How to shape noise spectra for continuous system simulation

Andreas Klöckner; Andreas Knoblach; Andreas Heckmann

Noise for continuous-time system simulation is relevant for many applications, whenever time domain results are required. Simulating such noise raises the need to consistently shape the frequency content of the signal. However, the methods for this task are not obvious and form filters are often used as approximate state space implementations. In this article, we address the problem with a new method which relies on directly using the specified power spectral density for a convolution filter. For the example of railway track irregularities, we explain how to derive the required filters, implement them in the open-source AdvancedNoise library, and verify the results. The new method produces correct results, is very simple to use, and enables new features for time simulation of physical systems.

MATHEMATICAL AND COMPUTER MODELLING OF DYNAMICAL SYSTEMS, 2017
VOL. 23, NO. 3, 284–300
http://dx.doi.org/10.1080/13873954.2017.1298622

How to shape noise spectra for continuous system simulation

Andreas Klöckner, Andreas Knoblach and Andreas Heckmann

Institute of System Dynamics and Control, DLR German Aerospace Center, Oberpfaffenhofen, Germany

ABSTRACT
Noise for continuous-time system simulation is relevant for many applications, whenever time domain results are required. Simulating such noise raises the need to consistently shape the frequency content of the signal. However, the methods for this task are not obvious and form filters are often used as approximate state space implementations. In this article, we address the problem with a new method which relies on directly using the specified power spectral density for a convolution filter. For the example of railway track irregularities, we explain how to derive the required filters, implement them in the open-source AdvancedNoise library, and verify the results. The new method produces correct results, is very simple to use, and enables new features for time simulation of physical systems.

ARTICLE HISTORY
Received 8 January 2016
Accepted 20 February 2017

KEYWORDS
Noise; power spectral density; track irregularity

1. Introduction

Modelling stochastic signals is of interest in a wide range of applications, such as sensor modelling, aerodynamic turbulence, and rail irregularities. Previous Modelica libraries, like the Statistics library [2], allow to precisely define statistical properties of such signals. However, other properties of the noise signals, for example, the underlying random number generator or the signal's frequency content could not be modelled as conveniently. A Modelica Noise library has thus recently been released in order to enable the engineer to conveniently and consistently define noise signals [3]. A subset of sampled noise generators and standard distributions was integrated in the Modelica standard library 3.2.2. The remaining functionality is available in the AdvancedNoise library.[1]

In this article, we extend the capabilities of the libraries with a general method to shape the frequency content of the noise signals. We do so using the popular example of railway track irregularities. Due to the importance of track irregularities, there is a vast amount of literature on the subject. See in particular the overview paper by Haigermoser et al. [4] and its extensive reference list of 165 publications. The irregularities may be included in a time domain simulation by replaying measured data (see [5]). The alternative way is to describe the irregularities statistically using a power spectral density (PSD). There are a few overview papers on realizing a stochastic signal representing a given PSD, for both a more general case [6] and for the specific track irregularity application [7].

One class of realizations uses orthogonal basis functions [7,8]. A natural choice for this approach is to use the cosine function with different frequencies, as it has a direct correspondence with the frequency content in the PSD via the Fourier transform (FT). Different signals for the

CONTACT Andreas Klöckner andreas.kloeckner@dlr.de Institute of System Dynamics and Control, DLR German Aerospace Center, Oberpfaffenhofen, Germany

This article is an improved version of the article 'How to Shape Noise Spectra for Continuous System Simulation' from the 11th International Modelica 2015 Conference. Some parts are reproduced in this article.
The results of this article can be reproduced using the code made available on http://dlr-sr.github.io.

same PSD can be generated by choosing random phase shifts for each cosine component. The periodicity of the signal can be countered by varying the phase shifts at the boundary of the periodic signal. The same effect may be achieved by varying the amplitude of the cosine components, if their statistical mean is preserved. A desired signal may also be formed using different orthogonal basis functions, such as square waves or other wavelets.

A more pragmatic approach is to use shape filters: Linear time invariant (LTI) systems driven by white noise. Typical multi-body simulation software uses this approach (e.g. [9]) and excellent theory is available (e.g. [6]). In order to correctly simulate the system behaviour, stochastic integration schemes should be used. However, these are usually not efficient to use practically or they are simply not available in many tools. Sampled implementations are thus used frequently as approximations of the continuous case.

Another approach is fractional-order modelling [10], which allows to simulate also non-rational transfer functions, such as $1/f^{\alpha}$ noise [6]. Additionally, the noise can be modelled as a weakly correlated sample sequence, which only considers the correlation of a noise sample with its immediate predecessor [7].

All of the methods described above may generate the signal either in a pre-processing step or online during the simulation. Generating noise signals online includes the signal's statistics in the simulation itself, and not in an additional pre-processing step. However, generating the signals online is either very demanding, for example, by evaluating many basis functions in each time step; or it is limited to discrete implementations, which require events to be generated in continuous system simulations.

The AdvancedNoise library provides a new class of random number generators for this purpose: DIRCS Immediate Random with Continuous Seed allows to generate random numbers directly from an input signal without internal states. It thus eliminates the need for time-events and can be used to generate a random signal directly from the time variable. This has been shown to positively affect the simulation performance [11]. Additionally, it allows to define noise signals in dimensions other than the time, which is advantageous in several applications. Rail irregularities, for example, are typically defined with respect to the location on the track. Turbulence models used in aviation also assume a static wind field flown through by the aircraft.

A remaining problem is to choose a proper linear shape filter. In the case that the PSD is a rational function with respect to the *squared frequency*, a spectral factorization of the PSD can be derived analytically. See Liepmann [12] for an aeronautical example. However, typical PSDs are specified as non-rational functions or even tabulated data. If the system excited by the noise signal is also an LTI system, the tabulated data can be applied in the frequency domain by multiplying the PSD with the squared transfer function of the model. See Frederich [13] for a railway application and [14] for a typical aircraft application. In the most general case, the PSD must be approximated by a suitable function, which is then translated into an LTI filter.

In summary, it is not at all obvious how to parameterize the frequency properties of a noise signal. The main contribution of this article is thus to provide a general and systematic method to shape the frequency content of such signals, directly using the specified PSD. The method will turn out to be simple to use and to be applicable to almost any kind of noise spectrum. Combining a discrete convolution algorithm with the DIRCS generator, this method is used to correctly and efficiently generate shape-filtered noise as a function of arbitrary input signals.

The article is structured as follows:

- Section 2 summarizes how noise is typically specified, using the example of rail irregularities.
- In Section 3, we shortly summarize the process of noise generation and define the basic properties of the generator.
- Starting from a given PSD, we rigorously derive a way to shape this frequency content onto a noise signal in Section 4.

- In Section 5, we implement the approach and verify that it yields the same results as conventional methods.
- Section 6 finally summarizes the approach and the results.

2. Specification of railway track irregularities as a noise signal

Besides safety and operating efficiency, it is an essential goal of railway vehicle design to provide an accepted level of vibration comfort. In order to take human perception into account, different methods and standards exist for passenger comfort assessment. However, all of these rely on the accelerations experienced by the passengers as input information.

From a vehicle dynamical point of view, these accelerations are the result of forced vibrations of the vehicle/track system that is excited by track irregularities. In [13], a large number of track measurements are analysed and representative PSDs for good, average and bad tracks are introduced, see Figure 1. Note, these numbers quantify the irregularity per meter track length, that is, with respect to the spatial frequency (unit: 1/m). They have to be transferred into the time domain by taking the vehicle speed into account, see for example, [15].

Regarding the vehicle/track system that is excited by the track unevenness, we confine ourselves to vertical dynamics and use the simplified quarter car model shown in Figure 2. The excitation input is introduced as a variable track height u defined as a stochastic function of the longitudinal track position. The wheel/rail contact is represented by a stiff but linear spring/damper system. The rail and its support constitute a dynamical subsystem on the track side of the model; the suspension and the car body form the vehicle subsystem. The acceleration of the car body a is the output quantity of the model. A Bode diagram of the considered system is depicted in Figure 3. It describes the output/input relationship in the frequency domain.

Here, the model is defined linear to serve as a reference. The equations of motion read

$$M\ddot{z} + D\dot{z} + Kz = h\ddot{u} \ ,$$

where u denotes the track irregularity. The vector of motion coordinates z consists of the spring lengths from mass to mass, that is,

$$z = \begin{pmatrix} z_t & z_w & z_b \end{pmatrix}^{\mathrm{T}}.$$

The mass, stiffness and damping matrix as well as the excitation influence vector h are defined using the parameters compiled in Table 1 as follows:

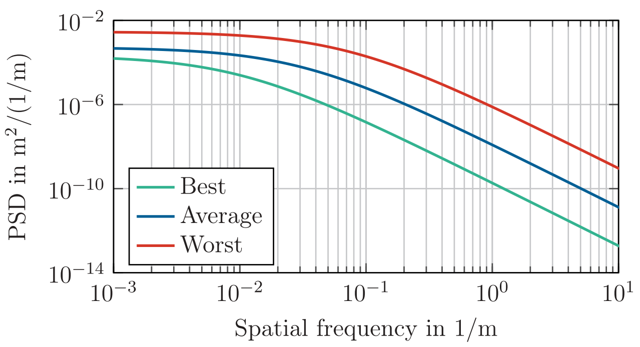

Figure 1. Representative track irregularity PSDs [13].

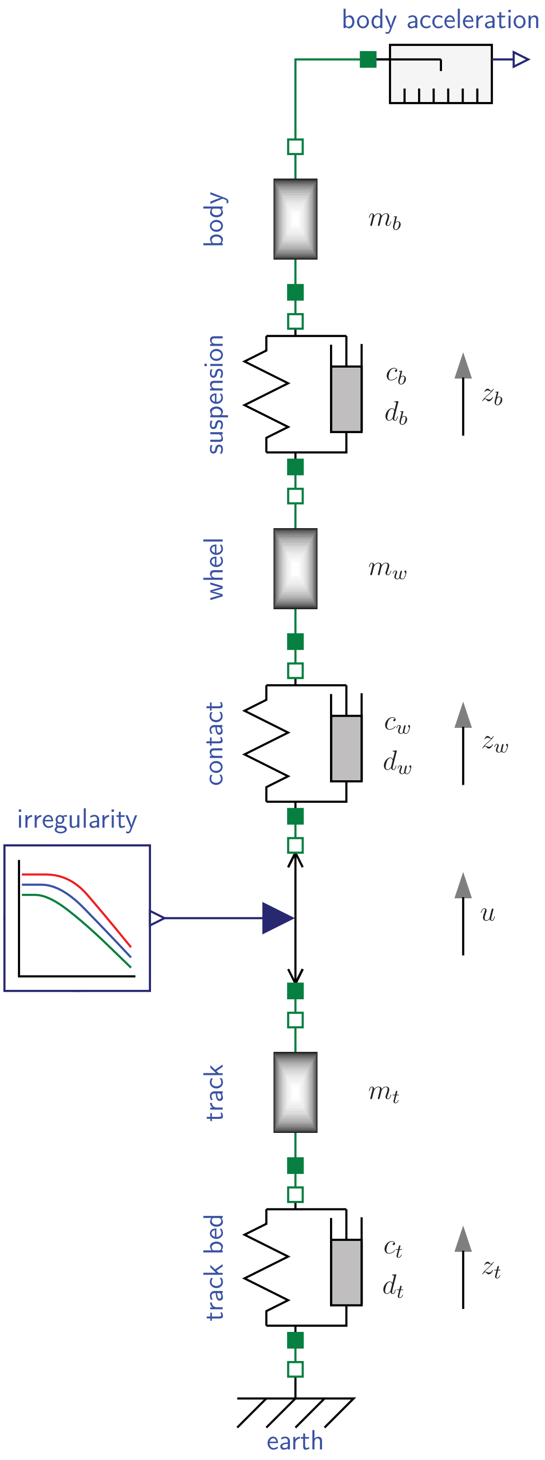

Figure 2. Simplified quarter car model of a railway vehicle in Modelica.

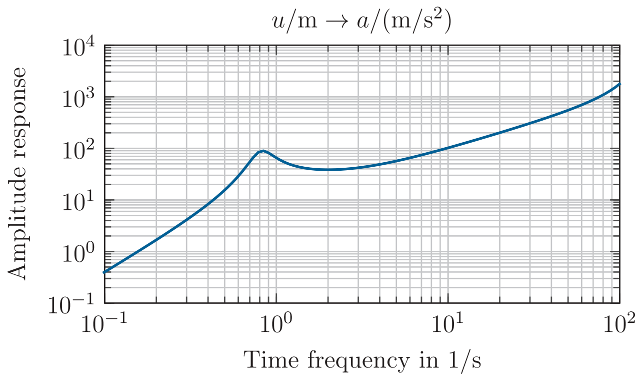

Figure 3. Amplitude response from track irregularity (in m) to body acceleration (in m/s^2).

Table 1. Exemplary quarter railway car parameters.

Track and track bed	m_t	0.165×10^3kg
	c_t	7.50×10^7N m^{-1}
	d_t	9.40×10^7N s m^{-1}
Contact and wheel	m_w	1.25×10^3kg
	c_w	9.90×10^8N m^{-1}
	d_w	1.00×10^5N s m^{-1}
Suspension and car body	m_b	6.75×10^3kg
	c_b	1.75×10^5N m^{-1}
	d_b	1.05×10^4N s m^{-1}

$$M := \begin{pmatrix} m_t + m_w + m_b & m_w + m_b & m_b \\ m_w + m_b & m_w + m_b & m_b \\ m_b & m_b & m_b \end{pmatrix} ,$$

$$K := \mathrm{diag}\{c_t, c_w, c_b\} ,$$

$$D := \mathrm{diag}\{d_t, d_w, d_b\} ,$$

$$h := -(\, m_w + m_b \quad m_w + m_b \quad m_b \,)^{\mathrm{T}} .$$

Presuming a constant running speed v of the vehicle, the PSD of the track irregularity can be transferred from the spatial frequency to the temporal frequency domain by scaling the spatial frequency with v. The acceleration response of the car body can then be evaluated directly in the temporal frequency domain ([16], Ch. 6). This solution allows for comparison and validation with results obtained from time domain simulations in Modelica. Figure 4 presents the pure frequency domain results that are based on the excitation by a track of all three qualities.

The results presented in this article are confined to linear systems in order to validate the time domain simulation approach with a well know frequency domain solution. However, this limitation can be dropped, once the results from time domain simulations with appropriately shaped noise spectra are validated. Time domain simulations are then available for non-linear systems, are capable of running with variable speed and may consider singular disturbances such as running over railway switches as well.

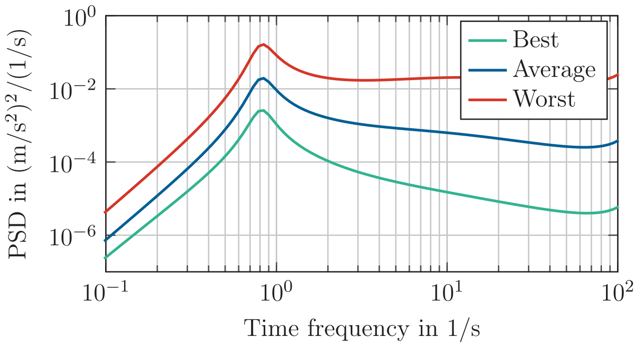

Figure 4. PSD of the body acceleration a for different track irregularities at a velocity of $v = 100\,\mathrm{m/s}$.

3. Review of general noise parameters

The general degrees of freedom in parameterizing noise have been described in detail earlier [3]. A noise signal can be specified in three steps:

(1) Select a random number generator, which generates uniformly distributed random numbers with certain statistical properties, such as subsequent numbers being independent from each other.
(2) Transform the uniformly distributed random numbers in order to match a given probability density function, such as for a normal distribution.
(3) Interpolate the resulting stream of correctly distributed random numbers.

The random number generators of the xorshift family [17] have been included in the Modelica standard library with its last release 3.2.2.[2] These generators have very strong statistical and computational properties and are thus used in the library without exception.

In all cases, where unsampled random numbers are required, the DIRCS generator is used. This random number generator does not require a state but generates a random number directly from a double input signal. To this end, the xorshift64* algorithm is first initialized with the double input signal casted to two integer values. Due to the good startup characteristics of the xorshift64* generator, a raw, uniformly distributed random number is obtained after a single iteration. In this way, high quality random numbers are produced with a low computational effort, for arbitrary input values, and without the need for sampling.

The standard normal distribution is chosen for all random numbers generated in this work. This does not allow to reproduce effects commonly found in measurement noise, such as discretization. However, the choice is reasonable when complex filters are used to shape the actual noise signal to be used in the simulation. Typical filter parameterizations for rail irregularities, for example, assume standard normal distributions of their input signals [9]. Additionally, the subsequent interpolation relies on computing the weighted sum of consequent random numbers. Following the central limit theorem, the result will inevitably be shaped towards the normal distribution.

In previous work, we have described three distinct interpolation functions for noise signals. These include piece-wise constant and linear interpolations as well as a smooth interpolation using the sinc function. The interpolations yield a continuous-time random signal $r(t)$ by computing the sum of consequent random numbers w_i, weighted with a real-valued kernel function $k(t)$:

$$r(t) = \sum_{i=\lfloor t/\Delta t \rfloor - n}^{\lfloor t/\Delta t \rfloor + n} w_i \cdot k(t - i\Delta t). \tag{1}$$

In order to interpolate the random numbers, the kernel function is constrained by

$$k(i\Delta t) \overset{!}{=} \begin{cases} 1 \text{ if } i = 0 \\ 0 \text{ if } i \neq 0 \end{cases}. \tag{2}$$

In these equations, Δt is the sample period of the random numbers and the interpolation base n has to be chosen according to the selected kernel $k(t)$.

The smooth interpolation using the sinc function is already tied strongly to the frequency content of an ideal band-pass filter. However, for the more general case of a given PSD, the final interpolation step has to be replaced by a more powerful approach relaxing the interpolation constraints as described in the following sections.

4. Application of a given PSD to a noise signal

As already explained in Section 2, instead of (band-limited) white noise, coloured noise is required for most practical applications. The required frequency content of the noise signal is usually specified by a given PSD $\Phi(f)$. In order to apply this PSD to a raw white noise signal with piece-wise constant interpolation a linear form filter is typically applied ([9], VIII-TE:8). This filter $H(f)$ defines a mapping in the frequency domain between the white noise input vector $w(t)$ and the coloured noise output vector $r(t)$:

$$R(f) = H(f)W(f), \tag{3}$$

where $R(f)$ and $W(f)$ are the FT of $r(t)$ and $w(t)$. White noise is defined by its flat PSD of $W(f) \equiv 1$. If the filter $H(f)$ is applied to white noise, the PSD of the coloured noise is thus simply

$$\Phi_r(f) = R(f)^2 \equiv H(f)^2. \tag{4}$$

In order to shape coloured noise to a given PSD, the required filter is hence constrained by

$$H(f)^2 \overset{!}{=} \Phi(f). \tag{5}$$

In practice, the filter is typically applied by fitting a rational transfer function on $\Phi(f)$ which is then simulated as an additional linear block in the model (see Section 4.1). An alternative exploiting the interpolation kernel from Equation (1) is proposed in Section 4.2.

4.1. *Using a transfer function*

Before the approximation of a given PSD with a rational transfer function is explained, important properties are briefly repeated. For the approximation, the filter $H(f)$ is restricted to rational functions which are usually expressed with respect to the Laplace variable $s = d + 2\pi jf$, that is,

$$H(s) = \frac{N(s)}{D(s)} = \frac{\sum_{k=0}^{n_z} a_k s^k}{\sum_{l=0}^{n_p} b_l s^l}. \tag{6}$$

The coefficients a_k of the numerator $N(s)$ and the coefficients b_l of the denominator $D(s)$ are real numbers. Both, the numerator and denominator can be factorized, which leads to

$$H(s) = \frac{a_{n_z}}{b_{n_p}} \cdot \frac{\prod_{k=1}^{n_z}(s - z_k)}{\prod_{l=1}^{n_p}(s - p_l)}. \tag{7}$$

Every zero z_k and every pole p_l is either a real number or two zeros (or poles) are each a complex conjugate pair. A necessary condition to express $H(s)$ in a state space representation is that $H(s)$ is proper, that is, the number of poles n_p is greater than or equal to the number of zeros n_z. If the real parts of all poles and zeros are negative, a transfer function is called minimum phase. [3]

Because a suitable transfer function $H(s)$ cannot be analytically computed from a given PSD $\Phi(f)$ in general, a least squares fit is performed:

$$\min_{a_k, b_l} \sum_{i}^{n_f} \left(|H(2\pi \mathrm{j} f)| - \sqrt{\Phi(f)} \right)^2. \tag{8}$$

The coefficients a_k and b_l are chosen as decision variables because they are real numbers and independent from each other. In order to ensure that the filter is proper, $n_z = n_p - 1$ is chosen. The optimization is pursued with MOPS (see [18]) and a Levenberg-Marquardt algorithm is used.

Finally, the minimum phase requirement is fulfilled by a subordinate step. To that end, the zeros z_k and poles p_l of the optimal solution are computed. Afterwards, the real part of every pole/zero is mirrored into the left half plane, for example,

$$\bar{p}_i = -\mathrm{Re}(p_i) + \mathrm{Im}(p_i). \tag{9}$$

Note that the latter operation alters only the phase but not the amplitude of $H(s)$.

Figure 5 compares the reference PSD to the PSDs of the fitted first- and second-order filters. It can be seen that the second-order filter is a very good approximation of the average track irregularity. The second-order filter is thus used to implement the conventional state-space form filter.

4.2. *Using the interpolation kernel*

The filter or transfer function is typically implemented using continuous-time states in Modelica. This approach has two major drawbacks: First, expressing the filter in the time domain limits the simulation to a fixed velocity in order to map the location to a time domain filter. Second, the additional states of the filter require the raw noise signal to be generated accurately using events,

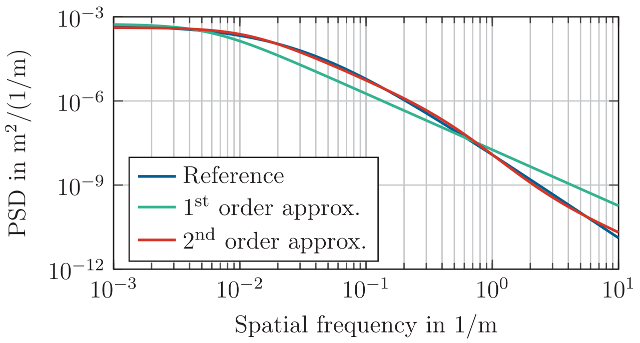

Figure 5. Approximation of the average track irregularity PSD with a first-order and a second-order filter. The second-order filter shows a good fit to the reference.

which considerably slows down simulation, even if only low accuracy is required. In this article, we introduce a different approach to shaping the frequency content of the noise signal using the interpolation kernel $k(t)$ from Equation (1) and the impulse response function (IRF) of a generic form filter.

4.2.1. *Efficient convolution with the IRF*

The idea is based on the convolution theorem. It relates the continuous-time integration of the filter states to a convolution integral of the raw noise signal $w_i(t)$ with the filter's IRF $h(t)$. Exploiting the piece-wise constant noise signal, this approach can be further reduced to a sum of weighted random numbers w_i[4]:

$$
\begin{aligned}
&R(f) = W(f)H(f)\\
&\begin{array}{c}\bullet\\ |\\ \circ\end{array}\\
r(t) &= w_i(t) * h(t)\\
&= \int_{-\infty}^{+\infty} w_i(t)\cdot h(t-\tau)\mathrm{d}\tau\\
&= \sum_{i=-\infty}^{+\infty}\left(w_i\cdot\int_{i\Delta t}^{(i+1)\Delta t} h(t-\tau)\mathrm{d}\tau\right).
\end{aligned}
\tag{10}
$$

The weights in Equation (10) are specified by the integral of the IRF $h(t)$. This integral can also be expressed by differences in the step response $\varsigma(t)$ of the filter. Assuming a strictly stable filter, the step response will converge to fixed values for $t\to\pm\infty$ and the convolution can finally be approximated using a truncated sum:

$$
\begin{aligned}
r(t) &= \sum_{i=-\infty}^{+\infty} w_i(\varsigma(t-(i+1)\Delta t)-\varsigma(t-i\Delta t))\\
&\approx \sum_{i=\lfloor t/\Delta t\rfloor-n}^{\lfloor t/\Delta t\rfloor+n} w_i\cdot(\varsigma(t-(i+1)\Delta t)-\varsigma(t-i\Delta t)).
\end{aligned}
\tag{11}
$$

Note the similar structure of Equations (11) and (1). In our convolution approach, we substitute the interpolation kernel $k(t)$ from Equation (1) by a generalized kernel function in terms of a filter step response:

$$
k(t) = \varsigma(t-(i+1)\Delta t)-\varsigma(t-i\Delta t). \tag{12}
$$

The generalized kernel does not fulfil the constraints from Equation (2). Hence, the resulting noise signal also does not interpolate the random samples: $r(i\Delta t)\neq w_i$. Instead, the random numbers are filtered according to the given PSD.

Note also that, if the generalized kernel $k(t)$ is tabulated and interpolated, the convolution can be further reduced to a discrete convolution with fixed multipliers $k_i = k(i\Delta t)$ and subsequent interpolation. This simplification improves the simulation performance notably, as the kernel $k(t)$ does not need to be evaluated during the simulation:

$$
\begin{aligned}
r(t) &\approx \sum_{i=\lfloor t/\Delta t\rfloor-n}^{\lfloor t/\Delta t\rfloor+n} w_i\left(k_{((\lfloor t/\Delta t\rfloor-i)\Delta t)}+\frac{\mathrm{mod}(t,\Delta t)}{\Delta t}\left(k_{((\lceil t/\Delta t\rceil-i)\Delta t)}-k_{((\lfloor t/\Delta t\rfloor-i)\Delta t)}\right)\right)\\
&= \sum_{i=-n}^{n} w_{\lfloor t/\Delta t\rfloor+i}k_{-i}+\frac{\mathrm{mod}(t,\Delta t)}{\Delta t}\sum_{i=-n}^{n} w_{\lfloor t/\Delta t\rfloor+i}\left(k_{-(i+1)}-k_{-i}\right).
\end{aligned}
\tag{13}
$$

Using the convolution approach described above, all continuous and discrete states can be eliminated from the noise generation. This was shown before to be advantageous for the simulation performance of complex system models [11]. Additionally, an arbitrary PSD can directly be used to shape the noise signal, if the step response of a corresponding filter is known. The derivation of such a step response is shown in the next section.

4.2.2. *Computation of the IRF*

As we have seen, only a suitable IRF is required to shape the desired frequency content. This IRF can easily be obtained from the transfer function derived in Section 4.1. However, in order to avoid the approximation with a rational function, the IRF is directly computed from the tabulated PSD using the framework of FT and inverse Fourier transform (iFT). The resulting IRF is then numerically integrated in order to yield the step response.

Because the PSD describes only the amplitude of the filter and because the filter need not be realized in state space representation it is possible to use the phase as an additional degree of freedom. Here, two different phases are considered: zero phase and minimum phase.

4.2.2.1. *Zero phase.* First the zero phase case is considered. For this case all phases are set to zero. The FT of the interpolation kernel is hence simply

$$K(f) = \sqrt{\Phi(f)}. \tag{14}$$

However, for a correct application of available FT algorithms, the frequency samples must be chosen carefully: In order to yield a real valued $k(t)$, $K(f) = \mathrm{conj}(K(-f))$ must hold. It is further helpful to remember that – because time and frequency are both discretized – $K(f)$ and $k(t)$ are periodically repeated. This is illustrated in Figure 6 for a simple example and the correct samples are marked.

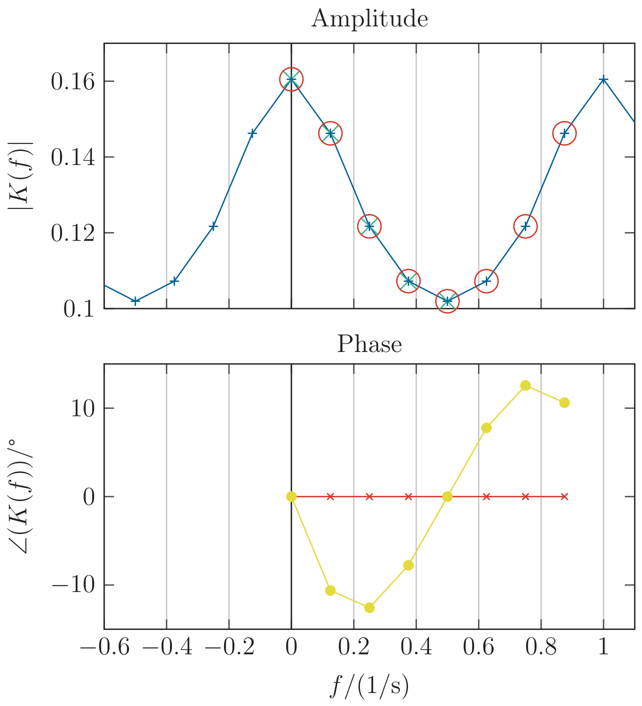

Figure 6. The selection of the correct samples for the iFT and Hilbert transform is illustrated in the frequency domain. The given amplitude data (✕) is mirrored at $f = 0$ and periodically repeated (—+—). Afterwards, the highlighted samples (○) are chosen. In the lower plot, the zero phase (—×—) and the minimum phase (—●—) are depicted.

After the iFT, the resulting $k(t)$ is periodically repeated, too. This allows to chose the correct samples as depicted in Figure 7. As it can be further seen, the zero phase yields a non-causal IRF which is symmetric to $t = 0$.

4.2.2.2. *Minimum phase.*

Second, the minimum phase case is considered. In this case, only the amplitude of the FT of the interpolation kernel is given by the PSD:

$$|K(f)| = \sqrt{\Phi(f)}. \tag{15}$$

The minimum phase $\angle(K(f))$ can be computed using the Hilbert transform (HT). For the HT, the same samples must be chosen as for the iFT. The resulting minimum phase is depicted in Figure 6. Afterwards, the full FT of the interpolation kernel is given by

$$K(f) = |K(f)| \exp(j \angle(K(f))). \tag{16}$$

The transformation into the time domain is subsequently performed in the same way as for the zero phase case. As it can be seen in Figure 7, the minimum phase filter is causal, that is, its IRF is non-zero only for non-negative times $t \geq 0$.

Figure 8 compares IRFs for the average track irregularities as obtained from the procedure outlined above. First, the IRF of the fitted second-order filter is evaluated by simulation and by iFT of the PSD shown in Figure 5. Both results are essentially the same, showing the correct iFT application. The IRFs obtained directly from the given PSD are also shown. The minimum phase IRF is very similar to the fitted filter's IRF, underlining the good fit of the filter. The zero phase IRF is non-causal, as it is non-zero for negative times.

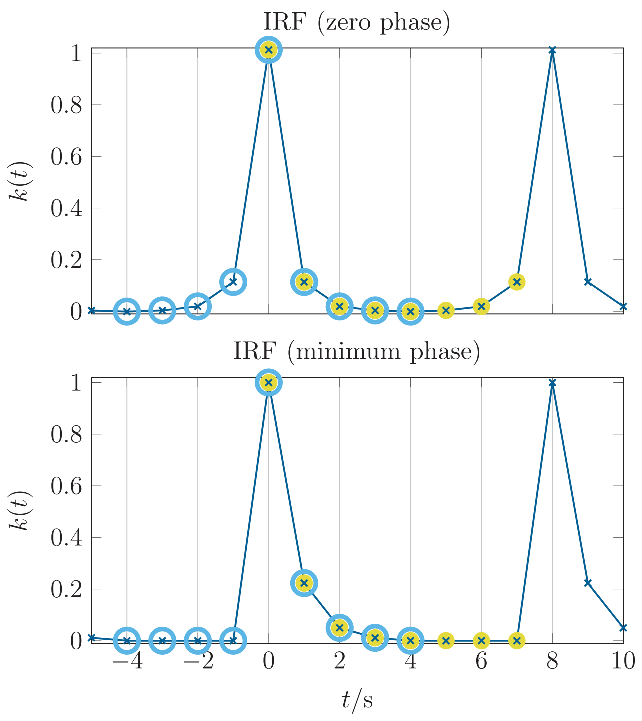

Figure 7. Sample selection in the time domain: The resulting IRF for the zero phase (upper plot) and minimum phase (lower plot) are depicted. The results from the iFT (●) are periodically repeated (—×—) in order to chose the correct samples (○).

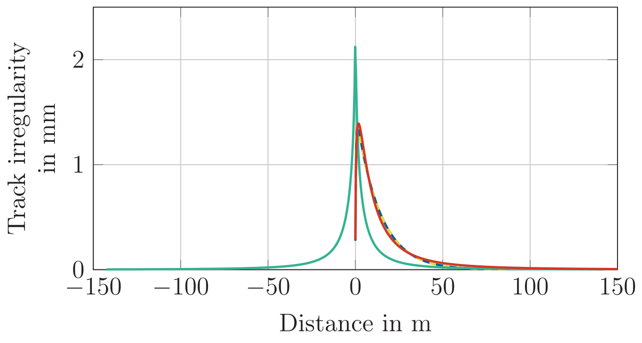

Figure 8. Comparison of different IRF: The IRF of the second-order filter is obtained by simulation (——) and by iFT (- - -). An excellent agreement can be recognized. Additionally, the minimum phase (——) and zero phase (——) IRF are depicted. These correspond directly to the specified PSD.

5. Results

The form filters for average track irregularities are implemented using the AdvancedNoise library according to the procedures outlined above. Using the Dymola 2016 RC2 simulation tool with its DASSL solver [19], the different steps of the implementation are then verified. To this end, the following simulation experiments are presented:

(1) The minimum phase convolution is verified against a state space filter implementation with identical white noise input.
(2) The white noise input of the state space implementation is exchanged by an independent noise source not using the DIRCS algorithm.
(3) The minimum phase and zero phase IRFs are compared to each other.
(4) The quarter car model of the railway vehicle is fed with the zero phase noise convolution filter and compared against the standard frequency domain solution.

5.1. *Convolution verification*

Figure 9 shows a comparison of the fitted second-order filter's state space implementation with its minimum phase convolution implementation. Both filters are driven by identical white noise generated with the DIRCS generator directly from the position on the track. The raw random numbers are generated with a sample period of $\Delta x = 0.4\text{m}$. In order to produce white noise in the spatial domain, a standard deviation of $\sigma = \sqrt{0.5\text{m}/\Delta x}$ is chosen. It can be seen that the white noise spectrum has indeed a spectral density of 1; except for its random nature. The shape of the white noise spectrum directly corresponds to the shape of the track irregularity PSDs. Both irregularity PSDs can be seen to be identical. This is also the case for the actual simulation output u of the track irregularity.

5.2. *Space domain noise verification*

In the second step, the minimum phase convolution implementation is compared to a traditional state space implementation fed by a time domain sampled noise. In order to yield comparable results, the time domain noise is sampled with $\Delta t = \Delta x/v$ and the train is running at a speed of $v = 100\,\text{m/s}$. The standard deviation of the normal distribution is kept the same. Figure 10 compares both results to the reference. Good agreements can be observed in both time frequency

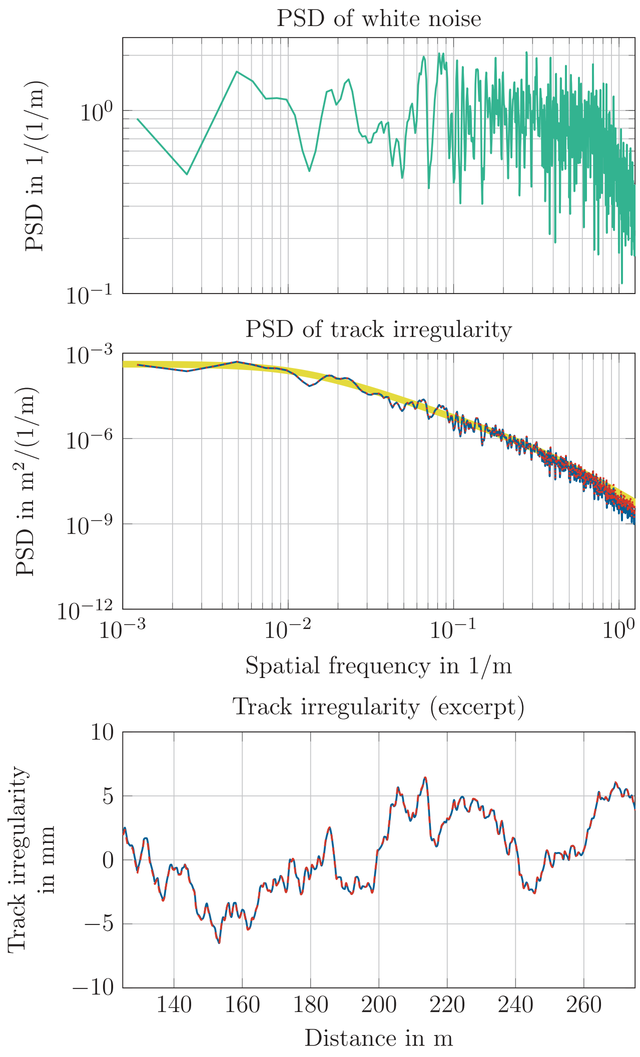

Figure 9. The identical white noise (upper plot) is applied to the second-order filter by simulation (——) and by convolution (······). The PSDs of both implementations (middle plot) are identical and correspond well with the reference PSD (▬). As it can be seen in the lower plot, the signals are also identical.

domain and spatial frequency domain PSDs. The time frequency and spatial frequency PSDs differ only by the constant running velocity of the train.

5.3. *Minimum and zero phase filters*

The comparison of the minimum phase and zero phase convolution implementations can be seen in Figure 11. The white noise generator is DIRCS in both cases and the sample periods are both $\Delta x = 0.4$m. Both IRFs are saved in a tabulated form with a sampling period of 0.1m. Both implementation variants agree very well with the reference in the low frequencies. At very high frequencies, the influence of the two sampling periods is marked with vertical lines. Characteristic marks on the PSDs can be seen both for the sampling of the white noise at 2.5/m and for the sampling of the IRFs at 10/m. The respective sampling periods must thus always be chosen high

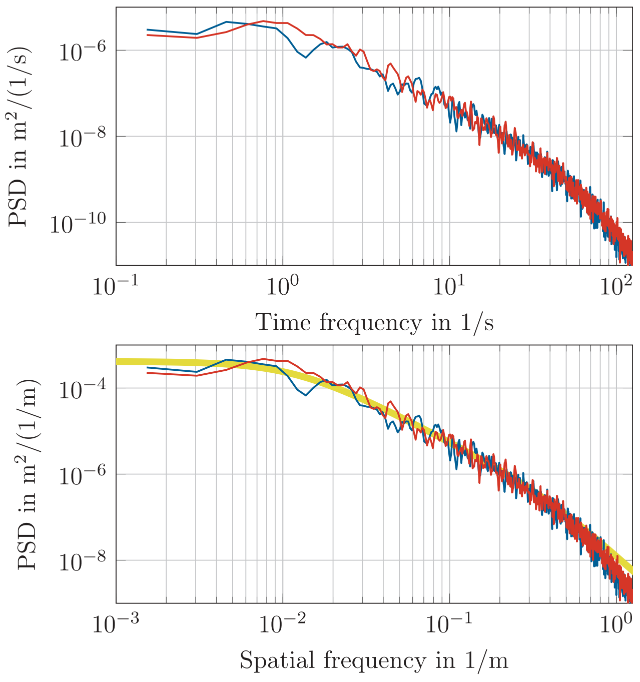

Figure 10. The PSD of the track irregularity from simulation (——) and from convolution (——) are compared in the time frequency domain (upper plot) and in the spatial frequency domain (lower plot). Both agree very well. In the spatial frequency domain, an excellent agreement with the reference (▬) can also be seen.

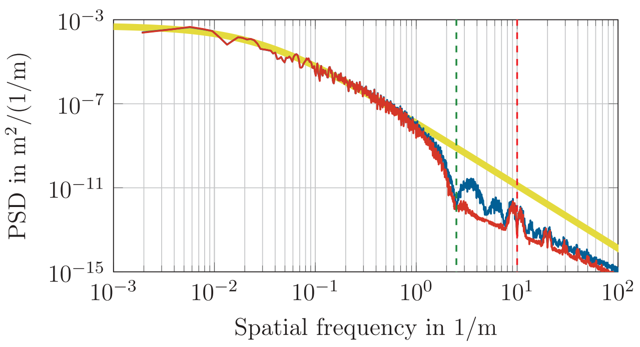

Figure 11. The PSD resulting from the minimum phase (——) and from the zero phase (——) iFT agree well with the reference (▬) at low frequencies. At high frequencies, effects of sampling the white noise (- - -) and of sampling the IRF (- - -) become visible.

enough to resolve all relevant effects. Another characteristic to be taken into account is the portion of the PSD covered by the resolution [8].

5.4. *Quarter car railway vehicle*

Finally, the zero phase convolution implementation is integrated with the quarter car model of a railway vehicle running at $v = 100\text{m/s}$. Figure 12 compares the acceleration of the car body from this simulation to the well trusted frequency domain solution. A very good agreement can be seen.

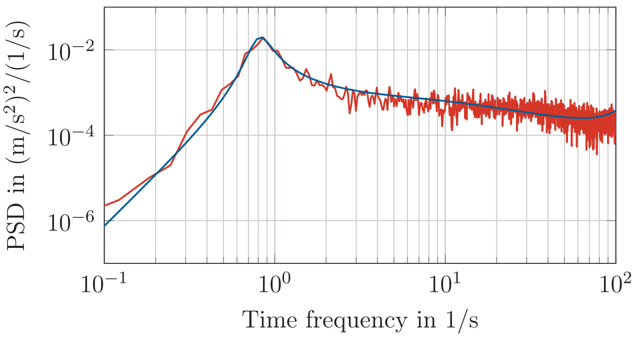

Figure 12. Frequency domain solution is compared (——) to simulation results (——).

6. Conclusions

In this article, we show how to consistently shape a noise signal to match a given frequency content specified by a PSD. The method is introduced at hand of the quarter car model of a railway vehicle, for which well trusted reference solutions are available.

Our method uses the non-recursive random number generator DIRCS. It generates normally distributed random numbers directly from the current location on the track. We then shape the frequency content of the random numbers using a convolution with the impulse response of a form filter. It is shown that the IRF of the form filter can be directly generated from the given PSD using the iFT. In summary, our method consists of the following steps:

(1) Obtain a PSD $\Phi(f)$ for the desired noise signal.
(2) Chose the appropriate sampling period $\Delta x = 0.5/f_{\max}$ according to the highest frequency $f_{\max}$ to be contained in the noise signal.
(3) Convert the PSD to a zero-phase step response $\varsigma(x)$ with sampling period Δx according to Paragraph 4.2.2.1.
(4) Generate random numbers w_i with sampling period Δx using the DIRCS generator and a normal distribution with standard deviation $\sigma = \sqrt{0.5/\Delta x}$.
(5) Apply the convolution of the random numbers with the step response according to equation (13).

The method is implemented in the open-source Modelica AdvancedNoise library and validated against the given spectrum. It is then used to excite a linear quarter car model running at a constant speed. Results obtained with our method agree very well with the standard solutions based on frequency domain computations.

Three fundamental error sources are to be addressed for a discrete convolution approach [6]: the effects of *windowing* and *sampling* during the PSD estimation are assumed to be solved beforehand, because the PSD is the input data to our method. The *aliasing* effect of a limited frequency range can approximately be solved by selecting a suitable discretization, which is able to cover all relevant effects.

Our method is thus shown to produce correct results in the example of a linear railway car model running at constant speed. Additionally, it can be employed not only for linear models but also for non-linear models, vehicles running at varying speed, or in more complex scenarios involving, for example, singular disturbances. Moreover, the method proposed in this article is straightforward to use and can also be applied to a variety of other problems, such as turbulence or street roughness.

In the presented example, the zero-phase convolution variant requires approximately 10 times the computation time as compared to the state-space filter implementation. However, the performance depends strongly on the system driven by the noise signal and the required accuracy [11]. The number of required steps to integrate the model is reduced by a factor of 10 using the convolution approach. More complex, non-linear models will benefit more from the removed sampling and the reduced number of required steps. The same holds if the number of steps can even further be reduced by accepting a lose tolerance.

In summary, we suggest to prefer the proposed convolution approach if in particular one of the three following circumstances is given: (a) it is intended to simulate a very complex system with strong performance penalties due to events. (b) It is required to consider an unusual spectrum, for example, given in different dimensions than the time. (c) It is difficult to approximate the given spectrum by an LTI transfer function since, for example, sharp edges or discontinuities are involved. In other cases, the conventional approach of driving a fitted linear filter with white noise is the straightforward alternative to our convolution approach.

The current implementation is based on the open-source AdvancedNoise library. This makes the method available to a wide audience and also gives room for further improvements. Contributions by the community are highly welcome. Since our implementation of the convolution is only fairly optimized, further developments should take into account the runtime performance of the approach. Extensions of the method itself could possibly be found in higher dimensional noise spectra, correlated noise, or additional effects such as discretization.

In further research, correlated and multi-dimensional noise will be considered. For example, the introductory railway application can be extended to consider two parallel tracks. In this case, it is to be expected that the left and right track irregularities are correlated in the low frequency range, but the correlation is decreasing for higher frequencies as it explicitly has been shown in road applications [20]. An example for multi-dimensional noise is to shape a surface like a parking area which can be used in arbitrary ways by a car.

Notes

1. https://github.com/DLR-SR/AdvancedNoise
2. see https://github.com/DLR-SR/Noise
3. Minimum phase means that both the filter and its inverse are stable and causal.
4. The notation $R(f) \multimap r(t)$ denotes an FT pair.

Acknowledgments

We thank our parents for giving us a great first name.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

[1] A. Klöckner, A. Knoblach, and A. Heckmann, *How to shape noise spectra for continuous system simulation*, in *Proceedings of the 11th International Modelica Conference*, no. 118 in Linköping Electronic Conference Proceedings, September 21-23, Linköping University Electronic Press, Linköpings universitet, Versailles, France, 2015, pp. 411–418.

[2] J. Haase, S. Wolf, and C. Clauß, *Monte carlo simulation with Modelica*, in *Proceedings of the 6th International Modelica Conference*, Vol. 1, Modelica Association and University of Applied Sciences Bielefeld, Bielefeld, Germany, 2008, pp. 601–604.

[3] A. Klöckner, F.L.J. Van der Linden, and D. Zimmer, *Noise generation for continuous system simulation*, in *Proceedings of the 10th International Modelica Conference*, no. 96 in Linköping Electronic Conference

Proceedings, March 10-12, iSBN: 978-91-7519-380-9. ISSN: 1650-3686. eISSN: 1650-3740, Modelica Association and Linköping University Electronic Press, Lund, Sweden, 2014, pp. 837–846.
[4] A. Haigermoser, B. Luber, J. Rauh, and G. Gräfe, *Road and track irregularities: measurement, assessment and simulation*, Vehicle Syst. Dyn 53 (2015), pp. 878–957. doi:10.1080/00423114.2015.1037312
[5] J. Pombo and J. Ambrósio, *An alternative method to include track irregularities in railway vehicle dynamic analyses*, Nonlinear Dyn 68 (2012), pp. 161–176. doi:10.1007/s11071-011-0212-2
[6] N. Kasdin, *Discrete simulation of colored noise and stochastic processes and 1/f alpha; power law noise generation*, Proceedings of the IEEE 83 (1995), pp. 802–827.
[7] V. Quarz, *Die Generierung von Fahrwegstörungen für vorgegebene Spektraldichten mit Hilfe orthogonaler Funktionen*, Ph.D. diss, TU Dresden, 2004.
[8] K.Y.R. Billah and M. Shinozuka, *Numerical method for colored-noise generation and its application to a bistable system*, Phys Rev. A. 42 (1990), pp. 7492–7495.
[9] SIMPACK. *SIMPACK Time Excitations Catalogue*, SIMDOC (2003), pp. v8.607.
[10] A. Pollok, D. Zimmer, and F. Casella, *Fractional-order modelling in Modelica*, in *Proceedings of the 11th International Modelica Conference*, Linköping University Electronic Press, 2015.
[11] F.L.J. van der Linden, A. Klöckner, and D. Zimmer, *Effects of event-free noise signals on continuous-time simulation perfomance*, in *8th Vienna International Conference on Mathematical Modelling*, Mathematical Modelling, Vol. 8, IFAC-PapersOnLine, Vienna, Austria, 2015, pp. 280–285.
[12] H.W. Liepmann, *On the application of statistical concepts to the buffeting problem*, J Aeronaut Sci (Institute Aeronautical Sciences) 19(1952). doi:10.2514/8.2491
[13] F. Frederich, *Die Gleislage aus fahrzeugtechnischer Sicht*, ZEV–Glasers Annalen, 1984, pp. 108–1984. ZEV: Berlin
[14] EASA CS-25, *Certification Specifications and Acceptable Means of Compliance for Large Aeroplanes*, European Aviation Safety Agency, 2013. EASA: Köln.
[15] K. Popp and W. Schiehlen, *Ground Vehicle Dynamics*, Springer, 2010. Springer: Berlin Heidelberg.
[16] K. Knothe and S. Stichel, *Schienenfahrzeugdynamik*, Springer, Berlin, 2003.
[17] S. Vigna, *Further scramblings of marsaglia's xorshift generators*, CoRR abs/1404.0390 (2014), Available at http://arxiv.org/abs/1404.0390, code available at http://xorshift.di.unimi.it/.
[18] H. Joos, J. Bals, G. Looye, K. Schnepper, and A. Varga, *A multi-objective optimisation-based software environment for control systems design*, in *Conference on Computer-Aided Control Systems Design*, IEEE, 2002.
[19] L. Petzold, *Description of DASSL: a differential/algebraic system solver*, in *10th International Mathematics and Computers Simulation Congress on Systems Simulation and Scientific Computation*, 9 Aug, Available at http://www.osti.gov/scitech/servlets/purl/5882821, Elsevier, Montreal, Canada, 1982.
[20] K. Bogsjö, *Coherence of road roughness in left and right wheel-path*, Vehicle Syst. Dyn 46 (2008), pp. 599–609. doi:10.1080/00423110802018289