        // v-axis in the horizontal plane, perpendicular to the u-axis
        Vector v = cross (z, u) ;
        normalize (v) ;
        // w-axis upwards, perpendicular to both u and v
        Vector w = cross (u, v) ;
        normalize (w) ;
        // source position in the middle of the segment
        Vector source = (p1 + p2)/2. ;
        // unit vector from the source in the direction of propagation
        Vector propa = rec - source ;
        normalize (propa) ;
        // determine angles in the local coordinate system
        // note that the v-axis corresponds to (sin_h = sin_v = 0)
        sin_v = dot (propa, w) ;
        sin_h = dot (propa, u) / sqrt (1 – sin_v * sin_v) ;
}
```

Note that the procedure will fail if sin_v = 1 or sin_v = -1. This is coherent with the fact that Euler angles are not uniquely defined in the direction of the Z-axis ; i.e. when the receiver is exactly above the source[2].

[2] It might therefore be more appropriate to set sin_h = dot (propa, u), i.e. to using the angle between the direction of propagation and the Y-Z plane for the calculation of the horizontal directivity. This would also solve the problems encountered with receivers exactly in line with the tracks (e.g. above tunnel openings) ; because the horizontal directivity is zero in the X-Z plane, no matter the elevation angle, no noise levels can be calculated at these positions.

ANNEXE 3: Equivalent source model for train barrier interaction

In the RailwayEmissionNMPB library, multiple reflections between the train and a nearby barrier is taken into account by calculating the sound power of a single equivalent source. In this calculation we assume that ground effects can be ignored[3].

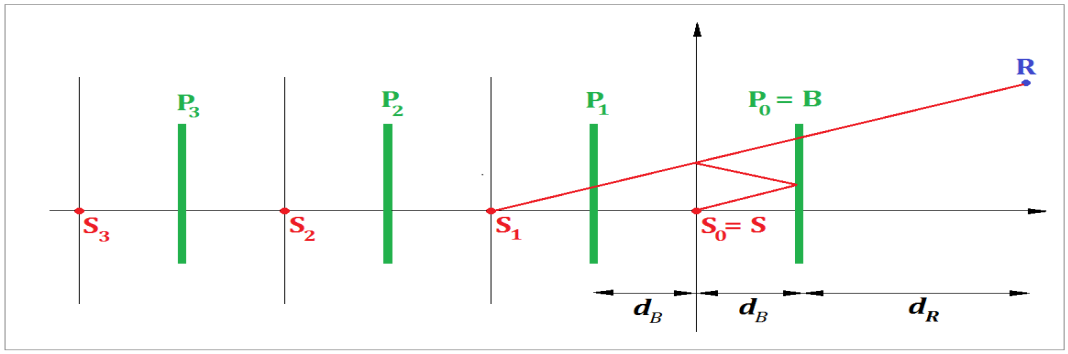
For the purpose of the derivation of the method we assume:

- the origin of the coordinate system is positioned at the head of the track
- a real source, positioned at S (d_S=0, h_S), where h_S is the height of the source relative to the head of the track
- the plane h=0 represents the car's body
- a vertical barrier with top at B (d_B, h_B)
- a receiver located at a distance d_R > 0 behind the barrier : R (d_B+d_R, h_R)

The acoustical properties associated with the train and the inner side of the barrier are:

- the absorption coefficients α(f) per 1/3 octave bands applicable to the inner side of the barrier ;
- an equivalent reflection coefficient C_{ref} applicable to the car's body. Note that C_{ref} is a parameter of the units of rolling stock and defined as such in the train database.

In this configuration, multiple reflections between the car's body and the barrier can be assimilated with image sources positioned at S_n (d_n =-2n.d_B, h_n = h_S), n = 0,1,2,... N ; as shown in the picture below.



The sound power of the equivalent source is expressed as :

$$L_{W,eq} = 10.\log_{10}\left(\sum_{n=0}^{N} 10^{L_{W,n}/10}\right)$$

Where the sound power of the partial sources is given by:

[3] Ground effects will be taken into account when coupling the equivalent source model to the propagation module. For the equivalent source model, individual ground effects can be ignored if we assume a fixed ground effect of either 0dB (soft ground, G=1) or 3dB (hard ground, G=0) for all image sources. Because the correction term for the equivalent source represents a relative level, the effects of ground will cancel out in the result.

$$L_{W,n} = L_W + \Delta L_n$$
$$\Delta L_n = \Delta L_{geo,n} + \Delta L_{dif,n} + \Delta L_{abs,n} + \Delta L_{ref,n} + \Delta L_{retrodif,n}$$

With:

L_W	the sound power of the real source;
$\Delta L_{geo,n}$	a correction term for spherical divergence
$\Delta L_{dif,n}$	a correction term for diffraction by the barrier top
$\Delta L_{abs,n}$	a correction term for absorption at the inner side of the barrier
$\Delta L_{ref,n}$	a correction term for reflection on the car's body
$\Delta L_{retrodif,n}$	a correction term for the finite height of the barrier as a reflector

The correction for spherical divergence is given by:

$$\Delta L_{geo,n} = 20 \log_{10} \frac{r_0}{r_n}$$
$$r_i = |S_i R| = \sqrt{(d_i - (d_B + d_R))^2 + (h_i - h_R)^2}$$

The correction for diffraction by the screen top is given by :

$$\Delta L_{dif,n} = D_0 - D_n$$

Where D_i is the attenuation due to diffraction calculated by means of formula (31) of NF S 31-133, assuming C_h=1, C"=1, for the path linking source S_n to the receiver R, taking into account diffraction at the top of the barrier B :

$$\delta_n = \pm\left(|S_n B| + |BR| - |S_n R|\right)$$

The correction for absorption on the inner side of the barrier is given by[4]:

$$\Delta L_{abs,n} = 10.n.\log_{10}(1-\alpha)$$

The correction for reflections on the car's body is given by:

$$\Delta L_{ref,n} = 10.n.\log_{10} C_{ref}$$

The correction for the finite height of the reflecting barrier is taken into account by means of retro-diffraction as provided in paragraph 9.5.2 of NF S 31-133. The ray path corresponding to an image of order n > 0 will be reflected n times by the barrier. In the cross section, these reflections take place at distances $d_i = -(2i-1).d_B$, i =1,2,...n. We note P_i (d = d_i, h=h_B), i = 1...n as the tops of these reflecting surfaces. At each of these points, a correction term is calculated as :

[4] Note that we assume that the formal identities $\log_{10}(0) = -\infty$, $10^{-\infty/10} = 0$ can be handled in an appropriate way in software.

$$\Delta L_{retrodif,n} = \begin{cases} = \sum_{i=1}^{n} \Delta L_{retrodif,n,i} & if\ n > 0 \\ = 0 & if\ n = 0 \end{cases}$$

Where $\Delta L_{retrodif,n,i}$ is calculated for a source at postion S_n, a barrier top at P_i and a receiver at position R'. The position of the equivalent receiver R' is given by R' = R if the receiver is above the line of sight from S_n to B ; otherwise the equivalent receiver positions is taken on the line of sight vertically above the real receiver ; i.e.

$$d_{R'} = d_R$$

$$h_{R'} = \max\left(h_R, h_B \frac{d_B + d_R - d_n}{d_B - d_n}\right)$$