Previous topic
gnuradio.blocks
Next topic
gnuradio.comedi
This Page
Show Source
Quick search
Go
Enter search terms or a module, class or function name.

gnuradio.channels

Blocks for channel models and related functions.

gnuradio.channels.**channel_model**(*double noise_voltage=0.0*, *double frequency_offset=0.0*, *double epsilon=1.0*, *pmt_vector_cfloat taps*, *double noise_seed=0*, *bool block_tags=False*) → channel_model_sptr

Basic channel simulator.

This block implements a basic channel model simulator that can be used to help evaluate, design, and test various signals, waveforms, and algorithms.

This model allows the user to set the voltage of an AWGN noise source (), a (normalized) frequency offset (), a sample timing offset (), and a seed () to randomize or make reproducible the AWGN noise source.

Multipath can be approximated in this model by using a FIR filter representation of a multipath delay profile with the parameter .

To simulate a channel with time-variant channel, use gr::channels::channel_model2.

Constructor Specific Documentation:

Build the channel simulator.

Parameters:
- **noise_voltage** – The AWGN noise level as a voltage (to be calculated externally to meet, say, a desired SNR).
- **frequency_offset** – The normalized frequency offset. 0 is no offset; 0.25 would be, for a digital modem, one quarter of the symbol rate.
- **epsilon** – The sample timing offset to emulate the different rates between the sample clocks of the transmitter and receiver. 1.0 is no difference.
- **taps** – Taps of a FIR filter to emulate a multipath delay profile.
- **noise_seed** – A random number generator seed for the noise source.
- **block_tags** – If true, tags will not be able to propagate through this block.

channel_model_sptr.**frequency_offset**(*channel_model_sptr self*) → double

channel_model_sptr.**message_subscribers**(*channel_model_sptr self*, *swig_int_ptr which_port*) → swig_int_ptr

channel_model_sptr.**noise_voltage**(*channel_model_sptr self*) → double

channel_model_sptr.**set_frequency_offset**(*channel_model_sptr self*, *double frequency_offset*)

channel_model_sptr.**set_noise_voltage**(*channel_model_sptr self*, *double noise_voltage*)

channel_model_sptr.**set_taps**(*channel_model_sptr self*, *pmt_vector_cfloat taps*)

channel_model_sptr.**set_timing_offset**(*channel_model_sptr self*, *double epsilon*)

channel_model_sptr.**taps**(*channel_model_sptr self*) → pmt_vector_cfloat

channel_model_sptr.**timing_offset**(*channel_model_sptr self*) → double

gnuradio.channels.**channel_model2**(*double noise_voltage=0.0*, *double epsilon=1.0*, *pmt_vector_cfloat taps*, *double noise_seed=0*, *bool block_tags=False*) → channel_model2_sptr

Basic channel simulator allowing time-varying frequency and timing inputs.

This block implements a basic channel model simulator that can be used to help evaluate, design, and test various signals, waveforms, and algorithms.

This model allows the user to set the voltage of an AWGN noise source (), an initial timing offset (), and a seed () to randomize the AWGN noise source.

Multipath can be approximated in this model by using a FIR filter representation of a multipath

delay profile with the parameter .

Unlike gr::channels::channel_model, this block is designed to enable time-varying frequency and timing offsets.

Since the models for frequency and timing offset may vary and what we are trying to model may be different for different simulations, we provide the time-varying nature as an input function that is user-defined. If only constant frequency and timing offsets are required, it is easier and less expensive to use gr::channels::channel_model.

Constructor Specific Documentation:

Build the channel simulator.

Parameters:
- **noise_voltage** – The AWGN noise level as a voltage (to be calculated externally to meet, say, a desired SNR).
- **epsilon** – The initial sample timing offset to emulate the different rates between the sample clocks of the transmitter and receiver. 1.0 is no difference.
- **taps** – Taps of a FIR filter to emulate a multipath delay profile.
- **noise_seed** – A random number generator seed for the noise source.
- **block_tags** – If true, tags will not be able to propagate through this block.

channel_model2_sptr.**message_subscribers**(*channel_model2_sptr self*, *swig_int_ptr which_port*) → swig_int_ptr

channel_model2_sptr.**noise_voltage**(*channel_model2_sptr self*) → double

channel_model2_sptr.**set_noise_voltage**(*channel_model2_sptr self*, *double noise_voltage*)

channel_model2_sptr.**set_taps**(*channel_model2_sptr self*, *pmt_vector_cfloat taps*)

channel_model2_sptr.**set_timing_offset**(*channel_model2_sptr self*, *double epsilon*)

channel_model2_sptr.**taps**(*channel_model2_sptr self*) → pmt_vector_cfloat

channel_model2_sptr.**timing_offset**(*channel_model2_sptr self*) → double

gnuradio.channels.**fading_model**(*unsigned int N*, *float fDTs=0.01*, *bool LOS=True*, *float K=4*, *int seed=0*) → fading_model_sptr

fading simulator

This block implements a basic fading model simulator that can be used to help evaluate, design, and test various signals, waveforms, and algorithms.

Constructor Specific Documentation:

Build the channel simulator.

Parameters:
- **N** – the number of sinusoids to use in simulating the channel; 8 is a good value
- **fDTs** – normalized maximum Doppler frequency, fD * Ts
- **LOS** – include Line-of-Site path? selects between Rayleigh (NLOS) and Rician (LOS) models
- **K** – Rician factor (ratio of the specular power to the scattered power)
- **seed** – a random number to seed the noise generators

fading_model_sptr.**K**(*fading_model_sptr self*) → float

fading_model_sptr.**active_thread_priority**(*fading_model_sptr self*) → int

fading_model_sptr.**declare_sample_delay**(*fading_model_sptr self*, *int which*, *int delay*)

declare_sample_delay(fading_model_sptr self, unsigned int delay)

fading_model_sptr.**fDTs**(*fading_model_sptr self*) → float

fading_model_sptr.**message_subscribers**(*fading_model_sptr self*, *swig_int_ptr which_port*) → swig_int_ptr

fading_model_sptr.**min_noutput_items**(*fading_model_sptr self*) → int

fading_model_sptr.**pc_input_buffers_full_avg**(*fading_model_sptr self*, *int which*) → float

pc_input_buffers_full_avg(fading_model_sptr self) -> pmt_vector_float

fading_model_sptr.**pc_noutput_items_avg**(*fading_model_sptr self*) → float

fading_model_sptr.**pc_nproduced_avg**(*fading_model_sptr self*) → float

fading_model_sptr.**pc_output_buffers_full_avg**(*fading_model_sptr self*, *int which*) → float

pc_output_buffers_full_avg(fading_model_sptr self) -> pmt_vector_float

fading_model_sptr.**pc_throughput_avg**(*fading_model_sptr self*) → float

fading_model_sptr.**pc_work_time_avg**(*fading_model_sptr self*) → float

fading_model_sptr.**pc_work_time_total**(*fading_model_sptr self*) → float

fading_model_sptr.**sample_delay**(*fading_model_sptr self*, *int which*) → unsigned int

fading_model_sptr.**set_K**(*fading_model_sptr self*, *float K*)

fading_model_sptr.**set_fDTs**(*fading_model_sptr self*, *float fDTs*)

fading_model_sptr.**set_min_noutput_items**(*fading_model_sptr self*, *int m*)

fading_model_sptr.**set_step**(*fading_model_sptr self*, *float step*)

fading_model_sptr.**set_thread_priority**(*fading_model_sptr self*, *int priority*) → int

fading_model_sptr.**step**(*fading_model_sptr self*) → float

fading_model_sptr.**thread_priority**(*fading_model_sptr self*) → int

gnuradio.channels.**selective_fading_model**(*unsigned int N*, *float fDTs*, *bool LOS*, *float K*, *int seed*, *pmt_vector_float delays*, *pmt_vector_float mags*, *int ntaps*) → selective_fading_model_sptr

fading simulator

This block implements a basic fading model simulator that can be used to help evaluate, design, and test various signals, waveforms, and algorithms.

Constructor Specific Documentation:

Build the channel simulator.

Parameters:

- **N** – the number of sinusoids to use in simulating the channel; 8 is a good value
- **fDTs** – normalized maximum Doppler frequency, fD * Ts
- **LOS** – include Line-of-Site path? selects between Rayleigh (NLOS) and Rician (LOS) models
- **K** – Rician factor (ratio of the specular power to the scattered power)
- **seed** – a random number to seed the noise generators
- **delays** – a vector of values that specify the time delay of each impulse
- **mags** – a vector of values that specify the magnitude of each impulse
- **ntaps** – the number of filter taps

selective_fading_model_sptr.**K**(*selective_fading_model_sptr self*) → float

selective_fading_model_sptr.**active_thread_priority**(*selective_fading_model_sptr self*) → int

selective_fading_model_sptr.**declare_sample_delay**(*selective_fading_model_sptr self*, *int which*, *int delay*)

declare_sample_delay(selective_fading_model_sptr self, unsigned int delay)

selective_fading_model_sptr.**fDTs**(*selective_fading_model_sptr self*) → float

selective_fading_model_sptr.**message_subscribers**(*selective_fading_model_sptr self*, *swig_int_ptr which_port*) → swig_int_ptr

selective_fading_model_sptr.**min_noutput_items**(*selective_fading_model_sptr self*) → int

selective_fading_model_sptr.**pc_input_buffers_full_avg**(*selective_fading_model_sptr self*, *int which*) → float

pc_input_buffers_full_avg(selective_fading_model_sptr self) -> pmt_vector_float

selective_fading_model_sptr.**pc_noutput_items_avg**(*selective_fading_model_sptr self*) → float

selective_fading_model_sptr.**pc_nproduced_avg**(*selective_fading_model_sptr self*) → float

selective_fading_model_sptr.**pc_output_buffers_full_avg**(*selective_fading_model_sptr self*, *int which*) → float

pc_output_buffers_full_avg(selective_fading_model_sptr self) -> pmt_vector_float

selective_fading_model_sptr.**pc_throughput_avg**(*selective_fading_model_sptr self*) → float

selective_fading_model_sptr.**pc_work_time_avg**(*selective_fading_model_sptr self*) → float

selective_fading_model_sptr.**pc_work_time_total**(*selective_fading_model_sptr self*) → float

selective_fading_model_sptr.**sample_delay**(*selective_fading_model_sptr self*, *int which*) → unsigned int

selective_fading_model_sptr.**set_K**(*selective_fading_model_sptr self*, *float K*)

selective_fading_model_sptr.**set_fDTs**(*selective_fading_model_sptr self*, *float fDTs*)

selective_fading_model_sptr.**set_min_noutput_items**(*selective_fading_model_sptr self*, *int m*)

selective_fading_model_sptr.**set_step**(*selective_fading_model_sptr self*, *float step*)

selective_fading_model_sptr.**set_thread_priority**(*selective_fading_model_sptr self*, *int priority*) → int

selective_fading_model_sptr.**step**(*selective_fading_model_sptr self*) → float

selective_fading_model_sptr.**thread_priority**(*selective_fading_model_sptr self*) → int

gnuradio.channels.**dynamic_channel_model**(*double samp_rate*, *double sro_std_dev*, *double sro_max_dev*, *double cfo_std_dev*, *double cfo_max_dev*, *unsigned int N*, *double doppler_freq*, *bool LOS_model*, *float K*, *pmt_vector_float delays*, *pmt_vector_float mags*, *int ntaps_mpath*, *double noise_amp*, *double noise_seed*) → dynamic_channel_model_sptr

dynamic channel simulator

This block implements a dynamic channel model simulator that can be used to help evaluate, design, and test various signals, waveforms, and algorithms.

This model allows the user to set up an AWGN noise cource, a random walk process to simulate carrier frequency drift, a random walk process to simulate sample rate offset drive, and a frequency selective fading channel response that is either Rayleigh or Ricean for a user specified power delay profile.

Constructor Specific Documentation:

Build the dynamic channel simulator.

Parameters:

- **samp_rate** – Input sample rate in Hz
- **sro_std_dev** – sample rate drift process standard deviation per sample in Hz
- **sro_max_dev** – maximum sample rate offset in Hz
- **cfo_std_dev** – carrier frequnecy drift process standard deviation per sample in Hz
- **cfo_max_dev** – maximum carrier frequency offset in Hz
- **N** – number of sinusoids used in frequency selective fading simulation
- **doppler_freq** – maximum doppler frequency used in fading simulation in Hz
- **LOS_model** – defines whether the fading model should include a line of site component. LOS->Rician, NLOS->Rayleigh
- **K** – Rician K-factor, the ratio of specular to diffuse power in the model
- **delays** – A list of fractional sample delays making up the power delay profile
- **mags** – A list of magnitudes corresponding to each delay time in the power delay profile
- **ntaps_mpath** – The length of the filter to interpolate the power delay profile over. Delays in the PDP must lie between 0 and ntaps_mpath, fractional delays will be sinc-interpolated only to the width of this filter.
- **noise_amp** – Specifies the standard deviation of the AWGN process
- **noise_seed** – A random number generator seed for the noise source.

dynamic_channel_model_sptr.**K**(*dynamic_channel_model_sptr self*) → double

dynamic_channel_model_sptr.**cfo_dev_max**(*dynamic_channel_model_sptr self*) → double

dynamic_channel_model_sptr.**cfo_dev_std**(*dynamic_channel_model_sptr self*) → double

dynamic_channel_model_sptr.**doppler_freq**(*dynamic_channel_model_sptr self*) → double

dynamic_channel_model_sptr.**message_subscribers**(*dynamic_channel_model_sptr self*, *swig_int_ptr which_port*) → swig_int_ptr

dynamic_channel_model_sptr.**noise_amp**(*dynamic_channel_model_sptr self*) → double

dynamic_channel_model_sptr.**samp_rate**(*dynamic_channel_model_sptr self*) → double

dynamic_channel_model_sptr.**set_K**(*dynamic_channel_model_sptr self*, *double arg2*)

dynamic_channel_model_sptr.**set_cfo_dev_max**(*dynamic_channel_model_sptr self*, *double arg2*)

dynamic_channel_model_sptr.**set_cfo_dev_std**(*dynamic_channel_model_sptr self*, *double arg2*)

dynamic_channel_model_sptr.**set_doppler_freq**(*dynamic_channel_model_sptr self*, *double arg2*)

dynamic_channel_model_sptr.**set_noise_amp**(*dynamic_channel_model_sptr self*, *double arg2*)

dynamic_channel_model_sptr.**set_samp_rate**(*dynamic_channel_model_sptr self*, *double arg2*)

dynamic_channel_model_sptr.**set_sro_dev_max**(*dynamic_channel_model_sptr self*, *double arg2*)

dynamic_channel_model_sptr.**set_sro_dev_std**(*dynamic_channel_model_sptr self*, *double arg2*)

dynamic_channel_model_sptr.**sro_dev_max**(*dynamic_channel_model_sptr self*) → double

dynamic_channel_model_sptr.**sro_dev_std**(*dynamic_channel_model_sptr self*) → double

gnuradio.channels.**cfo_model**(*double sample_rate_hz*, *double std_dev_hz*, *double max_dev_hz*, *double noise_seed=0*) → cfo_model_sptr

channel simulator

This block implements a carrier frequency offset model that can be used to simulate carrier frequency drift typically from mixer LO drift on either transmit or receive hardware.

A clipped gaussian random walk process is used.

Constructor Specific Documentation:

Build the carrier frequency offset model.

Parameters:
- **sample_rate_hz** – Sample rate of the input signal in Hz
- **std_dev_hz** – Desired standard deviation of the random walk process every sample in Hz
- **max_dev_hz** – Maximum carrier frequency deviation in Hz.
- **noise_seed** – A random number generator seed for the noise source.

cfo_model_sptr.**active_thread_priority**(*cfo_model_sptr self*) → int

cfo_model_sptr.**declare_sample_delay**(*cfo_model_sptr self*, *int which*, *int delay*)
declare_sample_delay(cfo_model_sptr self, unsigned int delay)

cfo_model_sptr.**max_dev**(*cfo_model_sptr self*) → double

cfo_model_sptr.**message_subscribers**(*cfo_model_sptr self*, *swig_int_ptr which_port*) → swig_int_ptr

cfo_model_sptr.**min_noutput_items**(*cfo_model_sptr self*) → int

cfo_model_sptr.**pc_input_buffers_full_avg**(*cfo_model_sptr self*, *int which*) → float
pc_input_buffers_full_avg(cfo_model_sptr self) -> pmt_vector_float

cfo_model_sptr.**pc_noutput_items_avg**(*cfo_model_sptr self*) → float

cfo_model_sptr.**pc_nproduced_avg**(*cfo_model_sptr self*) → float

cfo_model_sptr.**pc_output_buffers_full_avg**(*cfo_model_sptr self*, *int which*) → float
pc_output_buffers_full_avg(cfo_model_sptr self) -> pmt_vector_float

cfo_model_sptr.**pc_throughput_avg**(*cfo_model_sptr self*) → float

cfo_model_sptr.**pc_work_time_avg**(*cfo_model_sptr self*) → float

cfo_model_sptr.**pc_work_time_total**(*cfo_model_sptr self*) → float

cfo_model_sptr.**samp_rate**(*cfo_model_sptr self*) → double

cfo_model_sptr.**sample_delay**(*cfo_model_sptr self*, *int which*) → unsigned int

cfo_model_sptr.**set_max_dev**(*cfo_model_sptr self*, *double _dev*)

cfo_model_sptr.**set_min_noutput_items**(*cfo_model_sptr self*, *int m*)

cfo_model_sptr.**set_samp_rate**(*cfo_model_sptr self*, *double _rate*)

cfo_model_sptr.**set_std_dev**(*cfo_model_sptr self*, *double _dev*)

cfo_model_sptr.**set_thread_priority**(*cfo_model_sptr self*, *int priority*) → int

cfo_model_sptr.**std_dev**(*cfo_model_sptr self*) → double

cfo_model_sptr.**thread_priority**(*cfo_model_sptr self*) → int

gnuradio.channels.**sro_model**(*double sample_rate_hz*, *double std_dev_hz*, *double max_dev_hz*, *double noise_seed=0*) → sro_model_sptr
Sample Rate Offset Model.

This block implements a model that varies sample rate offset with respect to time by performing a random walk on the interpolation rate.

Constructor Specific Documentation:

Build the sample rate offset model.

Parameters:
- **sample_rate_hz** – Sample rate of the input signal in Hz
- **std_dev_hz** – Desired standard deviation of the random walk process every sample in Hz
- **max_dev_hz** – Maximum sample rate deviation from zero in Hz.
- **noise_seed** – A random number generator seed for the noise source.

sro_model_sptr.**active_thread_priority**(*sro_model_sptr self*) → int

sro_model_sptr.**declare_sample_delay**(*sro_model_sptr self*, *int which*, *int delay*)

declare_sample_delay(sro_model_sptr self, unsigned int delay)

sro_model_sptr.**max_dev**(*sro_model_sptr self*) → double

sro_model_sptr.**message_subscribers**(*sro_model_sptr self*, *swig_int_ptr which_port*) → swig_int_ptr

sro_model_sptr.**min_noutput_items**(*sro_model_sptr self*) → int

sro_model_sptr.**pc_input_buffers_full_avg**(*sro_model_sptr self*, *int which*) → float

pc_input_buffers_full_avg(sro_model_sptr self) -> pmt_vector_float

sro_model_sptr.**pc_noutput_items_avg**(*sro_model_sptr self*) → float

sro_model_sptr.**pc_nproduced_avg**(*sro_model_sptr self*) → float

sro_model_sptr.**pc_output_buffers_full_avg**(*sro_model_sptr self*, *int which*) → float

pc_output_buffers_full_avg(sro_model_sptr self) -> pmt_vector_float

sro_model_sptr.**pc_throughput_avg**(*sro_model_sptr self*) → float

sro_model_sptr.**pc_work_time_avg**(*sro_model_sptr self*) → float

sro_model_sptr.**pc_work_time_total**(*sro_model_sptr self*) → float

sro_model_sptr.**samp_rate**(*sro_model_sptr self*) → double

sro_model_sptr.**sample_delay**(*sro_model_sptr self*, *int which*) → unsigned int

sro_model_sptr.**set_max_dev**(*sro_model_sptr self*, *double _dev*)

sro_model_sptr.**set_min_noutput_items**(*sro_model_sptr self*, *int m*)

sro_model_sptr.**set_samp_rate**(*sro_model_sptr self*, *double _rate*)

sro_model_sptr.**set_std_dev**(*sro_model_sptr self*, *double _dev*)

sro_model_sptr.**set_thread_priority**(*sro_model_sptr self*, *int priority*) → int

sro_model_sptr.**std_dev**(*sro_model_sptr self*) → double

sro_model_sptr.**thread_priority**(*sro_model_sptr self*) → int