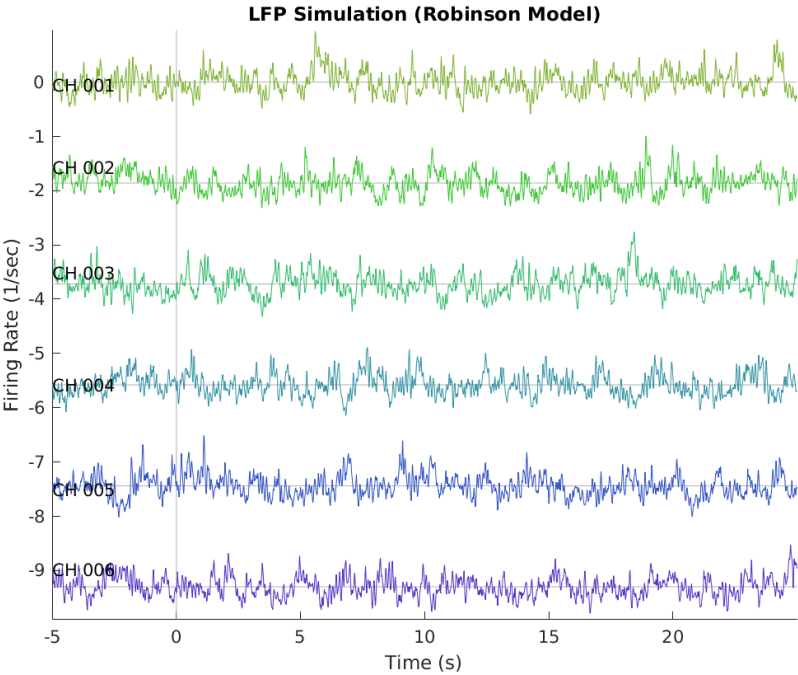


# Robinson / Freyer / Hindriks Model

## Library Reference

Written by Christopher Thomas – January 25, 2024.



# Contents

<b>1</b>	<b>Data Structures and Additional Notes</b>	<b>1</b>
1.1	MODELPARAMSROBINSON.txt . . . . .	1
<b>2</b>	<b>Function Reference</b>	<b>3</b>
2.1	synthRFH_addLoopGainInfo.m . . . . .	3
2.2	synthRFH_estimateOperatingPointExponential.m . . . . .	4
2.3	synthRFH_estimateOperatingPointLinear.m . . . . .	4
2.4	synthRFH_findLoops.m . . . . .	5
2.5	synthRFH_ftWrapper_simulateNetwork.m . . . . .	6
2.6	synthRFH_getEdgeGainGradients.m . . . . .	6
2.7	synthRFH_getEdgeGains.m . . . . .	7
2.8	synthRFH_getModelParamsFreyer.m . . . . .	7
2.9	synthRFH_getModelParamsHindriks.m . . . . .	8
2.10	synthRFH_getModelParamsRobinson.m . . . . .	8
2.11	synthRFH_getOperatingPointGradient.m . . . . .	9
2.12	synthRFH_getRegionInfo.m . . . . .	10
2.13	synthRFH_getSigmoid.m . . . . .	10
2.14	synthRFH_getSigmoidDerivative.m . . . . .	11
2.15	synthRFH_getSigmoidInverse.m . . . . .	11
2.16	synthRFH_makeFTDataFromMatrices.m . . . . .	11

2.17	<code>synthRFH_optimizeCouplings.m</code> . . . . .	12
2.18	<code>synthRFH_simulateNetwork.m</code> . . . . .	13
2.19	<code>synthRFH_stepCortexThalamus.m</code> . . . . .	14
<b>3</b>	<b>Sample Code</b>	<b>16</b>
3.1	<code>do_synth_robinson.m</code> . . . . .	16

# Chapter 1

## Data Structures and Additional Notes

### 1.1 MODELPARAMSROBINSON.txt

The synthRFH\_XXX functions accept a model parameters structure with the fields described below.

Relevant references:

(Robinson 2002)

<https://journals.aps.org/pre/abstract/10.1103/PhysRevE.65.041924>

(Freyer 2011)

<https://www.jneurosci.org/content/31/17/6353.short>

(Hindriks 2023)

<https://www.nature.com/articles/s42003-023-04648-x>

Sigmoid parameters (common to all):

"qmax" is the maximum firing rate (1/sec); typically 250.

"threshlevel" is the average neuronal threshold (mV); typically 15.

"threshsigma" is the standard deviation of the neuronal threshold (mV);  
typically 6.0.

Neural dynamics parameters (common to all):

"alpha" is the inverse decay time (1/sec); typically 50.

"beta" is the inverse rise time (1/sec); typically 200.

"gamma" is the inverse within-cortex propagation time (1/sec); typically 100.

Cortico-thalamic circuit parameters (for simulateNetwork):

"halfdelay\_ms" is half of the round-trip cortex/thalamus loop time (ms); typically 40.

Additional coupling parameters (for simulateNetwork):

"noisecoupling" is the coupling weight of noise into the specific nucleus. Typically 0.5.

"mixturecoupling" is the coupling weight of mixed-population cortex excitatory signals into the cortex. Typically 0.07.

Noise generation parameters (for simulateNetwork):

"noisemean" is the mean noise signal value; typically 0.

"noisesigma" is the standard deviation of additive noise; typically 0.1.

"noisemultfactor" is "chi" in Freyer 2011/Hindriks 2023; the standard deviation of multiplicative noise is  $\chi * \text{noisesigma}$ . Typically 0.3.

This is the end of the file.

## Chapter 2

# Function Reference

### 2.1 synthRFH\_addLoopGainInfo.m

```
% function newloopinfo = synthRFH_addLoopGainInfo( ...
%   oldloopinfo, edgегains, edgeгаingradients )
%
% This accepts a loop metadata structure array and augments each record
% with gain-related information.
%
% An empty cell array may be supplied for the gain gradients, to omit
% gradient information.
%
% "oldloopinfo" is a structure returned by synthRFH_findLoops().
% "edgегains" is a 4x4 matrix indexed by (destination, source) that contains
% the small-signal firing rate gains between each source and destination
% for the excitatory, inhibitory, specific nucleus, and reticular nucleus
% neural populations.
% "edgeгаingradients" is a 4x4 cell array indexed by (destination, source)
% that contains the gradient with respect to "intcouplings" of the
% small-signal firing rate gains in "edgегains". Passing an empty cell
% array skips calculation of loop gain gradients.
%
% "newloopinfo" is a copy of "oldloopinfo" with the following fields added
% to each record:
%   "cyclegainraw" is the small-signal gain from traversing once around the
%   loop, without taking into account filter attenuation.
%   "cyclegain" is the small-signal gain from traversing once around the
%   loop with filter attenuation taken into account.
%   "envelopetau" is the time constant for the growth (positive) or decay
%   (negative) of the oscillation envelope. The envelope is exp(t/tau).
%   "cyclegaingradient" is the gradient with respect to "intcouplings" of
%   "cyclegain". This is a 4x4 matrix (per "intcouplings"). If an empty
%   cell array is passed as "edgeгаingradients", this field is omitted.
```

## 2.2 synthRFH\_estimateOperatingPointExponential.m

```
% function [ firingrates potentials ] = ...
%   synthRFH_estimateOperatingPointExponential( ...
%       modelparams, intcouplings, startpotentials )
%
% This attempts to estimate the DC operating point of a Robinson neural model.
%
% Per the model guide, operating points with firing rates much less than the
% maximum are solutions to the equation:
%
% potentials = intcouplings * Q_0 * exp( potentials / sigmaprime )
%
% This function does a brute-force gradient descent search for operating
% points using "fsolve". This only finds one point; several points may
% exist.
%
% NOTE - Operating point firing rates must be examined to confirm that
% they are much less than modelparams.qmax. If they are not several times
% smaller, the operating point is not correct.
%
% "modelparams" is a model parameter structure with the fields described in
%   MODELPARAMSROBINSON.txt.
% "intcouplings" is a 4x4 matrix indexed by (destination,source) that
%   provides the coupling weights (in mV*s) between excitatory, inhibitory,
%   specific nucleus, and reticular nucleus neural populations.
% "startpotentials" is a vector containing cell potentials for the excitatory,
%   inhibitory, specific nucleus, and reticular nucleus populations used as
%   a starting point for further optimization. Set this to [] to call
%   synthRFH_estimateOperatingPointLinear() to generate starting potentials.
%
% "firingrates" is a vector containing firing rates for the excitatory,
%   inhibitory, specific nucleus, and reticular nucleus populations.
% "potentials" is a vector containing cell potentials for the excitatory,
%   inhibitory, specific nucleus, and reticular nucleus populations.
```

## 2.3 synthRFH\_estimateOperatingPointLinear.m

```
% function [ firingrates potentials ] = ...
%   synthRFH_estimateOperatingPointLinear( modelparams, intcouplings )
%
% This attempts to estimate the DC operating point of a Robinson neural model.
%
% Per the model guide, operating points with firing rates much less than the
% maximum are solutions to the equation:
%
% potentials = intcouplings * Q_0 * exp( potentials / sigmaprime )
```

```

%
% This function uses a linear approximation to exp(x) to estimate operating
% points for potentials that are small compared to sigma_prime. NOTE - This
% is not a robust assumption! The operating point must be examined to
% confirm that this condition holds. If it doesn't hold, the estimated
% operating point is not correct.
%
% "modelparams" is a model parameter structure with the fields described in
% MODELPARAMSROBINSON.txt.
% "intcouplings" is a 4x4 matrix indexed by (destination,source) that
% provides the coupling weights (in mV*s) between excitatory, inhibitory,
% specific nucleus, and reticular nucleus neural populations.
%
% "firingrates" is a vector containing firing rates for the excitatory,
% inhibitory, specific nucleus, and reticular nucleus populations.
% "potentials" is a vector containing cell potentials for the excitatory,
% inhibitory, specific nucleus, and reticular nucleus populations.

```

## 2.4 synthRFH\_findLoops.m

```

% function loopinfo = ...
%   synthRFH_findLoops( modelparams, intcouplings, minweight )
%
% This examines a Robinson model coupling matrix and identifies loops.
% Loop metadata is extracted.
%
% This does not extract loop gain, since that varies with operating point.
%
% "modelparams" is a model parameter structure with the fields described in
% MODELPARAMSROBINSON.txt.
% "intcouplings" is a 4x4 matrix indexed by (destination, source) that
% provides the coupling weights (in mV*s) between excitatory, inhibitory,
% specific nucleus, and reticular nucleus neural populations.
% "minweight" is the threshold to use when evaluating whether a coupling
% weight is nonzero (the absolute value must be at least "minweight").
%
% "loopinfo" is a structure array with one element per identified loop, and
% the following fields:
%   "label" is a unique identifier for the loop (the concatenation of the
%   letters associated with each region in the loop's path).
%   "regionsvisited" is a vector containing the indices of each region in the
%   loop's path. The first index is repeated as the last index.
%   "delay" is the propagation time for one cycle around the loop, in seconds.
%   "isinverting" is true if the product of the loop's edge couplings is
%   negative, and false if the product is positive.
%   "frequency" is the loop's fundamental mode frequency in Hz (1/delay if
%   non-inverting, half that if inverting).
%   "attenuation" is the loop's attenuation at its fundamental mode

```



```
% frequency from the alpha, beta, and gamma model parameters. This will
% be between 1 if the signal is passed perfectly and less than 1 if not.
```

## 2.5 synthRFH\_ftWrapper\_simulateNetwork.m

```
% function ftdata = synthRFH_ftWrapper_simulateNetwork( ...
%   trialcount, triggertime, poplabels, wantprogress, ...
%   duration, startup, timestep, modelparams, intcouplings, ...
%   popcount, cortexmixing, cortexdelays_ms )
%
% This is a wrapper for synthRFH_simulateNetwork().
% See that function's documentation for details.
%
% This simulates cortex and thalamus neural activity, using the model from
% Robinson 2002 with augmented input per Freyer 2011 and Hindriks 2023:
%
% https://journals.aps.org/pre/abstract/10.1103/PhysRevE.65.041924
% https://www.jneurosci.org/content/31/17/6353.short
% https://www.nature.com/articles/s42003-023-04648-x
%
% "trialcount" is the number of Field Trip trials to simulate.
% "triggertime" is the trigger offset from the start of simulation (seconds).
% "poplabels" is a cell array containing Field Trip channel names for the
%   neural populations, or {} to automatically generate channel names.
% "wantprogress" is true to write progress messages to the console, false
%   otherwise.
%
% Remaining arguments are per synthRFH_simulateNetwork().
%
% "ftdata" is a ft_datatype_raw structure containing trial data, including
%   a header (hdr) and a config structure with trial definitions (cfg.trl).
```

## 2.6 synthRFH\_getEdgeGainGradients.m

```
% function edgegaingradients = synthRFH_getEdgeGainGradients( ...
%   modelparams, intcouplings, firingrates, rategradients )
%
% This function calculates the gradient of the small-signal gains of each
% network edge in a Robinson neural model with respect to the internal
% coupling matrix, at a specified operating point.
%
% This assumes firing rates that are much less than the maximum rate, and
% assumes that the gradients of the firing rates are already known.
%
% "modelparams" is a model parameter structure with the fields described in
%   MODELPARAMSROBINSON.txt.
```

```

% "intcouplings" is a 4x4 matrix indexed by (destination, source) that
%   provides the coupling weights (in mV*s) between excitatory, inhibitory,
%   specific nucleus, and reticular nucleus neural populations.
% "firingrates" is a vector specifying the operating point firing rates of
%   excitatory, inhibitory, specific nucleus, and reticular nucleus neural
%   populations.
% "rategradients" is a cell array with 4 cells, containing gradient matrices
%   with respect to "intcouplings" for the excitatory, inhibitory, specific
%   nucleus, and reticular nucleus firing rates.
%
% "edgegaingradients" is a 4x4 cell array indexed by (destination, source)
%   that contains the gradient with respect to "intcouplings" of the
%   small-signal firing rate gains between each source and destination for
%   the excitatory, inhibitory, specific nucleus, and reticular nucleus
%   neural populations.

```

## 2.7 synthRFH\_getEdgeGains.m

```

% function edgegains = synthRFH_getEdgeGains( ...
%   modelparams, intcouplings, firingrates )
%
% This function estimates the small-signal gain of each network edge in a
% Robinson neural model, at a specified operating point.
%
% "modelparams" is a model parameter structure with the fields described in
%   MODELPARAMSROBINSON.txt.
% "intcouplings" is a 4x4 matrix indexed by (destination, source) that
%   provides the coupling weights (in mV*s) between excitatory, inhibitory,
%   specific nucleus, and reticular nucleus neural populations.
% "firingrates" is a vector specifying the operating point firing rates of
%   excitatory, inhibitory, specific nucleus, and reticular nucleus neural
%   populations.
%
% "edgegains" is a 4x4 matrix indexed by (destination, source) that contains
%   the small-signal firing rate gains between each source and destination
%   for the excitatory, inhibitory, specific nucleus, and reticular nucleus
%   neural populations.

```

## 2.8 synthRFH\_getModelParamsFreyer.m

```

% function [ modelparams intcouplings ] = synthRFH_getModelParamsFreyer()
%
% This returns model and coupling parameters for use with
% synthRFH_stepCortexThalamus() and related functions.
%
% Values are the ones used in Freyer 2011 (Table 1):

```

```
% https://www.jneurosci.org/content/31/17/6353.short
%
% These have slightly adjusted alpha and beta time constants and
% substantially different coupling weights vs Robinson 2002.
%
% No arguments.
%
% "modelparams" is a model parameter structure with the fields described
% in MODELPARAMSROBINSON.txt.
% "intcouplings" is a 4x4 matrix indexed by (destination,source) that
% provides the coupling weights (in mV*s) between excitatory, inhibitory,
% specific nucleus, and reticular nucleus neurons.
```

## 2.9 synthRFH\_getModelParamsHindriks.m

```
% function [ modelparams intcouplings ] = synthRFH_getModelParamsHindriks()
%
% This returns model and coupling parameters for use with
% synthRFH_stepCortexThalamus() and related functions.
%
% Values are the ones used in Hindriks 2023:
% https://www.nature.com/articles/s42003-023-04648-x
% https://github.com/Prejaas/amplitudecoupling
%
% These are identical to the Robinson 2002 values, except with a smaller
% noise coupling coefficient.
%
% No arguments.
%
% "modelparams" is a model parameter structure with the fields described
% in MODELPARAMSROBINSON.txt.
% "intcouplings" is a 4x4 matrix indexed by (destination,source) that
% provides the coupling weights (in mV*s) between excitatory, inhibitory,
% specific nucleus, and reticular nucleus neurons.
```

## 2.10 synthRFH\_getModelParamsRobinson.m

```
% function [ modelparams intcouplings ] = synthRFH_getModelParamsRobinson()
%
% This returns model and coupling parameters for use with
% synthRFH_stepCortexThalamus() and related functions.
%
% Values are the ones used in Robinson 2002 (Table 1):
% https://journals.aps.org/pre/abstract/10.1103/PhysRevE.65.041924
%
% No arguments.
```

```
%
% "modelparams" is a model parameter structure with the fields described
%   in MODELPARAMSROBINSON.txt.
% "intcouplings" is a 4x4 matrix indexed by (destination,source) that
%   provides the coupling weights (in mV*s) between excitatory, inhibitory,
%   specific nucleus, and reticular nucleus neurons.
```

## 2.11 synthRFH\_getOperatingPointGradient.m

```
% function [ rategradients potentialgradients ] = ...
%   synthRFH_getOperatingPointGradient( ...
%     modelparams, intcouplings, testpotentials, couplingstep, zerohandling )
%
% This function attempts to estimate the gradient of the DC operating point
% of a Robinson neural model with respect to the internal coupling matrix.
%
% This assumes firing rates that are much less than the maximum rate. Per the
% model guide, operating points under these conditions are solutions to:
%
% potentials = intcouplings * Q_0 * exp( potentials / sigmaprime )
%
% The gradient of this function is evaluated numerically, by perturbing the
% coupling matrix and finding operating points for each perturbed version.
%
% "modelparams" is a model parameter structure with the fields described in
%   MODELPARAMSROBINSON.txt.
% "intcouplings" is a 4x4 matrix indexed by (destination, source) that
%   provides the coupling weights (in mV*s) between excitatory, inhibitory,
%   specific nucleus, and reticular nucleus neural populations.
% "testpotentials" is a vector containing cell potentials for the excitatory,
%   inhibitory, specific nucleus, and reticular nucleus populations at the
%   test point to analyze. This is used as the starting point for the
%   operating point search.
% "couplingstep" is a scalar indicating the amount by which each coupling
%   value should be perturbed when evaluating the gradient numerically. This
%   should be much smaller than the magnitude of nonzero coupling values.
% "zerohandling" is 'all' to compute the gradient with respect to all
%   coupling values, and 'nonzero' to compute the gradient with respect to
%   all coupling values with magnitudes larger than "couplingstep" (storing
%   NaN as the gradient for coupling values that are smaller than this).
%
% "rategradients" is a cell array with 4 cells, containing gradient matrices
%   with respect to "intcouplings" for the excitatory, inhibitory, specific
%   nucleus, and reticular nucleus firing rates.
% "potentialgradients" is a cell array with 4 cells, containing gradient
%   matrices with respect to "intcouplings" for the excitatory, inhibitory,
%   specific nucleus, and reticular nucleus cell potentials.
```

## 2.12 synthRFH\_getRegionInfo.m

```
% function [ indices_lut names_lut ] = synthRFH_getRegionInfo()
%
% This returns metadata associating each region simulated by the Robinson
% 2002 model with a row/column index, a pretty name, and abbreviated names.
%
% No arguments.
%
% "indices_lut" is a structure with the following fields:
%   "cortex_excitatory" is the row/column index corresponding to excitatory
%   neurons in the cortex.
%   "cortex_inhibitory" is the row/column index corresponding to inhibitory
%   neurons in the cortex.
%   "thalamus_specific" is the row/column index corresponding to "specific
%   nucleus" neurons in the thalamus (also called the relay population).
%   "thalamus_reticular" is the row/column index corresponding to "reticular
%   nucleus" neurons in the thalamus.
%
% "names_lut" is a structure array indexed by region number, with the
% following fields (all character vectors):
%   "title" is a verbose plot-safe name.
%   "label" is a terse filename-safe and plot-safe name.
%   "letter" is a capital letter.
%   "lutfield" is the name of the corresponding field in "indices_lut".
```

## 2.13 synthRFH\_getSigmoid.m

```
% function firingrate = synthRFH_getSigmoid( ...
%   potential, maxrate, threshlevel, threshdeviation )
%
% This converts a cell-body potential (V in Robinson's model) to a firing
% rate (Q in Robinson's model).
%
% This is described in eq. 1 of Robinson 2002:
% https://journals.aps.org/pre/abstract/10.1103/PhysRevE.65.041924
% And in terms of sigma, not sigma-prime, in eq. m4 of Freyer 2011:
% https://www.jneurosci.org/content/31/17/6353.short
%
% "potential" is the cell body potential (V) to convert. This may be a
% vector or matrix, to convert several potentials.
% "maxrate" is the maximum firing rate (Q_max).
% "threshlevel" is the average neuron threshold (theta).
% "threshdeviation" is the standard deviation of the average neuron threshold
% (sigma - not sigma prime!).
%
% "firingrate" is the resulting firing rate (Q, or S(V)).
```

## 2.14 synthRFH\_getSigmoidDerivative.m

```
% function dQdV = synthRFH_getSigmoidDerivative( ...
%   potential, maxrate, threshlevel, threshdeviation )
%
% This gets the derivative of the Robinson 2002 activation function with
% respect to cell-body potential, for a given potential.
%
% "potential" is the cell body potential (V) to evaluate the derivative at.
% This may be a vector or matrix, to evaluate several potentials.
% "maxrate" is the maximum firing rate (Q_max).
% "threshlevel" is the average neuron threshold (theta).
% "threshdeviation" is the standard deviation of the average neuron threshold
%   (sigma - not sigma prime!).
%
% "dQdV" is the derivative of the firing rate with respect to potential,
%   at the specified potential.
```

## 2.15 synthRFH\_getSigmoidInverse.m

```
% function potential = synthRFH_getSigmoidInverse( ...
%   firingrate, maxrate, threshlevel, threshdeviation )
%
% This converts a firing rate (Q in Robinson's model) back to a cell-body
% potential (V in Robinson's model), performing the inverse of
% synthRFH_getSigmoid().
%
% "firingrate" is the firing rate (Q, or S(V)). This may be a vector or
%   matrix, to convert several firing rates into potentials.
% "maxrate" is the maximum firing rate (Q_max).
% "threshlevel" is the average neuron threshold (theta).
% "threshdeviation" is the standard deviation of the average neuron threshold
%   (sigma - not sigma prime!).
%
% "potential" is the cell body potential (V).
```

## 2.16 synthRFH\_makeFTDataFromMatrices.m

```
% function ftdata = synthRFH_makeFTDataFromMatrices( ...
%   wavedata, samprate, trigoffsetsamps, trigtimes, labelprefix )
%
% This builds a ft_datatype_raw structure containing supplied waveform data.
%
% "wavedata" is either a Nchans x Nsamples x Ntrials matrix or a cell array
%   with Ntrials cells, each containing a Nchans x Nsamples matrix.
```

```

% "samprate" is the sampling rate.
% "trigoffsetsamps" specifies when the trigger time is within each trial.
%   This is 0 if the first sample in the trial is at the trigger, positive
%   if a later sample in the trial is the trigger, and negative if the
%   trigger occurred before the first sample in the trial.
% "trigtimes" is a vector with trigger times for each trial. This is used for
%   constructign trial definitions. If necessary, an offset is added to
%   guarantee that all samples have positive indices in the global data.
%   Specify [] to automatically build trigger times for trial definitions.
% "labelprefix" is a character vector containing a prefix to use when
%   constructing channel labels.
%
% "ftdata" is a ft_datatype_raw structure containing trial data. This
%   includes a header and a config structure with cfg.trl.

```

## 2.17 synthRFH\_optimizeCouplings.m

```

% function [ bestcouplings besterr ] = synthRFH_optimizeCouplings( ...
%   modelparams, startcouplings, loopinfo, loopgoals, ...
%   taulimit, bestfactor, maxprobes )
%
% This attempts to optimize the internal coupling weights of a Robinson
% neural model to modify the loop gains to meet a specified set of
% constraints.
%
% This works via brute-force gradient descent (using "fsolve").
%
% Note that the output coupling matrix follows the constraints described
% in Robinson 2002, Freyer 2011, and Hindriks 2023: The nu_ix weights are
% set to the same values as the nu_ex weights, and the weights that are
% zero in the references are set to be zero here.
%
% "modelparams" is a model parameter structure with the fields described in
%   MODELPARAMSROBINSON.txt.
% "startcouplings" is a 4x4 matrix indexed by (destination,source) that
%   provides the coupling weights (in mV*s) between excitatory, inhibitory,
%   specific nucleus, and reticular nucleus neural populations.
% "loopinfo" is a loop metadata structure returned by synthRFH_findLoops().
% "loopgoals" is a Nx2 cell array. Each row contains a loop label in the
%   first column and a goal keyword in the second column. Goal keywords are
%   'biggest', 'grow', 'decay', or 'dontcare'. Loops with positive tau less
%   than the specified limit are growing, loops with negative tau with
%   absolute value less than the specified limit are decaying, and a growing
%   loop is "biggest" if its tau is smaller than all other growing tau
%   values by the specified factor. Loops not listed default to 'dontcare'.
% "taulimit" is the longest time constant (in seconds) that an envelope may
%   have to be counted as "growing" or "decaying". Typically 1.0 or less.
% "bestfactor" is the factor by which the "best" loop's tau must be smaller

```

```
% than all other growing tau values. Typically 1.2-1.5.
% "maxprobes" is the maximum number of probes to make, or [] or NaN for the
% default number (about 1400).
%
% "bestcouplings" is a perturbed version of "startcouplings" that produces
% the desired loop behaviors.
% "besterr" is the error value associated with "bestcouplings". This is in
% the range 0..1, with 0 being perfect and 1 being terrible.
```

## 2.18 synthRFH\_simulateNetwork.m

```
% function firingrates = synthRFH_simulateNetwork( ...
% duration, startup, timestep, modelparams, intcouplings, ...
% popcount, cortexmixing, cortexdelays_ms )
%
% This simulates cortex and thalamus neural activity, using the model from
% Robinson 2002 with augmented input per Freyer 2011 and Hindriks 2023:
%
% https://journals.aps.org/pre/abstract/10.1103/PhysRevE.65.041924
% https://www.jneurosci.org/content/31/17/6353.short
% https://www.nature.com/articles/s42003-023-04648-x
%
% This simulates N populations of excitatory and inhibitory neurons in the
% cortex, and N populations of neurons in the specific and reticular nuclei
% in the thalamus (per Freyer 2011). Excitatory neuron populations in the
% cortex interact with each other, per Hindriks 2023.
%
% NOTE - This uses the forward Euler method for evolving system state.
% This is numerically stable if and only if the time step is much smaller
% than the time scales of any system dynamics. Make this much smaller than
% you think you need to.
%
% NOTE - If multiple durations are specified, various parameters _may_ be
% specified for each epoch rather than globally. This is optional; any
% parameters that are not specified per-epoch are duplicated as needed.
%
% "duration" is the number of seconds to simulate. NOTE - This may be a
% vector, specifying several successive durations which may have
% different simulation parameters.
% "startup" is the number of seconds to simulate before the duration to
% allow the simulation to settle/converge. This is typically 2-5 seconds.
% "timestep" is the amount of time to advance the simulation during each
% sample, in seconds. NOTE - This must be much smaller than system
% dynamics timescales!
% "modelparams" is a structure specifying model tuning parameters, per
% MODELPARAMSROBINSON.txt. NOTE - If "duration" is a vector, this may
% optionally be a struct array specifying different parameters for each
% epoch.
```



```

% "intcouplings" is a 4x4 matrix indexed by (destination,source) that
% provides the coupling weights (in mV*s) between excitatory cortex
% neurons (1), inhibitory cortex neurons (2), specific nucleus neurons (3),
% and reticular neurons (4). Typical coupling range from -2 to +2.
% NOTE - If "duration" is a vector, this may optionally be a 4x4xN matrix
% specifying different couplings for each epoch.
% "popcount" is the number of neural populations to simulate (Npop).
% "cortexmixing" is a Npop x Npop matrix indexed by (destination,source)
% that specifies the internal communication mixing of excitatory neurons
% in the cortex. After mixing, these then get weighted by (scalar)
% modelparams.mixturecoupling before integration as neural inputs. Set
% this to [] to omit internal cortex communication/mixing.
% NOTE - The typical mixture coupling weight in MODELPARAMSROBINSON.txt
% assumes that the cortex mixing matrix has rows normalized so that the
% absolute values of each row's elements sums to 1 (or a value close to 1).
% NOTE - If "duration" is a vector, this may optionally be a Npop x Npop x N
% matrix specifying different mixing weights for each epoch.
% "cortexdelays_ms" is a Npop x Npop matrix indexed by (destination,source)
% that specifies the internal communication delays of excitatory neurons
% in the cortex, in milliseconds. Set this to [] to omit internal cortex
% communication/mixing.
% NOTE - If "duration" is a vector, this may optionally be a Npop x Npop x N
% matrix specifying different delays for each epoch.
%
% "firingrates" is a 4 x Npop x Nsamples matrix containing firing rates
% for excitatory cortex neurons (1), inhibitory cortex neurons (2),
% specific nucleus neurons (3), and reticular neurons (4). The excitatory
% cortex neuron firing rates are gamma-damped, per Robinson 2002.

```

## 2.19 synthRFH\_stepCortexThalamus.m

```

% function statefuture = synthRFH_stepCortexThalamus( modelparams, ...
% timestep, statepresent, statepast, intcouplings, extrates, extcouplings )
%
% This generates the future state of a cortico-thalamic loop given the
% present state, using the model from Robinson 2002 with augmented input
% per Freyer 2011 and Hindriks 2023:
%
% https://journals.aps.org/pre/abstract/10.1103/PhysRevE.65.041924
% https://www.jneurosci.org/content/31/17/6353.short
% https://www.nature.com/articles/s42003-023-04648-x
%
% This simulates N populations of excitatory and inhibitory neurons in the
% cortex, and N populations of neurons in the specific and reticular nuclei
% in the thalamus (per Freyer 2011). Population bins are independent of
% each other (communication is handled by the caller).
%
% NOTE - This uses the forward Euler method for evolving system state.

```

```

% This is numerically stable if and only if the time step is much smaller
% than the time scales of any system dynamics. Make this much smaller than
% you think you need to.
%
% "modelparams" is a structure with the following fields (as described in
%  MODELPARAMSROBINSON.txt):
%   "qmax" is the maximum firing rate (1/sec); typically 250.
%   "threshlevel" is the average neuronal threshold (mV); typ. 15.
%   "threshsigma" is the standard deviation of the neuronal threshold (mV);
%       typically 6.0.
%   "alpha" is the inverse decay time (1/sec); typically 50.
%   "beta" is the inverse rise time (1/sec); typically 200.
%   "gamma" is the inverse within-cortex propagation time (1/sec); typ. 100.
% "timestep" is the amount of time to advance the simulation by (in seconds).
%  NOTE - This must be much smaller than system dynamics timescales!
% "statepresent" is a structure with the following fields:
%   "potentials" is a 4xN matrix containing Ve_k, Vi_k, Vs_k, and Vr_k from
%       the Robinson model.
%   "velocities" is a 4xN matrix containing the approximate first time
%       derivative of "potentials".
%   "cortexrates" is a 1xN matrix containing the gamma-damped firing rate
%       of cortex excitatory neurons (phi_e).
%   "cortexvelocities" is a 1xN matrix containing the approximate first time
%       derivative of "cortexrates".
% "statepast" is a structure with the same fields as "statepresent", taken
%   from a past stimulation step. This should be delayed by the one-way
%   cortex/thalamus communication time (one half of the round-trip delay).
% "intcouplings" is a 4x4 matrix indexed by (destination,source) that
%   provides the coupling weights (in mV*s) between excitatory cortex
%   neurons (1), inhibitory cortex neurons (2), specific nucleus neurons (3),
%   and reticular nucleus neurons (4). Typical couplings range from -2 to +2.
% "extrates" is a MxN matrix containing the firing rates of M external inputs
%   to the system. These may represent noise (as with Freyer 2011) or
%   cross-coupled activity within the cortex (as with Hindriks 2023).
%   This may be empty (M = 0).
% "extcouplings" is a 4xM matrix indexed by (destination,source) that
%   provides the coupling weights (in mV*s) between the internal model
%   neurons (destinations) and the system's external inputs (sources).
%   This may be empty (M = 0).
%
% "statefuture" is a copy of "statepresent" advanced by one time step.

```

## Chapter 3

# Sample Code

### 3.1 do\_synth\_robinson.m

```
% Example Code - Data synthesis using the Robinson/Freyer/Hindriks model.
% Written by Christopher Thomas.

%
% Libraries and folders.

% Generated files get put in this folder.

outdir = 'output';

% Library paths.

% NOTE - These are specific to my test system; as long as you have the
% SynthRobinson library path and the Field Trip library path set up, you can
% remove these.
addpath(' ../../library');
addpath(' ../../../../neurolab/fieldtrip/fieldtrip-latest');

% Field Trip setup.

evalc('ft_defaults');
ft_notice('off');
ft_info('off');
ft_warning('off');
```

```

%
% Configuration.

% Number of trials per test configuration.
trialcount = 10;

% Trial duration before and after "stimulation".
% Signal perturbations take about 1 second to settle after parameter changes.
trial_secs_before = 5;
trial_secs_after = 10;

% Sampling rates and filter parameters.
% Remember that we're generating firing rates, not an LFP. This pretty much
% _is_ rectified multi-unit activity.
samprate_sim = 10000;
samprate_mua = 2000;
mua_lowpass = 50; % Hz.

% Simulation parameters.

chancount = 4;

sim_startup_secs = 2;

% Explicitly set this, rather than keeping the Hindriks baseline value.
% 0.07 gets us strong power spectrum peaks at harmonics, not just the
% fundamental mode, but can also go unstable and stay at the maximum rate.
mixture_coupling_coeff = 0.06;

% Factor by which to increase v_es when boosting activity. This is very
% sensitive; raising 1% or decreasing 2% has a very visible effect.
mua_enhance_coeff_factor = 1.01;
mua_suppress_coeff_factor = 0.98;

% Mixing is already gamma-delayed, so an extra delay isn't necessary.
mixing_delays_ms = zeros(chancount, chancount);

% Mixing matrices for various test cases.

% We have to keep each population's input consistent between cases, since
% behavior is very sensitive to it. Do this by having rows sum to consistent
% values but take contributions from different population distributions.

% Everything couples to everything. All links bidirectional.
mixing_matrix_uniform = ...
[ 0      0.333 0.333 0.333 ; ...

```

```

0.333 0      0.333 0.333 ; ...
0.333 0.333 0      0.333 ; ...
0.333 0.333 0.333 0      ];

% 1-2 and 3-4, bidirectional.
mixing_matrix_routing_A = ...
[ 0 1 0 0 ; ...
  1 0 0 0 ; ...
  0 0 0 1 ; ...
  0 0 1 0 ];

% 1-4 and 2-3, bidirectional.
mixing_matrix_routing_B = ...
[ 0 0 0 1 ; ...
  0 0 1 0 ; ...
  0 1 0 0 ; ...
  1 0 0 0 ];

% 1-2-3-4-1, bidirectional.
mixing_matrix_loop_bidirectional = ...
[ 0 0.5 0 0.5 ; ...
  0.5 0 0.5 0 ; ...
  0 0.5 0 0.5 ; ...
  0.5 0 0.5 0 ];

% 1->2->3->4->1, directional.
mixing_matrix_loop_ascending = ...
[ 0 0 0 1 ; ...
  1 0 0 0 ; ...
  0 1 0 0 ; ...
  0 0 1 0 ];

% 1<-2<-3<-4<-1, directional.
mixing_matrix_loop_descending = ...
[ 0 1 0 0 ; ...
  0 0 1 0 ; ...
  0 0 0 1 ; ...
  1 0 0 0 ];

%
% Generate and save the synthetic trials.

% Get baseline simulation parameters.

[ modelparams_baseline intcouplings_baseline ] = ...
  synthRFH_getModelParamsHindriks;

```

```

% Override the default mixture coupling.
modelparams_baseline.mixturecoupling = mixture_coupling_coeff;

% Set up two epochs (before and after stimulation).

durationlist = [ trial_secs_before, trial_secs_after ];

modelparams = modelparams_baseline;
modelparams(2) = modelparams_baseline;

intcouplings = intcouplings_baseline;
intcouplings(:, :, 2) = intcouplings_baseline;

% We never change this, so just make the original dual-epoch.
mixing_delays_ms(:, :, 2) = mixing_delays_ms(:, :, 1);

% Build test cases using two different baselines: uniform, and bidirectional
% loop.

% Uniform isn't actually that useful as a baseline, but include it anyways.
% The bidirectional loop is equal to the two routing states superimposed
% and to the two directed loop cases superimposed, so it's a better baseline.

baselinematrices = ...
    { mixing_matrix_uniform, mixing_matrix_loop_bidirectional };
baselinelabels = { 'unibase', 'loopbase' };
baselinetitles = { 'uniform', 'bidirected loop' };

muafactors = [ 1.0, mua_enhance_coeff_factor, mua_suppress_coeff_factor ];
mualabels = { 'baseline', 'strong', 'weak' };
muatitles = { 'baseline', 'stronger', 'weaker' };

mixcasematrices = { mixing_matrix_routing_A, mixing_matrix_routing_B, ...
    mixing_matrix_loop_ascending, mixing_matrix_loop_descending };
mixcaselabels = { 'routeA', 'routeB', 'loopup', 'loopdown' };
mixcasetitles = { 'routing state A', 'routing state B', ...
    'ascending loop', 'descending loop' };

[ idxlut namelut ] = synthRFH_getRegionInfo();

totaltime = 0;

for baseidx = 1:length(baselinematrices)

    disp([ '== Generating cases with ' baselinetitles{baseidx} ' baseline.' ]);

    mixing_matrix = baselinematrices{baseidx};

```

```

mixing_matrix(:, :, 2) = mixing_matrix(:, :, 1);

% Generate different strengths of MUA response. This includes the baseline.

ves_baseline = intcouplings(idxlut.cortex_excitatory, ...
    idxlut.thalamus_specific, 2);

for caseidx = 1:length(muafactors)
    disp([ '-- Generating ' muatitles{caseidx} ' MUA response.' ]);
    tic;

    intcouplings(idxlut.cortex_excitatory, idxlut.thalamus_specific, 2) = ...
        ves_baseline * muafactors(caseidx);

    ftdata = synthRFH_ftWrapper_simulateNetwork( ...
        trialcount, trial_secs_before, {}, true, ...
        durationlist, sim_startup_secs, 1 / samprate_sim, modelparams, ...
        intcouplings, chancount, mixing_matrix, mixing_delays_ms );

    fname = [ outdir filesep 'mua-' baselinelabels{baseidx} ...
        '-' mualabels{caseidx} '.mat' ];
    helper_writeFTData( fname, ftdata, modelparams, intcouplings, ...
        mixing_matrix, mixing_delays_ms, mua_lowpass, samprate_mua );

    totaltime = totaltime + toc;
    durstring = helper_makePrettyTime(toc);
    disp([ '.. Elapsed time: ', durstring ]);
end

% Restore the original couplings.
intcouplings(:, :, 2) = intcouplings(:, :, 1);

% Generate different routing states. Baseline is already covered.

for caseidx = 1:length(mixcasematrices)
    disp([ '-- Generating response with ' mixcasetitles{caseidx} ...
        ' mixing.' ]);

    mixing_matrix(:, :, 2) = mixcasematrices{caseidx};

    ftdata = synthRFH_ftWrapper_simulateNetwork( ...
        trialcount, trial_secs_before, {}, true, ...
        durationlist, sim_startup_secs, 1 / samprate_sim, modelparams, ...
        intcouplings, chancount, mixing_matrix, mixing_delays_ms );

    fname = [ outdir filesep 'mua-' baselinelabels{baseidx} ...
        '-' mixcaselabels{caseidx} '.mat' ];
    helper_writeFTData( fname, ftdata, modelparams, intcouplings, ...

```

```

        mixing_matrix, mixing_delays_ms, mua_lowpass, samprate_mua );

    totaltime = totaltime + toc;
    durstring = helper_makePrettyTime(toc);
    disp([ '.. Elapsed time: ', durstring ]);
end

% Restore the original mixing matrix.
mixing_matrix(:, :, 2) = mixing_matrix(:, :, 1);

end

disp('== Finished generating cases.');
```

durstring = helper\_makePrettyTime(totaltime);  
disp([ '== Total elapsed time: ', durstring ]);

%  
% Helper Functions

% This writes a Field Trip dataset and auxiliary metadata to a matlab file.  
% The Field Trip dataset is filtered and downsampled before writing.

```

function helper_writeFTData( fname, ftdata_mua, modelparams, intcouplings, ...
    mixing_matrix, mixing_delays_ms, lowpass_corner, resample_rate )

% Field Trip configuration structures.
ftconfig_filt = struct( 'lpfilter', 'yes', 'lpfilttype', 'but', ...
    'lpfreq', lowpass_corner, 'feedback', 'no' );
ftconfig_resample = struct( 'resamplefs', resample_rate, ...
    'detrend', 'no', 'feedback', 'no' );

% Extract native-rate information that we want to keep.
% NOTE - We know that synthRFH_makeFTDataFromMatrices() provides these.
origheader = ftdata_mua.hdr;
origconfigtrl = ftdata_mua.cfg.trl;

% Filter and downsample.
ftdata_mua = ft_preprocessing( ftconfig_filt, ftdata_mua );
ftdata_mua = ft_resampleddata( ftconfig_resample, ftdata_mua );

% Save data and metadata.
save( fname, 'ftdata_mua', 'origheader', 'origconfigtrl', ...
    'modelparams', 'intcouplings', 'mixing_matrix', 'mixing_delays_ms', ...
    '-v7.3' );

end
```



```
%  
% This is the end of the file.
```