PulseSpectra : User Guide

Justin Salez

July 26, 2005

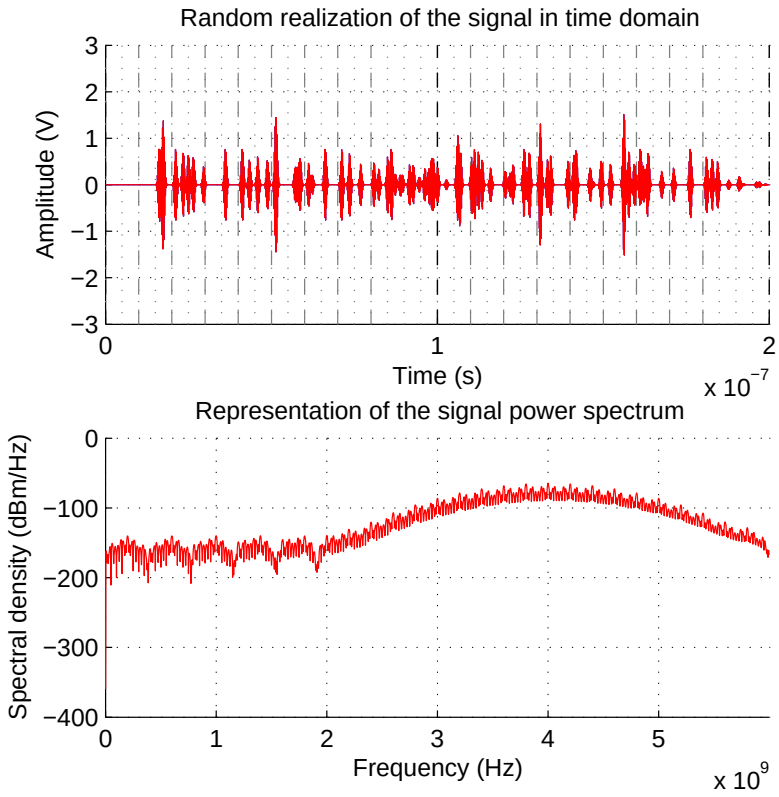

Figure 1: Time randomly generated realization (above) and frequency power spectrum (under) of a typical TH-IR signal in presence of multipath fading effects and jitter : PulseSpectra ('filter','gws','ppm','pam','th','jitter','multipath','time',2,'dbm','freq',[0;6e9])

1 Program Purpose

PulseSpectra is a swiss army knife Matlab toolbox for computing the spectra of pulse modulated signals, including UWB signals and multipath faded pulse modulated signals. This program is a direct implementation of the spectrum formula presented in the Ph.D. thesis "Power Spectra of Random Spikes and Related Complex Signals, with Application to Communications", Andrea Ridolfi, 2004.

Starting with a basic regular grid of customizable period, *PulseSpectra* enables the user to add, in a modular way, a wide range of specific features in order to model various coding operations (such as filtering, modulation and multi-user detection) as well as channel effects altering the signal (including jittering, random losses or multipath fading), by simply enumerating the corresponding keywords in an array of arguments and accompanying them with custom attributes whenever defaults settings are not satisfying.

This design approach thus combines the simplicity of predefined options with the expressivity power of customizable structures : for each available feature, the user simply has to choose between adding it or ignoring it, and if adding it, may assign any custom value to all corresponding parameters but is not obliged to do so since for each of them, standard and relevant default settings are automatically loaded in abscence of further precisions.

PulseSpectra also includes a few interesting additional features, such as a generator of random realizations of the signal in time domain, an optional separation between the density part and the dirac part of the signal spectrum as well as a fit-to-spectrum procedure that automatically resizes the frequency display window according to the frequency band of the signal.

Please report bugs or comments to `justin.salez@ens.fr`.

2 Arguments overview

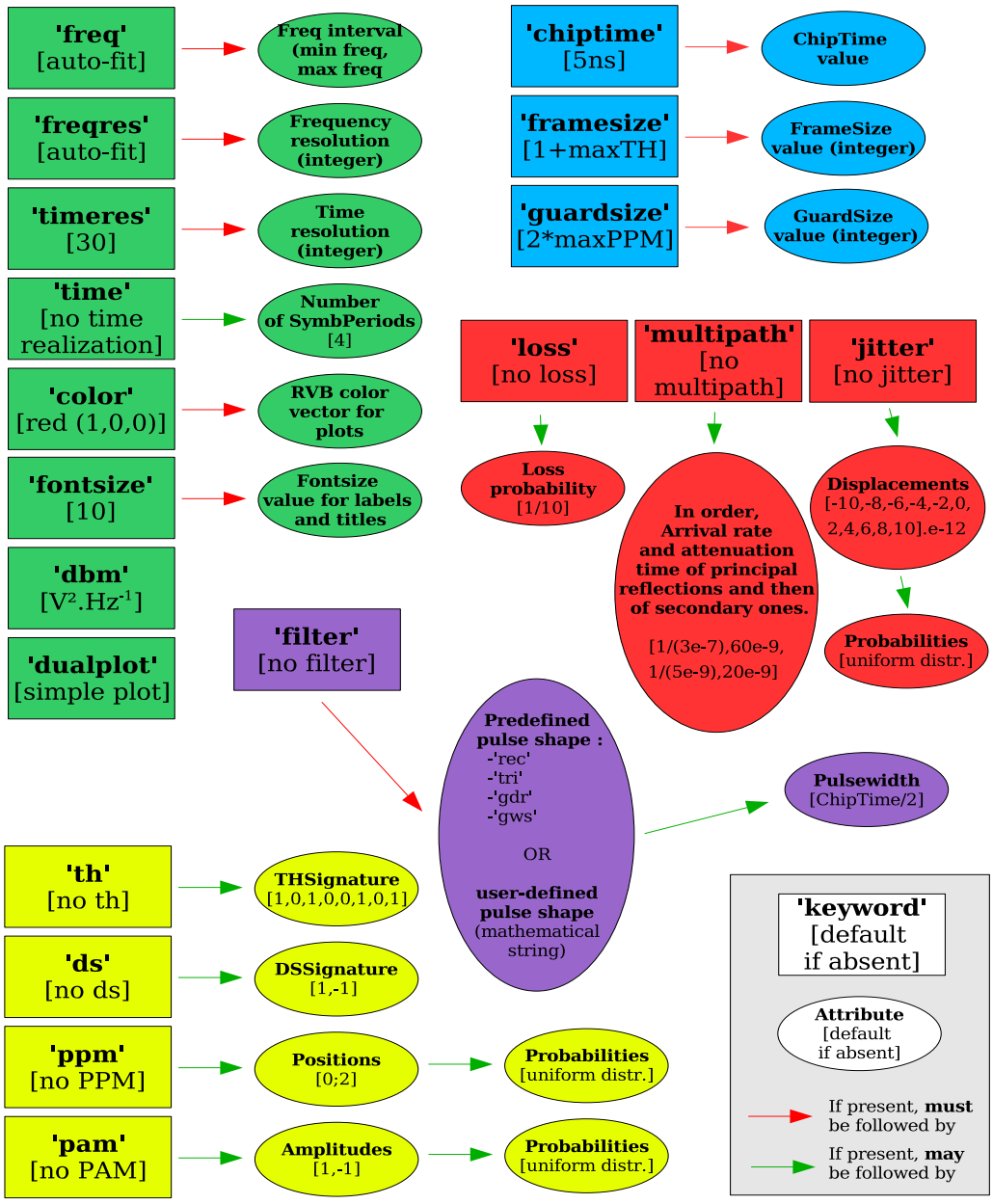

Figure 2: PulseSpectra('filter','gws','ppm','dualplot')

3 Arguments description

The argument taken by the *PulseSpectra* function is a (possibly empty) sequence of keywords and attributes. Each of the keywords corresponds to a specific property, option or feature that should be added to the process and is possibly followed by one or more litteral or numerical attribute(s) precising the value to which the corresponding property has to be set or modifying the various parameters on which the corresponding feature has to depend. All keywords are optional and case-insensitive. They can be specified in any order. If no argument at all is specified, the function simply plots the frequency representation of a basic regular grid according to some default settings. The optional keywords and their corresponding attributes allow the user to :

1. Specify himself the various parameters of the grid (chip time, guard size and frame size) if the default ones do not convene. See section 2.1 for a detailed description.

2. Add many different coding features, in a modular way, to the basic point process (including amplitude and position modulation, time-hopping and direct-sequencing) and precise their various parameters (symbols distibution for amplitude and position modulation, time-hopping and direct-sequencing signatures, etc) if the default settings do not convene. See section 2.2 for a detailed description.

3. Filter the so-obtained modulated random spike field using either one of the standard predefined pulse shapes or any user-specified impulse response given under the shape of its mathematical definition. See section 2.3 for a detailed description.

4. Model various random effects altering the resulting signal (such as jittering, thinning or multipath fading) and, here again, specify values for their various parameters if necessary (jittering distibution, loss probability, arrival rates and attenuation speeds of principal and secondary multipath reflections, etc). See section 2.4 for a detailed description.

5. Customize a few useful display parameters (such as time and frequency resolution, plot mode, Y-axis unit, size of the display window) if automatic adjustments are not satisfying enough. See section 2.5 for a detailed description.

3.1 Customizing the basic point process

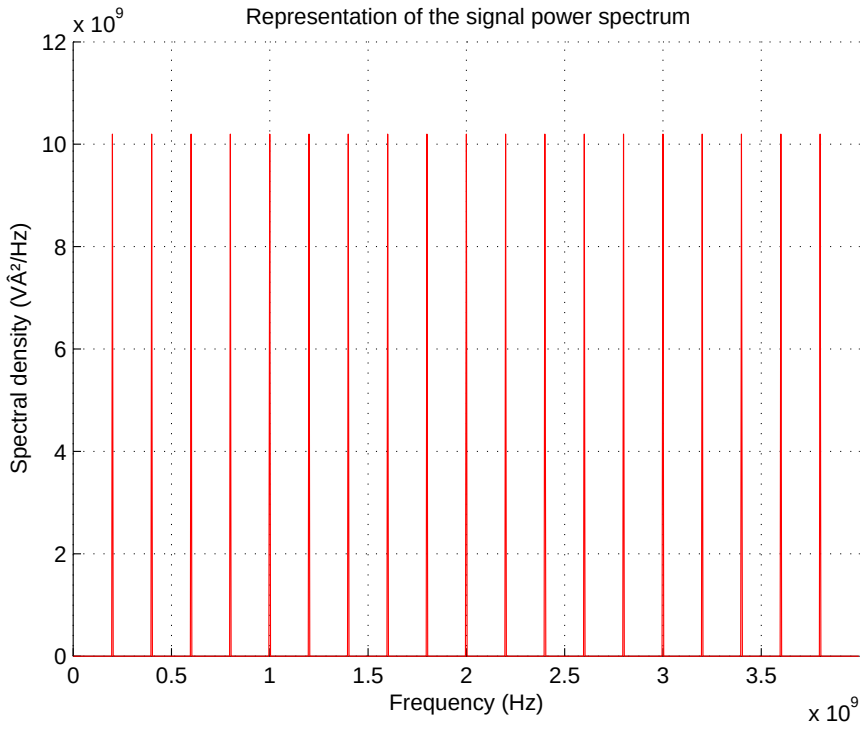

Figure 3: Spectrum of the basic regular grid : $PulseSpectra('freq', [0; 4e9])$

3.1.1 Chip time

The keyword '*chiptime*', immediately followed by any positive number, sets the basic unit of time of the process to the specified value in seconds. Default is 5 ns.

3.1.2 Frame size

The period of the basic regular grid corresponds to the total duration of one coding symbol. This symbol period is divided into one or more frames (in case of time-hopping or direct-sequencing), each frame containing one element of the symbol signature. The keyword '$framesize$', immediately followed by any positive integer, sets the frame period to $ChipTime$ times the specified value. Default is the smallest possible one allowing correct time-hopping, i.e $(1 + max(THSignature))$.

3.1.3 Guard size

In order to ensure that the symbol displacements due to position modulation and jitter will not result in the overlap of two consecutive symbols, a so called 'guard time' can be added at the end of each symbol period. The keyword '*guardsize*', immediately followed by any positive integer, sets the guard time to $ChipTime$ times the specified value. Default is the reasonnable value $2 \times max(PPMAmplitudes)$.

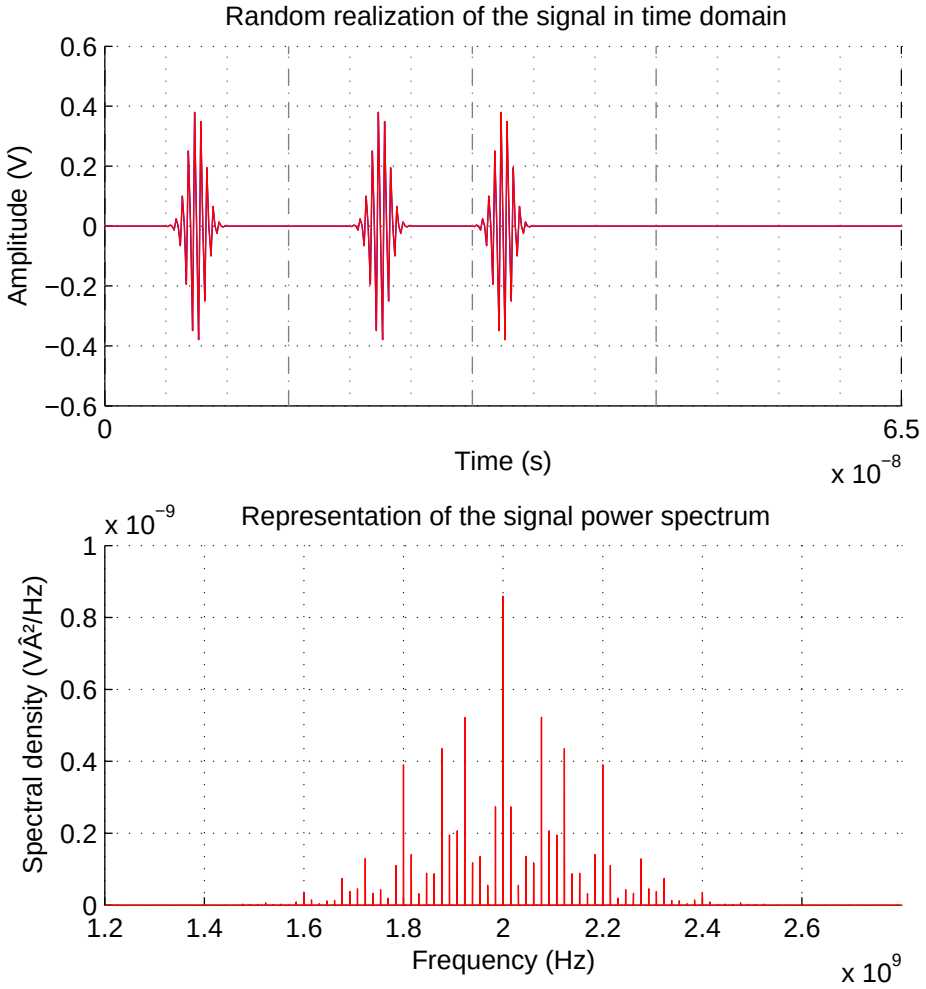

Figure 4: PulseSpectra('filter','gws',5e-9,'th',[1,1,0],'guardsize',4,'framesize',3,'time',1)

3.2 Adding coding features

3.2.1 Pulse position modulation

In pulse position modulation the information is carried by the relative distance of the pulses with respect to a regular grid. If present, the keyword '*ppm*' adds position modulation to the signal. An optional attribute may follow and specify the amplitudes of each possible random displacement (in terms of multiples of ChipTime) under the shape of a vector of non-negative integers, the size of which thus determining the number of PPM coding symbols. In case this attribute would be missing, the default value is the binary $[0; 2]$ modulation. A second vector with the same size may then optionally follow that first one to specify the probabilities of each single random displacement, order-respectively. If that vector is not specified, the default setting is the uniform distribution. If the sum of the specified probabilities does not equal 1, the vector is automatically normalized.

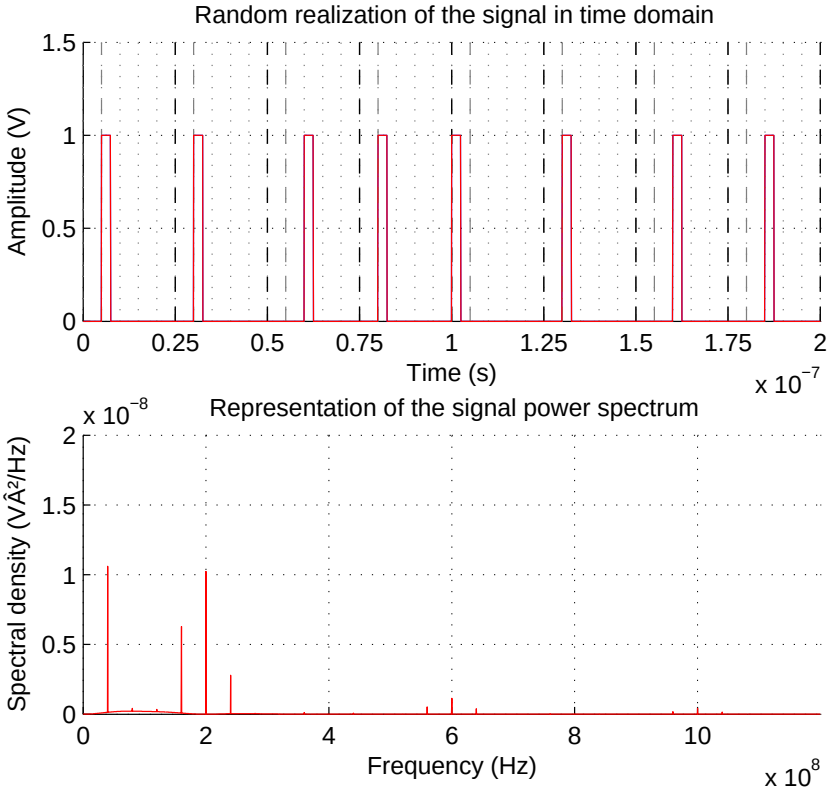

Figure 5: PulseSpectra('filter','rec','ppm',[0,1,2],[0.25,0.5,0.25],'time',8)

3.2.2 Pulse amplitude modulation

A pulse amplitude modulation transmits the information through the amplitude of the pulse. If present, the keyword '*pam*' adds amplitude modulation to the signal. A first optional attribute may follow to specify the possible multiplicative factors for amplitude under the shape of a vector, the size of which thus determining the number of PAM coding symbols. In case this attribute would be missing, the default value is the standard $[1; -1]$ BPSK, which ensures the disparition of the spikes. If that first optional vector is present, a second one, with the same size, may then specify the probabilies of the amplitude-coding symbols, order-respectively. If absent, the default setting is the uniform distribution. If the sum of the specified probabilities does not equal 1, the vector is automatically normalized.

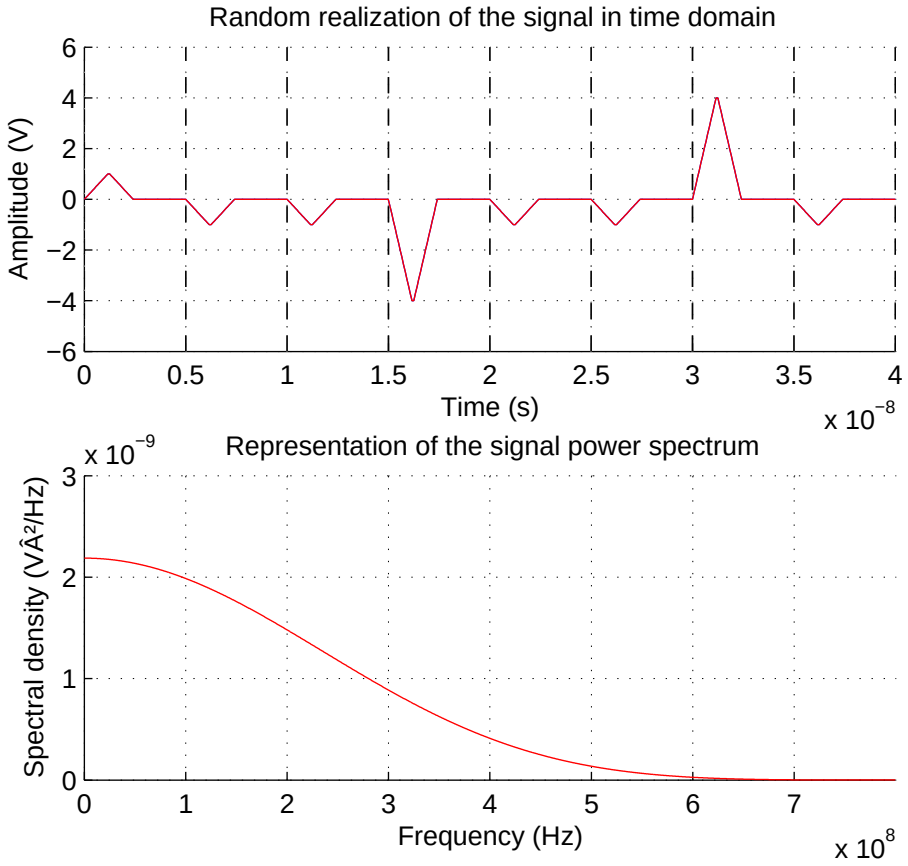

Figure 6: PulseSpectra('filter','tri','pam',[1,4,-1,-4],[0.3,0.2,0.3,0.2],'time',8)

3.2.3 Time-hopping

General time-hopping signals are characterized by a deterministic periodic pattern that allows for multi-user detection. Such a pattern, called signature, consists of a sequence of pulses positioned with respect to the regular grid. If present, the keyword '*th*' adds time-hopping to the signal, the basic time step being $ChipTime$. Default signature is [1,0,1,0,0,1,0,1], but an optional attribute may follow to specify any other time-hopping signature under the shape of a vector the size of which determines the number of frames composing a symbol period. In case both TH and DS would be present, the two signatures are periodically extended so that they have the same size, which is thus the LCM of the two original sizes. See section 2.1.2 for customizing the frame period.

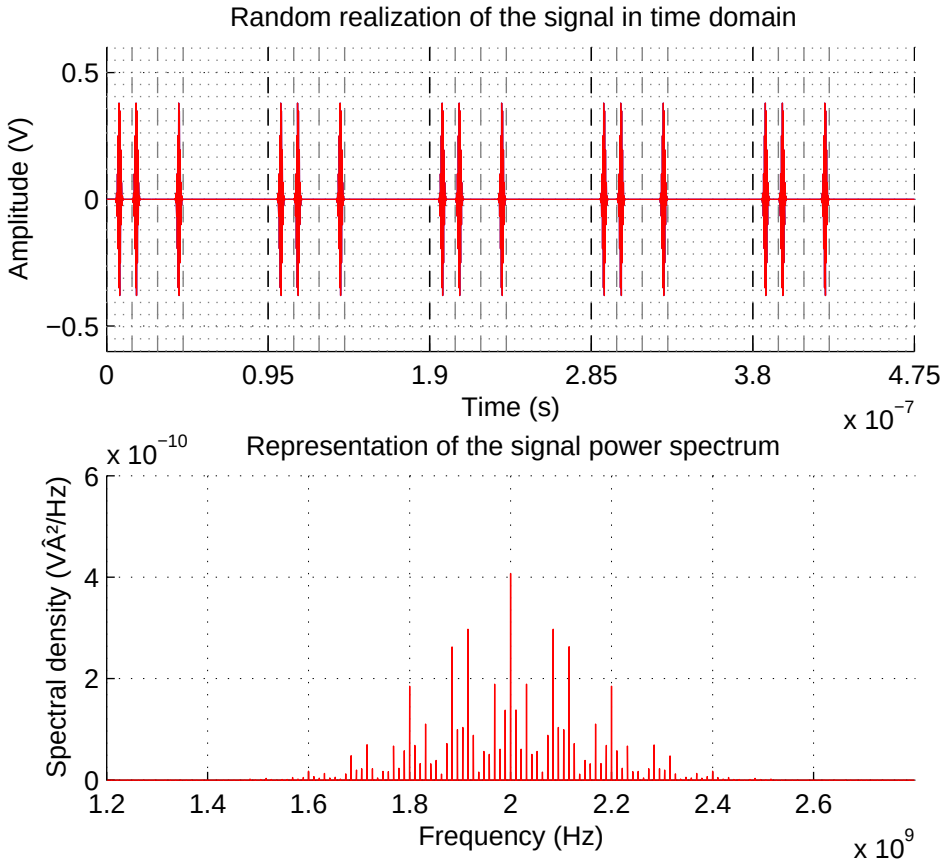

Figure 7: PulseSpectra('filter','gws',5e-9,'th',[1,0,2],'time',5,'guardsize',10)

3.2.4 Direct-sequencing

As for time-hopping, a direct-sequence signal allows for multi-user detection. It consists of multiplying the stream of pulses by a periodic deterministic sequence of +1 and -1 (the signature). If present, the keyword '*ds*' adds direct-sequencing to the signal. Default signature is [1,-1], but an optional attribute may then follow to specify any other DS signature under the shape of a vector the size of which determines the number of frames composing a symbol period. In case both TH and DS would be present, the two signatures are periodically extended so that they have the same size, which is thus the LCM of the two original sizes. See section 2.1.2 for customizing the frame period.

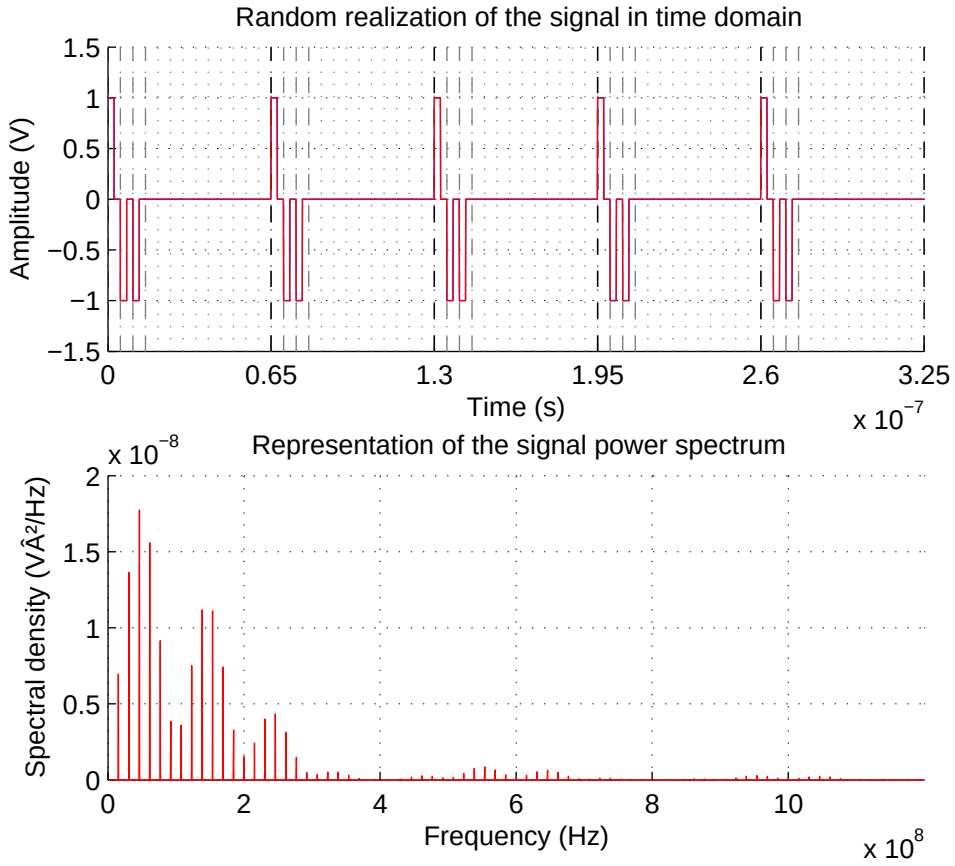

Figure 8: PulseSpectra('filter','rec','ds',[1,-1,-1],'time',5,'guardsize',10)

3.3 Filtering

The resulting process of any combination of the previous transformations over a basic regular grid is not a *bonafide* wss signal, and has no rigorous correspondance within the real world of communications. It can rather be seen as a random stream of modulated delta functions. The only thing missing to turn such a random spike field into a realistic signal is the filtering operation which in fact 'replaces' every dirac with a real pulse shape. If present, the keyword '$filter$' adds filtering to the basic process. It must be immediately followed by a string attribute either corresponding to one of the available predefined pulse shapes or expliciting the definition of any other impulse response. In the first case, the possible predefined pulse shapes are the following :

- 'rec' : rectangular pluse shape

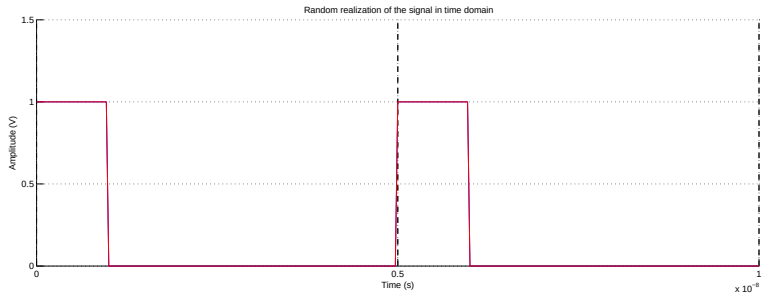

Figure 9: PulseSpectra('filter','rec',1e-9,'time',2)

- 'tri' : triangular pulse shape

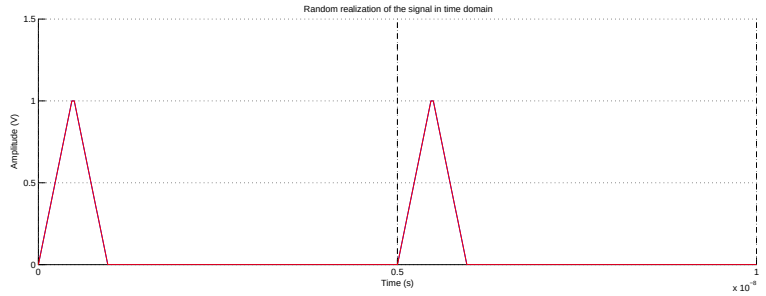

Figure 10: PulseSpectra('filter','tri',1e-9,'time',2)Predefined pulse shapes

- '*gdr*' : gaussian-derivative pulse shape

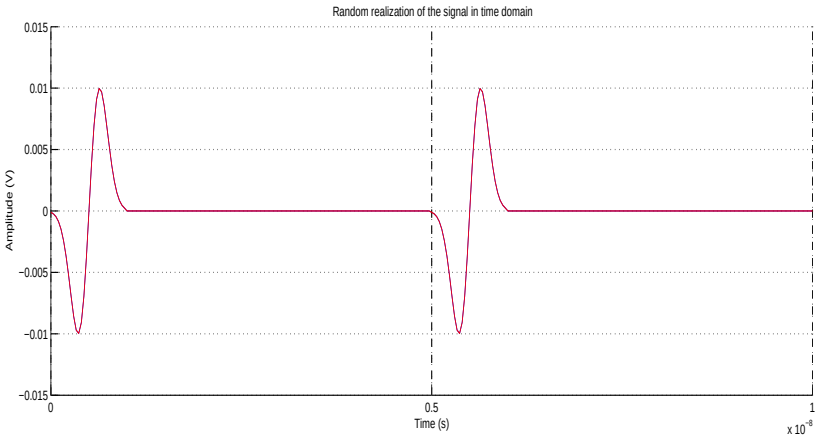

Figure 11: PulseSpectra('filter','gdr',1e-9,'time',2)

- '*gws*' : gaussian windowed sine pulse shape

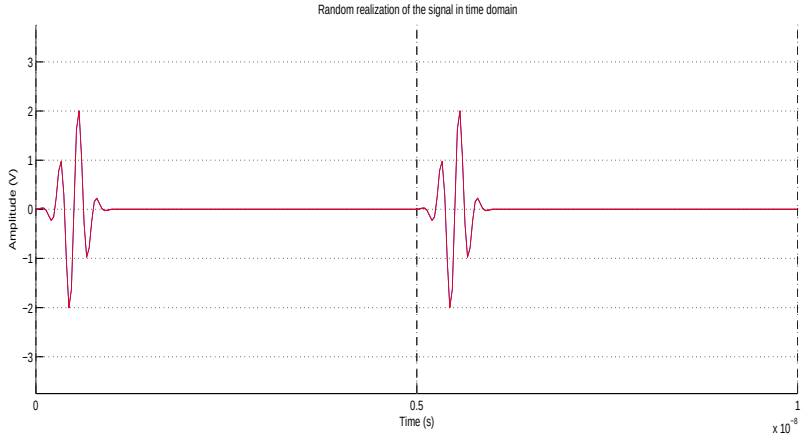

Figure 12: PulseSpectra('filter','gws',1e-9,'time',2)

The default pulse width is $\frac{ChipTime}{2}$, but any other positive value can be specified by adding the corresponding numerical attribute immediately after the pulse shape.

In the second case, the definition of the impulse response can be any string representing a mathematical function of time variable t. The signal will be then evaluated over the array $t = \left[0 : \frac{PulseWidth}{TimeResolution} : PulseWidth\right]$. The function definition can also contain any local constant used in the body of the $PulseSpectra$ function. For example, $PulseWidth$ can be useful.

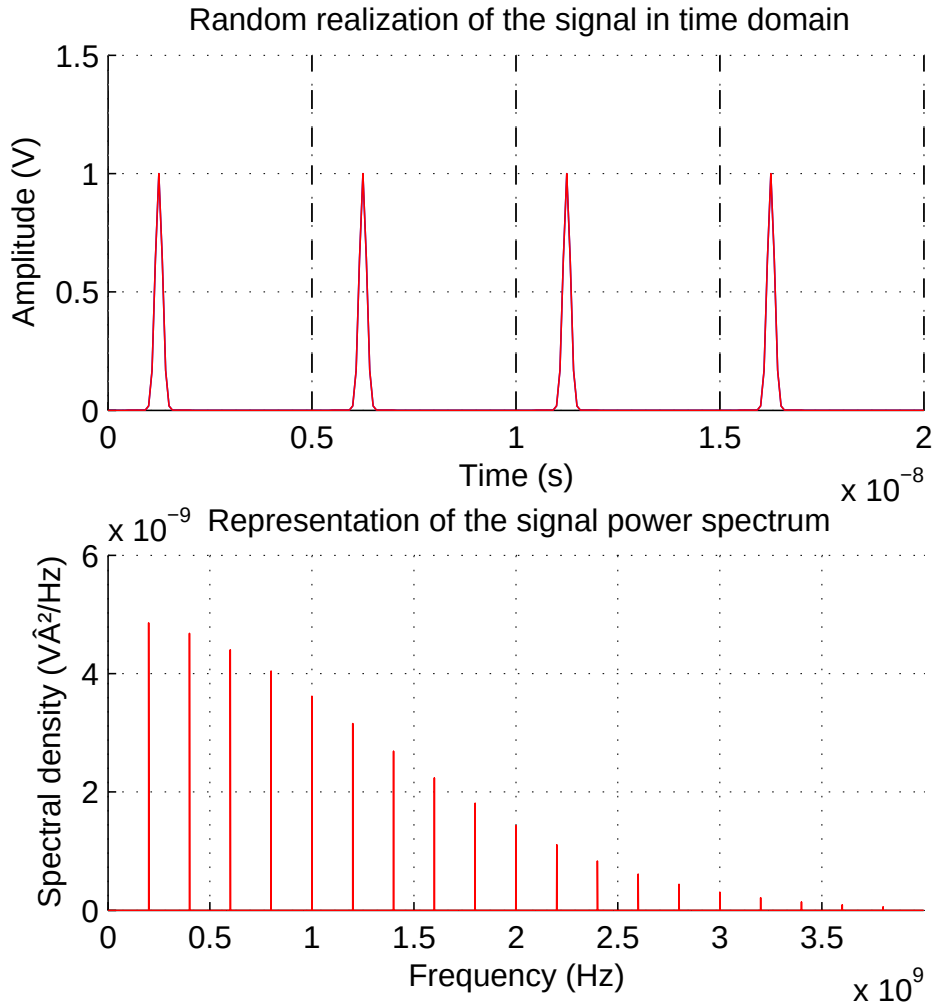

Figure 13: PulseSpectra('filter','exp(-((t-PulseWidth/2)/(PulseWidth/20)).^{2})','time')

3.4 Modeling random altering effects

3.4.1 Jittering

Jittering is the common term for describing random displacement of time events. If present, the keyword '*jitter*' adds random displacements of the pulses. A first optional attribute may follow under the shape of a vector specifying the different possible values, in seconds, for displacement. In case this attribute would be missing, the default setting is $[10, 8, 6, 4, 2, 0, 2, 4, 6, 8, 10] \times 10^{-12}$. If that first optional vector is present, a second one, with the same length, may then specify the probabilies of those different displacements, order-respectively. If absent, the default setting is the uniform distribution. If the sum of the specified probabilities does not equal 1, the vector is automatically normalized.

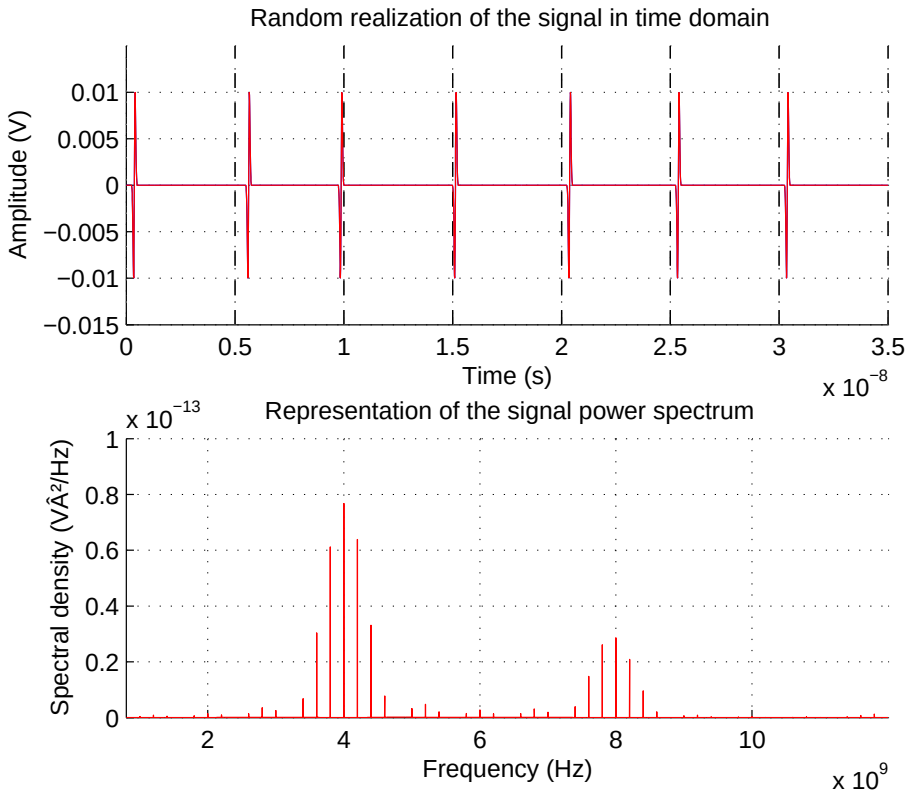

Figure 14: PulseSpectra('filter','gdr','jitter',[-5,-2.5,0,2.5,5]*1e-10,'time',7)

3.4.2 Thinning

If present, the keyword '*loss*' adds random losses to the process with loss probablity 0.1. Any different value may be specified by adding the corresponding number as an optional attribute immediately after.

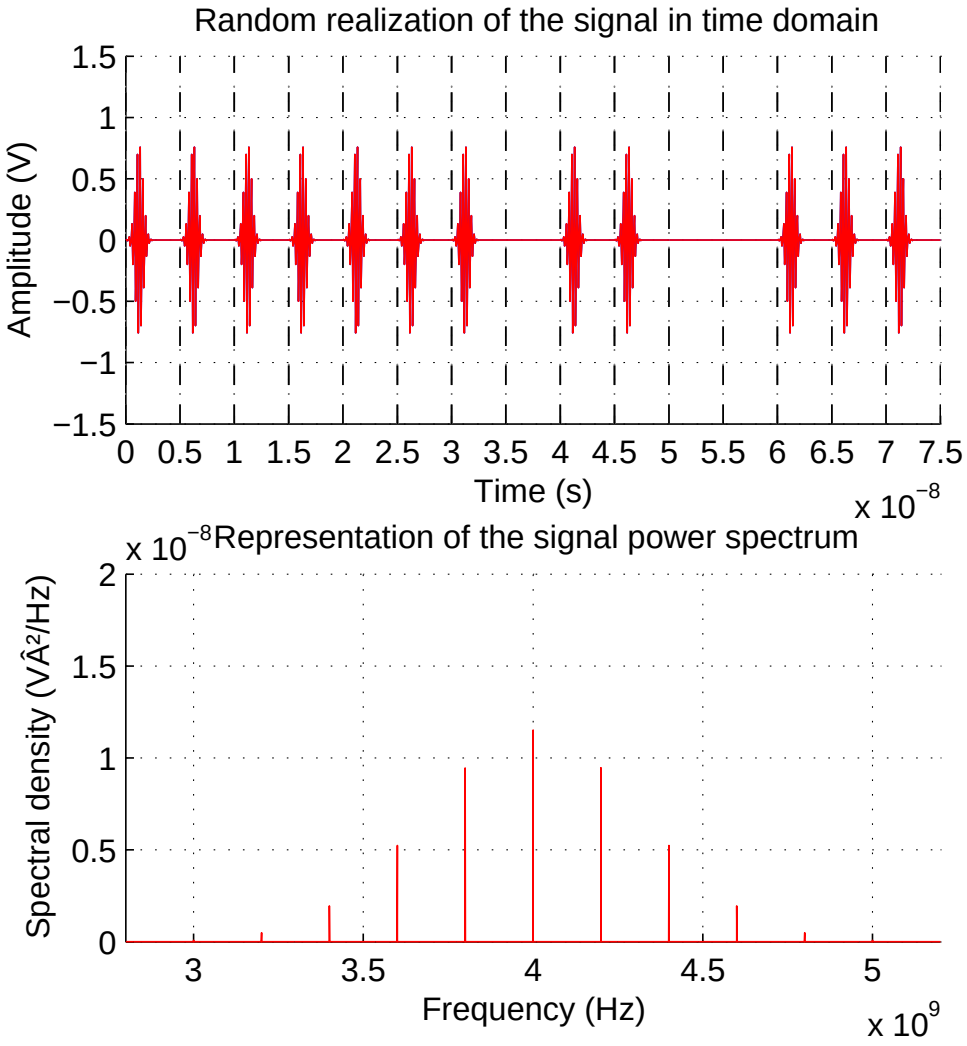

Figure 15: PulseSpectra('filter','gws','loss',0.15,'time',15)

3.4.3 Multipath fading

The multipath effects consist of attenuated repetitions of the transmitted pulse due to reflections by surrounding objects. If present the keyword '*multipath*' adds multipath fading effect according to the Saleh and Valenzuela double poisson model with deterministic exponential attenuation functions. Optionally, a set of four attributes may follow to specify, in order, the mean arrival rate of principal reflections (default : $\frac{1}{3\times10^{-7}}$, i.e. one every 300 ns), the caracteristic attenuation time of principal reflections (default : 60 ns), the mean arrival rate of secondary reflections (default : $\frac{1}{5\times10-9}$, i.e. one every 5 ns) and the caracteristic attenuation time of secondary reflections (default : 20 ns). Default values correspond to the Saleh and Valenzuela experimental measures presented in "A Statistical Model for Indoor Multipath Propagation, IEEE, 1987".

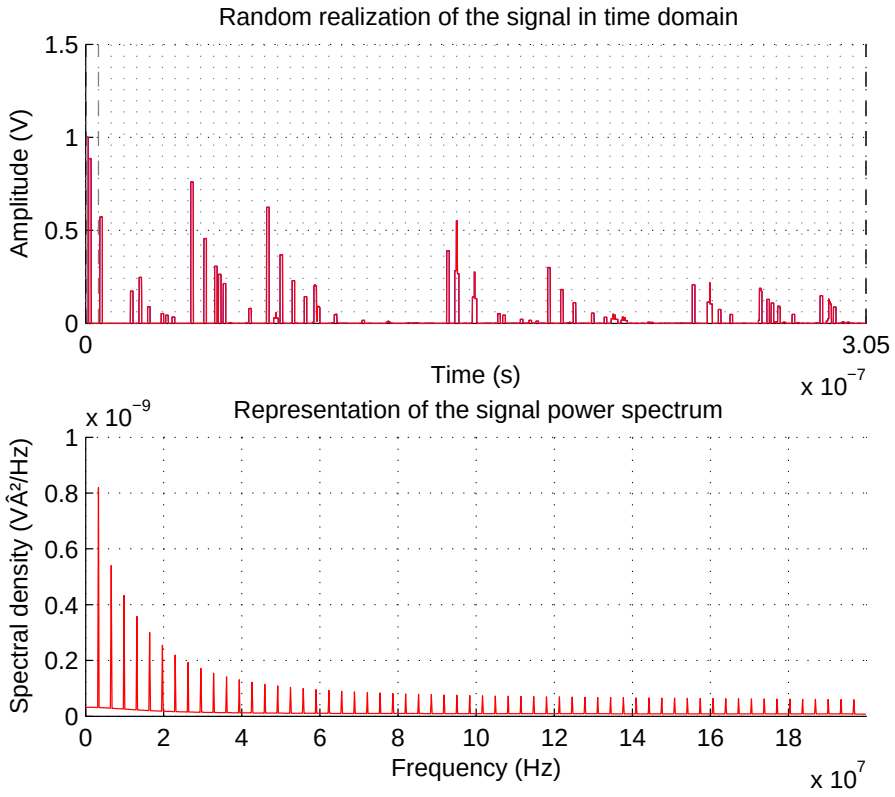

Figure 16: PulseSpectra('filter','rec',1e-9,'multipath',1/(85e-9),150e-9,1/(5e-9),10e-9,'time',1,'guardsize',60,'freq',[0;0.5e8])

3.5 Specifying display parameters

3.5.1 Frequency domain

The keyword '$freq$', immediately followed by a vector of 2 numbers, sets the lower and higher bounds of the displaid frequency domain to the specified values, in Hz. If absent, the program tries to automatically adjust the lower and higher bounds according to the signal frequency band so that at least 99% of total energy is displaid.

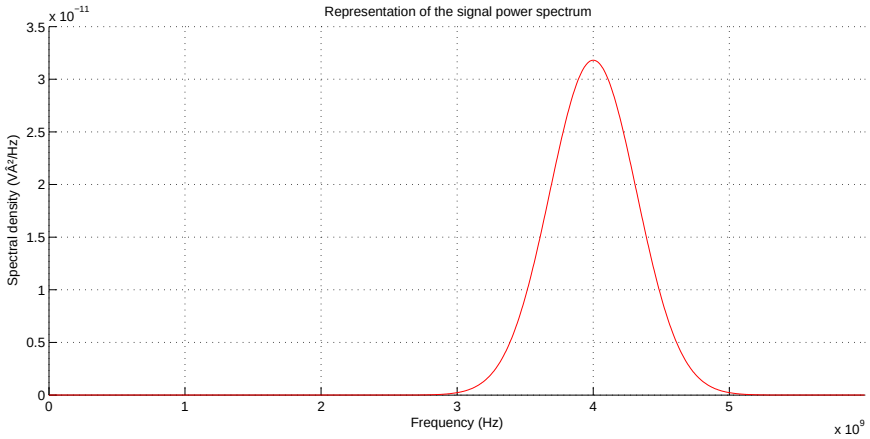

Figure 17: PulseSpectra('filter','gws','pam','freq',[0,6e9])

3.5.2 Frequency resolution

The keyword '$freqres$', immediately followed by a positive integer, sets the resolution in frequency domain (i.e. the number of points being considered within each $\frac{1}{SymbPeriod}$) to the specified value. If absent, the program automatically adjust the $FreqResolution$ according to the other parameters.

3.5.3 Time resolution

The keyword '$timeres$', immediately followed by a positive integer, sets the resolution in time domain (i.e. the number of points being considered within each $PulseWidth$) to the specified value. If absent, default value is at least 25 but can be automatically increased (in order to avoid spectrum periodisation due to the discrete Fourier transform) if the requested frequency display domain is too wide.

3.5.4 Random realization in time domain

In presence of the keyword '*time*', a randomly generated time realization of the signal over 4 symbol periods is also plotted. The number of symbol periods can be customized by simply adding the corresponding positive integer immediately after.

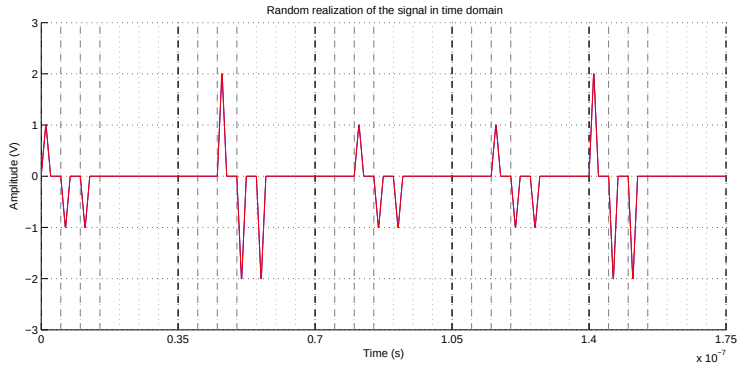

Figure 18: PulseSpectra('filter','tri','ds',[1,-1,-1],'pam',[1,2],'ppm','time',5)

3.5.5 Y-axis unit

In presence of the keyword '*dbm*', the power spectral measure is represented in $dBm.Hz^{-1}$ (i.e $10 \times log(\frac{P}{P_{1mW}})$) instead of $V^2.Hz^{-1}$.

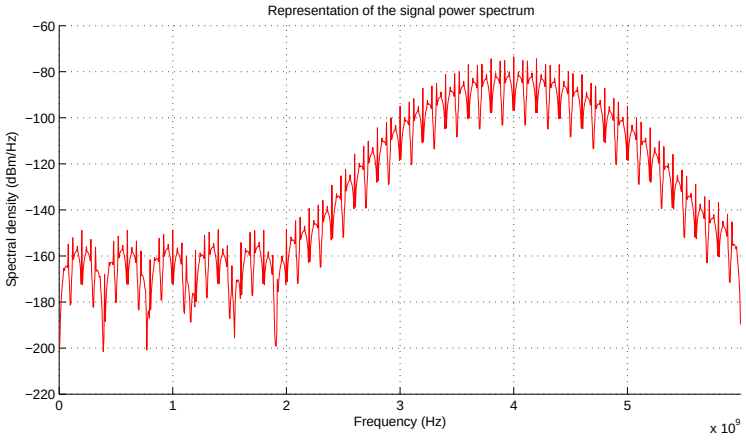

Figure 19: PulseSpectra('filter','gws','ppm','dbm','freq',[0,6e9])

3.5.6 Dual plot

In presence of the keyword '*dualplot*', the dirac part and the density part of the signal power spectral measure are plotted separately (instead of being added together by approximating each dirac with a triangle of appropriate surface).

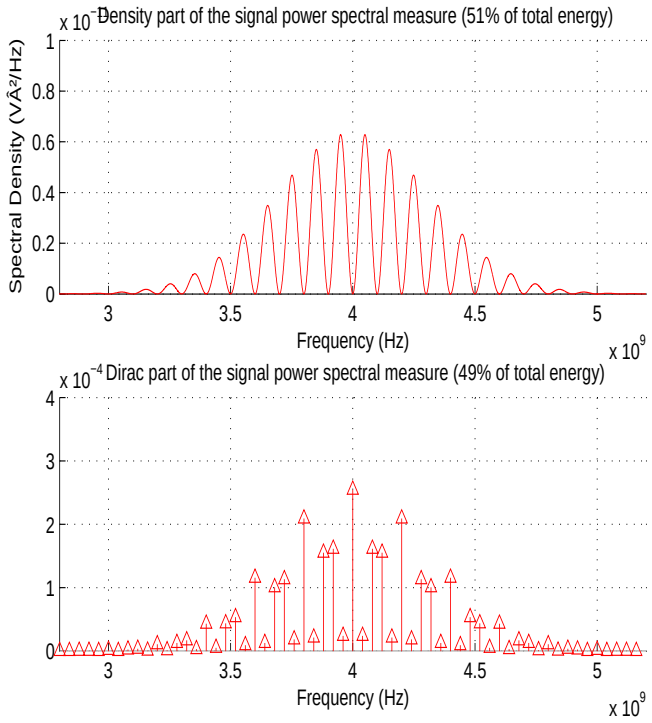

Figure 20: PulseSpectra('filter','gws','ppm','dualplot')

3.5.7 Font size

The keyword '$fontsize$', immediately followed by a positive integer, sets the font size of plot axes and titles to the specified value. Default is 10.

3.5.8 Color

The keyword '*color*', immediately followed by a 3 elements vector, sets the plot color to the specified RGB value. Default is red [1,0,0].