Manual for the Noisy Spike Generator MATLAB Software: version 1.2, released 27 July 2017 Minor update from version 1.1

Leslie S. Smith and Nhamoinesu Mtetwa
Department of Computing Science and Mathematics
University of Stirling
Stirling FK9 4LA, Scotland, UK
contact: lss@cs.stir.ac.uk

1 Introduction

This tool is used to generate noisy spike trains (perhaps better described as spike trains with interference) to simulate the kind of signals that an extracellular electrode (such as those on a multi-electrode array) record from a neural culture. This document is the user manual for the software: a separate document describing the analysis underpinning this is also available [2]. The tool is intended for use in testing spike detection and spike sorting algorithms: it provides a method of generating realistic spike trains for which the "ground truth" is known. It can also be used simply to generate background noise or interference of the sort that might be encountered by an extracellular electrode.

The software generates signals from a single electrode. It can allow for

- a number of neurons at different distances from the electrode, and with different neuron/electrode geometries, some generating correlated spikes, and some generating uncorrelated spikes
- the electrode being partially covered by glial cells

The signal that a real electrode records is a summation of intracellular spikes from many neurons, arising from ionic flow over the surface of these membranes. This signal is thus the result of the spatiotemporal distribution of the spikes on the spiking surface of the neuron, and of the path that the signal takes to the electrode. For the purposes of this tool, neurons in the "culture" are divided into target neurons (which are those that one might be hoping to detect and sort), correlated neurons (whose spikes are correlated with one of the target neurons, but which are to be considered as correlated interference), and uncorrelated neurons (which are intended to provide uncorrelated interference). The number of each type can be set either in the program or from the invoking MATLAB command.

Since the electrode is extracellular, we assume that the signal that it picks from any patch of the surface of a neuron will be a summation of a number of forms of transformation (discussed in detail in the report document, [2]). Signals crossing the neuron's membrane undergo differentiation due to the capacitance of the membrane, and also due to the mechanism of spike (re)generation. Further differentiation can occur because of the nature of signal transfer from the extracellular electrolyte to the electrode, particularly if the electrode is covered by insulating glial cells. But it is also the case that some signal might arrive directly, having undergone a primarily resistive path from intracellular signal to electrode. Additionally, there is the possibility that there might be other glial cells in the path from intracellular signal to electrode (for example a layer of glial cells covering electrodes at the bottom a an MEA culture dish), resulting in further differentiation. What is actually recorded at the electrode is the sum of all of these, weighted by their size, and integrated over the neural surface for many neurons.

Importantly, the signal generated by a spiking neuron does not all occur at the same time. In a real neuron, the spike is initiated at the axon hillock, then travels down the (probably branching) axon. The transmission speed depends on a number of factors: axon diameter and myelinisation in particular. What matters here is that spikes produced as a result of a single axon hillock action potential are present in the neuron for a relatively long time compared to the duration of the spike itself (with the default parameters this time can be up to 1.8 milliseconds, but this can easily be altered). Further, particularly for neurons which are reasonably close to the electrode, some parts of the structure of the spiking part of the neuron are likely to be much closer to the electrode than others. As a result, the shape of a signal received for a single spike from a single neuron is likely to be characteristic of that neuron, permitting spike sorting to occur. We model this by adding up weighted delayed versions of the signal that might be expected at the electrode from a single intracellular spike at the axon hillock. (In fact, we compute weighted summed versions of the intracellular potential, then use this and its derivatives (discussed above), relying on the linearity of summation and differentiation.)

Lastly, we can add some (Gaussian) noise, to mimic the effect of thermal and amplifier noise.

As the program stands, adjusting the values of the parameters can be achieved partly from the command line, but may also require going into the program, and altering values inside (reasonably well commented) MATLAB m-files. The shape of the intracellular spike is held in a file (which can be written using HHSim [3]). The characteristics of the effect of the neuron surface/electrode geometry are held in a file for each neuron. This file can be produced either using the GUI of the supplied Mac OSX application PrepareParams, or using the m-file *spatiotemporal.m* (see section 3) or simply edited by hand. Even although the MATLAB m-files are quite well commented, there really ought to be a better user interface. One day.

2 The program

The main function (m-file) is *generatenoisysamples*. This function calls a number of functions (and reads some files) to produce a noisy spike train.

generatenoisysamples returns three variables, namely,

1. the noisy signal (a 1-dimensional array)
2. a structure containing amongst other things the actual (official!) spike times. These include both the peaks of the (intracellular) spikes and the spike times used to generate the spike train. Note that the spike times are the *starting times* of the spike templates, so that the actual peak will be some little time later. It also returns the clean signal from each target neuron (as would be received if there was no noise and no other target neurons). This structure can be used to generate new data sets which have the same primary spikes as a previous data set, but different noise levels etc: see section 2.3 for details of the structure.
3. a structure used by the MATLAB random number generator. Returning it allows the same random numbers to be generated the next time the function is called, if this is desired.

2.1 generatenoisysamples

The function may be called as follows:

```
[signals target r1] = generatenoisysamples('Duration', 0.2, 'SampleRate', 100000) ;
```

and this returns the samples as a 1 by N array (where $N = 0.2 \times 100000 = 20000$) in *signals*. Information about the target signals is returned in *target* (see section 2.3). *r1* returns the state of the random number generator. The number of each type of neuron, and all the other parameters are then taken from the .m file itself. The intracellular spike shapes are taken from files (see section 5), and the information about the spatiotemporal characteristics are also taken from files (see section 4 and 3).

Calling the function a second time using

```
[signals1 target1 r1] = ...
generatenoisysamples('ReuseTargets', target, 'Duration', 0.2, 'SampleRate', 100000) ;
```

results in the original spike times being kept, but different noise being added. Calling the function using

```
[signals2 target2 r1] =  ...
generatenoisysamples('ReuseRNGstate', r1, 'Duration', 0.2, 'SampleRate', 100000) ;
```

should result in exactly the same times and the same random numbers being generated. If no changes are made to the internal variables, this should generate exactly the same signal. Normally, of course, one makes some changes (for example to the mixing coefficients), so that one can control exactly what is generated. (Bug: currently, the signal generated is not identical.)

The function starts by reading in the characteristic delays. These may be read from a file (using the functions *target_temporal_mac* (for the target neurons), *correlated_temporal_mac* for the correlated neurons, and *uncorrelated_temporal_mac* for the uncorrelated neurons, or incorporated inside functions *target_temporal* (for the target neurons), *correlated_temporal* for the correlated neurons, and *uncorrelated_temporal* for the uncorrelated neurons (details of these functions a provided in section 4 below). Which happens depends on the command line argument *TemporalDataInFile*.

The function then initialises a host of internal variables, see section 6.2 for details. After all the parameters are initialised a number of spike templates are loaded from files. Next, spike times are created for each target neuron using either a Gaussian or Poisson distribution. The spike times are stored in the *target* structure. The number of target spike trains is set using *N_Targets* (an optional argument to the function: see below): this defaults to two (unless the value has been reset in the m-file file: note that all the defaults below can easily be altered).

Gaussian or Poisson distributed uncorrelated spike times are created using the functions *genspikesgaussion* or *genspikespoisson*. The next function to be called is *gensampledspikes*. This generates sampled spikes from the spike times by resampling the appropriate template spike (see section 5) to make sure that the spikes are evenly sampled. The resulting signal is used to produce two other signals (once differentiated and twice differentiated) by calling the function *simplediff* which performs a differentiation operation on the signal. Differentiation also includes some smoothing to remove artefacts caused by inaccuracies in the original spike signal. (Note that further differentiation can be included by adding further calls to this function.) For both the original signal, and its two derivatives, the function *spatiotransform* is then called, and this adds in weighted delayed versions of the signal emulating the effect of the time dispersion of the spike. Lastly, these three signals are normalised to between -0.5 and +0.5 using the function *setlimits*.

A similar approach is taken to generating both the correlated and uncorrelated interference signals.

2.2 Command line arguments

generatenoisysamples can take a number of command line parameters, using the MATLAB *varargin* facility (thus to set the duration of the simulation one puts

```
'Duration', 0.3
```

in the command line). The command line parameters are

Duration This is the duration of the spike train in seconds. It defaults to 0.1 seconds.

SampleRate The number of samples/second. It defaults to 100000 samples/second

N_Targets The number of target neurons. Defaults to 2.

N_Jitter The number of jittered spike trains. Defaults to 7.

N_Uncorr The number of uncorrelated (independent) spike trains. Defaults to 15.

RefractoryPeriod The neuron refractory period (in seconds). Defaults to 0.001 seconds.

T_Delta_Integrate The single delay used in summing over the time that a spike takes throughout the neuron spiking surface. Defaults to 30μ seconds. Needs to be quite short for Nyquist reasons.

N_Delta_Integrate Number of delay times. Defaults to 60 (which gives a default spike duration of 1.8ms)

ReuseTargets Reuse the target structure (i.e. times, and all target characteristics): allows different sets of parameters to be used with the same target spiking times, useful for testing out the effects of different levels and forms of noise.

ReuseTargetTimes Reuse the target spiking times (i.e. just the times of the target neuron spikes). Can be used as above, and also to experiment with the effects of different delay characteristics.

ReuseRNGState Reuse the random number generator state. Should lead to exactly the same behaviour when the same random number calls are made (but currently has a bug).

DSLength Sets a smoothing parameter used in differentiation (*simplediff*): see section 2. Defaults to 60 (samples).

TemporalDataInFile Reports whether the data for delay characteristics is inside a MATLAB function (see section), or is to be read from file (default).

TemporalDirectory The name of the directory from which delay characteristics are to be read if *TemporalDataInFile* is true. Defaults to *example* inside the current directory.

ShowSNR A flag to determine if the Signal: Correlated Interference: Uncorrelated Interference should be displayed after data generation. Defaults to 0 (don't display).

Orig_Temporal_Supplied The time data for the target spikes is suppled (in the variable following), and is not to be read from a file or created by calling target_temporal

SameTargetSizes If set, then the different target neuron signals will all be stretched or compressed to a specific range, currently [-0.1 0.1].

Target1Weights The strengths for the components of the target 1 neuron are supplied in the following variable: this is a 3-tuple, being the strength of the original signal, of the differentiated signal, and of the twice differentiated signal.

2.3 The *target* structure returned

The target structure does not need to be accessed to use the software. However, it can provide useful information. The target structure returned is a 1 x *N_Targets* array of structures (struct array, in MATLAB terminology). There is one structure per target neuron. The structure fields are

targettimes The times of the start of each spike

sampletargets The intracellular spiking signal, prior to integration over the spiking surface of the neuron

stsamples The intracellular spiking signal, after integration over the spiking surface of the neuron

actualpeaks The actual peaks in the integrated intracellular spike signal

ssamples_1n Same as *stsamples* but normalised to be between the minimum and maximum value (-0.5 and +0.5).

ssamples_1dn Differentiated signal, normalised to be between the minimum and maximum value (-0.5 and +0.5).

ssamples_1ddn Double differentiated signal, normalised to be between the minimum and maximum value (-0.5 and +0.5).

final The signal received by the electrode from this neuron, without any noise added.

2.4 Jittered spikes

This is intended for modelling the effect of nearby neurons whose spike times are correlated with (but not identical to) one of the original spike times. The original spike times are jittered using *probjitter* which randomly shifts the position of each spike by a few samples to create a vector *jstimes* of jittered spike times. The jittered spike times are used to produce another membrane voltage signal based on a spike template (see section 5) with jittered spikes using the function *gensampledspikes*. As with the target neurons, the signal with jittered spikes is also differentiated twice using the function *simplediff* to produce jittered versions of once and twice differentiated spikes (again, further differentiation could easily be added), and the effect of the geometry of these neurons and the electrode implemented using *spatiotransform*. The differentiated signals are also normalised using the function *setlimits*. The number of jittered neurons used can be set by the user: the amount of jitter can be different for each.

2.5 Uncorrelated spikes

This is intended for modelling the effect of nearby neurons whose spike times are not correlated with the original spike times. Gaussian or Poisson distributed uncorrelated spike times are created using the functions *genspikesgaussion* or *genspikespoisson* and the uncorrelated spike times are stored in the vector *u_stimes*. These uncorrelated spike times are then used to create a membrane voltage signal based on a spike template (see section 5) with uncorrelated spikes using *gensampledspikes*. As before, two normalised differentiated versions of this signal are also created, and the effects of the delays in the neuron applied. The number of different neurons used can be set by the user.

2.6 Correlated spike noise

Correlated spike noise is created by linearly mixing the jittered spike train, once differentiated spike train and the twice differentiated jittered spike train created above. The mixing is done using coefficients in vectors namely *jmixcoeffs_orig*, *jmixcoeffs_d* and *jmixcoeffs_dd*. In addition, one can vary the overall size of each of these signals by using *jitteroverallsize_orig*, *jitteroverallsize_d*, and *jitteroverallsize_dd*, allowing the original, differentiated, and twice differentiated signals to be varied in size without adjusting each vector.

2.7 Uncorrelated spike noise

Uncorrelated spike noise is created by linearly mixing the uncorrelated spike train, once differentiated and the twice differentiated uncorrelated spike trains created above. The mixing is done using coefficients in vectors namely *umixcoeffs_orig*, *umixcoeffs_d* and *umixcoeffs_dd*. In addition, one can vary the overall size of each of these signals by using *u_overallsize_orig*, *u_overallsize_d*, and *u_overallsize_dd*, allowing the original, differentiated, and twice differentiated signals to be varied in size without adjusting each vector.

2.8 Final signal

The final signal is created in four stages:

1. the original signals are linearly mixed (using coefficients described in section 6.2 with their once and twice differentiated versions.

2. the result of this is linearly mixed with correlated and uncorrelated spike noise signals generated as above.

3. the result of the second mixture is corrupted by some additive Gaussian white noise of a chosen strength using the function *agwn*

4. the signal thus generated is linearly scaled to be between the final output values *fminval* and *fmaxval.*

3 Generating the spatiotemporal characteristics

The spatiotemporal characteristics can be read in from files. These files will be in the subdirectory specified in command line parameter *TemporalDirectory*, and have predefined names. for the target neurons, the filenames are 'target_temporal_*i*', where *i* starts at 0, and for the correlated neurons, the filenames are 'correlated_temporal_*i*', where *i* starts at 0. The files consist of tab-separated floating point numbers. The files can be created either using the m-file *makespatiotemparray*, or using the supplied Mac OSX application PrepareParams. The former stays within MATLAB, but has no GUI, and the latter need Mac OSX, and has a GUI. The PrepareParams application provides a GUI for producing the delay characteristics for the target and correlated neurons. It produces an array which is 3 by N (where N is the number of points). The array may then be written either in a form which can be read by the MATLAB function (use either the Save Params button, or the File: Save menu), or in a form which can be later incorporated into a MATLAB function (use the File:Save (Matlab) menu).

The user clicks in the view to set a value: this value is then linearly interpolated with other values (the start and end of the array is set to 0 initially). The user selects whether the values being set are for the original, first or second derivative using a radio button. The user can also select a multiplier for these values. However, only the sign of the value is eventually significant because the MATLAB programs scale the values.

It is also possible to read in a set of values using the File: Load from file menu. Note however, that these values cannot then be properly edited. This is, however, useful for displaying parameter files (whether created using PrepareParams or *makespatiotemparray*. Parameter file displays may also be saved as .pdf files (using File: Save as PDF image), and this can be useful for documentation.

4 MATLAB functions

We briefly describe below the other MATLAB functions used.

4.1 target_temporal_mac

Reads in the delay characteristic for the target neurons. The files are in the directory set using *TemporalDirectory*, and the filenames are 'target_temporal_*i*', where *i* starts at 0.

4.2 correlated_temporal_mac

Reads in the delay characteristic for the correlated neurons. The files are in the directory set using *TemporalDirectory*, and the filenames are 'correlated_temporal_*i*', where i starts at 0.

4.3 uncorrelated_temporal_mac

Provides the delay characteristics for the uncorrelated neurons. Currently this simply returns a matrix of 1's.

4.4 target_temporal

Used when *TemporalDataInFile* is false. Simply provides an array of the delay coefficients for the target neurons inside the function.

4.5 correlated_temporal

Used when *TemporalDataInFile* is false. Simply provides an array of the delay coefficients for the correlated neurons inside the function.

4.6 uncorrelated_temporal

Identical to *uncorrelated_temporal_mac*.

4.7 makespatiotemparray

This function takes a number of parameters: the first is the length (L of the spatiotemporal array. What it does is to take a set of times and values for the strengths (for the original signal, and its first and second derivatives), and using linear interpolation produce a 3 by L array. Where extrapolation would be necessary, the values are set to 0.

4.8 genspikespoisson

This generates a Poisson distributed spike train. Both λ and the mean inter-spike-interval are parameters. It is also possible to set the minimum inter-spike interval, using *varargin*. The keyword is *MinISI*.

4.9 genspikesgaussian

This generates a Gaussian distributed spike train. Both the mean inter-spike-interval and the standard deviation can be set. The minimum inter-spike interval is set as above.

4.10 gensampledspikes

This function takes in

1. a 1-d array of spike times in seconds
2. an array describing the intracellular spike template. This is a .csv (comma separated values) file, with each line describing one spike point: there may be many values per line, but only the first two are used. These are the time (in milliseconds), and the voltage (in millivolts). It produces from this a 1-d array of the membrane voltage at sample times with spikes placed at times specified in the input spike times 1-d array. The template spike is resampled to the required sample rate before being placed at spike position according to the predetermined spike times.
3. a variable number of other arguments:

   MinVal, MaxVal The minimum and maximum value of the signal to be produced. The signal will be linearly adjusted to be between these values.

   EndTime The duration of the signal to be produced

4.11 simplediff

This function performs a simple differential operation. It smooths the input signal first using a hamming window of a given length. The function returns a vector same length as input.

4.12 spatiotransform

Essentially convolves the delay coefficients with the signal. A little complexity arises because the inter-sample times differ, so that the delay coefficients need to be interpolated first.

4.13 probjitter

This function produces a new list of spike times from the old list. Each output spike is produced from each input spike with probability *probspike*, at a time determined from a gaussian distribution with mean 0 and standard deviation *jitterstd*.

4.14 setlimits

This function linearly scales the input values to be within the specified maximum and minimum values. The function basically normalises or stretches the spikes.

5 Spike Templates

The spike times are turned into intracellular voltages using spike templates. This allows different neurons to have different spike shapes. Each spike time is taken to be the start of a spike, and the appropriate spike template is added in. Where spike templates would overlap, they are not simply added. Instead, the first spike template sample with the value nearest to the current value is used as the start of the next spike. There are implicit assumptions in this: (i) spikes are spike-shaped, that is, the consist of an upswing, then a downswing (and probably a post-spike hyperpolarization), and (ii) the refractory period is longer than the spike upswing. Both of these are normally true for neurons, but are not necessarily true for the values supplied to a tool like this one.

One or more templates may be supplied. Two very similar ones (generated using a MATLAB Hodgkin-Huxley simulator [3]) are supplied (spike_1n.csv, and spike_2n.csv). The format for these files is simply a comma separated set of values. Each line contains a time and a membrane voltage, and possibly other values which are ignored. The times are in milliseconds, and the voltages are in millivolts. Note that the times need not start at 0 (but the template will be normalised so that it does start at 0). The times do not need to be equally spaced out: the system will interpolate (using cubic interpolation) so that a value for the template at each sample time is produced. The start and end values for the membrane voltages should be the same or very similar. In addition, we have supplied another one, based on [1], in a file naundorf_vivo.csv.

6 List of internal alterable values

Virtually anything that can be set is set near the start of *gennoisyspikes*.

6.1 Temporal Information

The aim is to allow the replication of the effect of the spike over the extent of the spiking surface of the neuron. To achieve this, we consider only the delay, and the relative strength of the signal for that delay. (The actual structure of the neuron is assumed to be captured purely by the strength of the delayed signal (see [2]).) The delay system uses two values set either in the code or in the call to *generatenoisyspikes* (see section 2.1). In the code the variables are

t_delta_integrate the timestep used in the delay integration. This needs to be quite small (default is 30μseconds) because the original signal rises very rapidly, and thus contains harmonics up to a high frequency. Logically, *t_delta_integrate* should be less than half the maximal frequency of interest.

n_delta_integrate the number of timesteps used. This defaults to 60.

The result is that the default implies a maximal spike generation duration of the product of these two, that is, 1.8 milliseconds.

Three arrays are used to determine the actual weights to be applied at each of the delay instants,

orig_temporal array determining the relative weights for each delay for the target signals

corr_temporal array determining the relative weights for each delay for the correlated signals

u_temporal array determining the relative weights for each delay for the uncorrelated signals

Each of these arrays is No of neurons by 3 by *n_delta_integrate*: they can be different for each neuron, and for the original, 1st derivative, and 2nd derivative. Each value is the weight for one neuron, for one of the three of (original, 1st and 2nd derivative), and for one time delay. One vector should be supplied for each target neuron. For the correlated and for the uncorrelated neurons, the weights to be used will simply go round and round the array supplied, so that there may be fewer vectors in this array than actual uncorrelated neurons. The values are read in either from a file or from numbers inside MATLAB m-files (see section 4).

6.2 Other parameters

The following parameters are set in *generatenoisysamples.m.*

1. *sample_rate* - the sampling rate for the signals: can also be set in the call.
2. *templatenames* - filenames for file with template for spike for primary neuron. Currently, these file names must all be of the same length
3. *duration* - length of the final signal in seconds: can also be set in the call
4. *n_targets* - number of target neurons: can also be set in the call.
5. *dslength* - used in smoothing during differentiation. It is a number of sample intervals. It can also be set in the call
6. *refractotoryperiod* - the default refractory period. It can be set in the call.
7. The arrays below must all be (at least) *n_targets* long
    (a) *orig_temporal* - a set of vectors detailing the relative effects of the delays inside the target neurons. Each vector must be *N_Delta_Integrate* long, and there must be at least *n_targets* of them.

(b) *spikedist* - 'P' for Poisson and 'G' for Gaussian distribution for the original signals from each neuron

(c) *poissionnumber* and *poissmeanISI* - for the Poisson distribution (array elements corresponding to Gaussian spike distributions are ignored)

(d) *gaussmeanITD* and *gaussstdev* - for the Gaussian distribution (array elements corresponding to Poisson spike distributions are ignored)

(e) *t_templateid* - the number of the template to be used in spike waveform generation for this target neuron. Must be between 1 and the number of templates.

(f) Mixing coefficients for the primary signal

   i. *origmixcoeffs_orig* - (undifferentiated) plain signal strength. For extracellular recordings this is 0 or very small.

   ii. *origmixcoeffs_d* - differentiated signal strength. The differentiated signals represent the signals from neighbouring neurons contributing to a given electrode. The strength of the differentiated signal is user determined. A reasonable guide is to make it proportional to $1/d^2$ where d is the Euclidean distance between a neuron and the electrode.

   iii. *origmixcoeffs_dd* - twice differentiated signal strength. This coefficient models the effect of glia cells around the electrode. The comment above also applies.

8. Jittered noise overall size. These are used to set the overall size of the total jittered signal. If desired, the last three can all be set to 1, and the jittered noise mixing coefficients (below) used on their own. It can also be useful to use *jitteroveralllevel* as a single value to adjust the overall jittered interference size.

   (a) *jitteroveralllevel* - overall size of the jittered noise: a simple scalar multiplier useful for scaling the values

   (b) *n_jitter* - number of jittered versions of the original spike train: can also be set in call

   (c) *jitteroverallsize_orig* - overall size of the original jittered signal

   (d) *jitteroverallsize_d* - overall size the differentiated jittered signal strength

   (e) *jitteroverallsize_dd* - overall size the twice differentiated jittered signal strength

9. The arrays below should be the same length as the number of vectors in *corr_temporal*. If this is less than n_jitter the values used will cycle through these over and over again.

   (a) *corr_temporal* - delay weight values for the jittered neurons: see *orig_temporal* above.

   (b) *jprobspikes* - probability that a spike produced actually results in an output spike

(c) *jjstd* - standard deviation of the jitter (mean=0)

(d) *jitter_orig* - the number of the target neuron on which this jittered neuron's spikes are based. Must be between 1 and the number of target neurons

(e) *j_templateid* - the number of the template to be used in spike waveform generation for this jittered neuron. Must be between 1 and the number of templates.

(f) Jittered noise mixing coefficients: set the level of the signal from each jittered neuron

  i. *jmixcoeffs_orig* - plain (intercellular) signal

  ii. *jmixcoeffs_d* - for 1st differential

  iii. *jmixcoeffs_dd* - for 2nd differential

10. Uncorrelated spike noise overall size. The last three can be used to set the overall size of the uncorrelated noise signal. Alternatively, if desired the last three can all be set to 1, and the Uncorrelated spikes mixing coefficients used to adjust signal levels. It can also be useful to use *ucoveralllevel* to adjust the overall size of the uncorrelated interference.

   (a) *ucoveralllevel* - a scalar used to adjust the overall size of the uncorrelated interference

   (b) *n_uncorr* - number of uncorrelated spike trains: can also be set in call

   (c) *u_overallsize_orig* - overall size of the original unjittered signal

   (d) *u_overallsize_d* - overall size of the differentiated unjittered signal

   (e) *u_overallsize_dd* - overall size of the twice differentiated unjittered signal

11. All arrays below should be the same length as *u_temporal.* If *n_uncorr* is greater than this, the values used will cycle through these over and over again.

   (a) *uncorr_temporal* - weights for the delayed uncorrelated neurons. see *orig_temporal* above.

   (b) *u_spikedist* -distribution type for these spikes

   (c) *u_poissonnumber* - number for use when generating Poisson distribution

   (d) *u_poissonmeanISI* - mean ISI for each poisson distribution

   (e) *u_gaussianmeanITD* - mean interspike time difference (should be ISI !)

   (f) *u_gaussstdev* - STD thereof

   (g) *u_templateid* - the number of the template to be used in spike waveform generation for this uncorrelated neuron. Must be between 1 and the number of templates.

   (h) Uncorrelated spikes mixing coefficients: set the level of the signal from each uncorrelatied neuron
      i. *umixcoeffs_orig* - for intracellular signal
      ii. *umixcoeffs_d* - for 1st differential
      iii. *umixcoeffs_d* - for 2nd differential

12. *noise_snr* - noise level, in dB (for the *awgn()*) (MATLAB add white Gaussian noise) function

13. *fminval* - final minimum value for the output signal

14. *fmaxval* - final maximum value for the output signal

7 Some examples

File *generatenoisysamples.m* provides an example which when called with no parameters produces a signal output lasting 0.5 seconds, with two target outputs, 7 jittered spike trains, and 15 independent trains (or whatever the defaults have been re-set to inside *gennoisyspikes.m*). When called

```
[signals1 target1 r1] = generatenoisysamples('N_targets', 4) ;
```

it produces a signal with four target outputs, and the same number of jittered and independent spike trains.

It is possible to produce purely noisy signals with no targets or correlated signals: simply set the number of targets and correlated outputs to 0.

Here's an example doing this:

```
 [ sig, spikeinfo, rngdata] = generatenoisysamples() ;
```

Creates a rather noisy spike train. This uses the defaults: sample rate 24,000, 2 target neurons, 7 jittered neurons, and 15 uncorrelated neurons. Total time is 0.5 seconds (actually, returns a vector slightly longer than 12000 samples.)

```
[ sig1, spikeinfo1, rngdata1] = generatenoisysamples('ReuseRNGState', ...
               rngdata, 'ReuseTargets', spikeinfo) ;
```

Creates a different noisy spike train, using the same paramaters.

```
 [  sig2, spikeinfo2, rngdata2] = generatenoisysamples('ReuseRNGState', rngdata, ...
                 'ReuseTargets', spikeinfo, 'N_Jitter', 0, 'N_Uncorr', 0) ;
```

Creates a spike train, with the spikes at the same time, but without any interfering spikes.

```
[  sig3, spikeinfo3, rngdata3] = generatenoisysamples('ReuseRNGState', c, ...
                'ReuseTargets', b, 'N_Jitter', 0, 'N_Uncorr', 0) ;
```

Should create the same spike train again, so that

```
 plot(1:length(sig2), sig2, 1: length(sig3), sig3) ;
```

should plot the two on top of each other.

In addition, the subdirectory *Extras* contains a number of m-files which were used for assessing the KwikKlusters (see `http://klustakwik.sourceforge.net/` and wave_clus (see `http://www.vis.caltech.edu/~rodri/Wave_clus/Wave_clus_home.htm`) spike sorting packages. The .m files in this directory have been commented to explain their function: they are probably not directly useful for further research, but may be a template.

And finally

These MATLAB m-files are not guaranteed in any way or sense whatsoever. Nonetheless, we do attempt to make sure that they are bug-free. If you find errors, please tell us about them. In addition, if you add new routines which might prove useful to others, tell us about them as well, and we'll put in links to them.

References

[1] B. Naundorf, F. Wolf, and M. Volgushev. Unique features of action potential initiation in cortical neurons. *Nature*, 440(7087):1060–1063, 2006.

[2] L.S. Smith and N. Mtetwa. Generating realistic extracellular noisy spike trains for testing detection and sorting algorithms. in preparation, 2006.

[3] D.S. Touretzky, M.V. Albert, N.D. Daw, and A. Ladsariya. HHsim: Graphical Hodgkin-Huxley simulator. at `http://www.cs.cmu.edu/~dst/HHsim/`, 2004.