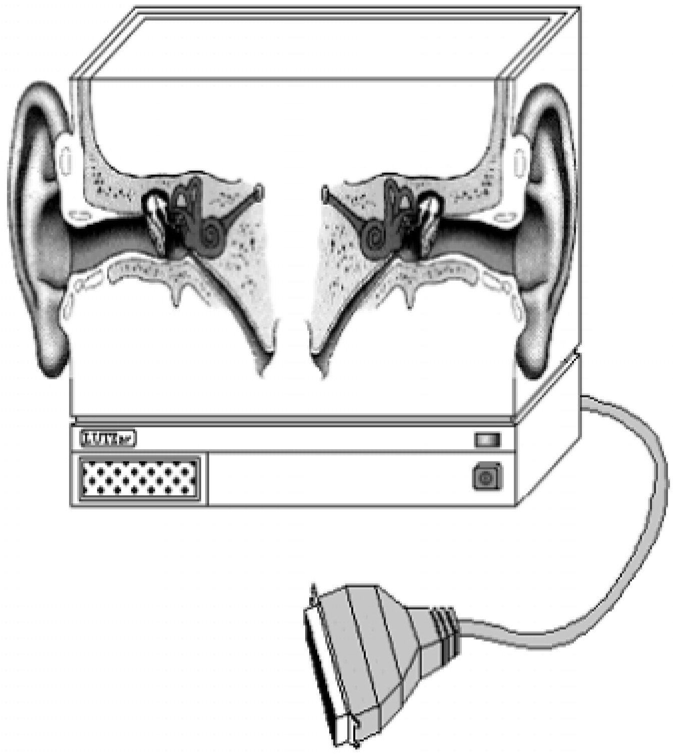
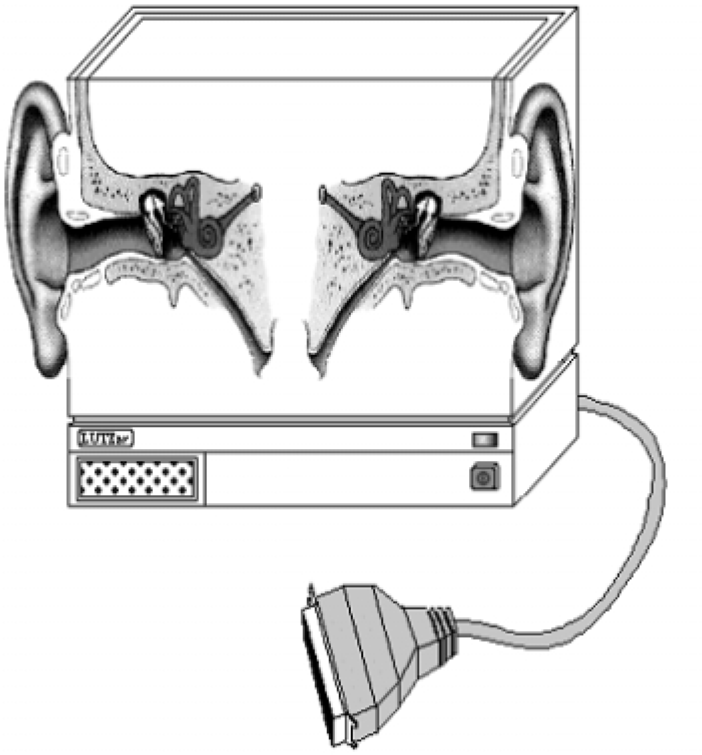
Development System for Auditory Modelling

Flexible auditory computer simulation development



Dr. Lowel P O’Mard <lowel@essex.ac.uk>

Development System for Auditory Modelling: Flexible auditory computer simulation development



by Dr. Lowel P O’Mard

Published 8th August 2005

This manual has been edited to the best of our ability. We do not accept any responsibility regarding problems caused by any errors in this document. However, we are always happy to receive communications regarding corrections or suggested revisions.

Dedication

I dedicate this work to God. I believe that without Him, nothing is that is, and without His support, nothing I have done would have been possible.

Acknowledgements

I would like to thank all those who have contributed to the making of this document. In particular Prof. R. Meddis, Dr. S. D. Holmes.

Table of Contents

List of Figures

List of Tables

List of Examples

Preface

The purpose of this document is to provide the information necessary for writing auditory simulations using the Development System for Auditory Modelling (DSAM). An overview of the units used in DSAM is first described. Users who are wishing to learn how to use the simulation script interface used by DSAM Applications such as AMS can initially go directly to the section on using simulations. This will provide a basic understanding of what is possible using DSAM. The following chapters provide the finer details of auditory simulation using DSAM.

Part I. General Concepts for the Core Routines Library

Introduction

The routines in DSAM are grouped in the respective process module types and the modules are listed in alphabetical order - with respect to the module file names (?? change this to module names). The units and naming conventions used in DSAM are described below.

Table of Contents

Chapter 1. Basic Concepts

Units

The following standard units are used in DSAM:

Table 1.1. Standard units in DSAM

Quantity	Unit	Abbreviation
Amplitude	micro-Pascals	uPa
Frequency:	hertz	Hz.
Intensity:	decibels SPL	dB SPL [a]
Time:	seconds	s. [b]

[a] Sound pressure level reference standard.

[b] The time units printed to the screen by the test programs have been converted from the internal, seconds units to milli-seconds (ms) to make the numbers more readable.

Module Naming

Each process module has a name by which it is accessed using simulation scripts or generic programming. Table 2 shows *process module prefix* that classifies the process modules. A list of all the process module names, and a brief description, is given below. Users who are not programming using DSAM may ignore the file-name prefix column in the table.

Table 1.2. Module naming conventions

Group Name	Module Prefix	File name Prefix
Analysis	Ana_	An
Filing Utilities	-	Fi
Filters	Filt_	Fl
General	-	Ge
Models: Auditory nerve spike generation	AN_SG_	MoANSG
Models: Basilar membrane	BM_	MoBM
Models: Inner hair cell	IHC_	MoHCRP
Models: Inner hair cell receptor potential	IHCRP_	MoIHC
Models: Neuron	Neur_	MoNC
Transform	Trans_	Tr
Stimuli	Stim_	St
Utilities	Util_	Ut

Chapter 2. Process Modules

Setting Process Parameters

There are a variety of methods for setting the parameters of process modules in DSAM. These methods are described in detail in the section called “Using simulation scripts ”, and only a brief overview is given here.

Using DSAM applications parameters can be set from parameter files, from dialog windows or from the command line. All of the process modules have default parameter values (with the exception of Filt_FIR and Util_SimScript). This means that only the parameters to be changed from the default values need to be explicitly set by the user.

Each process module parameter has a name/abbreviation by which it is referred. The parameters available for each process module are detailed in the section. Here each process module is described, and an example parameter file is given listing each parameter and giving the default values. Tables are also provided for process parameters that have different options.

Getting Results From Process Modules

Most modules transform input signals into output signals ready for further processing. However, some modules, such as the Ana_Intensity module, generate a result that is put into their *EarObject* output signal. When this happens the result (computed separately for each channel) is placed in the first sample of the channel although, rarely, more samples may be used. The advantage of this approach is that common memory allocation routines are used for all modules.

Accumulating Data with Process Modules

Some of the process modules, such as the ?? link Ana_Histogram module, are designed to accumulate data. When used within a loop, the results of their operation are added to the results of previous passes through the loop and accumulated in the process-*EarObjects* output signal. It is therefore necessary to ensure that the process *EarObject’s* output signal is set to zero, or *reset* before the beginning of the loop: using the ResetProcess_EarObject routine, or the *reset* command within simulation scripts. Process modules that accumulate data are automatically set to zero at the beginning of runs. This means that if such a module is used outside of a loop, then it need not be explicitly reset.

Using simulation scripts

Introduction to Simulation Scripts

Presently, the *simulation script* is the easiest method of producing a simulation. A single line defines each process in the simulation. It is also possible to use loops. Different simulations, describing different models, can be run using the same main driving program code. For example, program code for calculating frequency threshold tuning curves - see the filter shape test routine in the ’AutoTest.c’ test program is used for any model simply by changing the *simulation script*. It doesn’t matter whether it is the Carney ([5]) or the Meddis and Hewitt ([22]) auditory nerve models, a linear or non-linear basilar membrane model, and so on. This allows a great deal of flexibility, and the opportunity to re-use code.

Overview

Simulation scripts are used with the ’Util_SimScript’’ utility module (see the section called “Util_SimScript: Simulation Script Interface”). The simulation script is one of the parameters for the module, and is defined by a series of process lines surrounded by ’begin {...}’, e.g.

begin {

}

Any text after the last curly bracket, '{' is ignored. Blank lines in a simulation script are ignored, and can thus be inserted to ensure the specification is as readable as possible.

The indentation scheme employed in the examples given is purely cosmetic. However, the use of indentation enhances the clarity of the script.

The simulation script can contain comment lines. Any text after a hash, '#' character is ignored (until the end of the line is reached). Comments cannot follow onto the next line, so the only way to have comments on successive lines is to begin each comment line with a hash, '#' character. See the simulation script below for examples of comments.

A simple simulation may be described as a process *pipeline*. The process pipeline is first set up, then each of the process modules is initialised by it respective parameter file. With repeated runs, the initialisation of the process module parameters is only carried out once, on the first run.

Process lines

A single line describes each process in the simulation script, e.g.

Figure 2.1. Simple Example

```
# A simple simulation script
begin {

   Stim_PureTone   < PTone1.par   # Stimulus module
   Ana_ModulusFT

}
```

The example simulation script above generates a pure tone, which is then Fourier transformed. Each process line contains the module name, followed by the respective file name from which the module's parameters are to be read. These two elements of the process line are separated by the less-than, '<' character-indicating the direction of parameter 'flow' (from the .par file into the module). Modules, which do not have any parameters, need not have the less-than, '<' character or the parameter file name, as shown by the second line in the simulation script example.

Looping using 'Repeat'

There are occasions when it is necessary to repeat a particular section of a simulation. This can be done using the *repeat* loop instruction, enclosing everything to be repeated in curly brackets.

Figure 2.2. Repeat Example

```
# A simple simulation script
begin {
   Stim_PureTone     < PTone1.par   # Stimulus module
   Filt_BandPass     < PreEmph1.par
   BM_GammaT        < GammaT1k.par
   IHC_Meddis86      < Meddis96.par
   repeat 60 {
      AN_SG_Simple   < SpikeGen1.par
      Ana_Histogram  < PSTHHist1.par
   }
}
```

Note that the number of required repeats is inserted after the word, ’repeat’, but before the opening curly bracket, ’{’.

Reseting processes and Labels

The simulation script example shown in Figure 2.2, “Repeat Example” will work, as long as it is only run once. However, the ’Ana_Histogram’ module continues to accumulate data on successive runs, because it is not being reset at any point. The histogram process can be reset, (outside of the loop) using the ’reset’ command and by assigning the process line a label. An example of this is given in Figure 2.3, “Reset Example”.

Figure 2.3. Reset Example

```
# A simple simulation script
begin {
   Stim_PureTone      < PTone1.par   # Stimulus module
   Filt_BandPass      < PreEmph1.par
   BM_GammaT         < GammaT1k.par
   IHC_Meddis86       < Meddis96.par
   reset hi
   repeat 60 {
      AN_SG_Simple    < SpikeGen1.par
   hi% Ana_Histogram < PSTHHist1.par
   }
}
```

Nested Simulation Scripts

Simulation scripts can be nested-you can use the ’Util_SimScript’ module within a simulation script. An example of this is given in Figure 2.4, “Nested Simulation Script”.

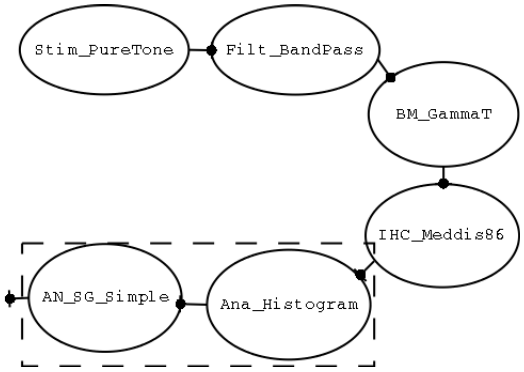
Figure 2.4. Nested Simulation Script

```
# A simple simulation script
begin {
   Util_SimSpec      < ANModel.sim
   repeat 60 {
      Util_SimSpec   < BushyCell.sim
   }
}
```

Complex Connections in Simulation Scripts

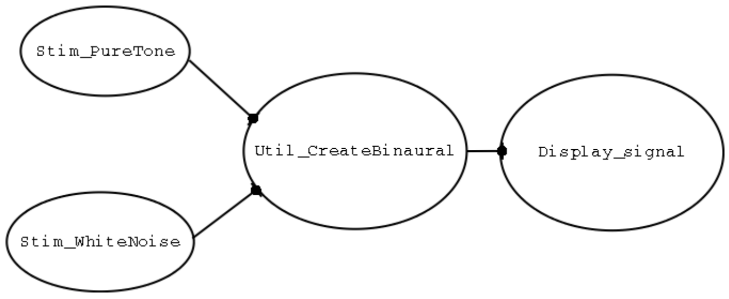
It is possible to create simulation scripts in which the connectivity between the process modules is more complicated than a simple pipeline. In the pipeline form described above the output of a process is used as the input of the next process in the line. The diagram in Figure 2.5, “Simple ’pipeline’ simulation schematic” represents the simple ’pipeline’ simulation in the simulation script shown in Figure 2.2, “Repeat Example”. The processes within the dotted line box are those within the ’repeat’ loop, as described above in section the section called “Looping using ’Repeat’ ”

Figure 2.5. Simple ’pipeline’ simulation schematic



Some process modules, such as the ’Util_CreatreBinaural’ utility, ’Neur_HHuxley’ neural cell or ’Ana_SAI’ stabilised auditory image can accept inputs from more than one input process. For instance, the ’Neur_HHuxley’ process module can accept excitatory, inhibitory and shunt-inhibitory process inputs. It can therefore be connected to three different inputs. A simpler example to consider is the ’Util_CreateBinaural’ process module, which requires two inputs from which it creates a binaural output. A schematic showing its use is given below in Figure 2.6, “Schematic of directed connections in a simulation using ’Util_CreateBinaural’”.

Figure 2.6. Schematic of directed connections in a simulation using ’Util_CreateBinaural’



Creating a script for such a simulation cannot use the simple pipeline assumption of a process accepting inputs from the previous process because the ’Util_CreateBinaural’ process module requires two inputs. In this case the inputs to the ’Util_CreateBinaural’ process must be specified explicitly. To direct the outputs for a process the following convention must be used:

label% <process_name> ([<input label list>]->[<output label list>]) ...

The *label* is case insensitive and must be present whenever the inputs or outputs for a process are specified. The *process name* is also case insensitive and should correspond to one of the library (or user) process modules. The *input and output label lists* are comma-separated label lists of other processes supplying input to or expecting output from the process. If either the input or output lists are omitted, then the default pipeline connection will be made, if possible.

The simulation script that corresponds to the schematic shown in Figure 2.6, “Schematic of directed connections in a simulation using ’Util_CreateBinaural’” is given below in Figure 2.7, “Simulation script showing directed connections using ’Util_CreateBinaural’”

Figure 2.7. Simulation script showing directed connections using ’Util_CreateBinaural’

```
# A simple simulation script

begin {

s1%   Stim_PureTone       (->cb)   < PTone2.par
s2%   Stim_WhiteNoise      (->cb)   < WNoise1.par
cb%   Util_CreateBinaural  (s1,s2->)
     Display_Signal

}
```

Please note that the formalism provided above for making connections between processes can be very unwieldy. However, it is general and allows complex and useful interconnection patterns.

In the future it is expected that such simulations will be designed using a graphical user interface simulation design feature that has yet to be implemented. The connection formalism provides a method by which graphically designed complex simulations can be described using simulation scripts.

Using the Basilar Membrane Filter Modules

Introduction

The decomposition of stimuli into different frequency channels is an important stage in auditory processing. It is generally visualised as occurring at the basilar membrane. Accordingly the modules that simulate this processing stage are called the basilar membrane filter modules . Within the DSAM library, these modules are distinctive because it is at this point that the number of channels representing a signal can be increased.

There are several different basilar membrane filter models implemented in the DSAM library, ranging from the simple gamma-tone filterbank to the more complex composite models, made up of multiple-filter components. There are two important requirements that are common to all of the basilar membrane filter modules, even though each particular model has its own specific parameters. The first of these is the centre frequency for each filter of the model producing an output channel. The second is the respective bandwidth at each centre frequency. The DSAM library provides a simple interface that is used by all of the basilar membrane modules to control these two requirements. This interface can be accessed from the basilar membrane module parameter files. This mode of use is described below, and example parameter files are given below.

Example 2.1. BM_GammaT Single Channel A parameter file for a single filter at 1 kHz.

```
CASCADE      4                   Filter cascade.
#CF List Parameters:-
DIAG_MODE  LIST                  Diagnostic mode ('list' or 'parameters').
CF_MODE      SINGLE              Centre frequency mode ('single', 'ERB', 'ERB_n', 'log', 'linear',
                                 'focal_log', 'user', 'human', 'cat', 'chinchilla', 'guinea-pig' or
                                 'macaque').
SINGLE_CF    1000                Centre Frequency (Hz).
B_MODE       ERB                 Bandwidth mode ('ERB', 'Custom_ERB', 'Guinea_Pig', 'user' or
                                 'Nonlinear').
```

Example 2.2. BM_GammaT Mulitple ChannelsA parameter file for a filter-bank with 20 filters ranging from 80-4500 Hz, equally spaced on a logarithmic scale.

```
CASCADE      4                   Filter cascade.
#CF List Parameters:-
DIAG_MODE  LIST                  Diagnostic mode ('list' or 'parameters').
CF_MODE      LOG                 Centre frequency mode ('single', 'ERB', 'ERB_n', 'log', 'linear',
                                 'focal_log', 'user', 'human', 'cat', 'chinchilla', 'guinea-pig' or
                                 'macaque').
MIN_CF       80                  Minimum centre frequency (Hz).
MAX_CF       4500                Maximum centre frequency (Hz).
CHANNELS     20                  No. of centre frequencies.
B_MODE       ERB                 Bandwidth mode ('ERB', 'Custom_ERB', 'Guinea_Pig', 'user' or
                                 'Nonlinear').
```

Both parameter files specify the equivalent rectangular bandwidth (ERB) bandwidth mode.

Centre Frequency modes

There are several centre frequency modes available, and the mode chosen defines how many parameters can be set in the parameter file. The centre frequency mode is specified by the 'CF_MODE' parameter, as shown in Example 2.1, “BM_GammaT Single Channel A parameter file for a single filter at 1 kHz.” and Example 2.2, “BM_GammaT Mulitple ChannelsA parameter file for a filter-bank with 20 filters ranging from 80-4500 Hz, equally spaced on a logarithmic scale.”. In the case of Example 2.1, “BM_GammaT Single Channel A parameter file for a single filter at 1 kHz.” where the 'SINGLE' mode is used, only one parameter is required, i.e. the centre frequency (CF). In Example 2.2, “BM_GammaT Mulitple ChannelsA parameter file for a filter-bank with 20 filters ranging from 80-4500 Hz, equally spaced on a logarithmic scale.” the 'LOG' centre frequency mode is used, which will produce a filter-bank with a range of centre frequencies equally spaced on a logarithmic scale. It requires three parameters i.e. the initial and final centre frequencies and the number of filter channels required for the filter-bank. These values are specified by the 'MIN_CF', 'MAX_CF' and 'CHANNELS' parameters respectively.

The centre frequency mode can be specified in upper or lower case, thus 'Single', 'single' and 'SINGLE' are equivalent. The order of the parameters after the mode parameters ('CF_MODE' or 'BW_MODE') is not important, however the mode should be specified first. As with all parameter files, only the parameter name and value is significant. The remaining text on a line is ignored, and can be used to supply descriptive comments. White space must separate the parameter name, value and any additional text.

The available frequency modes are given in Table 2.1, “Centre Frequency (CF) Modes for the basilar membrane filter modules (part 1).” and Table 2.1, “Centre Frequency (CF) Modes for the basilar membrane filter modules (part 1).”. Programmers should note that the ’UtCFList’ utility module controls the centre frequencies.

Table 2.1. Centre Frequency (CF) Modes for the basilar membrane filter modules (part 1).

Mode	Description	Required Parameters
Cat	Filter-bank with centre frequencies equally spaced using the Greenwood function (see the section called “The Greenwood CF Spacing Function”).	• *MIN_CF*, Initial filter-bank CF (Hz). • *MAX_CF*, Final filter-bank CF (Hz). • *CHANNELS*, Number of filter-bank output channels (integer).
Chinchilla	Filter-bank with centre frequencies equally spaced using the Greenwood function (see the section called “The Greenwood CF Spacing Function”).	• *MIN_CF*, Initial filter-bank CF (Hz). • *MAX_CF*, Final filter-bank CF (Hz). • *CHANNELS*, Number of filter-bank output channels (integer).
ERB	Filter-bank with centre frequencies equally spaced on an equivalent rectangular bandwidth (ERB) scale (Glasberg & Moore, 1990).	• *MIN_CF*, Initial filter-bank CF (Hz). • *MAX_CF* Final filter-bank CF (Hz). • *ERB_DENSITY*, ERB rate (Hz).
ERBN	Filter-bank with centre frequencies equally spaced on an equivalent rectangular bandwidth (ERB) scale (Glasberg & Moore, 1990). This mode allows the user to specify the number of channels directly.	• *MIN_CF*, Initial filter-bank CF (Hz). • *MAX_CF* Final filter-bank CF (Hz). • *CHANNELS*, Number of filter-bank output channels (integer).
Guinea_Pig	Filter-bank with centre frequencies set using the Greenwood function (see the section called “The Greenwood CF Spacing Function”).	• *MIN_CF*, Initial filter-bank CF (Hz). • *MAX_CF* Final filter-bank CF (Hz). • *CHANNELS*, Number of filter-bank output channels (integer).
Human	Filter-bank with centre frequencies equally spaced using the Greenwood function (see the section called “The Greenwood CF Spacing Function”).	• *MIN_CF*, Initial filter-bank CF (Hz). • *MAX_CF* Final filter-bank CF (Hz). • *CHANNELS*, Number of filter-bank output channels (integer).

Table 2.2. Centre Frequency (CF) Modes for the basilar membrane filter modules (part 2).

Linear	Filter-bank with centre frequencies equally spaced on a linear scale.	• *MIN_CF*, Initial filter-bank CF (Hz). • *MAX_CF* Final filter-bank CF (Hz). • *CHANNELS*, Number of filter-bank output channels (integer).
LOG	Filter-bank with centre frequencies equally spaced on a logarithmic scale.	• *MIN_CF*, Initial filter-bank CF (Hz). • *MAX_CF* Final filter-bank CF (Hz). • *CHANNELS*, Number of filter-bank output channels (integer).
Macaque	(Monkey) Filter-bank with centre frequencies set using the Greenwood function (see the section called "The Greenwood CF Spacing Function").	• *MIN_CF*, Initial filter-bank CF (Hz). • *MAX_CF* Final filter-bank CF (Hz). • *CHANNELS*, Number of filter-bank output channels (integer).
Single	Single filter mode	*SINGLE_CF*, Centre frequency (Hz).
User	Filter-bank with a custom centre frequency list, as specified by the user.	• *CHANNELS*, Number of filter-bank output channels (integer). • *CENTRE_FREQ*, 0:<CF1 Hz> • *CENTRE_FREQ*, 1:<CF1 Hz> • *CENTRE_FREQ*, 2:<CF1 Hz> • ... • *CENTRE_FREQ*, n:<CF1 Hz> -where 'n' is the number of channels specified by 'CHANNELS' less one.

The routines in the UtCFList utility module can also be used to produce frequency lists that can be used for purposes other than creating a filter-bank. For example, the routines are used for specifying the frequencies at which the output characteristics of a single filter are investigated in the filter tests in the AutoTest.c test program.

The Greenwood CF Spacing Function

The Greenwood function [9] allows the definition of filter centre frequencies that match data collected across a variety of species:

$$CF = A\left(10^{ax} - k\right)$$

Where *A* scales the value to give frequency in Hz. This value is varied between species to give the upper frequency limit of each species. Parameter, *a* is constant across all species tested. The parameter *x* is measured as a proportion of total basilar membrane length. It is measured in *mm* from the apical end, the value of *a* must be scaled by the total length of the species' basilar membrane). Parameter *k* determines the size of the divergence from log-linear, and also the lower frequency limit. The table below shows the parameters used for the respective modes in DSAM.

Species	**A**	**k**	*a*
Human	165.4	0.88	2.1
Cat	456	0.8	2.1
Chinchilla	163.5	0.85	2.1
Guinea Pig	350	0.85	2.1
Macaque Monkey	360	0.85	2.1

Bandwidth modes

The bandwidth mode defines the bandwidth of each filter. It is implemented in the same way as the centre frequency mode, so any readers who require more detail concerning the implementation should familiarise themselves with the description given above for the centre frequency modes.

The available bandwidth modes are given below in Table 2.3, “Bandwidth Modes for the BM Filter Modules.”. The ’UtBandwidth’ utility module controls the bandwidths.

Table 2.3. Bandwidth Modes for the BM Filter Modules.

Mode	**Description**	**Required Parameters**
ERB	Equivalent rectangular bandwidths (Glasberg & Moore, 1990).	None.
CUSTOM_ERB	Equivalent rectangular bandwidths (Glasberg & Moore, 1990). ERB = BwMin + 1/Quality * CF	’BW_MIN’, the minimum bandwidth (Hz). ’QUALITY’, the absolute quality factor for the filters.
CAT	Bandwidth formula fitted from experimental data (Kiang et al., 1965).	None.
GUINEA_PIG	Bandwidth formula fitted from unpublished data from Alan Palmer (Nottingham).	None.
USER	Custom bandwidths specified by the user.	Bandwidth list - one line for each bandwidth (Hz). The number of lines must correspond with the number of filter-bank output channels specified for the CF mode.
INTERNAL	The respective basilar membrane module defines bandwidths.	None.

The bandwidths, except for ’USER’ mode, are produce by a formula that relates the bandwidth to the filter centre frequency. It is therefore possible to add other bandwidth modes as required.

Using Parameter Arrays

It is sometimes necessary within a process module to define a series of values based upon a parametric function. This is done using the *Parameter Array* parameter input facility.

An example of this facility in use is in the ’BM_DRNL’ process module, where some of the parameters are defined as a function of the centre frequency. In each case, the parameter is defined by providing a function name, followed by a number of additional parameters as required by the function. This information is then used within the ’BM_DRNL’ process module, for example, to calculate the value of the non-linear bandwidth for each centre frequency.

Using the Segment-processing Mode

The *segment-processing* mode was introduced primarily by the request of users who need to process long signals. For example, people in the speech recognition field need to work with signal duration measured in seconds rather than milli-seconds. For this reason the segment-processing mode allows the processing of small or continuous signals in smaller segments. For instance:

- a two second signal can be processed in ten segments, two hundred milli-seconds long, avoiding the need for the use of virtual memory because of memory restrictions;
- Signals from a peripheral device running continuously can be processed in batches or *segments*.

All DSAM modules are configured so that they can operate continuously in segment-processing mode, with repeated calls. Any tables or *process variables* that the module requires will be created for the first run, thereafter these are kept up to date on subsequent calls to the routine. The nature of some of the stimulus routines governs how they interpret the segmented mode of processing:

- In segmented mode the click, step function and puretone_2 stimulus generation modules will produce a signal which repeats, i.e. the puretone_2 stimulus (a pure tone burst surrounded by silence) will repeat with ’silence’ intervals defined by the pre- and post stimulus ’silence’ periods. (Examples of the segmented mode processing behaviour are displayed in the respective module sections below). The other stimuli have a continuous response.
- Reading from files will be done in segments until the end of the file is reached.
- In segmented mode output will be added to the end of current files
- Module behaviour, as previously mentioned, is continuous under segmented mode. This means that the modules do not reset after each segment is processed, unless they are re-initialised or there is a change in the parameters - in segment-processing mode module processes need to be reset explicitly (see section the section called “Reseting processes and Labels ”).

. Some interesting animation effects can be achieved when the segment-processing mode is used in conjunction with the display module.

Using the Display Module

There is a simple graphical display interface available for use with the DSAM library. It was written using the public domain *wxWidgets* cross-platform GUI library. Once the library has been compiled with GUI support the *Display_Signal* module can be accessed when using *simulation scripts* with DSAM applications, or when using *generic* or *hard programming*.

The GUI support must be compiled using C++ and the wxWidgets library. The wxWidgets library can be used under X-windows (GTK, MOTIF or native X11), PC Windows (95, 98, NT, 2000 and XP) and there is also a Mac OS X version. See the URL www.wxWidgets.org [http://www.wxwidgets.org] for further information about wxWidgets. See the INSTALL file for information on installing DSAM with graphical support. Programs that use the display module can also be compiled using a DSAM library that does not have the graphics support included. In such a case the display module will be ignored.

See the section called “Display_Signal: Standard Signal Display” for further information on the use of the display module, and pointers to the test programs

Part II. Process Modules Reference

Introduction

Each module, file and each file’s routines are listed in this reference chapter. The ordering is alphabetic: by module name. The module routine names describe the routines’ function though additional explanations are provided where necessary.

Each process module has an illustrated example of its use, where appropriate. A simulation script is provided. The outputs shown for many of the illustrations can be produced using the AMS application. All the simulation scripts can be find in the ’AutoTest’ package available from the DSAM download page [http://www.essex.ac.uk/psychology/hearinglab/dsam/dloads_main.htm]. Some of the more complex figures, such as the tuning curves require repeated runs of the simulation script, and were produced using the ’AutoTest’ program. The simulation scripts for all of the process module tests can be found in the ’AutoTest’ package. The AutoTest package is an application designed to run tests on each process module. In general the test paradigms check the correct operation of the module, but in the case of the Models group tests have been designed to reproduce known results, most of which have been published. Thus investigations by users can begin from a known position.

Table of Contents

Chapter 3. Analysis Modules

Ana_ACF: Auto-correlation Function

File name: AnACF.[ch]

Description

This module generates an auto-correlation function from the connected input signal. It puts the result in the output signal, overwriting any previous data if the signal has already been initialised. The function is calculated backwards from the time, **OFFSET** (s) up to a lag of **MAX_LAG** (s). An exponential decay function is included with the time constant, **TIME_CONST** (s). The same binning as for the input signal is used.

Calculated by:

$$h(t,l)=\sum_{i=1}^{N} p(t-T)\,p(t-T-l)\exp(-T/\Omega)$$

$$T=idt$$

$$l=0\Rightarrow MAX_{LAG}-dt$$

where *t* is the time the ACF is sampled
l is the autocorrelation lag
dt is the sampling period
Ω is the time constant of integration (TIME_CONST)
p(x) is the spike probability at time *x*
N is the sample length of the integration window (equal to *3Ω/dt*).

Inputs Arbitrary single input as a function of time. E.g. membrane voltage or action potential stream.

Outputs Same number of channels as the input, but the length is defined by the maximum lag. The first point has a lag of 0, the last point has a lag of MAX_LAG - dt.

References Licklider J. C. R. (1959) ’Three auditory theories’ in Psychology: A Study of a Science, edited by S. Koch, MacGraw-Hill, New York.

Meddis R. and Hewitt M. J. (1991) "Virtual pitch and phase sensitivity of a model of the auditory periphery. I Pitch identification" J. Acoust. Soc. Am., 89, 2866-2882

Module Parameters

Example 3.1. Ana_ACF Parameter File

MODE	STANDARD	Normalisation mode (’none’, ’standard’ or ’unity’).
T_CONST_MODE	LICKLIDER	Time constant mode (’Licklider’ or ’Wiegrebe’.)
OFFSET	-1	Time offset, t0 (if -ve the end of the signal is assumed) (s).
TIME_CONST	0.0025	Time constant (or minimum time constant in ’Wiegrebe’ mode) (s).
T_CONST_SCALE	2	Time constant scale (only used in ’Wiegrebe’ mode).
MAX_LAG	0.0075	Maximum autocorrelation lag (s).

Table 3.1. Ana_ACF ’mode’ options

Mode	Description
Standard	This mode normalises the autocorrelation by dividing by the maximum lag index (the number of points along the lag axis). This is the standard method for autocorrelation.
Unity	This mode normalises the autocorrelation by dividing by the **TIME_CONST** parameter (i.e. the area under the integration window function), then taking the square root.

Table 3.2. Ana_ACF ’t_const_mode’ options

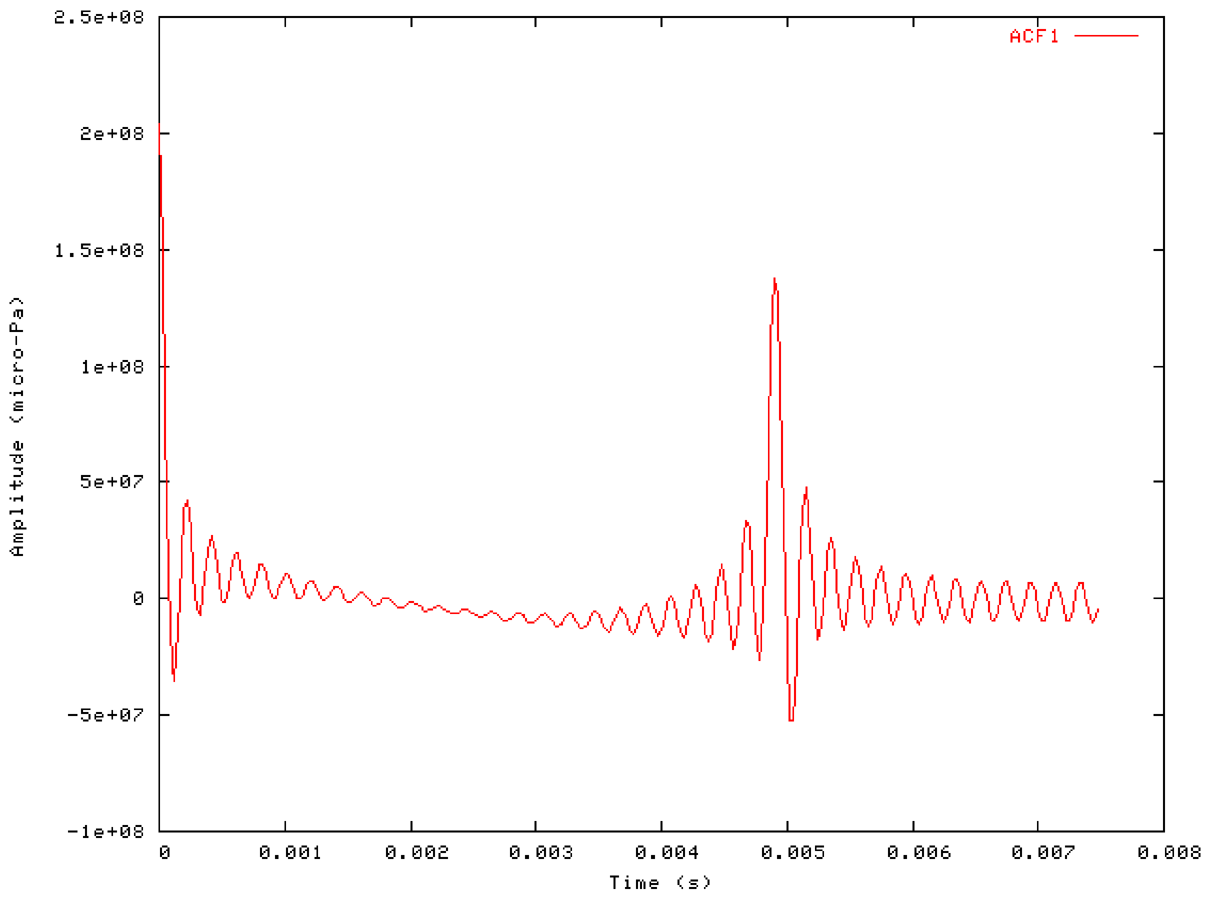
Mode	Description
Licklider	In this mode the time constant, tau has a constant value as defined by the **time_const** parameter.
Wiegrebe	The in this mode the time constant varies proportionally to the autocorrelation lag. The time constant is defined by the autocorrelation lag multiplied by the **time_const_scale**, subject to a minimum defined by the **time_const** parameter.

Examples Using the ’Ana_ACF’ Process Module

Example: AutoTest/A_Di/ACF1.sim

```
# This example simulation script with the associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

#ACF1.sim
begin {

   Stim_Harmonic   < Harmonic1.par
   Ana_ACF         < AutoCor1.par
   Display_Signal
   #DataFile_Out

}
```

Figure 3.1. ACF simulation outputThe stimulus is a harmonic series with alternating phases for the components.



Comments

The auto-correlation function is used to investigate periodicity in signal processing. A pure tone stimulus is generated in this test, and the section of the signal up to the ’offset’ is auto-correlated. The output of the module is the value of the autocorrelation at the time of the offset. The data is output to file with each line in the format:-

Time (s), ACF

Ana_ALSR: Averaged Localised Synchronised Rate

File name: AnALSR.[ch]

Description

This analysis module produces as its output the averaged localized synchronized rate (ALSR) function. It was designed to provide a representation of population responses of the auditory nerve to vowel sounds. The function combines rate, place and temporal information to provide a good representation of the vowel’s spectra. The function is stable with changes in sound level at least up to 80 dB SPL in animal recordings. This is in sharp contrast to the representation of vowels’ spectra in terms of auditory nerve discharge rate that degenerates at such high levels.

NOTE: This is a complex statistic. Please read Young and Sachs, 1979 (see References below) before proceeding.

The ALSR for the kth harmonic of the vowel is defined as:

$$ALSR(k) = \frac{1}{M_k} \sum_{l \in C_k} R_{kl}$$

ALSR function

Where Rkl is the Fourier magnitude of the kth component of the response of the lth channel,
Ck is the set of channels with CFs close to kf0, (how close is determined by UPPER_LIMIT and LOWER_LIMIT)
Mk is the number of channels in Ck,
f0 is the fundamental frequency of the vowel.

Input to the module must be in the form of a period histogram for each channel. The PH must be fixed to the period of the stimulus (or integer multiples of this period).

The frequencies sampled by the ALSR are determined by the period of the PH. The inverse of the period (in seconds) gives the frequency spacing. E.g. 0.01 s gives 100 Hz spacing.

In addition, the number of bins in the preceding histogram should be a power of 2 in order to get this spacing. The bin width should be chosen such that the number of bins across the PH is a power of 2. For example, a PH period of 10 ms, bin width of 0.078125 ms gives 128 bins (2^7).

The channel CFs should be chosen to cover the range of ALSR frequencies. If ALSR sampling begins at X Hz, channel CFs should be included down to X * (2^UPPER_LIMIT), or no value for X is calculated. The highest frequency at which the ALSR is calculated is determined by the highest channel CF. The maximum is Y = Max_CF * (2^LOWER_LIMIT).

Inputs	Multi-channel Period Histogram
Outputs	Single channel ALSR function vs Frequency (Hz).
Reference	Young E.D. and Sachs M.B (1979) "Representation of steady-state vowels in the temporal aspects of the discharge patterns of populations of auditory-nerve fibers", J Acoust Soc Am, 66:1381-1403

Module Parameters

Example 3.2. Ana_ALSR parameter file

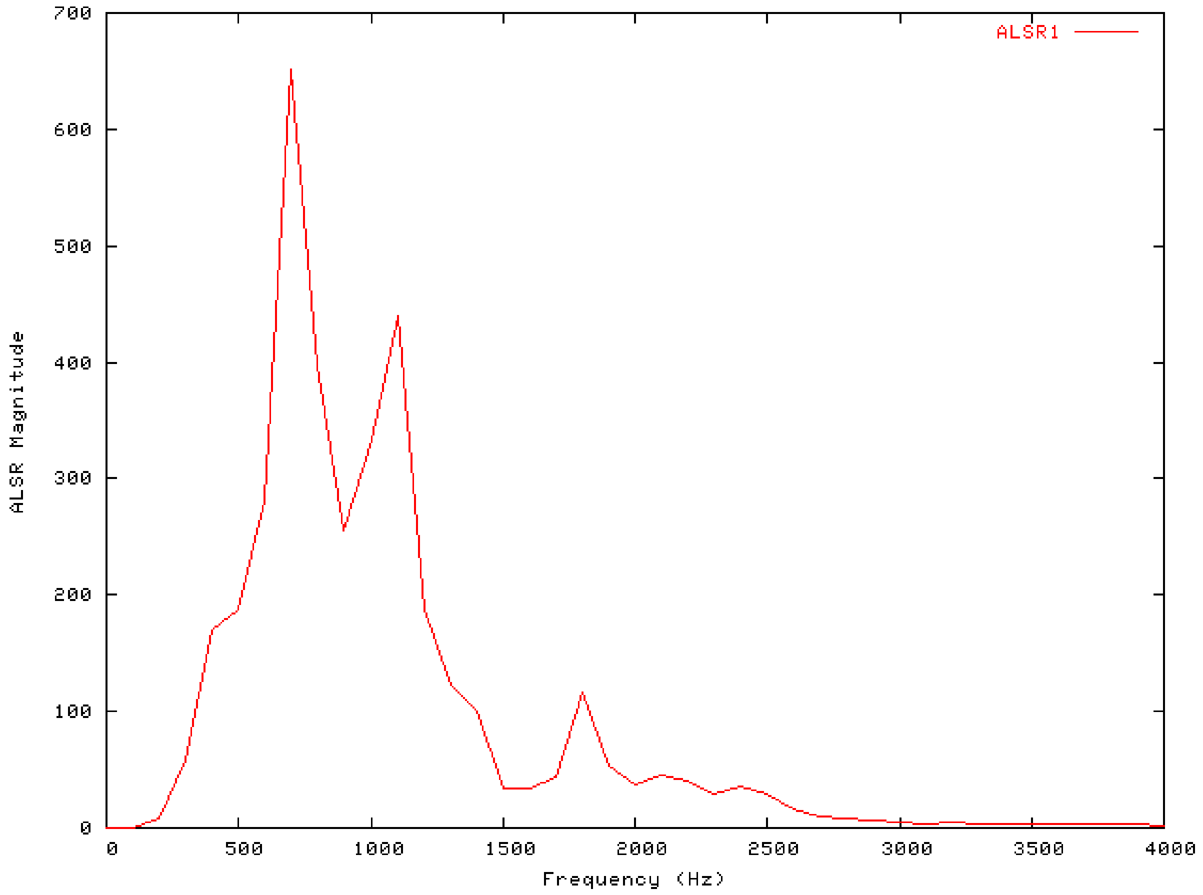
LOWER_LIMIT	-0.25	Averaging window limit below CF (octaves)
UPPER_LIMIT	0.25	Averaging window limit above CF (octaves)
NORMALISE	1	Normalisation factor (units.)

Examples Using the ’Ana_ALSR’ Process Module

Example: AutoTest/A_Di/ALSR1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.
# ./A_Di/ALSR1.sim
begin  {

 DataFile_In     < DFIn1.par
 Util_ReduceDT    < UpSample.par
 Trans_gate      < RampOn.par
 Filt_MultiBPass  < MBPass_GP4a.par
 Filt_MultiBPass  < MBPass_GP4b.par
 BM_DRNL         < DRNL_GP4.par
 Display_Signal
 IHCRP_Shamma3StateVelIn < IHCRP_GP3.par
 IHC_Meddis2000   < Med2000MSRp1.par
 reset hi
 repeat 50 {
 AN_SG_Carney     < ANCarney_GP4.par
 hi% Ana_Histogram    < ALSR_PH2.par
 }
 Display_Signal
 Ana_ALSR        < ALSR1.par

}
```

Figure 3.2. ALSR function for the /a/ vowel



Comments:

Please note that the above simulation takes some time to run.

Ana_Averages : Channel Averaging

File name: AnAverages.[ch]

Description

This routine calculates the average signal value over a specified time range and time offset, for each channel of the input signal. The result for each channel is returned in the first sample output signal channels.

Three different types of averages are calculated, according to the *mode* required: full, half wave rare averages, '-wave' and half-wave complement averages, '+wave'. (The -/+ wave options were revised from code from Trevor Shackleton, MRC, Nottingham).

Inputs	Arbitrary single input
Outputs	The output signal has the same number of channels as the input, but with only one result sample per channel.
Reference	
See also	Util_BinSignal

Module Parameters

Example 3.3. Ana_Averages parameter file

```
MODE        FULL   Averaging mode ('full', '-wave' or '+wave').
OFFSET      0      Time offset for start of averaging period (s).
DURATION    -1     Averaging period duration: -ve assumes to end of signal (s).
```

Table 3.3. Ana_Averages 'Mode' options

Mode	Description
full	A full average for each channel of the signal.
-wave	An average of only the negative components of the channels.
+wave	An average of only the positive components of the channels.

Examples Using the 'Ana_Averages' Process Module

Example: AutoTest/A/Averages.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A/Averages1.sim
begin  {

 Stim_PureTone   < PTone1.par
 Ana_Averages    < Average1.par
}
```

Comments

This particular simulation script produces no display. If the “DataFile_Out” module is uncommented, then the result will be output to the file “output.dat”. The output value should be approximately zero as the pure tone is averaged over the positive and negative cycles.

Ana_Convolution :

File name: AnConvolute.[ch]

Description

This routine creates convoluted signal using the output signals from two input processes. It is useful for creating complex stimuli as well as other convolution purposes. The processed output signal will be monaural or binaural (with interleaved channels: LRLRLR) depending upon the format of the two input signals, which must both have the same sampling interval and number of channels. The duration of the two signals need not be the same.

The convolution of R and S (R*S)j is calculated for each point j, by:

$$(R*S)_j = \sum_{k=1}^{N} R_j \cdot S_k$$

where N is the number of samples in S.

Inputs	Expects input from two similarly dimensioned processes as described above.
Outputs	The output signal with the same dimensions as the input.
Reference	
See also	Filt_FIR

Module Parameters

Example 3.4. Ana_Convolution parameter file

This module has no parameters

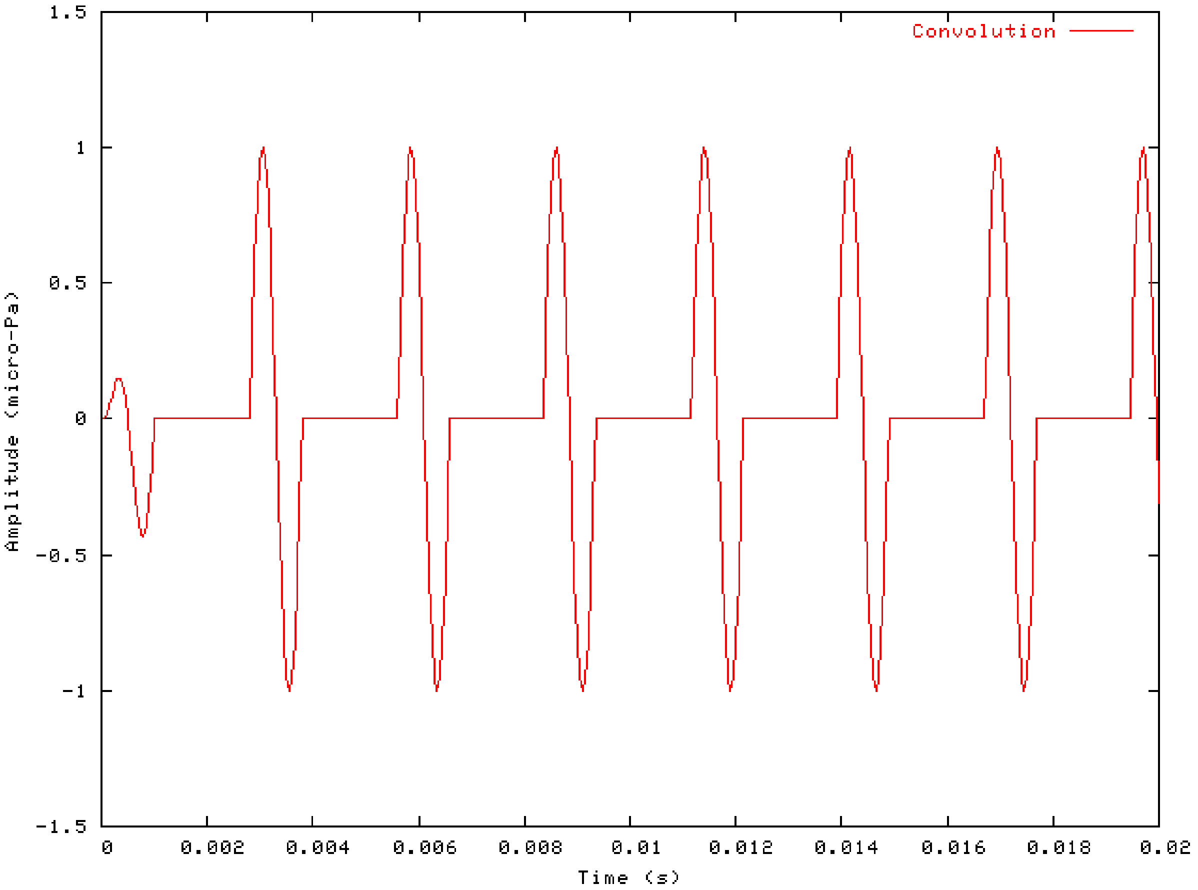
Examples Using the ’Ana_Convolution’ Process Module

Example: AutoTest/A/Convolution.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A/Convolution.sim
begin  {
```

```
click% Stim_PulseTrain (->conv) < PulseTr5.par
ptone% Stim_PureTone  (->conv) < PTone7.par
conv% Ana_Convolution (click, ptone->)
 Trans_Gate     < Ramp1.par
 Display_Signal

}
```

Figure 3.3. Output for convolution test. A signal cycle of a pure tone is convolved with a click train, creating a train of repeated pure tone pulses.



Comments:

The output should contain a pure tone, which begins at the click position. The output is given in the form:

time (s), convolution.

Ana_CCF : Cross-correlation Function

File name: AnCrossCorr.[ch]

Description

This module generates a cross-correlation function between the left and right ear components of the connected input signal. The input signal must be binaural. It puts the result in the output signal, overwriting any previous data if the signal has already been initialised. The function is calculated backwards from the time, **OFFSET** (s) for a period of **MAX_LAG** (s). The time offset must be more than MAX_LAG seconds from the start or end of the signal. An exponential decay function is included with the time constant, **TIME_CONST** (s). The same binning as for the input signal is used.

The cross correlation function c is:

$$c(t,l) = \sum_{i=-L}^{L} p1(t-T)\,p2(t-T-l)\exp(-T/\Omega)$$

- where T = idt, t is the offset at which the CCF is calculated, l is the cross-correlation lag, dt is the sampling period, W is the time constant of integration (TIME_CONST), p1(x) and p2(x) are the sample values in the two channels at time x and L is the number of points within the integration window (equal to MAX_LAG / dt).

Inputs	Binaural Signal
Outputs	Signal with the same number of channels, but with a length defined by twice the **MAX_LAG** parameter. The first point has a value of **-MAX_LAG**, the last point has a value of **+MAX_LAG**
Reference	Shackleton T. M., Meddis R. and Hewitt M. J. (1991) ’Across frequency integration in a model of lateralization’, J. Acoust. Soc Am. 91, 2276-2279.

Module Parameters

Example 3.5. Ana_CCF parameter file

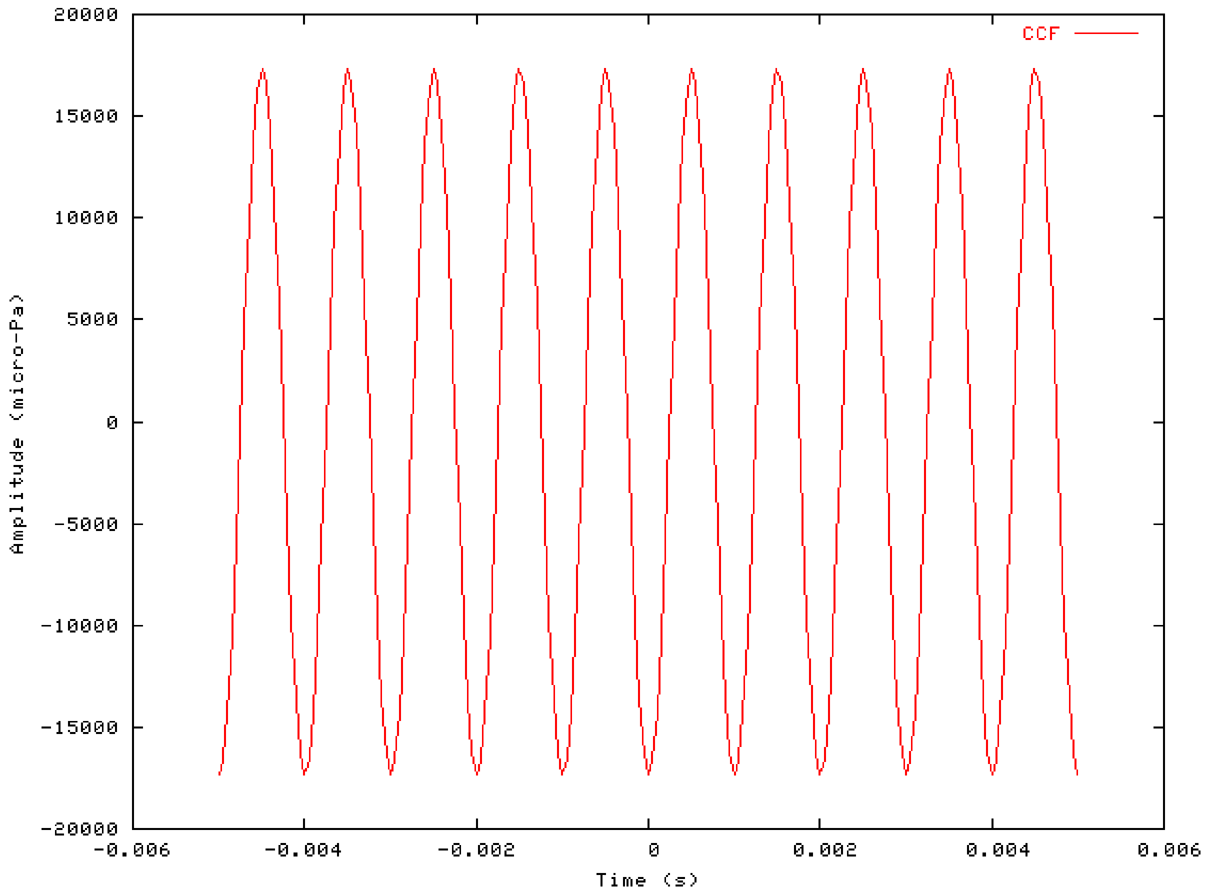
OFFSET	0	Time offset, t0 (s).
TIME_CONST	0.0025	Time constant, tw (s).
MAX_LAG	0.005	Maximum lag period, t (s).

Examples Using the ’Ana_CCF’ Process Module

Example: AutoTest/A/CCF.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A/CCF.sim
begin  {

 Stim_PureTone_Binaural < BPTone1.par
 Ana_CCF        < CrossCo1.par
 Display_Signal

}
```

Figure 3.4. CCF test outputOutput for CCF test. Cross correlation of dichotic 1000-Hz pure tones 180 degrees out of phase at the two ears.



Comments:

The output file is in the form:

delay period (s), CCF.

Ana_FindBin: Bin searching Analysis

File name: AnFindBin.[ch]

Description

This module bins adjacent values of a signal and finds the bin with the maximum or minimum bin value and returns either the value or the sample index of the bin. The search is implemented using a bin ’window’ of **BIN_WIDTH**. . The signal is searched forward from the **OFFSET** for a time set by the **WIDTH** parameter. In the case of the sample index being returned, it corresponds to the centre of the bin window with respect to the input signal.

The units of **BIN_WIDTH**, **OFFSET**, and **WIDTH** will usually be time (s). However, if the input comes, for instance, from Ana_FourierT, the units will be frequency (Hz), e.g. an **OFFSET** of 1000 Hz might be set.

Inputs	Arbitrary single input
Outputs	A signal with 1 sample result for each channel.
Reference	

Module Parameters

Example 3.6. Ana_FindBin parameter file

MODE	MIN_VALUE	Search mode (’max_value’, ’max_index’, ’min_value’ or ’min_index’).
BIN_WIDTH	-1	Bin width (time window) for search bin: -ve assumes dt for previous signal (s).
OFFSET	0	Offset from which to start search (s?).
WIDTH	-1	Analysis window width for search: -ve assume to end of signal (s?).

Table 3.4. Ana_FindBin ’mode’ options

Mode	Description
Max_value	Return the maximum value found.
Max_index	Return the sample index at which the maximum value was found.
Min_value	Return the minimum value found.
Min_index	Return the sample index at which the minimum value was found.

Examples Using the ’Ana_FindBin’ Process Module

Example: AutoTest/A/FindBin.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A/FindBin.sim
begin  {

 Stim_Click      < Click1.par
 Ana_FindBin     < FindBin1.par

}
```

Comments

This simulation script produces no display. If the ’DataFile_Out’ module is uncommented, then the result will be output to the file ’output.dat’. The output value should be 119 for this test.

Ana_FindNextIndex: Sample index Analysis

File name: AnFindNextIndex.[ch]

Description

This routine looks for a peak or a trough in the input signal. It finds the index position of the next minimum or maximum after a specified time, **OFFSET** (s) for each channel of its process input signal. The result for each channel is returned in the output signal. Note that the returned value is the index, not the value. The algorithm looks for the first minimum after a negative gradient. A negative gradient is where cPtr(i + 1) - cPtr(i) < 0.

This module has many uses but could be used, for example, to find the peak value in a histogram after a specified time.

Inputs	Arbitrary single input
Outputs	A signal with 1 sample result for each channel.
Reference	
See also	the section called “Ana_FindBin: Bin searching Analysis ”

Module Parameters

Example 3.7. Ana_FindNextIndex parameter file

MODE	MAXIMUM	Index search mode (’minimum’ or ’maximum’).
OFFSET	0	Time offset for start of search (s).

Table 3.5. Ana_FindNextIndex ’mode’ options

Mode	Description
minimum	Return the sample index of the mnimum value found.
maximum	Return the sample index of the maximum value found.

Examples Using the ’Ana_ACF’ Process Module

Example: AutoTest/A_Di/FindInd1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A_Di/FindInd1.sim
begin  {

 Stim_Harmonic    < Harmonic1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_GammaT       < GammaTLog1.par
 IHC_Meddis86     < Meddis86.par
 AN_SG_Simple     < SimpleSG1.par
 Display_Signal
 Ana_FindNextIndex < FindInd1.par

}
```

Comments

This simulation script produces no display. If the ’DataFile_Out’ module is uncommented, then the result will be output to the file ’output.dat’. The file should contain the following:

```
Time (s)      100       215.443    464.159    1000       2154.43    4641.59    1000
2.50000e-05   198       21         73         44         116        12         83
```

Ana_FourierT: Fourier Analysis Function

File name: AnFourierT.[ch]

Description

This function calculates the modulus or phase of the Fourier transformation of a real data set. A Fast Fourier Transform algorithm is used. The routine generates the modulus and other FFT properties of the process *EarObject’s* input signal. It puts the result in the output. It also changes the sampling interval, dt field, of the output signal to the appropriate frequency interval. This frequency interval (DeltaF) is calculated from the duration of the input signal.

ΔF = 1/Duration

This only holds if the number of samples in the signal (duration / dt) is a power of 2 (necessary for an FFT). If this is not so,

ΔF = (1/Duration) * (no. of samples/next largest power of 2)

For example, 200-ms signal, sample rate 100 kHz. No. of samples = 20 000, therefore next power of 2 is 32,768. Therefore ΔF = (1/0.2) * (20 000/32 768) = 3.05177 Hz.

Inputs	Arbitrary single input
Outputs	Produces the FFT output with the same channel and length dimensions as the input, however the time dimension is now the replaced by the frequency interval. In ’complex’ output mode, each channel if input produces two channels of output for both the ’real’ and ’imaginary’ complex components.
Reference	Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). Numerical Recipes in C, 2nd Ed. pp 504-508.

Module Parameters

Example 3.8. Ana_FourierT parameter file

```
OUTPUT_MODE  MODULUS  Output mode: ’modulus’, ’phase’, ’complex’ or ’dB_SPL’
                      (approximation).
```

Table 3.6. Ana_FourierT ’output’ mode options

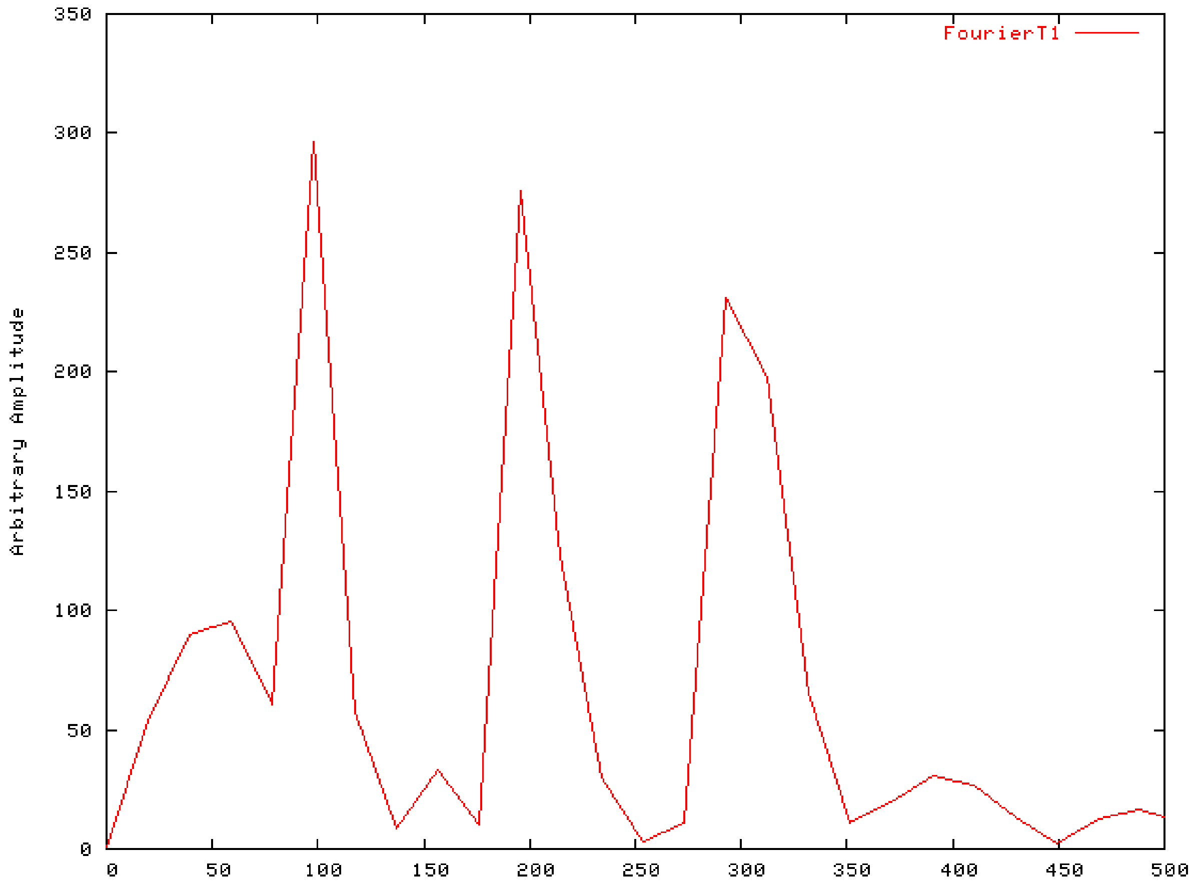
Mode	Description
DB_SPL	This mode approximates the dB SPL output for each frequency component of the FFT. If the input is a pure tone at one of the frequency values for which the FFT is defined, then the output will be the intensity of the pure tone in dB SPL.
Complex	This mode outputs two channels for each input channel. The two channels contain the 'real' and 'imaginary' components of the FFT for each sample respectively.
Modulus	In this mode the output is the FFT modulus
Phase	This mode outputs the FFT raw phase - no phase unwrapping is done.

Examples Using the 'Ana_FourierT' Process Module

Example: AutoTest/A/FourierT1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A/FourierT1.sim
begin {

 Stim_PureTone_Multi < MPTone1.par
 Ana_FourierT    < FourierT1.par
 Display_Signal
 #DataFile_Out

}
```

Figure 3.5. Fourier Transform process outputThe frequency data for this figure has been truncated for display purposes.



Comments

This analysis process module is used for the analysis of the frequency components in a waveform.

Ana_Histogram: Histogram analysis function

File name: AnHistogram.[ch]

Description

This routine generates a period histogram (PH) or a post stimulus time histogram (PSTH) from the process' input. It supplies the result as its output signal, adding to previous data if the process has already been run. Accurate binning must be achieved, so real values must be used in determining the **BIN_WIDTH** (s), bin widths. In 'spike detection' mode the histogram events are calculated from the first minimum after the specified **OFFSET** (s) of the input signal. There are several operating mode options. Not all of the parameters are used for all modes, i.e. In the 'continuous' detection mode, the **THRESHOLD** parameter is not used.

This module accumulates data cross loops until the process is reset - see the section called "Accumulating Data with Process Modules".

Please note that in PH mode accurate binning must be achieved; **PERIOD** / **BIN_WIDTH** (s) must be equal to an integer number of bins. If this is not so, the period will no longer match that desired and accumulation of data will not be accurate. The histogram is calculated from the first minimum after the specified **OFFSET** (s) of the input signal.

Inputs	Arbitrary single input
Outputs	A histogram for each input channel, with the length defined by the **BIN_WIDTH** and **PERIOD** parameters (depending upon the **TYPE_MODE** option selected. The first point has a time of **BIN_WIDTH** and the last point has a time of **PERIOD** or stimulus duration.
Reference	

Module Parameters

Example 3.9. Ana_Histogram parameter file

DETECT_MODE	SPIKE	Detection mode: ’spike’ (detection) or ’continuous’ (signal).
OUTPUT_MODE	BIN_COUNTS	Output mode: ’bin_counts’ or ’spike_rate’.
TYPE_MODE	PSTH	Histogram type: ’PSTH’, or ’PH’.
THRESHOLD	0	Event threshold for bin counts (units as appropriate).
BIN_WIDTH	-1	Histogram Bin width (-ve: previous signal dt, zero: one bin = input signal length) (s).
PERIOD	-1	Period for period histogram: used in ’PH’ mode only (s).
OFFSET	0	Time offset for histogram (s).

Table 3.7. Ana_Histogram Detection Mode Option

Detection Mode	Description
spike	This detection mode counts positive-going **THRESHOLD** crossing events in the signal.
continuous	This detection mode bins the input without further processing (i.e. sums all adjacent values within the bin range).

Table 3.8. Ana_Histogram Output Mode Options

Output Mode	Description
bin_counts	
spike_rate	With this mode the output is given as a spike rate (spikes/s). This option can only be used with the ’spike’ detection mode.

Table 3.9. Ana_Histogram Histogram Type Options

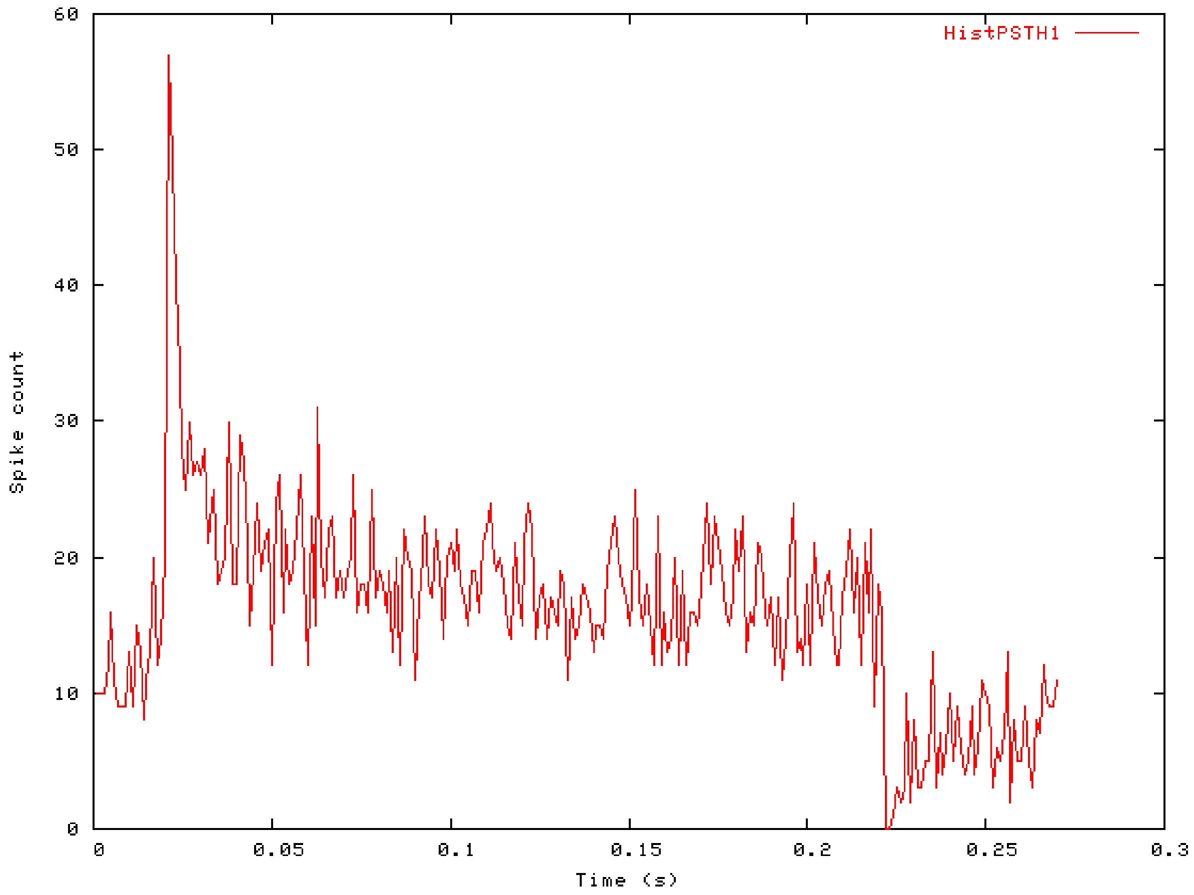
Histogram Type	Description
psth	The post-stimulus time histogram option. The output signal length will depend upon the **BIN_WIDTH**, the **OFFSET** and the length of the input signal.
ph	The period histogram option. The output signal length will depend upon the **BIN_WIDTH** and the **PERIOD** parameters.

Examples Using the ’Ana_Histogram’ Process Module

Example: AutoTest/A_Di/HistPSTH1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A_Di/HistPSTH1.sim
begin  {

 Stim_Puretone_2  < P2Tone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 IHC_Meddis86    < Meddis86.par
 repeat 20 {
 AN_SG_Simple    < SimpleSG1.par
 Ana_Histogram    < PSTH1.par
 }
 Display_Signal

}
```

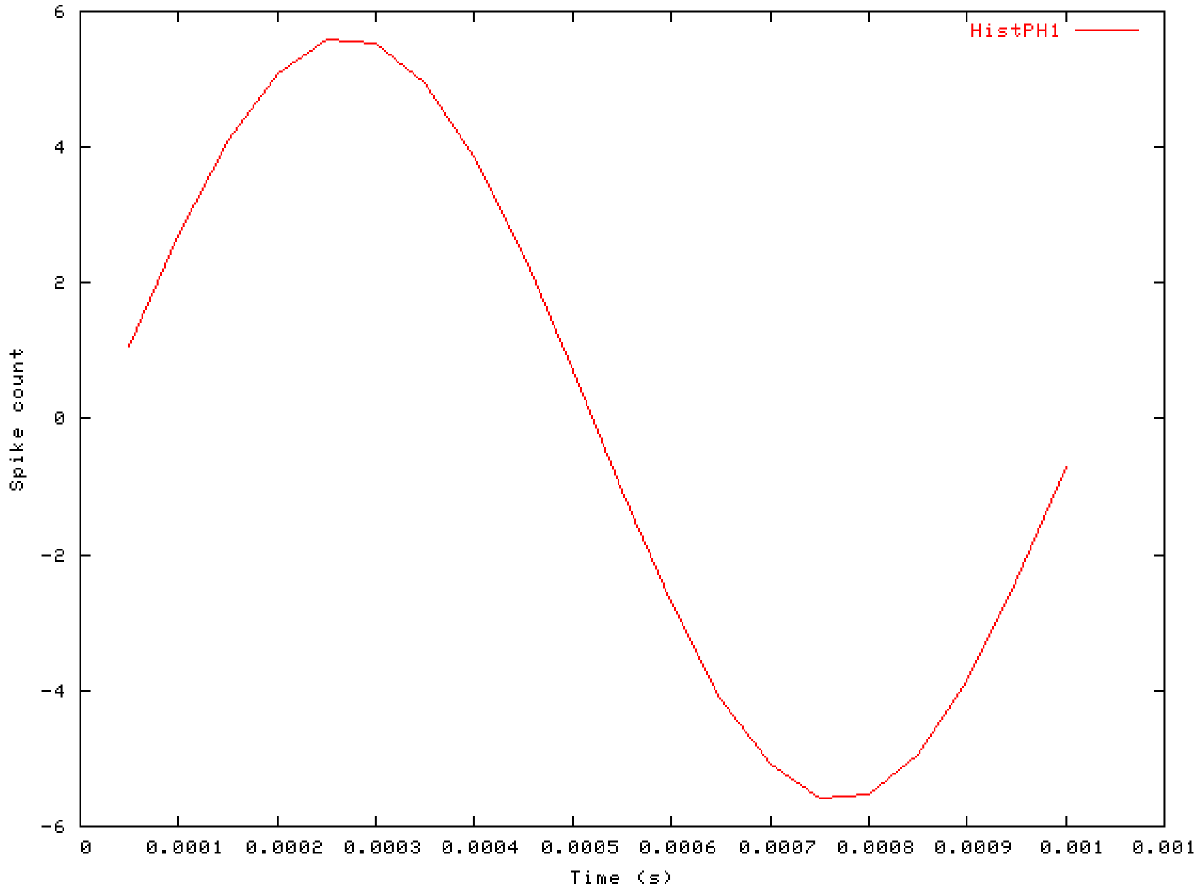
Figure 3.6. Histogram output for HistPSTH1.simThe figure shows the puretone response from a linear auditory nerve model.



Example: AutoTest/A/HistPH1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A/HistPH1.sim
begin  {

 Stim_PureTone    < PTone3.par
 Ana_Histogram    < HistPH1.par
 Display_Signal

}
```

Figure 3.7. PH Output from Histogram function



Comments:

The period histogram, like the auto-correlation function, is used to investigate periodicity effects in signals. In constructing a period histogram the occurrence of each spike is summed into a histogram bin corresponding to time, but the time axis is reset for a constant period on the stimulus time scale.

Ana_Intensity: RMS Intensity Analysis Function

File name: AnIntensity.[ch]

Description

This routine calculates the (rms) intensities for each channel of the process *EarObject's* input signal. The result for each channel is output in the respective first sample of the output signal in dB SPL. The intensity calculation starts from the **offsetTime** (s). Please note that the intensity calculation is only accurate for pure tone signals. For other signals it can be used as a guide.

Intensity I is calculated by:

$$I(x)=20.0\times\log_{10}\left(\frac{\sqrt{\sum_{i=1}^{N} x_{i}^{2}/N}}{20}\right)$$

-where x is the input signal and N is the number of samples in the input.

Inputs	Arbitrary single input, but to produce an output in dB SPL, the input signal must be pressure amplitudes in micro-pascals (mPa).
Outputs	A single result sample in for each channel in dB re. 20 microPa.
Reference	

Module Parameters

Example 3.10. Ana_Intensity parameter file

OFFSET	0.0025	Time from which to start calculation (s).
EXTENT	-1	Time over which calculation is performed: -ve value assumes end of signal (s).

Examples Using the 'Ana_Intensity' Process Module

Example: AutoTest/A/Intensity.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A/Intensity.sim
begin {

 Stim_PureTone   < PTone5.par
 Ana_Intensity   < Intensty1.par

}
```

Comments:

This simulation script produces no display. If the ’DataFile_Out’ module is uncommented, then the result will be output to the file ’output.dat’. The output value should be 1 (db SPL) for this test.

Ana_ISIH: Inter-spike Interval Histogram

File name: AnInterSIH.[ch]

Description

This routine generates an inter-spike interval histogram from the input signal. A spike event is counted for every maximum following a positive-going crossing of **THRESHOLD**. The **ORDER** parameter defines how many spikes beyond the originating spike should be included in the analysis. Note that an **ORDER** of -1, all-order intervals, calculates an autocorrelation function.

This module accumulates data cross loops until the process is reset - see section the section called “Accumulating Data with Process Modules”

Inputs	Arbitrary single input.
Outputs	An ISIH for each channel, the length being defined by the **MAX_INTERVAL** parameter and the sampling interval of the input signal. The ISIH uses the same sampling interval as for the input signal.
Reference	

Module Parameters

Example 3.11. Ana_ISIH parameter file

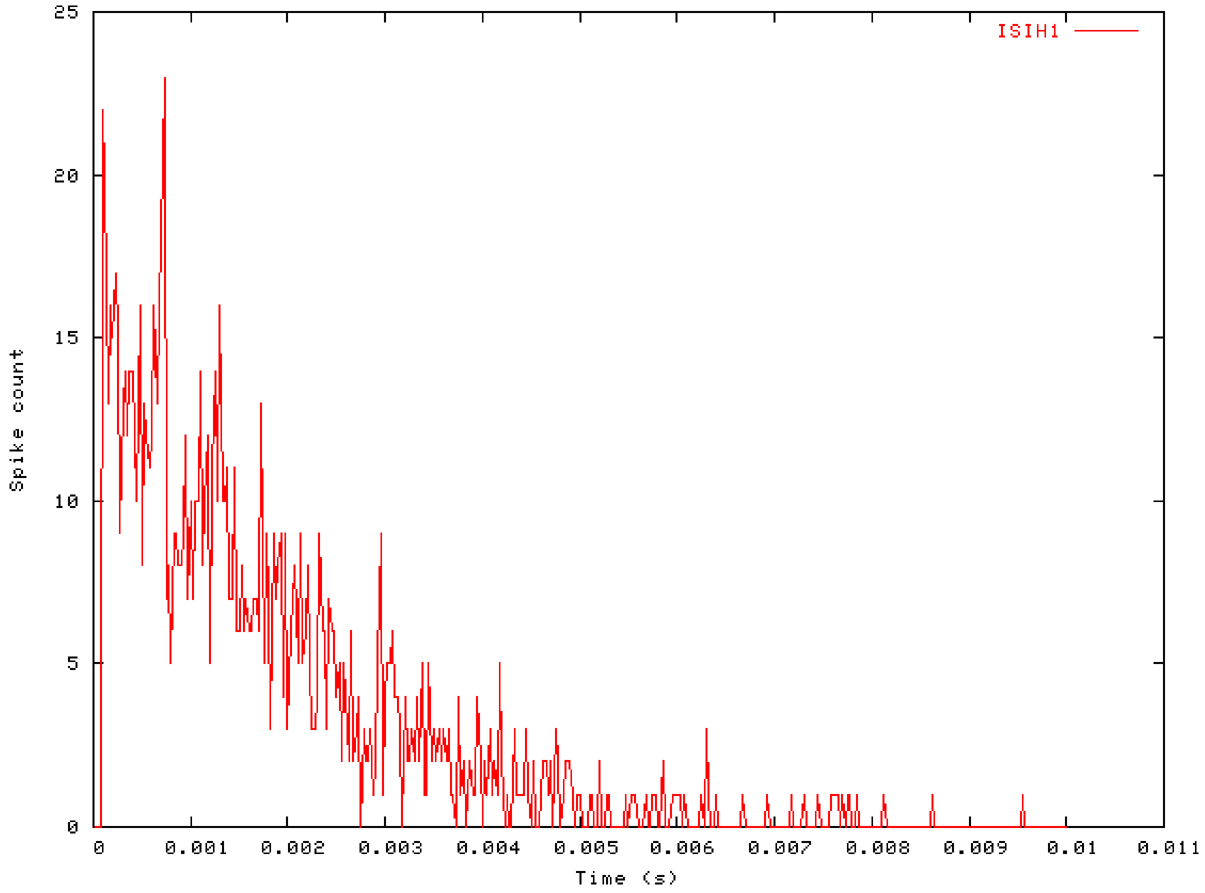
ORDER	-1	Order of spike interactions (1 = 1st, 2 = 2nd, -1 = all order).
THRESHOLD	0	Event threshold (arbitrary units).
MAX_INTERVAL	-1	Max. interval for histogram: -ve assumes end of signal (s).

Examples Using the ’Ana_ISIH’ Process Module

Example: AutoTest/A_Di/ISIH1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A_Di/ISIH1.sim
begin  {

 Stim_Harmonic    < Harmonic2.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_GammaT       < GammaT500Hz.par
 IHC_Meddis86     < Meddis86.par
 repeat 2 {
 AN_SG_Simple    < SimpleSG1.par
 Ana_ISIH       < InterSIH1.par
 }
 Display_Signal

}
```

Figure 3.8. ISIH process outputThe results shown are for two auditory nerve fibres.



Ana_SAI: Stabilised Auditory Image function

File name: AnSAImage.[ch]

Description

This module implements the auditory image model (AIM) Strobed Temporal Integration (STI) process (see [27]). STI may be visualised as a bank of delay lines used to form a buffer store for the process input signal, one delay line per channel. The input signal is usually the output from the auditory nerve process, or neural activity pattern (NAP). As the process input proceeds along the buffer, it decays linearly with time, at about 2.5 %/ms, so there is no activity beyond about 40 ms in the NAP buffer. Each channel of the buffer is assigned a strobe unit that monitors activity in that channel, looking for local maxima in the stream of NAP pulses. When one is found, the unit initiates temporal integration in that channel: it transfers a copy of the entire NAP function in that channel at that instant, to the corresponding channel of an image buffer. The NAP is added point-for-point with whatever is already in that channel of the image buffer. The local maximum itself is mapped to the 0 ms point in the image buffer.

The multi-channel version of this STI process is AIM’s representation of our auditory image of a sound. Periodic and quasi-periodic sounds typically produce a single local maximum per cycle, per channel of the NAP. In any given channel, this leads to regular strobing and the transfer into the auditory image of a sequence of NAP functions that are all virtually identical. As a result the auditory images of periodic sounds lead to static auditory images, and quasi-periodic sounds lead to nearly static images. These images, however, have the same temporal resolution as the NAP. Dynamic sounds are represented as a sequence of auditory image frames. If the rate of change in a sound is not too rapid, as is diphthongs, features are seen to move smoothly as the sound proceeds, much as objects move smoothly in animated cartoons.

In this implementation of STI the different auditory image frames can only be produced if this module is used in segment processing mode (see the section called “Using the Segment-processing Mode”). In the standard processing mode, only the final image will be output.

This module uses the Util_Strobe module (see the section called “Util_Strobe: Signal Strobe Utility Module”), where a detailed description of the strobe modes is presented.

Inputs	Arbitrary single input, though it is used in AIM with the neural activity pattern (NAP).
Outputs	A ’static’ signal with the same number of channels as the input signal. The channel length is defined by the **nwidth** and **pwidth** parameters.
Reference	Patterson R. D. and Allerhand M. H. (1995): ’Time-domain modeling of peripheral auditory processing: A modular architecture and a software platform’, J. Acoust. Soc. Am. [98], 1890-1894.

Module Parameters

Example 3.12. Ana_SAI parameter file

DIAGNOSTICS	OFF	Diagnostic mode (’off’, ’screen’, ’error’ or <file name>).
INT_MODE	STI	Integration mode: ’STI’ - stabilised temporal integration, ’AC’ - autocorrelation.
CRITERION	PEAK_SHADOW+	Strobe criterion (mode): ’user ’, ’threshold’ (0), ’peak’ (1), ’peak_shadow-’ (3), or ’peak_shadow+’ (4/5).
STROBE_DIAGNOSTICS	OFF	Diagnostic mode (’off’, ’screen’, ’error’ or <file name>).
THRESHOLD	0	Threshold for strobing: ’user’, ’threshold’ and ’peak’ modes only.
THRESHOLD_DECAY	5000	Threshold decay rate (%/s).
STROBE_LAG	0.005	Strobe lag (delay) time (s).
TIMEOUT	0.01	Strobe lag (delay) timeout (s).
NWIDTH	-0.035	Negative width of auditory image (s).
PWIDTH	0.005	Positive width of auditory image (s).
NAP_DECAY	2500	Neural activity pattern (input) decay rate (%/s)
IMAGE_DECAY	0.03	Auditory image decay half-life (s).

Table 3.10. Ana_SAI Diagnostics Mode Options

Diagnostic Mode	Description
off	No diagnostics are output.;
screen	Diagnostics from the module are output to the screen.
error	Diagnostics are output to the standard error output stream (stderr on Unix)
<file name>	Diagnostics are output to the specified file.

Table 3.11. Ana_SAI Integration Mode Options

Integration Mode	Description
sti	This option provides the standard stabilised temporal integration behaviour
ac	This option makes this module implement an autocorrelation integration method.

Examples Using the ’Ana_ACF’ Process Module

Note that for the following tests the *segmented processing* mode must be ’off’.

Example: AutoTest/A/SAI1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A/SAI1.sim
begin  {

stim% Util_SimScript  (->image) < SAIStim.sim
trig% Stim_PulseTrain (->image) < PulseTr3.par
image% Ana_SAI       (stim, trig->) < SAImage1.par
 Display_Signal

}
```

Figure 3.9. Stimulus input for the stabilised auditory image (SAI) analysis module tests.

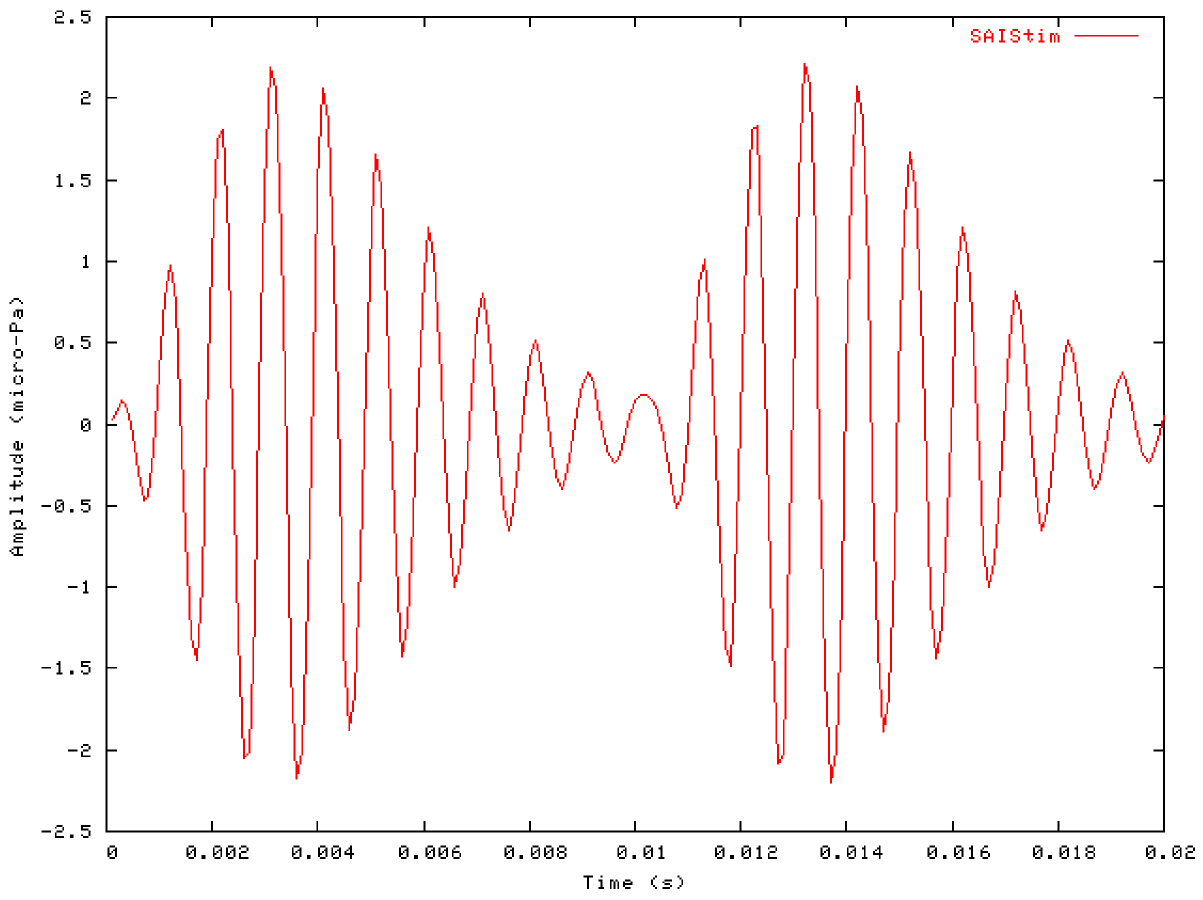
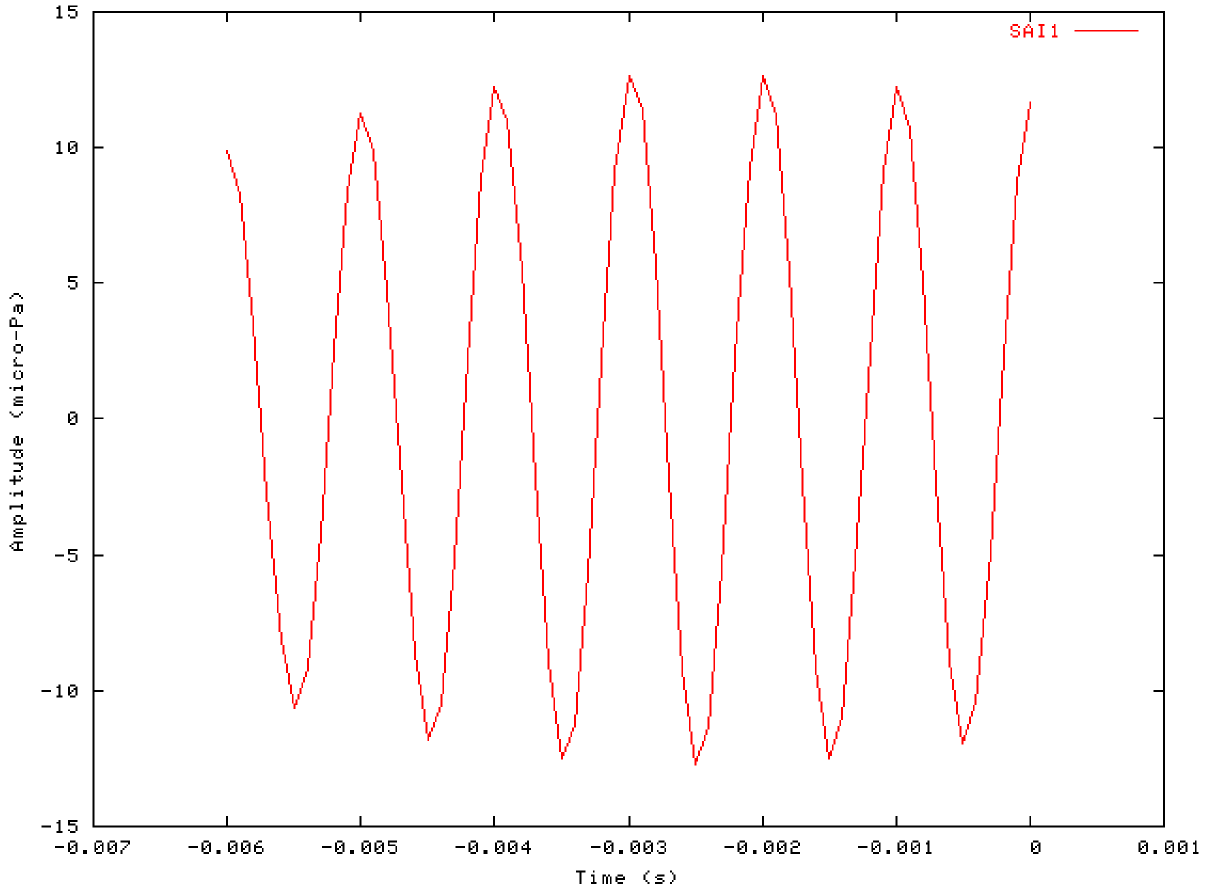


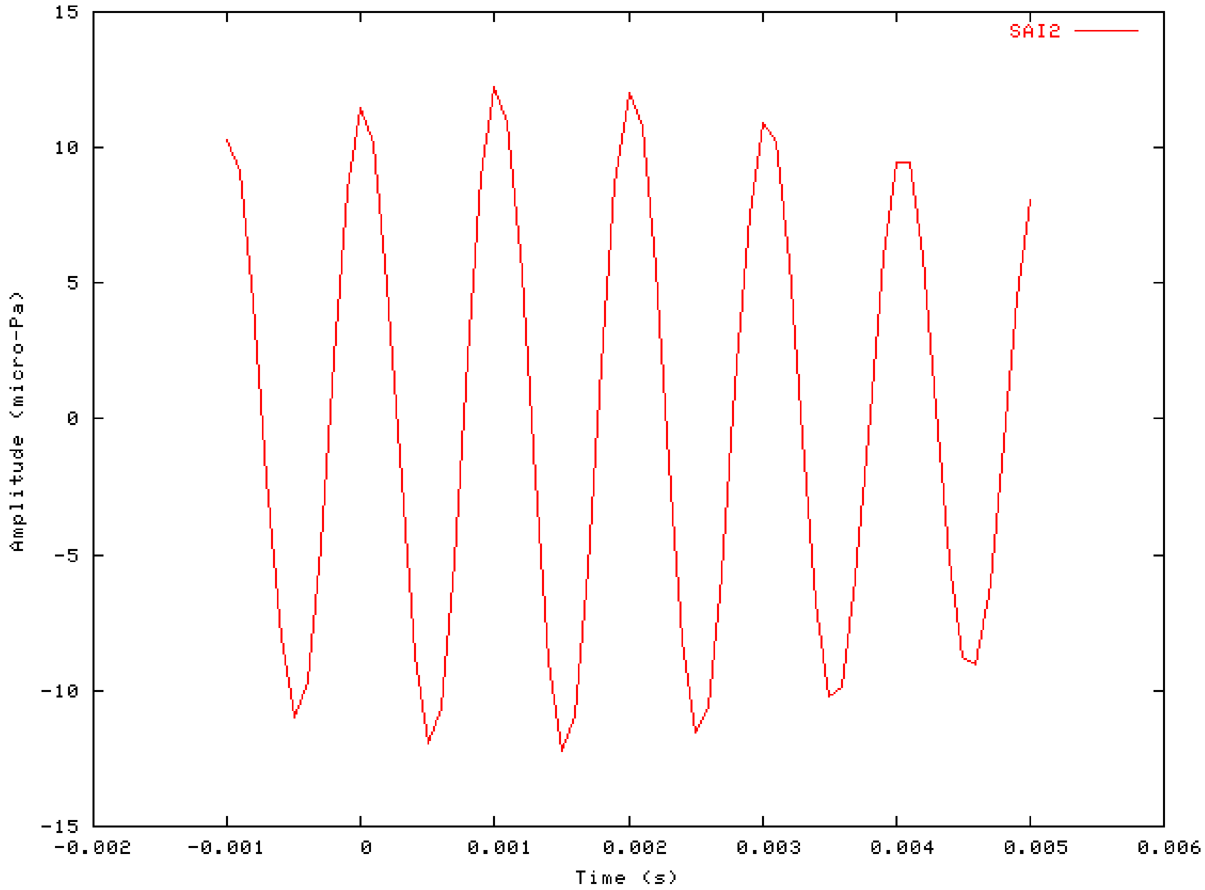
Figure 3.10. SAI module output using 'user' strobe modeThe strobe input is a train of pulses. The image frame is negative.



Example: AutoTest/A/SAI2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A/SAI2.sim
begin  {

stim% Util_SimScript  (->image) < SAIStim.sim
trig% Stim_PulseTrain (->image) < PulseTr3.par
image% Ana_SAI       (stim, trig->) < SAImage2.par
 Display_Signal

}
```

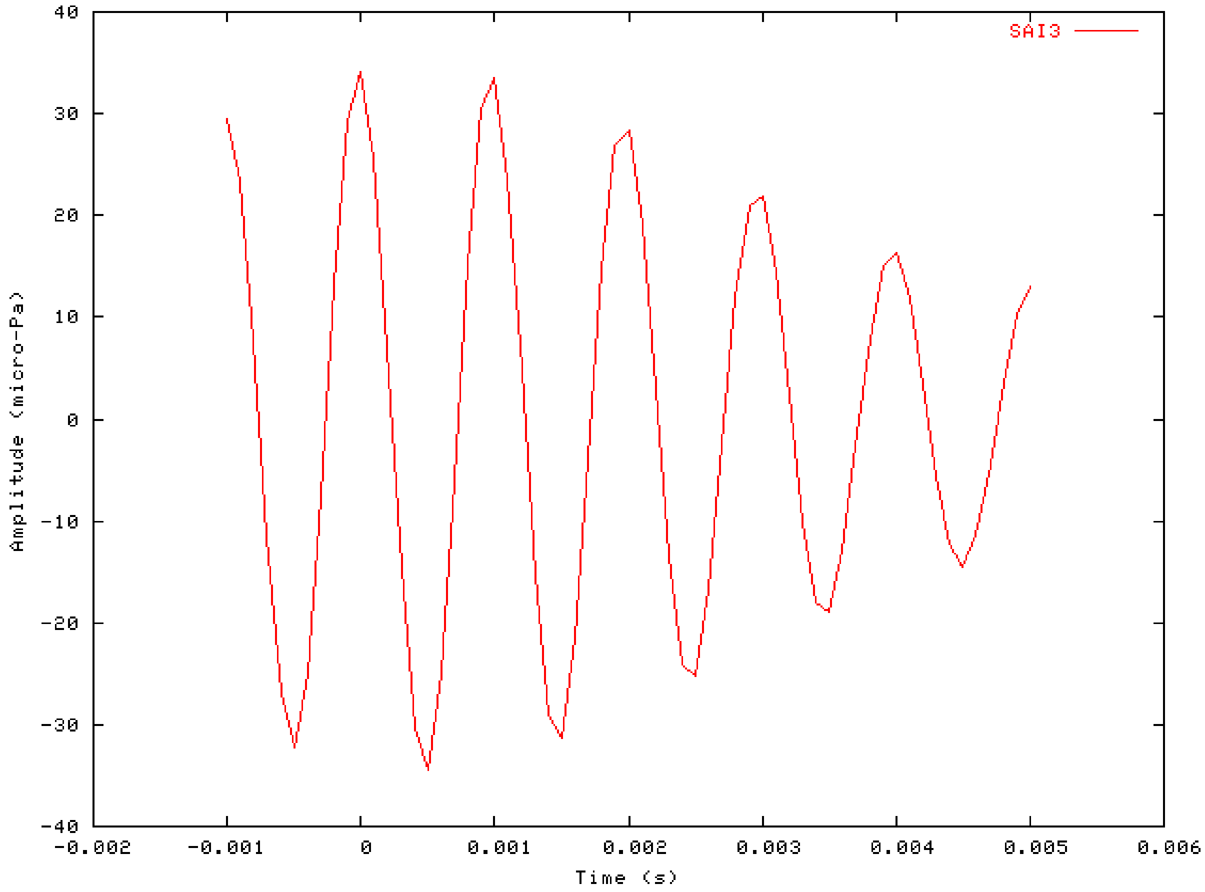
Figure 3.11. SAI module output 2 using ’user’ strobe modeThe strobe input is a train of pulses. The image frame is from -1.0 to 5.0 ms.



Example: AutoTest/A/SAI3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A/SAI3.sim
begin  {

 Util_SimScript   < SAIStim.sim
 Ana_SAI        < SAImage3.par
 Display_Signal

}
```

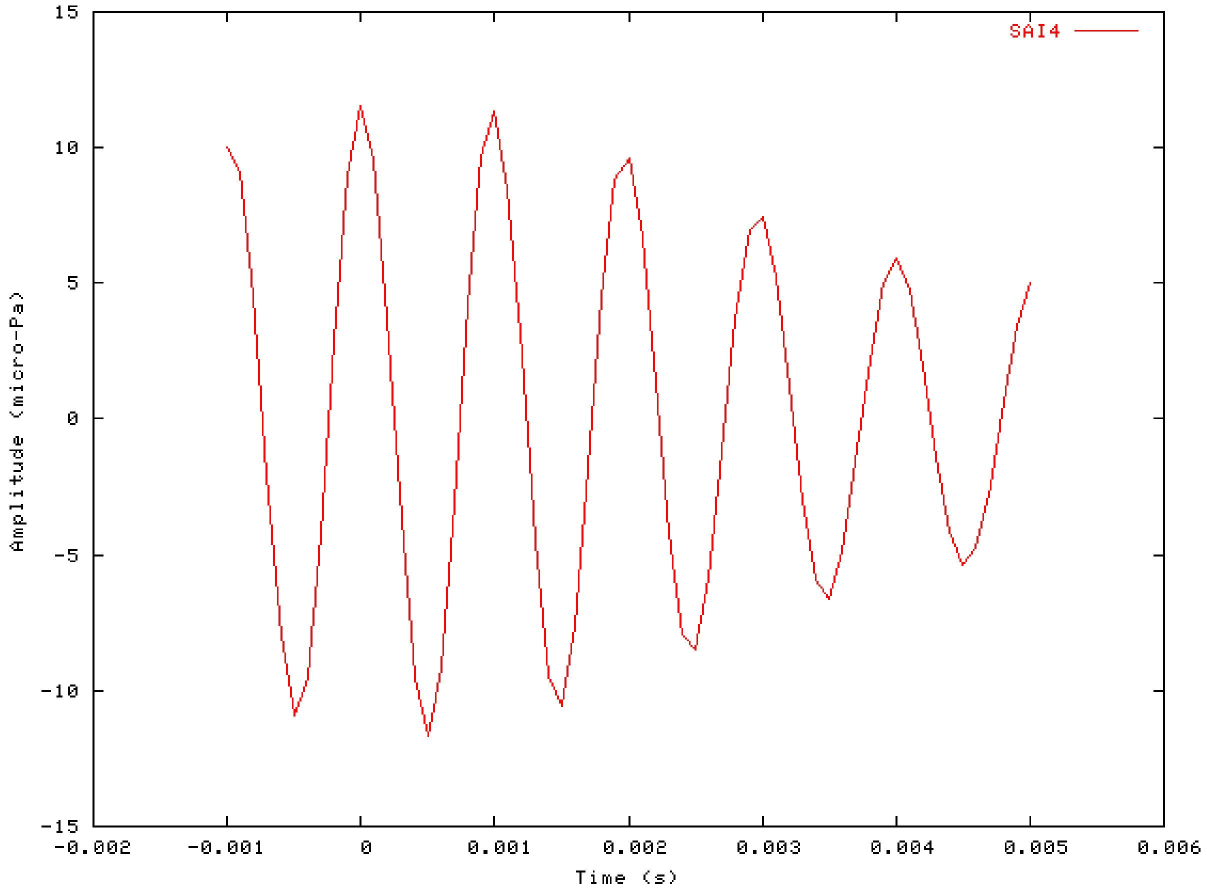
Figure 3.12. SAI module output using ’threshold’ strobe modeIn this mode strobes are triggered when the process input rises over a specified threshold.



Example: AutoTest/A/SAI4.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A/SAI4.sim
begin  {

 Util_SimScript   < SAIStim.sim
 Ana_SAI        < SAImage4.par
 Display_Signal

}
```

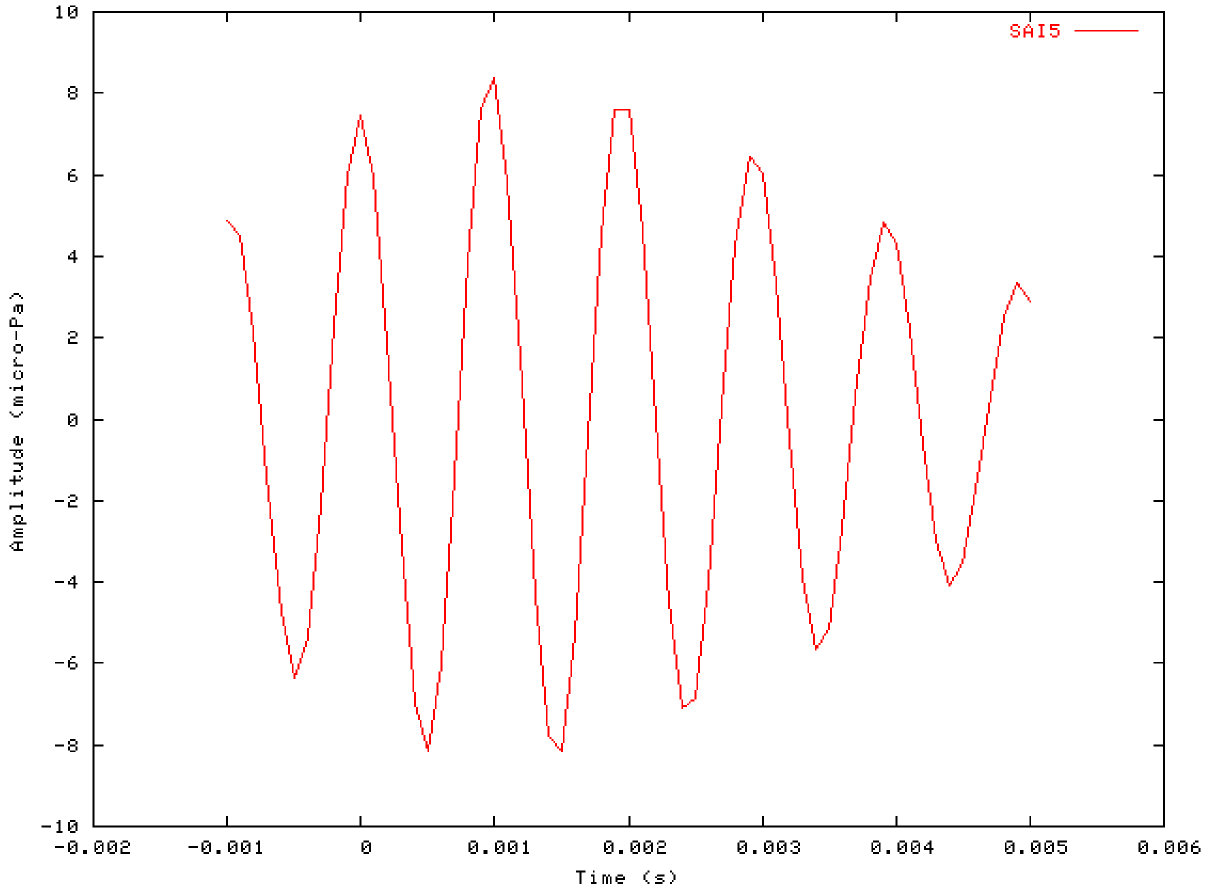
Figure 3.13. SAI module output using ’peak’ strobe modeIn this mode strobes are triggered when a peak over a specified threshold occurs in the process input.



Example: AutoTest/A/SAI5.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A/SAI5.sim
begin  {

stim% Util_SimScript  (->image) < SAIStim.sim
trig% Stim_PulseTrain (->image) < PulseTr3.par
 repeat 10 {
 image% Ana_SAI       (stim, trig->) < SAImage5.par
 }
 Display_Signal

}
```

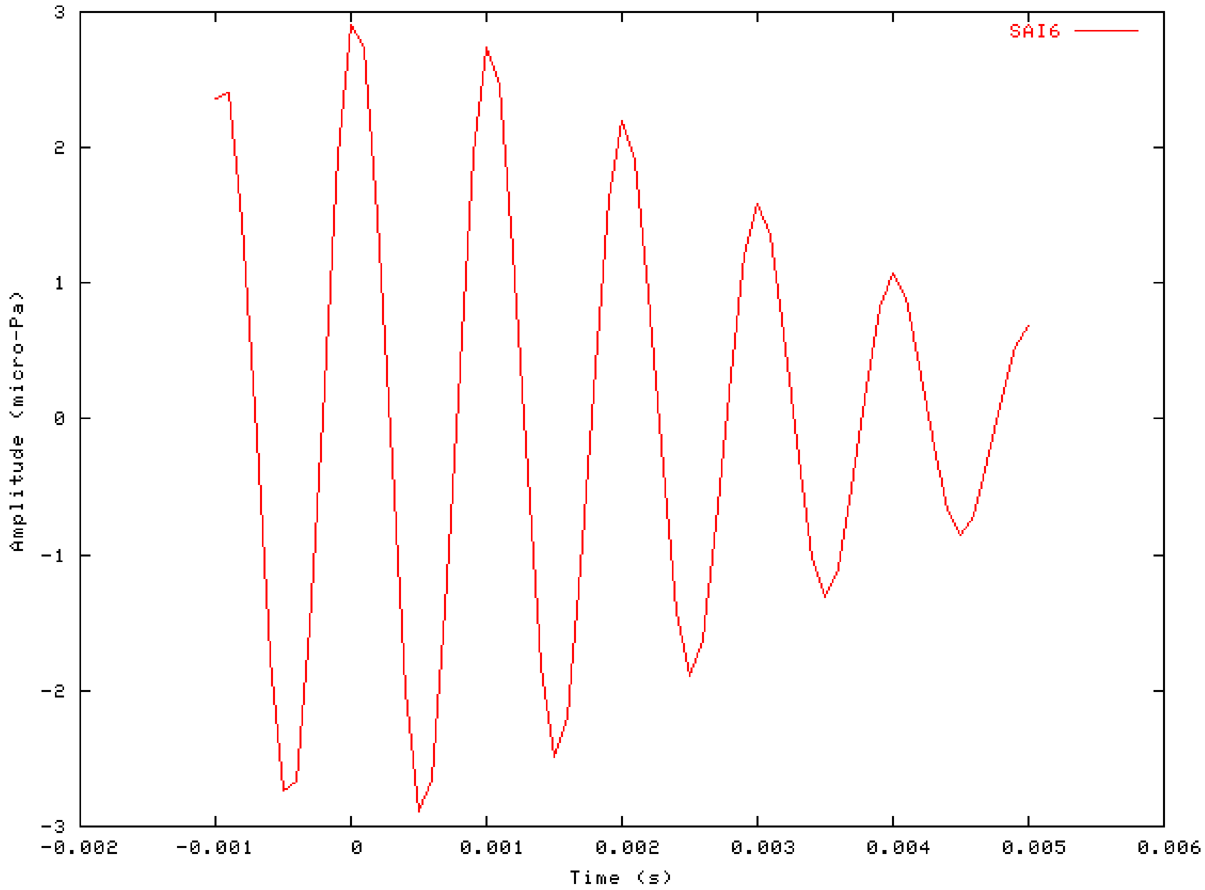
Figure 3.14. SAI module output using the ’peak_Shadow-’ strobe modeIn this mode strobes are triggered on peaks which are not followed by a larger peak.



Example: AutoTest/A/SAI6.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A/SAI6.sim
begin  {

 Util_SimScript   < SAIStim.sim
 Ana_SAI         < SAImage6.par
 Display_Signal

}
```

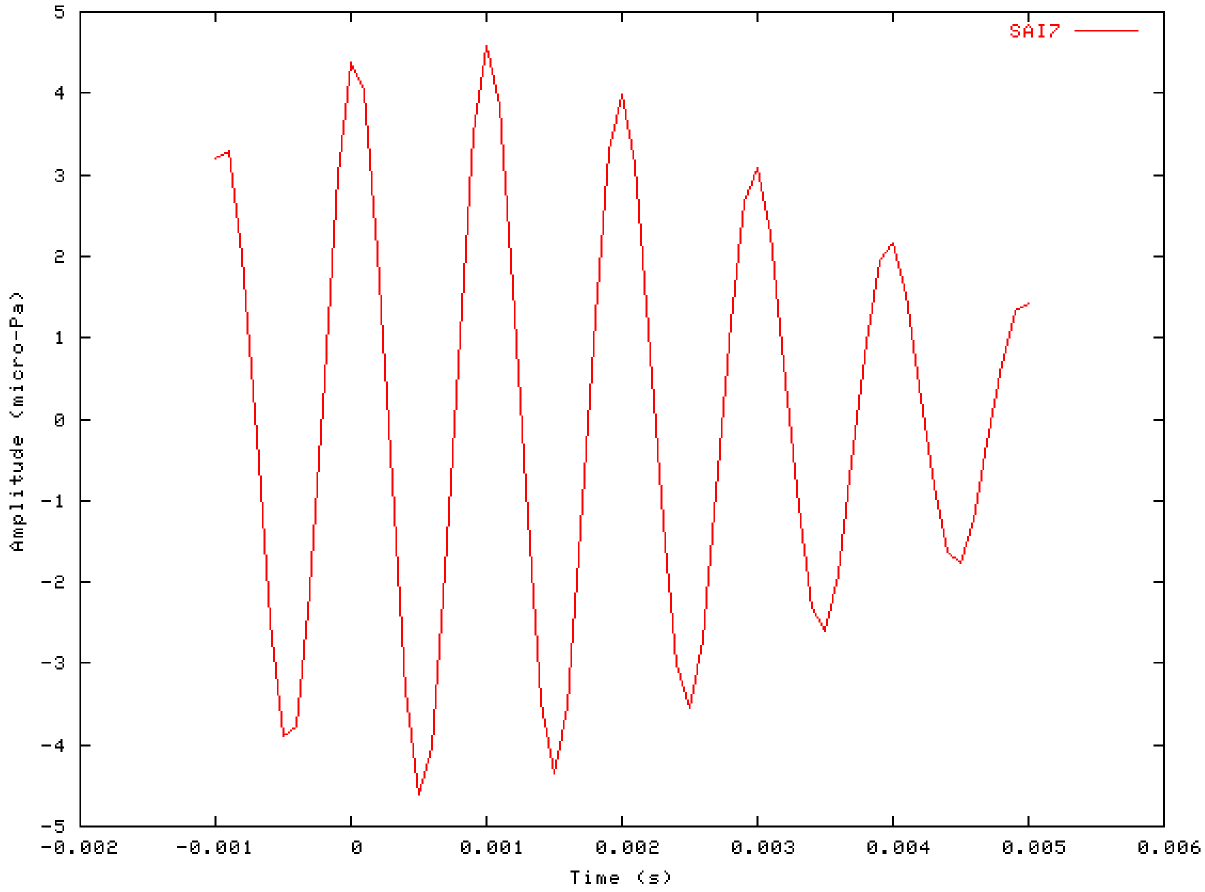
Figure 3.15. SAI module output using the 'peak_shadow+' strobe modeIn this mode peaks that are not followed by a larger peak trigger strobe events.



Example: AutoTest/A/SAI7.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A/SAI7.sim
begin  {

stim% Util_SimScript  (->image) < SAIStim.sim
trig% Stim_PulseTrain (->image) < PulseTr3.par
 repeat 10 {
 image% Ana_SAI       (stim, trig->) < SAImage7.par
 }
 Display_Signal

}
```

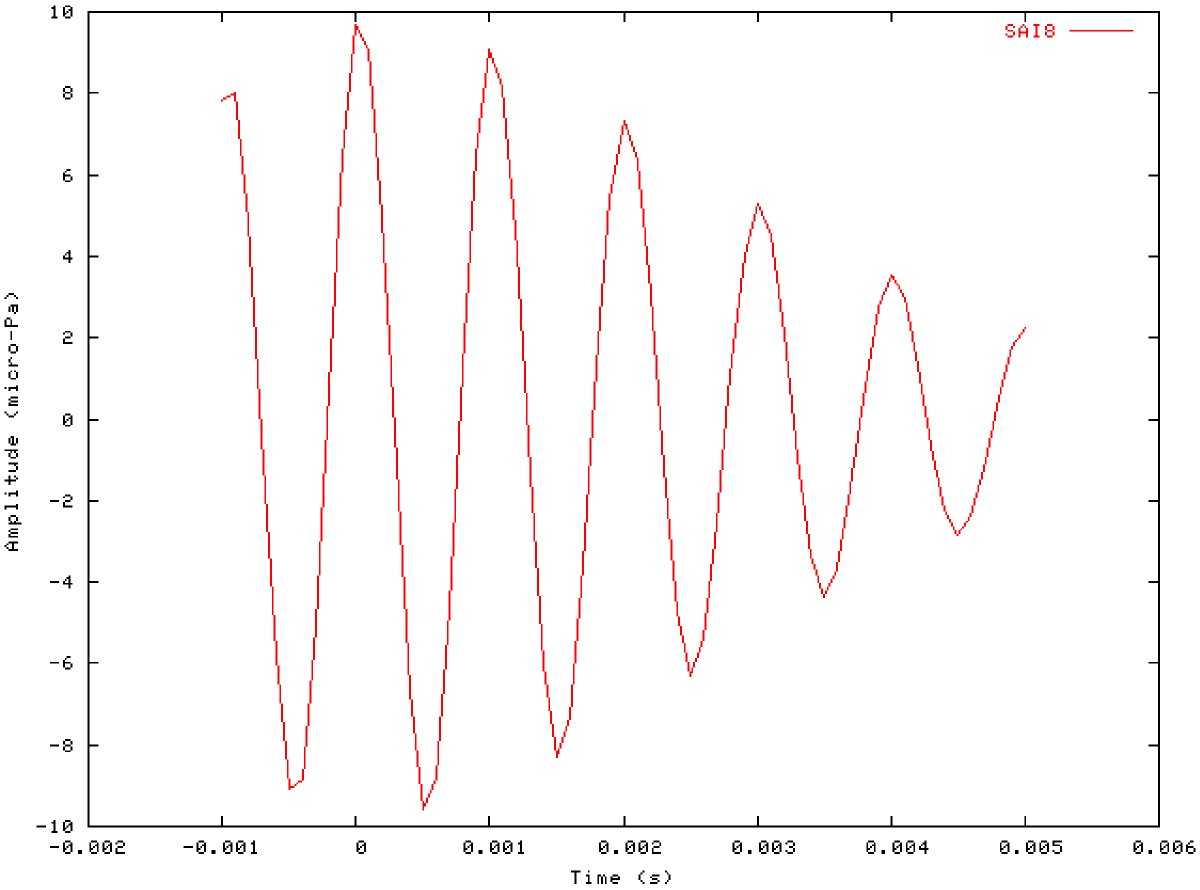
Figure 3.16. SAI module output using the ’peak_shadow+’ strobe mode with the timeout parameter set.In this mode peaks that are not followed by a larger peak trigger strobe events, but when the timeout period expires, a strobe event is triggered anyway.



Example: AutoTest/A/SAI8.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A/SAI8.sim
begin  {

 Util_SimScript   < SAIStim.sim
 Ana_SAI        < SAImage8.par
 Display_Signal

}
```

Figure 3.17. SAI module output using the autocorrelation, ’ac’ integration mode with the ’threshold’ strobe mode.With the autocorrelation integration mode, at each strobe event the input signal added to the stabilised auditory image is multiplied by the input signal value at the strobe instant.



Comments

For these test *simulation scripts* the Ana_SAI process module is tested using the impulse response of a gamma-tone filter. The ’SAIStim.sim’ *Simulation script* is used to create the stimulus. The stimulus is shown in Figure 3.9, “Stimulus input for the stabilised auditory image (SAI) analysis module tests.” This stimulus particular stimulus was chosen because it contains peaks surrounded on both sides by successively lower peaks.

Figure 3.10, “SAI module output using ’user’ strobe modeThe strobe input is a train of pulses. The image frame is negative.” and Figure 3.11, “SAI module output 2 using ’user’ strobe modeThe strobe input is a train of pulses. The image frame is from -1.0 to 5.0 ms.” show the response of the Ana_SAI process module in ’user’ strobe mode. In this mode the Ana_SAI module strobe trigger is defined by the ’Stim_PulseTrain’ stimulus module, set using the ’PulseTr1.par’ parameter file. This stimulus generation process module produces a train of pulses.

Ana_SpikeRegularity: Spike Regularity Analysis Function

File name: AnSpikeReg.[ch]

Description

This module carries out a spike regularity analysis, calculating the mean, standard deviation and covariance measures of successive intervals for a spike train in the process input signal. The results for each channel are stored in the first three samples of the output signal in the order - mean, standard deviation, covariance (coefficient of variance corrected for refractoriness, CV’) (see [30]).

CV’ is a measure of regularity, defined as the ratio of the standard deviation to the mean of the interspike intervals (ISI), i.e.

$$CV' = \frac{StdDev(ISI)}{\overline{ISI} - t_0}$$

where t0 is the **DEAD_TIME.**

Note: To calculate a single CV value for the entire signal, **WINDOW_WIDTH** should be set to -1. Values of **WINDOW_WIDTH** other than -1 generate a CV output every **WINDOW_WIDTH** seconds.

Inputs	Arbitrary single input
Outputs	The following output is output for each channel: *mean, standard deviation, corrected covariance*. Values of the refractoriness corrected covariance, CV’ are only calculated if there are two or more spike events. The output length is defined by the **WINDOW_WIDTH, OFFSET** and **RANGE** parameters.
Reference	Young, E. D., Robert, J. and Shofner, W. P. (1988). ’Regularity and latency of units in ventral cochlear nucleus: Implications for unit classification and generation of response properties’. J. Neurophysiology, 60(1), 1 – 29.Rothman J. S., Young E. D., and Manis P. B. (1993). "Convergence of Auditory Nerve Fibers in the Ventral Cochlear Nucleus: Implications of a Computational Model" J. Neurophysiology, 70, 2562-2583.

Module Parameters

Example 3.13. Ana_SpikeRegularity parameter file

THRESHOLD	0	Event threshold (arbitrary units).
WINDOW_WIDTH	-1	Width of analysis window: -ve assumes total analysis range (s).
OFFSET	0	Time offset for start of analysis (s).
RANGE	-10	Time range for analysis: -ve assumes to end of signal (s).
DEAD_TIME	0	Spike dead time or absolute refractory period (s).

Examples Using the ’Ana_SpikeRegularity’ Process Module

Example: AutoTest/A/SpikeRegularity1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A/SpikeRegularity1.sim
begin  {

 Stim_PulseTrain  < PulseTr1.par
 repeat 100 {
 Ana_SpikeRegularity < SpikeReg1.par
 }
 Display_Signal

}
```

Example: AutoTest/A/SpikeRegularity2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A/SpikeRegularity2.sim
begin  {

 Stim_PulseTrain  < PulseTr1.par
 repeat 100 {
 reset sr
 sr% Ana_SpikeRegularity < SpikeReg1.par
 }
 Display_Signal

}
```

Comments

These simulation scripts produce no displays. If the ’DataFile_Out’ module in the respective script is uncommented, then the results for the test are output to a file.

Ana_SynchIndex: Synchronisation Index

File name: AnSynchIndex.[ch]

Description

This routine calculates the synchronisation index (vector strength) from the process’ input. **It expects its input to come from a period histogram** (see the section called “Ana_Histogram: Histogram analysis function”). The results are put into the first and only sample of the output signal for each respective channel.

The synchronisation index measures the ability of a system to follow the periodicity of a stimulus. (see [10]).

The synchronisation index is calculated from:

$$SI = \frac{\sqrt{\left[\sum_{i=1}^{N} x_i \sin\left(2\frac{i}{N}\right)\right]^2 + \left[\sum_{i=1}^{N} x_i \sin\left(2\frac{i}{N}\right)\right]^2}}{\sum_{i=1}^{N} |x_i|}$$

N is the number of points in the PH (period/bin width).

Note: The alternative is to use an FFT of a histogram to obtain synchronisation to all periods.

Inputs	This process requires input from a period histogram (see the section called “Ana_Histogram: Histogram analysis function”
Outputs	The results are set in the first sample of the output signal for each respective channel.
Reference	
See also	See the section called “Ana_Histogram: Histogram analysis function”

Module Parameters

This module has no parameters.

Examples Using the ’Ana_SynchIndex’ Process Module

Example: AutoTest/A/SynchIndex.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./A/SynchIndex.sim
begin  {

 Stim_PureTone_AM < AMPTone1.par
 Ana_Histogram   < HistPH1.par
 Ana_SynchIndex
 Display_Signal

}
```

Comments

This simulation script produces no display. If the ’DataFile_Out’ module line is uncommented, then the result will be output to the file ’output.dat’. The output should look as shown below:-

Time (s)	100	146	213	311	453	662	965	1409	2056	3000
5e-05	0.0320	0.0294	0.0057	0.0186	0.0562	0.0428	0.2903	0.0400	0.0394	0.0352

Chapter 4. Auditory Nerve Model Modules

AN_SG_Binomial: Auditory Nerve Binomial Approximation

File name: MoANSGBinom.[ch]

Description

This module expects to receive spike probabilities as an input signal, which is then converted to spike events using a random number generator. It uses the geometric distribution method to produce the results for s specified number of fibres (without the need for a loop). A refractory effect is also applied (see module Util_RefractoryAdjust).

This approximation runs much faster then the other less efficient auditory nerve modules. There is no speed gain when there is only one fibre but efficiency increases substantially as the number of fibres increases. The approximation is only valid when the probabilities are small (say less than 0.2). If in doubt, compare the output against the output using the AN_SG_Simple module.

Inputs	Arbitrary signal input. Expects spike probability input.
Outputs	
Reference	

Module Parameters

Example 4.1. AN_SG_Binomial Parameter File

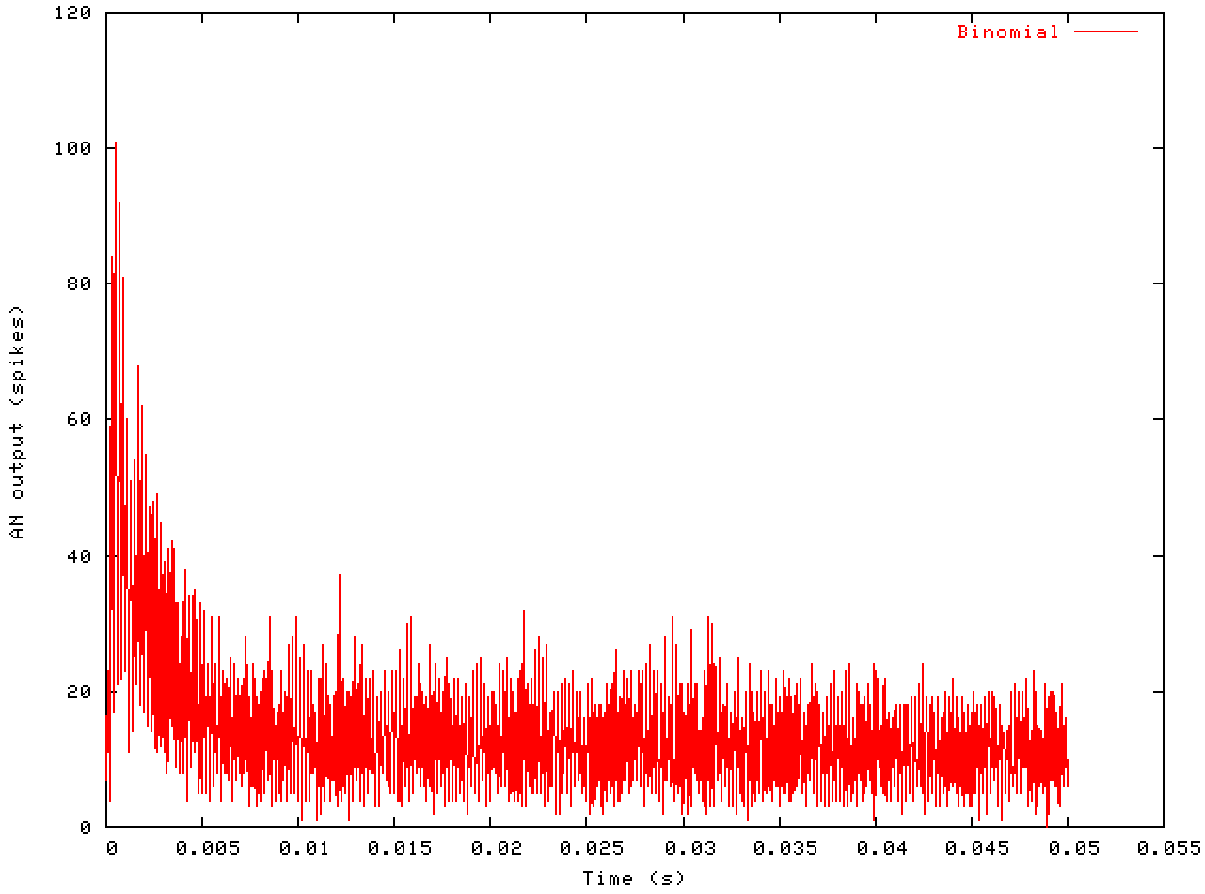
NUM_FIBRES	5000	Number of fibres.
RAN_SEED	-1	Random number seed (0 for different seed for each run).
PULSE_DURATION	2e-05	Pulse duration (s).
MAGNITUDE	1	Pulse magnitude (arbitrary units).
REFRAC_PERIOD	0.001	Refractory period (s).

Examples Using the ’AN_SG_Binomial’ Process Module

Example: AutoTest/AN/Binomial1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./AN/Binomial.sim
begin  {

 Stim_PureTone    < PTone1.par
 Trans_Gate      < Ramp1.par
 IHC_Meddis86    < Meddis86.par
 AN_SG_Binomial   < BinomSG1.par
 Display_Signal
}
```

Figure 4.1. Binomial Approximation Spike Generation Output



Comments

This simulation script shows the Binomial spike generation routine used in conjunction with the Meddis ’86 hair cell. It shows the stochastic response expected of a real AN fibre.

AN_SG_Carney: Auditory Nerve Spike Generation (1993)

File name: MoANSGCarney.[ch]

Description

This module expects to receive spike probabilities as an input signal, which is then converted to spike events using a random number generator. A refractory period is introduced whose charactreristics are controlled by the module parameters. It uses Carney’s spike generation algorithm described in [5], and her code.

Inputs	Arbitrary signal input. Expects spike probability input.
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	Carney L. H. **(1993)** ’A model for the responses of low-frequency auditory-nerve fibers in cat’, JASA, [93], pp 401-417.

Module Parameters

Example 4.2. AN_SG_Carney Parameter File

INPUT_MODE	ORIGINAL	Input mode, ’corrected’ (2001), or ’original’ (1993) setting.
RAN_SEED	-1	Random number seed (0 produces a different seed each run).
NUM_FIBRES	1	Number of fibres.
PULSE_DURATION	-1	Excitary post-synaptic pulse duration (s).
MAGNITUDE	1	Pulse magnitude (arbitrary units).
REFRAC_PERIOD	0.00075	Refractory period (s).
THRESHOLD_INC	1	Maximum threshold increase following discharge, Hmax (spikes/s).
C0	0.5	Coefficient for discharge history effect, c0 (s).
C1	0.5	Coefficient for discharge history effect, c1 (s).
S0	0.001	Time constant for discharge history effect, s0 (s).
S1	0.0125	Time constant for discharge history effect, s1 (s).

Table 4.1. AN_SG_Carney ’input_mode’ options

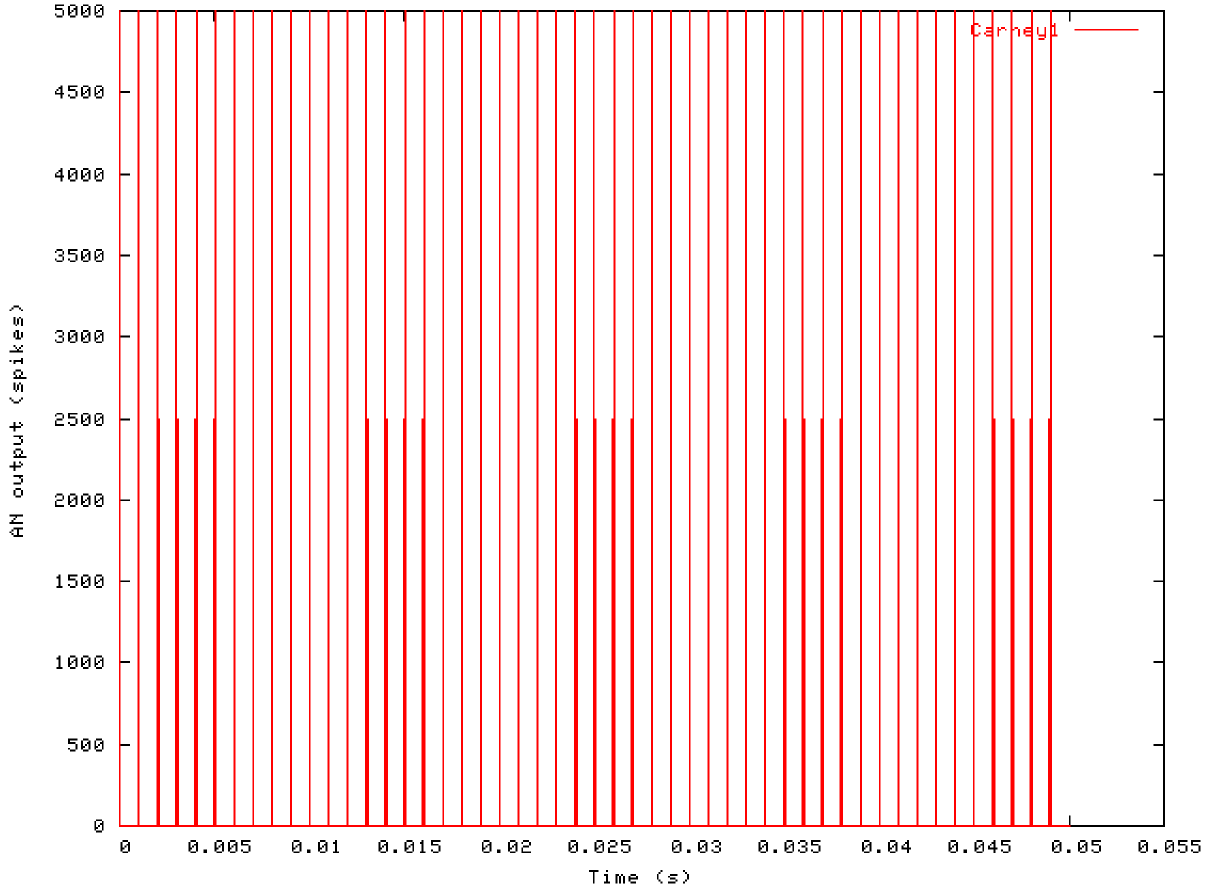
Mode	Description
original	This is the original input mode used in the 1993 paper. It is also used for the AN model of Sumner et al. described in [32]. In this mode the input to the process is divided by the sampling interval.
corrected	This mode employs the corrected input mode as used in the new AN model by Zhang et al. described in ,

Examples Using the ’AN_SG_Carney’ Process Module

Example: AutoTest/AN_Di/Carney1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./AN_Di/Carney1.sim
begin {

 Stim_PureTone    < PTone1.par
 Trans_Gate      < Ramp1.par
 IHC_Meddis86     < Meddis86.par
 Util_MathOp     < UpScale1.par
 AN_SG_Carney     < CarneySG1.par
 Display_Signal

}
```

Figure 4.2. Carney AN Spike Generation Output



Comments

In this simulation test the scaled output from the IHC_Meddis86 process has been used to allow a comparison between the output of this process module and that of other spike generation process modules.

AN_SG_Simple: Auditory Nerve Simple Spike Generation

File name: MoANSGSimple.[ch]

Description

This module expects to receive spike probabilities as an input signal, which are then converted to spike events using a random number generator. It uses a simple spike generation.

Inputs	Arbitrary signal input. Expects spike probability input.
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	

Module Parameters

Example 4.3. AN_SG_Simple Parameter File

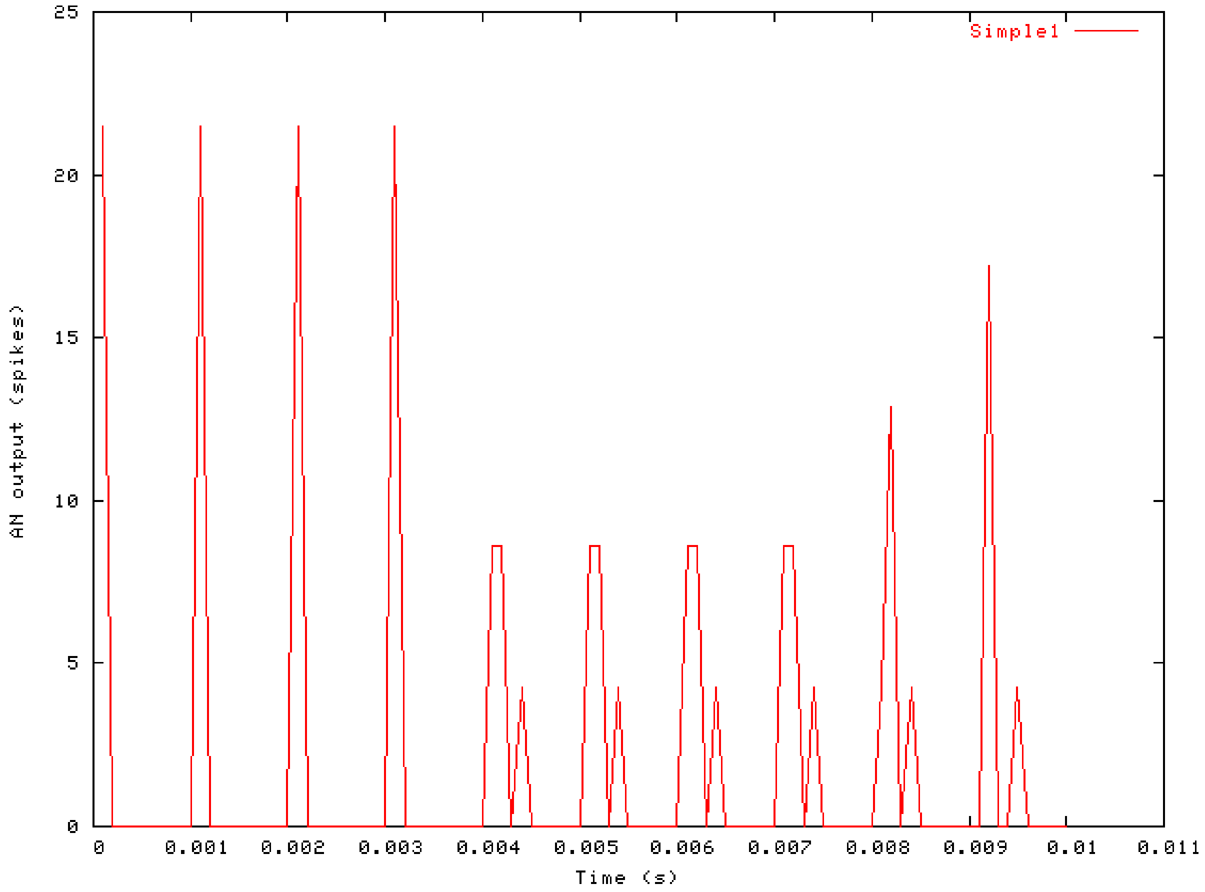
RAN_SEED	-1	Random number seed (0 produces a different seed each run.
NUM_FIBRES	5	Number of fibres.
PULSE_DURATION	0.0001	Pulse duration (s).
MAGNITUDE	4.3	Pulse magnitude (arbitrary units).
REFRAC_PERIOD	0.001	Refractory period (s).

Examples Using the ’AN_SG_Simple’ Process Module

Example: AutoTest/AN/Simple1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./AN/Simple1.sim
begin  {

 Stim_StepFun    < StepFun1.par
 AN_SG_Simple    < SpikeGen1.par
 Display_Signal

}
```

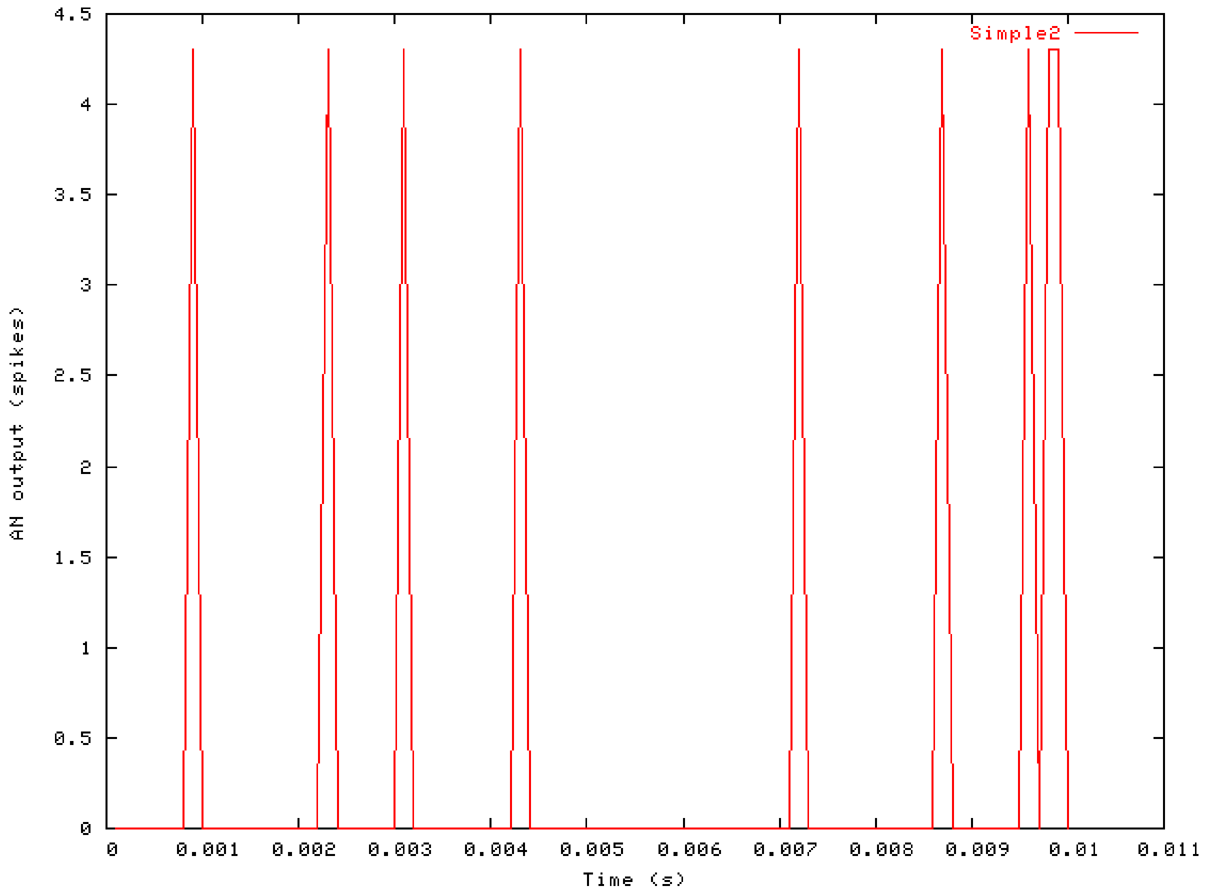
Figure 4.3. Simple AN Spike Generation Output (1)



Example: AutoTest/AN/Simple2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./AN/Simple2.sim
begin  {

 Stim_StepFun     < StepFun2.par
 AN_SG_Simple     < SpikeGen1.par
 Display_Signal

}
```

Figure 4.4. Simple AN Spike Generation Output (2)



Comments

By applying a constant high input to the model, it spikes at the refractory rate. Simulation script ’Simple1.sim’ shows the accumulation of a number of runs, so the total number of spikes should be dependent upon the number of runs specified in the main parameter file for the test program. Simulation script ’Simple2.sim’ is also the result of multiple runs, but it is reset each time so the total spikes should not be dependent upon the number of runs.

Chapter 5. Basilar Membrane Model Modules

BM_Carney: Non-linear BM (1993)

File name: MoBMCarney.[ch]

Description

This is the Carney non-linear basilar membrane filter model, revised from her original C code. The centre frequency mode is controlled using the UtCFList module as described in the section called “Using the Basilar Membrane Filter Modules”.

For further information on the use of BM filterbanks see the section called “Using the Basilar Membrane Filter Modules”.

Inputs	Arbitrary signal input.
Outputs	Produces multi-channel output signal, according to the CF list parameters.
Reference	[5]

Module Parameters

Example 5.1. BM_Carney Parameter File

CASCADE	4	Filter cascade.
FC	1100	Cut-off frequency for OHC low-pass filter, Fc (Hz).
P_DFB	5000	Operating point of OHC (feedback) non-linearity, P_Dfb (Pa).
P0	0.462	Asymmetrical bias for OHC non-linearity, P0 (rad).
V_MAX	10	Maximum depolarising hair cell voltage, Vmax (V).
#CF List Parameters:-		
DIAG_MODE	PARAMETERS	Diagnostic mode (’list’ or ’parameters’).
CF_MODE	LOG	Centre frequency mode (’single’, ’ERB’, ’ERB_n’, ’log’, ’linear’, ’focal_log’, ’user’, ’human’, ’cat’, ’chinchilla’, ’guinea-pig’ or ’macaque’).
MIN_CF	200	Minimum centre frequency (Hz).
MAX_CF	5000	Maximum centre frequency (Hz).
CHANNELS	30	No. of centre frequencies.
B_MODE	INTERNAL_DYNAMIC	Bandwidth mode (’ERB’, ’Custom_ERB’, ’Guinea_Pig’, ’user’ or ’Nonlinear’).

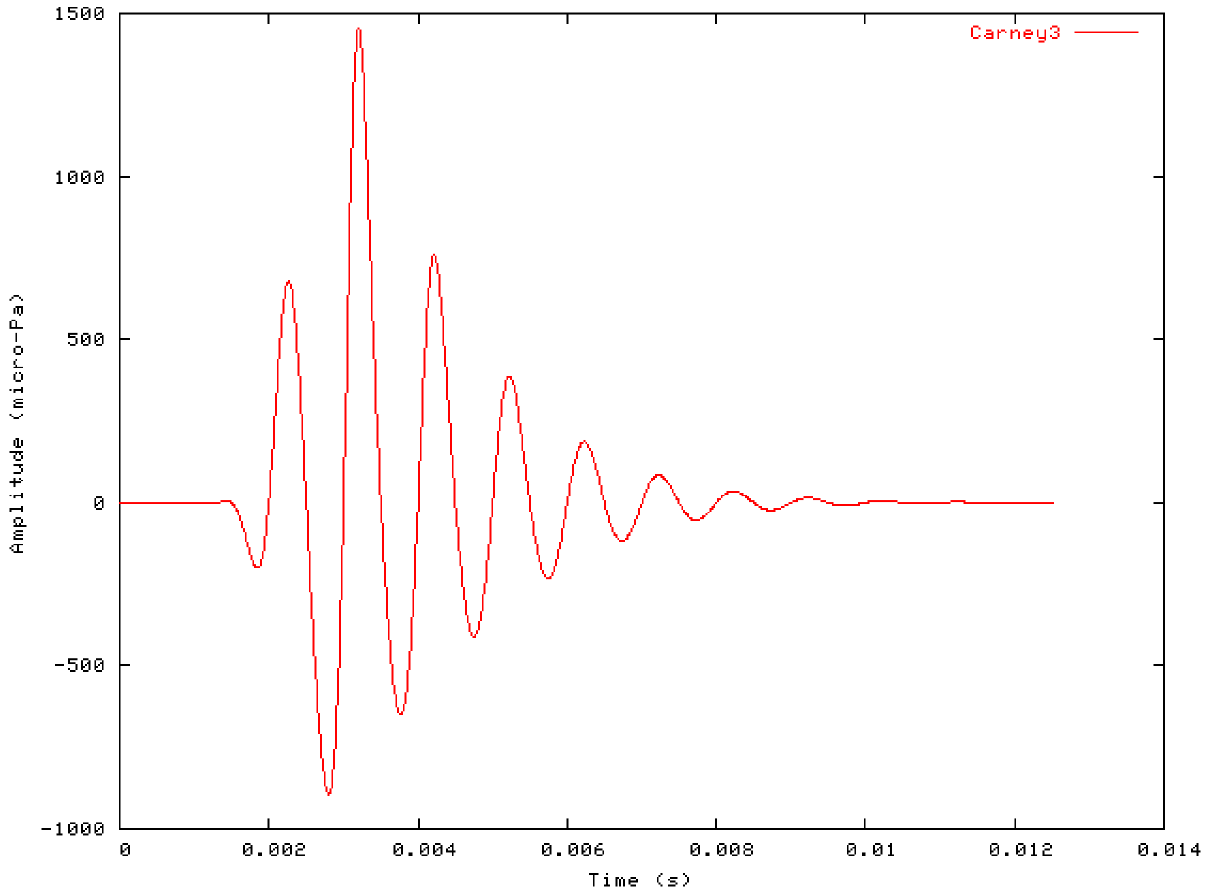
Examples Using the ’BM_Carney’ Process Module

Some of the following tests require multiple observations using a range of stimuli and cannot be produced directly with a single run of the *AMS* program. These simulation scripts are run using the *AutoTest* program, which is employed to test the operation of the DSAM process modules.

Impulse Response Example: AutoTest/BM/Carney3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Carney3.sim
begin  {

 Stim_Click      < Click1.par
 BM_Carney      < Carney1k.par

}
```

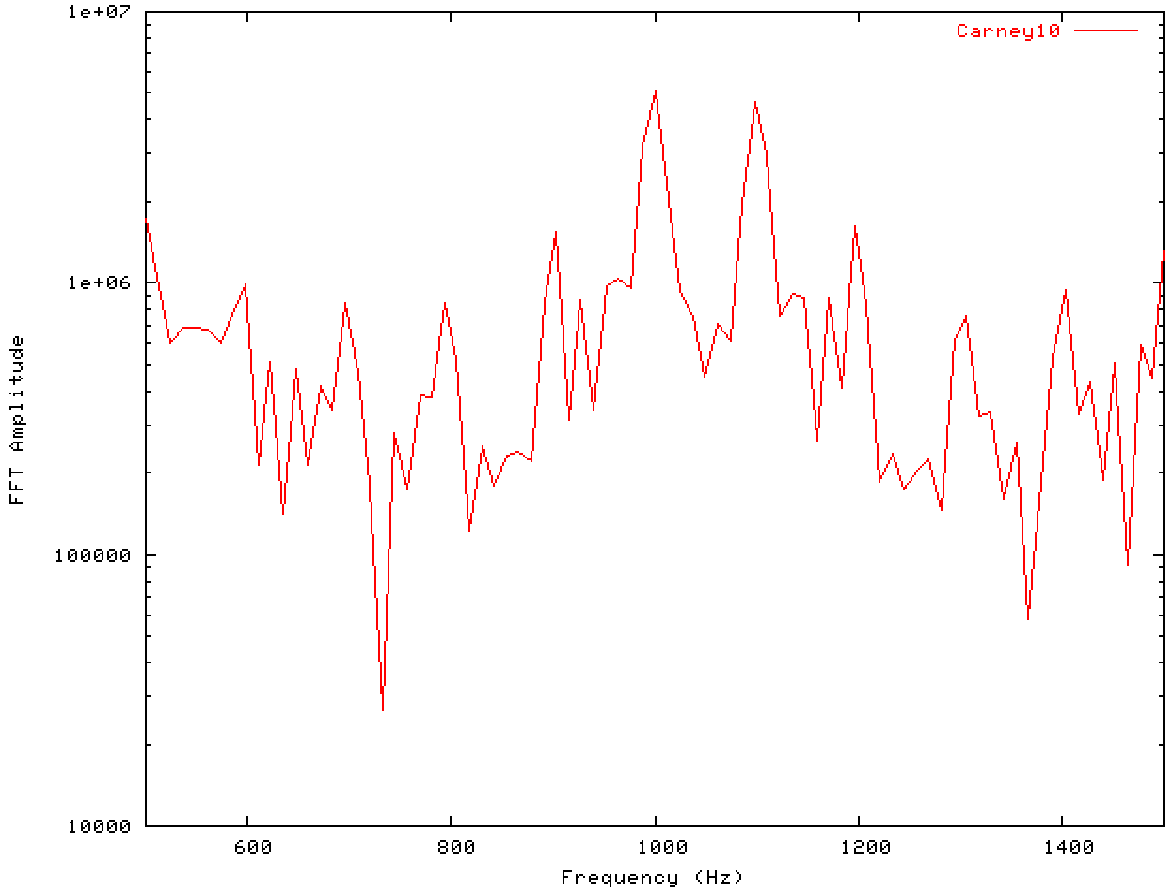
Figure 5.1. The impulse response of the Carney BM modelCF = 1 kHz.



FT Analysis Example: AutoTest/BM/Carney10.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Carney10.sim
begin  {

 Stim_Puretone_Multi < MPTone2.par
 Trans_Gate      < Ramp1.par
 BM_Carney       < Carney1k.par
 Ana_FourierT     < FourierT1.par

}
```

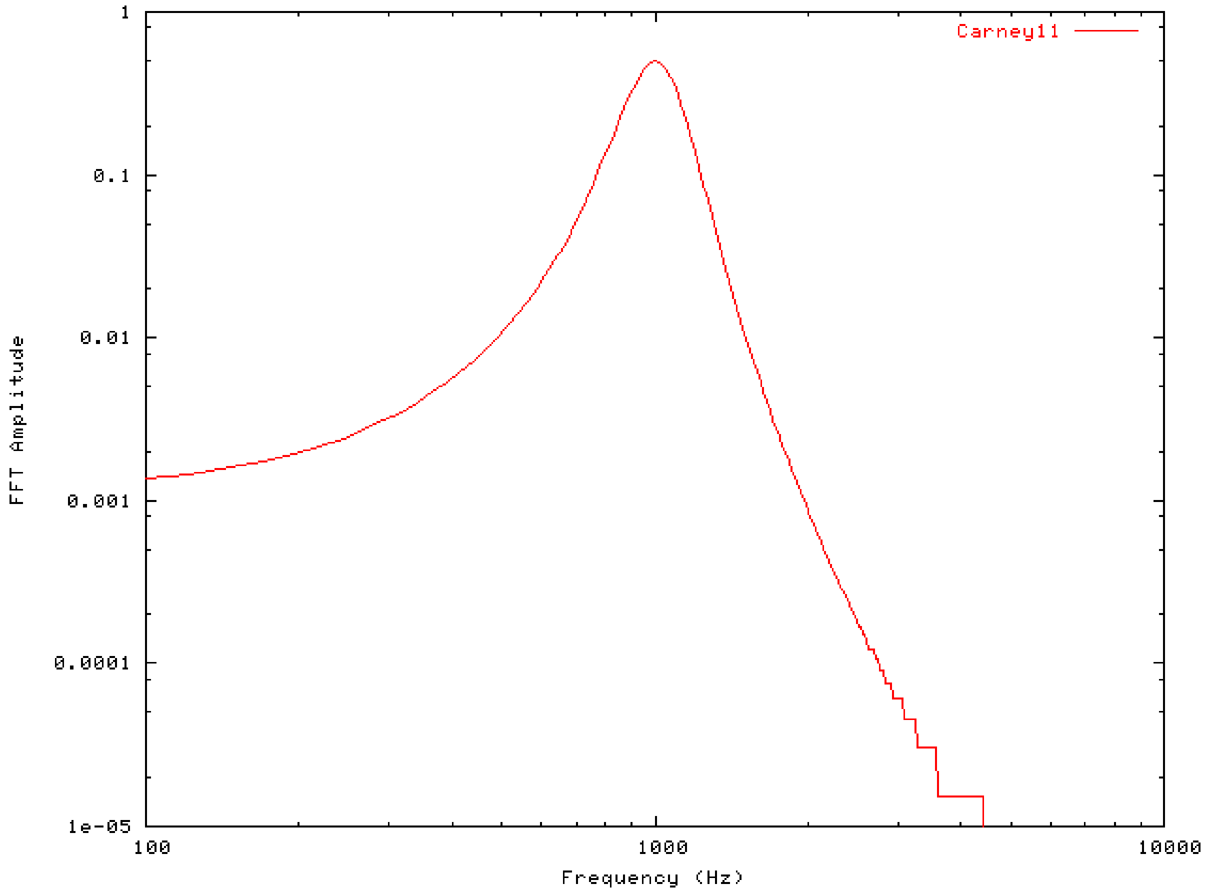
Figure 5.2. Fourier analysis of the Carney BM model response to a two-tone stimulus.CF = 1 kHz.



Gain Response Example: AutoTest/BM/Carney11.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Carney11.sim
begin  {

 Stim_Click       < Click3.par
 BM_Carney       < Carney1k.par
 Ana_FourierT     < FourierT1.par

}
```

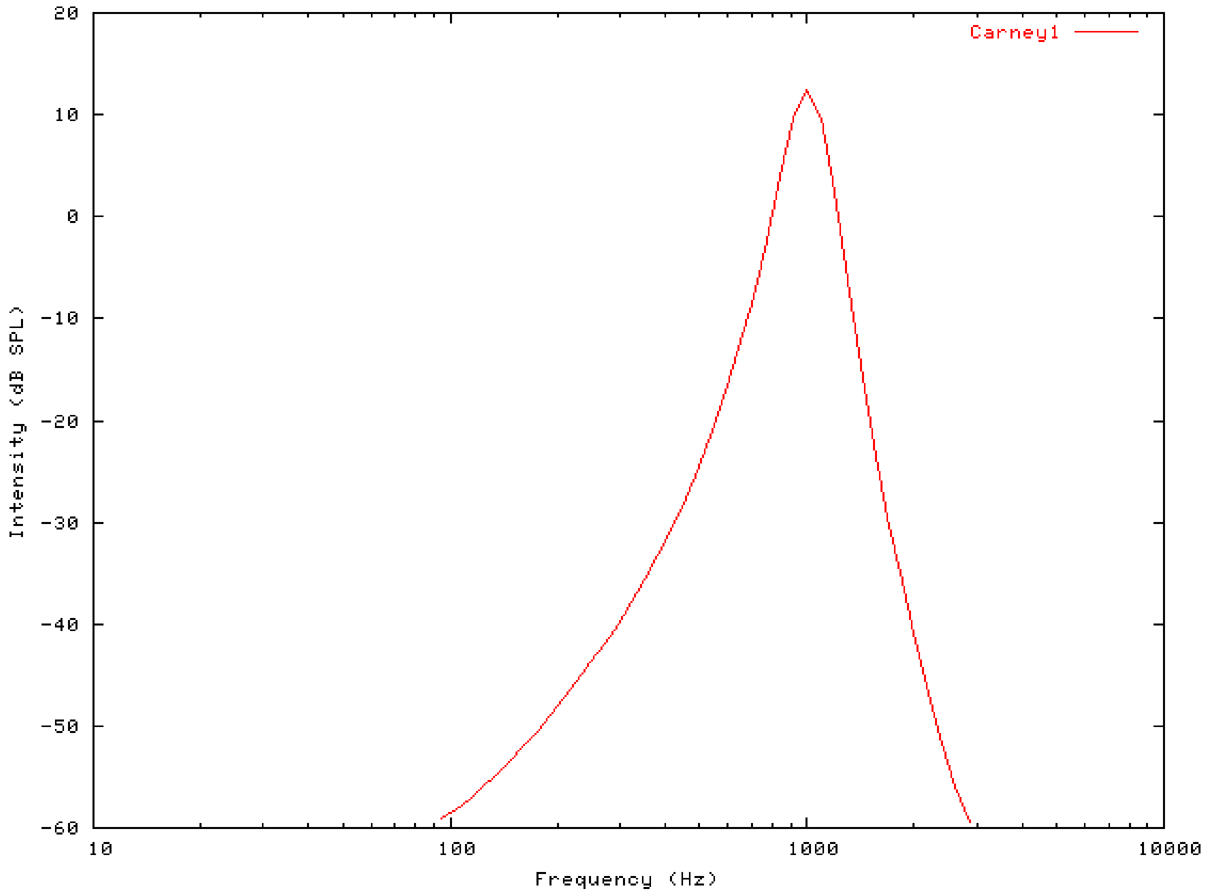
Figure 5.3. Fourier analysis of the Carney BM model response to a single toneCF = 1 kHz.



Filter Shape: AutoTest/BM/Carney1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Carney1.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_Carney       < Carney1k.par
 Ana_Intensity    < Intensty1.par

}
```

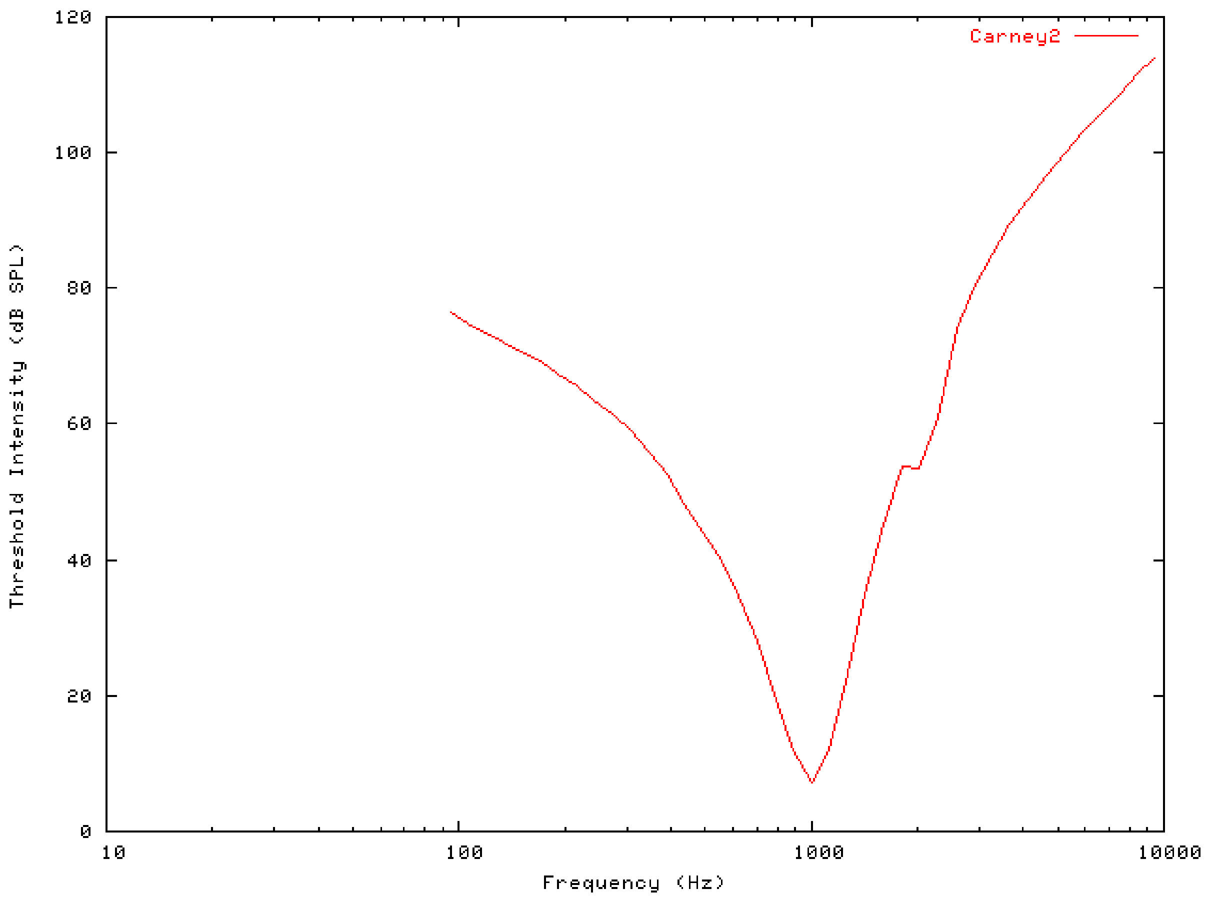
Figure 5.4. Carney BM model filter function.CF = 1 kHz.



Tuning Curve: AutoTest/BM/Carney2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Carney2.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_Carney       < Carney1k.par
 Ana_Intensity    < Intensty1.par

}
```

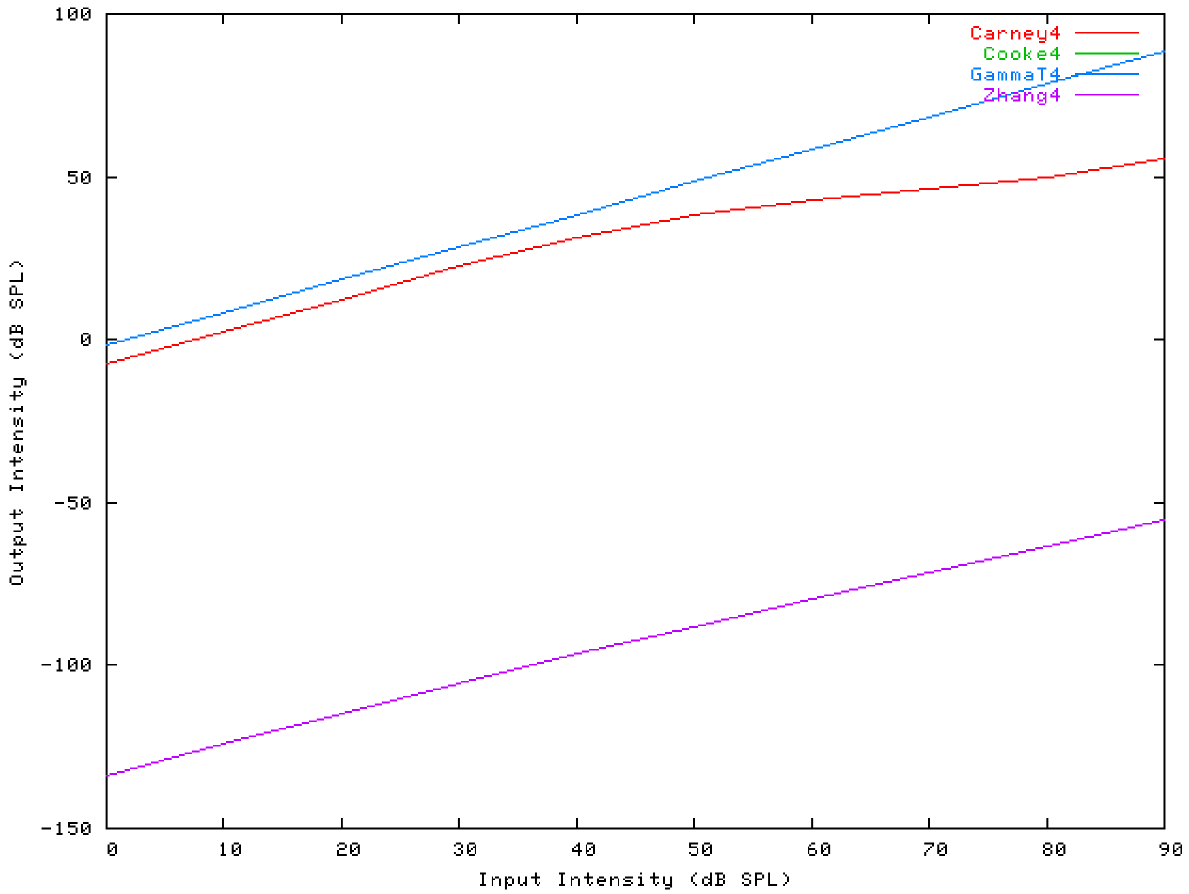
Figure 5.5. Carney BM model Tuning curve.CF = 1 kHz. This particular Carney model is unstable above ~ 4 kHz (see [5]



I/O function AutoTest/BM/Carney4.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Carney4.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_Carney       < Carney1k.par
 Ana_Intensity    < Intensty1.par

}
```

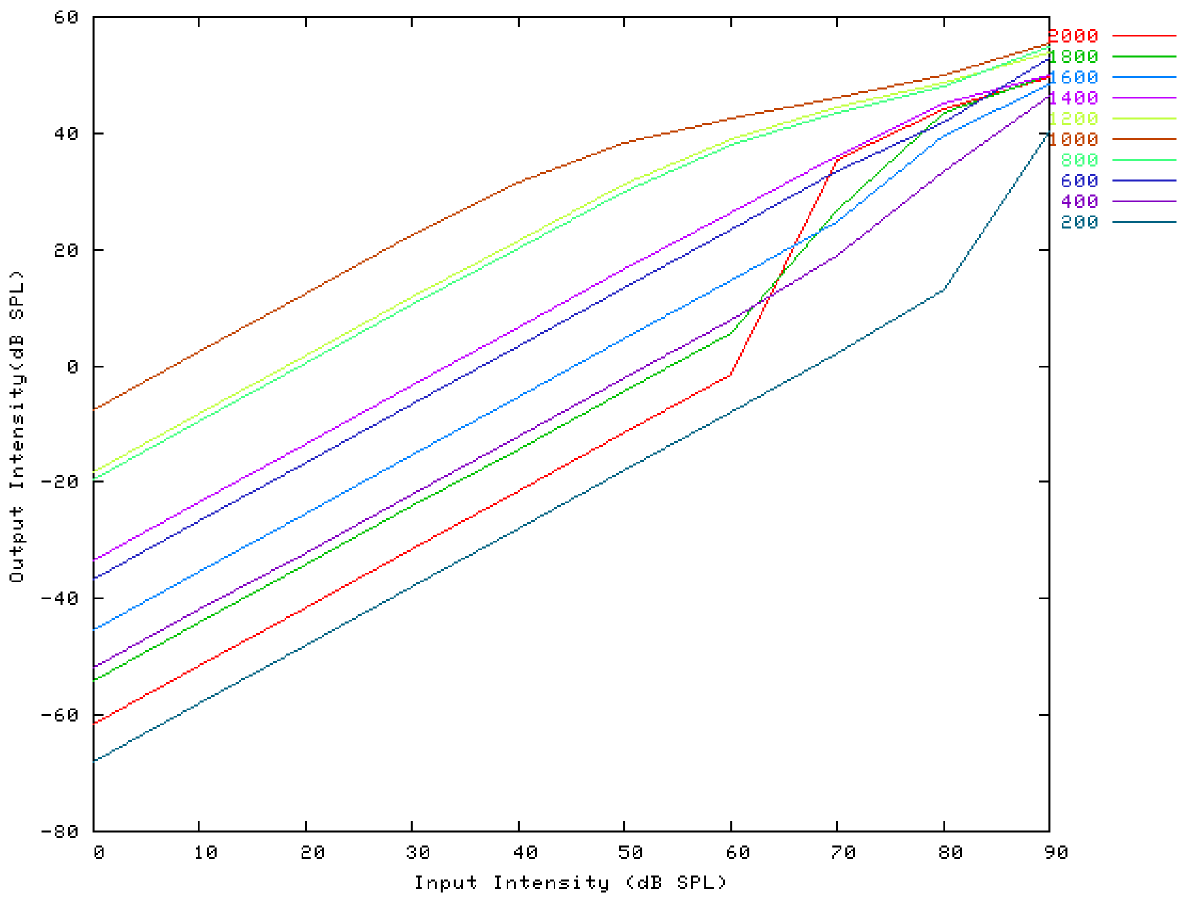
Figure 5.6. The input-output response of all the BM models.The ’GammaT’ and ’Cooke’ functions show the linear response.



Multiple I/O Functions: AutoTest/BM/Carney5.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Carney5.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_Carney       < Carney1k.par
 Ana_Intensity    < Intensty1.par

}
```

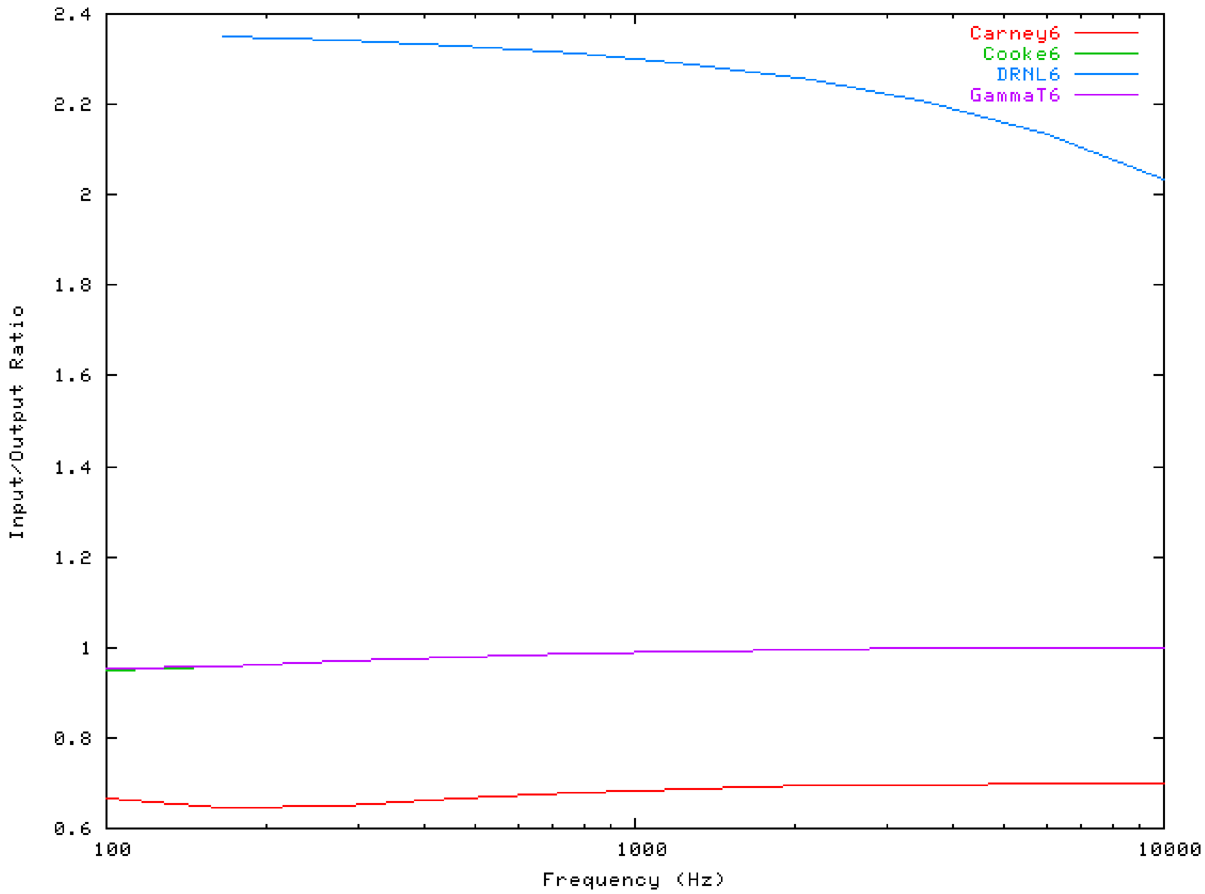
Figure 5.7. The input-output functions for the Carney BM model.Both on and off CF functions are shown.



I/O Ratio vs Frequency: AutoTest/BM/Carney6.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Carney6.sim
begin  {

 Stim_Puretone    < PTone2.par
 Trans_Gate      < Ramp1.par
 BM_Carney       < Carney1k.par
 Ana_Intensity    < Intensty1.par

}
```

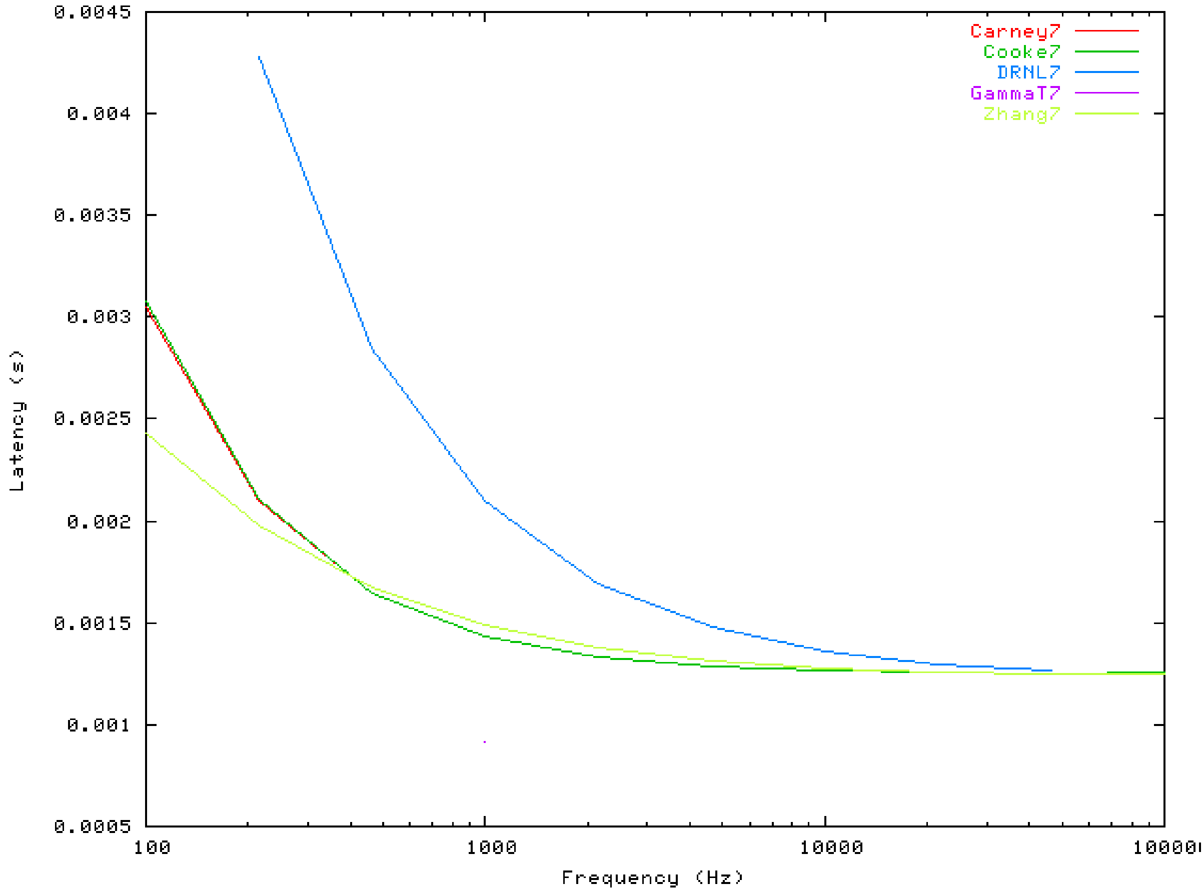
Figure 5.8. Input-output (I/O) ratio functions for all the BM models.The I/O characteristics are plotted against changing frequency.



Latency: AutoTest/BM/Carney7.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Carney7.sim
begin  {

 Stim_Click      < Click2.par
 Trans_Gate      < Ramp1.par
 BM_Carney      < CarneyLog1.par
 Ana_FindNextIndex < FindInd1.par

}
```

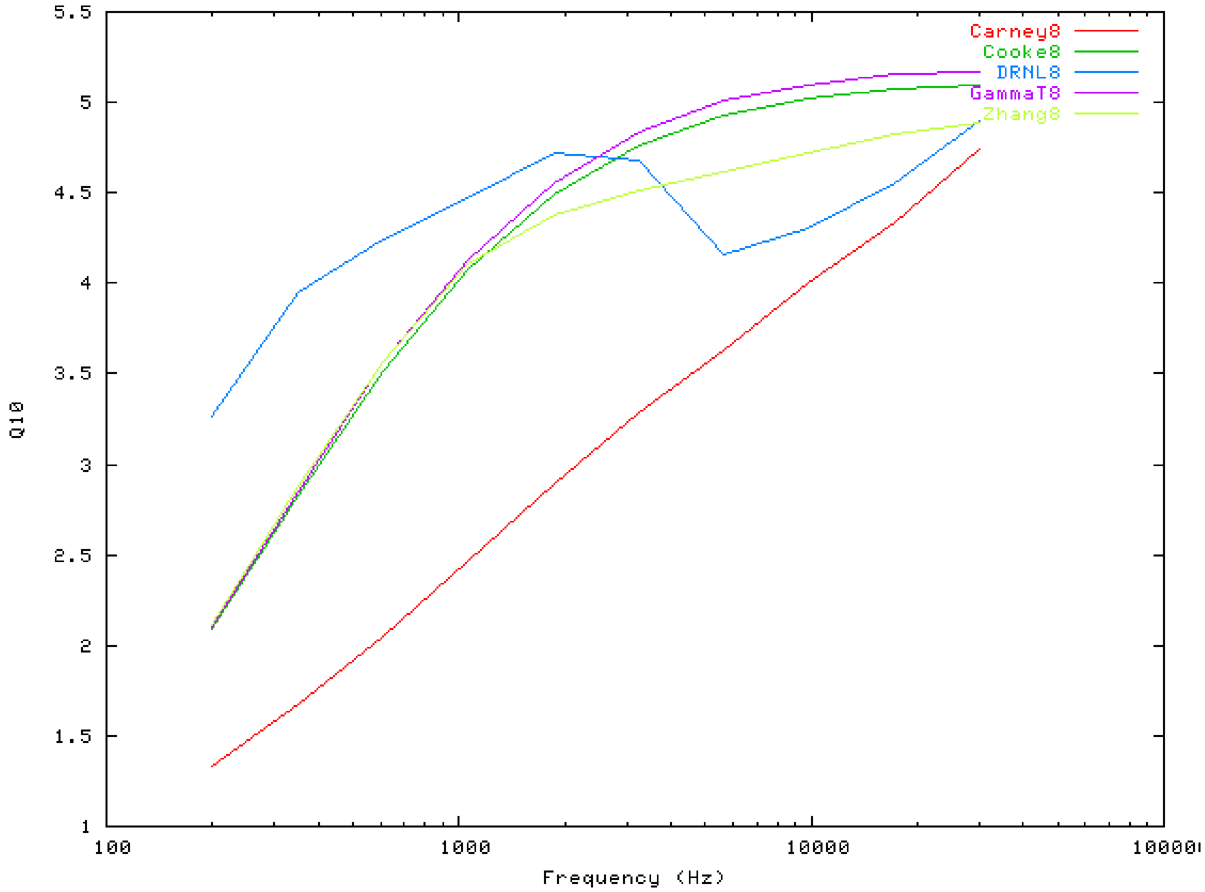
Figure 5.9. Latency of response for the BM models.



Q vs Frequency: AutoTest/BM/Carney8.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Carney8.sim
begin  {

 Stim_Puretone    < PTone3.par
 Trans_Gate      < Ramp1.par
 BM_Carney       < Carney1k.par
 Ana_Intensity    < Intensty1.par

}
```

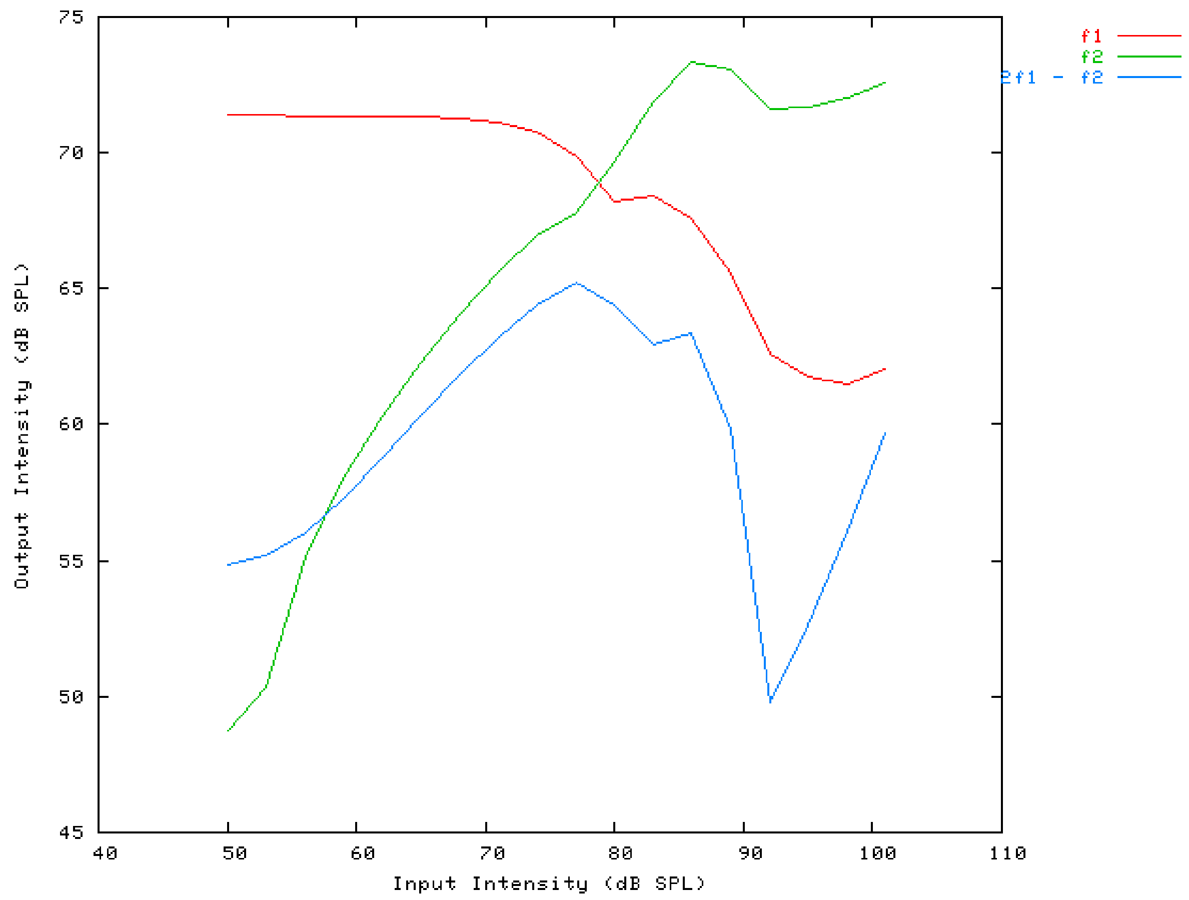
Figure 5.10. The Q10 tuning characteristics for the BM models.



Distortion Products: AutoTest/BM/Carney9.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Carney9.sim
begin  {

 Stim_Puretone_Multi < MPTone1.par
 Trans_Gate       < Ramp1.par
 BM_Carney       < Carney1k.par
 Ana_FourierT     < FourierT1.par

}
```

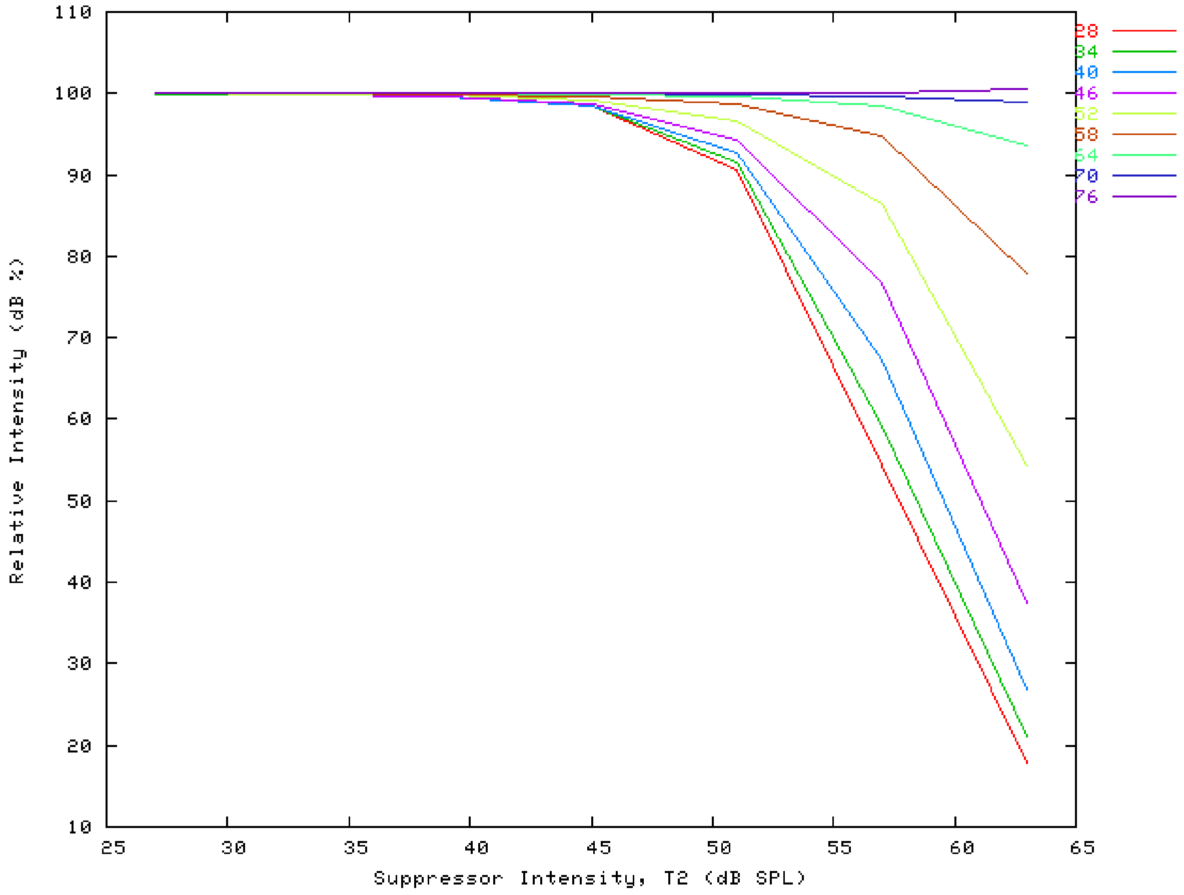
Figure 5.11. Distortion products investigation for the Carney BM model. A similar analysis paradigm as used in [29] was employed: the f1 tone was kept at a constant level while f2 was increased in level. The outputs shown are the respective frequency component intensities as determined from a fourier analysis. The response is similar to that of the DRNL filter, and both compare well with experimental data (see Figure 5.25, “Distortion products investigation for the DRNL BM model. A similar analysis paradigm as used in was employed: the f1 tone was kept at a constant level while f2 was increased in level. The outputs shown are the respective frequency component intensities as determined from a fourier analysis. The response is similar to that of the Carney filter, and both compare well with experimental data (see).”).



Two Tone Suppression: AutoTest/BM/Carney12.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Carney12.sim
begin  {

 Trans_Gate      < Ramp1.par
 BM_Carney       < Carney1k.par

}
```

Figure 5.12. Two-tone suppression response for the Carney BM model. Two stimuli are used. There is a single pure tone, and the other consisting of two pure tones: a probe tone, T1 and a suppresser tone, T2. The relative suppression is then calculated by dividing the unsuppressed model FFT modulus by the suppressed output modulus and multiplying by 100. This is calculated from the Fourier component at the probe frequency.



BM_Cooke: Linear BM (1991)

File name: MoBMCooke.[ch]

Description

This is an implementation of Martin Cooke’s model of the response of the Basilar Membrane. This module uses sine and cosine tables, so it requires a lot of memory. It has three different output options.

For further information on the use of BM filterbanks see the section called “Using the Basilar Membrane Filter Modules”.

Inputs	Arbitrary signal input.
Outputs	Produces multi-channel output signal, according to the CF list parameters.
Reference	

Module Parameters

Example 5.2. BM_Cooke Parameter File

```
BWI_FACTOR     1.019          ERB Bandwidth correction factor.
OUTPUT_MODE    BM             Output mode ('bm', 'pow', 'amp').
#CF List Parameters:-
DIAG_MODE      PARAMETERS     Diagnostic mode ('list' or 'parameters').
CF_MODE        LOG            Centre frequency mode ('single', 'ERB', 'ERB_n', 'log', 'linear',
                              'focal_log', 'user', 'human', 'cat', 'chinchilla', 'guinea-pig' or
                              'macaque').
MIN_CF         100            Minimum centre frequency (Hz).
MAX_CF         10000          Maximum centre frequency (Hz).
CHANNELS       30             No. of centre frequencies.
B_MODE         ERB            Bandwidth mode ('ERB', 'Custom_ERB', 'Guinea_Pig', 'user' or
                              'Nonlinear').
```

Table 5.1. BM_Cooke ’output_mode’ options

Output Mode	Description
bm	Produce basilar membrane detailed output.
pow	Produce power envelope output.
amp	Produce amplitude envelope output.

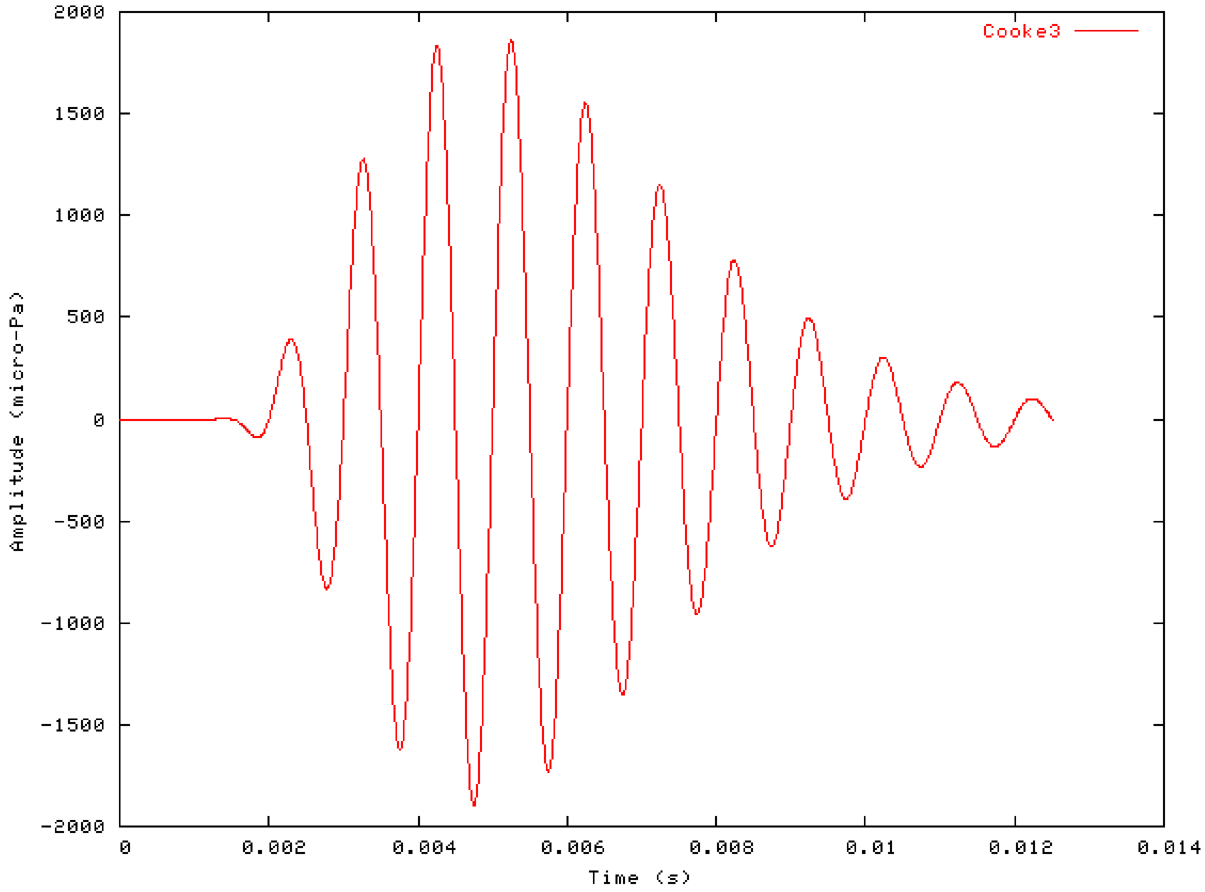
Examples Using the ’BM_Cooke’ Process Module

Some of the following tests require multiple observations using a range of stimuli and cannot be produced directly with a single run of the AMS program. These simulation scripts are run using the *AutoTest* program, which is employed to test the operation of this process module.

Impulse Response Example: AutoTest/BM/Cooke3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Cooke3.sim
begin  {

 Stim_Click      < Click1.par
 BM_Cooke       < CookeBM1k.par

}
```

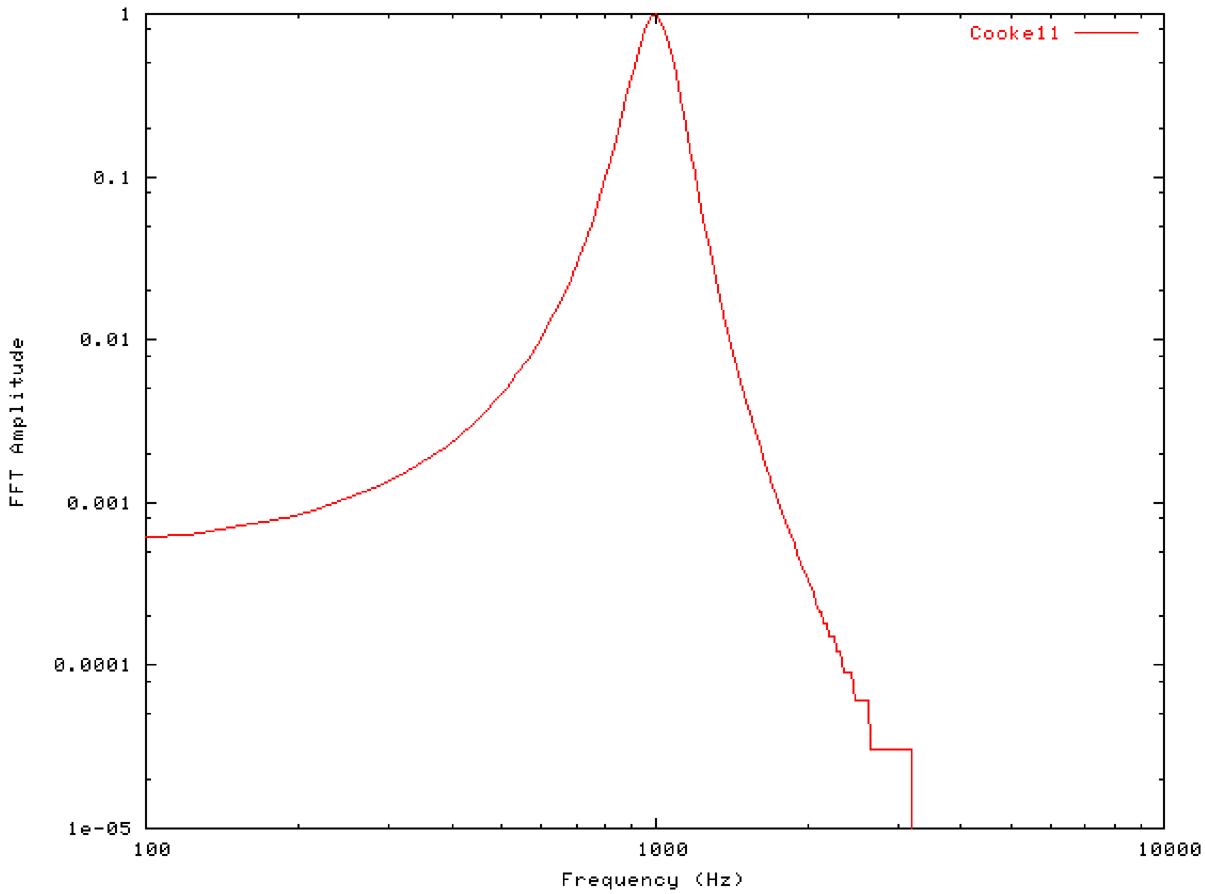
Figure 5.13. The impulse response of the Cooke BM modelCF = 1 kHz.



Gain Response Example: AutoTest/BM/Cooke11.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Cooke11.sim
begin  {

 Stim_Click      < Click3.par
 BM_Cooke       < CookeBM1k.par
 Ana_FourierT     < FourierT1.par

}
```

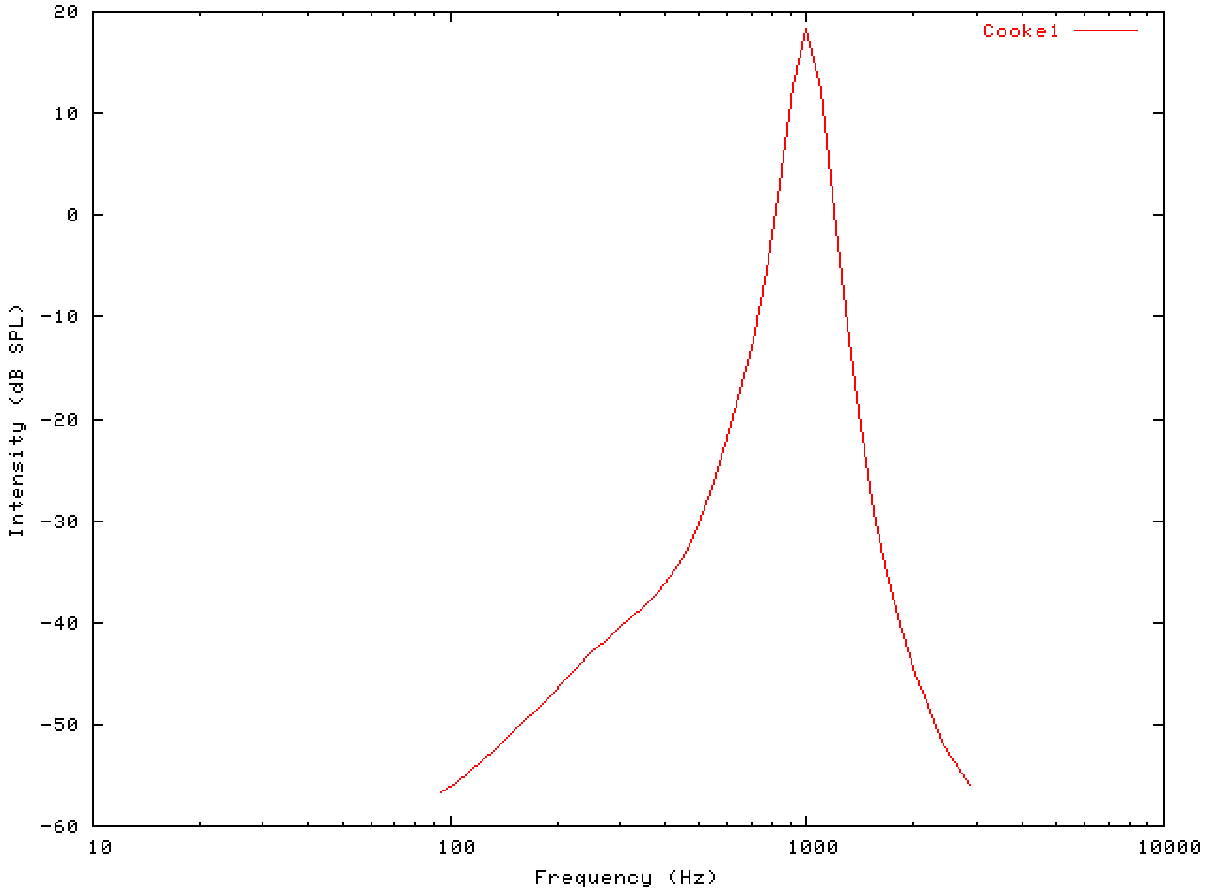
Figure 5.14. Fourier analysis of the Cooke BM model response to a single toneCF = 1 kHz.



Filter Shape: AutoTest/BM/Cooke1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Cooke1.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_Cooke        < CookeBM1k.par
 Ana_Intensity    < Intensty1.par

}
```

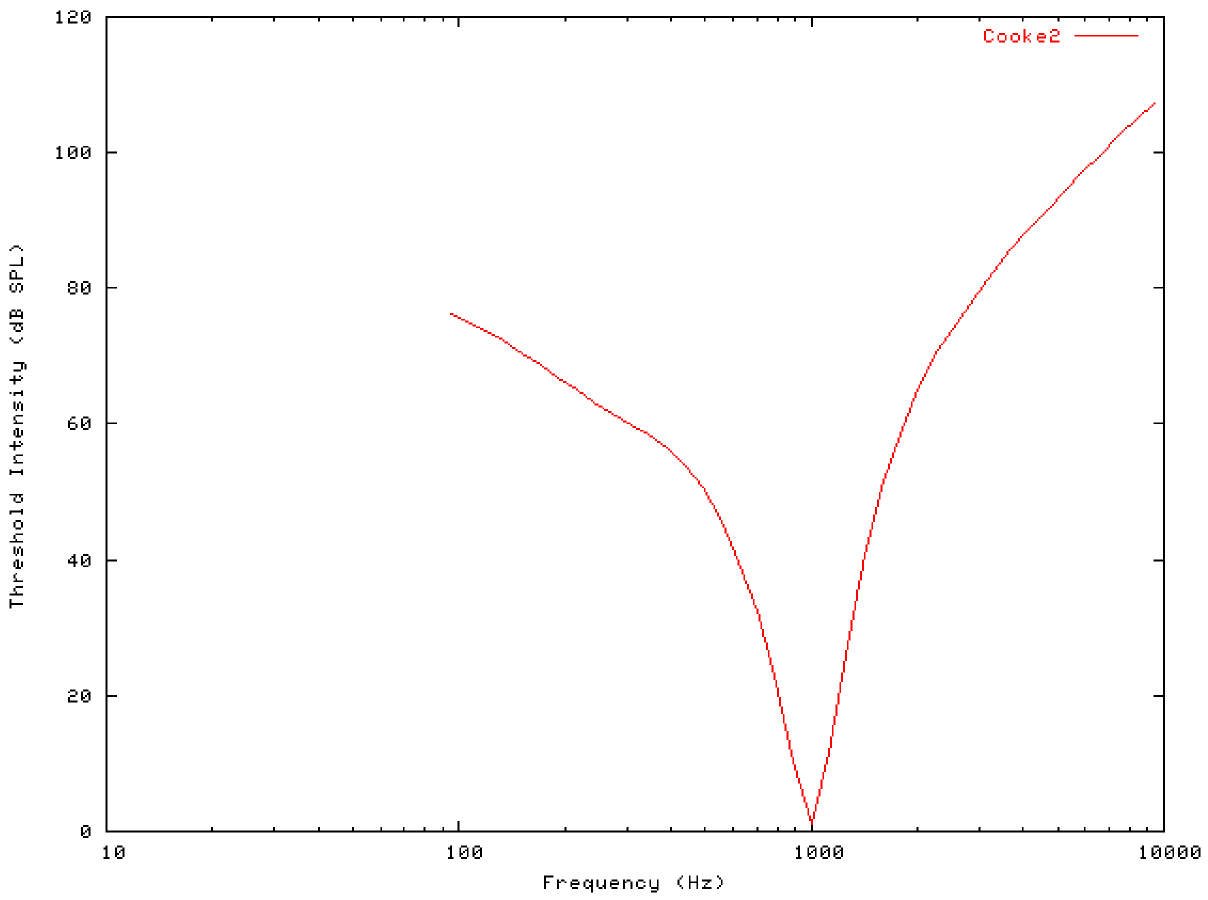
Figure 5.15. Cooke BM model filter function.CF = 1 kHz.



Tuning Curve: AutoTest/BM/Cooke2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Cooke2.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_Cooke        < CookeBM1k.par
 Ana_Intensity    < Intensty1.par

}
```

Figure 5.16. Cooke BM model Tuning curve.CF = 1 kHz.



I/O function AutoTest/BM/Cooke4.sim

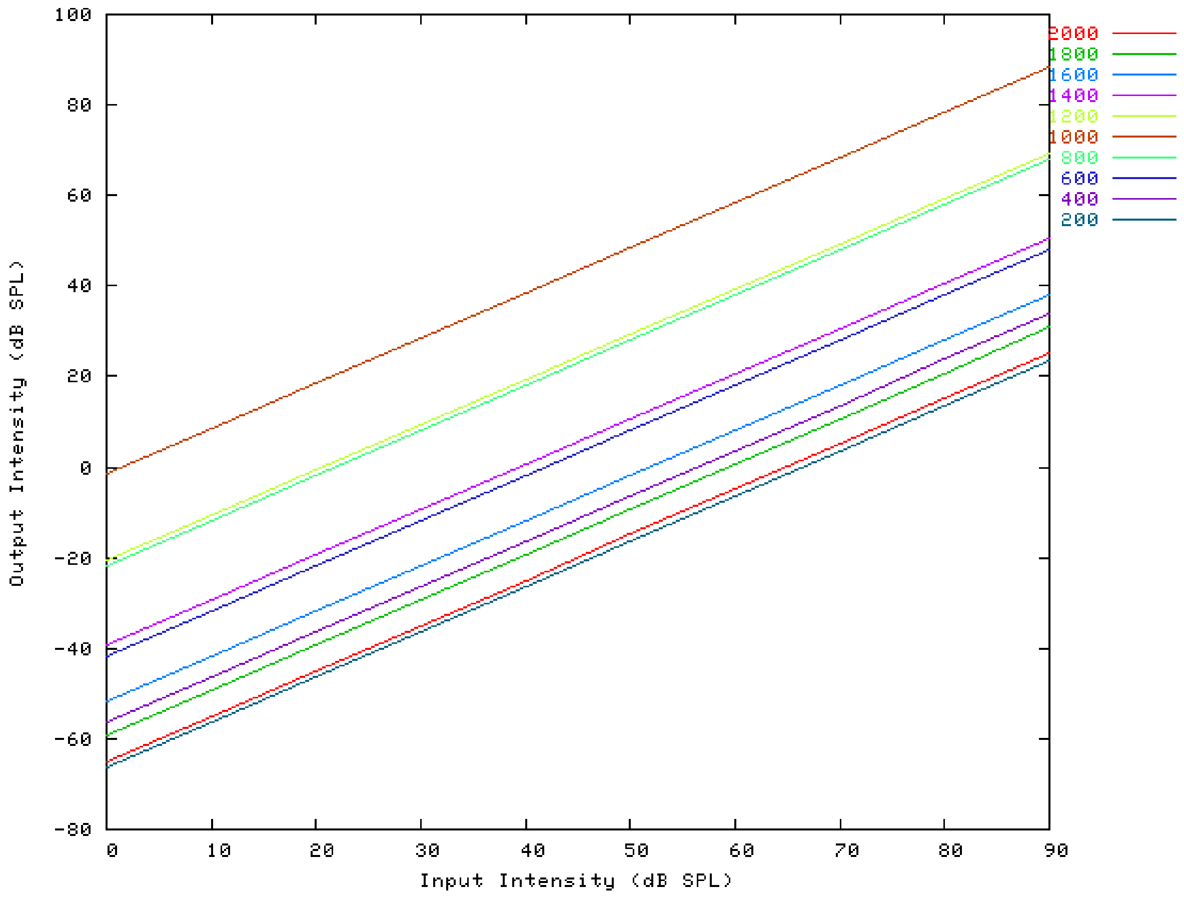
```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Cooke4.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_Cooke        < CookeBM1k.par
 Ana_Intensity    < Intensty1.par

}
```

See Figure 5.6, “The input-output response of all the BM models.The ’GammaT’ and ’Cooke’ functions show the linear response.”. Note that this filter has a linear input-output response.

Multiple I/O Functions: AutoTest/BM/Cooke5.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Cooke5.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_Cooke        < CookeBM1k.par
 Ana_Intensity    < Intensty1.par

}
```

Figure 5.17. The input-output functions for the Cooke BM model.Both on and off CF functions are shown.



I/O Ratio vs Frequency: AutoTest/BM/Cooke6.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Cooke6.sim
```

```
begin  {

 Stim_Puretone    < PTone2.par
 Trans_Gate      < Ramp1.par
 BM_Cooke       < CookeBM1k.par
 Ana_Intensity    < Intensty1.par

}
```

See Figure 5.8, “Input-output (I/O) ratio functions for all the BM models.The I/O characteristics are plotted against changing frequency.”. As this is a purely functional model, the fact that it attenuates signals is not important. It is the qualitative behaviour that is significant, rather than the quantitative behaviour.

Latency: AutoTest/BM/Cooke7.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Cooke7.sim
begin  {

 Stim_Click      < Click2.par
 Trans_Gate      < Ramp1.par
 BM_Cooke       < CookeBMLog1.par
 Ana_FindNextIndex < FindInd1.par

}
```

See Figure 5.9, “Latency of response for the BM models.”.

Q vs Frequency: AutoTest/BM/Cooke8.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Cooke8.sim
begin  {

 Stim_Puretone    < PTone3.par
 Trans_Gate      < Ramp1.par
 BM_Cooke       < CookeBM1k.par
 Ana_Intensity    < Intensty1.par

}
```

See Figure 5.10, “The Q10 tuning characteristics for the BM models.”.

BM_DRNL: Non-linear BM (1995)

File name: MoBMDRNL.[ch]

Description

This model implements the Dual Resonance Non-linear (DRNL) filter in a filterbank configuration. [23] Several of the parameters are computed using functions that depend on the centre frequency of each channel. For example, the *non-linear bandwidth*, *compression A (linear) scale*, *compression b (gain) scale*, *linear CF*, *linear bandwidth* and *linear filter scale* variables are all computed as a function of CF. This method is described in the section called “Using Parameter Arrays ”. Each parameter is defined by providing a function name, followed by a number of additional parameters as required by the function. The process module then uses the function to calculate the value of the respective variable parameter at each centre frequency. Tables of appropriate parameters are given in the publications cited.

If a single filter is required where the parameters can be set directly (rather than computed) use the BM_DRNL_Test process module instead (see the section called “BM_DRNL_Test: Test version of DRNL model”.

For further information on the use of BM filterbanks see the section called “Using the Basilar Membrane Filter Modules”.

Inputs	Arbitrary signal input.
Outputs	Produces multi-channel output signal, according to the CF list parameters.
Reference	[23]

Module Parameters

Example 5.3. BM_DRNL Parameter File

NL_GT_CASCADE	3	Nonlinear gammatone filter cascade.
NL_LP_CASCADE	4	Nonlinear low-pass filter cascade.
NONLINBWIDTH_MODE	LOG_FUNC1	Variable ’NonLinBwidth’ mode (’EXP_FUNC1’, ’LINEAR_FUNC1’, ’LOG_FUNC1’, ’LOG_FUNC2’ or ’POLY_FUNC1’).
NONLINBWIDTH_PARAMETER	0:0.8	
NONLINBWIDTH_PARAMETER	1:58	
COMPRSCALEA_MODE	LOG_FUNC1	Variable ’ComprScaleA’ mode (’EXP_FUNC1’, ’LINEAR_FUNC1’, ’LOG_FUNC1’, ’LOG_FUNC2’ or ’POLY_FUNC1’).
COMPRSCALEA_PARAMETER	0:1.67	
COMPRSCALEA_PARAMETER	1:0.45	
COMPRSCALEB_MODE	LOG_FUNC1	Variable ’ComprScaleB’ mode (’EXP_FUNC1’, ’LINEAR_FUNC1’, ’LOG_FUNC1’, ’LOG_FUNC2’ or ’POLY_FUNC1’).
COMPRSCALEB_PARAMETER	0:-5.85	
COMPRSCALEB_PARAMETER	1:0.875	
COMP_N_EXPON	0.1	Compression exponent, n (units).
L_GT_CASCADE	3	Linear gammatone filter cascade.
L_LP_CASCADE	4	Linear low-pass filter cascade.
LINCF_MODE	LOG_FUNC1	Variable ’LinCF’ mode (’EXP_FUNC1’, ’LINEAR_FUNC1’, ’LOG_FUNC1’, ’LOG_FUNC2’ or ’POLY_FUNC1’).
LINCF_PARAMETER	0:0.14	
LINCF_PARAMETER	1:0.95	
LINBWIDTH_MODE	LOG_FUNC1	Variable ’LinBwidth’ mode (’EXP_FUNC1’, ’LINEAR_FUNC1’, ’LOG_FUNC1’, ’LOG_FUNC2’ or ’POLY_FUNC1’).
LINBWIDTH_PARAMETER	0:1.3	
LINBWIDTH_PARAMETER	1:0.53	
LINSCALEG_MODE	LOG_FUNC1	Variable ’linScaleG’ mode (’EXP_FUNC1’, ’LINEAR_FUNC1’, ’LOG_FUNC1’, ’LOG_FUNC2’ or ’POLY_FUNC1’).
LINSCALEG_PARAMETER	0:5.48	
LINSCALEG_PARAMETER	1:-0.97	
#CF List Parameters:-		
DIAG_MODE	PARAMETERS	Diagnostic mode (’list’ or ’parameters’).
CF_MODE	LOG	Centre frequency mode (’single’, ’ERB’, ’ERB_n’, ’log’, ’linear’, ’focal_log’, ’user’, ’human’, ’cat’, ’chinchilla’, ’guinea-pig’ or ’macaque’).
MIN_CF	100	Minimum centre frequency (Hz).
MAX_CF	10000	Maximum centre frequency (Hz).
CHANNELS	30	No. of centre frequencies.
B_MODE	INTERNAL_STATIC	Bandwidth mode (’ERB’, ’Custom_ERB’, ’Guinea_Pig’, ’user’ or ’Nonlinear’).

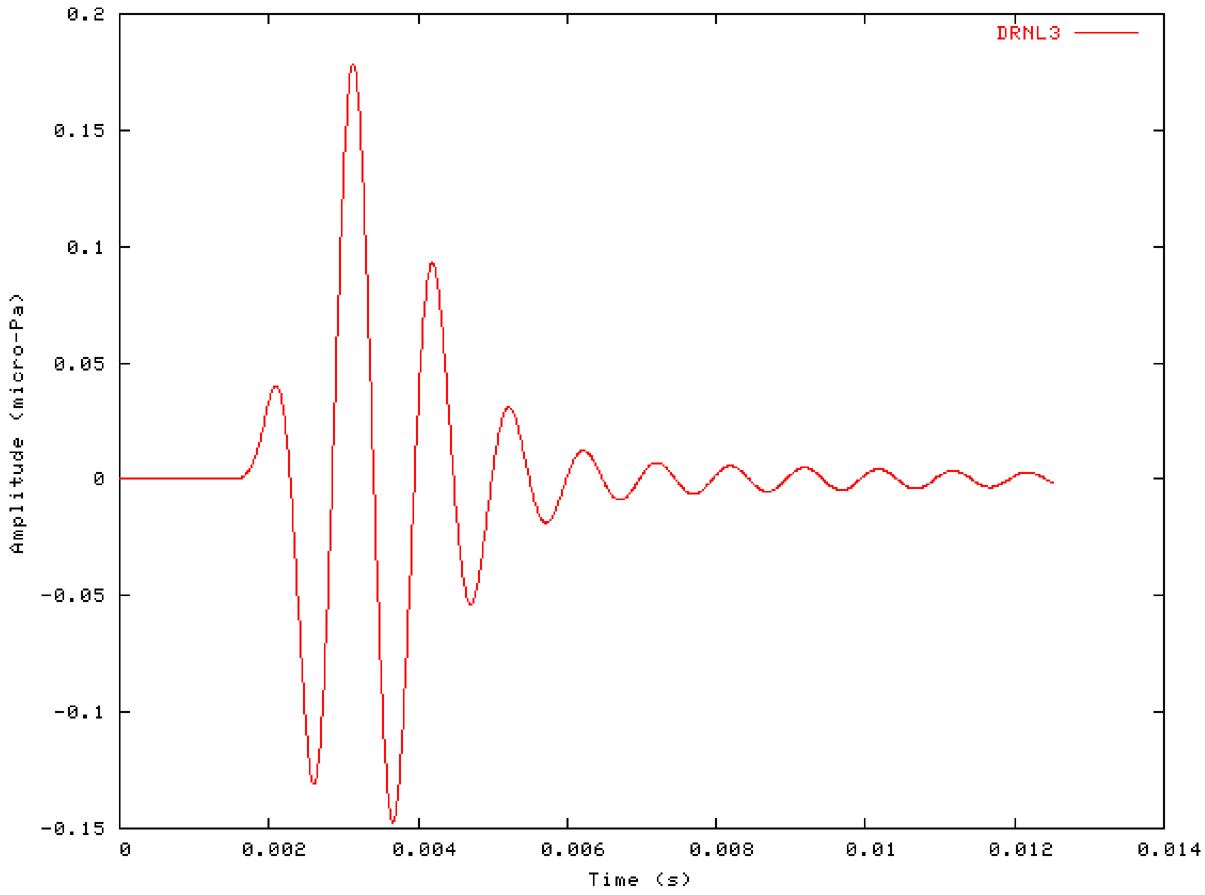
Examples Using the ’BM_DRNL’ Process Module

Some of the following tests require multiple observations using a range of stimuli and cannot be produced directly with a single run of the AMS program. These simulation scripts are run using the *AutoTest* program, which is employed to test the operation of this process module.

Impulse Response Example: AutoTest/BM/DRNL3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/DRNL3.sim
begin {

 Stim_Click      < ClickHu1.par
 Filt_BandPass    < PEHuY0.par
 BM_DRNL        < DRNL1k.par

}
```

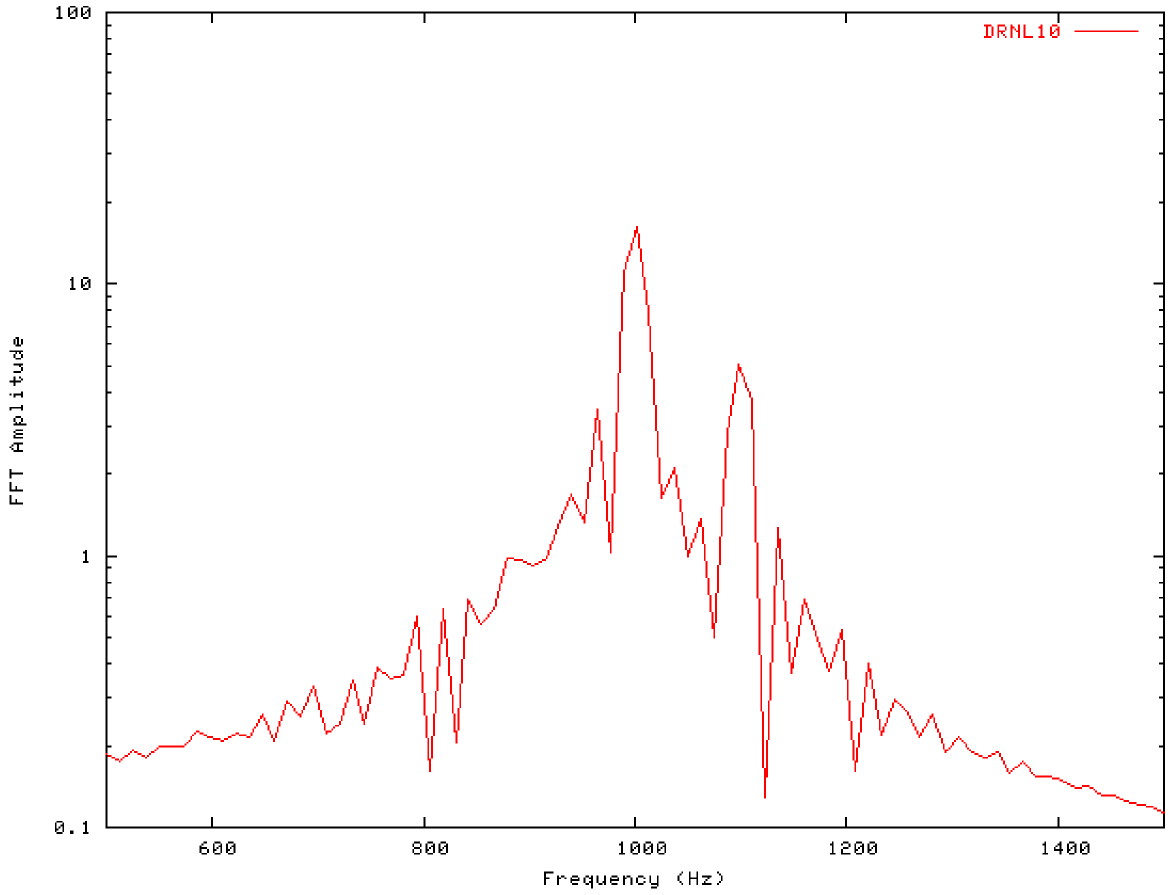
Figure 5.18. The impulse response of the DRNL BM modelCF = 1 kHz.This model's 'ringing' response is shorter than that for the linear gammatone filter.



FT Analysis Example: AutoTest/BM/DRNL10.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/DRNL10.sim
begin  {

 Stim_Puretone_Multi < MPTone2.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY0.par
 BM_DRNL        < DRNL1k.par
 Ana_FourierT     < FourierT1.par

}
```

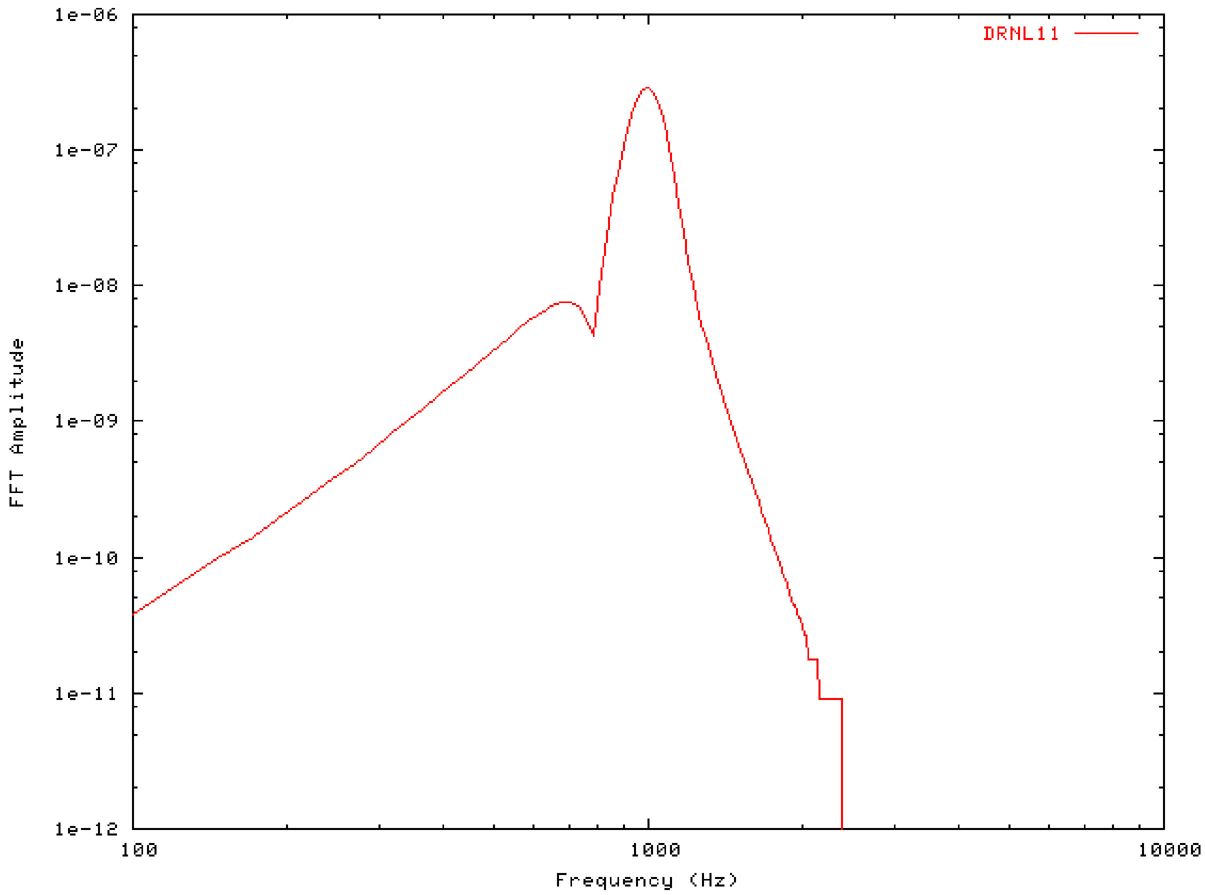
Figure 5.19. Fourier analysis of the DRNL BM model response to a two-tone stimulus.CF = 1 kHz.



Gain Response Example: AutoTest/BM/DRNL11.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.
# ./BM/DRNL11.sim
begin  {

 Stim_Click      < Click3.par
 Filt_BandPass    < PEHuY0.par
 BM_DRNL        < DRNL1k.par
 Ana_FourierT     < FourierT1.par

}
```

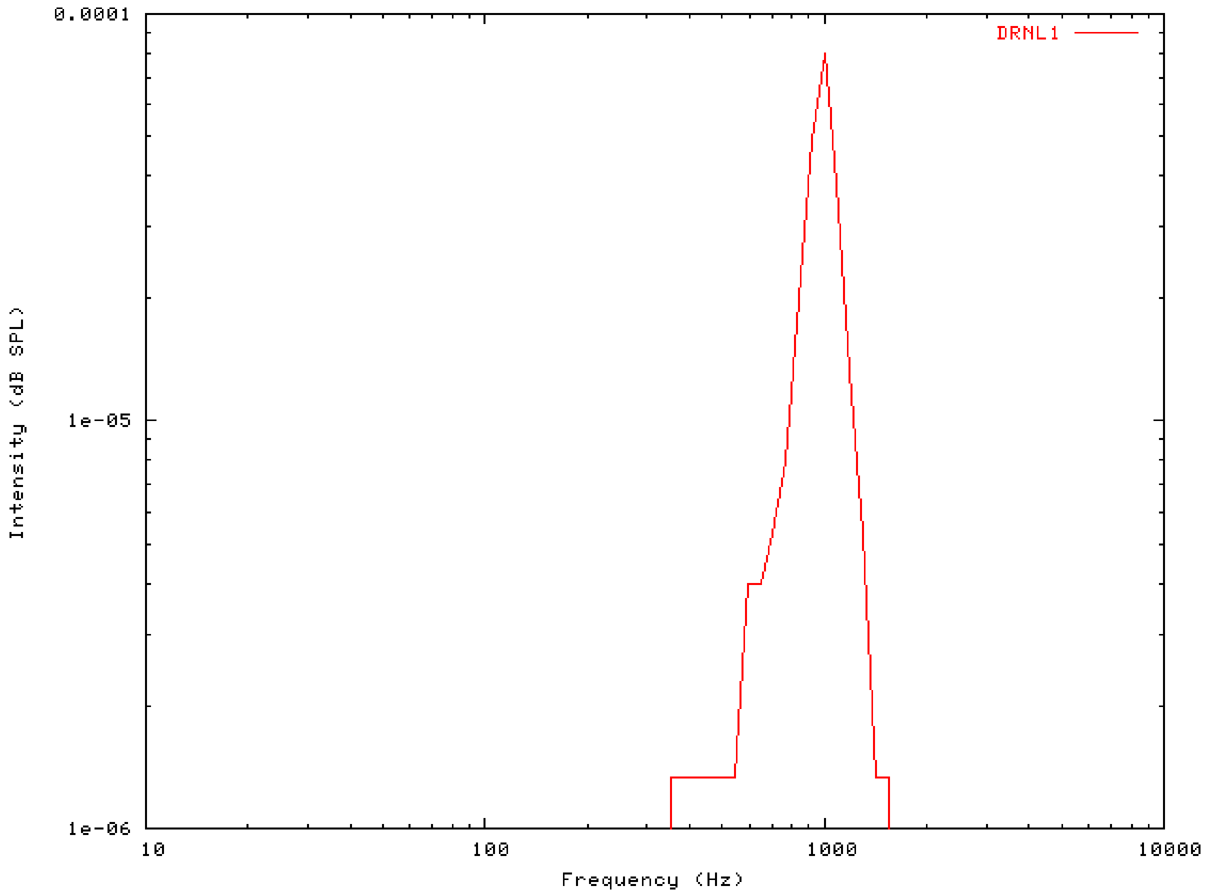
Figure 5.20. Fourier analysis of the DRNL BM model response to a single toneCF = 1 kHz.



Filter Shape: AutoTest/BM/DRNL1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/DRNL1.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY0.par
 BM_DRNL        < DRNL1k.par
 Ana_FindBin     < FindBin1.par

}
```

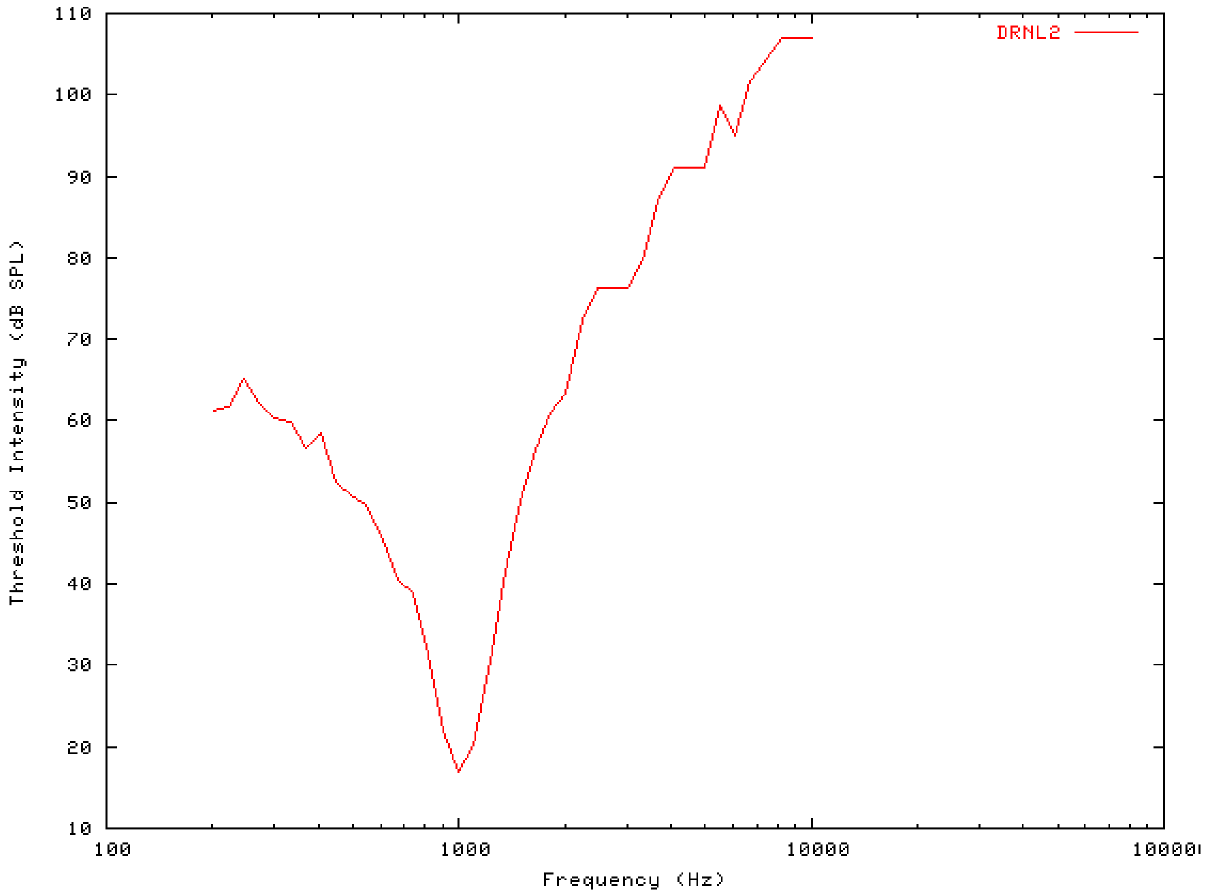
Figure 5.21. DRNL BM model filter function.CF = 1 kHz.



Tuning Curve: AutoTest/BM/DRNL2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/DRNL2.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY0.par
 BM_DRNL        < DRNL1k.par
 Ana_FindBin     < FindBin1.par
}
```

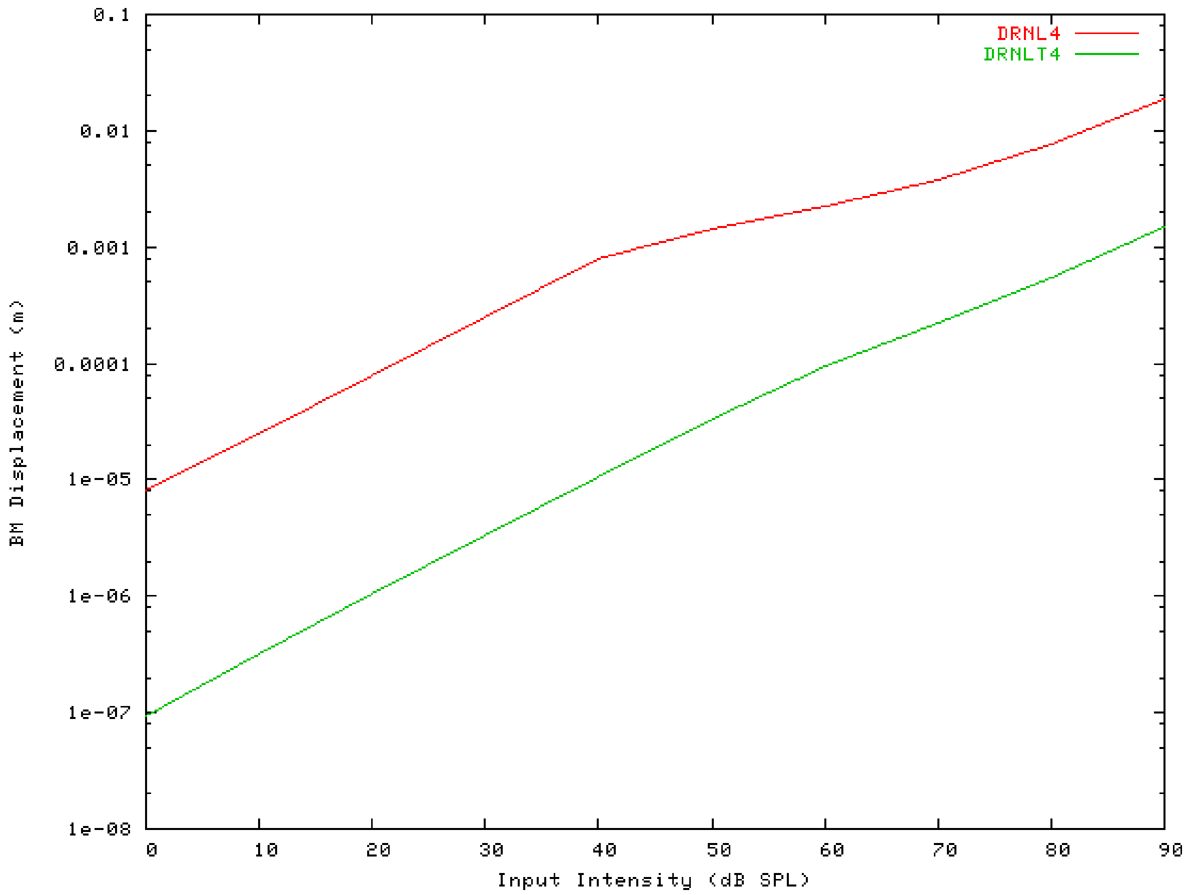
Figure 5.22. DRNL BM model Tuning curve.CF = 1 kHz.



I/O function AutoTest/BM/DRNL4.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/DRNL4.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY0.par
 BM_DRNL        < DRNL1k.par
 Ana_FindBin      < FindBin1.par

}
```

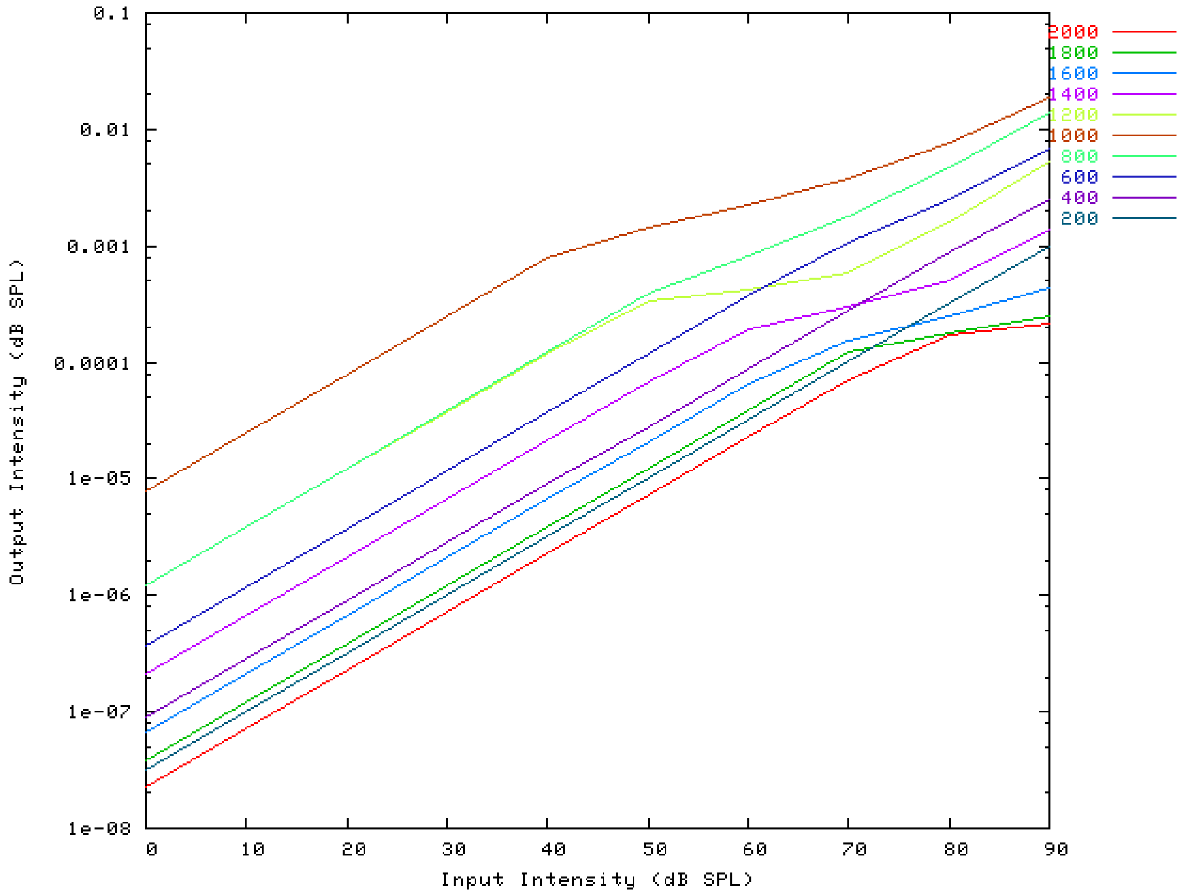
Figure 5.23. The input-output respnose of the DRNL BM modules’BM_DRNL’ and ’BM_DRNL_Test’ proces module responses.



Multiple I/O Functions: AutoTest/BM/DRNL5.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/DRNL5.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY0.par
 BM_DRNL        < DRNL1k.par
 Ana_FindBin     < FindBin1.par

}
```

Figure 5.24. The input-output functions for the DRNL BM model.Both on and off CF functions are shown. Note that the off-CF input-output functions show compression at higher intensities.



I/O Ratio vs Frequency: AutoTest/BM/DRNL6.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/DRNL6.sim
begin  {

 Stim_Puretone    < PTone2.par
 Trans_Gate      < Ramp1.par
 BM_DRNL        < DRNL1k.par
 Ana_Intensity    < Intensty1.par

}
```

See Figure 5.8, “Input-output (I/O) ratio functions for all the BM models.The I/O characteristics are plotted against changing frequency.”.

Latency: AutoTest/BM/DRNL7.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/DRNL7.sim
begin  {

 Stim_Click     < Click2.par
 Trans_Gate     < Ramp1.par
 Filt_BandPass   < PEHuY0.par
 BM_DRNL       < DRNLLog1.par
 Ana_FindNextIndex < FindInd1.par

}
```

See Figure 5.9, “Latency of response for the BM models.”.

Q vs Frequency: AutoTest/BM/DRNL8.sim

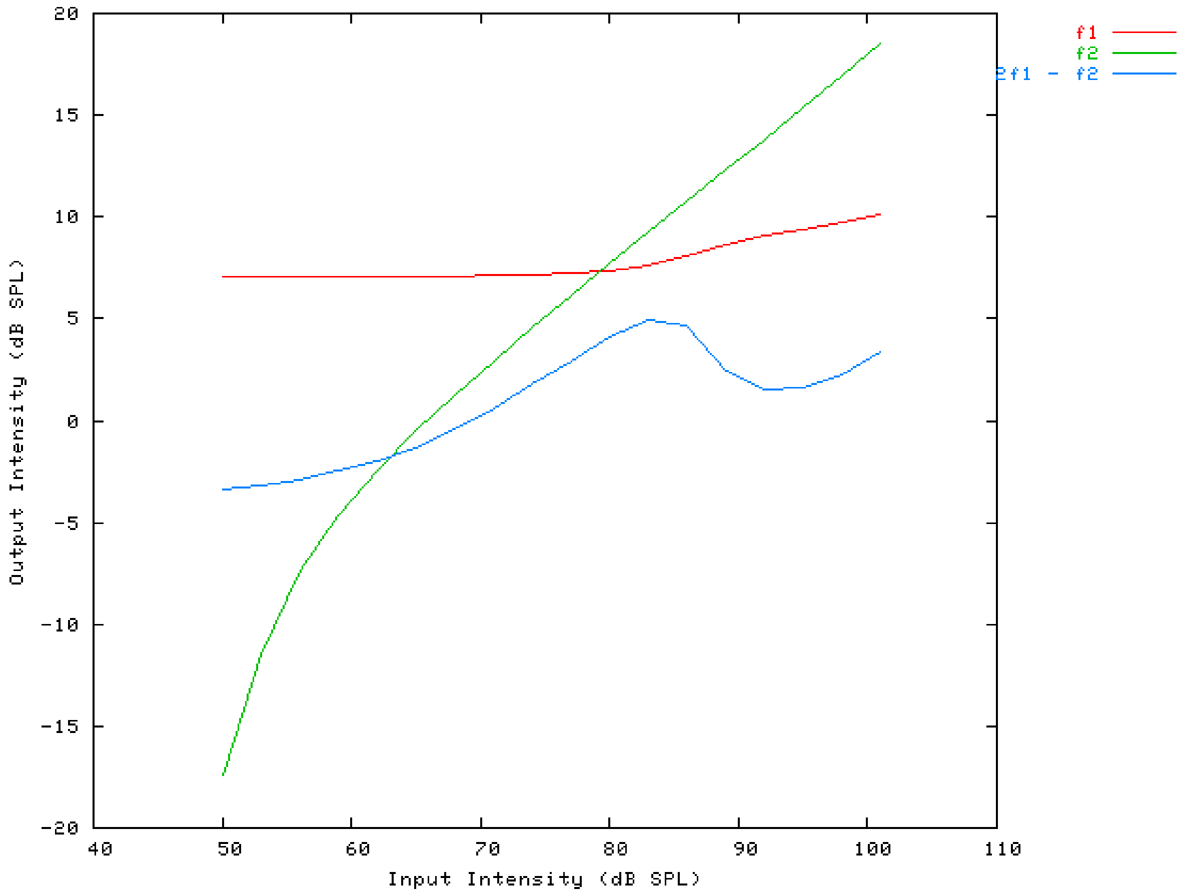
```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/DRNL8.sim
begin  {

 Stim_Puretone   < PTone3.par
 Trans_Gate     < Ramp1.par
 Filt_BandPass   < PEHuY0.par
 BM_DRNL       < DRNL1k.par
 Ana_Intensity   < Intensty1.par

}
```

See Figure 5.10, “The Q10 tuning characteristics for the BM models.”.

Distortion Products: AutoTest/BM/DRNL9.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/DRNL9.sim
begin  {

 Stim_Puretone_Multi < MPTone1.par
 Trans_Gate     < Ramp1.par
 Filt_BandPass   < PEHuY0.par
 BM_DRNL       < DRNL1k.par
 Ana_FourierT    < FourierT1.par
}
```

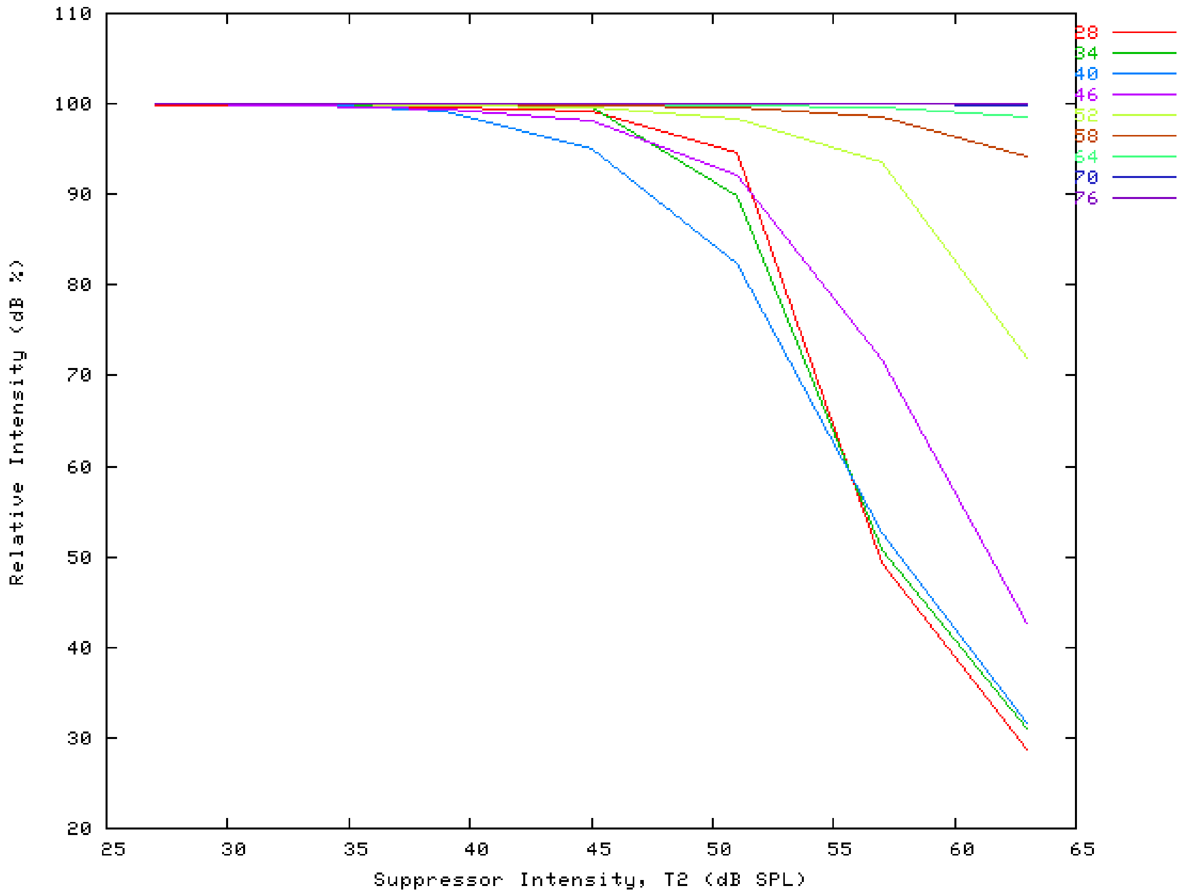
Figure 5.25. Distortion products investigation for the DRNL BM model. A similar analysis paradigm as used in [29] was employed: the f1 tone was kept at a constant level while f2 was increased in level. The outputs shown are the respective frequency component intensities as determined from a fourier analysis. The response is similar to that of the Carney filter, and both compare well with experimental data (see Figure 5.11, “Distortion products investigation for the Carney BM model. A similar analysis paradigm as used in was employed: the f1 tone was kept at a constant level while f2 was increased in level. The outputs shown are the respective frequency component intensities as determined from a fourier analysis. The response is similar to that of the DRNL filter, and both compare well with experimental data (see). ”).



Two Tone Suppression: AutoTest/BM/DRNL12.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/DRNL12.sim
begin  {
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY0.par
 BM_DRNL        < DRNL1k.par

}
```

Figure 5.26. Two-tone suppression response for the DRNL BM model. Two stimuli are used. There is a single pure tone, and the other consisting of two pure tones: a probe tone, T1 and a suppresser tone, T2. The relative suppression is then calculated by dividing the unsuppressed model FFT modulus by the suppressed output modulus and multiplying by 100. This is calculated from the Fourier component at the probe frequency.



BM_DRNL_Test: Test version of DRNL model

File name: MoBM0DRNL.[ch]

Description

DRNL (Dual resonance non-linear) single filter module. This module provides the user with comprehensive control of the model's individual parameters. If a full filterbank is required, see the section called "BM_DRNL: Non-linear BM (1995)".

For further information on the use of BM filterbanks see the section called “Using the Basilar Membrane Filter Modules”.

Inputs	Arbitrary signal input.
Outputs	Produces multi-channel output signal, according to the CF list parameters.
Reference	[23]

Module Parameters

Example 5.4. BM_DRNL Parameter File

```
NL_GT_CASCADE    3              Nonlinear gammatone filter cascade (’order’).
NL_LP_CASCADE    3              Nonlinear low-pass filter cascade (’order’).
LP_FILTER        BUTTERWORTH    Low-pass filter mode (’butterworth’ or ’Beauchamp’).
COMP_MODE        BROKEN_STICK1  Compression mode (’original’, ’inv_power’,
                                ’broken_stick1’ or ’upton_bstick’).
COMP_PAR         0:50
COMP_PAR         1:0.008
COMP_PAR         2:0.25
L_GT_CASCADE     2              Linear gammatone filter cascade (’order’).
L_LP_CASCADE     4              Linear low-pass filter cascade (’order’).
L_CF             700            Linear filter Centre Frequency (Hz).
L_BWIDTH         130            Linear filter band width (Hz).
L_SCALER         83             Linear filter scale (units).
#CF List Parameters:-
DIAG_MODE        PARAMETERS     Diagnostic mode (’list’ or ’parameters’).
CF_MODE          SINGLE         Centre frequency mode (’single’, ’ERB’, ’ERB_n’, ’log’,
                                ’linear’, ’focal_log’, ’user’, ’human’, ’cat’, ’chinchilla’,
                                ’guinea-pig’ or ’macaque’).
SINGLE_CF        750            Centre frequency (Hz).
B_MODE           ERB            Bandwidth mode (’ERB’, ’Custom_ERB’, ’Guinea_Pig’,
                                ’user’ or ’Nonlinear’).
```

Table 5.2. BM_DRNL_Test compression modes.

Mode	Formula	Parameters
Original	??	Threshold, Exponent.
Inv_Power	??	Shift, Slope.
Broken_Stick1	??	a, b, c.
Upton_BStick	??	a, b, c, d.

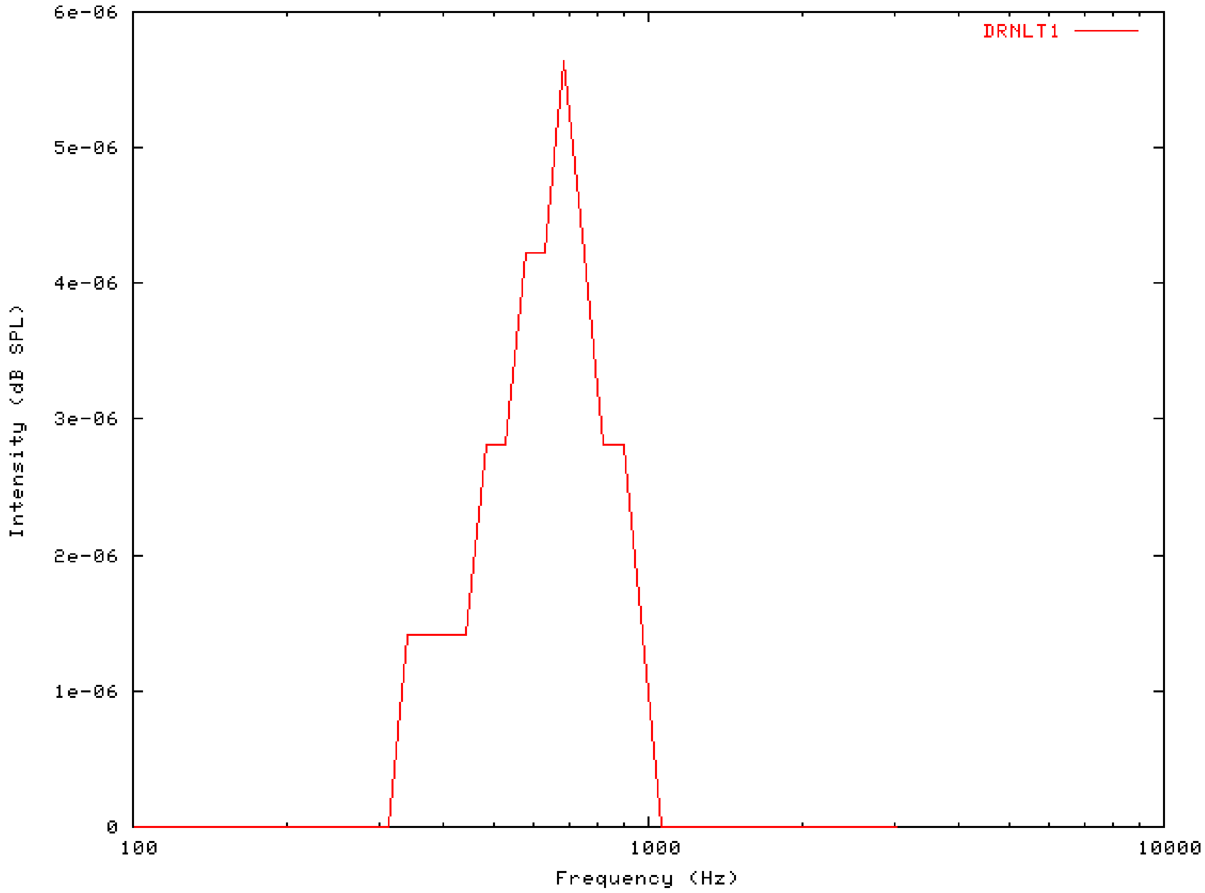
Examples Using the ’BM_DRNL’ Process Module

The following tests require multiple observations using a range of stimuli and cannot be produced directly with a single run of the AMS program. These simulation scripts are run using the *AutoTest* program, which is employed to test the operation of this process module.

Filter Shape: AutoTest/BM/DRNLT1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.
# ./BM/DRNLT1.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEChinch1.par
 BM_DRNL_Test     < DRNLT750.par
 Ana_FindBin      < FindBin1.par

}
```

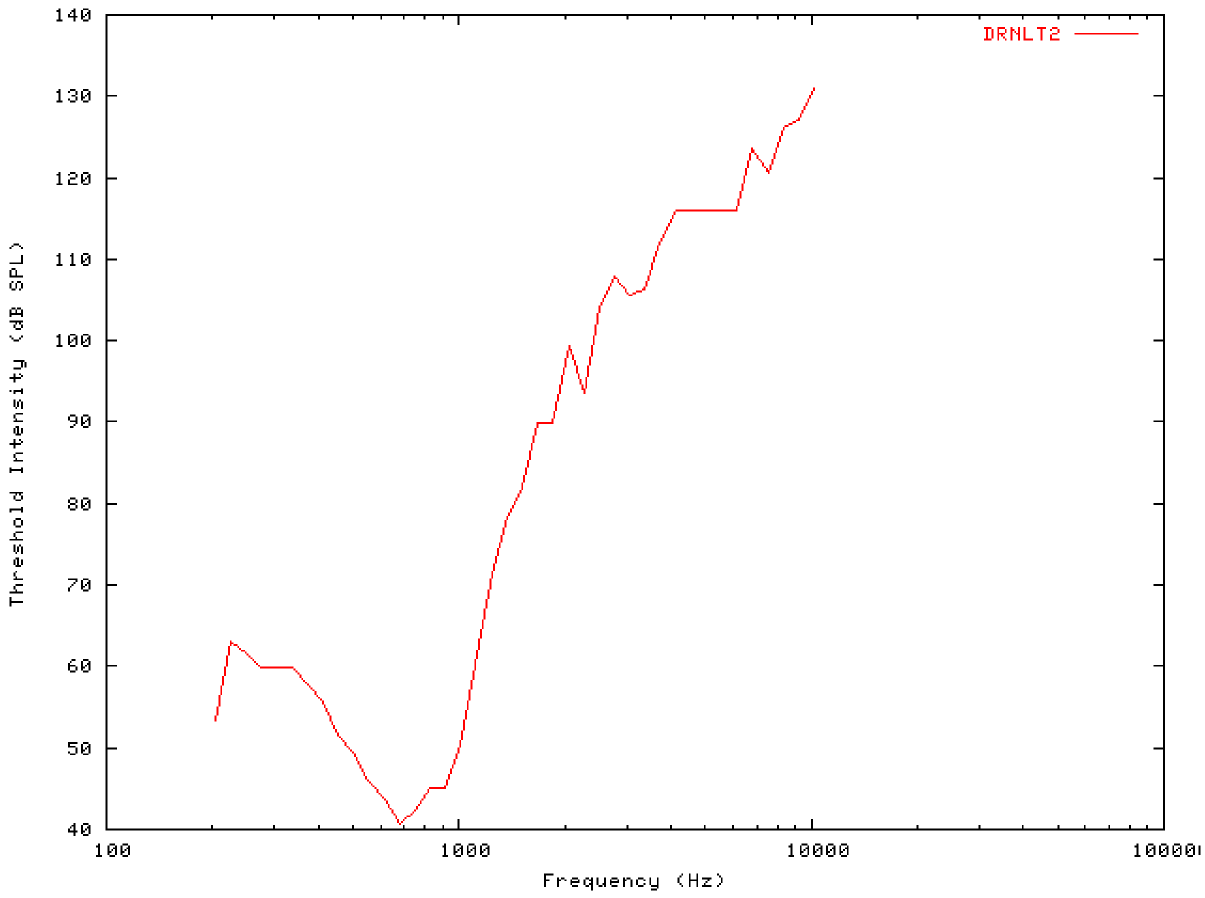
Figure 5.27. Filter function for the test version of the DRNL BM model.CF = 300 Hz.



Tuning Curve: AutoTest/BM/DRNLT2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/DRNLT2.sim
begin  {
 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEChinch1.par
 BM_DRNL_Test     < DRNLT750.par
 Ana_FindBin     < FindBin1.par

}
```

Figure 5.28. Tuning Curve for the test version of the DRNL BM model.CF = 300 Hz.



I/O function AutoTest/BM/DRNLT4.sim

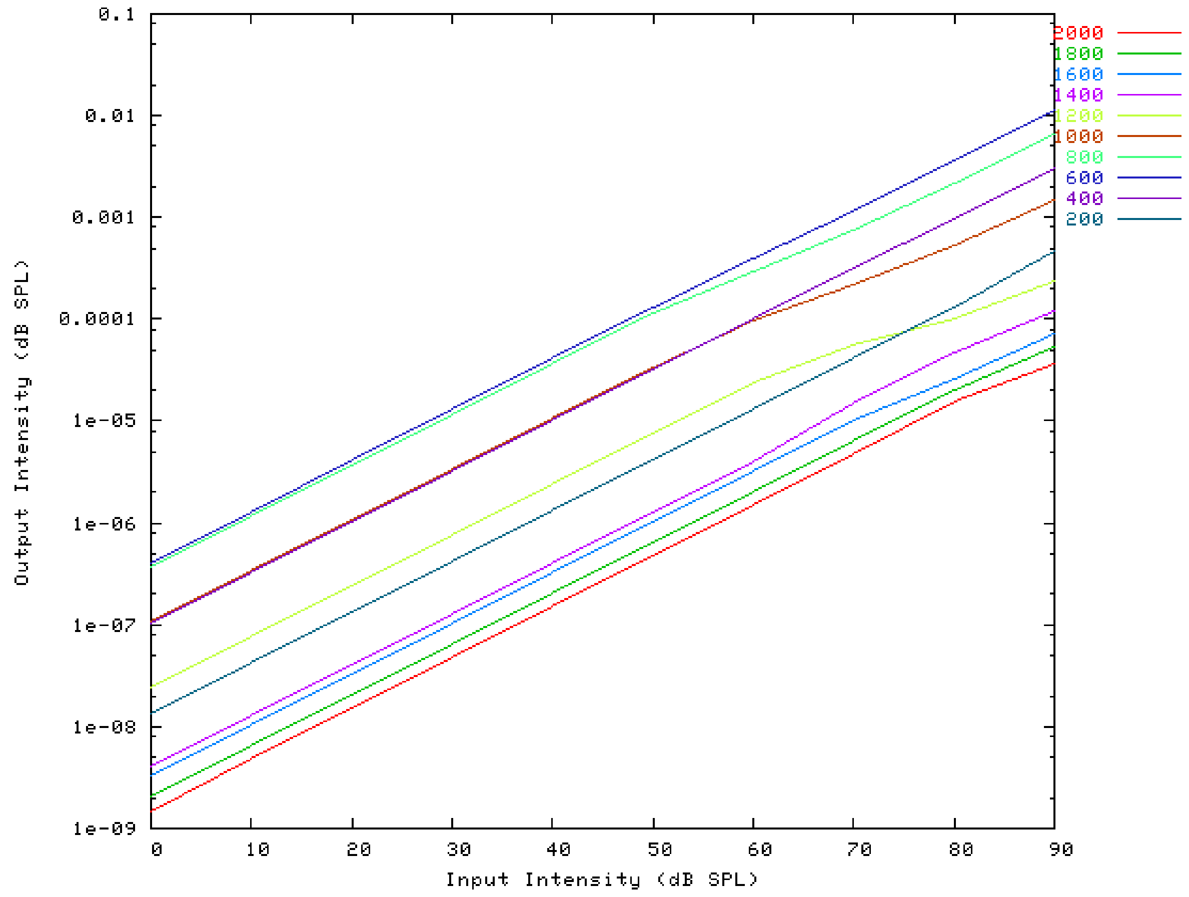
```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/DRNLT4.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEChinch1.par
 BM_DRNL_Test     < DRNLT750.par
 Ana_FindBin     < FindBin1.par

}
```

See Figure 5.23, “The input-output respnose of the DRNL BM modules’BM_DRNL’ and ’BM_DRNL_Test’ proces module responses.”.

Multiple I/O Functions: AutoTest/BM/DRNLT5.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/DRNLT5.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEChinch1.par
 BM_DRNL_Test     < DRNLT750.par
 Ana_FindBin     < FindBin1.par

}
```

Figure 5.29. The input-output functions for the test version of the DRNL BM model.Both on and off CF functions are shown.   Note that the off-CF input-output functions show compression at higher intensities.



I/O Ratio vs Frequency: AutoTest/BM/DRNLT6.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/DRNLT6.sim
begin  {

 Stim_Puretone    < PTone2.par
 Trans_Gate      < Ramp1.par
 BM_DRNL_Test     < DRNLT750.par
 Ana_Intensity    < Intensty1.par

}
```

See Figure 5.8, “Input-output (I/O) ratio functions for all the BM models.The I/O characteristics are plotted against changing frequency.”.

BM_GammaT: Linear BM

File name: MoBMGammaT.[ch]

Description

Simple, linear gamma tone filter basilar membrane model. The filters can be implemented as a full filterbank where the CF's and bandwidths are computed across a range or as a single filter where the CF and bandwidth are specified directly. A single site on the BM is modelled as a cascade of gammatone filters. A fourth-order gammatone filter is therefore modelled with a cascade value of 4. The bandwidth parameter applies to a single filter before cascading.

For further information on the use of BM filterbanks see the section called "Using the Basilar Membrane Filter Modules".

Inputs	Arbitrary signal input.
Outputs	Produces multi-channel output signal, according to the CF list parameters.
Reference	

Module Parameters

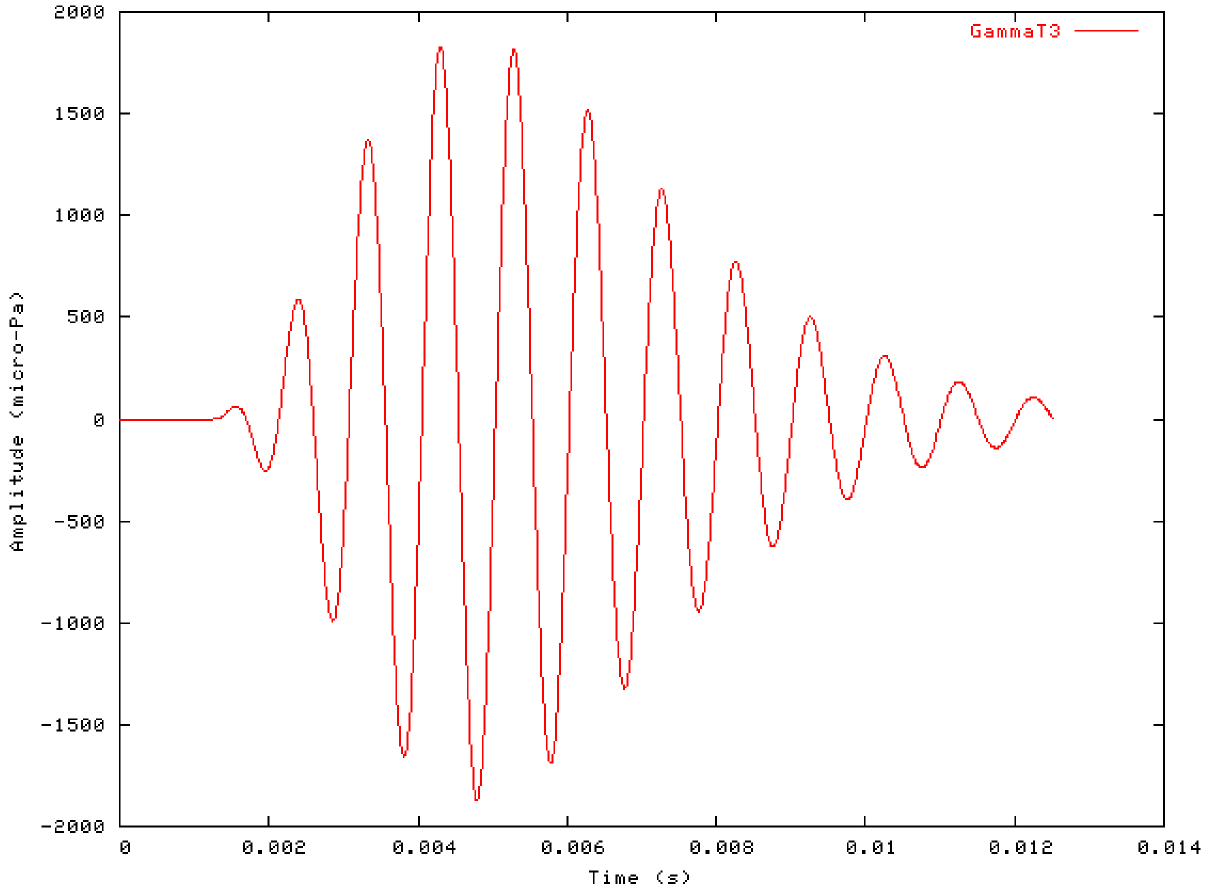
Example 5.5. BM_GammaT Parameter File

CASCADE	4	Filter cascade.
#CF List Parameters:-		
DIAG_MODE	PARAMETERS	Diagnostic mode ('list' or 'parameters').
CF_MODE	LOG	Centre frequency mode ('single', 'ERB', 'ERB_n', 'log', 'linear', 'focal_log', 'user', 'human', 'cat', 'chinchilla', 'guinea-pig' or 'macaque').
MIN_CF	100	Minimum centre frequency (Hz).
MAX_CF	10000	Maximum centre frequency (Hz).
CHANNELS	30	No. of centre frequencies.
B_MODE	ERB	Bandwidth mode ('ERB', 'Custom_ERB', 'Guinea_Pig', 'user' or 'Nonlinear').

Impulse Response Example: AutoTest/BM/GammaT3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/GammaT3.sim
begin  {

 Stim_Click      < Click1.par
 BM_GammaT      < GammaT1k.par

}
```

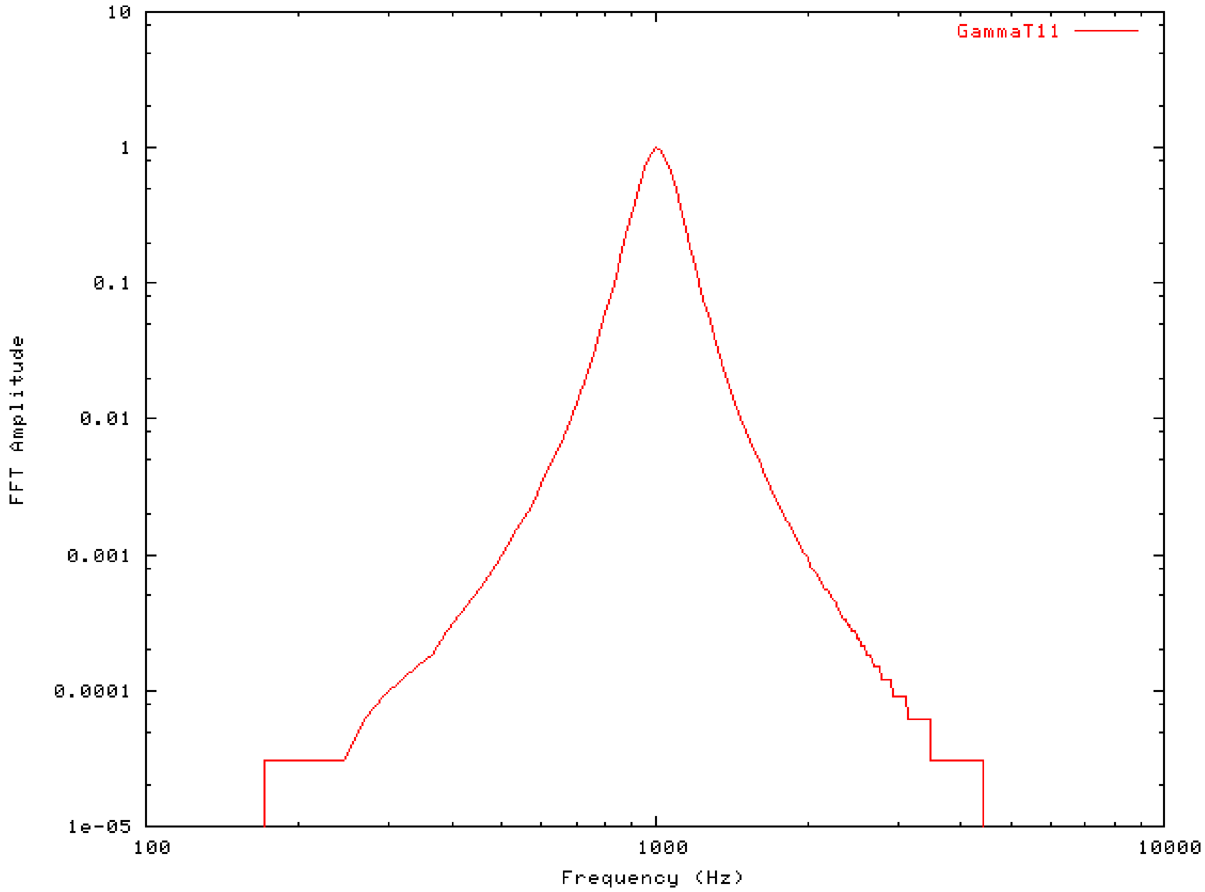
Figure 5.30. The impulse response of the GammaT BM modelCF = 1 kHz.



Gain Response Example: AutoTest/BM/GammaT11.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/GammaT11.sim
begin  {

 Stim_Click      < Click3.par
 BM_GammaT      < GammaT1k.par
 Ana_FourierT     < FourierT1.par

}
```

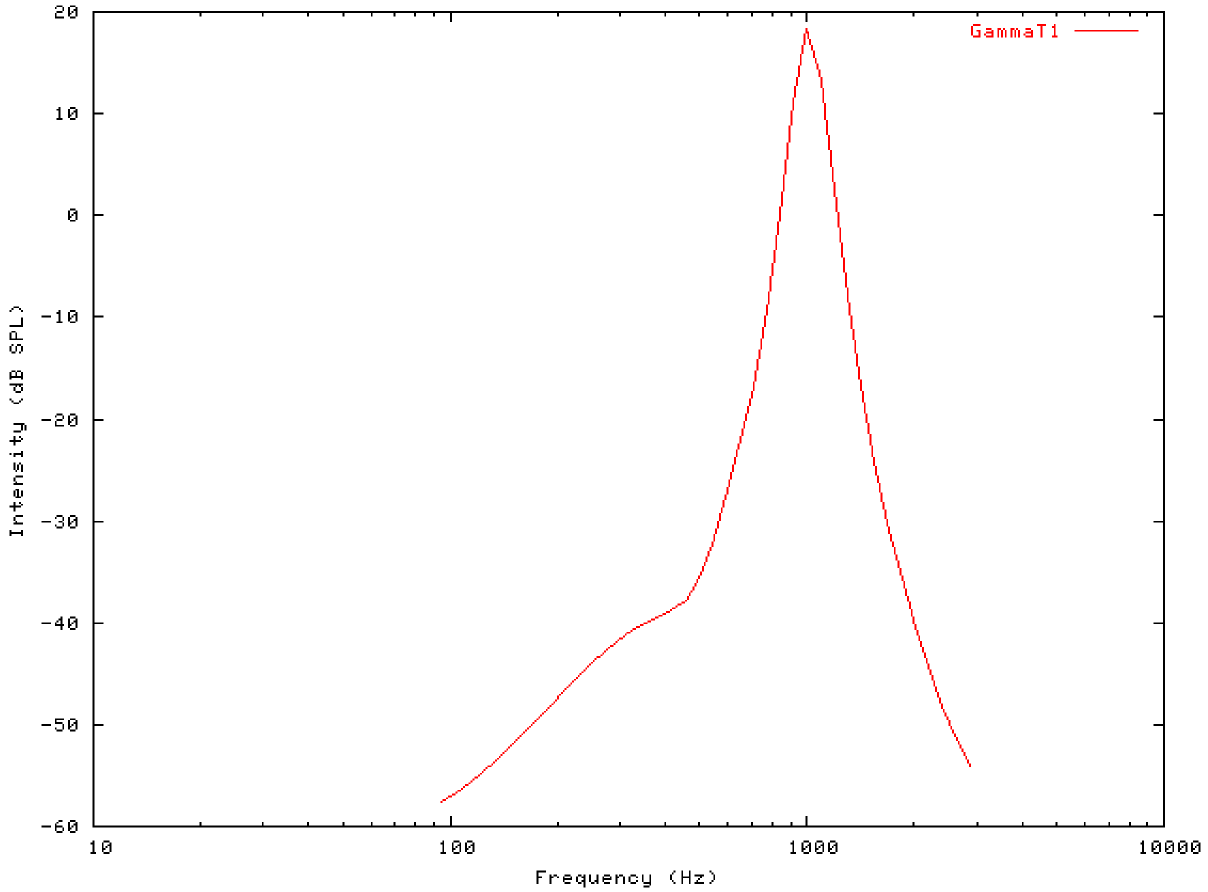
Figure 5.31. Fourier analysis of the GammaT BM model response to a single toneCF = 1 kHz.



Filter Shape: AutoTest/BM/GammaT1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/GammaT1.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_GammaT       < GammaT1k.par
 Ana_Intensity    < Intensty1.par

}
```

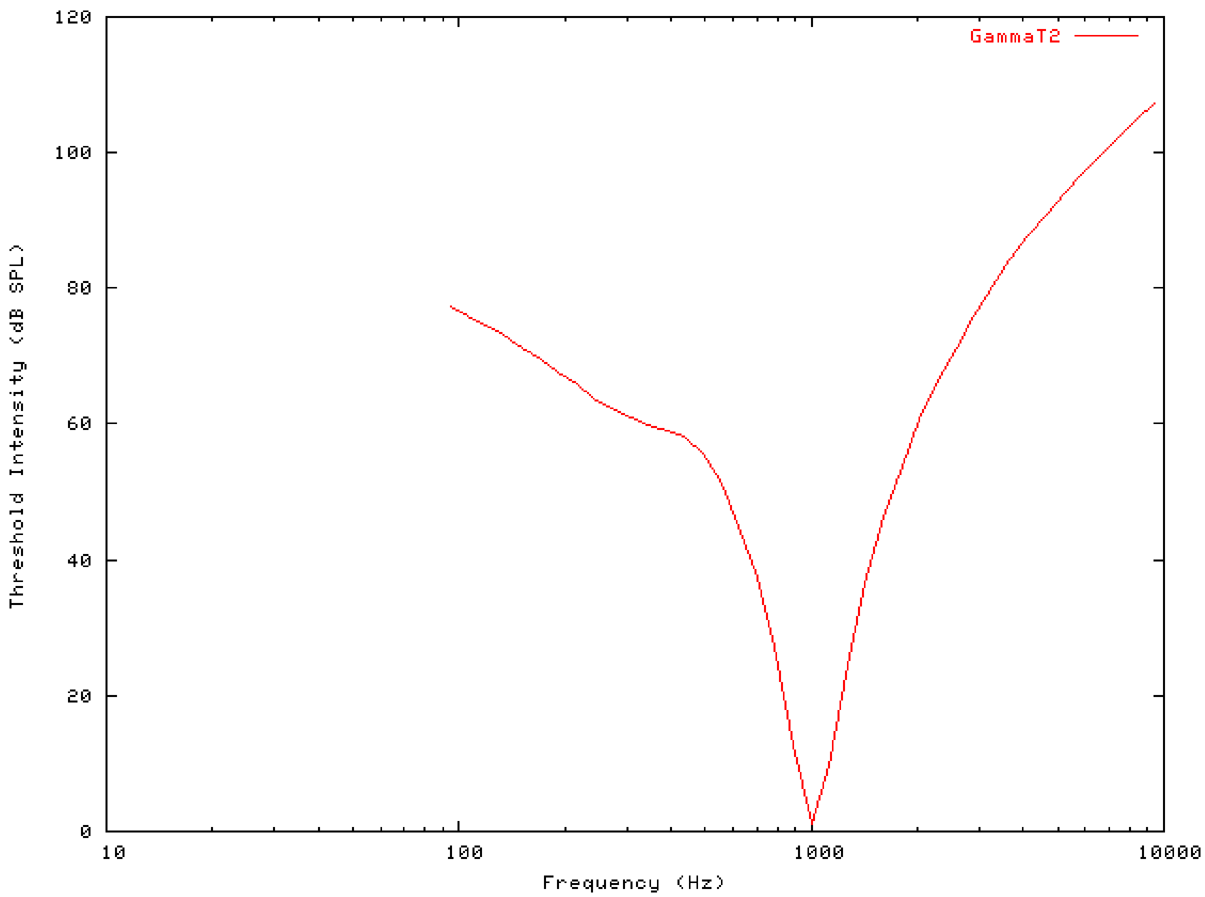
Figure 5.32. GammaT BM model filter function.CF = 1 kHz.



Tuning Curve: AutoTest/BM/GammaT2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/GammaT2.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_GammaT       < GammaT1k.par
 Ana_Intensity    < Intensty1.par

}
```

Figure 5.33. GammaT BM model Tuning curve.CF = 1 kHz.



I/O function AutoTest/BM/GammaT4.sim

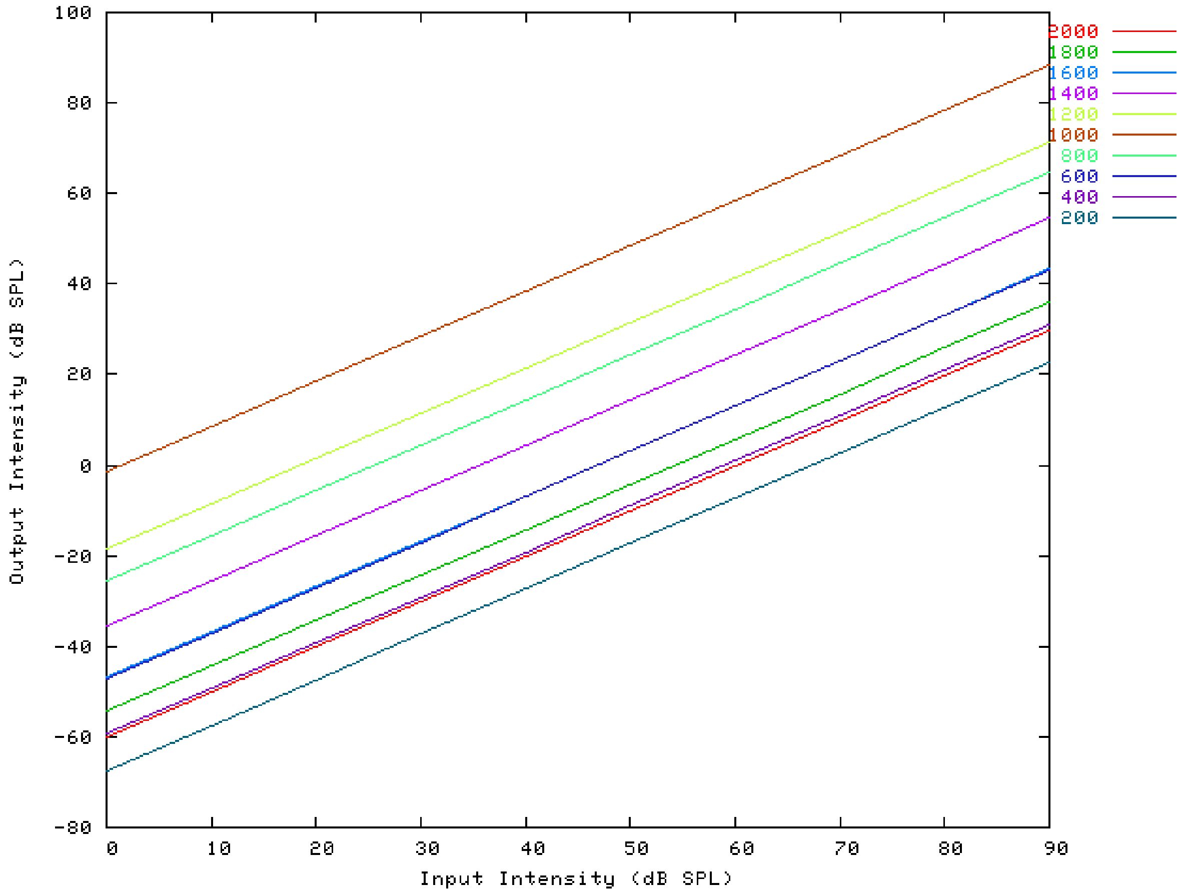
```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/GammaT4.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_GammaT       < GammaT1k.par
 Ana_Intensity    < Intensty1.par

}
```

See Figure 5.6, “The input-output response of all the BM models.The ’GammaT’ and ’Cooke’ functions show the linear response.”. Note that this filter has a linear input-output response.

Multiple I/O Functions: AutoTest/BM/GammaT5.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/GammaT5.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_GammaT       < GammaT1k.par
 Ana_Intensity    < Intensty1.par

}
```

Figure 5.34. The input-output functions for the GammaT BM model.Both on and off CF functions are shown.



I/O Ratio vs Frequency: AutoTest/BM/GammaT6.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/GammaT6.sim
```

```
begin  {

 Stim_Puretone    < PTone2.par
 Trans_Gate      < Ramp1.par
 BM_GammaT      < GammaT1k.par
 Ana_Intensity    < Intensty1.par

}
```

See Figure 5.8, “Input-output (I/O) ratio functions for all the BM models.The I/O characteristics are plotted against changing frequency.”.

Latency: AutoTest/BM/GammaT7.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/GammaT7.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_GammaT      < GammaTLog1.par
 Ana_Intensity    < Intensty1.par

}
```

See Figure 5.9, “Latency of response for the BM models.”. The latency for this model compares very well with experiment (see [16]).

Q vs Frequency: AutoTest/BM/GammaT8.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/GammaT8.sim
begin  {

 Stim_Puretone    < PTone3.par
 Trans_Gate      < Ramp1.par
 BM_GammaT      < GammaT1k.par
 Ana_Intensity    < Intensty1.par

}
```

See Figure 5.10, “The Q10 tuning characteristics for the BM models.”.

BM_Zhang: Non-linear BM (1993)

File name: MoBMZhang.[ch]

Description

This is the Zhang et al [34] non-linear basilar membrane filter model. It was revised from C code provided by the authors (the ’ARLO’ code). The centre frequency mode is controlled using the UtCFList module as described in the section called “Using the Basilar Membrane Filter Modules”.

For further information on the use of BM filterbanks see the section called “Using the Basilar Membrane Filter Modules”.

Inputs	Arbitrary signal input.
Outputs	Produces multi-channel output signal, according to the CF list parameters.
Reference	[34]

Module Parameters

Example 5.6. BM_Zhang Parameter File

MODEL	FEED_FORWARD_NL	Model type (’feed_forward_nl’).
SPECIES	HUMAN	Species type (’Human’).
MICRO_PASCALS	ON	Input expected in micro pascals instead of pascals (’on’ or ’off’).
N_ORDER	3	Order of the narrow bandpass filter (int).
GAMMA_CP	3	Order of the wide bandpass filter (int).
CORNER_CP	8	Corner parameter at which the BM starts compression
SLOPE_CP	0.22	Slope of compression nonlinearity in Boltzmann combination.
STRENGTH_CP	0.08	Strength of compression nonlinearity in Boltzmann combination.
X0_CP	5	Parameter in Boltzman function.
S0_CP	12	Parameter in Boltzman function.
X1_CP	5	Parameter in Boltzman function.
S1_CP	5	Parameter in Boltzman function.
SHIFT_CP	7	Parameter in Boltzman function.
CUT_CP	800	Cut-off frequency of control-path low-path filter (Hz).
K_CP	3	Order of control-path low-pass filter.
R0	0.05	Ratio of tau_LB (lower bound of tau_SP) to tau_narrow
#CF List Parameters:-		
DIAG_MODE	PARAMETERS	Diagnostic mode (’list’ or ’parameters’).
CF_MODE	LOG	Centre frequency mode (’single’, ’ERB’, ’ERB_n’, ’log’, ’linear’, ’focal_log’, ’user’, ’human’, ’cat’, ’chinchilla’, ’guinea-pig’ or ’macaque’).
MIN_CF	100	Minimum centre frequency (Hz).
MAX_CF	10000	Maximum centre frequency (Hz).
CHANNELS	30	No. of centre frequencies.
B_MODE	INTERNAL_DYNAMIC	Bandwidth mode (’ERB’, ’Custom_ERB’, ’Guinea_Pig’, ’user’ or ’Nonlinear’).

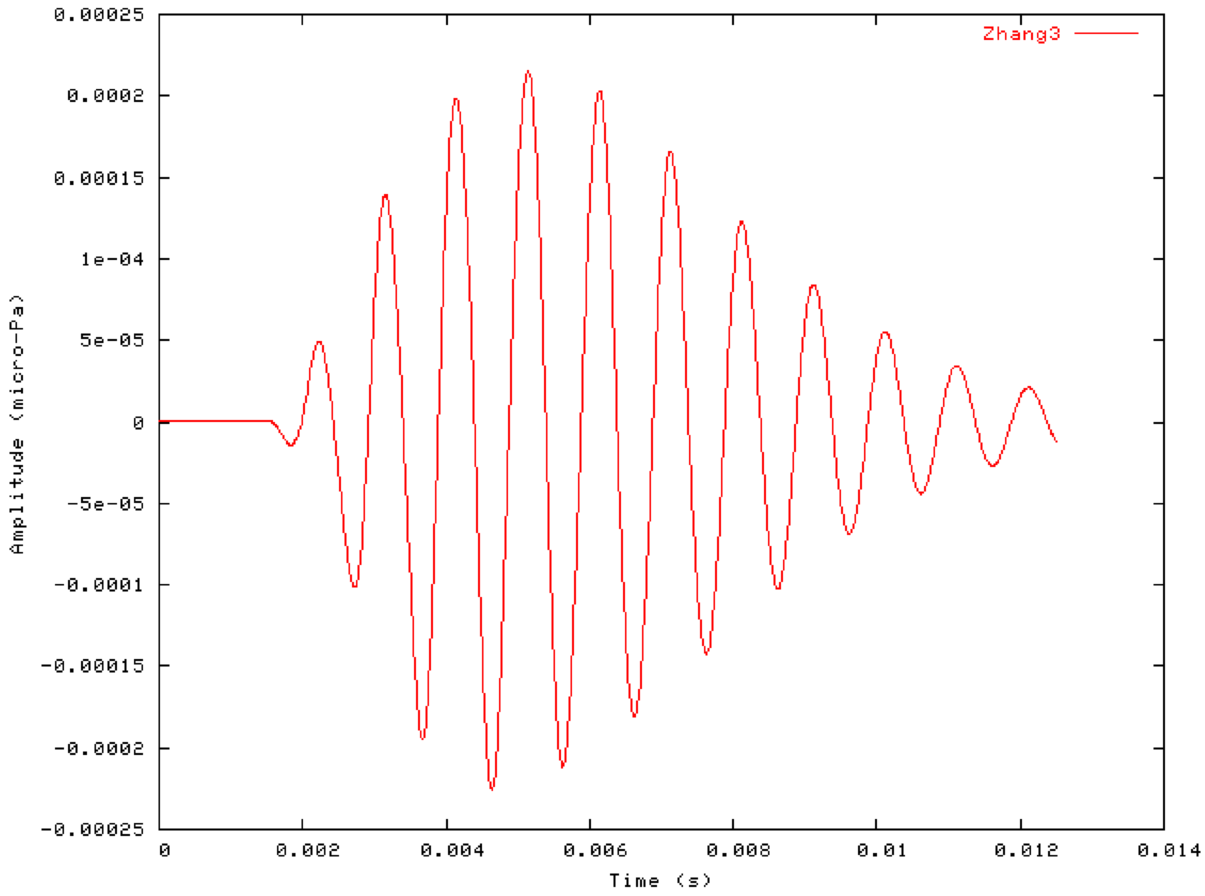
Examples Using the ’BM_Zhang’ Process Module

Some of the following tests require multiple observations using a range of stimuli and cannot be produced directly with a single run of the AMS program. These simulation scripts are run using the *AutoTest* program, which is employed to test the operation of this process module.

Impulse Response Example: AutoTest/BM/Zhang3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Zhang3.sim
begin  {

 Stim_Click      < Click1.par
 Filt_BandPass    < PEHuY1.par
 BM_Zhang       < Zhang1k.par
 Display_Signal

}
```

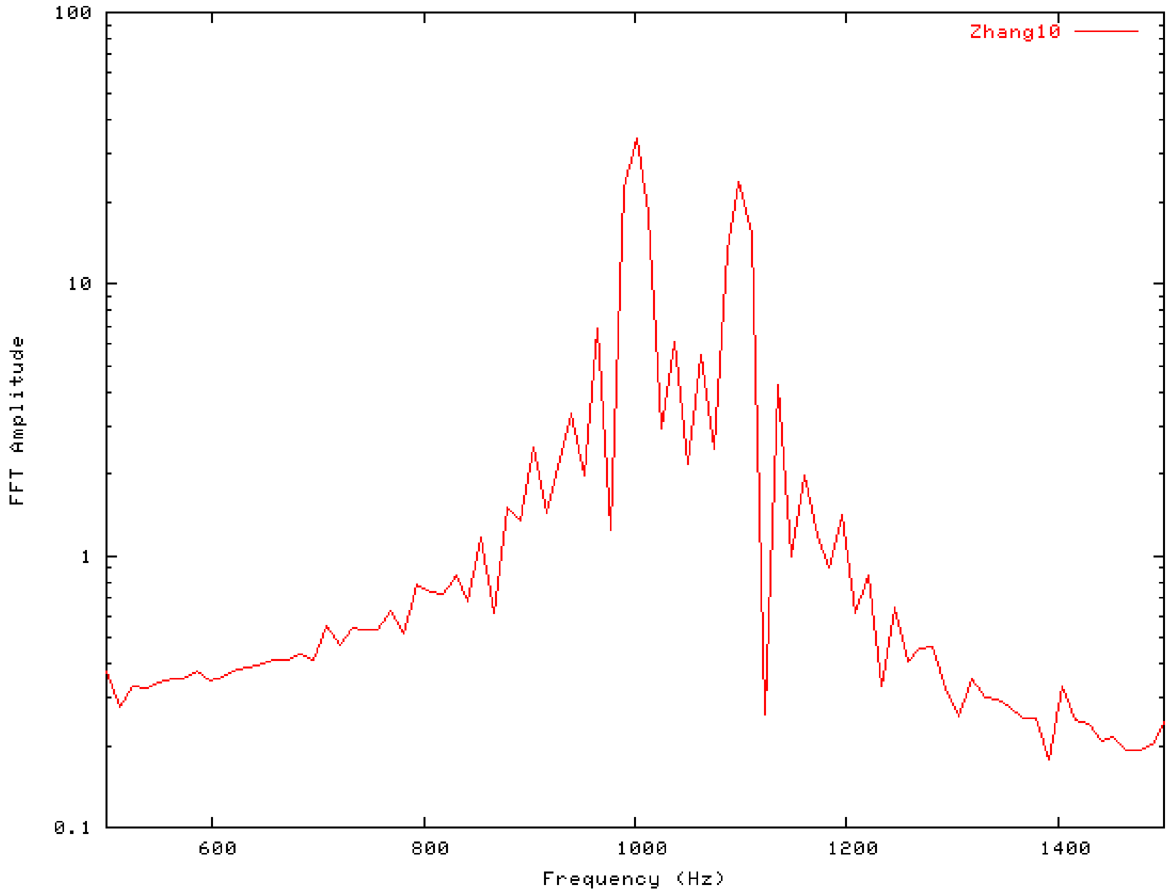
Figure 5.35. The impulse response of the Zhang BM modelCF = 1 kHz.



FT Analysis Example: AutoTest/BM/Zhang10.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Zhang10.sim
begin  {

 Stim_Puretone_Multi < MPTone2.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY1.par
 BM_Zhang        < Zhang1k.par
 Ana_FourierT     < FourierT1.par

}
```

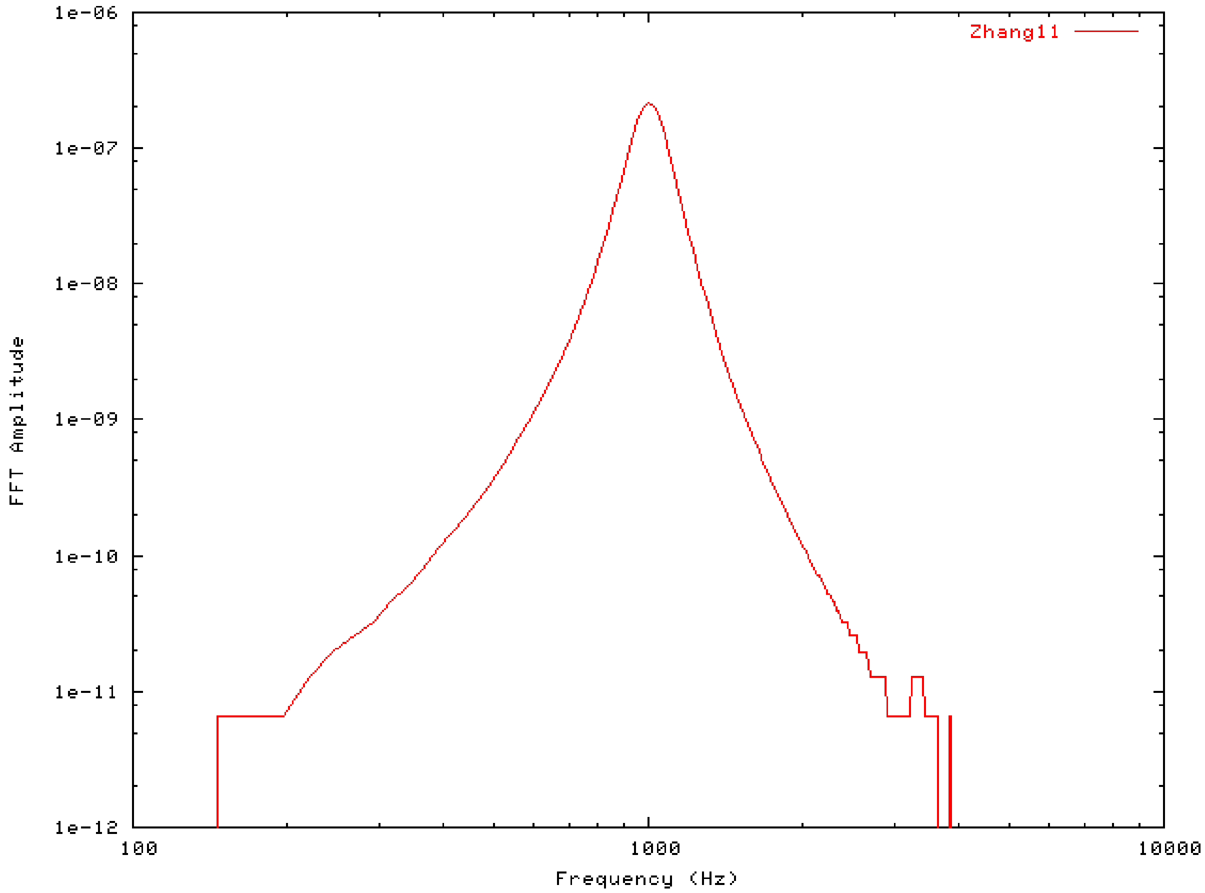
Figure 5.36. Fourier analysis of the Zhang BM model response to a two-tone stimulus.CF = 1 kHz.



Gain Response Example: AutoTest/BM/Zhang11.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.
# ./BM/Zhang11.sim
begin  {

 Stim_Click      < Click3.par
 Filt_BandPass    < PEHuY1.par
 BM_Zhang        < Zhang1k.par
 Ana_FourierT     < FourierT1.par

}
```

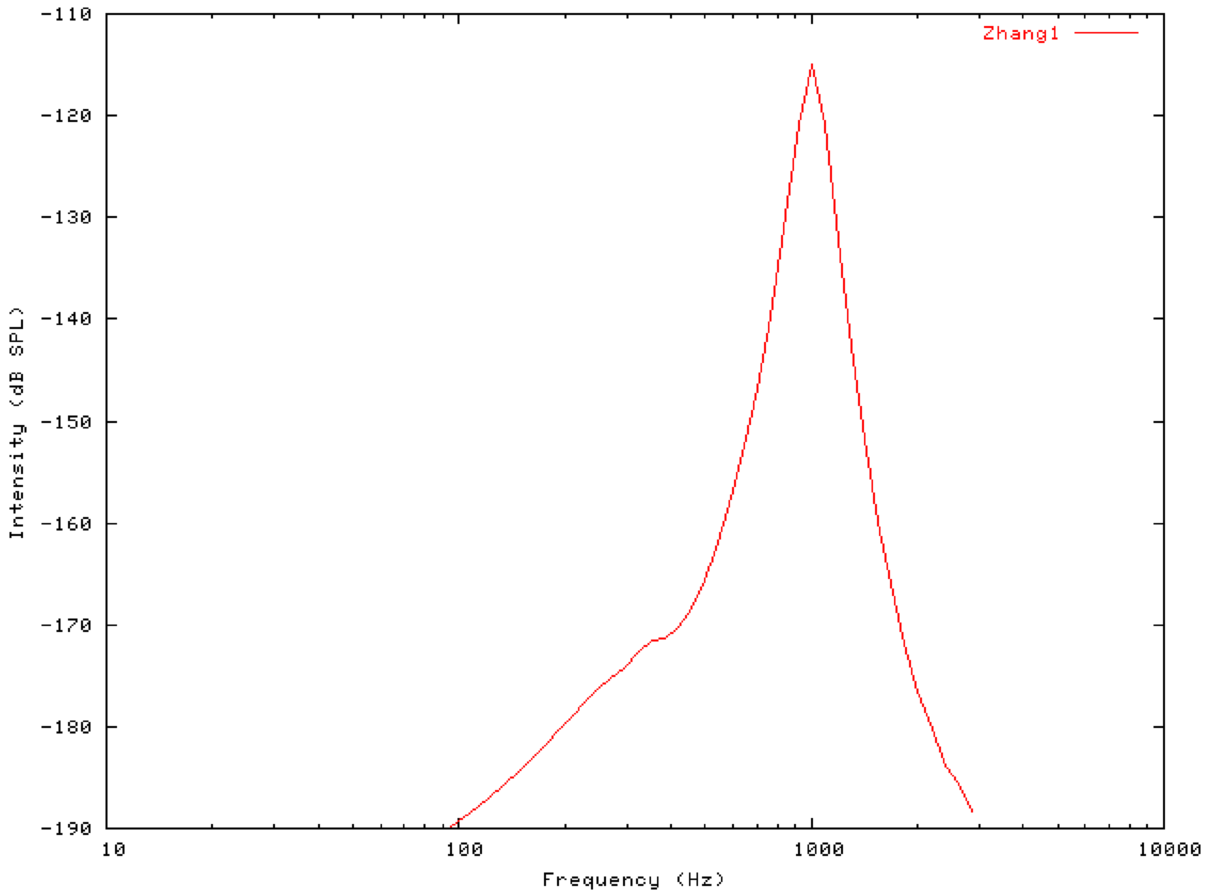
Figure 5.37. Fourier analysis of the Zhang BM model response to a single toneCF = 1 kHz.



Filter Shape: AutoTest/BM/Zhang1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Zhang1.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY1.par
 BM_Zhang        < Zhang1k.par
 Ana_Intensity    < Intensty1.par

}
```

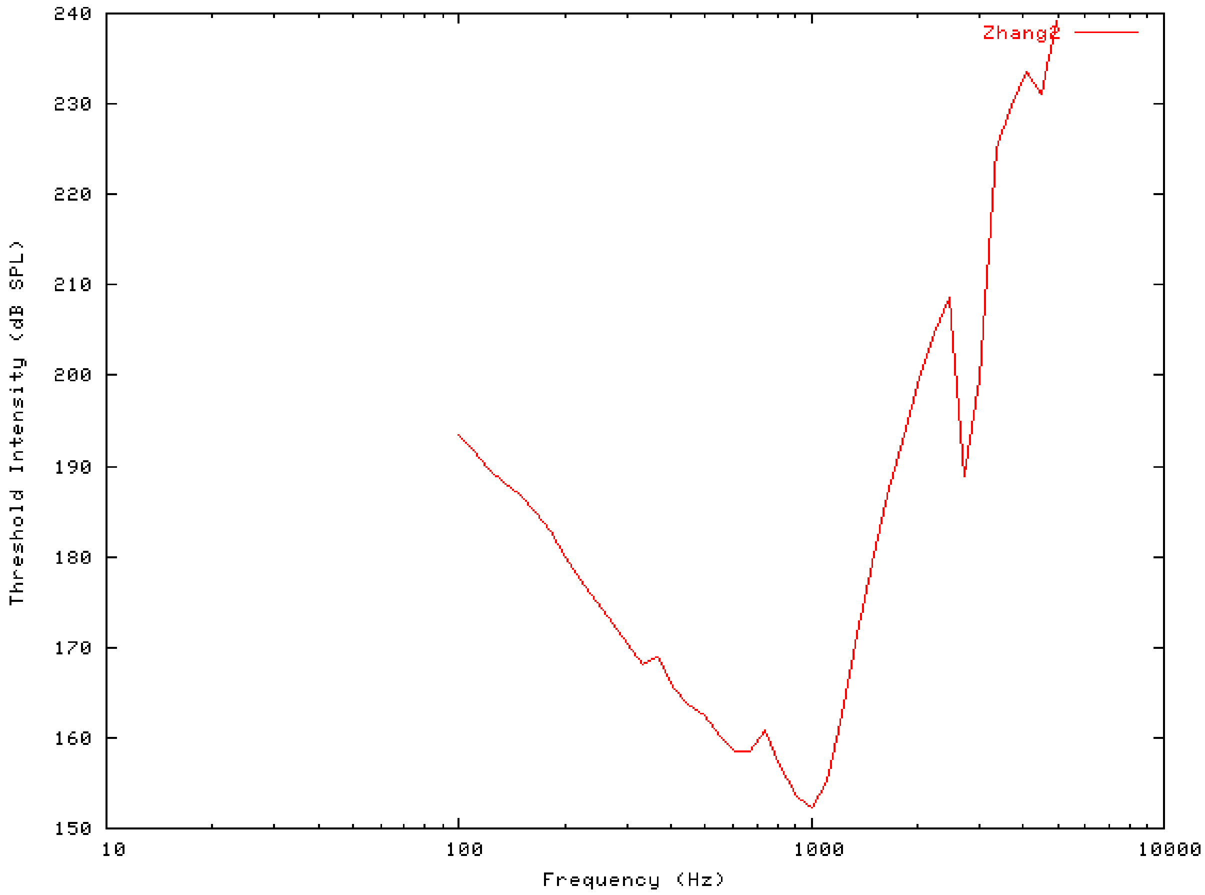
Figure 5.38. Zhang BM model filter function.CF = 1 kHz.



Tuning Curve: AutoTest/BM/Zhang2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Zhang2.sim
begin {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY1.par
 BM_Zhang        < Zhang1k.par
 Ana_Intensity    < Intensty1.par
}
```

Figure 5.39. Zhang BM model Tuning curve.CF = 1 kHz.



I/O function AutoTest/BM/Zhang4.sim

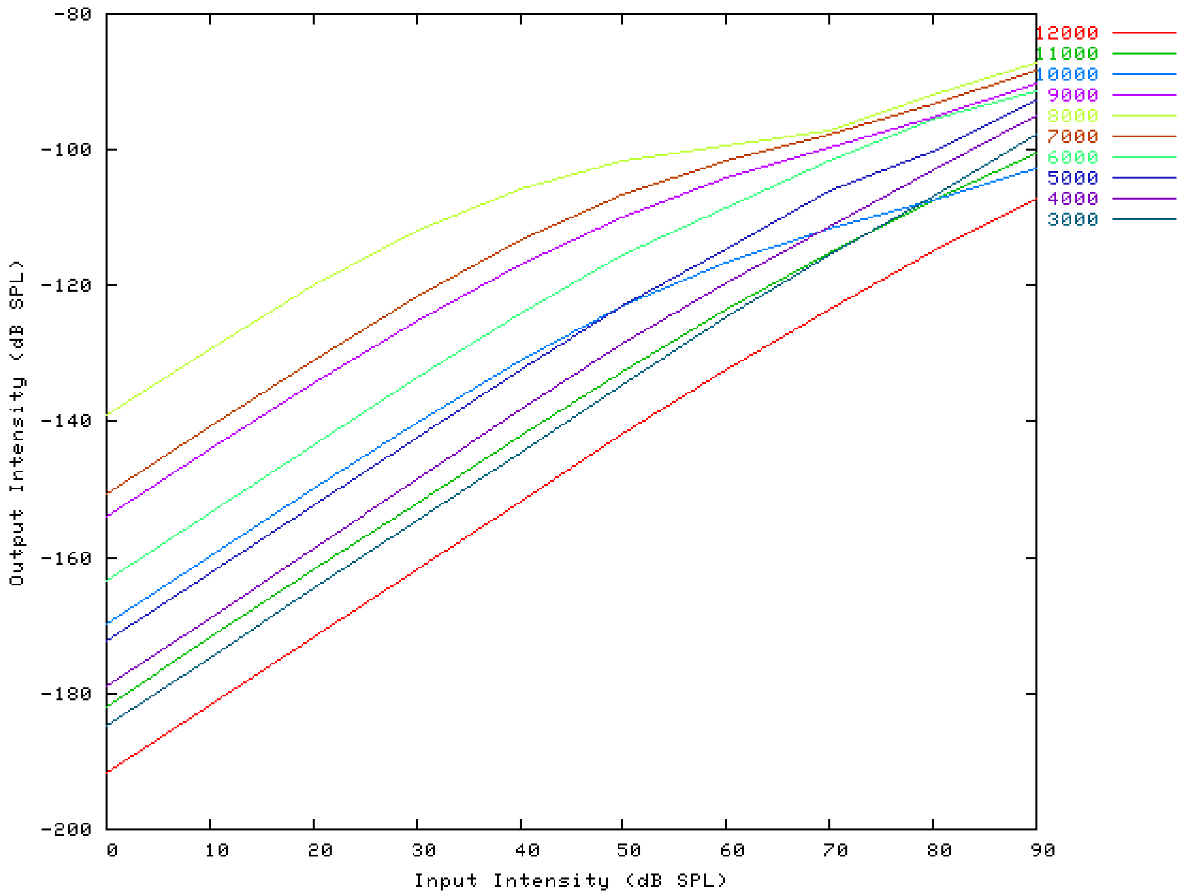
```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Zhang4.sim
begin  {

 Stim_Puretone    < PTone4.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY1.par
 BM_Zhang        < Zhang1k.par
 Ana_Intensity    < Intensty1.par

}
```

See Figure 5.6, “The input-output response of all the BM models.The ’GammaT’ and ’Cooke’ functions show the linear response.”.

Multiple I/O Functions: AutoTest/BM/Zhang5.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Zhang5.sim
begin  {

 Stim_Puretone    < PTone4.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY1.par
 BM_Zhang        < Zhang8k.par
 Ana_Intensity    < Intensty1.par

}
```

Figure 5.40. The input-output functions for the Zhang BM model.Both on and off CF functions are shown.



I/O Ratio vs Frequency: AutoTest/BM/Zhang6.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Zhang6.sim
begin  {

 Stim_Puretone    < PTone2.par
 Trans_Gate      < Ramp1.par
 BM_Zhang       < Zhang1k.par
 Ana_Intensity    < Intensty1.par

}
```

See Figure 5.8, “Input-output (I/O) ratio functions for all the BM models.The I/O characteristics are plotted against changing frequency.”.

Latency: AutoTest/BM/Zhang7.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Zhang7.sim
begin  {

 Stim_Click      < Click2.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY1.par
 BM_Zhang       < ZhangLog1.par
 Ana_FindNextIndex < FindInd1.par

}
```

See Figure 5.9, “Latency of response for the BM models.”.

Q vs Frequency: AutoTest/BM/Zhang8.sim

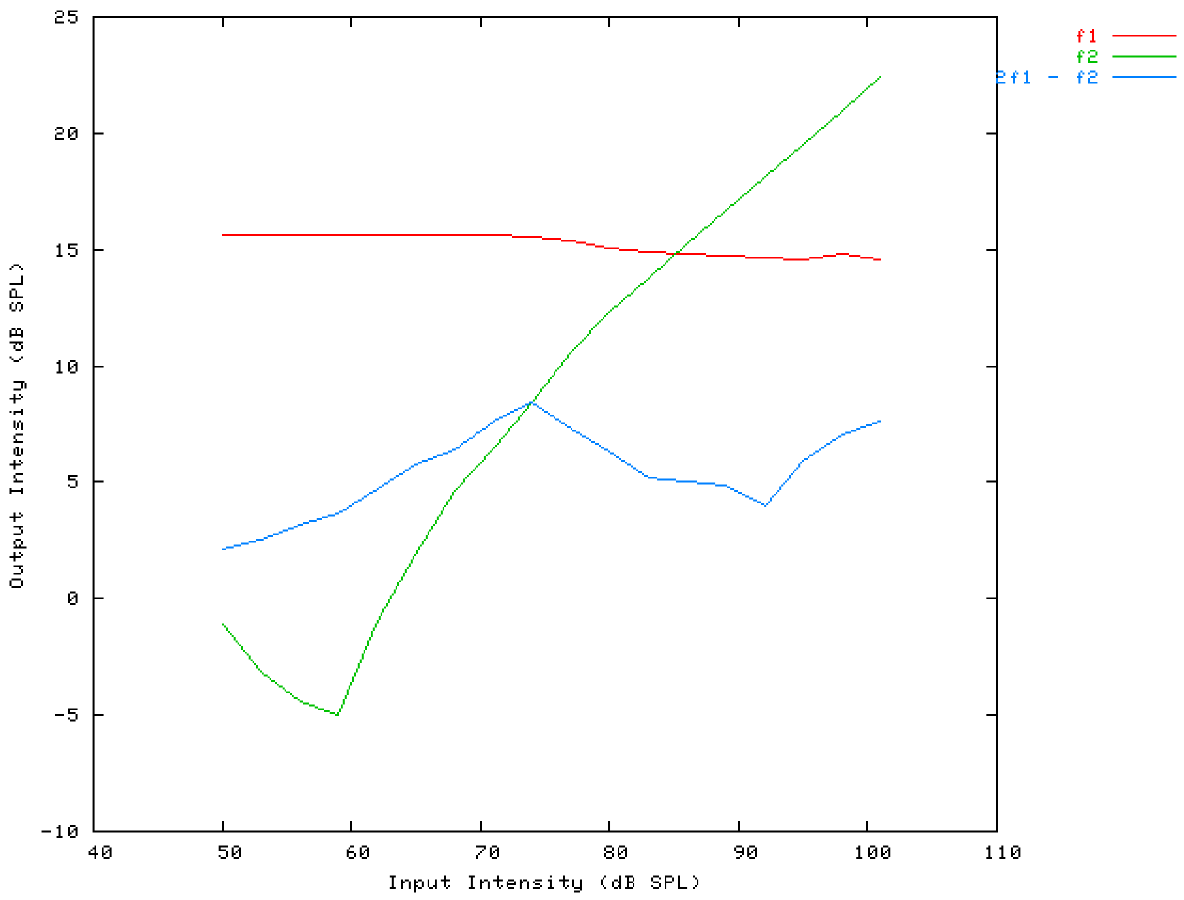
```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Zhang8.sim
begin  {

 Stim_Puretone    < PTone3.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY1.par
 BM_Zhang       < Zhang1k.par
 Ana_Intensity    < Intensty1.par
}
```

See Figure 5.10, “The Q10 tuning characteristics for the BM models.”.

Distortion Products: AutoTest/BM/Zhang9.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Zhang9.sim
begin  {

 Stim_Puretone_Multi < MPTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY1.par
 BM_Zhang       < Zhang1k.par
 Ana_FourierT    < FourierT1.par

}
```

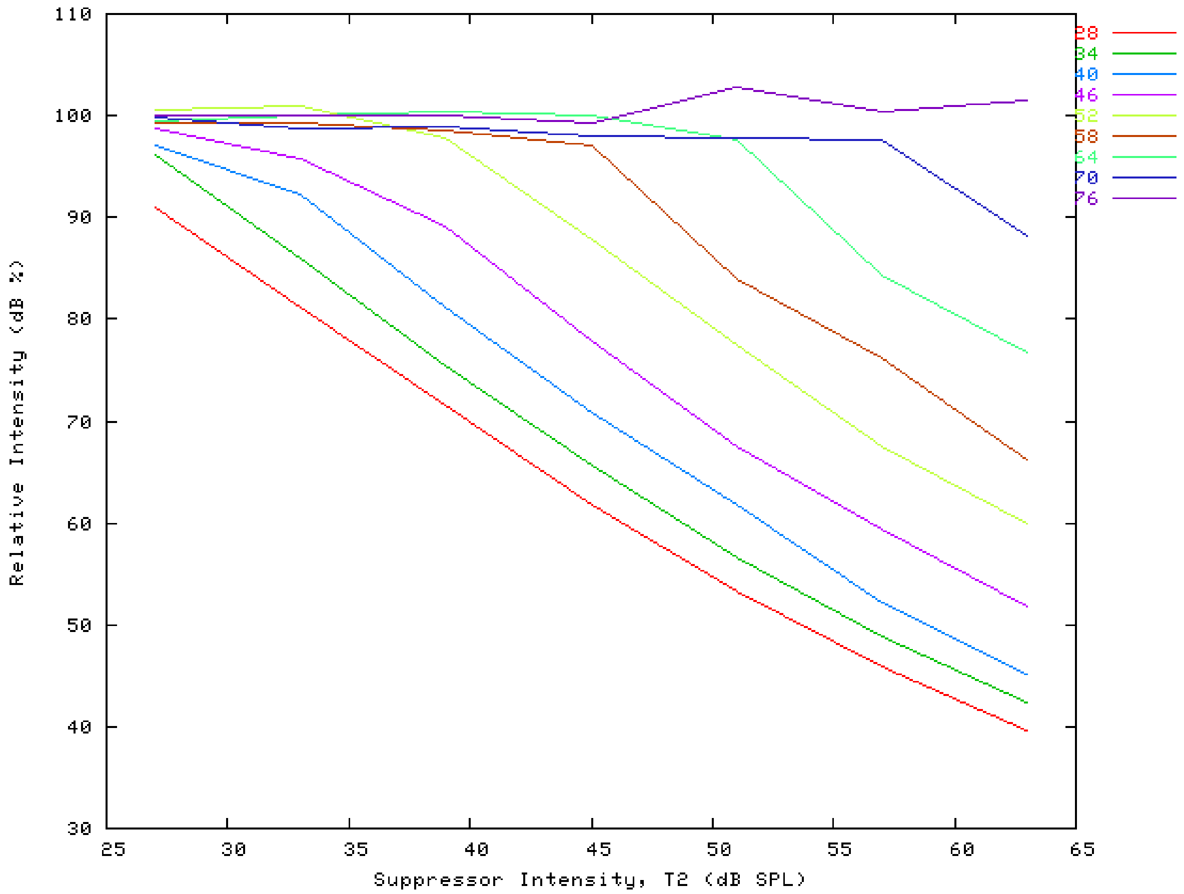
Figure 5.41. Distortion products investigation for the Zhang BM model. A similar analysis paradigm as used in [29] was employed: the f1 tone was kept at a constant level while f2 was increased in level. The outputs shown are the respective frequency component intensities as determined from a fourier analysis.. The response is similar to that of the DRNL filter, and both compare well with experimental data (see Figure 5.25, “Distortion products investigation for the DRNL BM model. A similar analysis paradigm as used in was employed: the f1 tone was kept at a constant level while f2 was increased in level. The outputs shown are the respective frequency component intensities as determined from a fourier analysis. The response is similar to that of the Carney filter, and both compare well with experimental data (see).”).



Two Tone Suppression: AutoTest/BM/Zhang12.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./BM/Zhang12.sim
begin  {

 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY1.par
 BM_Zhang        < Zhang1k.par

}
```

Figure 5.42. Two-tone suppression response for the Zhang BM model. Two stimuli are used. There is a single pure tone, and the other consisting of two pure tones: a probe tone, T1 and a suppresser tone, T2. The relative suppression is then calculated by dividing the unsuppressed model FFT modulus by the suppressed output modulus and multiplying by 100. This is calculated from the Fourier component at the probe frequency.


110
100
90
80
70
60
50
40
30
Relative Intensity (dB %)
25
30
35
40
45
50
55
60
65
Suppressor Intensity, T2 (dB SPL)
28
34
40
46
52
58
64
70
76

Chapter 6. Display Modules

Display_Signal: Standard Signal Display

File name: GrSignalDisp.[cpp,h]

Description

This module allows the signal data held by a *process* to be displayed. Programs may always include this module, however the signal display will only be produced if the program has been compiled using the DSAM library with *Graphics Support* included. The display module does not change the input signal passed to it. This means that, the display module is ’transparent’.

Inputs	Arbitrary signal input.
Outputs	Passes the input signal on to the next process unchanged.
Reference	

Module Parameters

Example 6.1. Display_Signal Parameter File

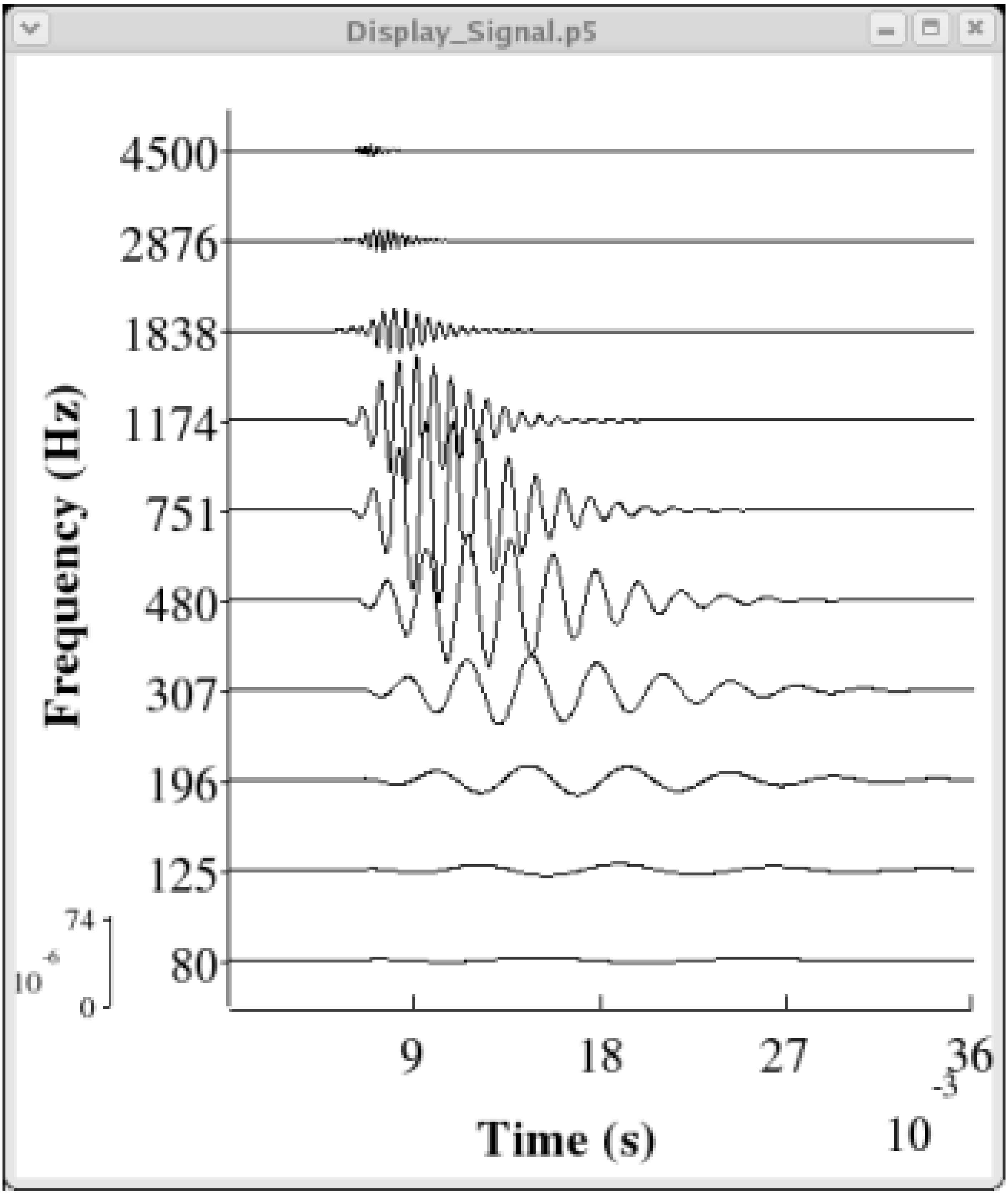
```
MAGNIFICATION     1        Signal magnification.
NORMALISATION     MIDDLE   Y normalisation mode (’bottom’ or ’middle’).
CHANNEL_STEP      1        Channel stepping mode.
NUMGREYSCALES     10       Number of grey scales.
X_RESOLUTION      0.01     Resolution of X scale (1 - low, fractions are higher).
WIDTH             -1       Displayed signal width (seconds or x units).
Y_AXIS_TITLE      ""       Y-axis title.
Y_AXIS_MODE       AUTO     Y-axis mode (’channel’ (No.) or ’scale’).
AUTO_SCALING      ON       Automatic scaling (’on’ or ’off’).
AUTO_Y_SCALE      ON       Automatic y-axis scale (’on’ or ’off’).
MAXY              0        Maximum Y value (for manual scaling).
MINY              0        Minimum Y Value (for manual scaling).
Y_NUMBER_FORMAT   "y"      Y axis scale number format, (e.g. y.yye-3).
Y_DEC_PLACES      0        Y axis scale decimal places.
Y_TICKS           15       Y axis tick marks.
Y_INSET_SCALE     ON       Y inset scale mode (’on’ or ’off’).
X_AXIS_TITLE      ""       X axis title.
AUTO_X_SCALE      ON       Autoscale option for x-axis (’on’ or ’off’)
X_NUMBER_FORMAT   "xe-3"   X axis scale number format, (e.g. x.xxe-3).
X_DEC_PLACES      0        X axis scale decimal places.
X_TICKS           6        X axis tick marks.
X_OFFSET          0        X offset for display in zoom mode (x units).
X_EXTENT          -1       X extent for display in zoom mode (x units or -1 for end of signal).
WIN_TITLE         ""       Display window title.
MODE              LINE     Display mode (’off’, ’line’ or ’gray_scale’).
SUMMARYDISPLAY    OFF      Summary display mode (’on’ or ’off’).
FRAMEDELAY        0        Delay between display frames (s)
TOPMARGIN         5        Top margin for display (percent of display height).
WIN_HEIGHT        500      Display frame height (pixel units).
WIN_WIDTH         440      Display frame width (pixel units).
WIN_X_POS         1        Display frame X position (pixel units).
WIN_Y_POS         16       Display frame Y position (pixel units).
```

Examples Using the ’Display_Signal’ Process Module

Example: AutoTest/Di/Meddis86.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Di/Meddis86.sim
begin  {

 Stim_PureTone_2    < P2Tone1.par
 Trans_Gate     < Ramp1.par
 Filt_BandPass   < PreEmph1.par
 BM_GammaT      < GammaTLog1.par
 IHC_Meddis86    < Meddis86.par
 Display_Signal < Disp1.par

}
```

Figure 6.1. Signal Display WindowOutput from the Hair Cell process in the Simulation is shown.


Display_Signal.p5
4500
2876
1838
1174
751
480
307
196
125
80
Frequency (Hz)
74
10 -6
0
9
18
27
36
-3
10
Time (s)

Chapter 7. Filter Process Modules

Filt_BandPass: Band pass filter process

File name: FlBandPass.[ch]

Description

This is a band pass filter process, which is typically used as the outer-/middle-ear filter model but can be applied generally. It uses a first order band pass filter, cascaded **order** times.

Inputs	Arbitrary signal input.
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	[3]

Module Parameters

Example 7.1. Filt_Bandpass Parameter File

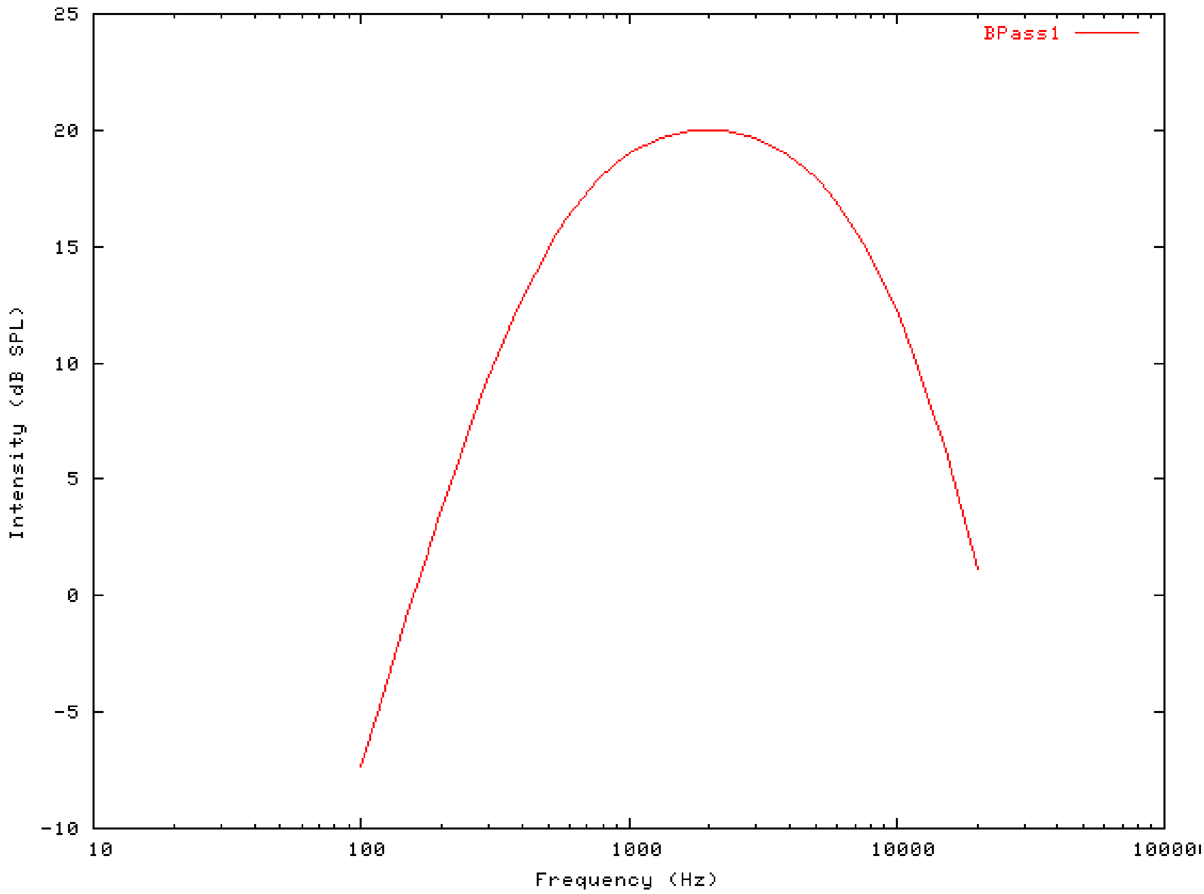
CASCADE	2	Filter cascade.
GAIN	0	Pre-attenuation for filter (dB).
LOWER_FREQ	450	Lower, 3 dB down cut-off frequency (Hz).
UPPER_FREQ	8500	Upper, 3 dB down cut-off frequency (Hz).

Examples Using the ’Filt_Bandpass’ Process Module

Example: AutoTest/Fl/BPass1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Fl/BPass1.sim
begin {

 Stim_PureTone   < PTone1.par
 Trans_Gate     < Ramp1.par
 Filt_BandPass   < PreEmph1.par
 Ana_Intensity   < Intensty1.par

}
```

Figure 7.1. Filter shape for the Bandpass FilterOften used as outer-/middle-ear pre-emphasis model.



Filt_FIR: Finite Impulse Response filter.

File name: FlFIR.[ch]

Description

This routine process is an FIR filter. For each channel of the signal, the filtered output samples, y are produced from the input samples, x using the FIR equation

$$y_n = \sum_{k=0}^{M} c_k x_{n-k}$$

- where C_k are the filter coefficients (the impulse response function). This module is under development, and at present only works in 'user' mode when the FIR coefficients are supplied in the parameter file.

Inputs	In user mode the input sampling interval must correspond to the sampling rate used to produce the filter coefficients.
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	

Module Parameters

Example 7.2. Filt_FIR Parameter File

```
DIAG_MODE  OFF   Diagnostic mode ('off', 'screen' or <file name>).
TYPE       USER  FIR filter type 'bandpass', 'differentiator' or 'Hilber').
NUM_TAPS   0     Number of filter tapsl
NUM_BANDS  0     No. of frequency bands.
```

Table 7.1. Filt_FIR: Diagnostic mode options

Diagnostic Mode	Description
off	No diagnostics are output.;
screen	Diagnostics from the module are output to the screen.
error	Diagnostics are output to the standard error output stream (stderr on Unix)
<file name>	Diagnostics are output to the specified file.

Table 7.2. Filt_FIR: Type mode options

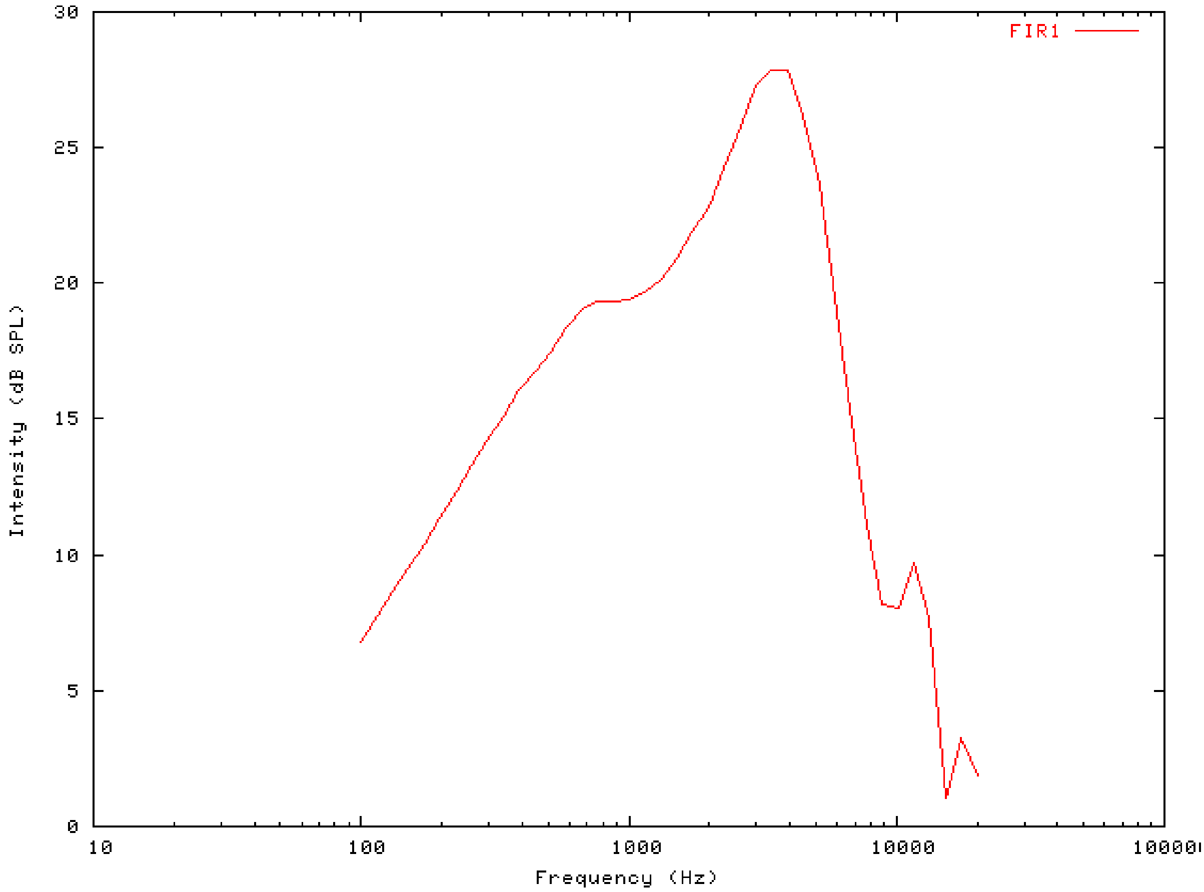
Type	Description
user	Each of the **freq_resp** array entries is the corresponding filter coefficient. The **num_bands** parameter should be set to the number of filter coefficients..

Examples Using the 'Filt_FIR' Process Module

Example: AutoTest/Fl/FIR1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Fl/FIR1.sim
begin  {

 Stim_PureTone    < PTone3.par
 Trans_Gate      < Ramp1.par
 Filt_FIR       < FIRff_50k.par
 Ana_Intensity    < Intensty1.par

}
```

Figure 7.2. Filter shape for the FIR process module.



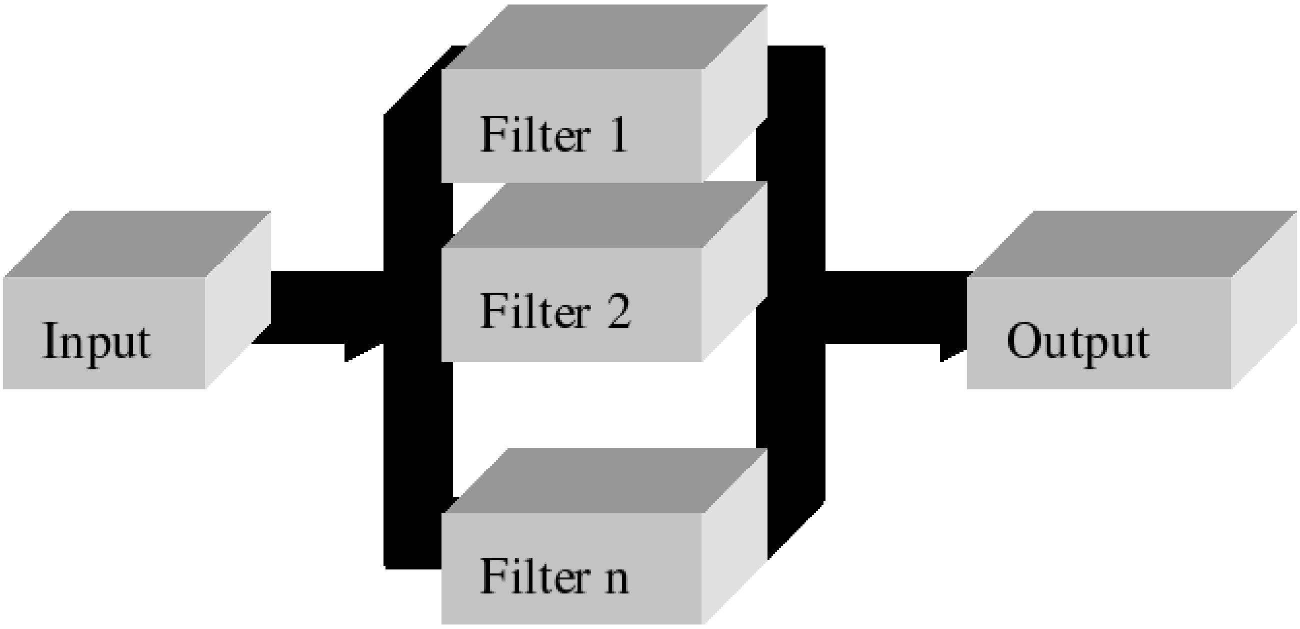
Filt_MultiBPass: Multiple band pass filter.

File name: FlMultiBPass.[ch]

Description

This is the multi-filter band pass process, which is often used as an outer-/middle-ear filter model but may have applications elsewhere. It uses a specified number of first order band pass filter, each capable of being cascaded **order** times, to model the effects of this section of the auditory periphery. The filters operate in parallel:

Figure 7.3. Multi-Bandpass filter



Inputs	Arbitrary signal input.
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	

Module Parameters

Example 7.3. Filt_MultiBPass Parameter File

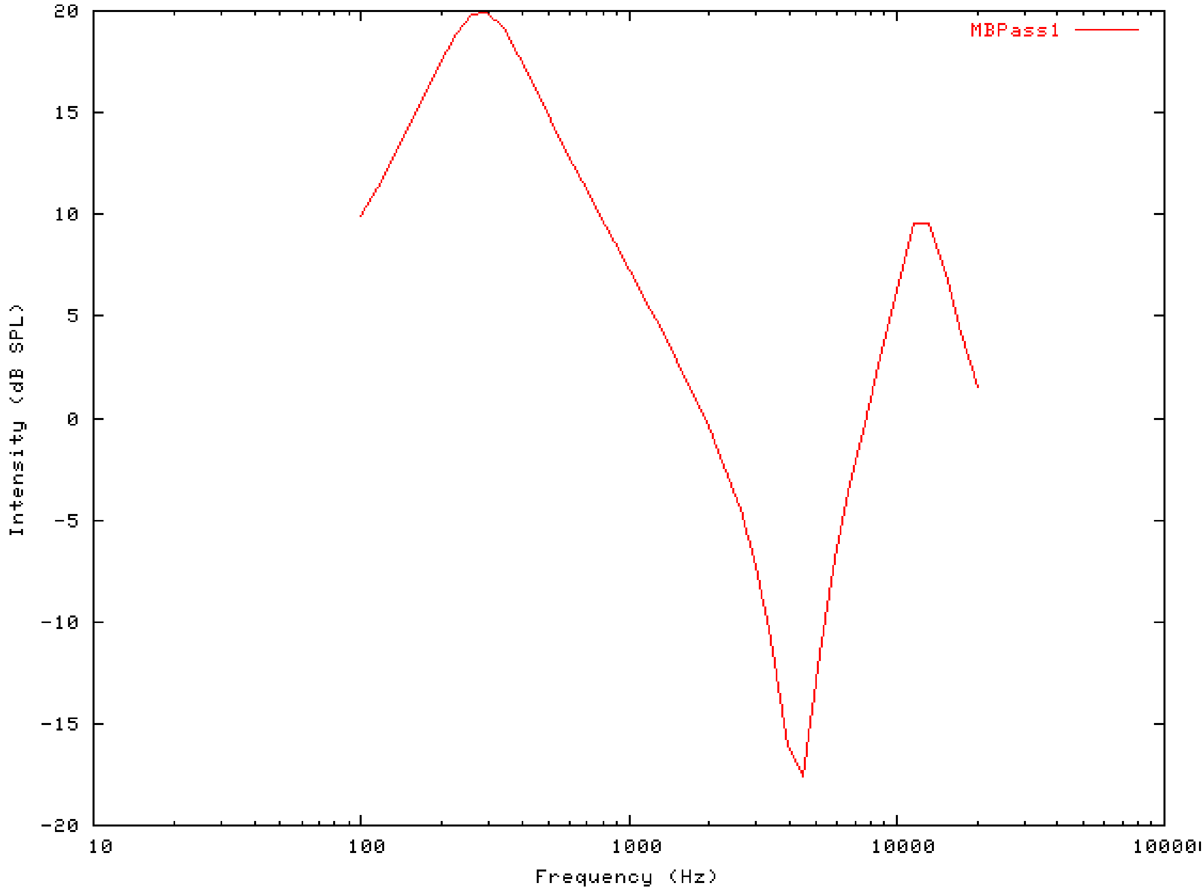
```
NUM_FILTERS    3        No. of parallel band pass filters.
CASCADE        0:2
CASCADE        1:2
CASCADE        2:2
GAIN           0:1.5
GAIN           1:6
GAIN           2:-11
LOWER_FREQ     0:330
LOWER_FREQ     1:1900
LOWER_FREQ     2:7500
UPPER_FREQ     0:5500
UPPER_FREQ     1:5000
UPPER_FREQ     2:14000
```

Examples Using the ’Filt_MultiBPass’ Process Module

Example: AutoTest/Fl/MBPass1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.
# ./Fl/MBPass1.sim
begin  {

 Stim_PureTone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_MultiBPass  < MBPassPE1.par
 Ana_Intensity    < Intensty1.par

}
```

Figure 7.4. Filter shape for the multi-bandpass filterOften used as outer-/middle-ear pre-emphasis model.



Filt_LowPass: Low pass filter process

File name: FlLowPass.[ch]

Description

This simple low-pass is filter is used as a dendritic filter model but also has general applications. In the dendritic filter model it converts summed spike events into a post-synaptic potential.

Inputs	Arbitrary signal input.
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	

Module Parameters

Example 7.4. Filt_LowPass Parameter File

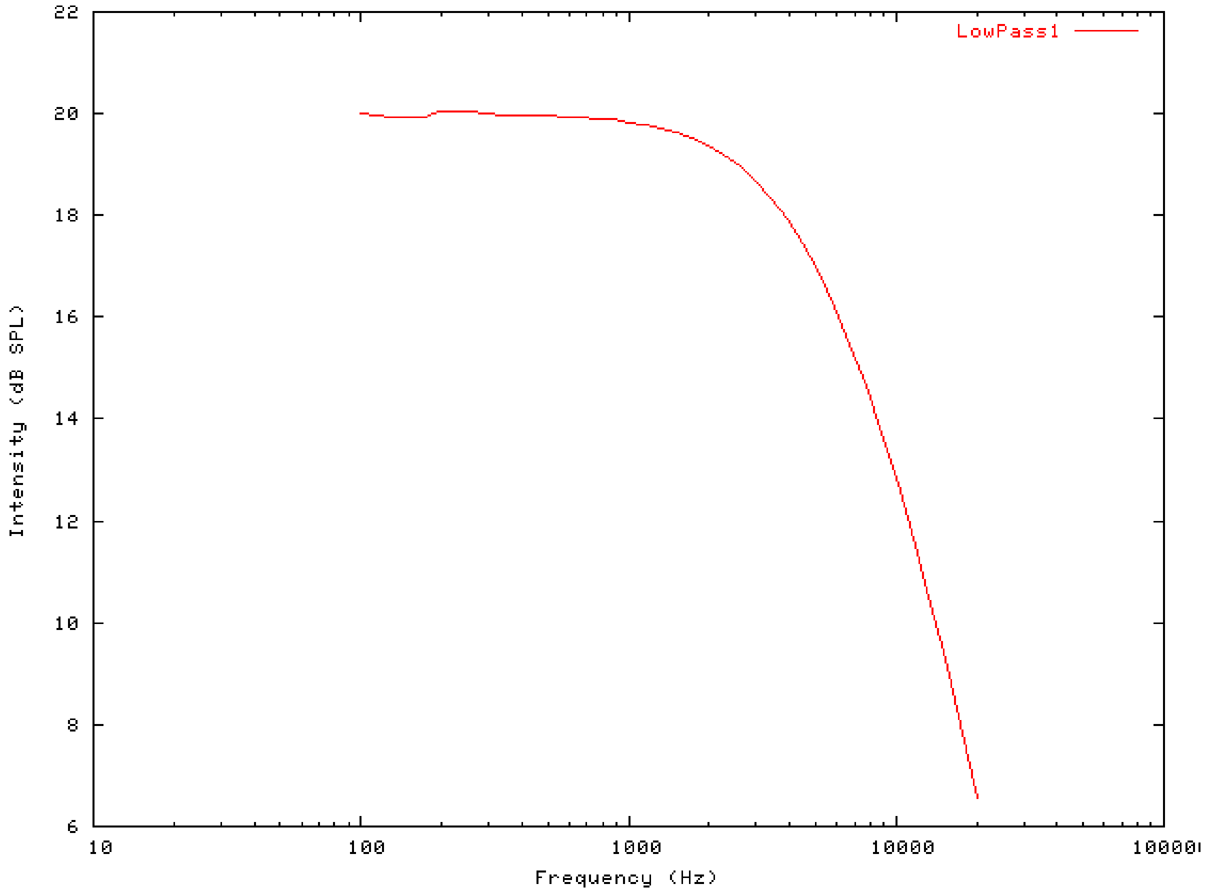
MODE	NORMAL	Output mode for model ('normal' or 'scaled').
CUT_OFF_FREQ	5000	Cut-off frequency (3 dB down point - Hz).
SCALE	1	Signal multiplier scale ('scaled' mode only).

Examples Using the 'Filt_LowPass' Process Module

Example: AutoTest/Fl/LowPass1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Fl/LowPass1.sim
begin {

 Stim_PureTone   < PTone2.par
 Trans_Gate     < Ramp1.par
 Filt_LowPass    < Dendrite1.par
 Ana_Intensity   < Intensty1.par

}
```

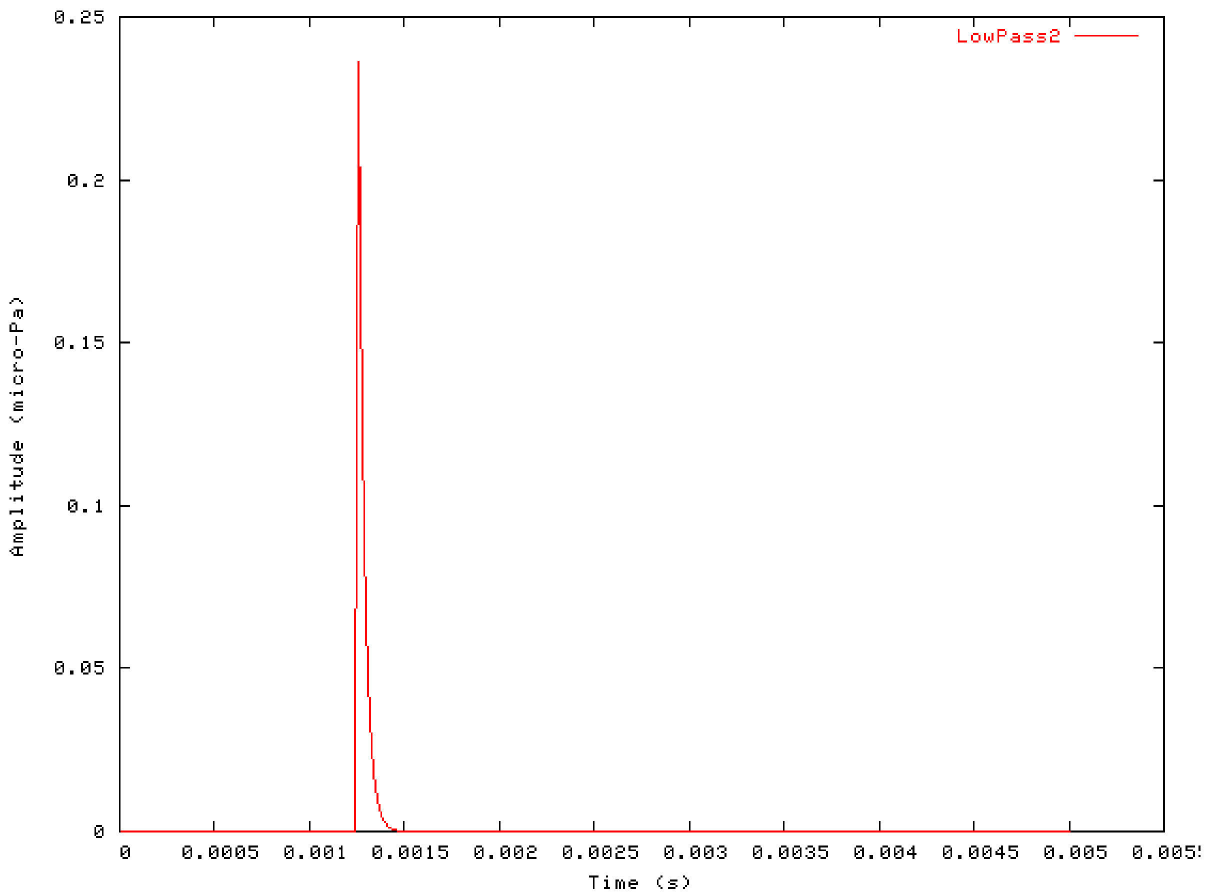
Figure 7.5. Filter shape for the low-pass filterThis process module is often used as a dendritic filtering model.



Example: AutoTest/Fl/LowPass2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Fl/LowPass2.sim
begin  {

 Stim_Click      < Click1.par
 Filt_LowPass     < Dendrite1.par

}
```

Figure 7.6. Response of the low pass filter process click stimulusThis process module is often used as a dendritic filtering model.


0.25
0.2
0.15
0.1
0.05
0
Amplitude (micro-Pa)
LowPass2
0
0.0005
0.001
0.0015
0.002
0.0025
0.003
0.0035
0.004
0.0045
0.005
0.0055
Time (s)

Chapter 8. Input/Output Modules

DataFile_In/DataFile_Out: Sound File Format Support

File name: FiDataFile.[ch]

Description

The *DataFile_In* and *DataFile_Out* process modules provide the sound file format support. For all intents and purposes these are two processes, however, the same code module represents them both. This means that both modules use the same parameter sets, according to the file format being used. Table 8.1, “Supported File Formats”, shows the file formats that can be read or written respectively using these modules.

Table 8.1. Supported File Formats

Extension	Description
’.aiff’ or ’.aif’	AIFF (audio interchange file format),
’.wav’	Microsoft WAV format
’.dat’	Multi-column ASCII (text) files (without a time column)
’.raw’	Raw binary file.

If the main portion of the file name is given as a minus sign, e.g. ’-.aif’, the *DataFile_In/Out* module will read or write to *stdin* or *stdout* . This allows data to be passed using UNIX pipes (e.g. ’dsam_program1 | other_program’).

The AIFF standard has no provision to save the magnitude of a signal, however, ’. AIFF’ files created using this module include a custom AIFF *chunk* or header that records the signal magnitude. This means that such a file will have the correct magnitude set when read in by this module.

Some parameters are technical in nature. For example, *endian_mode* is specific to your type of computer. Use the default values whenever possible.

Several parameters are ignored when their values are given in the data file itself. For example, *samplerate* is specified within the ’AIFF’ and ’WAV’ file formats, and is therefore not controlled by the module.

The *Gain* parameter is important when the gain is not specified within the respective file format. For example, data in ’.wav’ files are normalised to lie between -1 and +1. The user must therefore supply a gain parameter in (dB) to restore the input signal to a level in micro-Pascals. To do this, it is normally necessary to know the normalisation factor used when creating the ’.wav’ file.

Inputs	Arbitrary single process: For DataFile_Out, none for DataFile_In.
Outputs	For DataFile_In, an output signal is produced, dependent upon the input sound file and the process parameters.
Reference	

Module Parameters

Example 8.1. DataFile_In/Out Parameter File

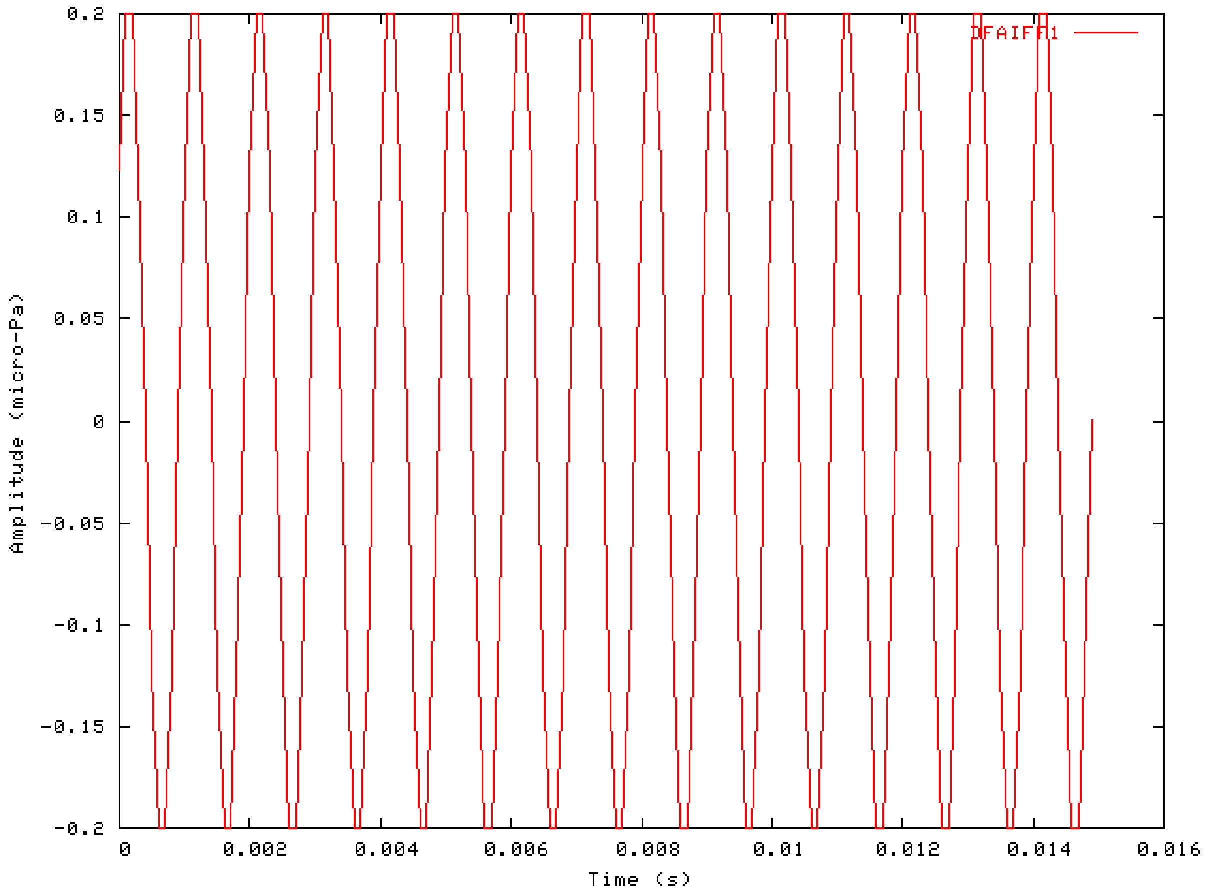
FILENAME	output.dat	Stimulus file name.
WORDSIZE	2	Default word size for sound data (1,2 or 4 bytes)
ENDIAN_MODE	DEFAULT	Default (for format), ’little’ or ’big’-endian.
CHANNELS	1	No. of Channels: for raw binary and ASCII files.)
NORM_MODE	-1	Normalisation factor for writing (either 0 or -ve: automatic).
SAMPLERATE	8000	Default sample rate: for raw binary and ASCII files (Hz).
DURATION	-1	Duration of read signal: -ve = unlimited (s)
STARTTIME	0	Start time (offset) for signal (s).
GAIN	0	Relative signal gain (dB).

Examples Using the ’DataFilt_In/Out’ Process Module

AIFF Read Example: AutoTest/IO/DFAIFF1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IO/DFAIFF1.sim
begin  {

 DataFile_In     < DFAIFF1.par
 Display_Signal

}
```

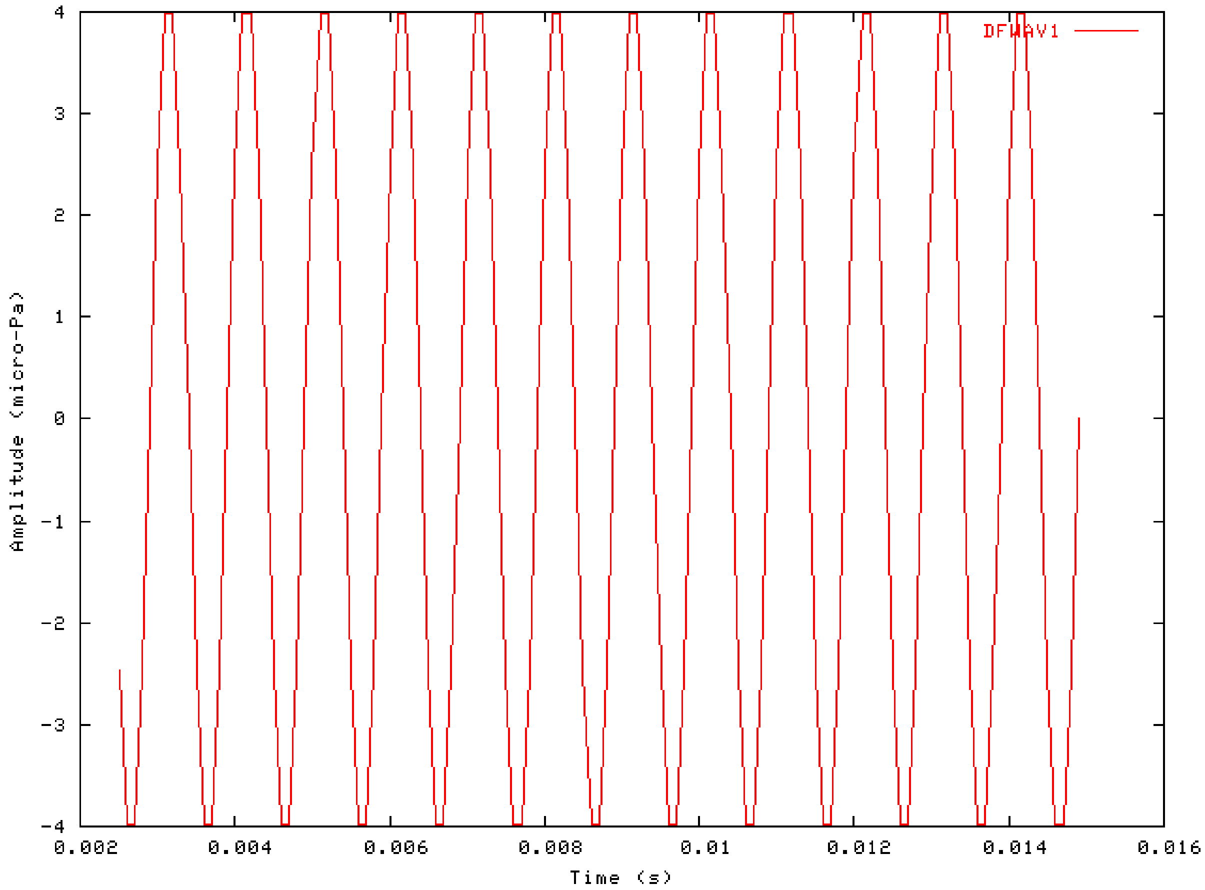
Figure 8.1. AIFF stimulus read by the DataFile module



Microsoft WAVE Read Example: AutoTest/IO/DFWAV1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IO/DFWAV1.sim
begin  {

 DataFile_In     < DFWAV1.par
 Display_Signal

}
```

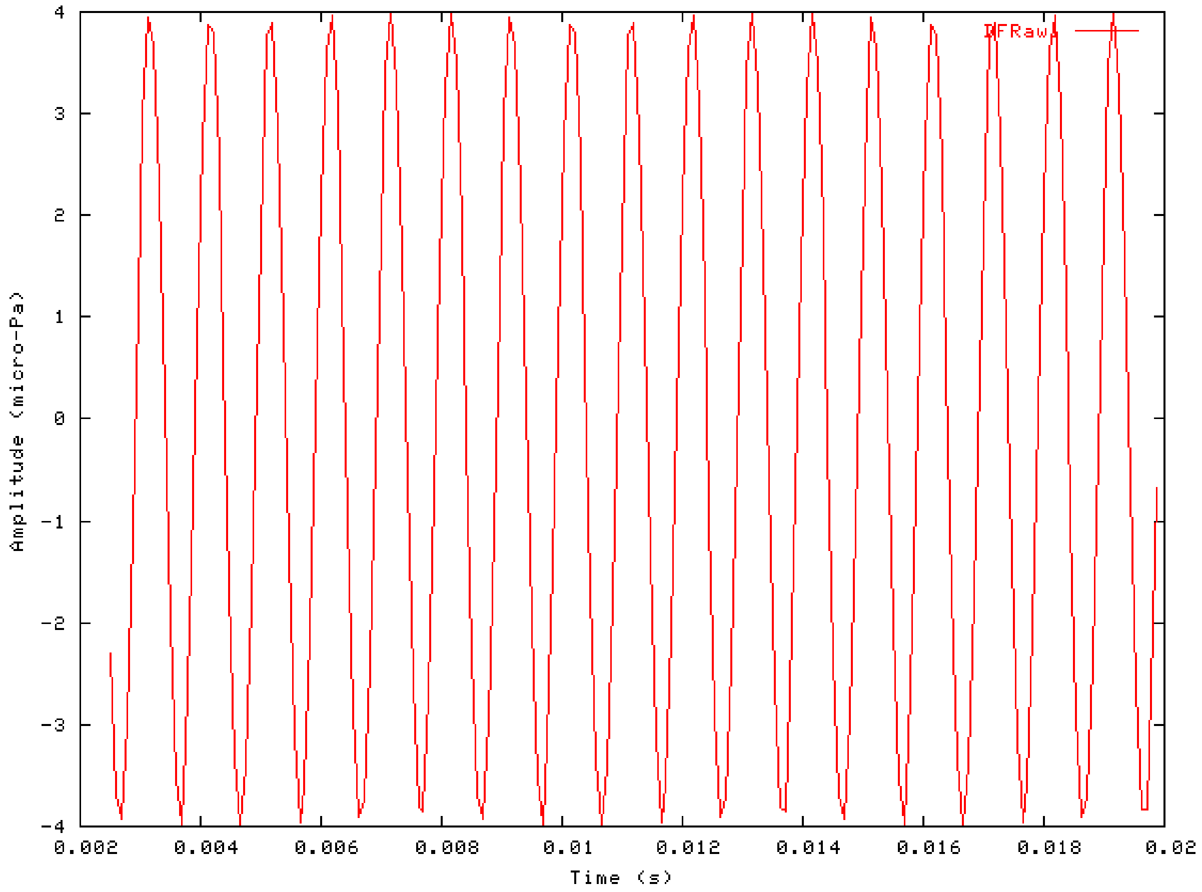
Figure 8.2. Microsoft WAVE stimulus read by the DataFile module



Raw binary Read Example: AutoTest/IO/DFRaw1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IO/DFRaw1.sim
begin  {

 DataFile_In      < DFRaw1.par
 Display_Signal

}
```

Figure 8.3. Raw binary stimulus read by the DataFile module


IDFRaw1
Amplitude (micro-Pa)
4
3
2
1
0
-1
-2
-3
-4
0.002
0.004
0.006
0.008
0.01
0.012
0.014
0.016
0.018
0.02
Time (s)

Chapter 9. Transduction Model Modules

IHC_Carney: Inner Hair Cell Synapse (1993)

File name: MoIHCCarney.[ch]

Description

Laurel H. Carney inner hair cell (IHC) synapse module ([5]). It requires an IHC receptor potential module as input (e.g. see IHCRP_Carney It outputs spike probabilities.

Inputs	Arbitrary single input, however, it expects input from an inner hair cell receptor potential process.
Outputs	Produces an output signal of spike probabilities, with the same dimensions as the input signal.
Reference	[5]

Module Parameters

Example 9.1. IHC_Carney Parameter File

```
V_MAX    10        Maximum depolarising hair cell voltage, Vmax (V).
R0       70        Resting release from synapse, R0 (spikes/s).
P_REST   0.015     Resting permeability, Prest (’volume’/s).
PG_MAX   0.08      Maximum global permeability, PGmax (’volume’/s).
PL_MAX   0.1       Maximum local permeability, PLmax (’volume’/s).
PI_MAX   1.5       Maximum immediate permeability, PImax (’volume’/s).
VL_MAX   0.003     Maximum local volume, VLmax (’volume’).
VL_MIN   0.001     Minimum local volume, VLmin (’volume’).
VI_MAX   0.0003    Maximum immediate volume, VImax (’volume’).
VI_MIN   0.0001    Minimum immediate volume, VImin (’volume’).
```

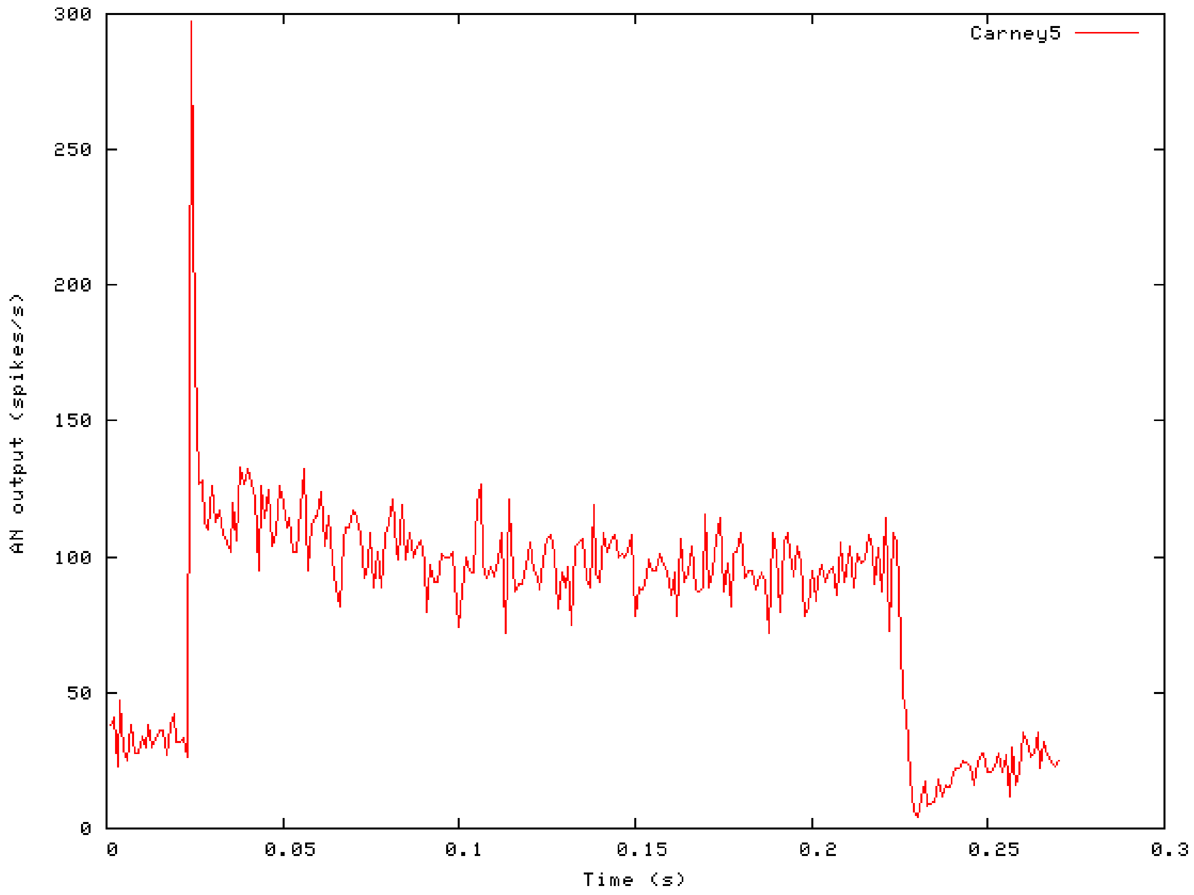
Examples Using the ’IHC_Carney’ Process Module

Some of the following tests require multiple observations using a range of stimuli and cannot be produced directly with a single run of the *AMS* program. These simulation scripts are run using the *AutoTest* program, which is employed to test the operation of the DSAM process modules.

Simple Response Example: AutoTest/IHC/Carney5.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Carney5.sim
begin  {

 Stim_Puretone_2  < P2Tone1.par
 Trans_Gate      < Ramp1.par
 BM_Carney      < Carney1k.par
 IHCRP_Carney     < CarneyRP1.par
 IHC_Carney      < CarneyIHC1.par
 AN_SG_Binomial   < BinomSG1.par
 Ana_Histogram    < PSTH3.par

}
```

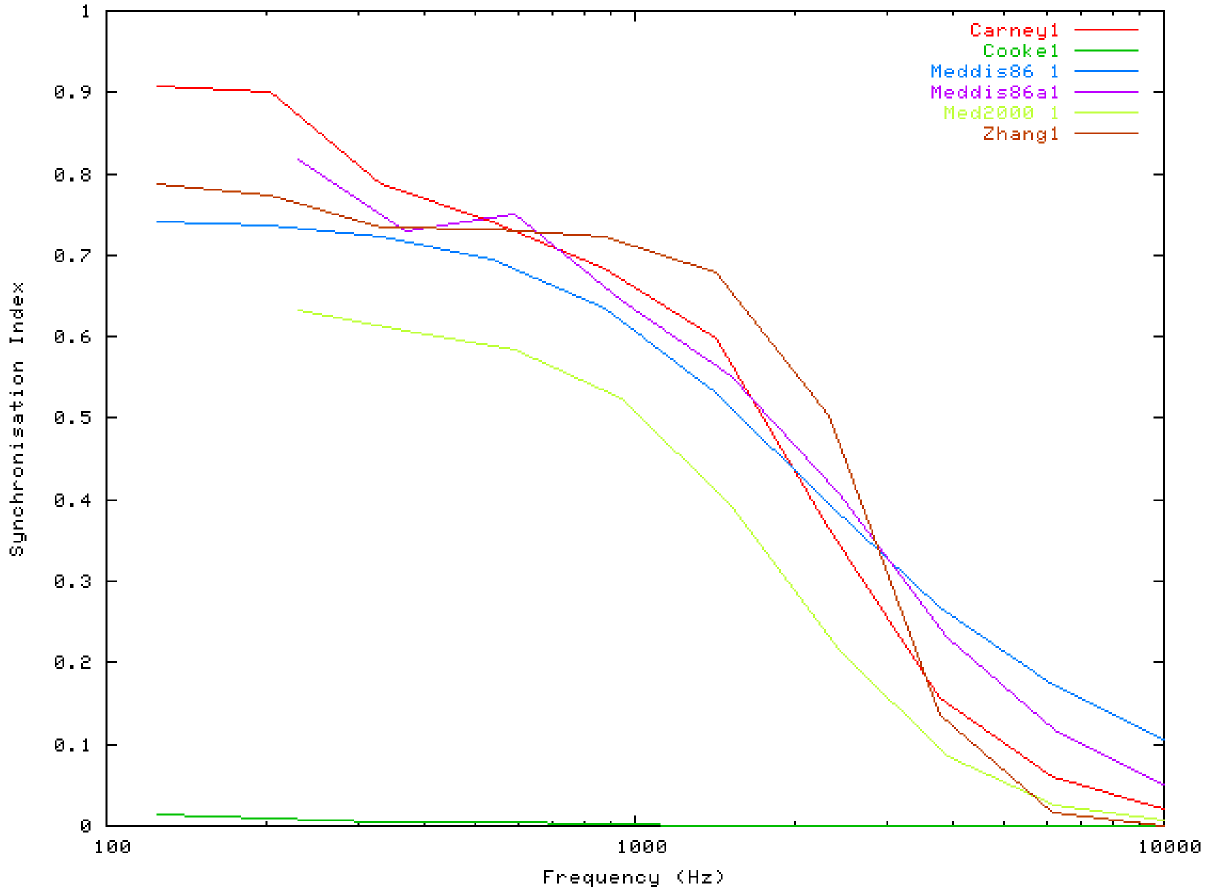
Figure 9.1. Post stimulus time histogram (PSTH) for the Carney AN model



Phase-locking vs Frequency Example: AutoTest/IHC/Carney1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Carney1.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 BM_Carney      < Carney1k.par
 IHCRP_Carney     < CarneyRP1.par
 IHC_Carney      < CarneyIHC1.par
 Ana_Histogram    < PH1.par

}
```

Figure 9.2. Phase-locking vs frequency Firing characteristics for the DSAM AN models



Phase-locking vs Intensity Example: AutoTest/IHC/Carney2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Carney2.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 BM_Carney       < Carney1k.par
 IHCRP_Carney     < CarneyRP1.par
 IHC_Carney      < CarneyIHC1.par

}
```

Figure 9.3. Carney AN Model: Firing characteristics vs intensity

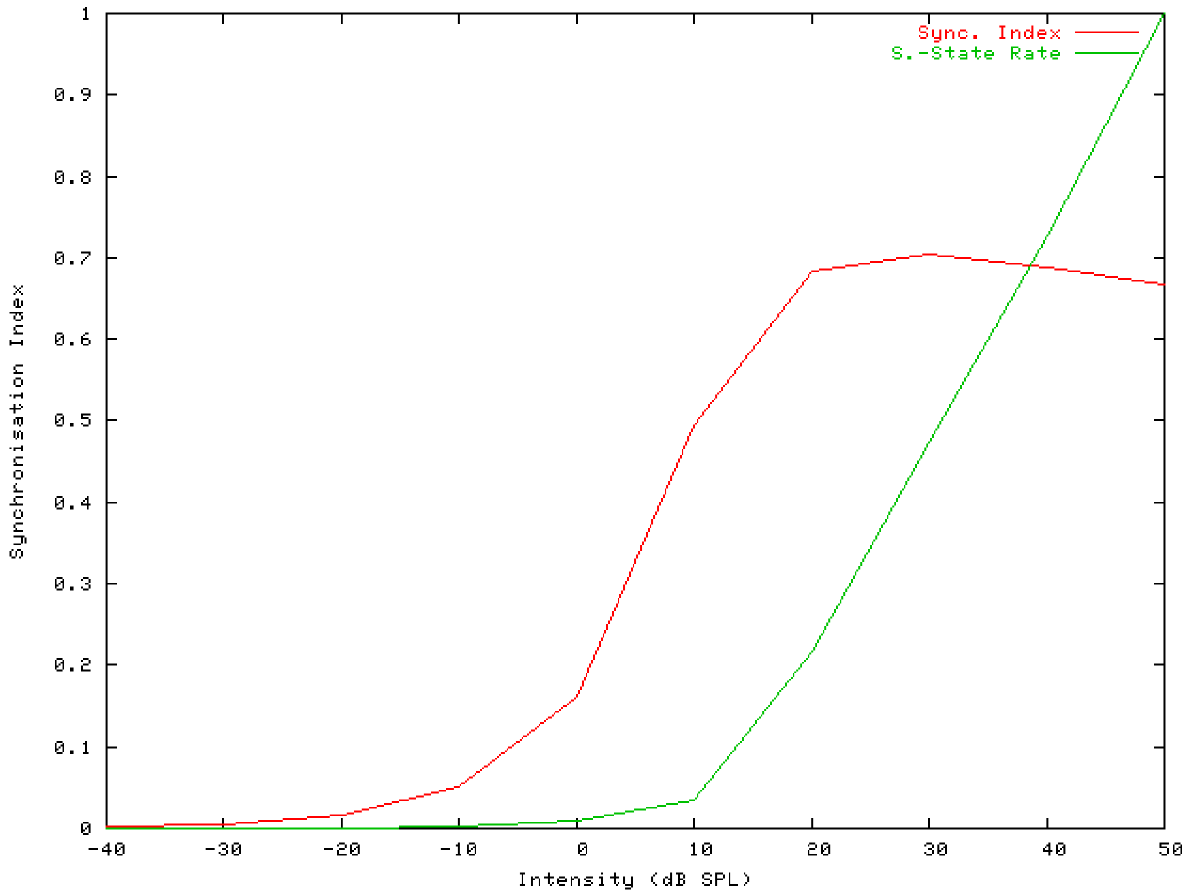
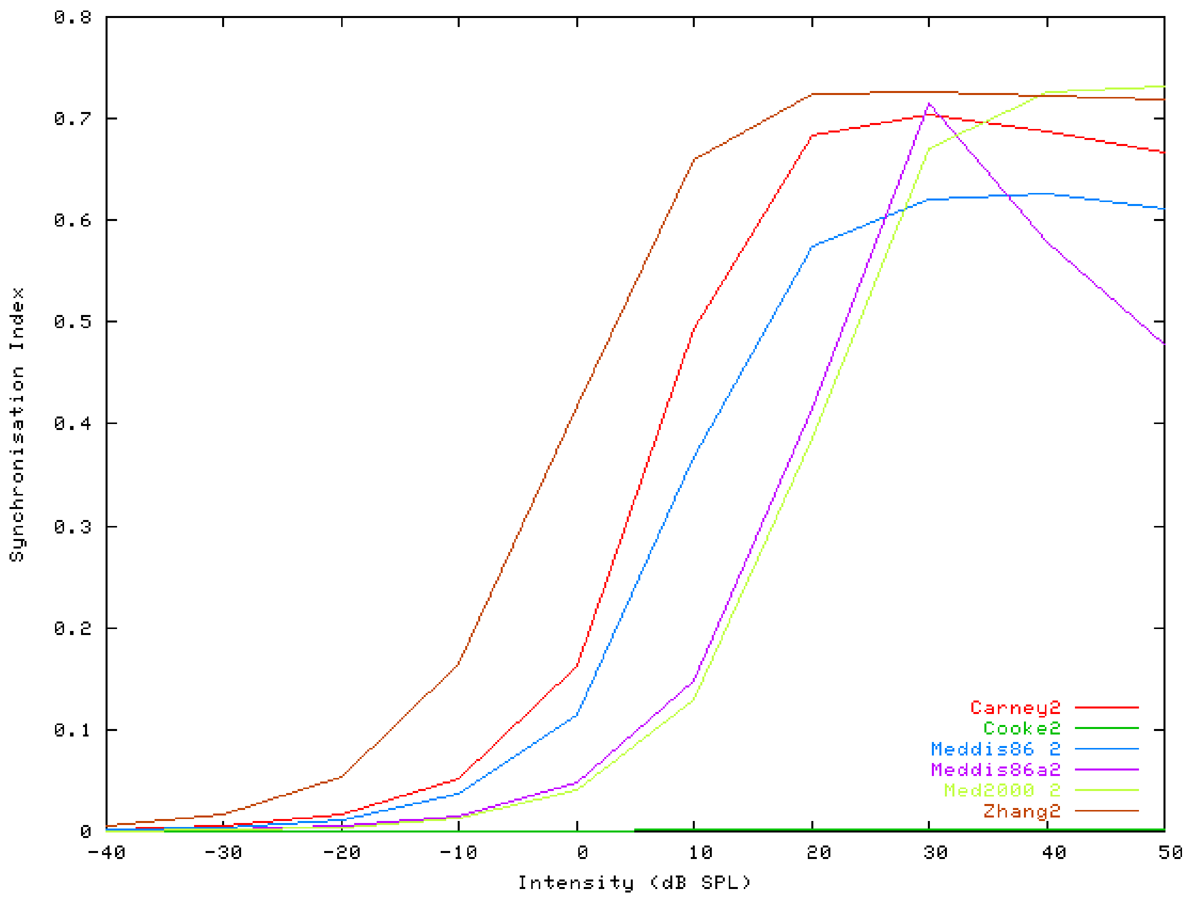


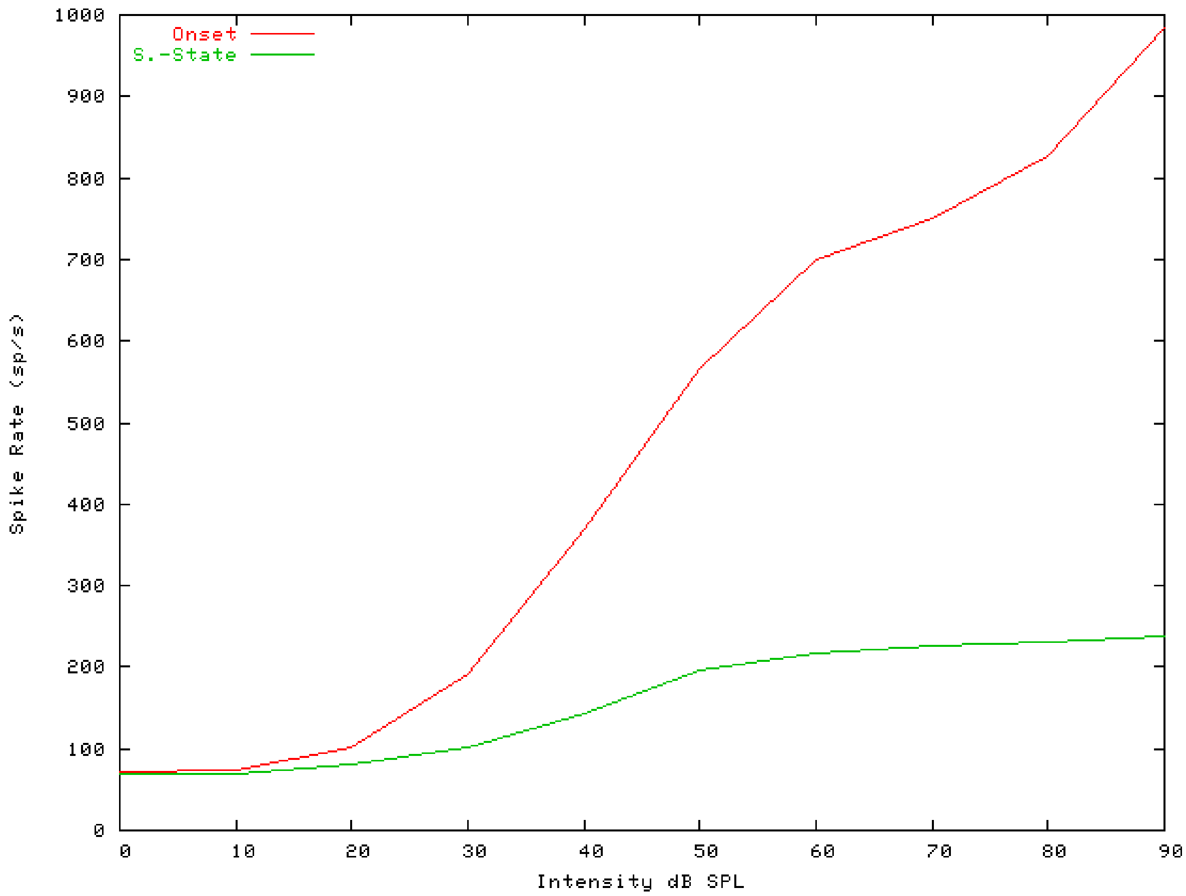
Figure 9.4. Phase-locking vs Intensity vs intensity.Firing characteristics for the DSAM AN models



Rate-Intensity Functions (Spike Probabilities) Example: AutoTest/IHC/Carney3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Carney3.sim
begin  {

 Stim_Puretone    < PTone2.par
 Trans_Gate      < Ramp1.par
 BM_Carney       < Carney1k.par
 IHCRP_Carney     < CarneyRP1.par
 reset hi
mr% repeat 1 {
 IHC_Carney      < CarneyIHC1.par
 hi% Ana_Histogram    < PSTH1.par
 }

}
```

Figure 9.5. Carney AN model: Rate-intensity functions



Rate-Intensity Functions (Multiple Runs) Example: AutoTest/IHC/Carney4.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Carney4.sim
begin  {

 Stim_Puretone   < PTone2.par
 Trans_Gate     < Ramp1.par
 BM_Carney      < Carney1k.par
 IHCRP_Carney    < CarneyRP1.par
 IHC_Carney     < CarneyIHC1.par
 reset hi
mr% repeat 200 {
 AN_SG_Carney    < Carney1SG.par
 hi% Ana_Histogram   < PSTH2.par
 }

}
```

Figure 9.6. Carney AN model: Rate-intensity functions Rate-intensity functions for the Carney AN model. Note that the main parameter file was changed to produce this diagram: the *AN_SG_Carney* module was put in (using the ’Carney1SG.par’ module parameter file, the ’Histogm2.par’ module parameter file was replaced by’Histogm4.par’ and the number of model runs was set to 200.

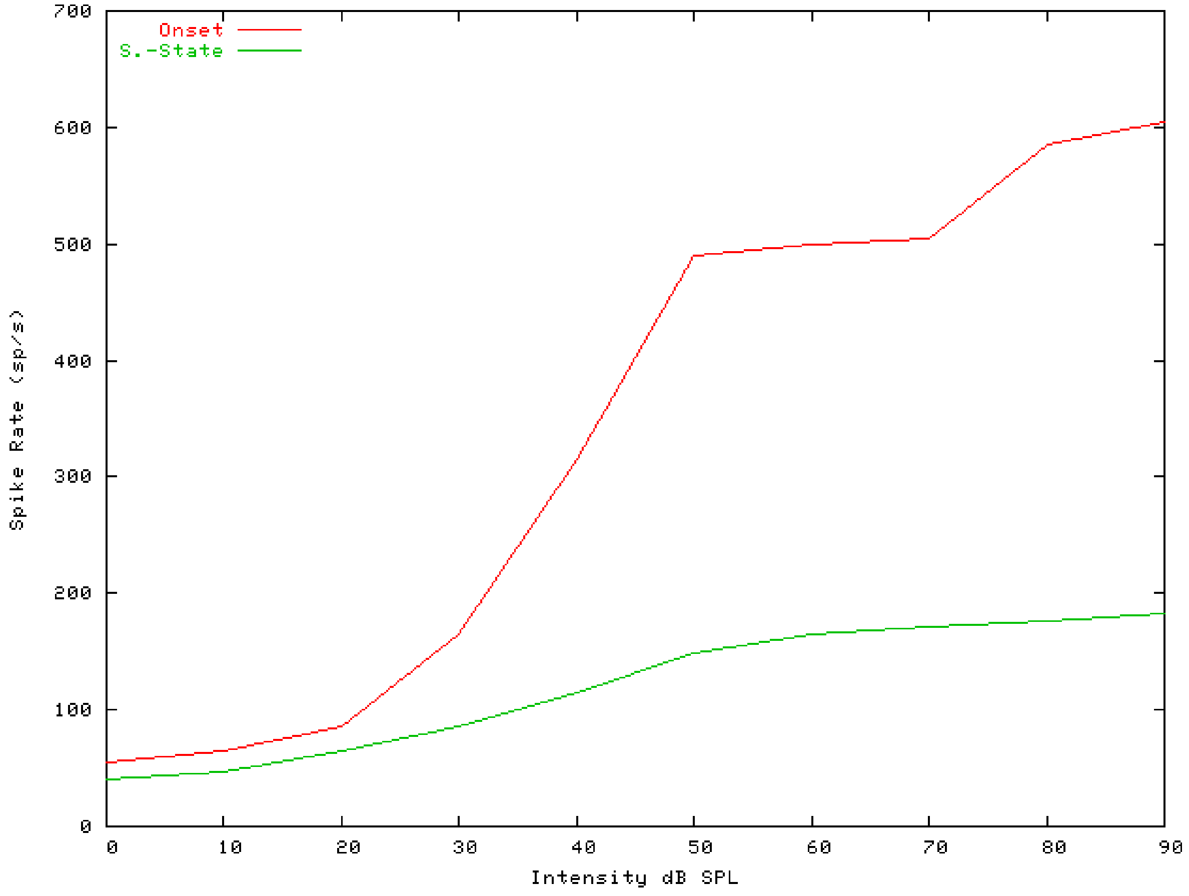
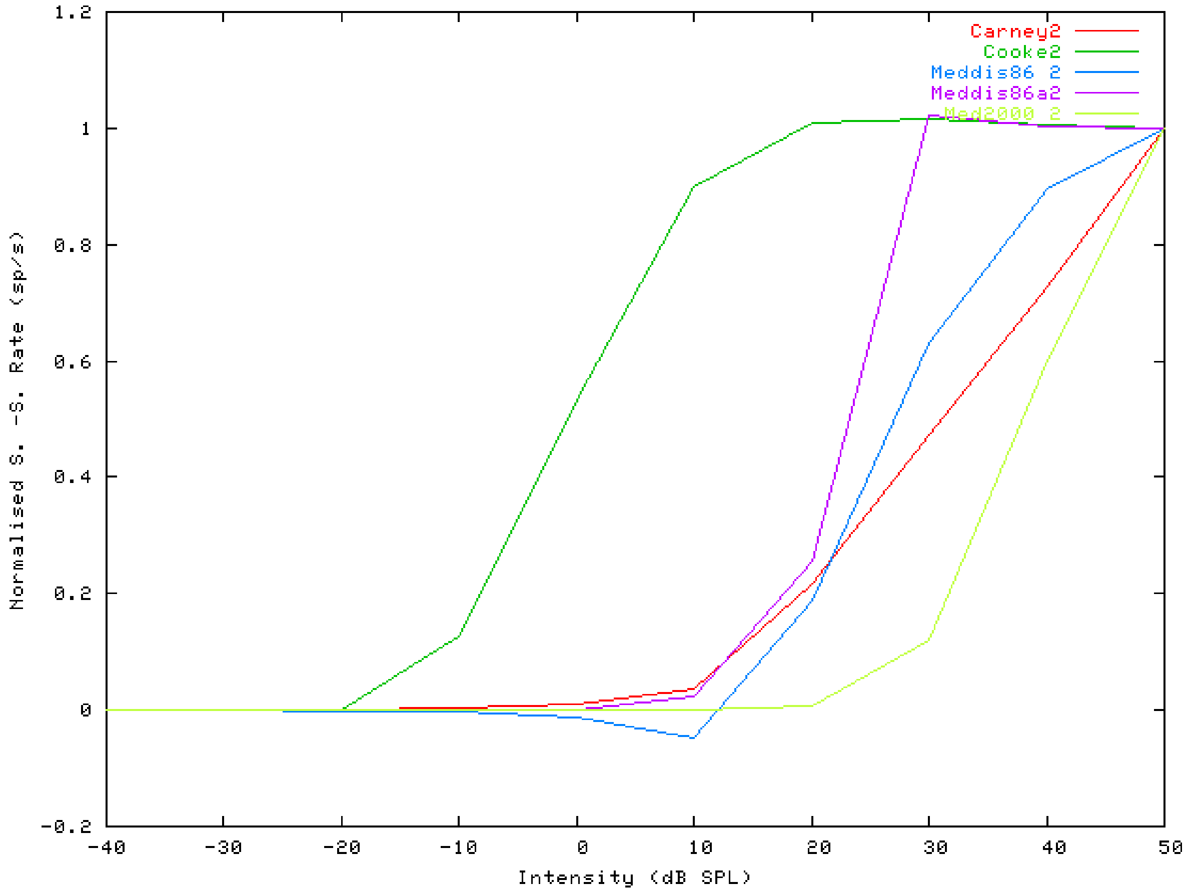


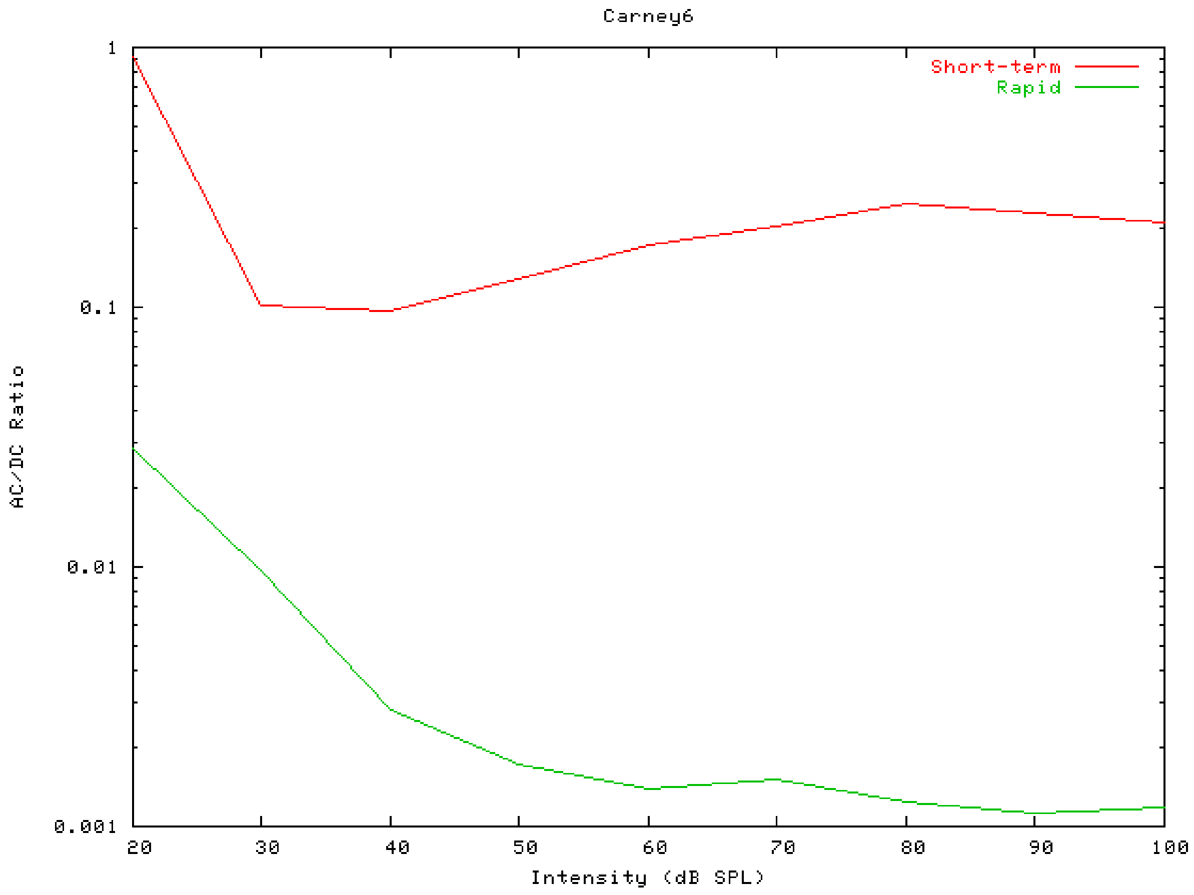
Figure 9.7. Steady-state firing rate vs intensity comparisonFiring characteristics for the DSAM AN models



Two-component Adaptation Example: AutoTest/IHC/Carney6.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Carney6.sim
begin  {

 Stim_Puretone    < PTone4.par
 Trans_Gate      < Ramp1.par
 BM_Carney       < Carney1k.par
 IHCRP_Carney     < CarneyRP1.par
 IHC_Carney      < CarneyIHC1.par
 reset hi
hi% Ana_Histogram    < PSTH5.par

}
```

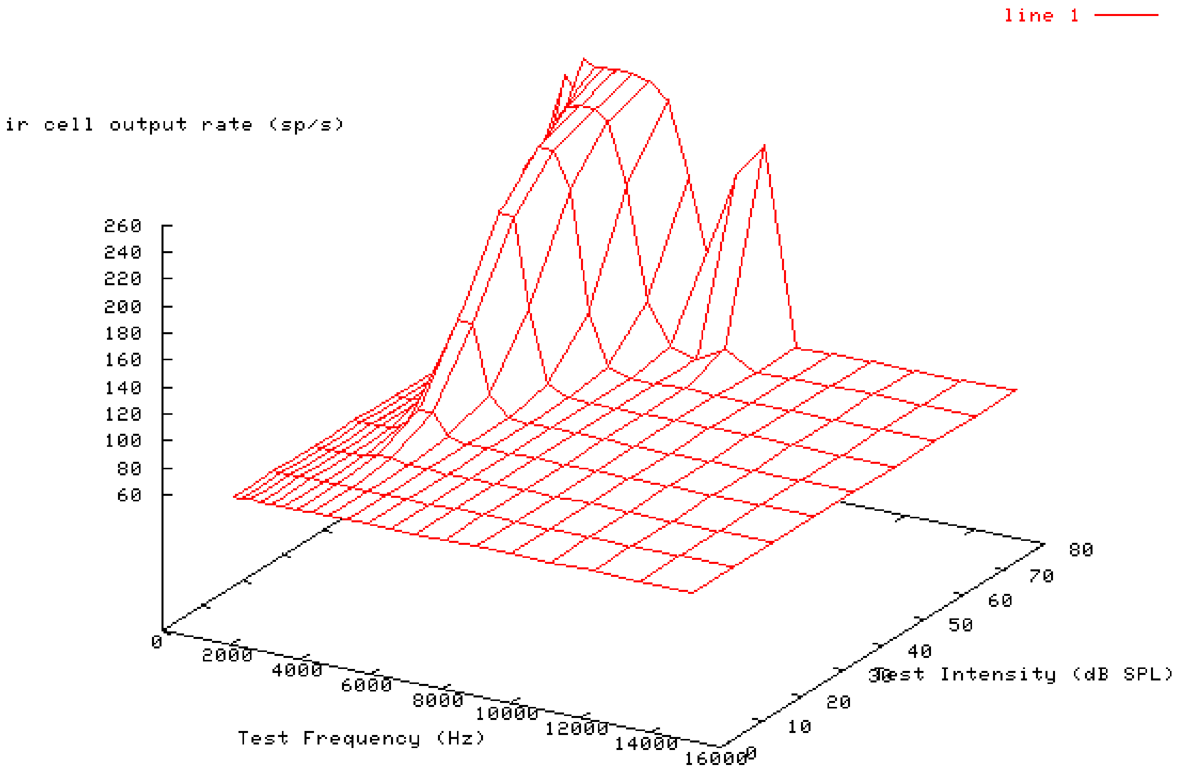
Figure 9.8. Carney AN Model: Two-component adaptation characteristics



Two-Tone Suppression Example: AutoTest/IHC/Carney7.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Carney7.sim
begin  {

 Stim_Puretone_Multi < MPTone1.par
 Trans_Gate      < Ramp1.par
 BM_Carney       < Carney4k.par
 IHCRP_Carney     < CarneyRP1.par
 IHC_Carney      < CarneyIHC1.par
 reset hi
mr% repeat 1 {
 hi% Ana_Histogram    < PSTH5.par
 }

}
```

Figure 9.9. Carney AN Model: Two-tone suppression characteristics.



IHC_Cooke91: Inner Hair Cell (1991)

MoIHCCooke.[ch]

Description

Martin Cooke (1991) inner hair cell model.

Inputs	Aribrary single input, however, it expects input from an inner hair cell receptor potential process.
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	

Module Parameters

Example 9.2. IHC_Cooke91 Parameter File

C_VALUE	100	Crawford and Fettiplace c value.
RELEASE	24	Release fraction.
REFILL	6	Replenishment (refill) fraction.
SPONT_FIRING	50	Desired spontaneous firing rate (spikes/s).
MAX_FIRING	1000	Maximum possible firing rate (spikes/s).

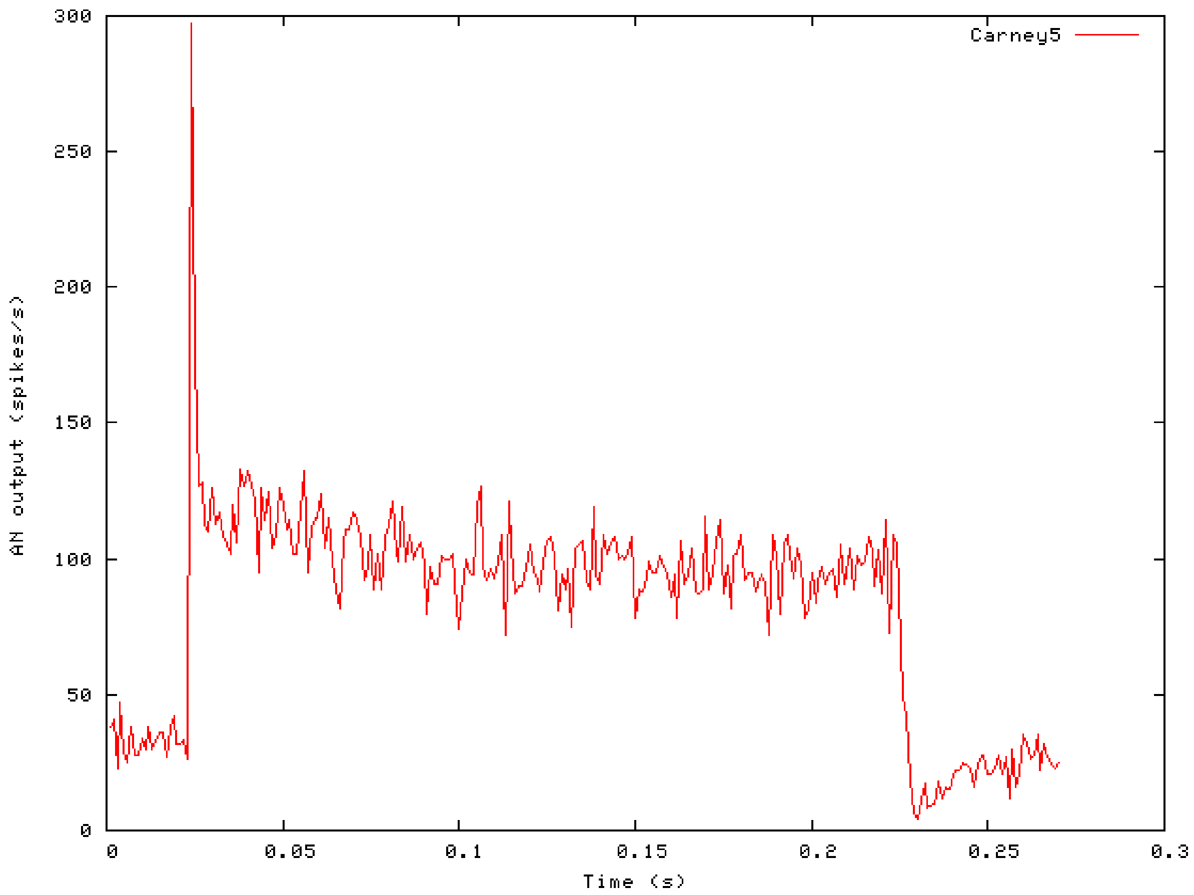
Examples Using the ’IHC_Cooke91’ Process Module

Some of the following tests require multiple observations using a range of stimuli and cannot be produced directly with a single run of the *AMS* program. These simulation scripts are run using the *AutoTest* program, which is employed to test the operation of the DSAM process modules.

Simple Response Example: AutoTest/IHC/Cooke5.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Cooke5.sim
begin  {

 Stim_Puretone_2  < P2Tone1.par
 Trans_Gate      < Ramp1.par
 BM_Cooke       < CookePow1k.par
 IHC_Cooke91     < Cooke91.par
 Ana_Histogram    < PSTH3.par

}
```

Figure 9.10. Post stimulus time histogram (PSTH) for the Cooke AN model



Phase-locking vs Frequency Example: AutoTest/IHC/Cooke1.sim

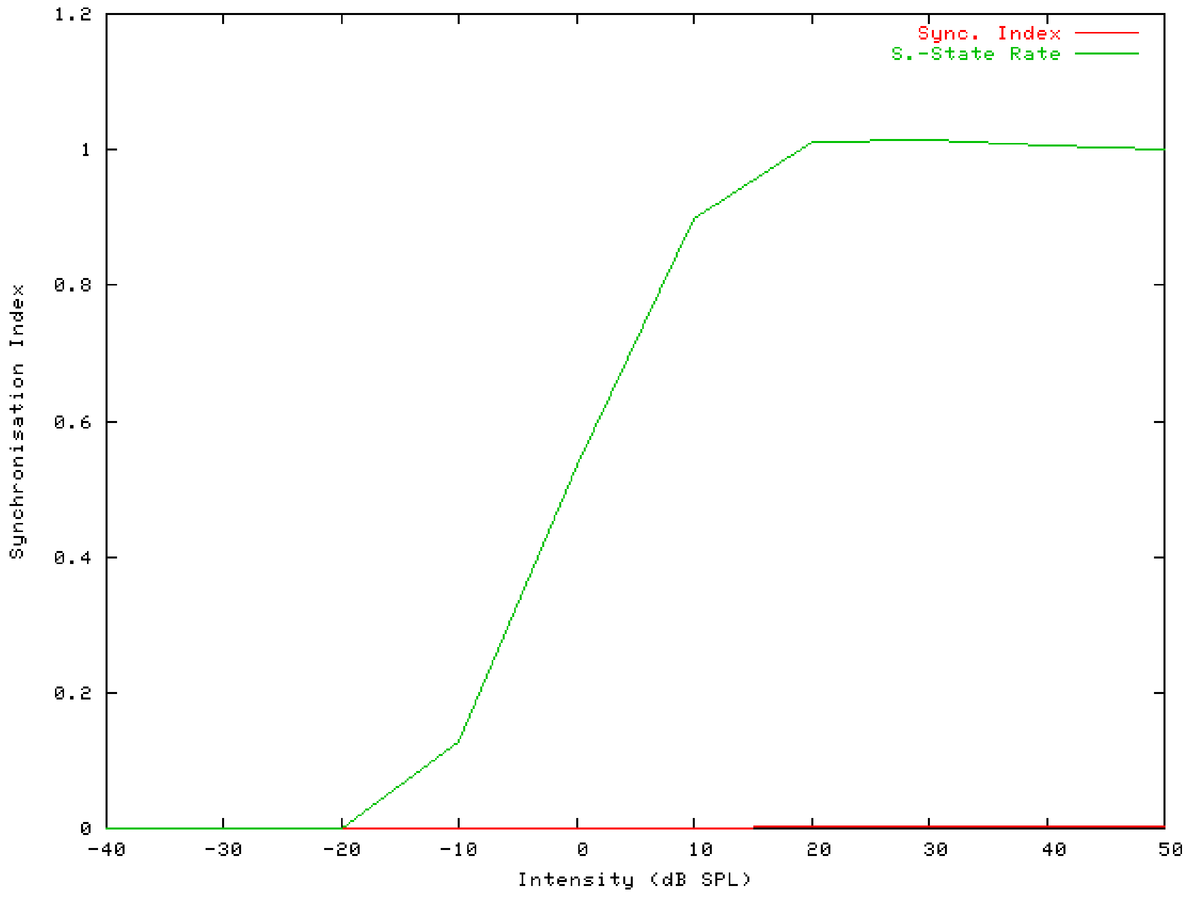
```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Cooke1.sim
begin  {

 Stim_Puretone   < PTone1.par
 Trans_Gate     < Ramp1.par
 BM_Cooke       < CookeAM1k.par
 IHC_Cooke91     < Cooke91.par
 Ana_Histogram   < PH1.par

}
```

See Figure 9.2, “Phase-locking vs frequency Firing characteristics for the DSAM AN models”. Compared to the other models, the Cooke AN model has very poor synchronisation characteristics.

Phase-locking vs Intensity Example: AutoTest/IHC/Cooke2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Cooke2.sim
begin  {

 Stim_Puretone   < PTone1.par
 Trans_Gate     < Ramp1.par
 BM_Cooke       < CookeAM1k.par
 IHC_Cooke91     < Cooke91.par

}
```

Figure 9.11. Cooke AN Model: Firing characteristics vs intensity

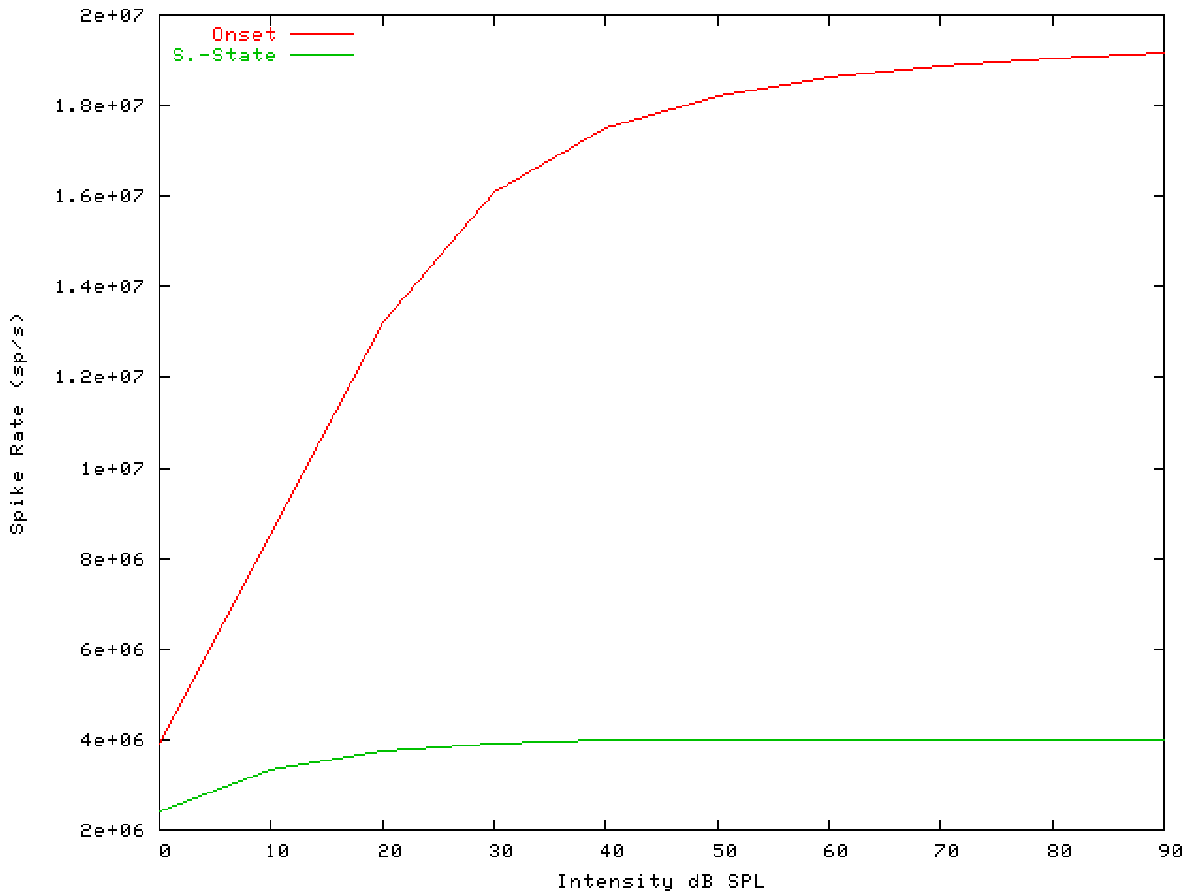


See also Figure 9.4, “Phase-locking vs Intensity vs intensity.Firing characteristics for the DSAM AN models”.

Rate-Intensity Functions (Spike Probabilities) Example: AutoTest/IHC/Cooke3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Cooke3.sim
begin  {

 Stim_Puretone    < PTone2.par
 Trans_Gate      < Ramp1.par
 BM_Cooke       < CookeAM1k.par
 IHC_Cooke91     < Cooke91.par
 reset hi
mr% repeat 1 {
 hi% Ana_Histogram    < PSTH1.par
 }

}
```

Figure 9.12. Cooke AN model: Rate-intensity functions

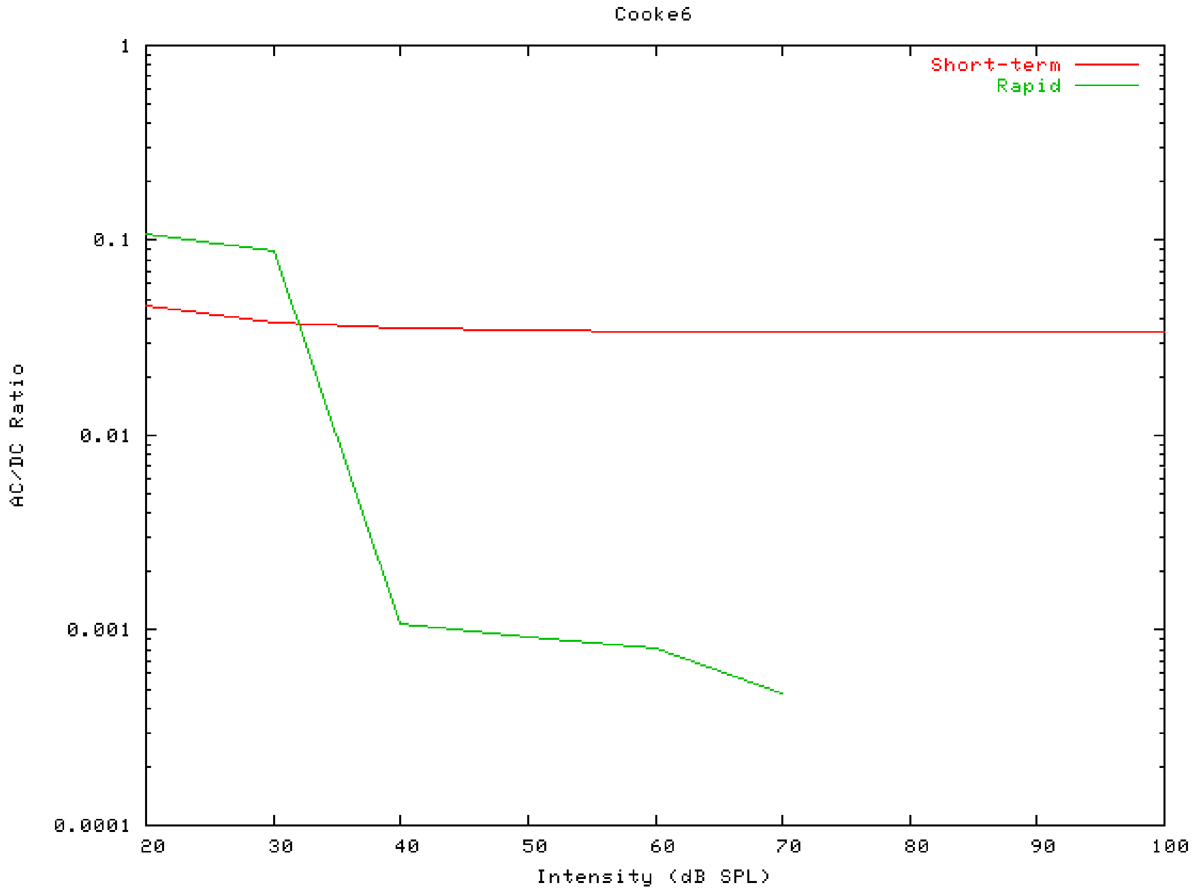


See also Figure 9.7, “Steady-state firing rate vs intensity comparisonFiring characteristics for the DSAM AN models”.

Two-component Adaptation Example: AutoTest/IHC/Cooke6.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Cooke6.sim
begin  {

 Stim_Puretone    < PTone4.par
 Trans_Gate      < Ramp1.par
 BM_Cooke       < CookeAM1k.par
 IHC_Cooke91     < Cooke91.par
 reset hi
hi% Ana_Histogram    < PSTH5.par

}
```

Figure 9.13. Cooke AN Model: Two-component adaptation characteristics



IHC_Meddis86: Inner Hair Cell (1986)

File name: MoIHC86Meddis.[ch]

Description

Inputs	Aribrary single input, usually from the basilar membrane process modules.
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	[20]

Module Parameters

Example 9.3. IHC_Meddis86 Parameter File

PERM_CONST_A	100	Permeability constant A (units/s).
PERM_CONST_B	6000	Permeability constant B (units/s).
RELEASE_G	2000	Release rate (units/s).
REPLENISH_Y	5.05	Replenishment rate (units/s).
LOSS_L	2500	Loss rate (units/s).
REPROCESS_X	66.31	Reprocessing rate (units/s).
RECOVERY_R	6580	Recovery rate (units/s).
MAX_FREE_POOL_M	1	Max. no. of transmitter packets in free pool.
FIRING_RATE_H	50000	Firing rate (spikes/s)

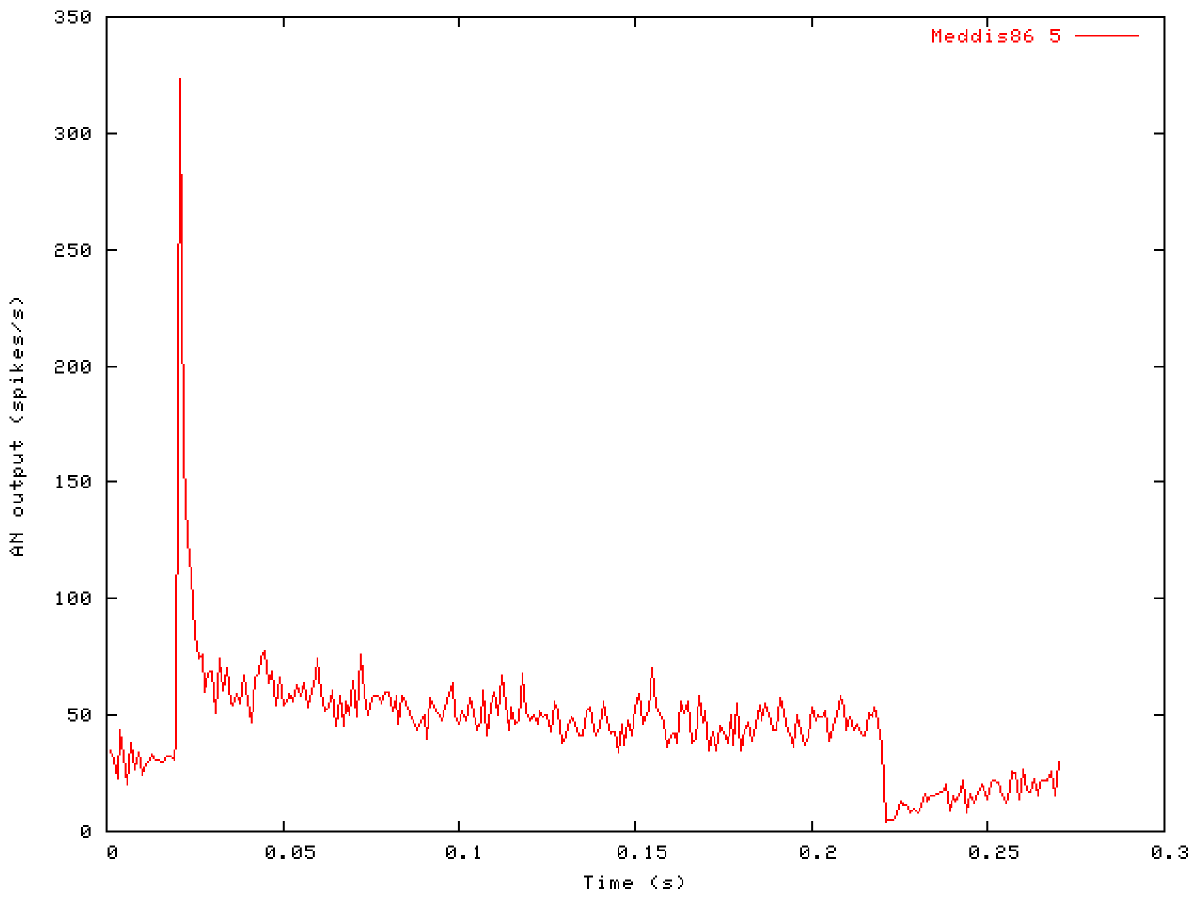
Examples Using the ’IHC_Meddis86’ Process Module

Some of the following tests require multiple observations using a range of stimuli and cannot be produced directly with a single run of the *AMS* program. These simulation scripts are run using the *AutoTest* program, which is employed to test the operation of the DSAM process modules.

Simple Response Example: AutoTest/IHC/Meddis86_5.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Meddis86_5.sim
begin  {

 Stim_Puretone_2  < P2Tone1.par
 Trans_Gate      < Ramp1.par
 IHC_Meddis86    < Meddis86.par
 AN_SG_Binomial   < BinomSG1.par
 Ana_Histogram   < PSTH3.par

}
```

Figure 9.14. Post stimulus time histogram (PSTH) for the Carney AN model



Phase-locking vs Frequency Example: AutoTest/IHC/Meddis86_1.sim

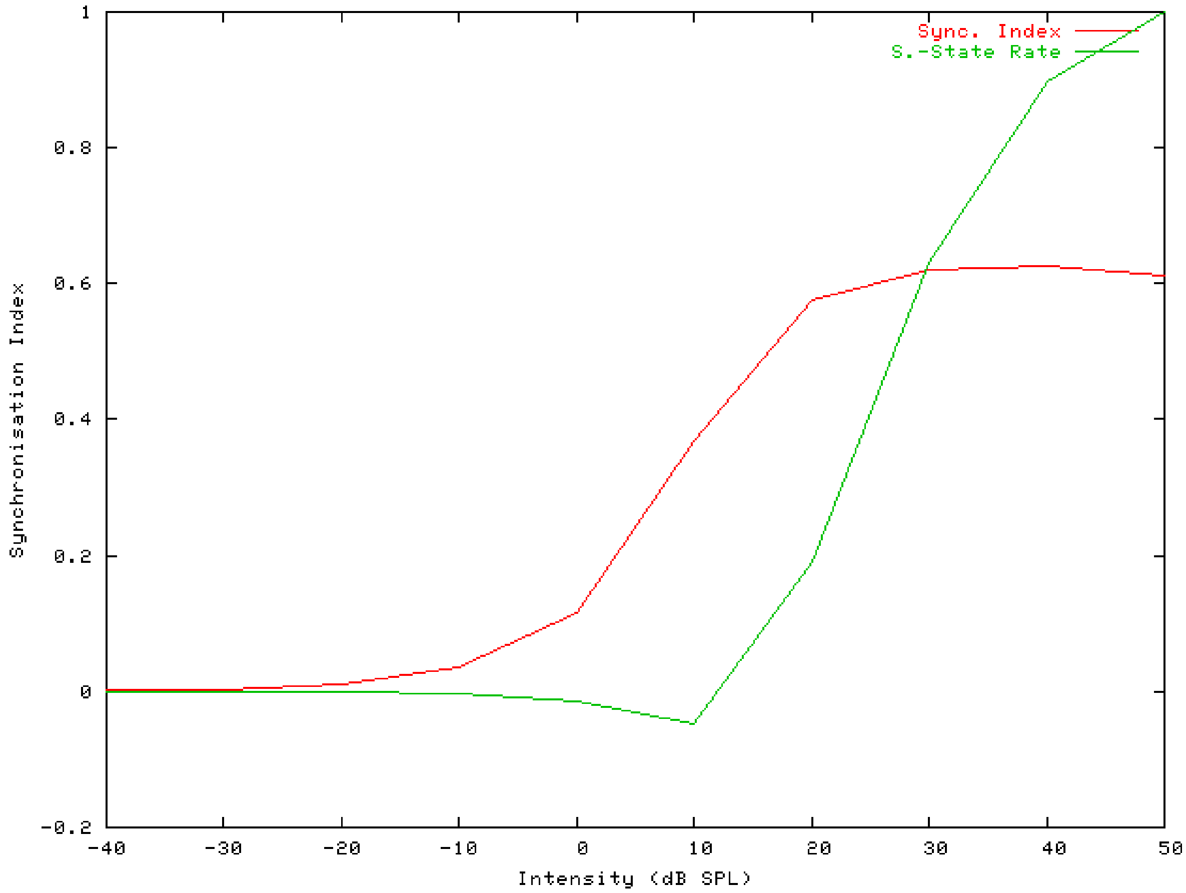
```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Meddis86_2.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 IHC_Meddis86     < Meddis86.par
 Ana_Histogram    < PH1.par

}
```

See Figure 9.2, “Phase-locking vs frequency Firing characteristics for the DSAM AN models”.

Phase-locking vs Intensity Example: AutoTest/IHC/Meddis86_2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.
# ./IHC/Meddis86_2.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 IHC_Meddis86     < Meddis86.par

}
```

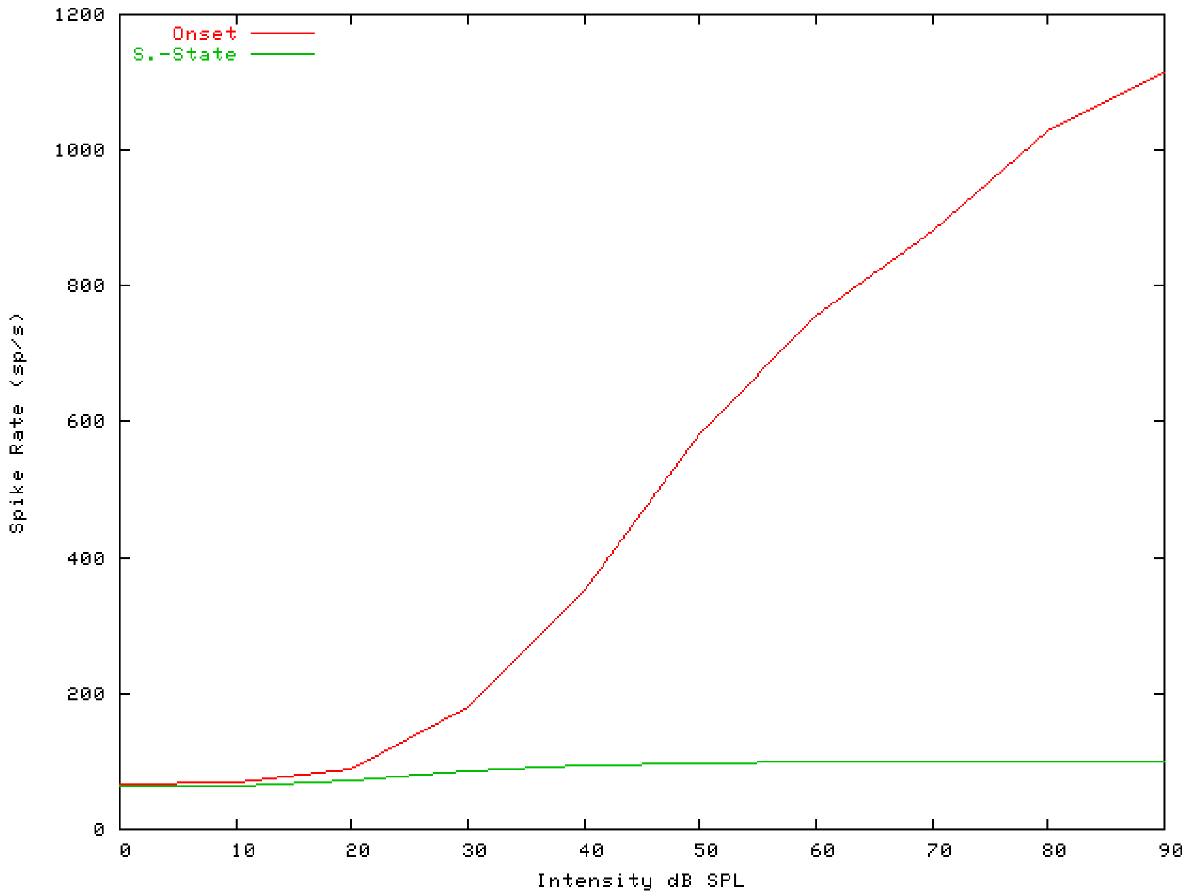
Figure 9.15. Meddis86 AN Model: Firing characteristics vs intensity



Rate-Intensity Functions (Spike Probabilities) Example: AutoTest/IHC/Meddis86_3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Meddis86_3.sim
begin  {

 Stim_Puretone    < PTone2.par
 Trans_Gate      < Ramp1.par
 IHC_Meddis86    < Meddis86.par
 reset hi
mr% repeat 1 {
 hi% Ana_Histogram    < PSTH1.par
 }

}
```

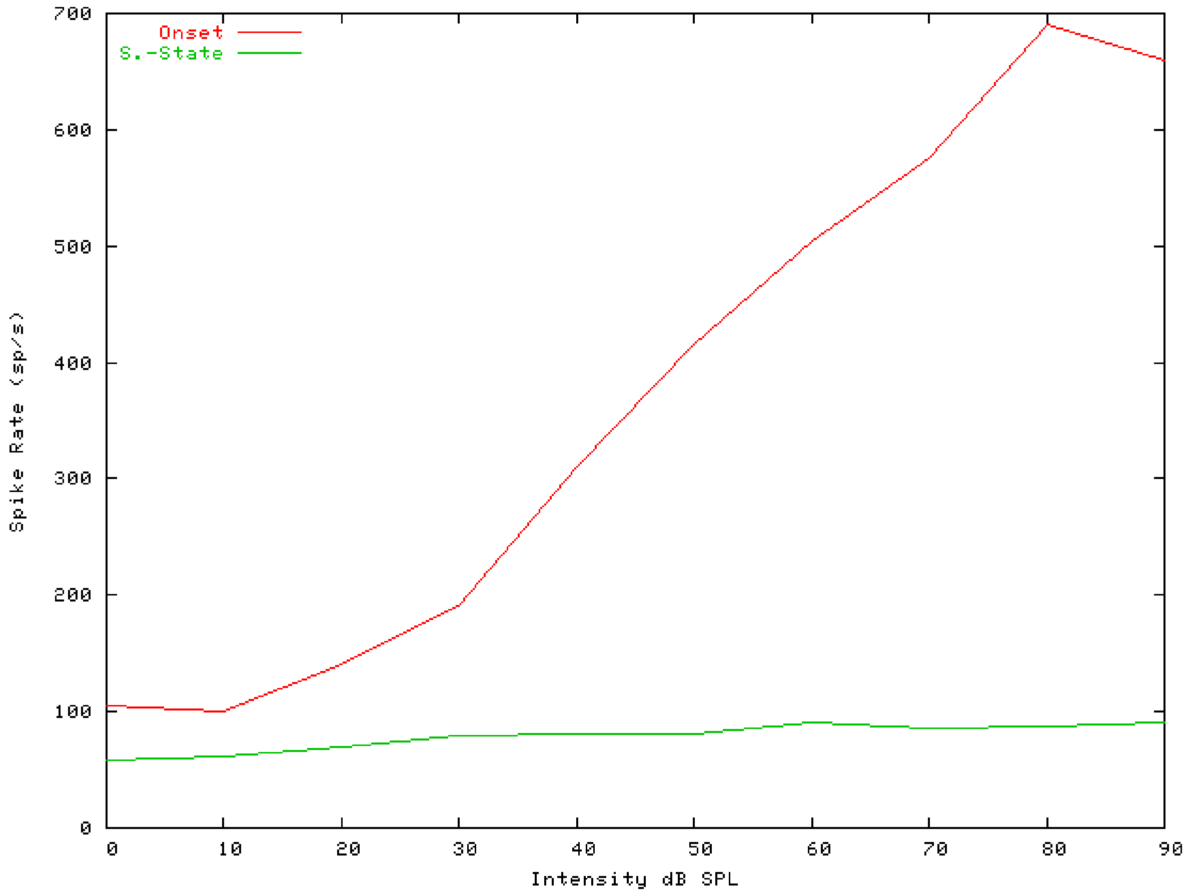
Figure 9.16. Meddis86 AN model: Rate-intensity functions



Rate-Intensity Functions (Multiple Runs) Example: AutoTest/IHC/Meddis86_4.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Meddis86_4.sim
begin  {

 Stim_Puretone    < PTone2.par
 Trans_Gate      < Ramp1.par
 IHC_Meddis86    < Meddis86.par
 reset hi
mr% repeat 200 {
 AN_SG_Simple    < SpikeGn1.par
 hi% Ana_Histogram    < PSTH2.par
 }

}
```

Figure 9.17. Meddis86 AN model: Rate-intensity functions Rate-intensity functions for the Meddis86_ AN model. Note that the main parameter file was changed to produce this diagram: the *AN_SG_Meddis86_* module was put in (using the ’Meddis86_1SG.par’ module parameter file, the ’Histogm2.par’ module parameter file was replaced by’Histogm4.par’ and the number of model runs was set to 200.

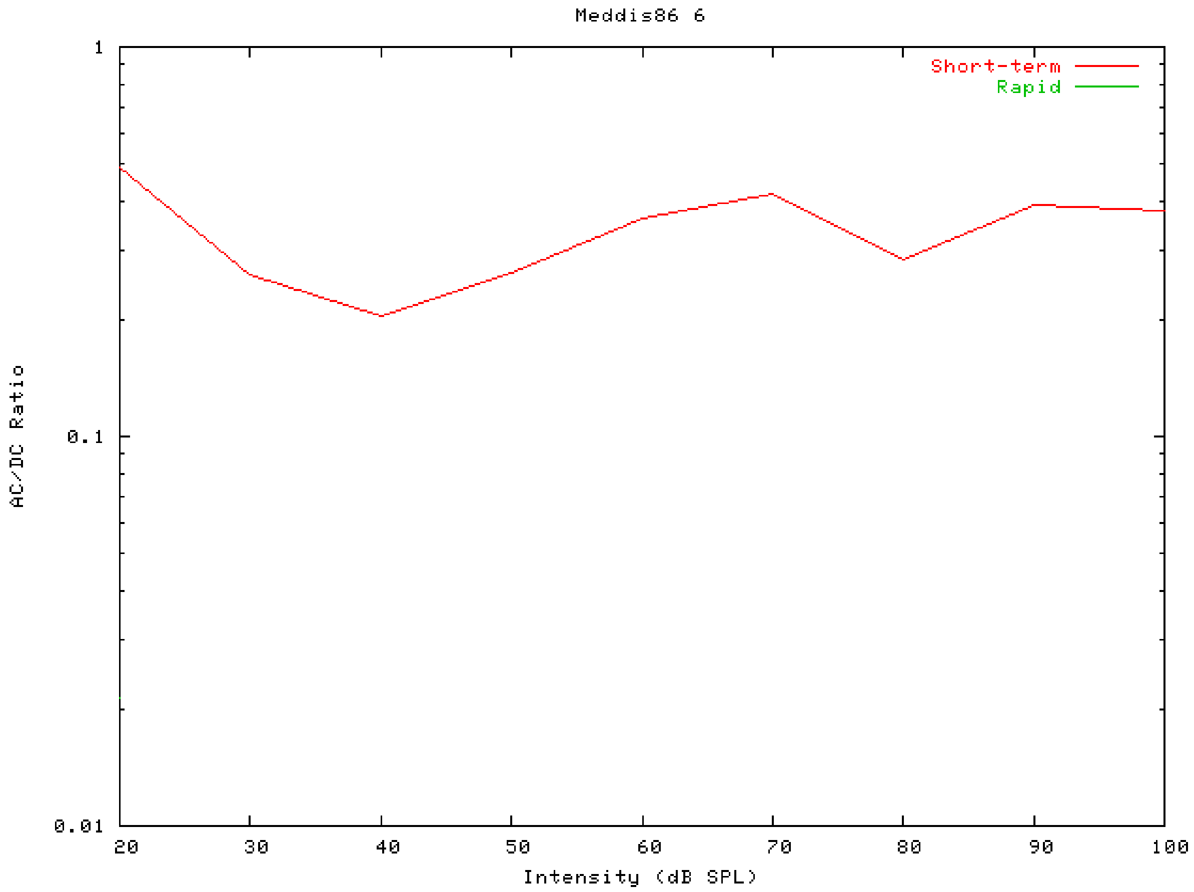


See also Figure 9.7, “Steady-state firing rate vs intensity comparisonFiring characteristics for the DSAM AN models”.

Two-component Adaptation Example: AutoTest/IHC/Meddis86_6.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.
# ./IHC/Meddis86_6.sim
begin  {

 Stim_Puretone    < PTone4.par
 Trans_Gate      < Ramp1.par
 IHC_Meddis86    < Meddis86.par
 reset hi
hi% Ana_Histogram    < PSTH5.par

}
```

Figure 9.18. Meddis86 AN Model: Two-component adaptation characteristics



IHC_Meddis86a: Inner Hair Cell Synapse

File name: MoIHC86aMeddis.[ch]

Description

This module is the Meddis 86a inner hair cell (IHC) synapse module. It is the same as the Meddis 86 IHC model, but using an exponential permeability function. It requires an IHC receptor potential module as input. It outputs spike probabilities.

Inputs	Aribrary single input, however, it expects input from an inner hair cell receptor potential process.
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	

Module Parameters

Example 9.4. IHC_Meddis86a Parameter File

MAX_FREE_POOL	1	Max. no. of transmitter packets in free pool.
REPLENISH_Y	5.05	Replenishment rate (units per second).
LOSS_L	2500	Loss rate (units per second).
REPROCESS_X	66.31	Reprocessing rate (units per second).
RECOVERY_R	6580	Recovery rate (units per second).
PERM_CONST_H	800	Permeability constant, h.
PERM_CONST_Z	8.6538e+10	Permeability constant, z.
FIRING_RATE_H2	50000	Firing rate (spikes per second).

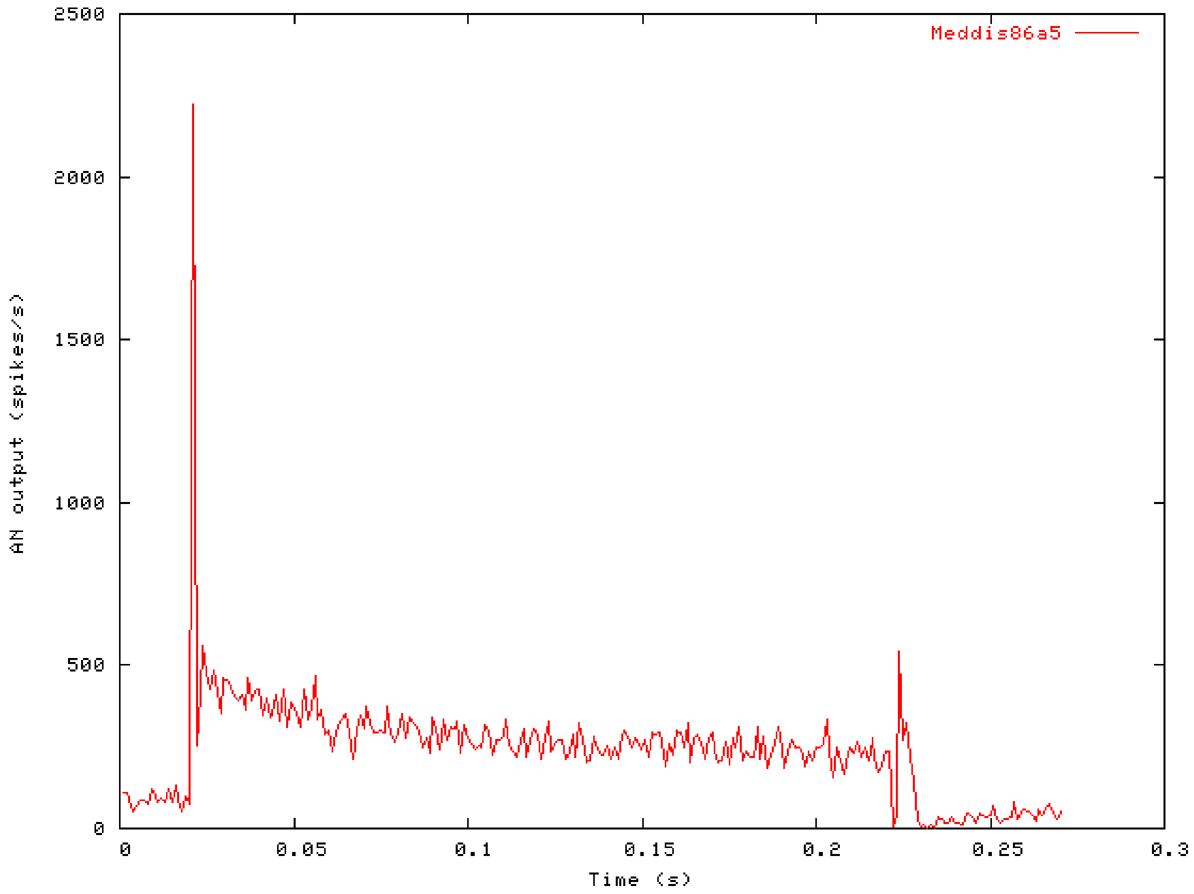
Examples Using the ’IHC_Meddis86a’ Process Module

Some of the following tests require multiple observations using a range of stimuli and cannot be produced directly with a single run of the *AMS* program. These simulation scripts are run using the *AutoTest* program, which is employed to test the operation of the DSAM process modules.

Simple Response Example: AutoTest/IHC/Meddis86a5.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Meddis86a5.sim
begin  {

 Stim_Puretone_2  < P2Tone3.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEDRNL1.par
 BM_DRNL        < DRNL18k.par
 IHCRP_Shamma    < ShammaRP1.par
 IHC_Meddis86a    < Medd86aH.par
 AN_SG_Binomial   < BinomSG1.par
 Ana_Histogram    < PSTH3.par

}
```

Figure 9.19. Post stimulus time histogram (PSTH) for the Meddis 86a AN model



Phase-locking vs Frequency Example: AutoTest/IHC/Meddis86a1.sim

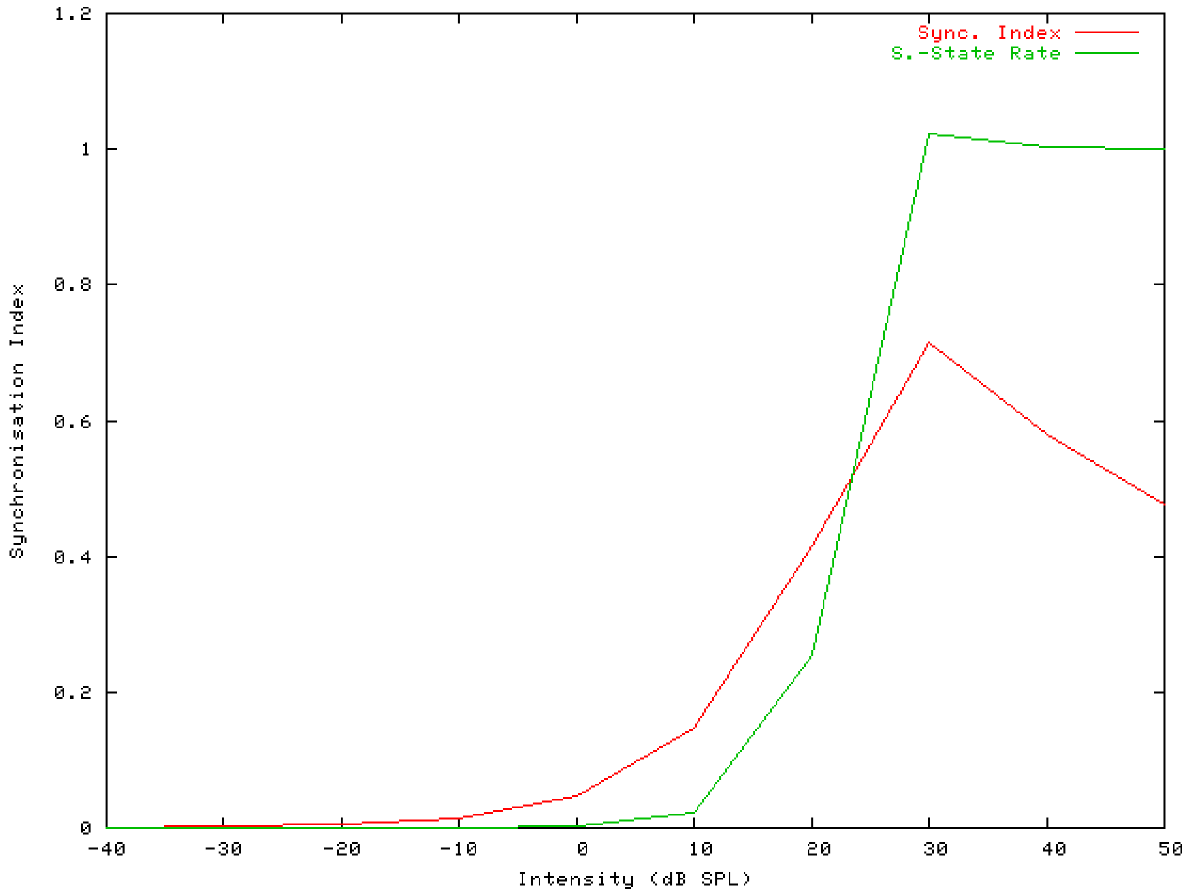
```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Meddis86a1.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEDRNL1.par
 BM_DRNL         < DRNL18k.par
 IHCRP_Shamma     < ShammaRP1.par
 IHC_Meddis86a    < Medd86aM.par
 Ana_Histogram    < PH1.par

}
```

See Figure 9.2, “Phase-locking vs frequency Firing characteristics for the DSAM AN models”.

Phase-locking vs Intensity Example: AutoTest/IHC/Meddis86a2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Meddis86a2.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEDRNL1.par
 BM_DRNL         < DRNL1k.par
 IHCRP_Shamma     < ShammaRP1.par
 IHC_Meddis86a    < Medd86aH.par

}
```

Figure 9.20. Meddis 86a AN Model: Firing characteristics vs intensity

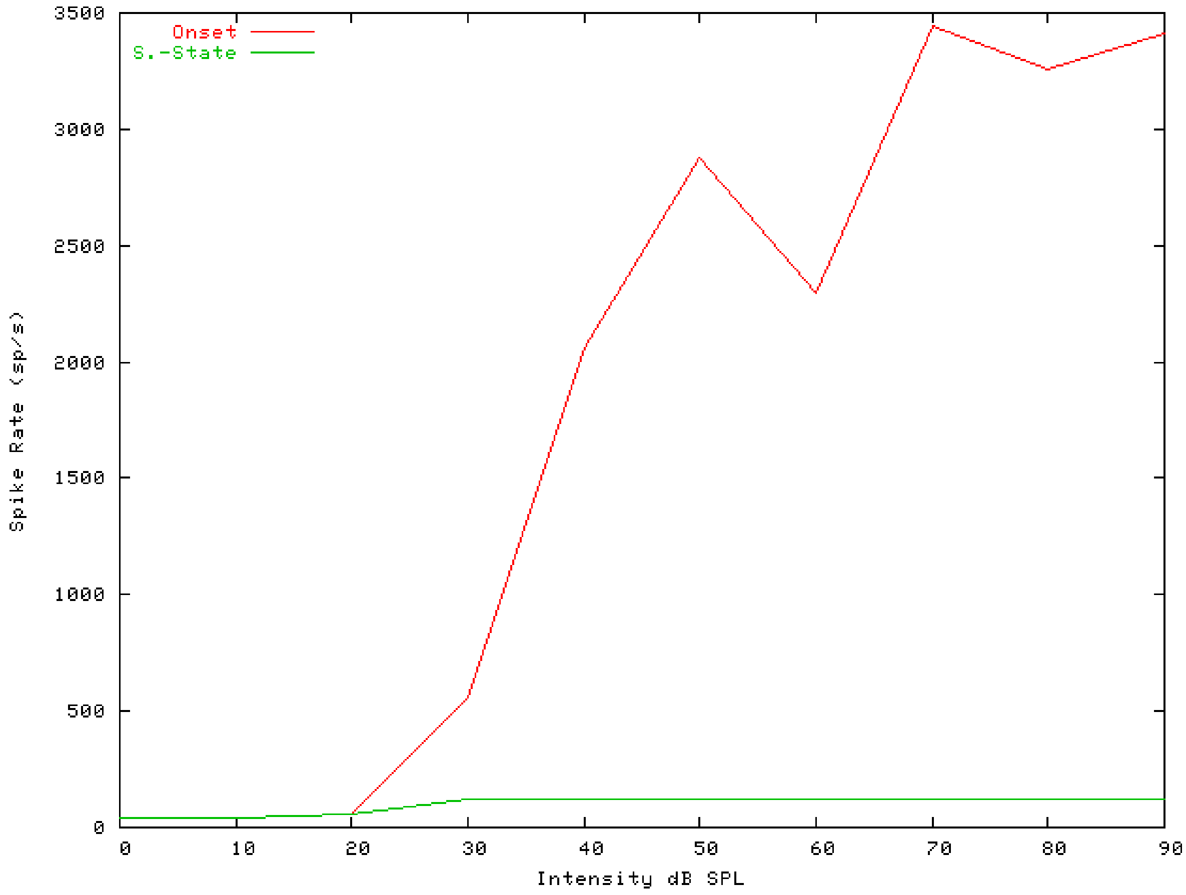


See also Figure 9.4, “Phase-locking vs Intensity vs intensity.Firing characteristics for the DSAM AN models”.

Rate-Intensity Functions (Spike Probabilities) Example: AutoTest/IHC/Meddis86a3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Meddis86a3.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEDRNL1.par
 BM_DRNL        < DRNL1k.par
 IHCRP_Shamma     < ShammaRP1.par
 IHC_Meddis86a    < Medd86aH.par
 reset hi
mr% repeat 1 {
 hi% Ana_Histogram    < PSTH1.par
 }

}
```

Figure 9.21. Meddis86a AN model: Rate-intensity functions

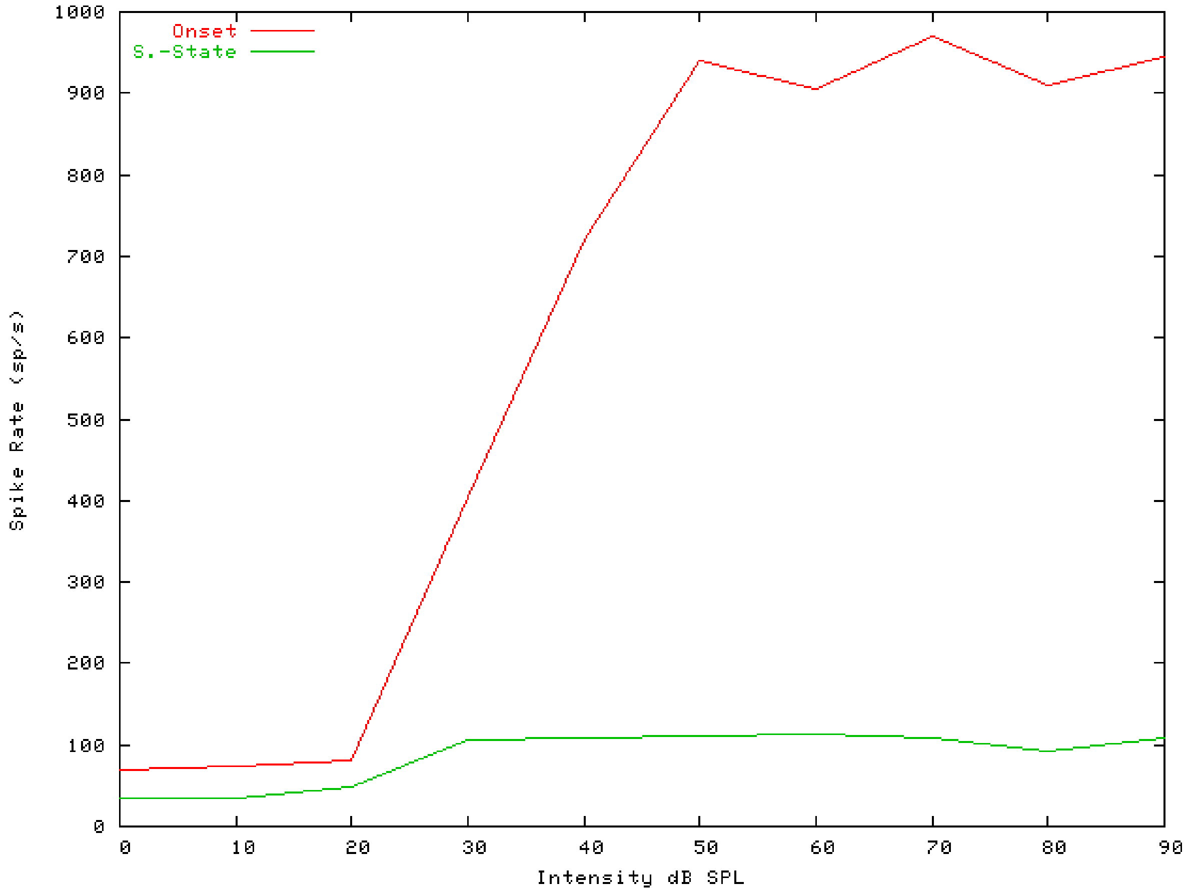


See also Figure 9.7, “Steady-state firing rate vs intensity comparisonFiring characteristics for the DSAM AN models”.

Rate-Intensity Functions (Multiple Runs) Example: AutoTest/IHC/Meddis86a4.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Meddis86a4.sim
begin  {

 Stim_Puretone   < PTone1.par
 Trans_Gate     < Ramp1.par
 Filt_BandPass   < PEDRNL1.par
 BM_DRNL       < DRNL1k.par
 IHCRP_Shamma    < ShammaRP1.par
 IHC_Meddis86a   < Medd86aH.par
 reset hi
mr% repeat 200 {
 AN_SG_Simple    < SpikeGn1.par
 hi% Ana_Histogram   < PSTH2.par
 }

}
```

Figure 9.22. Meddis 86a AN model: Rate-intensity functions Rate-intensity functions for the Meddis 86a AN model. Note that the main parameter file was changed to produce this diagram: the *AN_SG_Meddis86a* module was put in (using the ’Meddis86a1SG.par’ module parameter file, the ’Histogm2.par’ module parameter file was replaced by’Histogm4.par’ and the number of model runs was set to 200.

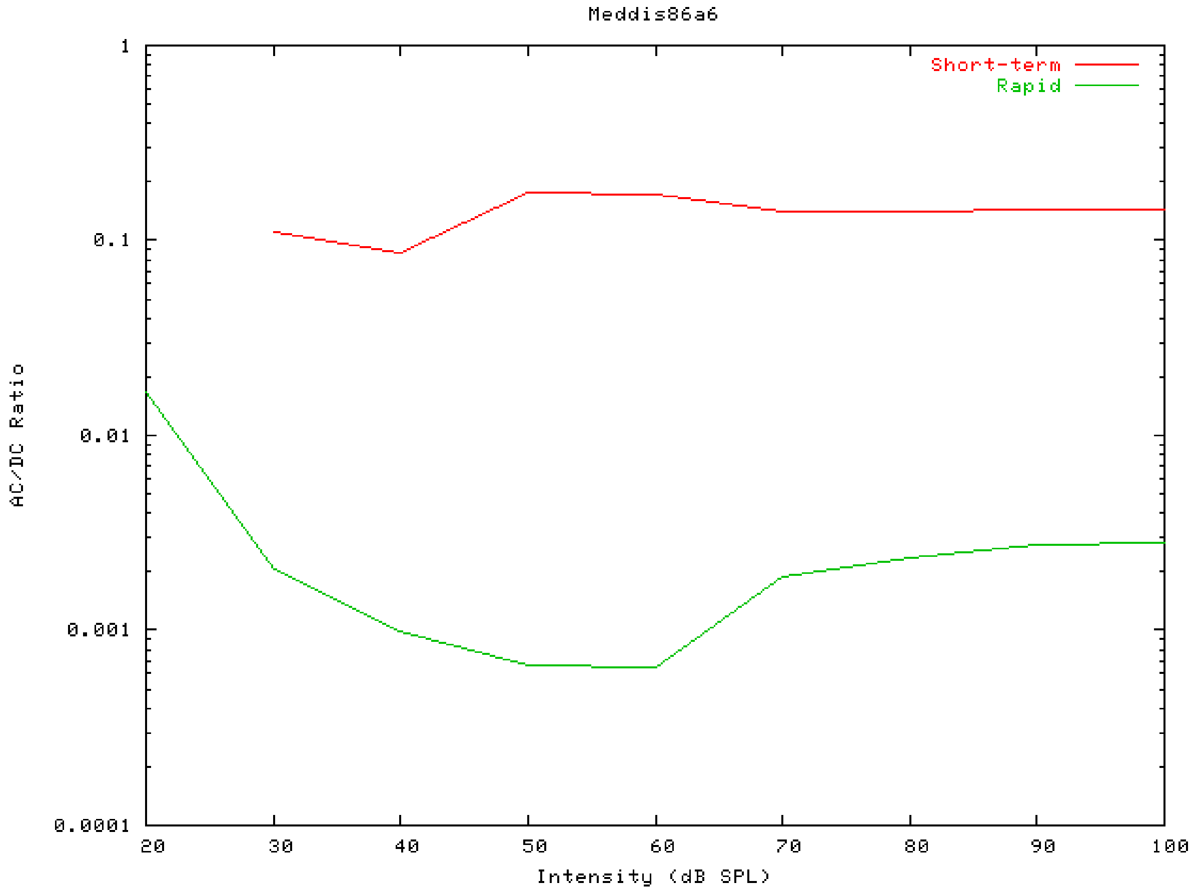


See also Figure 9.7, “Steady-state firing rate vs intensity comparisonFiring characteristics for the DSAM AN models”.

Two-component Adaptation Example: AutoTest/IHC/Meddis86a6.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Meddis86a6.sim
begin  {

 Stim_Puretone    < PTone4.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEDRNL1.par
 BM_DRNL        < DRNL1k.par
 IHCRP_Shamma     < ShammaRP1.par
 IHC_Meddis86a    < Medd86aH.par
 reset hi
hi% Ana_Histogram    < PSTH5.par

}
```

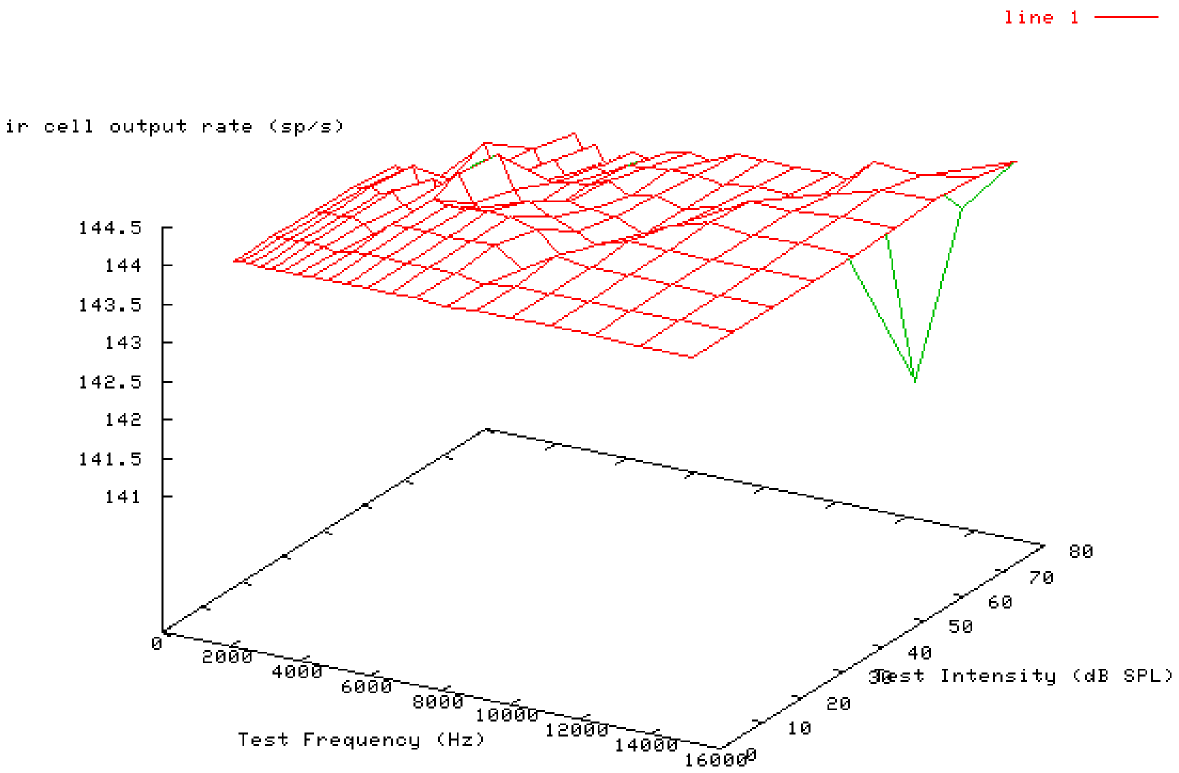
Figure 9.23. Meddis 86a AN Model: Two-component adaptation characteristics



Two-Tone Suppression Example: AutoTest/IHC/Meddis86a7.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Meddis86a7.sim
begin  {

 Stim_Puretone_Multi < MPTone2.par
 Trans_Gate       < Ramp1.par
 Filt_BandPass     < PEDRNL1.par
 BM_DRNL         < DRNL8k.par
 IHCRP_Shamma     < ShammaRP1.par
 IHC_Meddis86a     < Medd86aH.par
 reset hi
mr% repeat 1 {
 hi% Ana_Histogram    < PSTH5.par
 }

}
```

Figure 9.24. Meddis 86a AN Model: Two-tone suppression characteristics.



IHC_Meddis2000: Inner Hair Cell Synapse

File name: MoIHCMeddis2000.[ch]

Description

This is model of the hair cell synapse. The release of neurotransmitter into the synaptic cleft is mediated by calcium ions in all synapses. The transmitter release function is a three-part process.

Part I

Depolarisation of the IHC membrane leads to the opening of calcium ion channels. We employ a third-order process similar to Hudspeth and Lewis (1988) and Kidd and Weiss (1990), but with a time constant that is not voltage dependent. Calcium current (Ica) is a function of the RP

$$I_{Ca}(t) = G_{Ca}^{\max} m_{I_{Ca}}^{3}(t)(V(t) - E_{Ca}$$

where E_{Ca} is the reversal potential for calcium and $G_{Ca}{}^{max}$ is the calcium conductance in the vicinity of the synapse, with all the channels open and $m_{I_{Ca}}$(t) is the fraction of calcium channels that are open. The steady state value of the latter, $m_{I\ Ca,\infty}$, is modeled by a Boltzmann function

$$m_{I_{Ca},\infty}=[1+\beta_{Ca}^{-1}\exp(\gamma_{Ca}V(t))]^{-1}$$

where β_{Ca} and γ_{Ca} are constants chosen to reflect published observations of calcium currents, and $m_{I_{Ca}}$ (t) is a low pass filtered function of $m_{I\ Ca,\infty}$

$$m_{I_{Ca},\infty}=\tau_{I_{Ca}}\frac{dm_{I_{Ca}}(t)}{dt}+m_{I_{Ca}}(t)$$

where $\tau_{I_{Ca}}$ is a time constant.

Part II

Calcium ions enter the cell, and accumulate briefly in the vicinity of the synapse., calcium concentration $[Ca^{2+}](t)$ is modeled as a first-order low-pass filtered function of calcium current, $I_{Ca}(t)$ (after Hudspeth and Lewis (1988)).

$$I_{Ca}(t)=\tau_{[Ca]}\frac{d[Ca^{2+}]^{3}(t)}{dt}-[Ca^{2+}](t)$$

where τ_{Ca} is a time constant.

Part III

The probability of the release of transmitter is proportional to the cube of the Ca^{2+} concentration. We employ the function:

$$k(t)=\max\left\{\left([Ca^{2+}]^{3}(t)-[Ca^{2+}]_{thr}^{3}\right)_{z,0}\right\}$$

where $[Ca^{2+}]^{3}_{thr}$ is a threshold constant, z is a scalar for converting calcium concentration levels into release rate, and the cube function reflects data by Augustine et al. [1].

The transmitter release rate, k(t), drives a model of synaptic adaptation identical to that given in Lopez-Poveda et. al. (1998) [18]. It simulates the functional characteristics of adaptation, which are assumed here to be due to pre-synaptic transmitter depletion. The scheme is the same as that proposed by Meddis [20] except that release of transmitter into the cleft is now quantal and stochastic. It is described by the following eqns:

$$\frac{dq(t)}{dt}=N([w(t,x)]+N([M-q(t)]y)-N[(q(t),k(t)]$$

$$\frac{dc(t)}{dt}=N[q(t,k(t))]-lc(t)-rc(t)$$

$$\frac{dw(t)}{dt}=rc(t)-N[W(t),x]$$

Inputs	Aribrary single input, however, it expects input from an inner hair cell recptor potential process.
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	[32]

Module Parameters

Example 9.5. IHC_Meddis2000 Parameter File

OP_MODE	PROB	Output mode: stochastic ('spike') or probability ('prob').
DIAG_MODE	OFF	Diagnostic mode. Outputs internal states of running model in non-threaded mode('off', 'screen' or <file name>).
CA_COND_MODE	ORIGINAL	Calcium conductance mode ('original' or 'revision_1').
CLEFT_REPLENISH_MODE	ORIGINAL	Cleft replenishment mode ('original' or 'unity').
RAN_SEED	-1	Random number seed (0 for different seed for each run).
REV_POT_ECA	0.066	Calcium reversal potential, E_Ca (Volts).
BETA_CA	400	Calcium channel Boltzmann function parameter, beta.
GAMMA_CA	130	Calcium channel Boltzmann function parameter, gamma.
POWER_CA	3	Calcium channel transmitter release exponent (power).
GMAX_CA	8e-09	Maximum calcium conductance (Siemens).
CONC_THRESH_CA	4.48e-11	Calcium threshold Concentration.
PERM_Z	2e+32	Transmitter release permeability, Z (unitless gain)
TAU_M	0.0001	Calcium current time constant (s).
TAU_CA	0.0001	Calcium ion diffusion (accumulation) time constant (s).
MAX_FREE_POOL_M	10	Max. no. of transmitter packets in free pool (integer).
REPLENISH_Y	10	Replenishment rate (units per second).
LOSS_L	2580	Loss rate (units per second).
REPROCESS_X	66.31	Reprocessing rate (units per second).
RECOVERY_R	6580	Recovery rate (units per second).

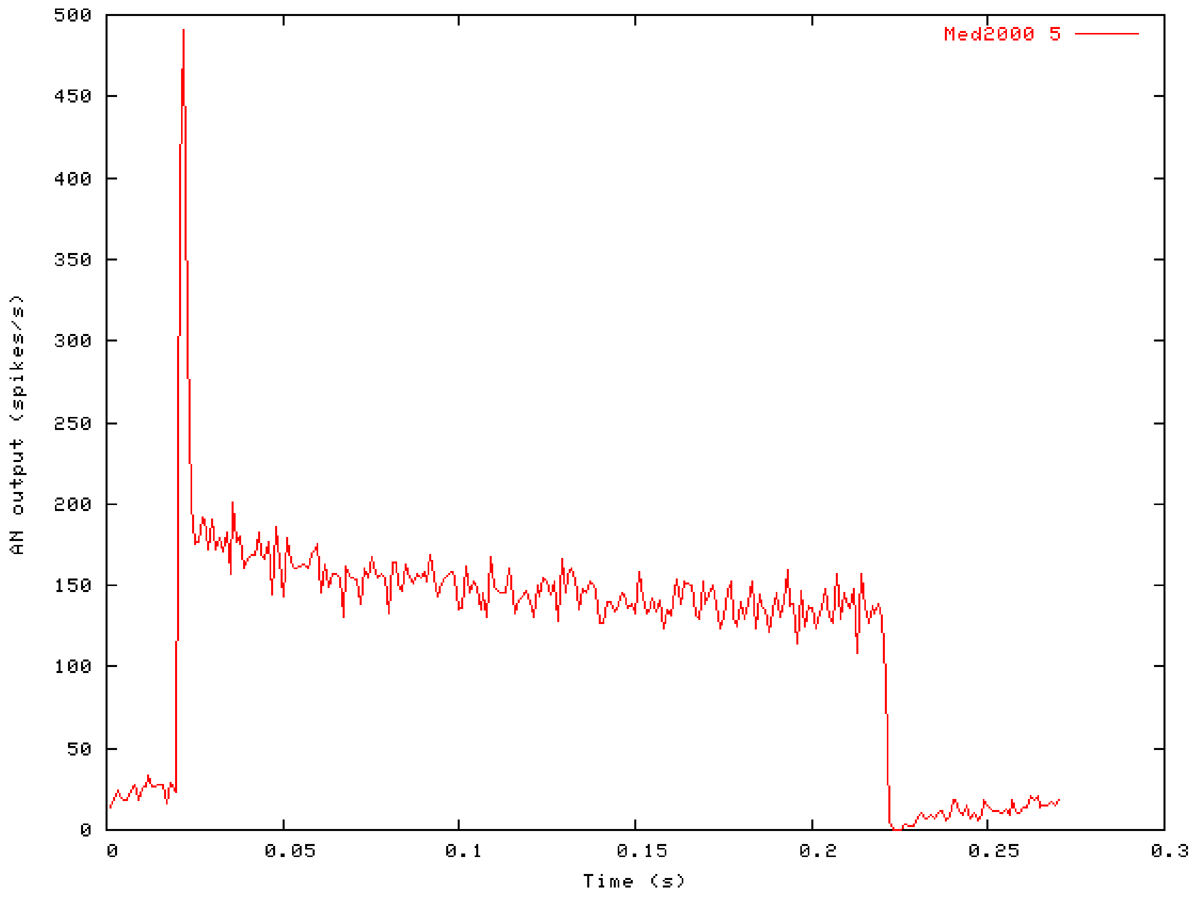
Examples Using the 'IHC_Meddis2000' Process Module

Some of the following tests require multiple observations using a range of stimuli and cannot be produced directly with a single run of the *AMS* program. These simulation scripts are run using the *AutoTest* program, which is employed to test the operation of the DSAM process modules.

Simple Response Example: AutoTest/IHC/Med2000_5.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Med2000_5.sim
begin  {

 Stim_Puretone_2  < P2Tone4.par
 Trans_gate      < Ramp1.par
 Filt_BandPass    < BPassGP2381.par
 BM_DRNL_Test    < DRNLT16_7k.par
 IHCRP_Shamma3StateVelIn < Shamma3sRP1.par
 IHC_Meddis2000   < Med2000HSRp1.par
 reset hi
 repeat 500 {
 AN_SG_Carney    < CarneyANSG1.par
 hi% Ana_Histogram    < PSTH4.par
 Display_Signal
 }

}
```

Figure 9.25. Post stimulus time histogram (PSTH) for the Meddis 2000 AN model



Phase-locking vs Frequency Example: AutoTest/IHC/Med2000_1.sim

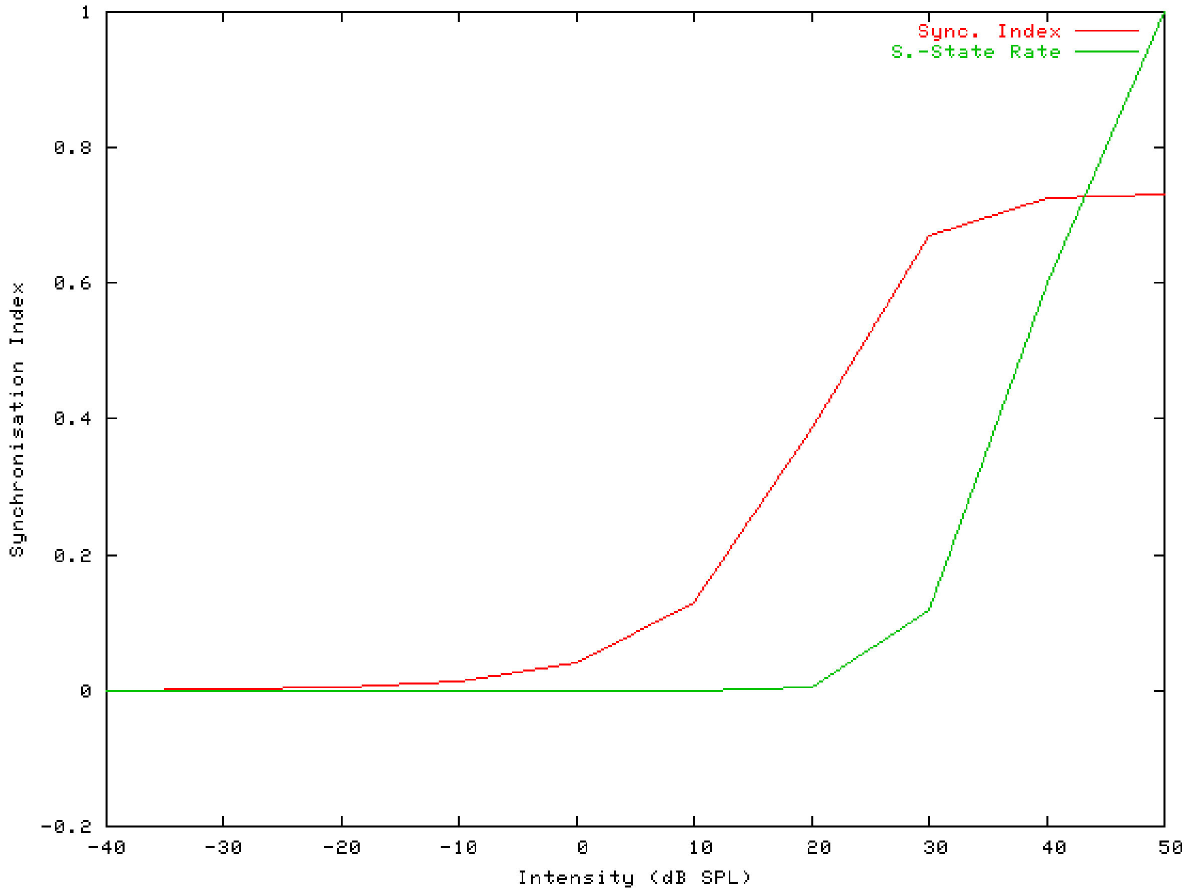
```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Med2000_1.sim
begin  {

 Stim_Puretone    < PTone3.par
 Trans_gate      < Ramp1.par
 Filt_BandPass    < BPassGP2381.par
 BM_DRNL_Test    < DRNLT16_7k.par
 IHCRP_Shamma3StateVelIn < Shamma3sRP1.par
 IHC_Meddis2000   < Med2000HSRp1.par
 Ana_Histogram    < PH1.par

}
```

See Figure 9.2, “Phase-locking vs frequency Firing characteristics for the DSAM AN models”.

Phase-locking vs Intensity Example: AutoTest/IHC/Med2000_2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Med2000_2.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < BPassGP4a.par
 Filt_BandPass    < BPassGP4b.par
 BM_DRNL        < DRNL_GP4_1k.par
 IHCRP_Shamma3StateVelIn < IHCRP_GP3.par
 IHC_Meddis2000   < Med2000HSRp1.par
 Display_Signal

}
```

Figure 9.26. Meddis 2000 AN Model: Firing characteristics vs intensity

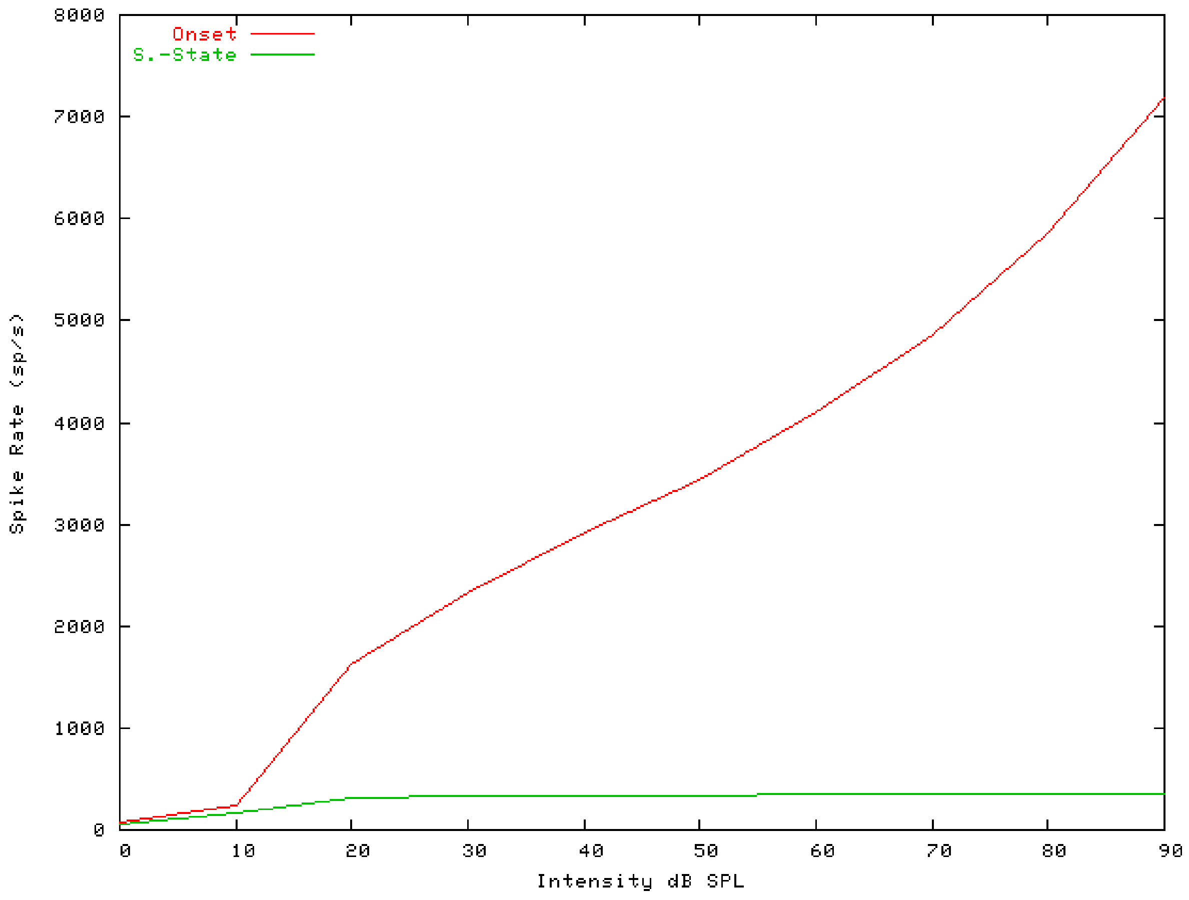


See also Figure 9.4, “Phase-locking vs Intensity vs intensity.Firing characteristics for the DSAM AN models”.

Rate-Intensity Functions (Spike Probabilities) Example: AutoTest/IHC/Med2000_3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Med2000_3.sim
begin  {

 Stim_Puretone    < PTone3.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < BPassGP2381.par
 BM_DRNL_Test     < DRNLT16_7k.par
 IHCRP_Shamma3StateVelIn < Shamma3sRP1.par
 IHC_Meddis2000   < Med2000HSRp1.par
 reset hi
mr% repeat 1 {
 hi% Ana_Histogram    < PSTH1.par
 }

}
```

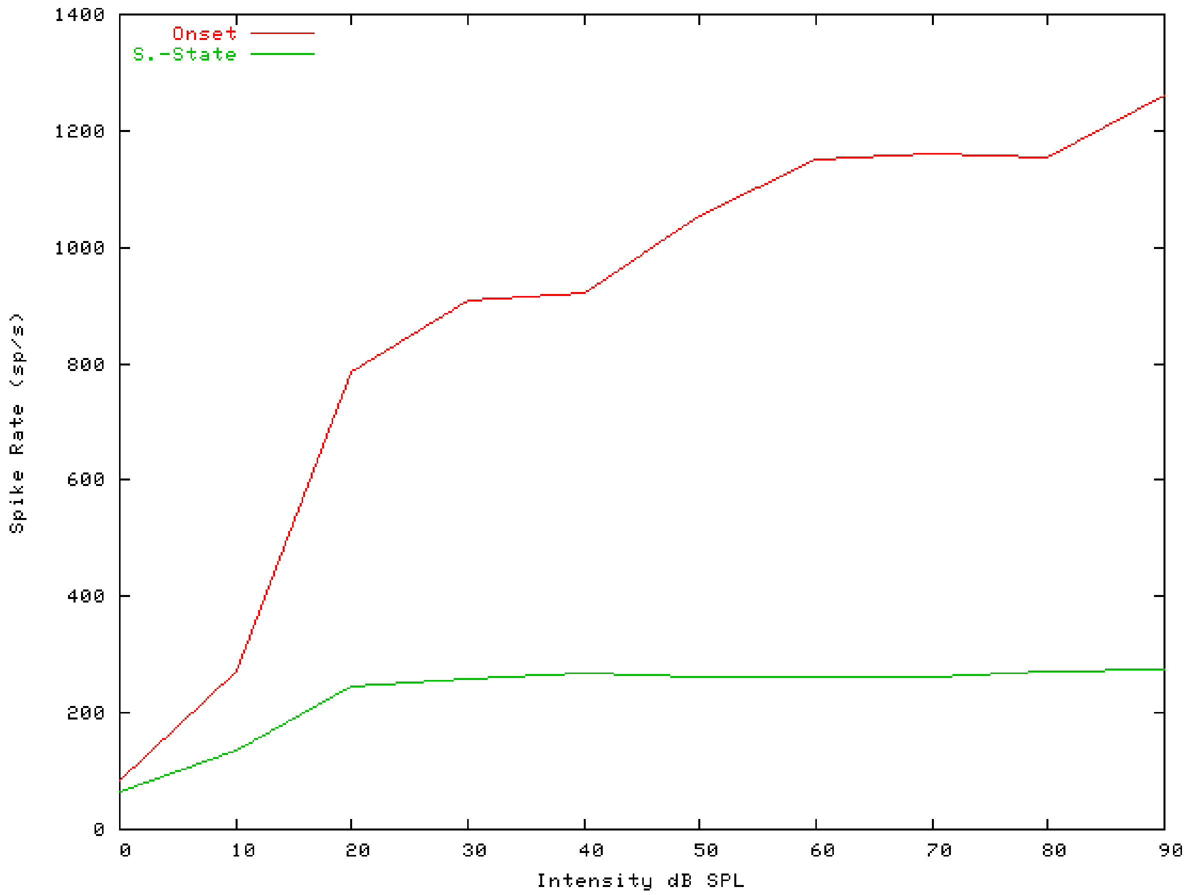
Figure 9.27. Meddis 2000 AN model: Rate-intensity functions



Rate-Intensity Functions (Multiple Runs) Example: AutoTest/IHC/Med2000_4.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Med2000_4.sim
begin  {

 Stim_Puretone    < PTone3.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < BPassGP2381.par
 BM_DRNL_Test    < DRNLT16_7k.par
 IHCRP_Shamma3StateVelIn < Shamma3sRP1.par
 IHC_Meddis2000   < Med2000HSRp1.par
 reset hi
mr% repeat 200 {
 AN_SG_Carney    < CarneyANSG1.par
 hi% Ana_Histogram    < PSTH2.par
 }

}
```

Figure 9.28. Meddis 2000 AN model: Rate-intensity functions Rate-intensity functions for the Meddis 2000_ AN model. Note that the main parameter file was changed to produce this diagram: the *AN_SG_Meddis2000_* module was put in (using the ’Meddis2000_1SG.par’ module parameter file, the ’Histogm2.par’ module parameter file was replaced by’Histogm4.par’ and the number of model runs was set to 200.

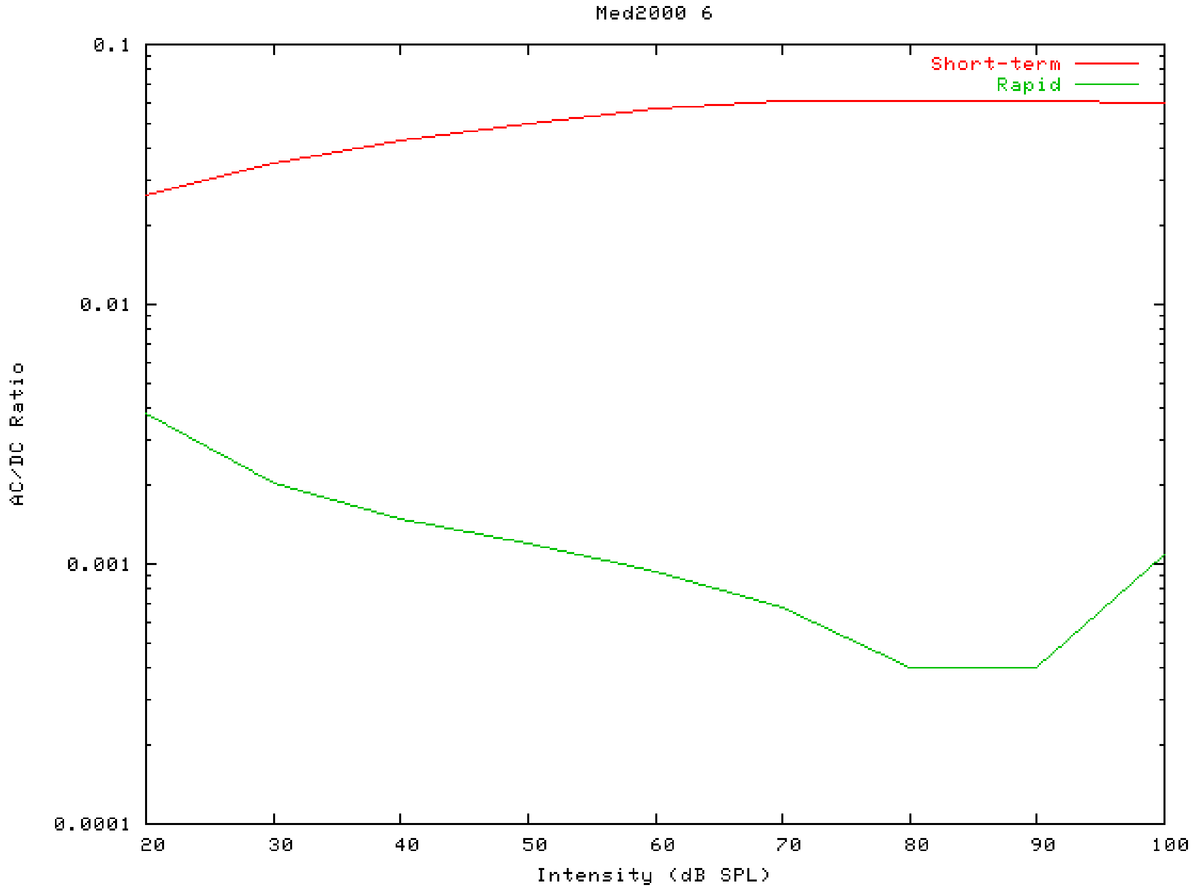


See also Figure 9.7, “Steady-state firing rate vs intensity comparisonFiring characteristics for the DSAM AN models”.

Two-component Adaptation Example: AutoTest/IHC/Med2000_6.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Med2000_6.sim
begin {

 Stim_Puretone    < PTone3.par
 Trans_gate      < Ramp1.par
 Filt_BandPass    < BPassGP2381.par
 BM_DRNL_Test    < DRNLT16_7k.par
 IHCRP_Shamma3StateVelIn < Shamma3sRP1.par
 IHC_Meddis2000   < Med2000HSRp1.par
 reset hi
hi% Ana_Histogram   < PSTH5.par
 Display_Signal

}
```

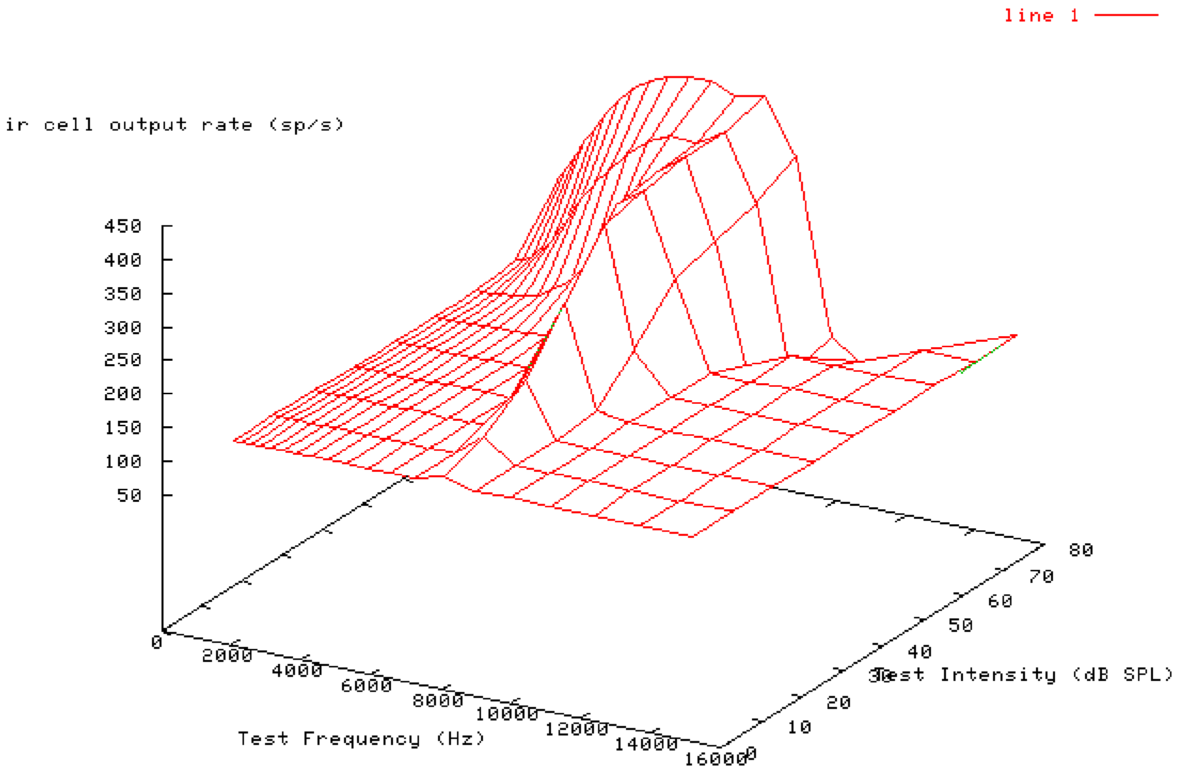
Figure 9.29. Meddis 2000 AN Model: Two-component adaptation characteristics



Two-Tone Suppression Example: AutoTest/IHC/Med2000_7.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Med2000_7.sim
begin  {

 Stim_Puretone_Multi < MPTone2.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < BPassGP4a.par
 Filt_BandPass    < BPassGP4b.par
 BM_DRNL        < DRNL_GP4_8k.par
 IHCRP_Shamma3StateVelIn < IHCRP_GP3.par
 reset hi
mr% repeat 1 {
 IHC_Meddis2000   < Med2000HSRp1.par
 hi% Ana_Histogram    < PSTH1.par
 }

}
```

Figure 9.30. Meddis 2000 AN Model: Two-tone suppression characteristics.



IHC_Zhang: Inner Hair Cell Synapse (2001)

File name: MoIHCZhang.[ch]

Description

This is the Zhang et al [34] inner hair cell (IHC) synapse. It was revised from C code provided by the authors (the ’ARLO’ code). It requires an IHC receptor potential process module as input. It outputs spike probabilities, and was designed to be used with the ’AN_SG_Carney’ spike generation module.

Inputs	Arbitrary single input, however, it expects input from an inner hair cell receptor potential process.
Outputs	Produces an output signal of spike probabilities, with the same dimensions as the input signal.
Reference	[34]

Module Parameters

Example 9.6. IHC_Zhang Parameter File

SPONT	50	Spontaneous rate of fiber (spikes/s).
A_SS	130	Steady-state rate (spikes/s).
TAU_ST	0.06	Short-term time constant (s).
TAU_R	0.002	Rapid time constant (s).
AR_AST	6	Rapid response amplitude to short-term response amplitude ratio.
PI_MAX	0.6	Permeability at high sound level.
K_S	1.3	Species dependend voltage staturation parameter.
V_S1	60	Species dependend voltage staturation parameter.
V_S2	1	Species dependend voltage staturation parameter.
V_S3	6	Species dependend voltage staturation parameter.

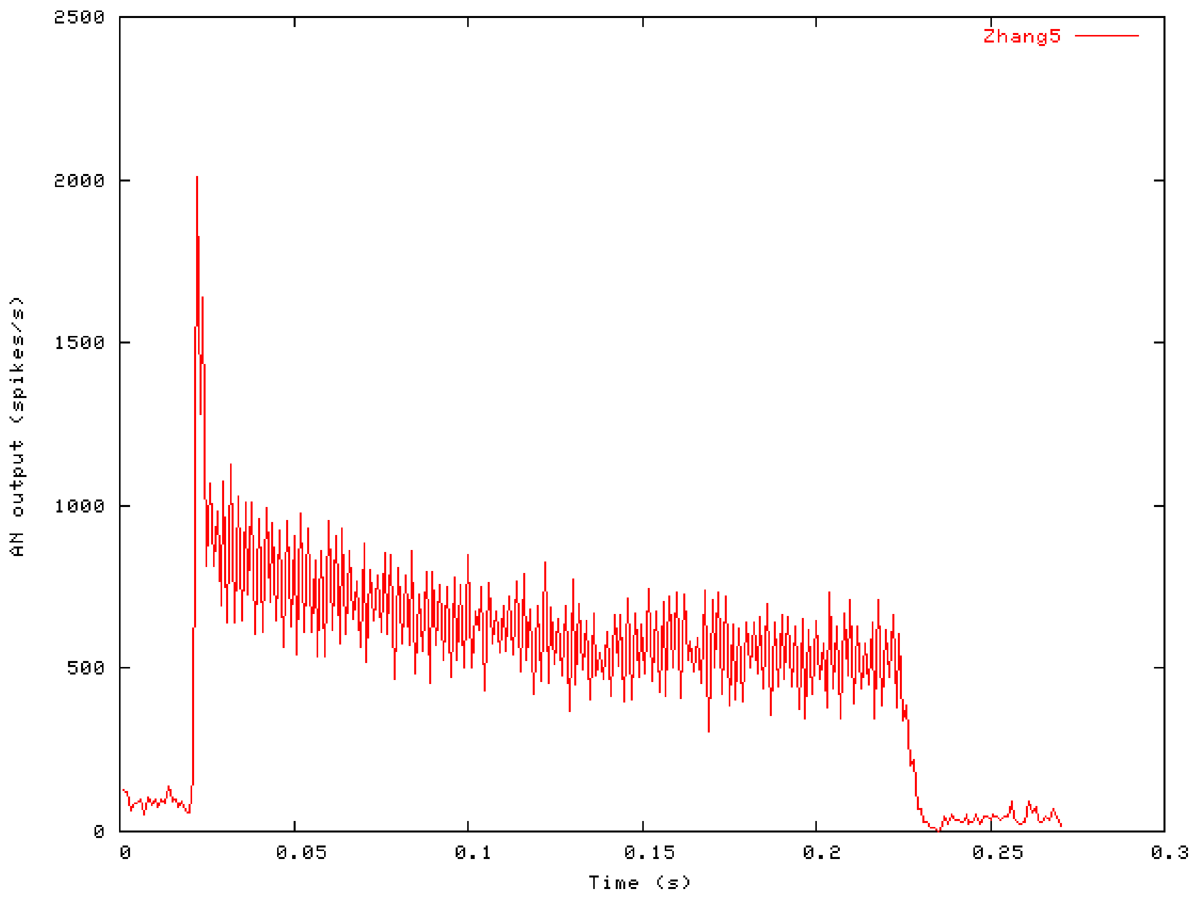
Examples Using the ’IHC_Zhang’ Process Module

Some of the following tests require multiple observations using a range of stimuli and cannot be produced directly with a single run of the *AMS* program. These simulation scripts are run using the *AutoTest* program, which is employed to test the operation of the DSAM process modules.

Simple Response Example: AutoTest/IHC/Zhang5.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Zhang5.sim
begin {

 Stim_Puretone_2  < P2Tone5.par
 Trans_Gate     < Ramp1.par
 Filt_BandPass   < PEHuY1.par
 BM_Zhang       < Zhang1_5k.par
 IHCRP_Zhang     < ZhangRPCat1.par
 IHC_Zhang      < ZhangIHCCat1.par
 AN_SG_Carney    < Carney3SG.par
 Ana_Histogram   < PSTH3.par
 Display_Signal

}
```

Figure 9.31. Post stimulus time histogram (PSTH) for the Zhang AN model



Phase-locking vs Frequency Example: AutoTest/IHC/Zhang1.sim

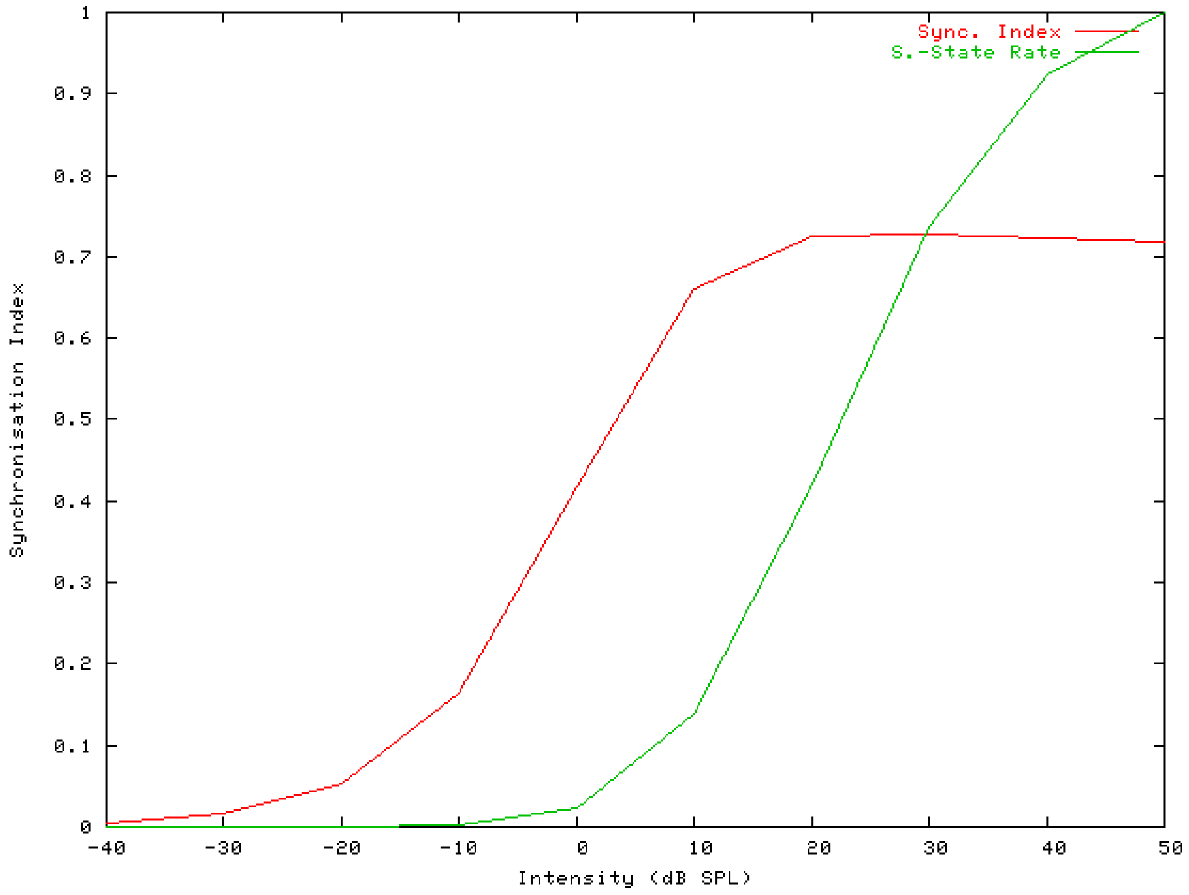
```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Zhang1.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY1.par
 BM_Zhang       < Zhang1k.par
 IHCRP_Zhang      < ZhangRPCat1.par
 IHC_Zhang       < ZhangIHCCat1.par
 Ana_Histogram    < PH1.par

}
```

See Figure 9.2, “Phase-locking vs frequency Firing characteristics for the DSAM AN models”.

Phase-locking vs Intensity Example: AutoTest/IHC/Zhang2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Zhang2.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY1.par
 BM_Zhang       < Zhang1k.par
 IHCRP_Zhang     < ZhangRPCat1.par
 IHC_Zhang      < ZhangIHCCat1.par

}
```

Figure 9.32. Zhang AN Model: Firing characteristics vs intensity

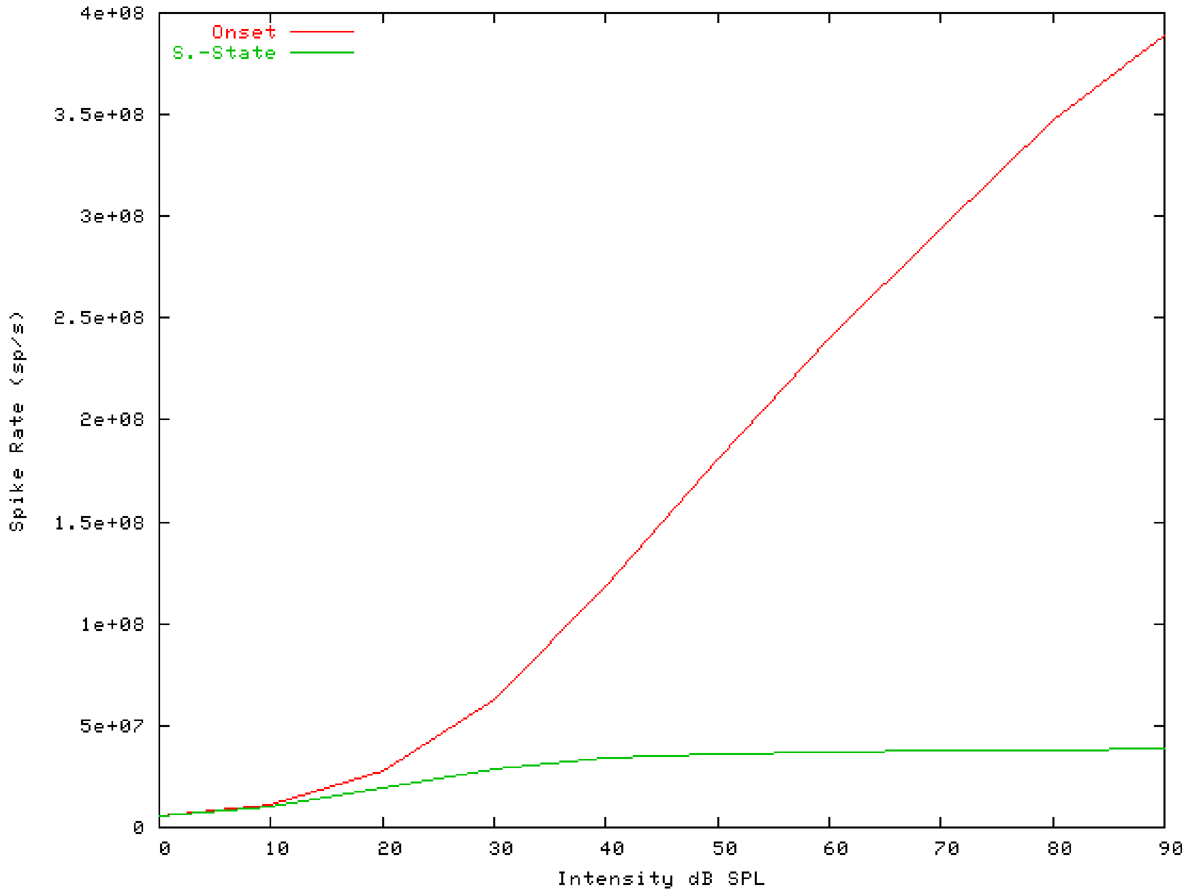


See also Figure 9.4, “Phase-locking vs Intensity vs intensity.Firing characteristics for the DSAM AN models”.

Rate-Intensity Functions (Spike Probabilities) Example: AutoTest/IHC/Zhang3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Zhang3.sim
begin  {

 Stim_Puretone    < PTone5.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY1.par
 BM_Zhang        < Zhang1k.par
 IHCRP_Zhang     < ZhangRPCat1.par
 IHC_Zhang       < ZhangIHCCat1.par
 reset hi
mr% repeat 1 {
 hi% Ana_Histogram    < PSTH1.par
 }

}
```

Figure 9.33. Zhang AN model: Rate-intensity functions

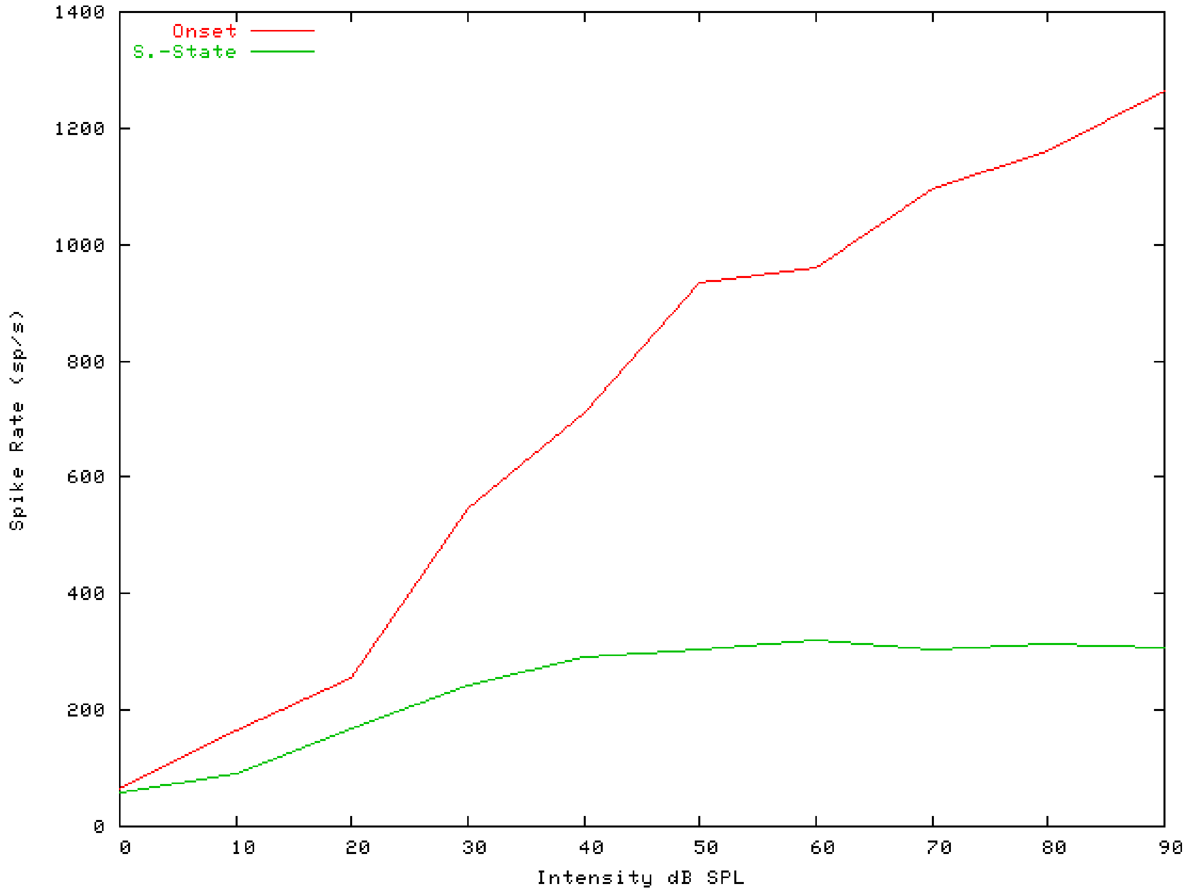


See also Figure 9.7, “Steady-state firing rate vs intensity comparisonFiring characteristics for the DSAM AN models”.

Rate-Intensity Functions (Multiple Runs) Example: AutoTest/IHC/Zhang4.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Zhang4.sim
begin  {

 Stim_Puretone    < PTone5.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY1.par
 BM_Zhang       < Zhang1k.par
 IHCRP_Zhang     < ZhangRPCat1.par
 IHC_Zhang      < ZhangIHCCat1.par
 Display_Signal
 reset hi
mr% repeat 200 {
 AN_SG_Carney    < Carney4SG.par
 hi% Ana_Histogram    < PSTH2.par
 }

}
```

Figure 9.34. Zhang AN model: Rate-intensity functions Rate-intensity functions for the Zhang AN model. Note that the main parameter file was changed to produce this diagram: the *AN_SG_Zhang* module was put in (using the ’Zhang1SG.par’ module parameter file, the ’Histogm2.par’ module parameter file was replaced by’Histogm4.par’ and the number of model runs was set to 200.

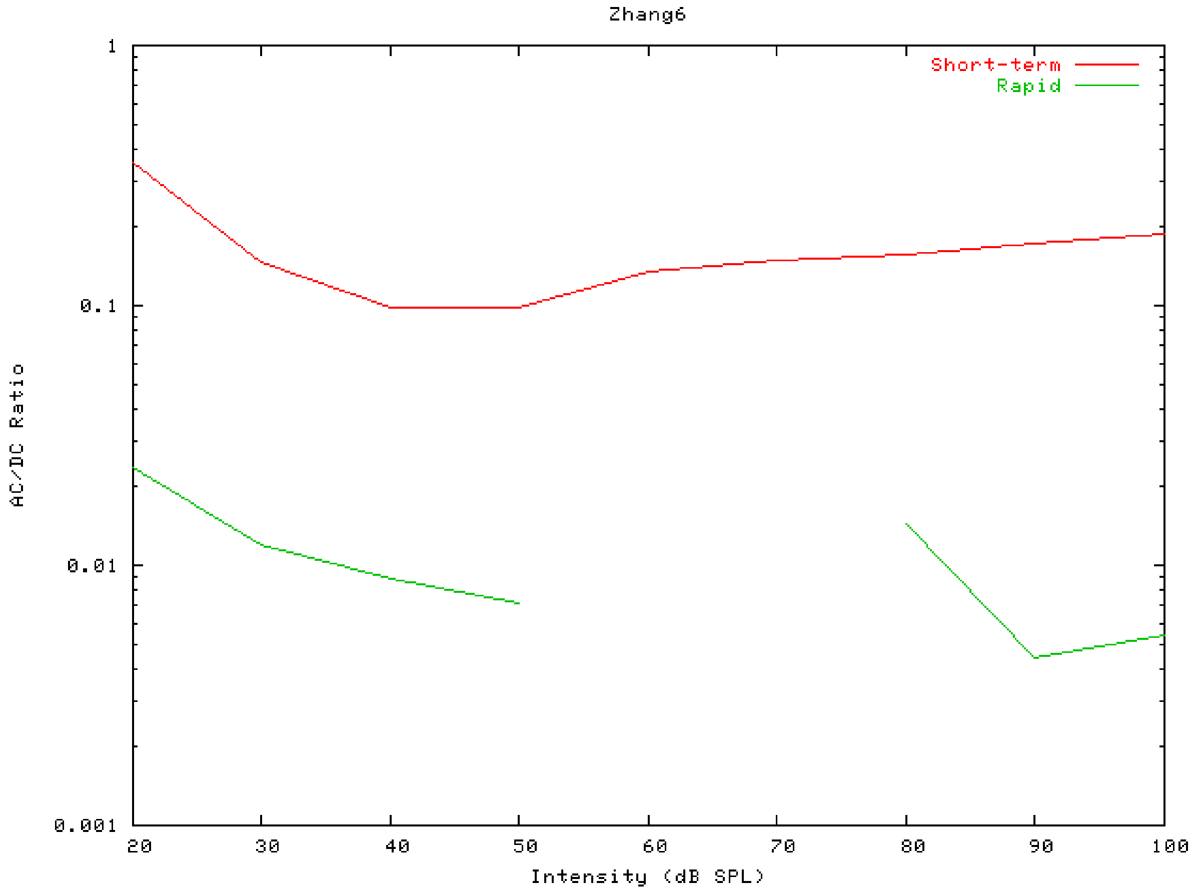


See also Figure 9.7, “Steady-state firing rate vs intensity comparisonFiring characteristics for the DSAM AN models”.

Two-component Adaptation Example: AutoTest/IHC/Zhang6.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Zhang6.sim
begin  {

 Stim_Puretone    < PTone4.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY1.par
 BM_Zhang        < Zhang1k.par
 IHCRP_Zhang     < ZhangRPCat1.par
 IHC_Zhang       < ZhangIHCCat1.par
 reset hi
hi% Ana_Histogram    < PSTH5.par

}
```

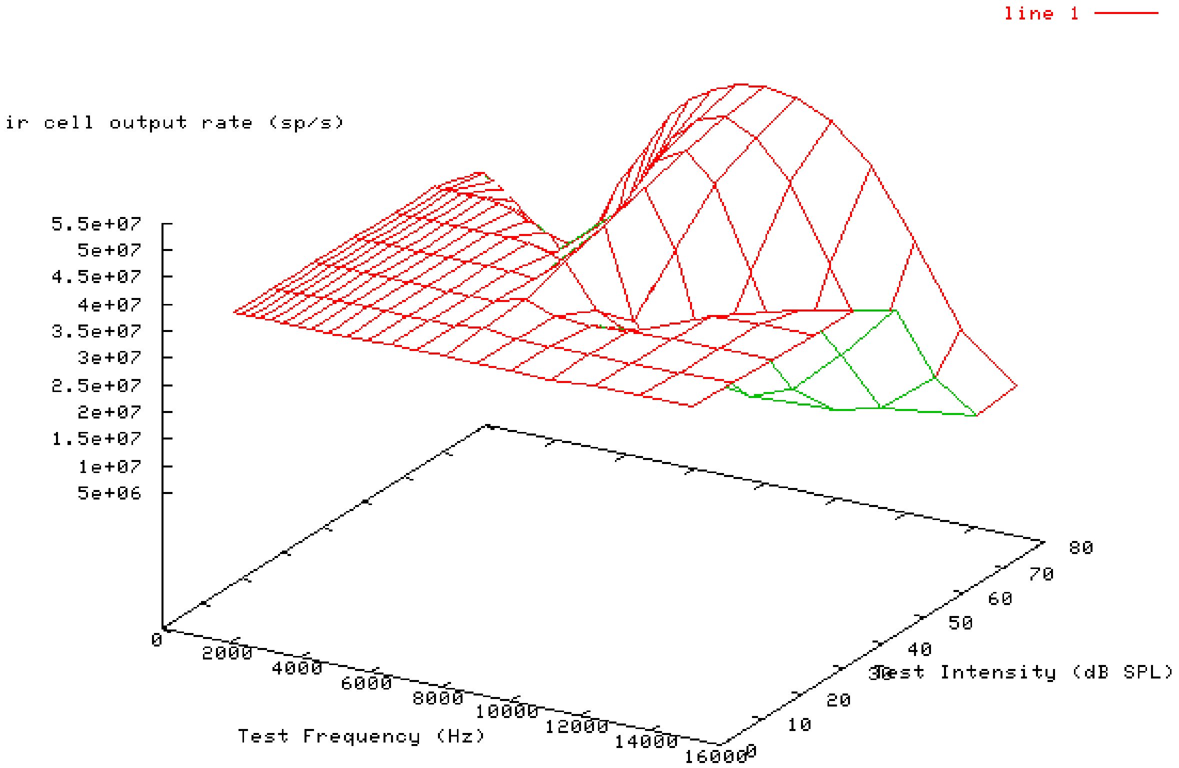
Figure 9.35. Zhang AN Model: Two-component adaptation characteristics



Two-Tone Suppression Example: AutoTest/IHC/Zhang7.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./IHC/Zhang7.sim
begin  {

 Stim_Puretone_Multi < MPTone3.par
 Trans_Gate       < Ramp1.par
 BM_Zhang        < Zhang8k.par
 IHCRP_Zhang      < ZhangRPCat1.par
 IHC_Zhang       < ZhangIHCCat1.par
 reset hi
mr% repeat 1 {
 hi% Ana_Histogram    < PSTH5.par
 }

}
```

Figure 9.36. Zhang AN Model: Two-tone suppression characteristics.



IHCRP_Carney: Inner Hair Cell Receptor Potential (1993)

MoHCRPCarney.[ch]

Description

Laurel H. Carney IHC receptor potential module [5]. It supplies input to the inner hair cell synapse models, e.g. IHC_Carney, IHC_Meddis2000 , IHC_Zhang or IHC_Meddis86a .

Inputs	Arbitrary single process. Usually a basilar membrane process supplies input to this process..
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	[5]

Module Parameters

Example 9.7. HCRP_Carney Parameter File

FC	1100	Cut-off frequency for OHC low-pass filter, Fc (Hz).
P_D_IHC	1000	Operating point of OHC (feedback) non-linearity, P_Dihc (uPa).
P0	0.462	Asymmetrical bias for OHC non-linearity, P0 (rad)
V_MAX	10	Maximum depolarising hair cell voltage, Vmax (V).
A_D	0.00813	Travelling wave delay coefficient, A_D (s).
A_L	0.00649	Travelling wave delay length constant, A_L (m).
V_REF	0	Reference potential (V).

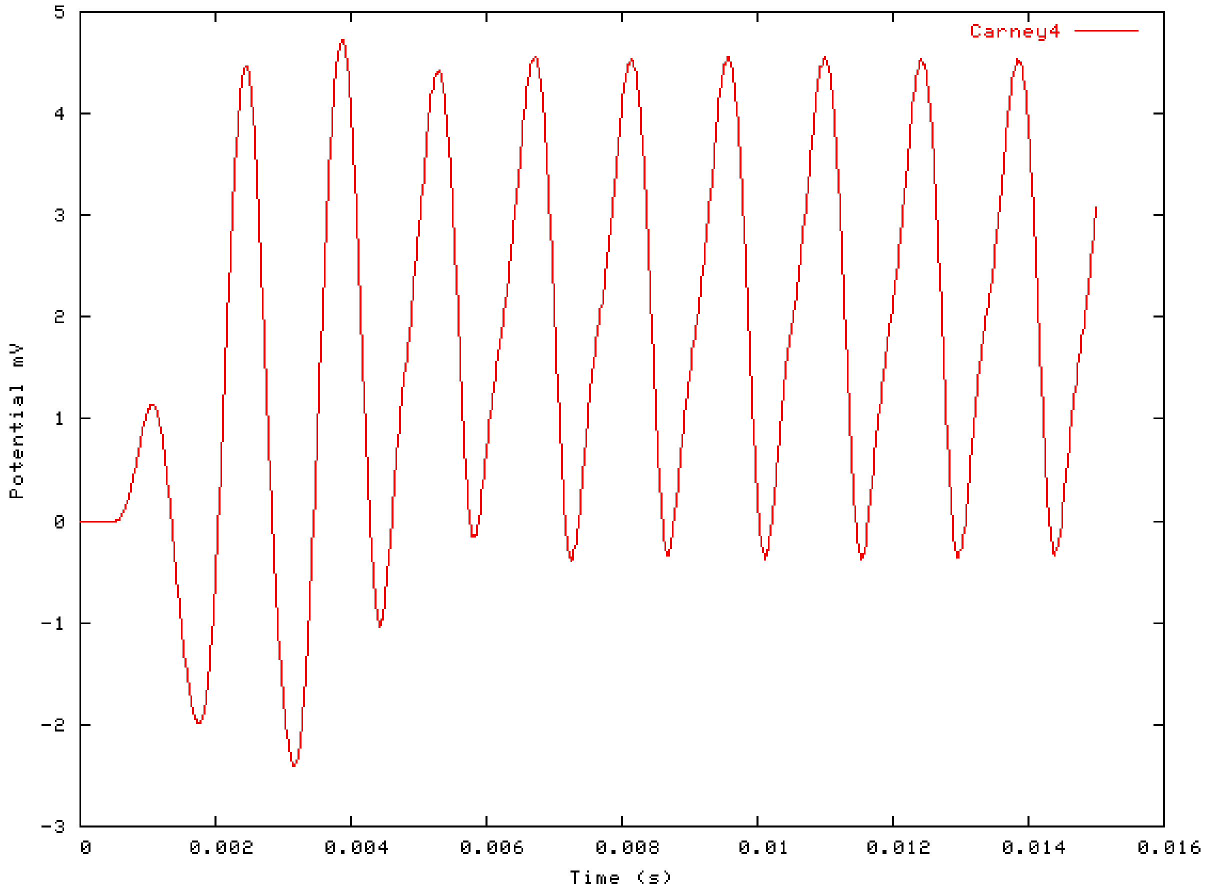
Examples Using the ’IHCRP_Carney’ Process Module

Some of the following tests require multiple observations using a range of stimuli and cannot be produced directly with a single run of the *AMS* program. These simulation scripts are run using the *AutoTest* program, which is employed to test the operation of the DSAM process modules.

Simple Response Example: AutoTest/HCRP/Carney4.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Carney4.sim
begin  {

 Stim_Puretone    < PTone3.par
 Trans_Gate      < Ramp1.par
 BM_Carney      < Carney18k.par
 IHCRP_Carney    < CarneyRP1.par
 Display_Signal

}
```

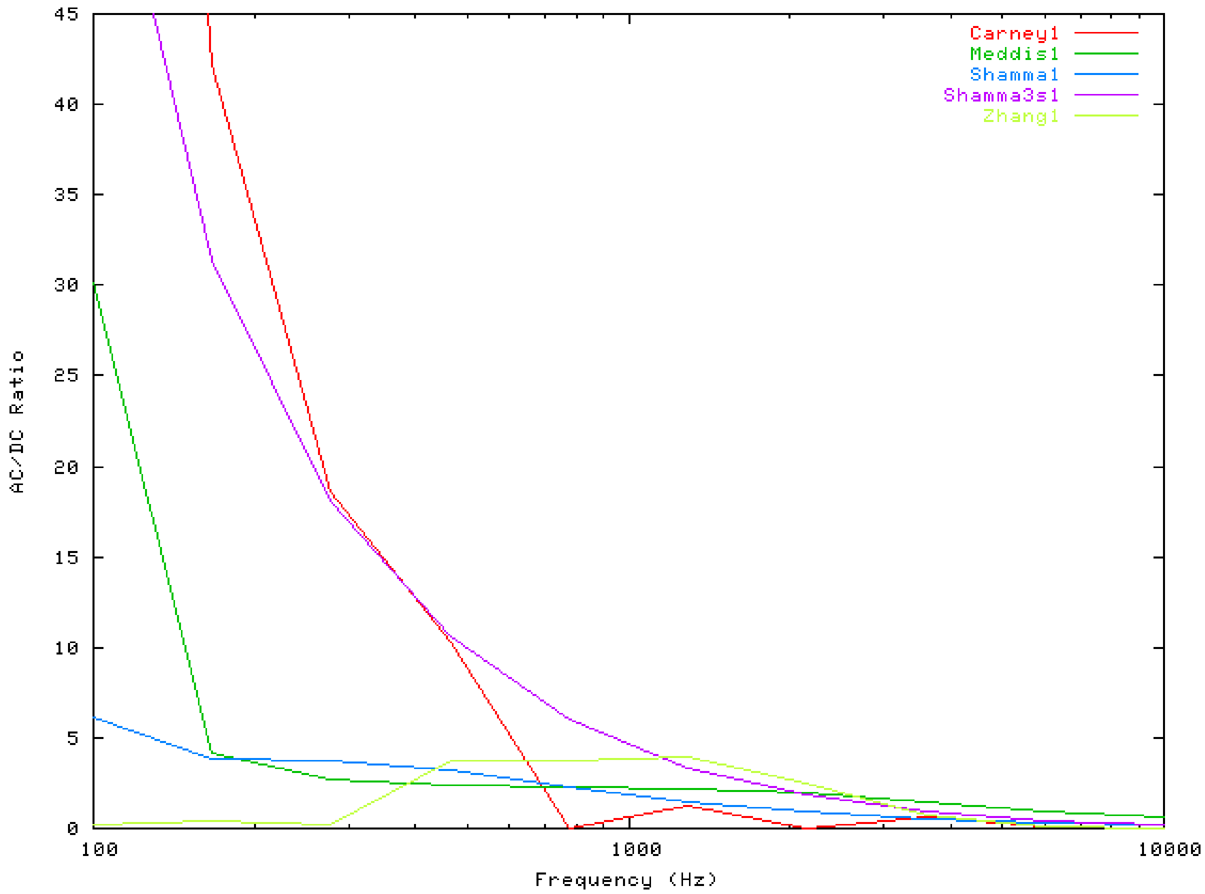
Figure 9.37. Pure tone response of the Carney hair cell receptor process module.



AC/DC Ratio vs Frequency Example: AutoTest/HCRP/Carney1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Carney1.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < ../BM/PreEmph1.par
 BM_Carney       < Carney18k.par
 IHCRP_Carney     < CarneyRP1.par
 @ Display_Signal

}
```

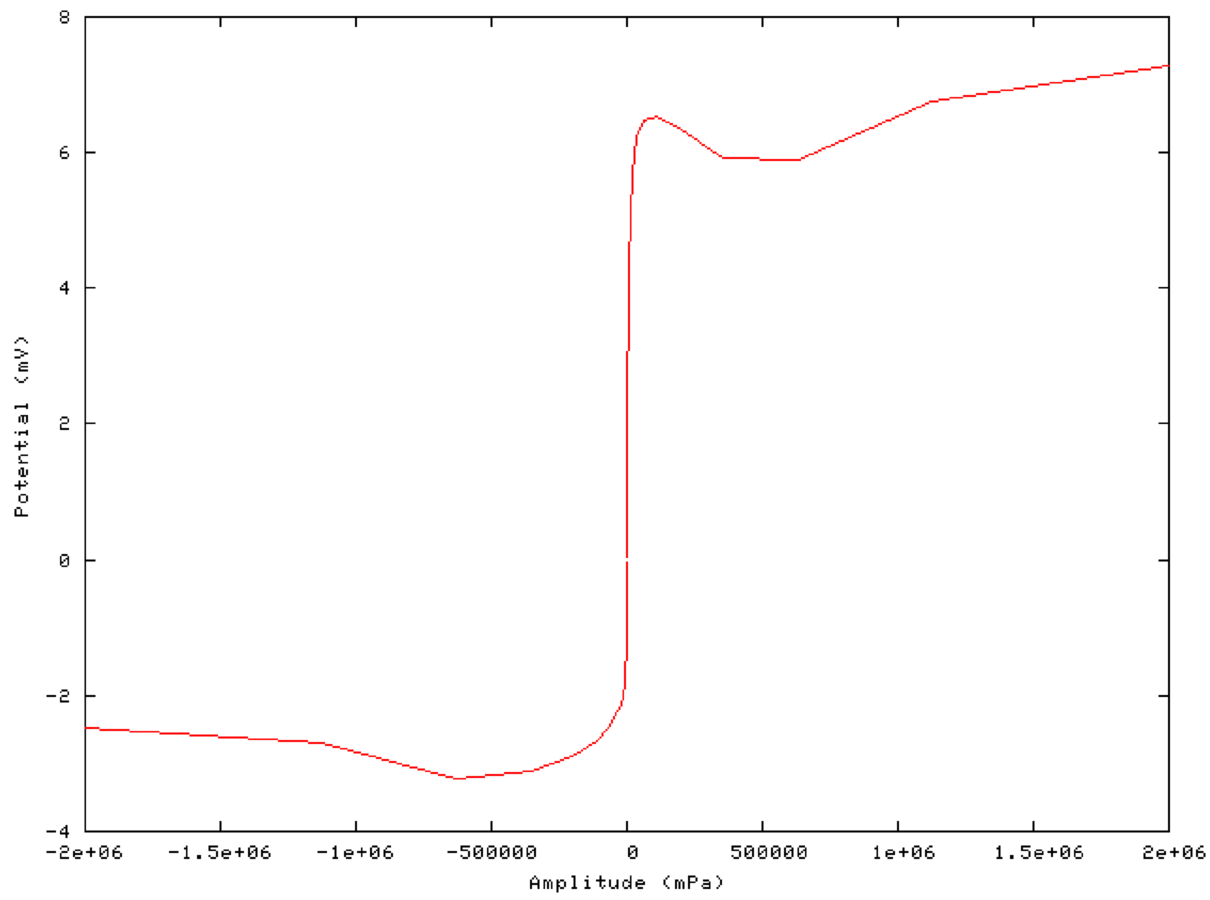
Figure 9.38. AC/DC frequency dependent characteristics for the DSAM IHCRP models.



Peak/Troughs function Example: AutoTest/HCRP/Carney2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Carney2.sim
begin  {

 Stim_Puretone    < PTone2.par
 Trans_Gate      < Ramp1.par
 BM_Carney       < Carney700.par
 IHCRP_Carney     < CarneyRP1.par

}
```

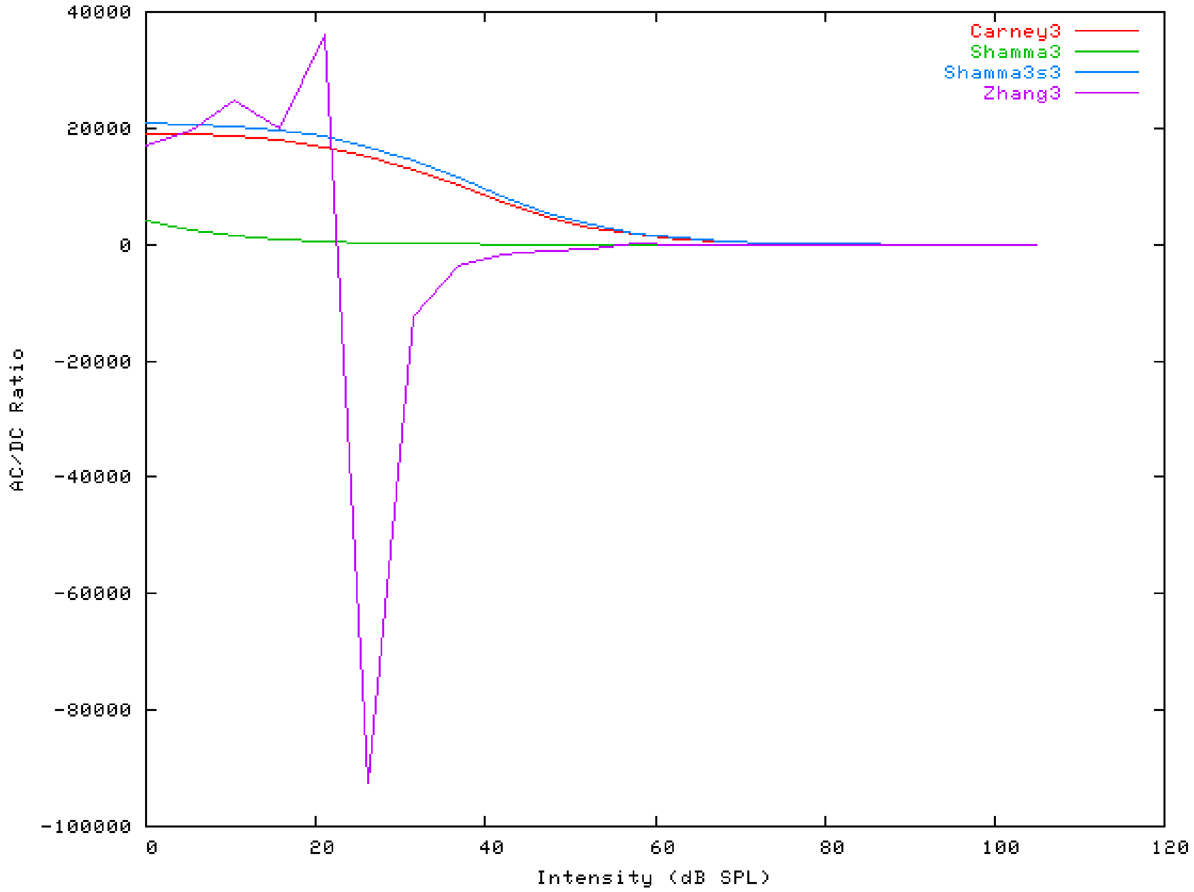
Figure 9.39. Peak-Trough characteristics for the Carney IHCRP model



AC/DC Ratio vs Intensity Example: AutoTest/HCRP/Carney3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Carney3.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 BM_Carney       < Carney18k.par
 IHCRP_Carney     < CarneyRP1.par

}
```

Figure 9.40. AC/DC intensity dependent characteristics for the DSAM IHCRP models.



IHCRP_Meddis: Inner Hair Cell Receptor Potential

MoHCRPMeddis.[ch]

Description

This is the Meddis receptor potential module. It supplies input to the inner hair cell synapse models, e.g. IHC_Meddis2000 , IHC_Zhang, IHC_Carney or IHC_Meddis86a .

Inputs	Arbitrary single process. Usually a basilar membrane process supplies input to this process..
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	

Module Parameters

Example 9.8. HCRP_Meddis Parameter File

```
PERM_CONST_A     100       Permeability constant A (units/s).
PERM_CONST_B     6000      Permeability constant B (units/s).
RELEASE_G        2000      Release rate (units/s).
TIME_CONST_TM    0.0001    Receptor potential time constant for IHC model (s).
```

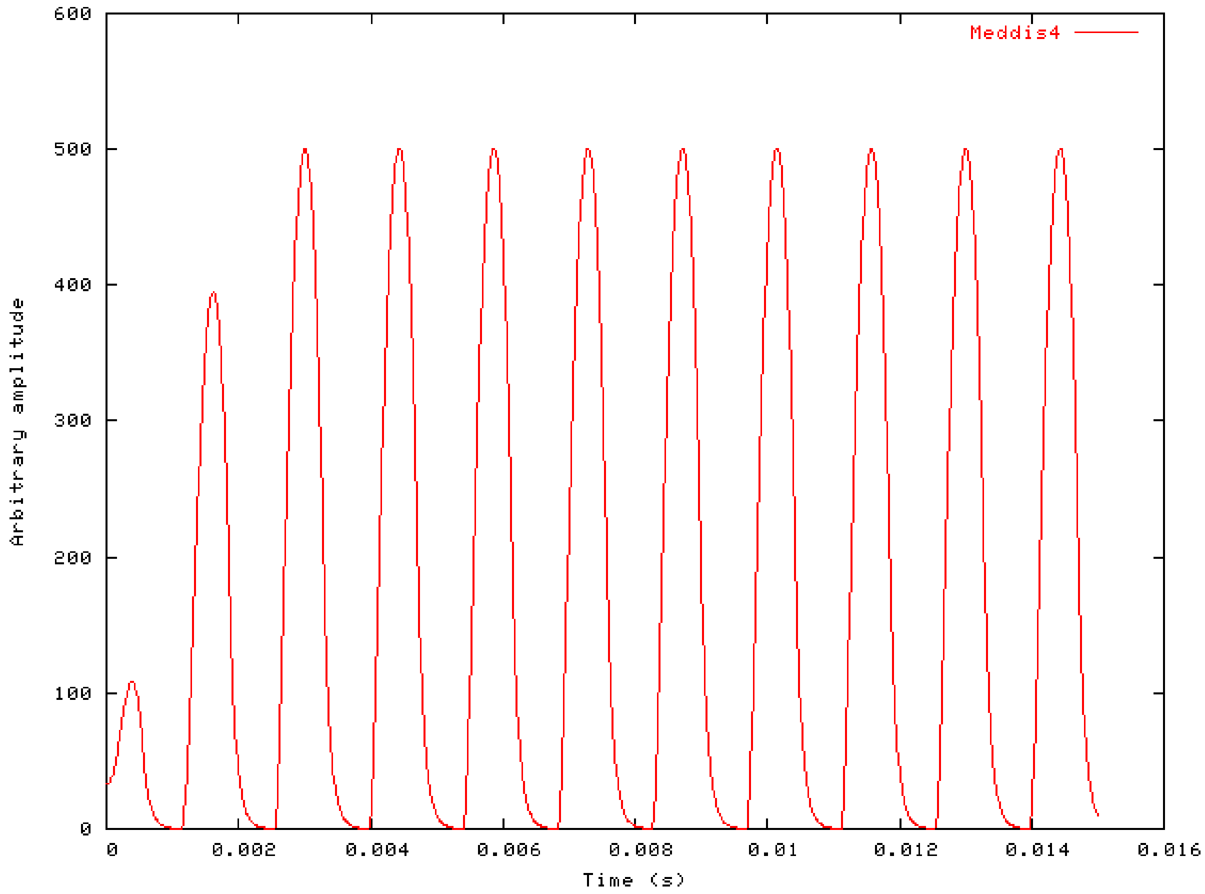
Examples Using the ’IHCRP_Meddis’ Process Module

Some of the following tests require multiple observations using a range of stimuli and cannot be produced directly with a single run of the *AMS* program. These simulation scripts are run using the *AutoTest* program, which is employed to test the operation of the DSAM process modules.

Simple Response Example: AutoTest/HCRP/Meddis4.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Meddis4.sim
begin  {

 Stim_Puretone    < PTone3.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_GammaT       < GammaT18k.par
 IHCRP_Meddis     < MeddisRP1.par
 @ Display_Signal

}
```

Figure 9.41. Pure tone response of the Meddis hair cell receptor process module.



AC/DC Ratio vs Frequency Example: AutoTest/HCRP/Meddis1.sim

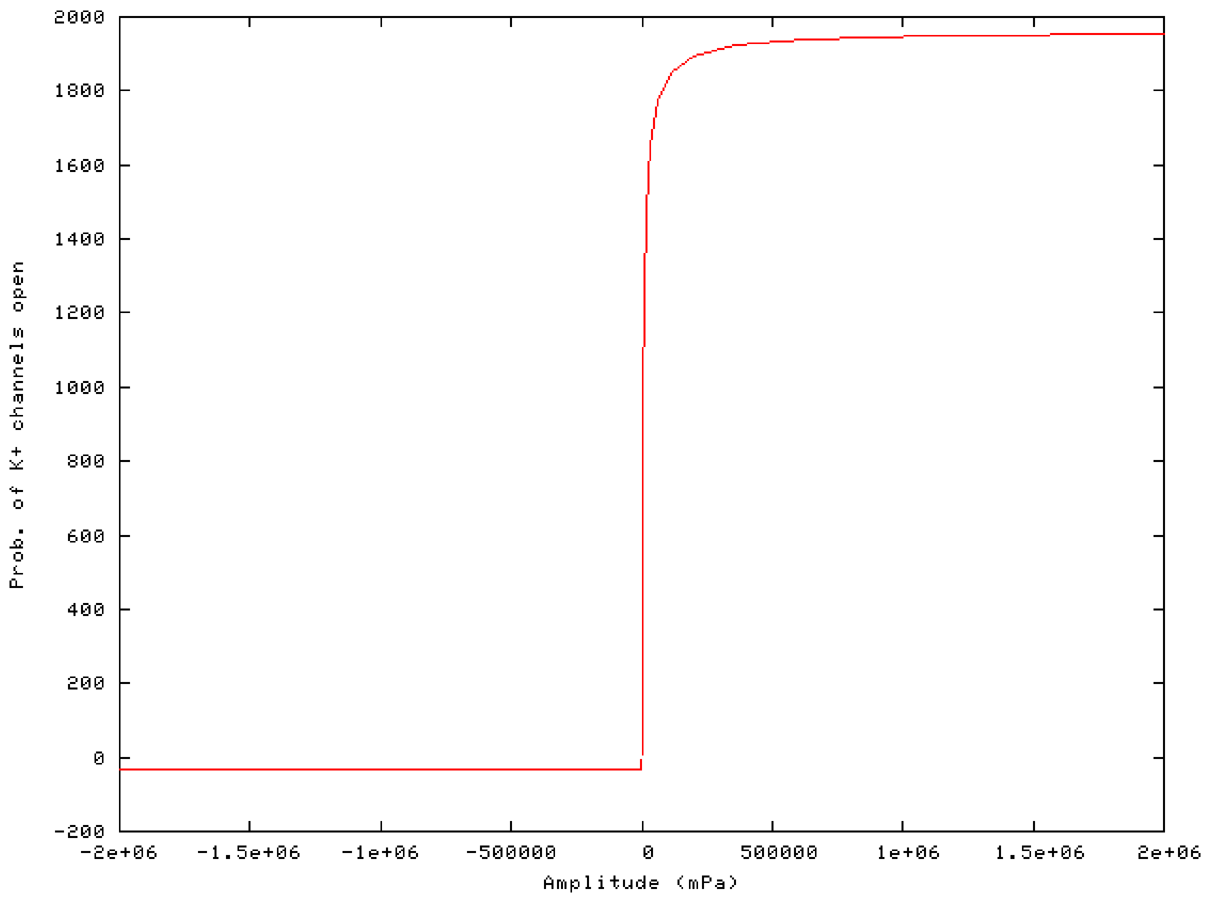
```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Meddis1.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_GammaT       < GammaT18k.par
 IHCRP_Meddis    < MeddisRP1.par
 @ Display_Signal

}
```

See Figure 9.38, “AC/DC frequency dependent characteristics for the DSAM IHCRP models.”.

Peak/Troughs function Example: AutoTest/HCRP/Meddis2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Meddis2.sim
begin  {

 Stim_Puretone    < PTone2.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_GammaT       < GammaT700.par
 IHCRP_Meddis    < MeddisRP1.par

}
```

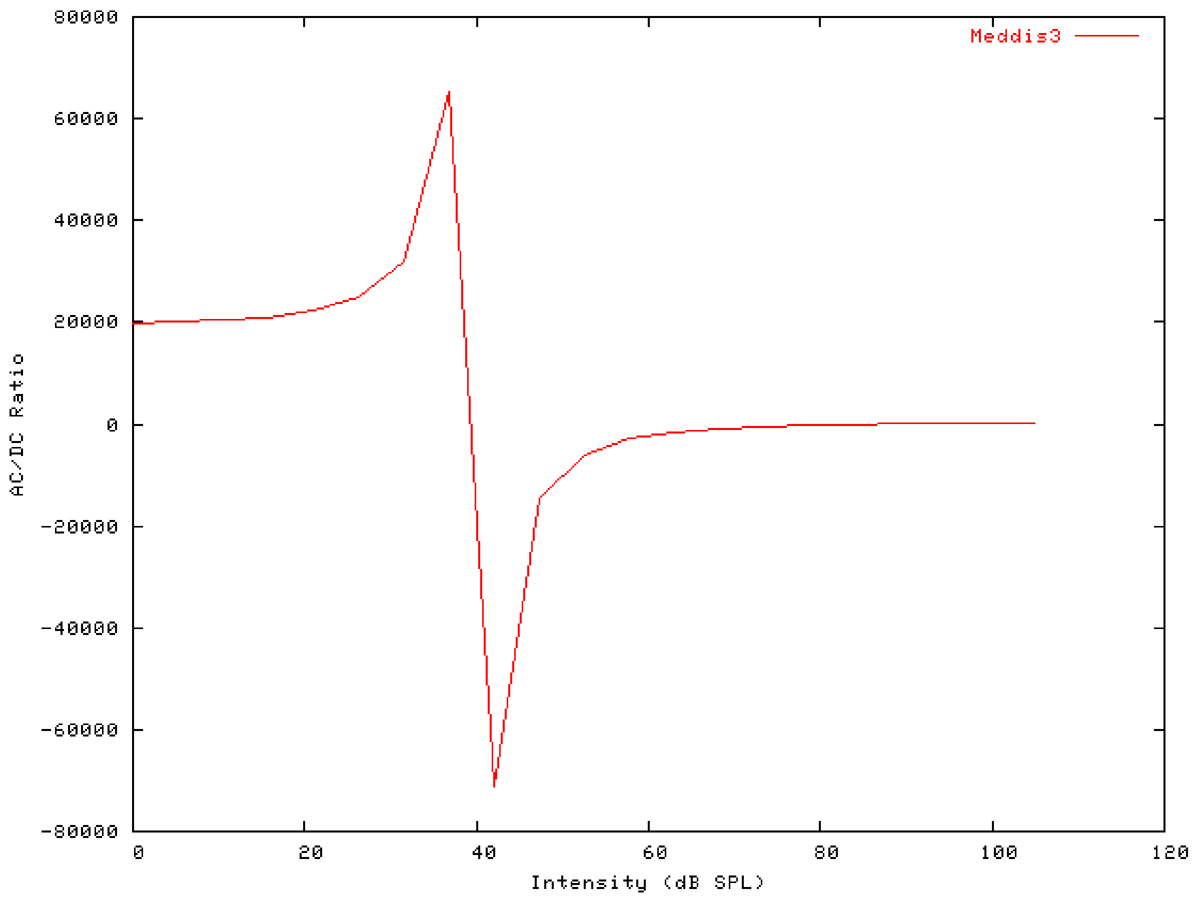
Figure 9.42. Peak-Trough characteristics for the Meddis IHCRP model



AC/DC Ratio vs Intensity Example: AutoTest/HCRP/Meddis3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Meddis3.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_GammaT       < GammaT18k.par
 IHCRP_Meddis     < MeddisRP1.par

}
```

Figure 9.43. AC/DC intensity dependent characteristics for the DSAM IHCRP models.



IHCRP_Shamma: Inner Hair Cell Receptor Potential (1986)

MoHCRPCarney.[ch]

Description

This module contains the model for the Shamma hair cell receptor potential [31]. It supplies input to the inner hair cell synapse models, e.g. IHC_Meddis2000 , IHC_Zhang, IHC_Carney or IHC_Meddis86a .

Inputs	Arbitrary single process. Usually a basilar membrane process supplies input to this process..
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	[31]

Module Parameters

Example 9.9. HCRP_Shamma Parameter File

E_T	0.1	Endocochlear potential, Et (V).
E_K	-0.084	Reversal potential, Ek (V).
RP_CORRECTION	0.04	Reversal potential correction, Rp/(Rt+Rp).
C_TOTAL	6.21e-12	Total capacitance, C = Ca + Cb (F).
G0	4.3e-09	Resting conductance, G0 (S).
G_K	1.07e-08	Potassium conductance, Gk (S = Siemens).
G_MAXC	4.18e-09	Maximum mechanical conductance, Gmax (S).
BETA	0.25	Beta constant, exp(-G1/RT).
GAMMA	1e+07	Gamma constant, Z1/RT (/m).
T_C	0.0003	Cilia/BM time constant (s).
GAIN_C	16	Cilia/BM coupling gain, C (dB).
REF_POT	0	Reference potential (V).

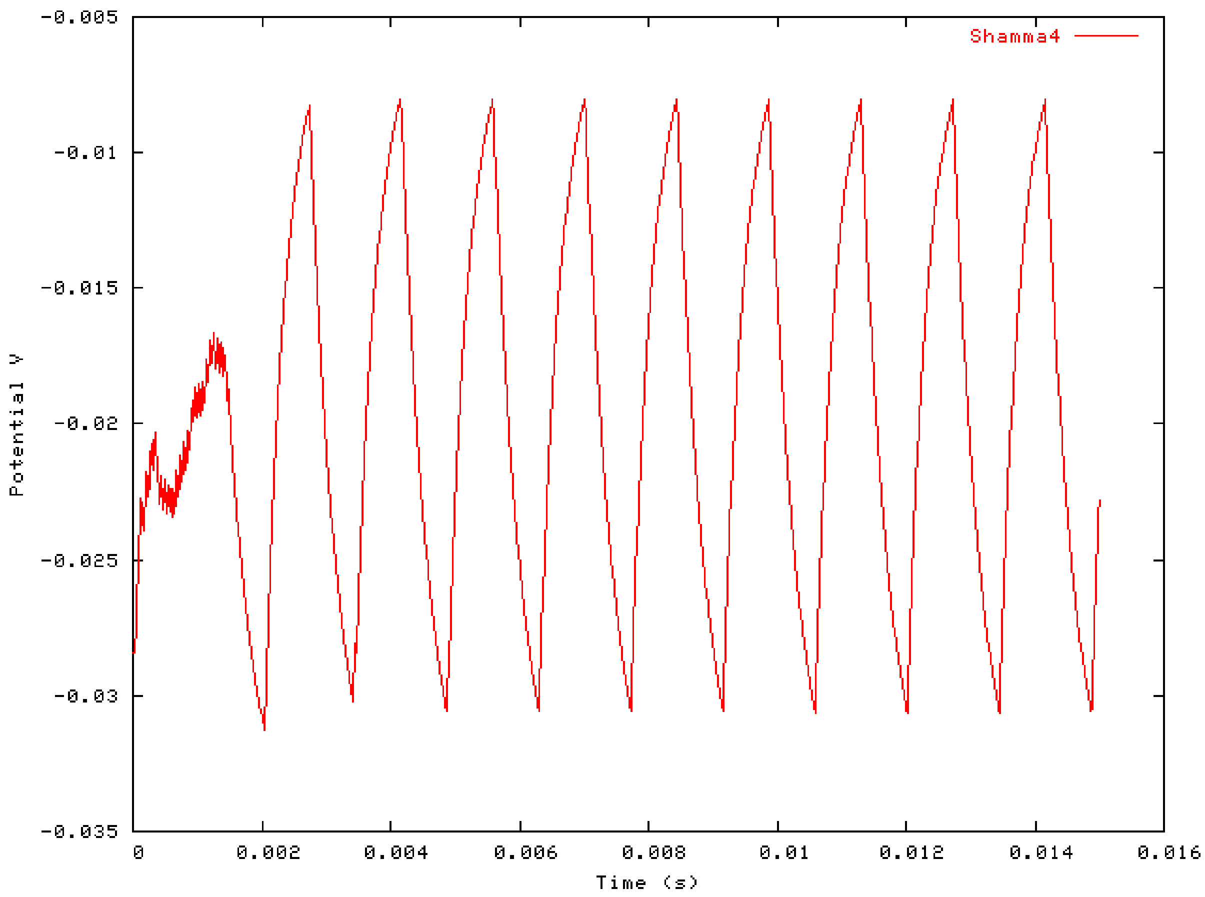
Examples Using the ’IHCRP_Shamma’ Process Module

Some of the following tests require multiple observations using a range of stimuli and cannot be produced directly with a single run of the *AMS* program. These simulation scripts are run using the *AutoTest* program, which is employed to test the operation of the DSAM process modules.

Simple Response Example: AutoTest/HCRP/Shamma4.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Shamma4.sim
begin  {

 Stim_Puretone    < PTone3.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < BPassGP2381.par
 BM_DRNL_Test    < DRNLT17_3k.par
 IHCRP_Shamma    < ShammaRP1.par
 @ Display_Signal

}
```

Figure 9.44. Pure tone response of the Shamma hair cell receptor process module.



AC/DC Ratio vs Frequency Example: AutoTest/HCRP/Shamma1.sim

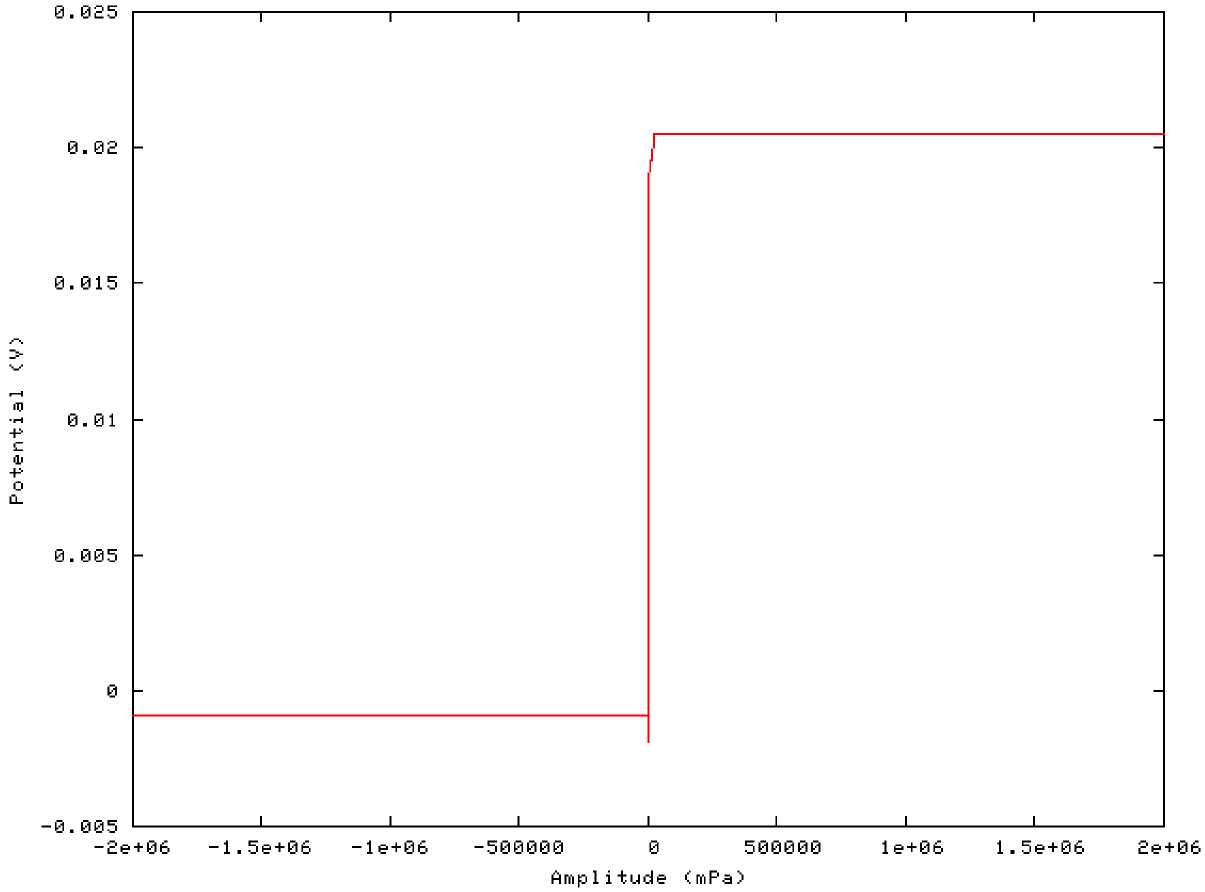
```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Shamma1.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < BPassGP2381.par
 BM_DRNL_Test     < DRNLT17_3k.par
 IHCRP_Shamma     < ShammaRP1.par

}
```

See Figure 9.38, “AC/DC frequency dependent characteristics for the DSAM IHCRP models.”.

Peak/Troughs function Example: AutoTest/HCRP/Shamma2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.
# ./HCRP/Shamma2.sim
begin  {

 Stim_Puretone    < PTone2.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < BPassGP2381.par
 BM_DRNL        < DRNL700.par
 IHCRP_Shamma     < ShammaRP1.par

}
```

Figure 9.45. Peak-Trough characteristics for the Shamma IHCRP model



AC/DC Ratio vs Intensity Example: AutoTest/HCRP/Meddis3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Meddis3.sim
begin  {
 Stim_Puretone   < PTone1.par
 Trans_Gate     < Ramp1.par
 Filt_BandPass   < PreEmph1.par
 BM_GammaT      < GammaT18k.par
 IHCRP_Meddis    < MeddisRP1.par

}
```

See Figure 9.40, “AC/DC intensity dependent characteristics for the DSAM IHCRP models.”.

IHCRP_Shamma3StateVelIn: Inner Hair Cell Receptor Potential (2002)

MoHCRPSham3StVIn.[ch]

Description

This module incorporates a modified version of the model by Shamma et al. ([31]) of the transduction of basilar membrane (BM) motion into receptor potential. It supplies input to the inner hair cell synapse models, e.g. IHC_Meddis2000 , IHC_Zhang, IHC_Carney or IHC_Meddis86a . The displacement of the IHC cilia, u(t), is given by

$$\tau_c \frac{du(t)}{dt} + u(t) = \tau_c C_{cilia} v(t)$$

where C_{cilia} is a gain factor and τ_c is a time constant. Cilia therefore move in phase with BM velocity at low frequencies and with displacement at high frequencies. Cilia displacement causes a change in the number of open ion channels and, consequently, in the apical conductance G(u). The total apical conductance is given by

$$G(u) = G_{cilia}^{\max} \left\{ 1 + \exp\left(-\frac{u(t) - u_0}{s_0} \right) \times \left[1 + \exp\left(-\frac{u(t) - u_1}{s_1} \right) \right] \right\}^{-1} + G_a$$

- where G^{max}_{cilia} is the intracellular conductance with all channels open, and G_a is the passive conductance in the apical membrane. The proportion of open channels is modeled as a three-state energy barrier (Boltzmann) function (see [24]), where s_0, u_0, s_1 and u_1 are constants determining the exact shape of the nonlinearity. Equation 9.0, “” replaces a first-order Bolztmann function employed by Shamma et al. The membrane potential of the cell body is described by

$$C_m \frac{dV(t)}{dt} + G(u) \left[V(t) - E_t \right] + G_k \left[V(t) - E'_{sk} \right] = 0$$

where V(t) is the intracellular hair cell potential; C_m is the cell capacitance; G_k is the voltage-invariant basolateral membrane conductance; E_t is the endocochlear potential; and $E'_k = E_k + E_tR_p/(R_t + R_p)$ is the reversal potential of the basal current E_k, corrected for the resistance (E_t, R_p) of the supporting cells (see [31]).

Inputs	Arbitrary single process. Usually a basilar membrane process supplies velocity input to this process.
Outputs	Produces an output signal with the same dimensions as the input signal. The output is a potential in volta (V). It usually supplies input to the inner hair cell synapse models, e.g. the section called “IHC_Meddis2000: Inner Hair Cell Synapse”.
Reference	[32]

Module Parameters

Example 9.10. IHCRP_Shamma3StateVelIn Parameter File

E_T	0.1	Endocochlear potential, Et (V).
E_K	-0.07045	Reversal potential, Ek (V).
RP_CORRECTION	0.04	Reversal potential correction, Rp/(Rt+Rp).
C_TOTAL	6e-12	Total capacitance, C = Ca + Cb (F).
G0	1.974e-09	Resting conductance, G0 (S).
G_K	1.8e-08	Potassium conductance, Gk (S = Siemens).
G_MAXC	8e-09	Maximum mechanical conductance, Gmax (S).
T_C	0.00213	Cilia/BM time constant (s).
GAIN_C	16	Cilia/BM coupling gain, C (dB).
REF_POT	0	Reference potential (V).
S0	8.5e-08	Sensitivity constant, S0 (/m).
S1	5e-09	Sensitivity constant, S1 (/m).
U0	7e-09	Offset constant, U0 (m).
U1	7e-09	Offset constant, U1 (m).

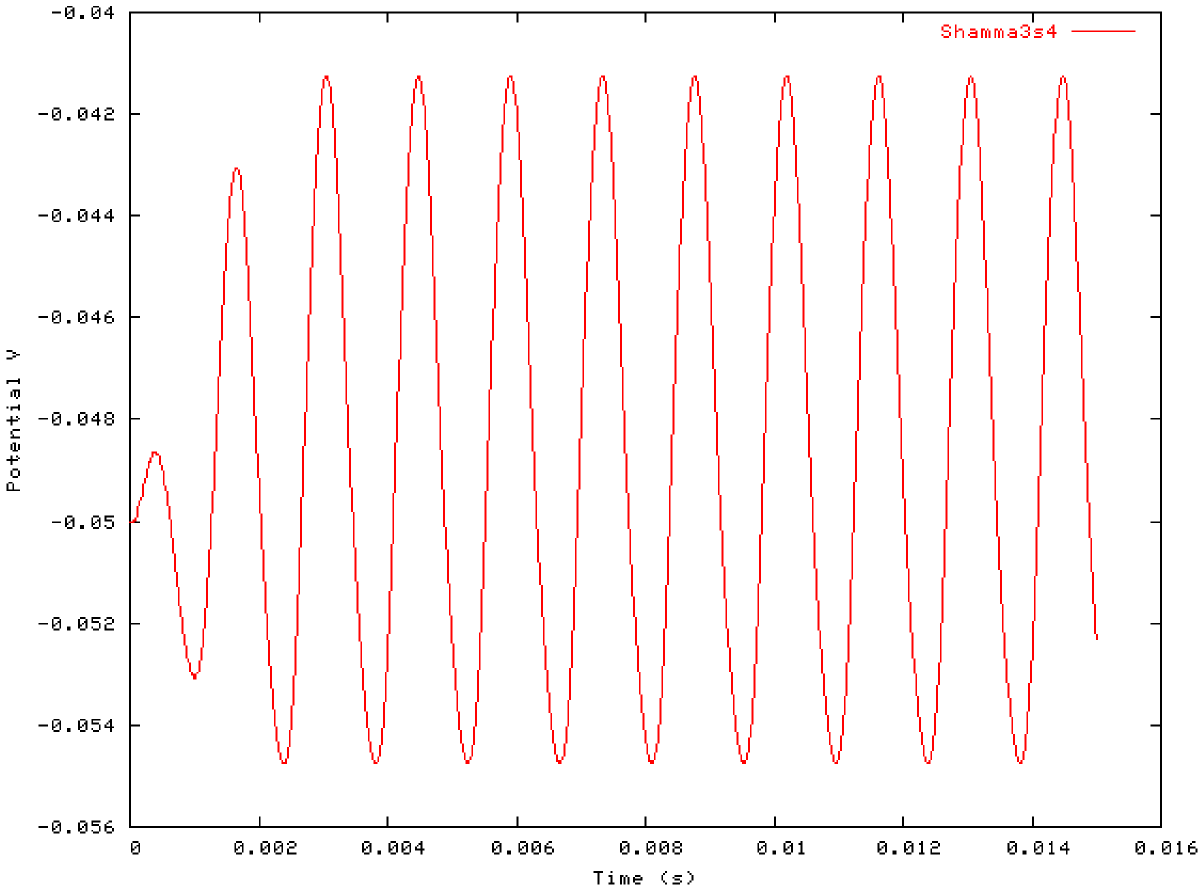
Examples Using the ’IHCRP_Shamma3StateVelIn’ Process Module

Some of the following tests require multiple observations using a range of stimuli and cannot be produced directly with a single run of the *AMS* program. These simulation scripts are run using the *AutoTest* program, which is employed to test the operation of the DSAM process modules.

Simple Response Example: AutoTest/HCRP/Shamma3s4.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Shamma3s4.sim
begin {

 Stim_Puretone   < PTone3.par
 Trans_Gate     < Ramp1.par
 Filt_BandPass   < BPassGP2381.par
 BM_DRNL_Test    < DRNLT17_3k.par
 IHCRP_Shamma3StateVelIn < Shamma3sRP1.par
 Display_Signal
}
```

Figure 9.46. Pure tone response of the Shamma 3 State Velocity In hair cell receptor process module.



AC/DC Ratio vs Frequency Example: AutoTest/HCRP/Shamma3s1.sim

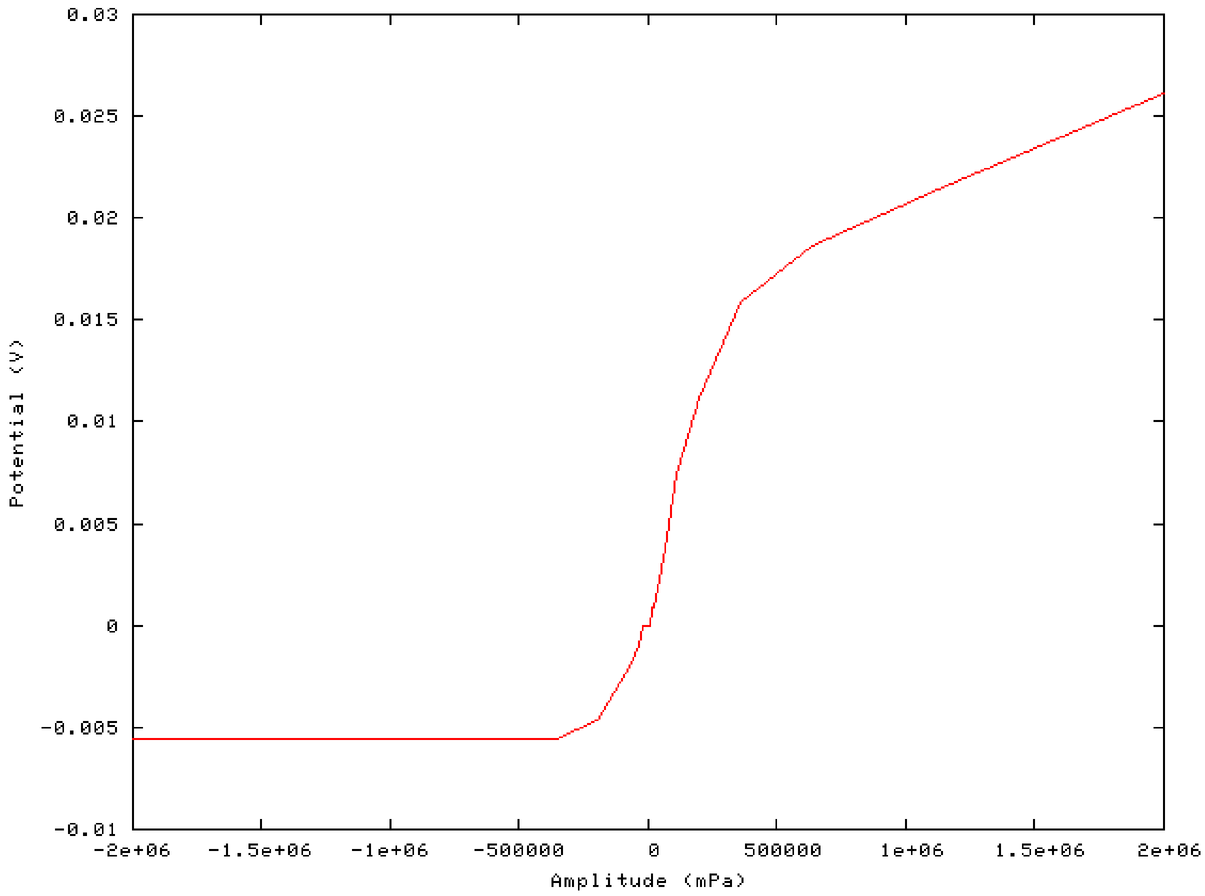
```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Shamma3s1.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < BPassGP2381.par
 BM_DRNL_TEST     < DRNLT17_3k.par
 IHCRP_Shamma3StateVelIn < Shamma3sRP1.par

}
```

See Figure 9.38, “AC/DC frequency dependent characteristics for the DSAM IHCRP models.”.

Peak/Troughs function Example: AutoTest/HCRP/Shamma3s2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Shamma3s2.sim
begin  {

 Stim_Puretone    < PTone2.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < BPassGP2381.par
 BM_DRNL        < DRNL700.par
 IHCRP_Shamma3StateVelIn < Shamma3sRP1.par

}
```

Figure 9.47. Peak-Trough characteristics for the Shamma 3 State Velocity In IHCRP model



AC/DC Ratio vs Intensity Example: AutoTest/HCRP/Meddis3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Meddis3.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_GammaT       < GammaT18k.par
 IHCRP_Meddis     < MeddisRP1.par

}
```

See Figure 9.40, “AC/DC intensity dependent characteristics for the DSAM IHCRP models.”.

IHCRP_Zhang: Inner Hair Cell Receptor Potential (2001)

MoHCRPZhang.[ch]

Description

This is the Zhang et al [34] receptor potential module. It was revised from C code provided by the authors (the ’ARLO’ code). It supplies input to the inner hair cell synapse models, e.g. IHC_Zhang, IHC_Meddis2000 , IHC_Carney or IHC_Meddis86a .

Inputs	Arbitrary single process. Usually a basilar membrane process supplies input to this process..
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	[34]

Module Parameters

Example 9.11. HCRP_Shamma Parameter File

E_T	0.1	Endocochlear potential, Et (V).
E_K	-0.084	Reversal potential, Ek (V).
RP_CORRECTION	0.04	Reversal potential correction, Rp/(Rt+Rp).
C_TOTAL	6.21e-12	Total capacitance, C = Ca + Cb (F).
G0	4.3e-09	Resting conductance, G0 (S).
G_K	1.07e-08	Potassium conductance, Gk (S = Siemens).
G_MAXC	4.18e-09	Maximum mechanical conductance, Gmax (S).
BETA	0.25	Beta constant, exp(-G1/RT).
GAMMA	1e+07	Gamma constant, Z1/RT (/m).
T_C	0.0003	Cilia/BM time constant (s).
GAIN_C	16	Cilia/BM coupling gain, C (dB).
REF_POT	0	Reference potential (V).

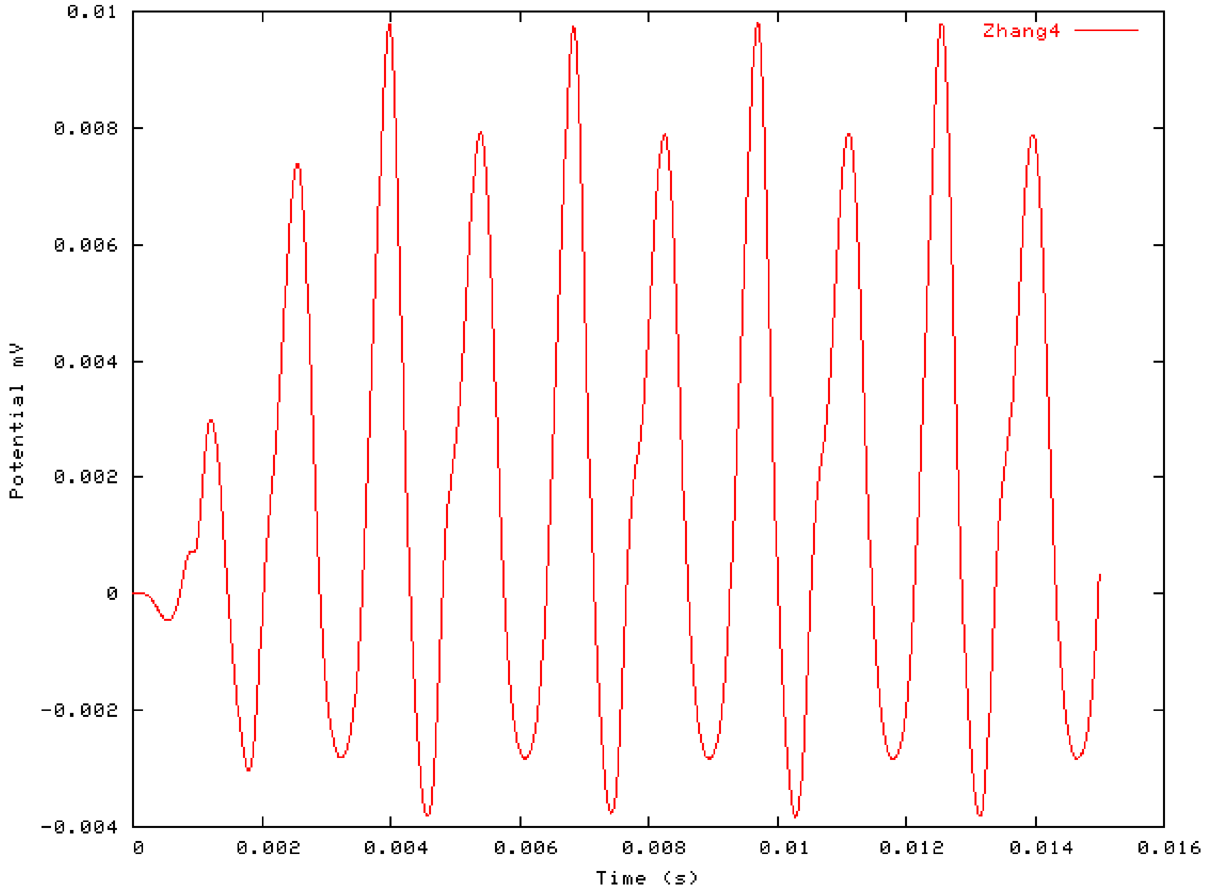
Examples Using the ’IHCRP_Shamma’ Process Module

Some of the following tests require multiple observations using a range of stimuli and cannot be produced directly with a single run of the *AMS* program. These simulation scripts are run using the *AutoTest* program, which is employed to test the operation of the DSAM process modules.

Simple Response Example: AutoTest/HCRP/Zhang4.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Zhang4.sim
begin  {

 Stim_Puretone    < PTone3.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PEHuY1.par
 BM_Zhang        < Zhang18k.par
 IHCRP_Zhang      < ZhangRPCat1.par
 @ Display_Signal

}
```

Figure 9.48. Pure tone response of the Zhang hair cell receptor process module.



AC/DC Ratio vs Frequency Example: AutoTest/HCRP/Zhang1.sim

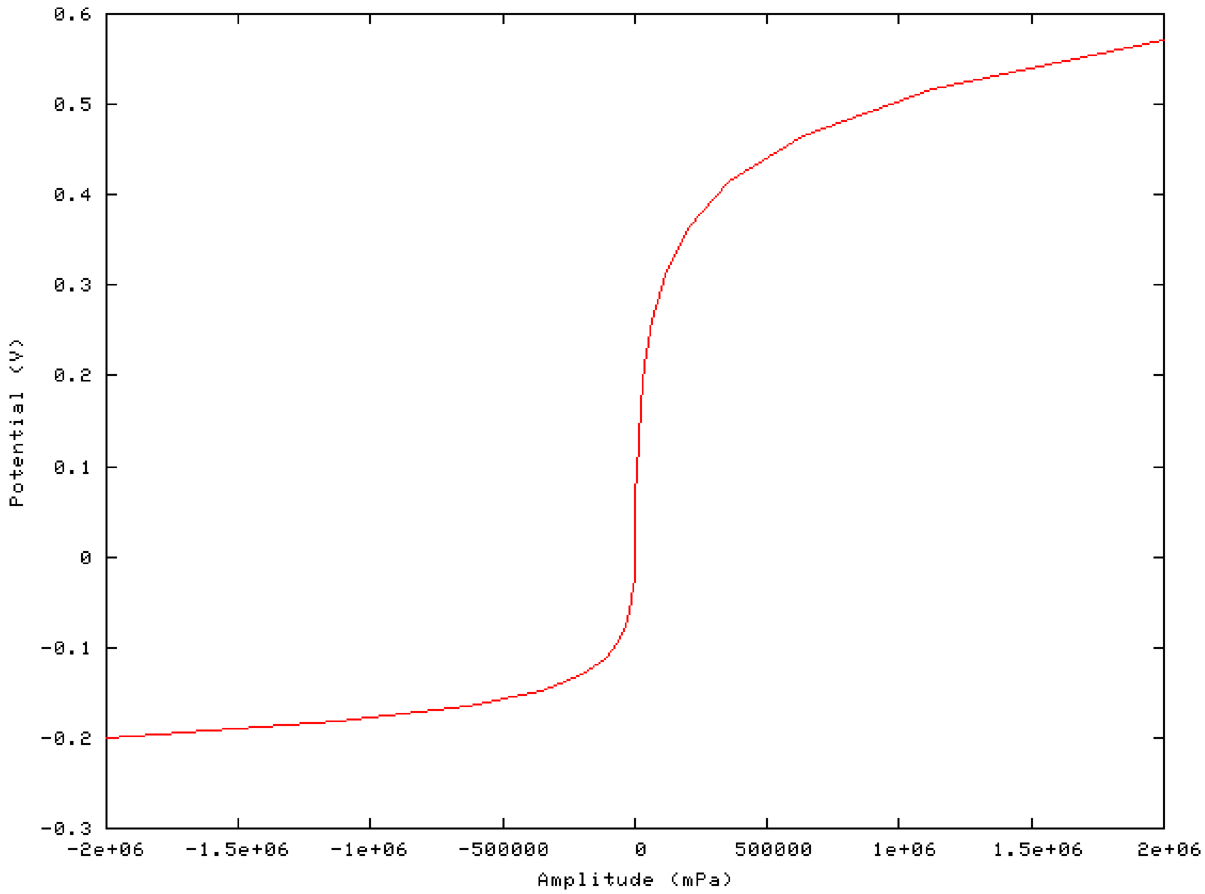
```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Zhang1.sim
begin  {

 Stim_Puretone   < PTone1.par
 Trans_Gate     < Ramp1.par
 Filt_BandPass   < PEHuY1.par
 BM_Zhang       < Zhang18k.par
 IHCRP_Zhang     < ZhangRPCat1.par
 @ Display_Signal

}
```

See Figure 9.38, “AC/DC frequency dependent characteristics for the DSAM IHCRP models.”.

Peak/Troughs function Example: AutoTest/HCRP/Zhang2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Zhang2.sim
begin  {

 Stim_Puretone   < PTone2.par
 Trans_Gate     < Ramp1.par
 Filt_BandPass   < PEHuY1.par
 BM_Zhang       < Zhang700.par
 IHCRP_Zhang     < ZhangRPCat1.par

}
```

Figure 9.49. Peak-Trough characteristics for the Zhang IHCRP model



AC/DC Ratio vs Intensity Example: AutoTest/HCRP/Meddis3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./HCRP/Meddis3.sim
begin  {

 Stim_Puretone    < PTone1.par
 Trans_Gate      < Ramp1.par
 Filt_BandPass    < PreEmph1.par
 BM_GammaT       < GammaT18k.par
 IHCRP_Meddis     < MeddisRP1.par

}
```

See Figure 9.40, “AC/DC intensity dependent characteristics for the DSAM IHCRP models.”.

Chapter 10. Neuron Model Modules

Neur_ArleKim: Neural Cell Model

File name: MoNCArkeKim.[ch]

Description

This module implements a version of the McGregor model of a neural cell. The changes implemented in [2] have been added to this version of the McGregor model. Several different cells can be modelled.

Inputs	Arbitary single inputs. This process is normally uses inputs from a dendrite (low-pass filtered) process.
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	[2]

Module Parameters

Example 10.1. Neur_ArleKim Parameter File

T_M	0.005	Cell membrane time-constant, tm (s).
T_GK	0.002	Potassium decay time-constant, tGk (s).
T_TH	0.002	Threshold time-constant, tTh (s).
C	0	Accommodation constant, c (dimensionless).
B	0.00025	Delayed rectifier postassium conductance, b (nano-Siemens).
TH0	12	Cell resting Threshold, Th0 (mV).
ACTION_POT	50	Action potential (mV).
V_NL	0.5	Non-linear voltage constant, Vnl (mV).
E_K	-65	Reversal potential of the potassium conductance, Ek (mV).
E_B	-65	Reversal potential of all other conductances, Eb (mV).
G_K	0.02	Resting potassium conductance, gk (nS).
G_B	0	Resting component of all other conductances, gb (nS).

Examples Using the ’Neur_ArleKim’ Process Module

Simple Bushy Cell Response Examples

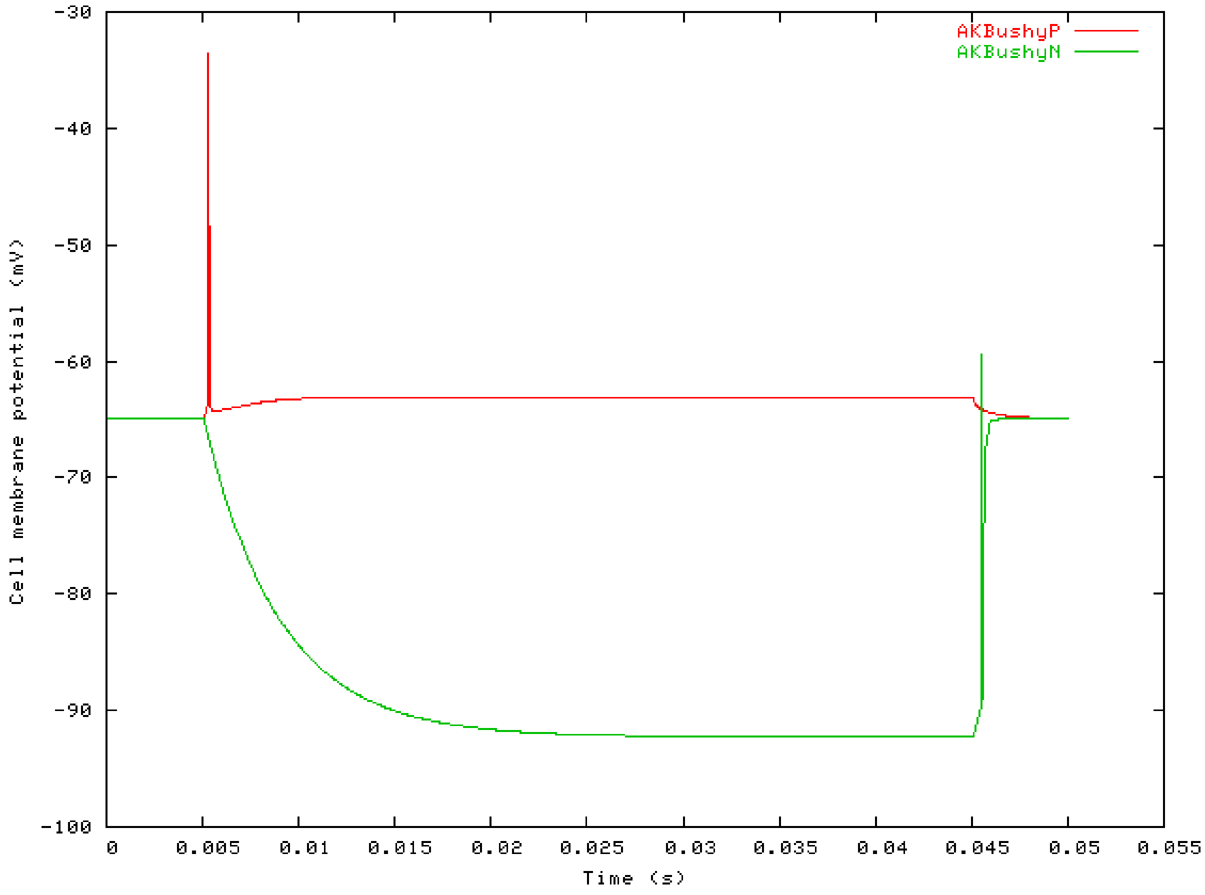
Example 10.2. AutoTest/NC/AKBushyP.sim: Super-polarizing Injected current

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./NC/AKBushyP.sim
begin {

 Stim_StepFun    < StepFun1.par
 Neur_ArleKim    < AKBushy.par

}
```

Example 10.3. AutoTest/NC/AKBushyN.sim: Hyper-polarizing Injected current

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./NC/AKBushyN.sim
begin  {

 Stim_StepFun    < StepFun2.par
 Neur_ArleKim    < AKBushy.par

}
```

Figure 10.1. The current step response of the Arle-Kim neural cell model The hyperpolarised and superpolarised bushy cell responses were produced using the 'AKBushyP.sim' and 'AKBushyN.sim' Simulation scripts respectively.



Simple Fusiform Cell Response Examples

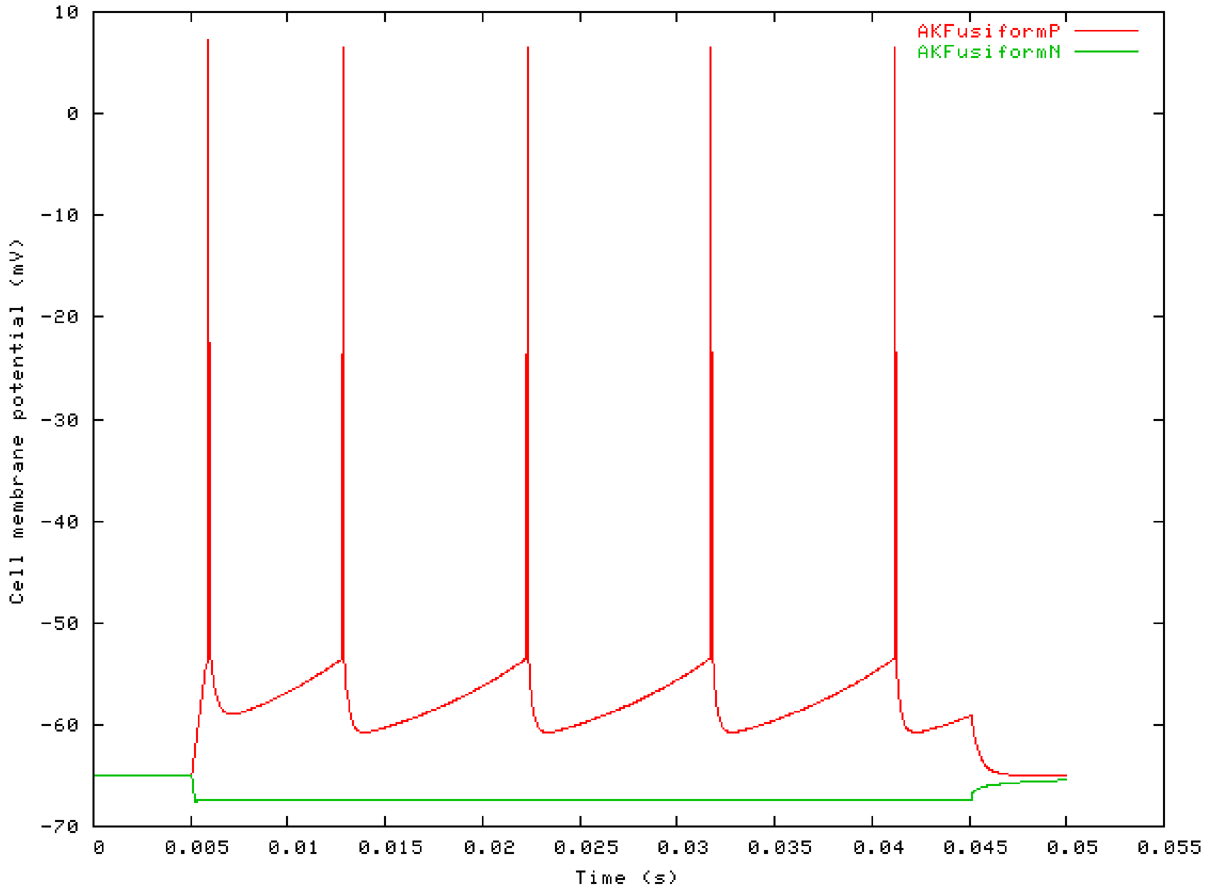
Example 10.4. AutoTest/NC/AKFusiformP.sim: Super-polarizing Injected current

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./NC/AKFusiformP.sim
begin  {

 Stim_StepFun    < StepFun3.par
 Neur_ArleKim    < AKFusiform.par

}
```

Example 10.5. AutoTest/NC/AKFusiformN.sim: Hyper-polarizing Injected current

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./NC/AKFusiformN.sim
begin  {

 Stim_StepFun    < StepFun4.par
 Neur_ArleKim    < AKFusiform.par

}
```

Figure 10.2. The current step response of the Arle-Kim neural cell model The hyperpolarised and superpolarised Fusiform cell responses were produced using the ’AKFusiformP.sim’ and ’AKFusiformN.sim’ Simulation scripts respectively.



Simple Stellate Cell Response Examples

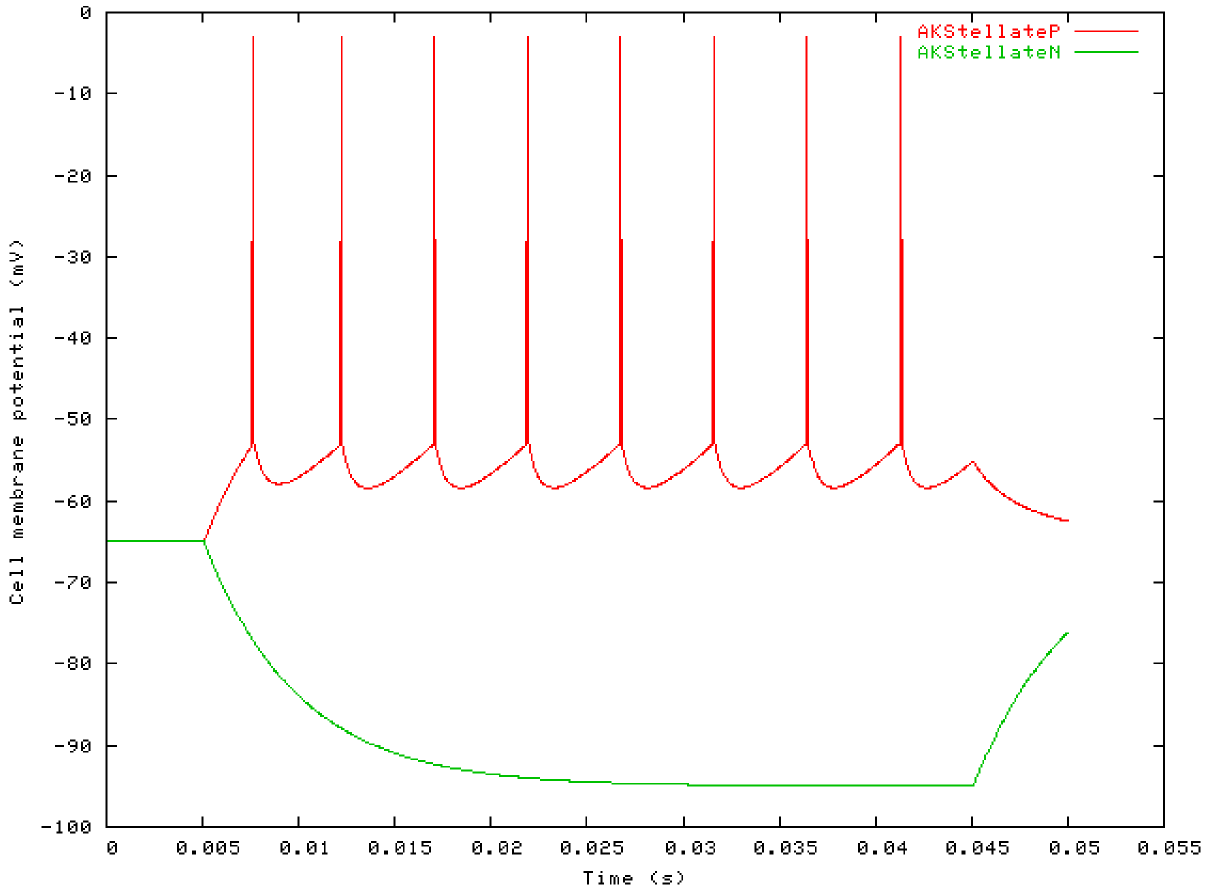
Example 10.6. AutoTest/NC/AKStellateP.sim: Super-polarizing Injected current

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./NC/AKStellateP.sim
begin  {

 Stim_StepFun     < StepFun1.par
 Neur_ArleKim     < AKStellate.par

}
```

Example 10.7. AutoTest/NC/AKStellateN.sim: Hyper-polarizing Injected current

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./NC/AKStellateN.sim
begin  {

 Stim_StepFun     < StepFun2.par
 Neur_ArleKim     < AKStellate.par

}
```

Figure 10.3. The current step response of the Arle-Kim neural cell model The hyperpolarised and superpolarised Stellate cell responses were produced using the ’AKStellateP.sim’ and ’AKStellateN.sim’ Simulation scripts respectively.



Potential vs. Current Example

Example 10.8. AutoTest/NC/AKBushy1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./NC/AKBushy1.sim
begin  {

 Stim_StepFun    < StepFun5.par
 Neur_ArleKim    < AKBushy.par

}
```

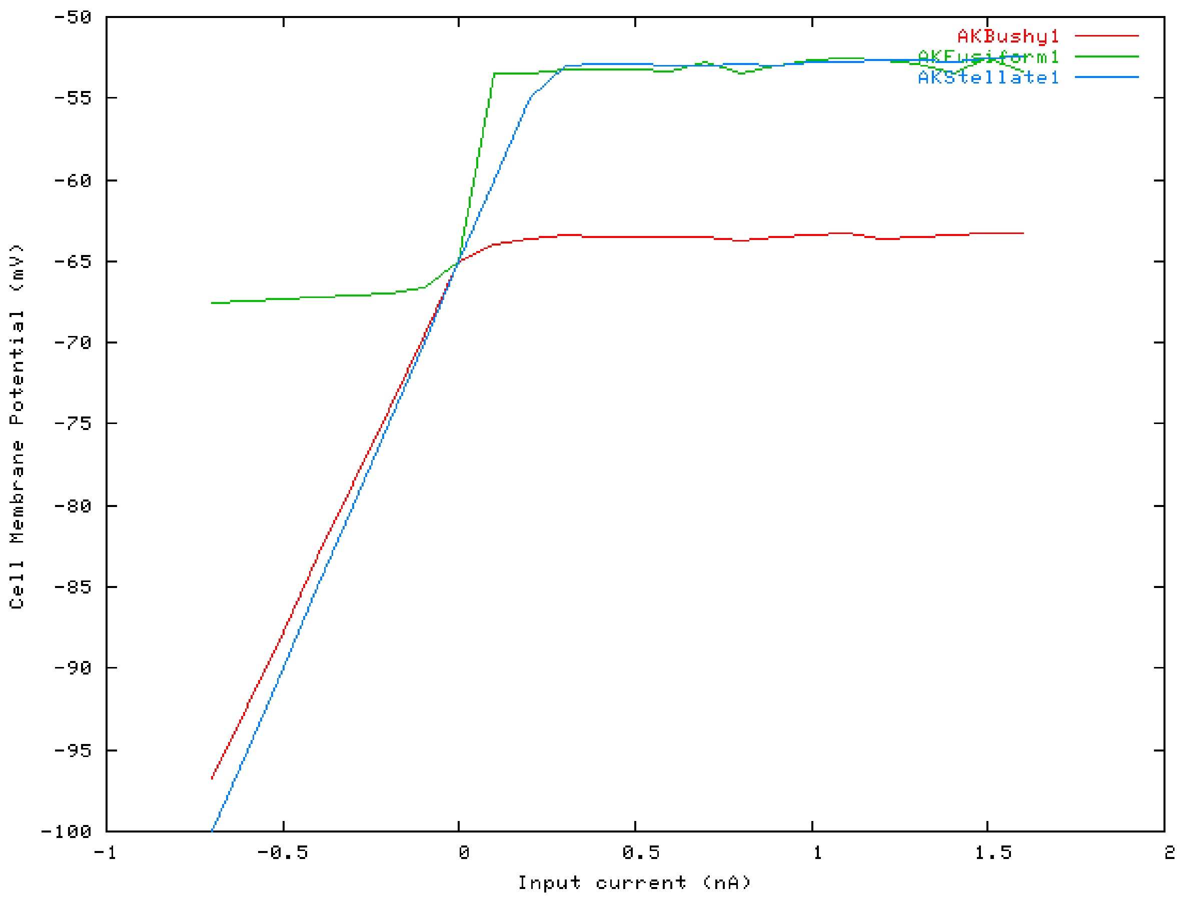
Example 10.9. AutoTest/NC/AKFusiform1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./NC/AKFusiform1.sim
begin  {

 Stim_StepFun    < StepFun5.par
 Neur_ArleKim    < AKFusiform.par

}
```

Example 10.10. AutoTest/NC/AKStellate1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./NC/AKStellate1.sim
begin  {

 Stim_StepFun    < StepFun5.par
 Neur_ArleKim    < AKStellate.par

}
```

Figure 10.4. Input-output current characteristics for the Arle-Kim neural cell model The functions for the respective cell characteristics were produced using the ’AKStellate1.sim’, ’AKFusiform1.sim’ and ’AKBush1.sim’ simulation script files.



Neur_HHuxley: Hodkin Huxley Neural Cell Model

File name: MoNCHHuxley.[ch]

Description

This module implements an efficient version of the Hodgkin-Huxley general neural cell model. Its fast operation is enabled through the use of tables for the ion-channel voltage-dependent conductances.

The module has two *operating modes*: 'normal' and 'voltage_clamp' respectively. If these operating modes are prefaced with 'debug_', e.g. 'debug_normal', then a signal file 'test.dat' will be output in the format:

membrane potential, leakage current, IC 1 (activating, inactivating currents), IC2 ...

This module can be connected to up to four processes (*EarObjects*). When the *injection mode* is 'on', the first connected process is always treated as an injected current. Otherwise the first, second and third processes (or second, third and fourth if the *injection mode* is 'on') are used as excitatory, inhibitory and shunt-inhibitory signals respectively.

The ion channels will be listed if the *print ion channel tables mode* is 'On', but as these tables can be very long, it is useful to set this option to 'off'. One would set this option to 'On' for diagnostic or de-bugging purposes.

This cell model is defined by differential equations for which the resting state has not been analytically determined. For this reason the *resting potential* for a particular cell configuration can be set as a parameter. If the value of the *resting potential* parameter is set to a value greater or equal to zero, then before the process is first run in a simulation, the resting potential will be determined in a calibration run. The calculated *resting potential* value can then be put into the parameter file for subsequent simulation runs, as long as the other cell parameters remain unchanged. If the other cell parameters are changed, then the resting potential may change. For this reason it is simpler to leave the *resting potential* parameter set to do a calibration run each time it is first run. Setting the *resting potential* parameter should only be done when many runs of an established cell are required, and the calibration would unduly increase the experiment time. It is important to reiterate that a calibration run is only done the first time a particular simulation is run, only when the simulation is restarted, or the process reset will there be another calibration run.

Inputs	One to four arbitary inputs. This process is normally uses inputs from a dendrite (low-pass filtered) process.
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	[14]

Module Parameters

Example 10.11. Neur_HHuxley Parameter File

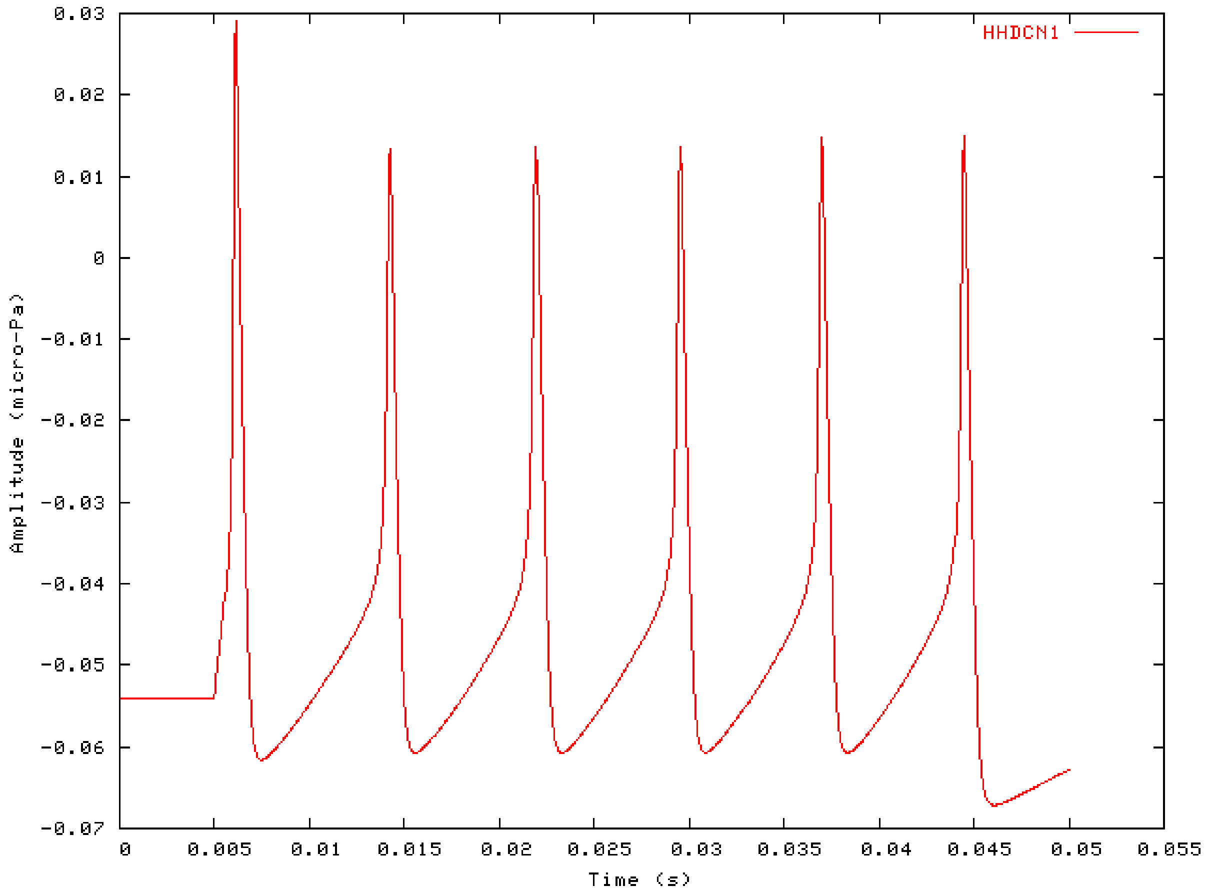
DIAG_MODE	OFF	Diagnostic mode ('off', 'screen' or <file name>).
OP_MODE	NORMAL	Operation mode ('normal' or 'voltage clamp').
INJECT_MODE	OFF	Injection mode ('on' or 'off').
EXCIT_REV_POT	-0.01	Excitatory reversal potential (V).
INHIB_REV_POT	-0.01	Inhibitory reversal potential (V).
SHUNT_INHIB_REV_POT	-0.01	Shunt inhibitory reversal potential (V).
CAPACITANCE	1e-11	Cell capacitance (Farads).
RESTING_POT	-0.06553	Resting potential (>=0 for calibration run).
RESTING_DUR	0.03	Resting signal duration - used when finding V0 (s).
RESTING_DV	1e-09	Resting criteria, dV (V).
DESCRIPTION	"Na_Conner"	Description.
MODE	BOLTZMANN	Mode option ('file', 'hHuxley' or 'boltzmann').
ENABLED	ON	Ion channel enabled status ('on' or 'off).
EQUIL_POT	0.055	Equilibrium potential (V).
BASE_MAX_COND	3e-07	Base maximum conductance (S).
ACTIVATION	1	Activation exponent (real).
COND_Q10	2.5	Conductance Q10.
V_HALF	0:-0.0191	
V_HALF	1:-0.053	
Z	0:3.7	
Z	1:-3.6	
TAU	0:0.0001	
TAU	1:0.0005	
DESCRIPTION	"K+_FastHiT"	Description.
MODE	BOLTZMANN	Mode option ('file', 'hHuxley' or 'boltzmann').
ENABLED	ON	Ion channel enabled status ('on' or 'off).
EQUIL_POT	-0.073	Equilibrium potential (V).
BASE_MAX_COND	5.4e-08	Base maximum conductance (S).
ACTIVATION	1	Activation exponent (real).
COND_Q10	2.5	Conductance Q10.
V_HALF	0:-0.015	
V_HALF	1:0	
Z	0:4	
Z	1:0	
TAU	0:0.00161	
TAU	1:0	
DESCRIPTION	"K+_SlowLoT"	Description.
MODE	BOLTZMANN	Mode option ('file', 'hHuxley' or 'boltzmann').
ENABLED	ON	Ion channel enabled status ('on' or 'off).
EQUIL_POT	-0.073	Equilibrium potential (V).
BASE_MAX_COND	1.8e-08	Base maximum conductance (S).
ACTIVATION	1	Activation exponent (real).
COND_Q10	2.5	Conductance Q10.
V_HALF	0:-0.028	
V_HALF	1:0	
Z	0:4.6	
Z	1:0	
TAU	0:0.01	
TAU	1:0	
DESCRIPTION	"K+_BoltzmannTr"	Description.
MODE	BOLTZMANN	Mode option ('file', 'hHuxley' or 'boltzmann').
ENABLED	ON	Ion channel enabled status ('on' or 'off).
EQUIL_POT	-0.073	Equilibrium potential (V).
BASE_MAX_COND	2.4e-08	Base maximum conductance (S).
ACTIVATION	1	Activation exponent (real).
COND_Q10	2.5	Conductance Q10.
V_HALF	0:-0.0302	
V_HALF	1:-0.0612	
Z	0:0.6	
Z	1:-4.9	
TAU	0:0.001	
TAU	1:0.003	

Examples Using the ’Neur_ArleKim’ Process Module

Simple DCN Cell Response Example: AutoTest/NC/HHDCN1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./NC/HHDCN1.sim
begin  {

 Stim_StepFun      < StepFun6.par
 Neur_HHuxley      < HHDCN1.par
 Display_Signal

}
```

Figure 10.5. Current step response of the Hodgkin-Huxley neural cell model



Neur_McGregor: McGregor Neural Cell Model

File name: MoNCMcGregor.[ch]

Description

This module implements a version of the McGregor model of a neural cell ([19]). Dependent upon the parameters used, it can model several different cells.

Inputs	Arbitary single inputs. This process is normally uses inputs from a dendrite (low-pass filtered) process.
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	[19]

Module Parameters

Example 10.12. Neur_McGregor Parameter File

T_M	0.002	Cell membrane time-constant, tm (s).
T_GK	0.0009	Potassium decay time-constant, tGk (s)
T_TH	0.02	Threshold time-constant, tTh (s).
C	0.3	Accommodation constant, c (dimensionless).
B	0.017	Delayed rectifier postassium conductance, b (nano-Siemens).
TH0	10	Cell resting Threshold, Th0 (mV).
ACTION_POT	50	Action potential (mV).
E_K	-10	Reversal potential of the potassium conductance, Ek (mV).
E_R	-60	Cell resting potential Er (mV).

Examples Using the ’Neur_McGregor’ Process Module

Simple Stellate Cell Response Examples

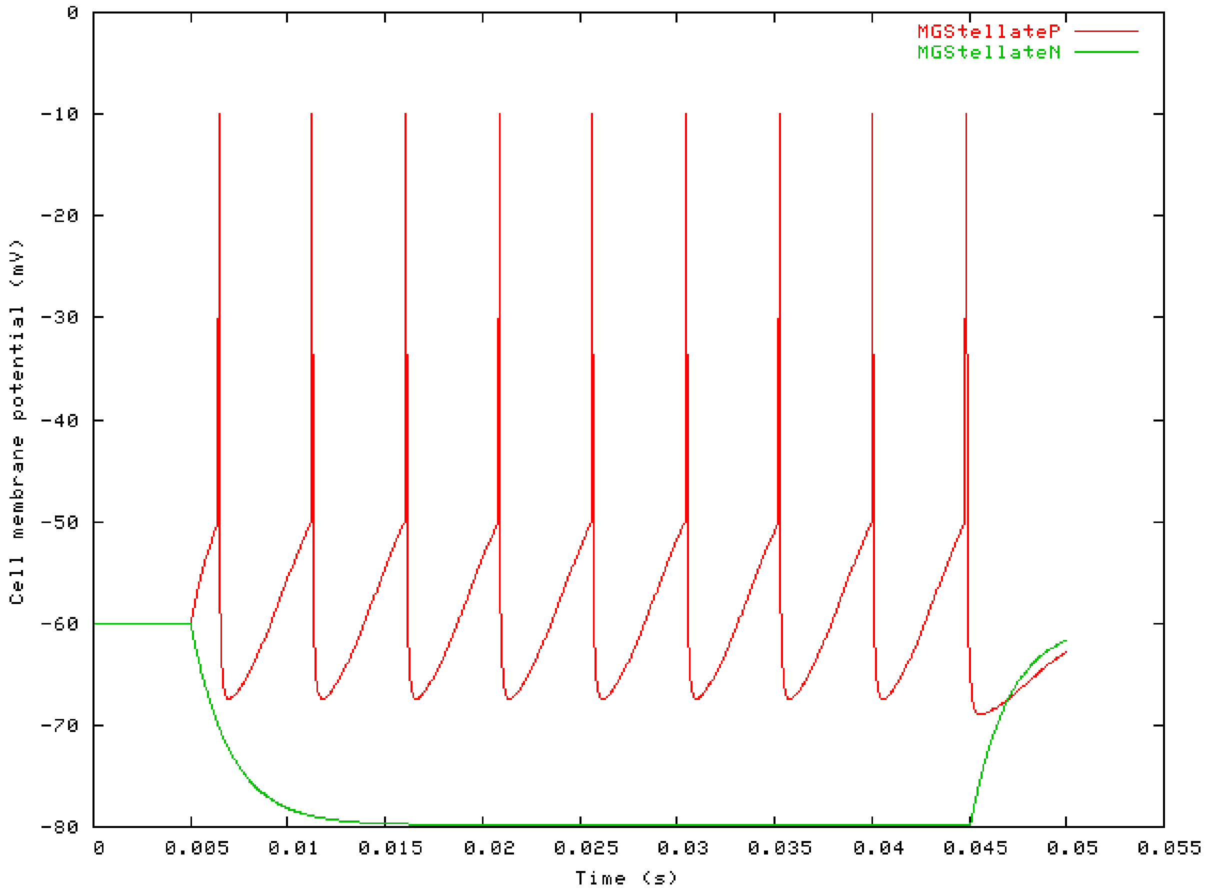
Example 10.13. AutoTest/NC/MGStellateP.sim: Super-polarizing Injected current

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./NC/MGStellateP.sim
begin  {

 Stim_StepFun     < StepFun7.par
 Neur_McGregor    < MGStellate92.par

}
```

Example 10.14. AutoTest/NC/MGStellateN.sim: Hyper-polarizing Injected current

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./NC/MGStellateN.sim
begin  {

 Stim_StepFun     < StepFun8.par
 Neur_McGregor     < MGStellate92.par

}
```

Figure 10.6. The current step response of the Arle-Kim neural cell model The hyperpolarised and superpolarised Stellate cell responses were produced using the ’MGStellateP.sim’ and ’MGStellateN.sim’ Simulation scripts respectively.



Chapter 11. Misc. Modules

Null: Used to Skip Process stage in Program

File name: GeModuleMgr.[ch]

Description

This module can be used in generic programming to maintain the continuity of a process pipeline. It does not carry out any processing; it merely connects the input signal of the process *EarObject* directly to the output signal. This means that it can be replaced by another process module if required.

Module Parameters

This module has no parameters.

Chapter 12. Stimulus Modules

Stim_Click: Click Stimulus Module

File name: StClick.[ch]

Description

Click stimulus generation module.

Inputs	None
Outputs	Single channel.
Reference	

Module Parameters

Example 12.1. Stim_Click Parameter File

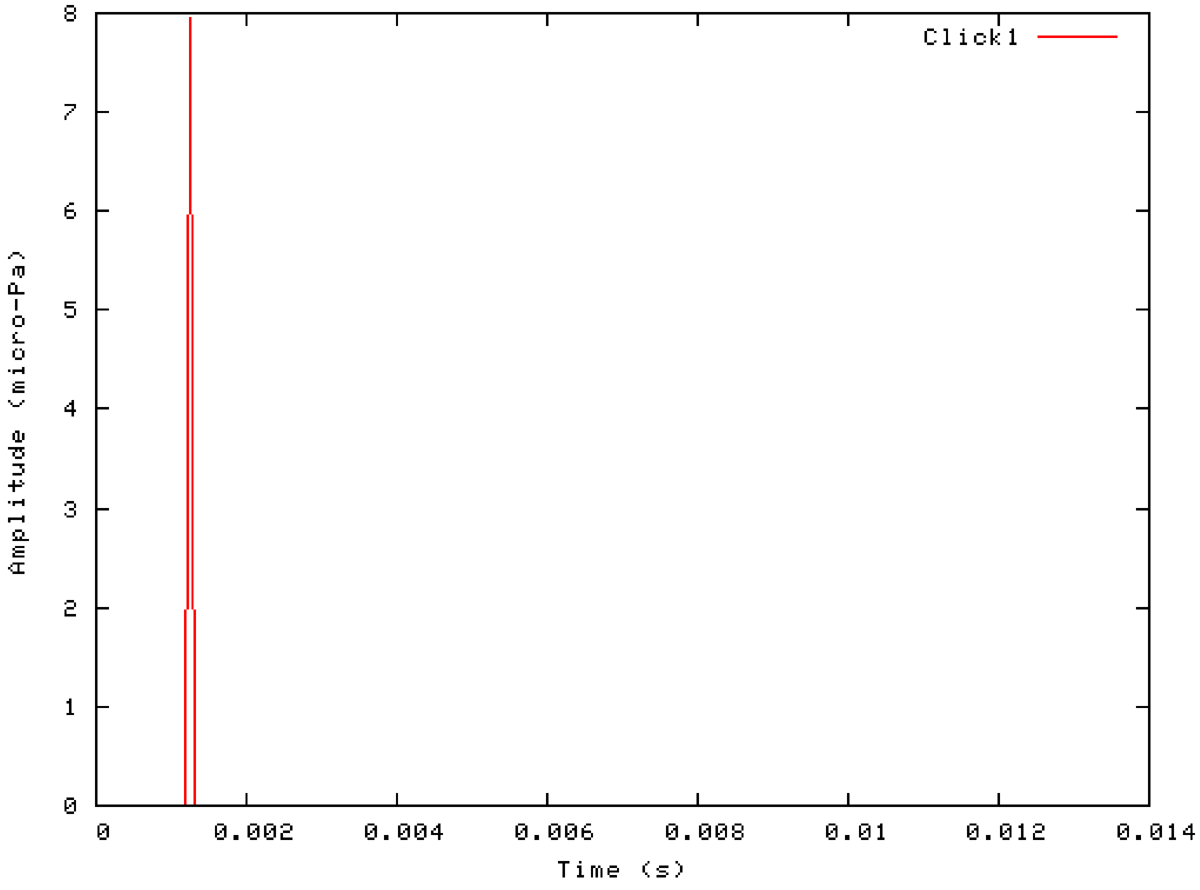
TIME	0.01	Time for the delta-function click (s).
AMPLITUDE	1	Amplitude (uPa).
DURATION	0.1	Duration (s).
DT	1e-05	Sampling interval, dt (s).

Examples Using the ’Stim_Click’ Process Module

Example: AutoTest/St/Click1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/Click1.sim
begin {

 Stim_Click     < Click1.par
 Display_Signal

}
```

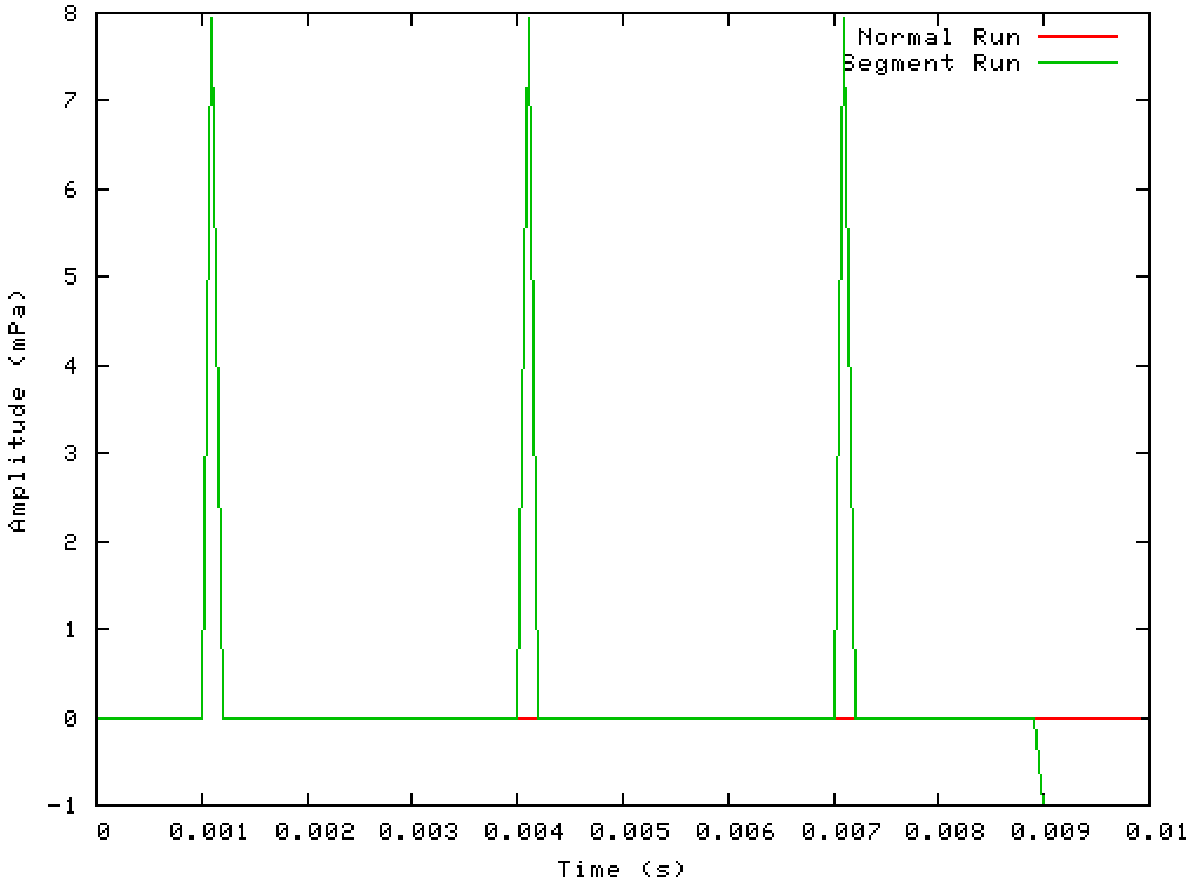
Figure 12.1. A stimulus generated by the Stim_Click module.



Segmented Mode Example: AutoTest/St/Click2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/Click2.sim
begin  {

 Stim_Click     < Click2.par
 Display_Signal

}
```

Figure 12.2. Click stimulus behaviour in *segmented rocessing* mode. The full stimulus is generated once in normal mode, then the stimulus is generated in segments in *segment-processing* mode. Note that in *segmented-processing* mode the stimulus is repeated.



Stim_ExpGatedTone: Exponentially Gated Pure Tone Stimulus Module

File name: StEGatedTone.[ch]

Description

Exponentially gated pure tone stimulus generation routine.

Inputs	None
Outputs	Single channelled signal.
Reference	[26]

Module Parameters

Example 12.2. Stim_Click Parameter File

T_MODE	RAMPED	Type mode (’ramped’ or ’damped’).
F_MODE	ON	Floor mode (’on’ or ’off’).
FREQUENCY	1000	Carrier frequency (Hz).
AMPLITUDE	10000	Amplitude (uPa).
PHASE	0	Phase (degrees).
SILENCE	0	Begin period duration - silence (s).
R_RATE	40	Repetition rate (Hz).
HALF_LIFE	0.005	Half life (s).
FLOOR	0	Floor value (uPa).
DURATION	0.1	Duration (s).
DT	0.0001	Sampling interval, dt (s).

Table 12.1. Stim_ExpGatedTone: ’t_mode’ options

Type Mode	Description
ramped	This produces a ramped (ascending) pulse signal.
damped	This produces a damped (descending) pulse signal

Table 12.2. Stim_ExpGatedTone: ’floor_mode’ options

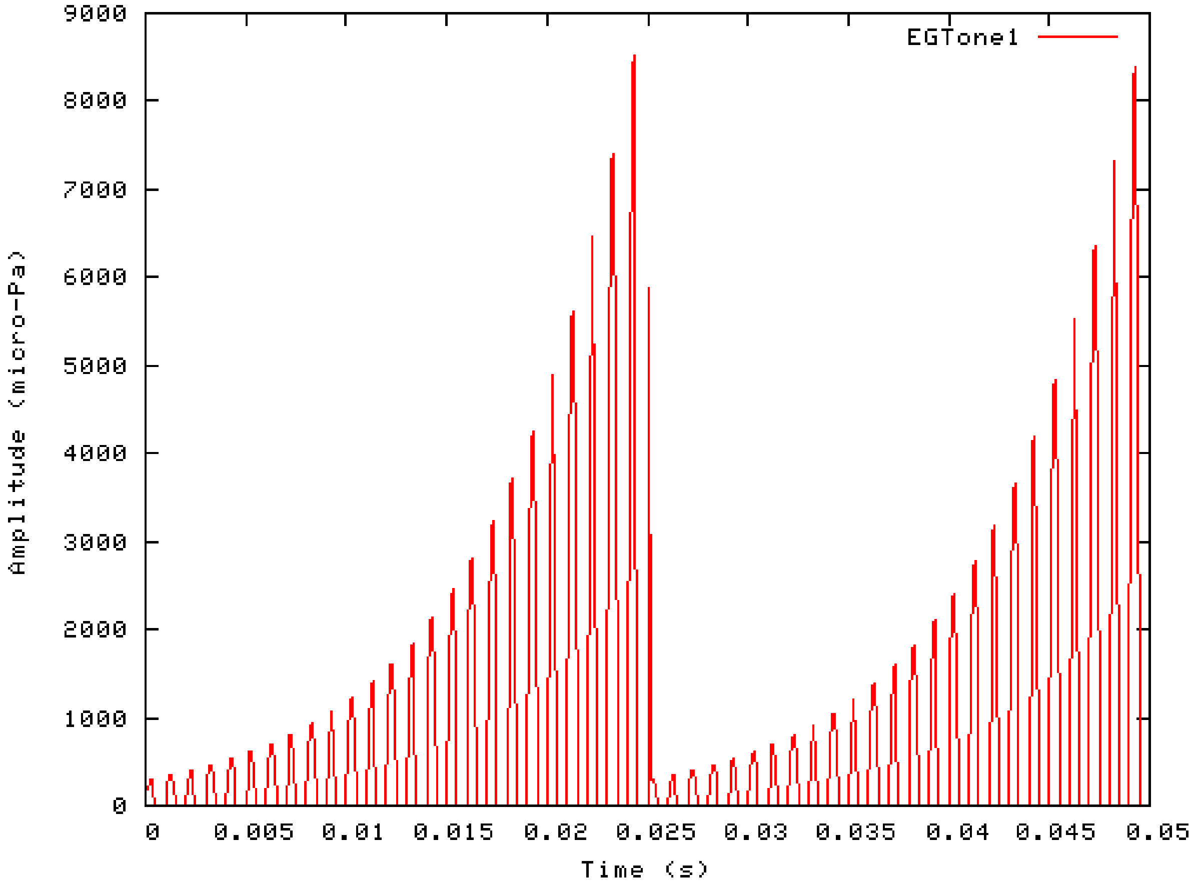
Floor Mode	Description
on	When this mode is on, the output signal is not allowed to fall below the **floor** parameter value.
off	

Examples Using the ’Stim_ExpGatedTone’ Process Module

Example: AutoTest/St/EGTone1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/EGTone1.sim
begin  {

 Stim_ExpGatedTone < EGTone1.par
 Display_Signal

}
```

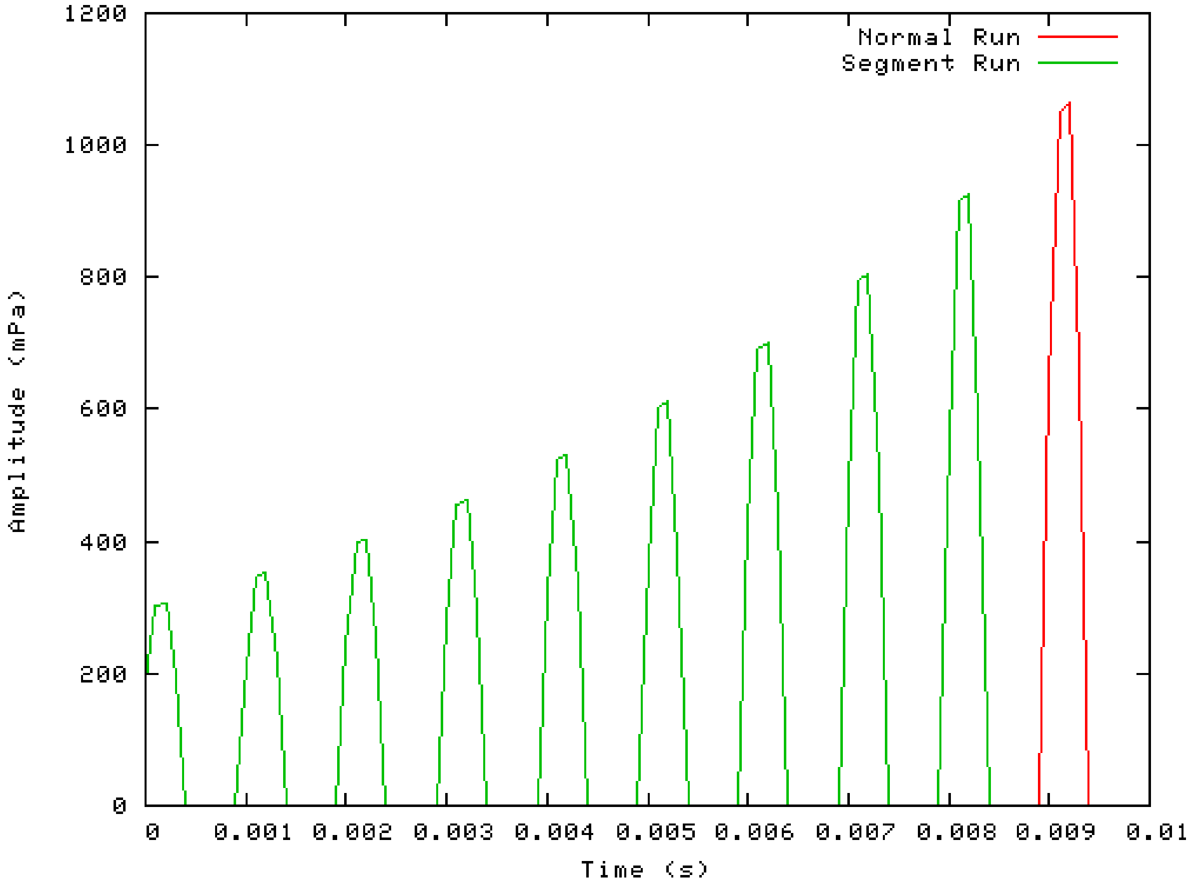
Figure 12.3. A stimulus generated by the exponentially gated pure tone module.



Segmented Mode Example: AutoTest/St/EGTone2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/EGTone2.sim
begin {

 Stim_ExpGatedTone < EGTone1.par
 Display_Signal

}
```

Figure 12.4. Exponentially gated pure tone stimulus behaviour in *segmented rocessing* mode. The full stimulus is generated once in normal mode, then the stimulus is generated in segments in *segment-processing* mode.



Stim_Harmonic: Harmonic Stimulus Module

File name: StHarmonic.[ch]

Description

This module generates a harmonic series stimulus. There are frequency modulation and Butterworth bandpass filter options that can be used. The bandpass filter is in the frequency domain where the filter amplitude is given by

$$FilterAmplitude = \sqrt{\frac{1}{1+\left[\frac{2f-(f_u+f_{sl})}{f_u-f_l}\right]^{2p}}}$$

- where f is the instantaneous frequency, f_u is the upper cut-off frequency, f_l is the lower cut-off frequency and p is the filter order.

Inputs	None.
Outputs	Single channelled signal.
Reference	

Module Parameters

Example 12.3. Stim_Harmonic Parameter File

```
LOW_HARMONIC    1       Lowest harmonic number.
HIGH_HARMONIC   10      Highest harmonic number.
PHASE_MODE      SINE    Phase mode (SINE, COSINE, RANDOM, SCHROEDER,
                        ALTERNATING).
PHASE_PAR       1       Phase parameter (Shroeder phase: C value, Random: random number
                        seed).
MISTUNED_HARM   -1      Mistuned harmonic number (0 = F0, -ve implies none mistuned).
MT_FACTOR       40      Mistuning factor (%).
FUND_FREQ       100     Fundamental frequency (Hz).
INTENSITY       56      Intensity (dB SPL).
DURATION        0.1     Duration (s).
DT              1e-05   Sampling interval, dt (s).
MOD_FREQ        1       Modulation Frequency (Hz).
MOD_PHASE       0       Modulation Phase (degrees).
MOD_DEPTH       0       Modulation depth (%).
ORDER           0       Filter order.
LOW_CUTOFF      200     Lower cut off frequency 3 dB down (Hz).
UPPER_CUTOFF    600     Upper cut off frequency 3 dB down (Hz).
```

Table 12.3. Stim_Harmonic: ’Phase_mode’ options

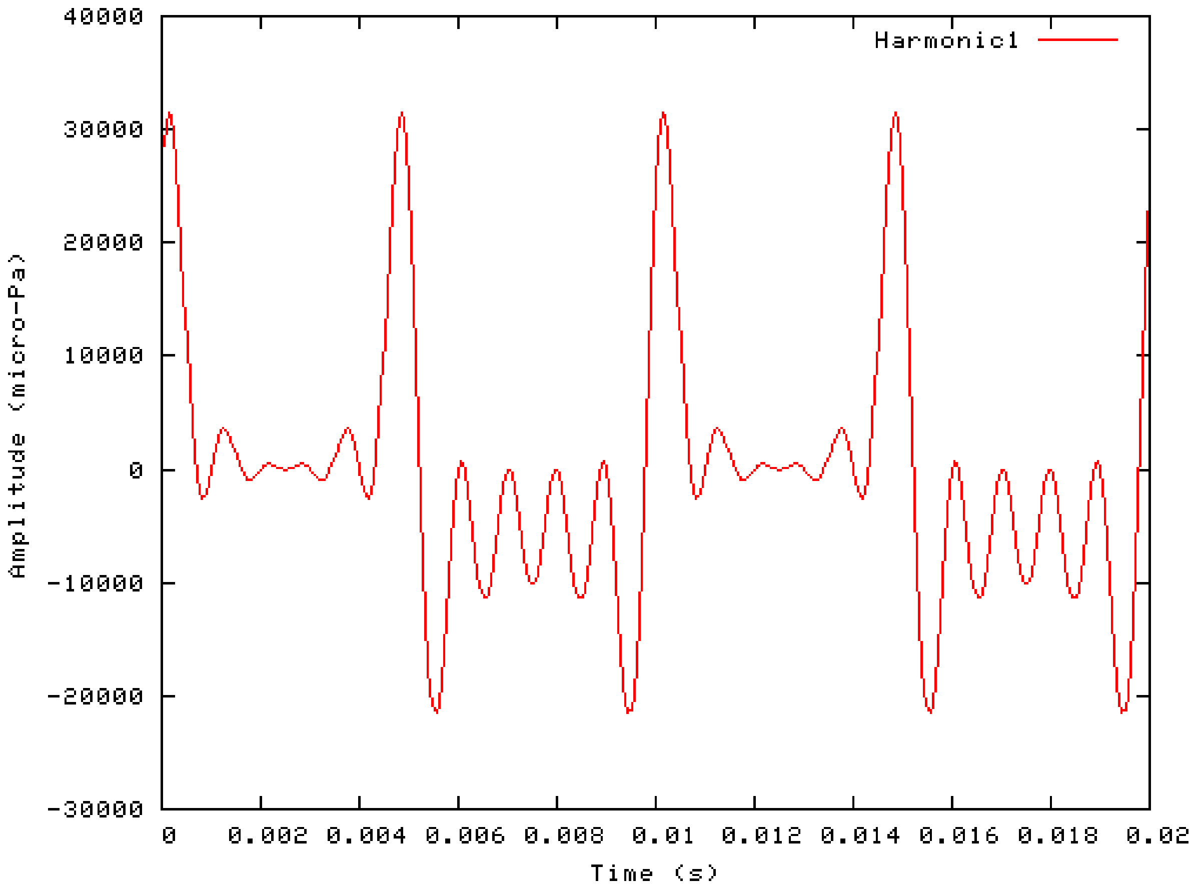
Phase_Mode	Description
sine	The harmonics are all in sine phase.
cosine	The harmonics are all in cosine phase.
random	The harmonics are in random phase. In this mode the **phase_par** value is used as the random number seed. It should be set to zero for a completely random phase on each run. Set to a negative number so that the random number seed is initialised to the same series each time.
Shroeder	The harmonics are in Shroeder phase: phase = C*PI *n*(n – 1) / N. C is the **phase_par** parameter in this mode, ‘n’ is the harmonic number and ‘N’ is the total number of harmonics.
alternating	The harmonics alternate between zero and PI / 2 (90 degrees).

Examples Using the ’Stim_Harmonic’ Process Module

Example: AutoTest/St/Harmonic1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/Harmonic1.sim
begin  {

 Stim_Harmonic   < Harmonic1.par
 Display_Signal

}
```

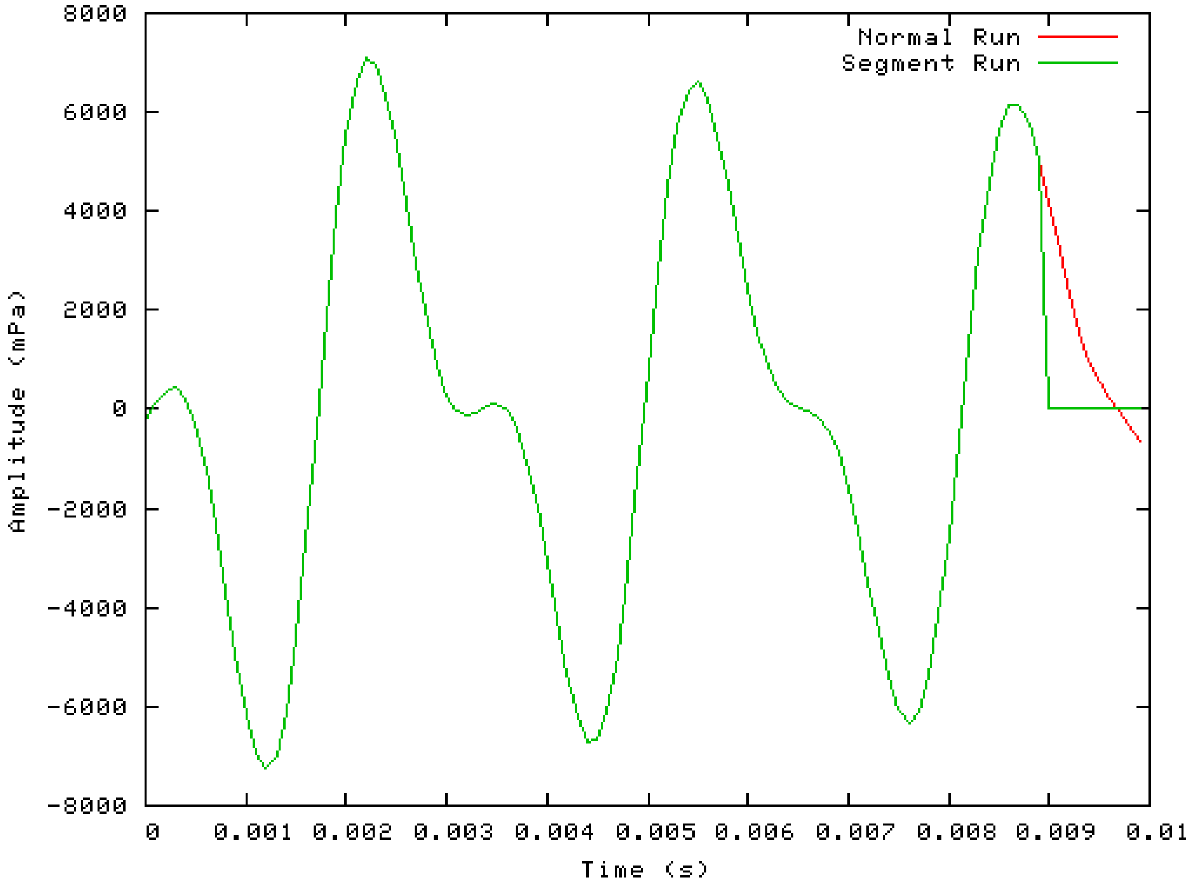
Figure 12.5. A stimulus generated by the Stim_Harmonic module.



Segmented Mode Example: AutoTest/St/Harmonic2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/Harmonic2.sim
begin  {

 Stim_Harmonic    < Harmonic2.par
 Display_Signal

}
```

Figure 12.6. Harmonic stimulus behaviour in *segmented rocessing* mode. The full stimulus is generated once in normal mode, then the stimulus is generated in segments in *segment-processing* mode. Note that in *segmented-processing* mode the stimulus is repeated.



Stim_PulseTrain: Pulse Train Stimulus Module

File name: StPulseTrain.[ch]

Description

This module generates a stimulus consisting of a train of pulses.

Inputs	None.
Outputs	Single channelled output.
Reference	

Module Parameters

Example 12.4. Stim_Harmonic Parameter File

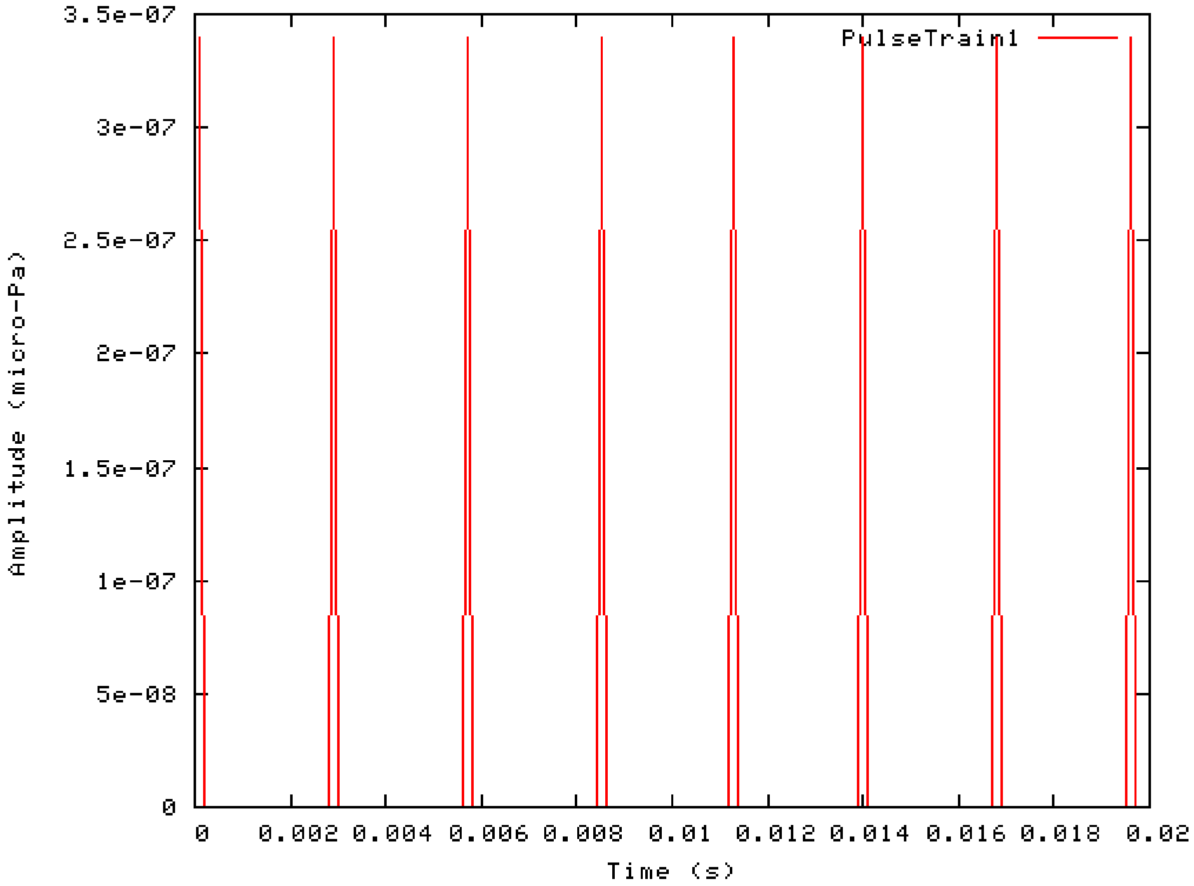
RATE	360	Pulse rate (pulses/s)
PULSE_DURATION	0.0001	Pulse duration (s).
AMPLITUDE	3.4e-07	Amplitude (arbitary units).
DURATION	0.1	Duration (s).
DT	1e-05	Sampling interval, dt (s).

Examples Using the ’Stim_Harmonic’ Process Module

Example: AutoTest/St/PulseTrain1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/PulseTrain1.sim
begin  {

 Stim_PulseTrain  < PulseTr1.par
 Display_Signal

}
```

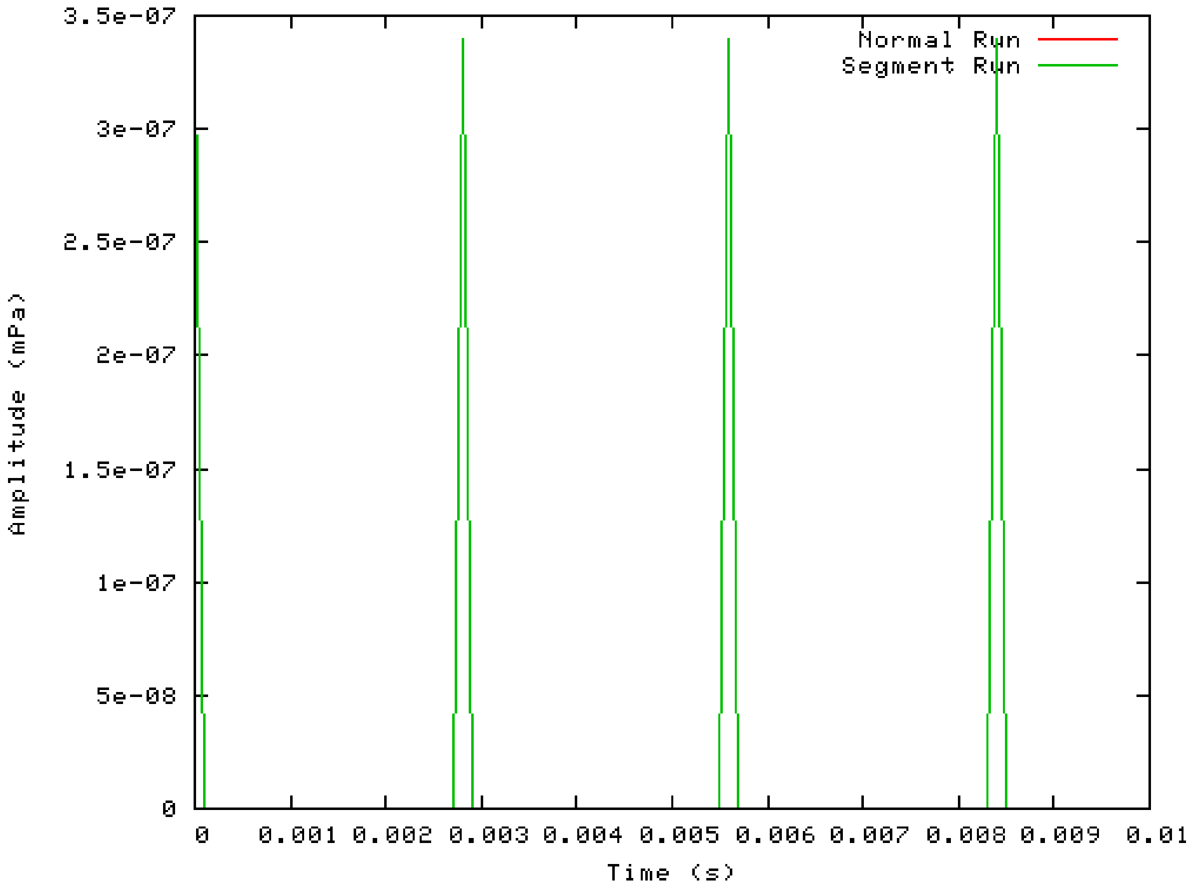
Figure 12.7. A stimulus generated by the Stim_PulseTrain module.



Segmented Mode Example: AutoTest/St/PulseTrain2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.
# ./St/PulseTrain2.sim
begin {

 Stim_PulseTrain  < PulseTr1.par
 Display_Signal

}
```

Figure 12.8. PulseTrain stimulus behaviour in *segmented rocessing* mode. The full stimulus is generated once in normal mode, then the stimulus is generated in segments in *segment-processing* mode.



Stim_PureTone: Pure Tone Stimulus Module

File name: StPTone.[ch]

Description

Simple pure tone stimulus generation module.

Inputs	None.
Outputs	Single channelled output.
Reference	

Module Parameters

Example 12.5. Stim_PureTone Parameter File

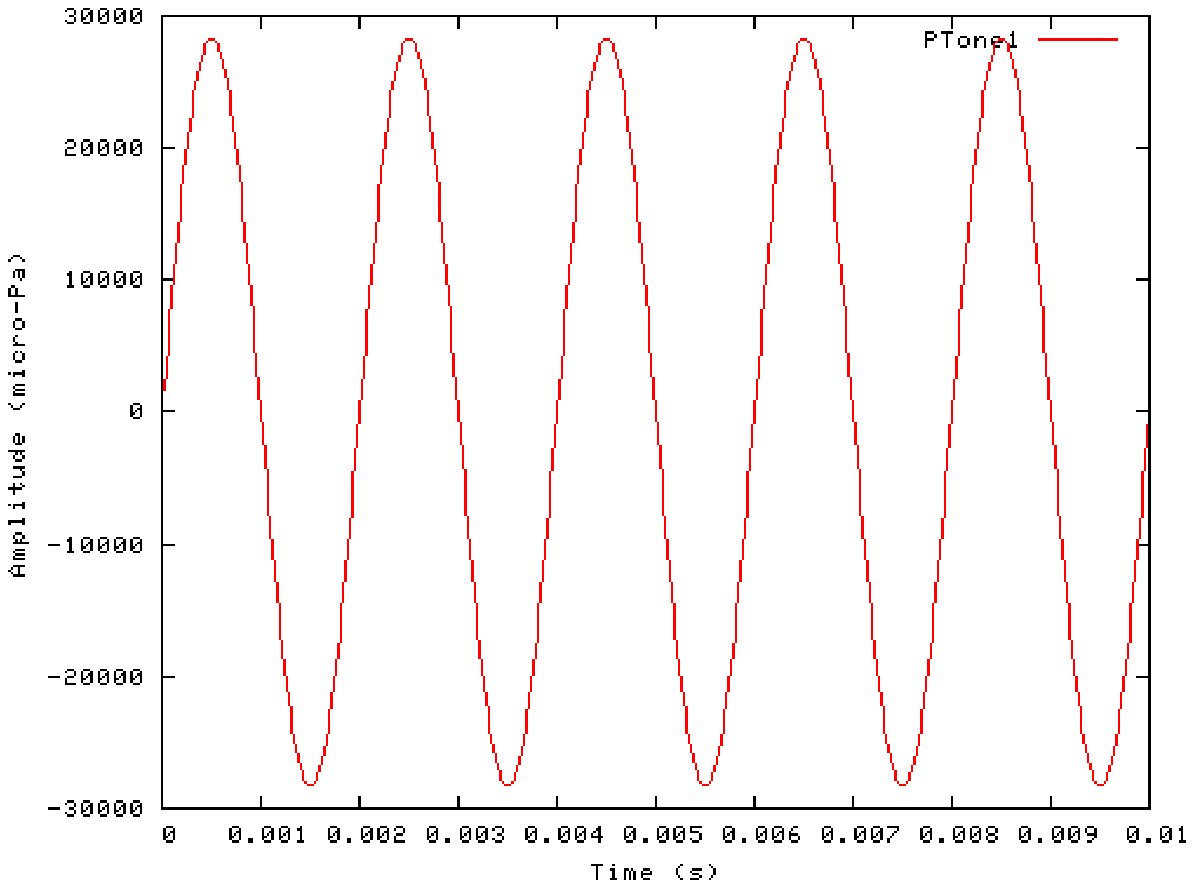
FREQUENCY	1000	Frequency (Hz).
INTENSITY	56	Intensity (dB SPL).
DURATION	0.1	Duration (s).
DT	1e-05	Sampling interval, dt (s).

Examples Using the ’Stim_PureTone’ Process Module

Example: AutoTest/St/PTone1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/PTone1.sim
begin  {

 Stim_PureTone   < PTone1.par
 Display_Signal

}
```

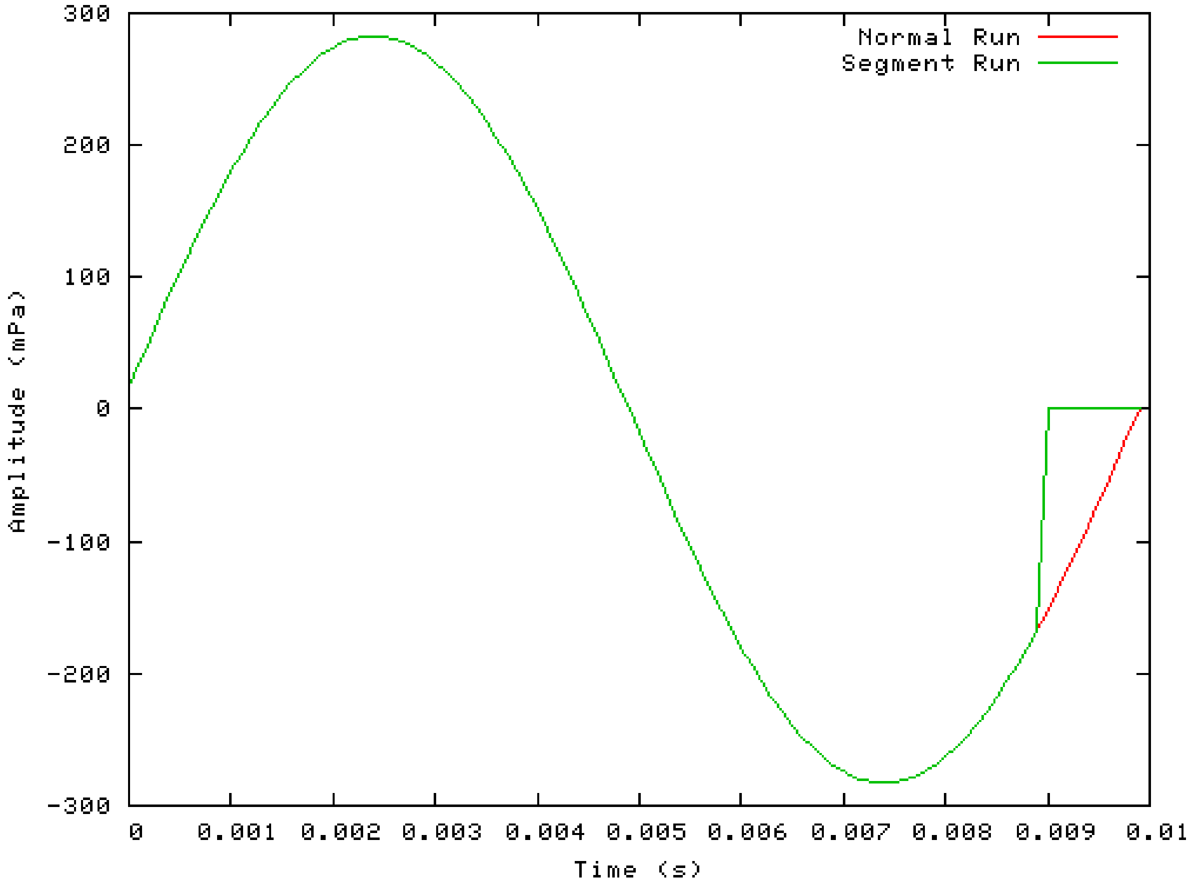
Figure 12.9. A stimulus generated by the Stim_PureTone module.



Segmented Mode Example: AutoTest/St/PTone2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/PTone2.sim
begin  {

 Stim_PureTone    < PTone2.par
 Display_Signal

}
```

Figure 12.10. PureTone stimulus behaviour in *segmented rocessing* mode. The full stimulus is generated once in normal mode, then the stimulus is generated in segments in *segment-processing* mode.



Stim_PureTone_2: Pure Tone Stimulus with Starting and Ending Periods Module

File name: StP2Tone.[ch]

Description

This module generates a pure tone, preceded and ended by periods of constant amplitudes. e.g. the beginning and ending periods could be set to zero amplitude for silence.

Inputs	None.
Outputs	Single channelled output.
Reference	

Module Parameters

Example 12.6. Stim_PureTone_2 Parameter File

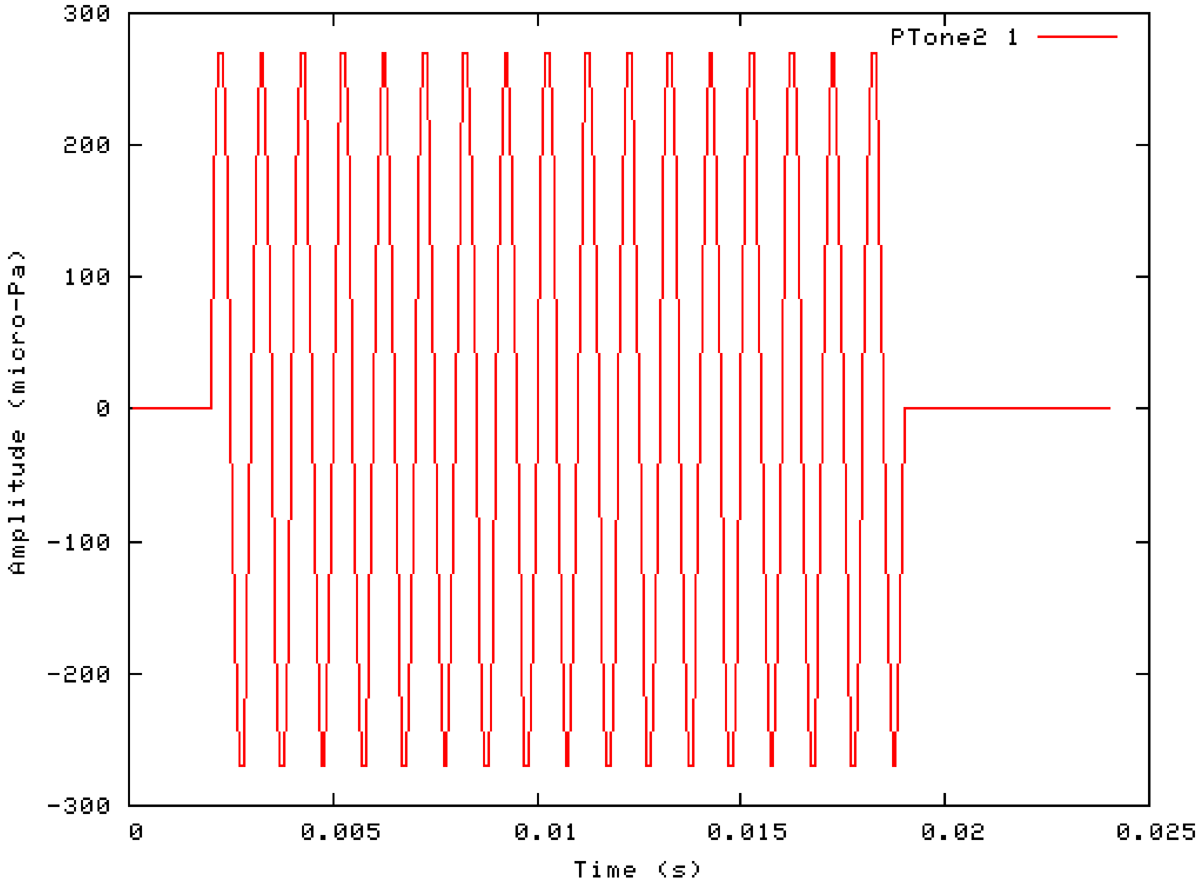
FREQUENCY	1000	Frequency (Hz).
INTENSITY	56	Intensity (dB SPL).
DURATION	0.08	Duration (s).
DT	1e-05	Sampling interval, dt (s).
BEGIN_SILENCE	0.01	Silence period before the signal begins (s).
END_SILENCE	0.01	Silence period after the signal ends (s).

Examples Using the ’Stim_PureTone_2’ Process Module

Example: AutoTest/St/PTone2_1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/PTone2_1.sim
begin  {

 Stim_PureTone_2  < P2Tone1.par
 Display_Signal

}
```

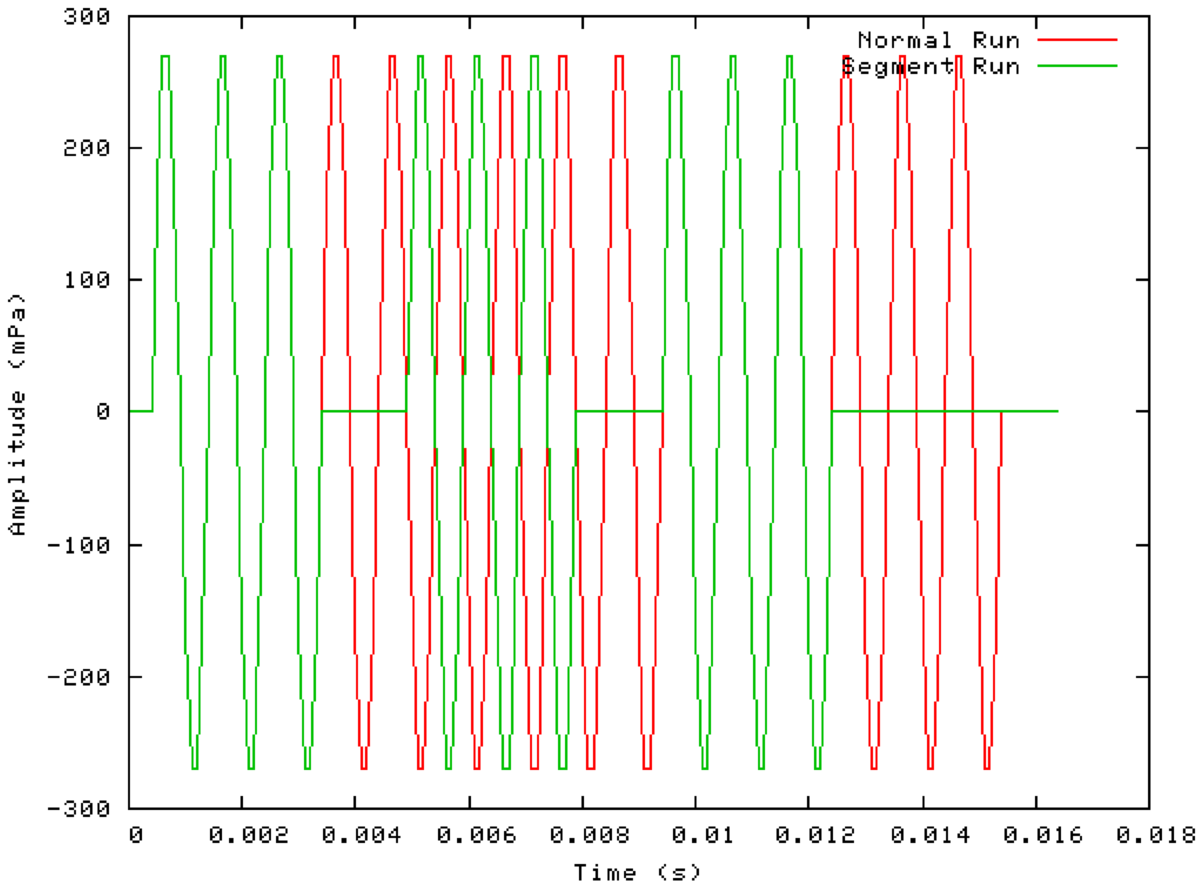
Figure 12.11. A stimulus generated by the Stim_PureTone_2 module.



Segmented Mode Example: AutoTest/St/PTone2_2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/PTone2_2.sim
begin  {

 Stim_PureTone_2  < P2Tone2.par
 Display_Signal

}
```

Figure 12.12. PureTone 2 stimulus behaviour in *segmented rocessing* mode. The full stimulus is generated once in normal mode, then the stimulus is generated in segments in *segment-processing* mode. Note that in *segmented-processing* mode the stimulus is repeated.



Stim_PureTone_AM: Amplitude Modulated Pure Tone Stimulus Module

File name: StAMTone.[ch]

Description

Amplitude modulated pure tone generation algorithm.

Inputs	None.
Outputs	Single Channelled output.
Reference	

Module Parameters

Example 12.7. Stim_PureTone_AM Parameter File

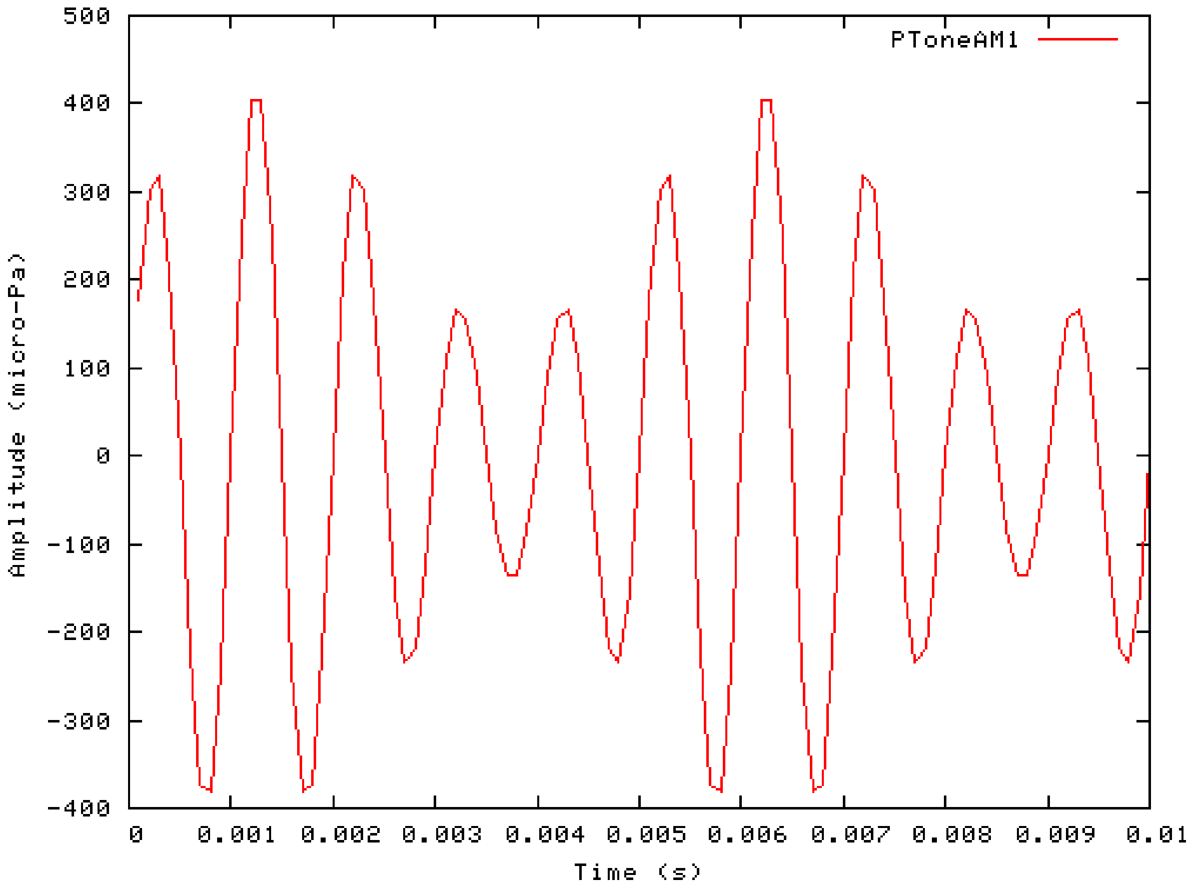
CARRIER_FREQ	1000	Carrier frequency (Hz).
MOD_FREQUENCY	50	Modulation frequency (Hz).
DEPTH	100	Amplitude modulation depth (%).
INTENSITY	56	Intensity (dB SPL).
DURATION	0.1	Duration (s).
DT	1e-06	Sampling interval, dt (s).

Examples Using the ’Stim_PureTone_AM’ Process Module

Example: AutoTest/St/PToneAM1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/PToneAM1.sim
begin  {

 Stim_PureTone_AM < AMPTone1.par
 Display_Signal

}
```

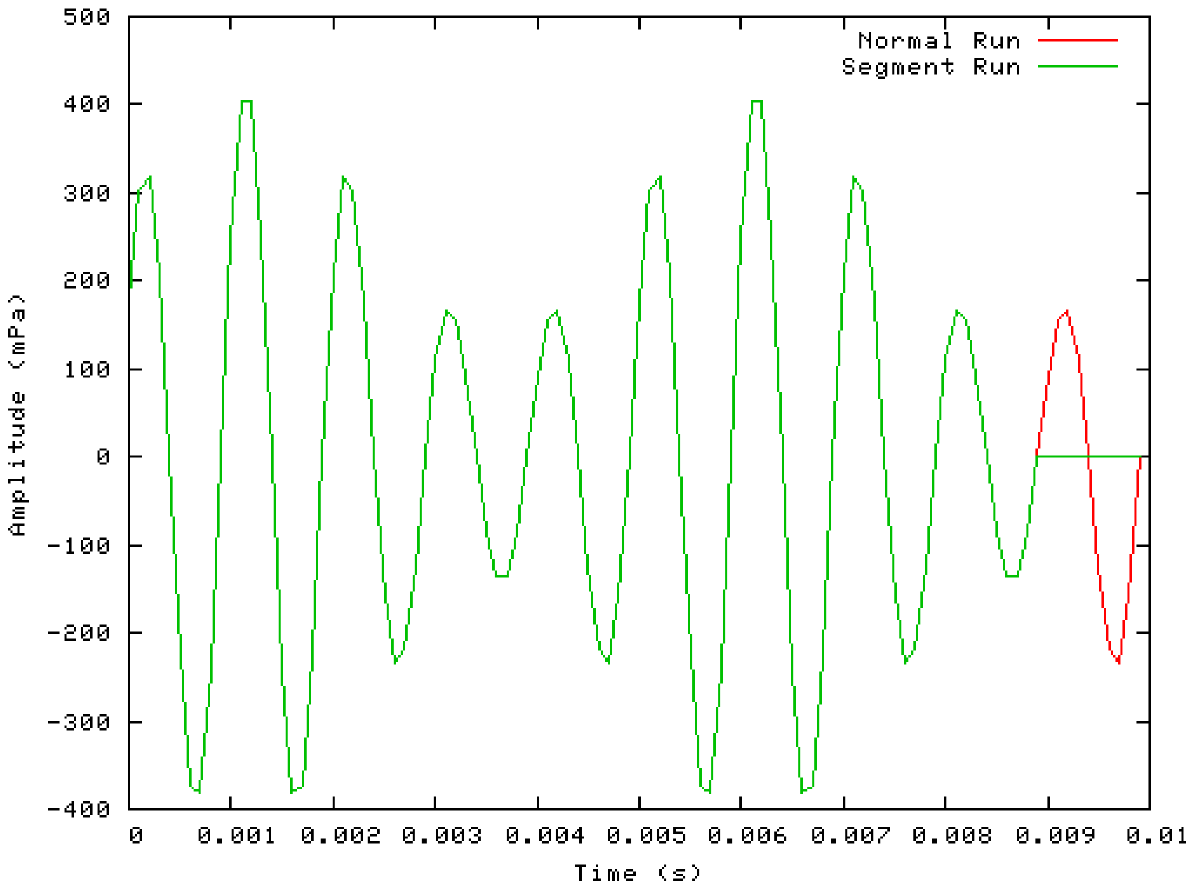
Figure 12.13. A stimulus generated by the Stim_PureTone_AM module.



Segmented Mode Example: AutoTest/St/PToneAM2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/PToneAM2.sim
begin  {

 Stim_PureTone_AM < AMPTone1.par
 Display_Signal

}
```

Figure 12.14. Amplitude modulated pure tone stimulus behaviour in *segmented rocessing* mode. The full stimulus is generated once in normal mode, then the stimulus is generated in segments in *segment-processing* mode.



Stim_PureTone_Binaural: Binaural Pure Tone Stimulus Module

File name: StBPTone.[ch]

Description

This module generates a binaural pure tone stimulus.

Inputs	None.
Outputs	Binaural (two-channelled) output
Reference	

Module Parameters

Example 12.8. Stim_PureTone_Binaural Parameter File

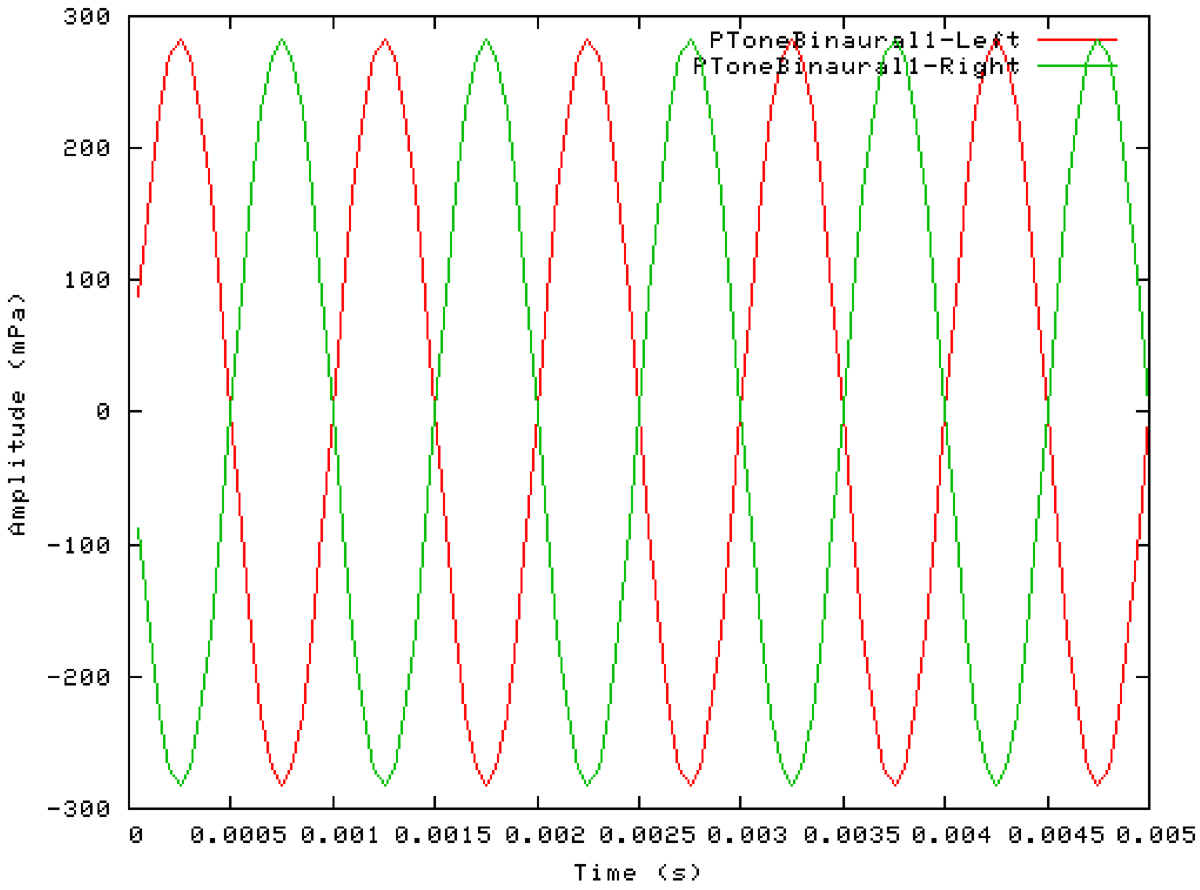
LEFT_FREQUENCY	1000	Left channel frequency (Hz).
RIGHT_FREQUENCY	1000	Right channel frequency (Hz).
LEFT_INTENSITY	56	Left channel intensity (dB SPL).
RIGHT_INTENSITY	56	Right channel intensity (dB SPL).
PHASE_DIFF	180	Phase difference between channels (degrees).
DURATION	0.1	Duration (s).
DT	1e-05	Sampling interval, dt (s).

Examples Using the ’Stim_PureTone_AM’ Process Module

Example: AutoTest/St/PToneBinaural1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/PToneAM1.sim
begin {

 Stim_PureTone_AM < AMPTone1.par
 Display_Signal

}
```

Figure 12.15. A stimulus generated by the Stim_PureTone_Binaural module.



Stim_PureTone_FM: Frequency Modulated Pure Tone Stimulus Module

File name: StFMTone.[ch]

Description

This program generates a frequency modulated (FM) pure tone.

Inputs	None.
Outputs	Single channelled output.
Reference	

Module Parameters

Example 12.9. Stim_PureTone_FM Parameter File

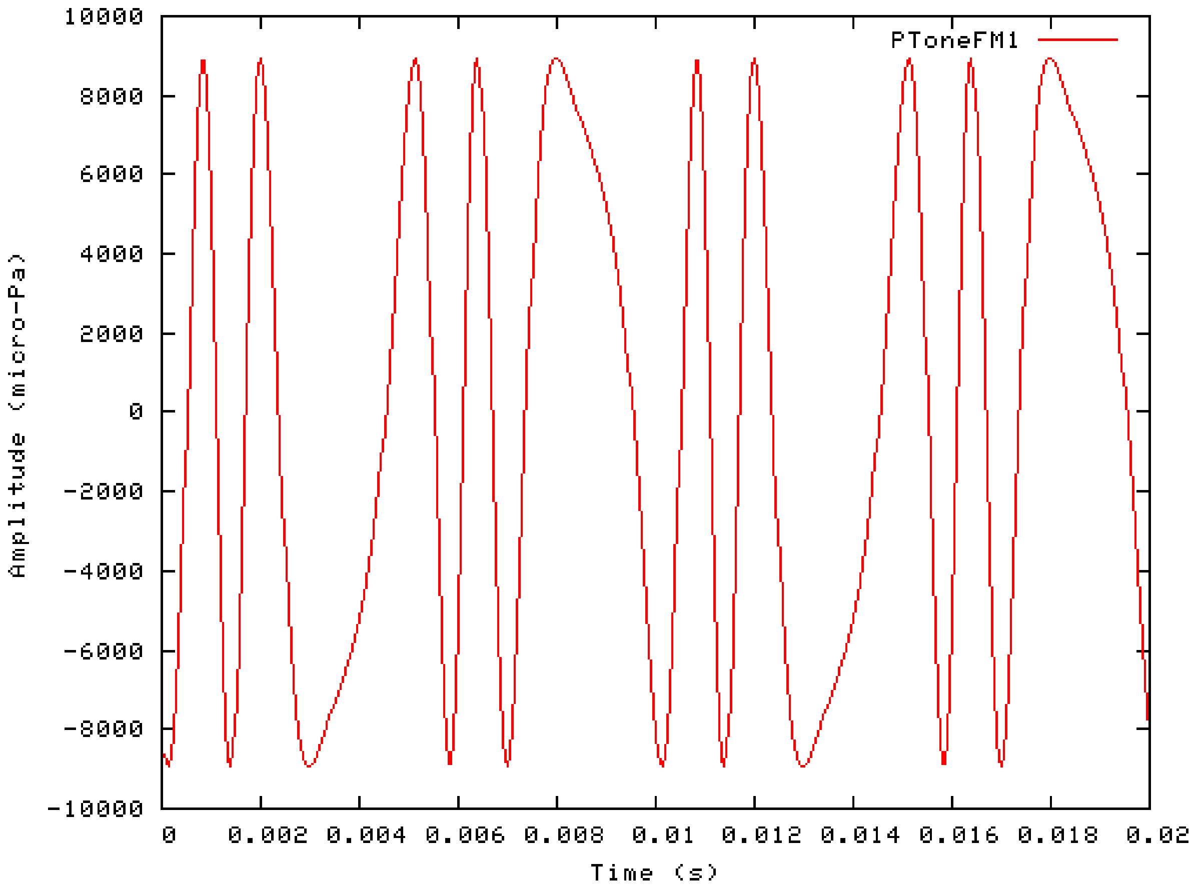
FREQUENCY	1000	Frequency (Hz).
INTENSITY	56	Intensity (dB SPL).
PHASE	0	Phase (degrees).
MOD_DEPTH	80	Modulation depth (%).
MOD_FREQ	2000	Modulation frequency (Hz).
MOD_PHASE	0	Modulation phase (degrees).
DURATION	0.1	Duration (s).
DT	1e-05	Sampling interval, dt (s).

Examples Using the ’Stim_PureTone_AM’ Process Module

Example: AutoTest/St/PToneFM1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/PToneFM1.sim
begin  {

 Stim_PureTone_FM < FMPTone1.par
 Display_Signal

}
```

Figure 12.16. A stimulus generated by the Stim_PureTone_FM module.



Stim_PureTone_Multi: Multiple Pure Tone Stimulus Module

File name: StMPTone.[ch]

Description

This module generates a multiple pure tone stimulus. Each pure tone is specified individually.

Inputs	None.
Outputs	Single channelled output.
Reference	

Module Parameters

Example 12.10. Stim_PureTone_Multi Parameter File

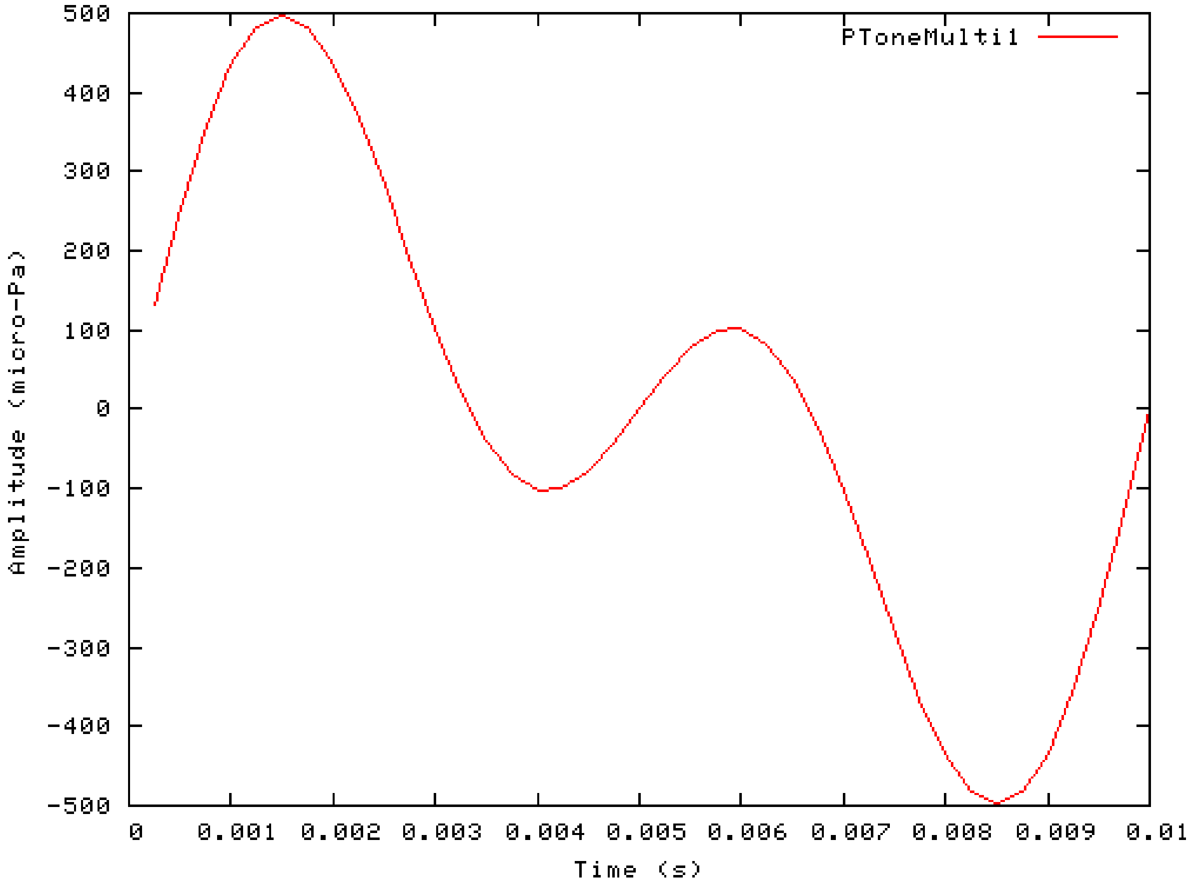
NUM	2	Number of pure tones.
FREQUENCY	0:100	
FREQUENCY	1:200	
INTENSITY	0:56	
INTENSITY	1:56	
PHASE	0:0	
PHASE	1:0	
DURATION	0.1	Duration (s).
DT	1e-05	Sampling interval, dt (s).

Examples Using the ’Stim_PureTone_Multi’ Process Module

Example: AutoTest/St/PToneMulti1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/PToneAM1.sim
begin  {

 Stim_PureTone_AM < AMPTone1.par
 Display_Signal

}
```

Figure 12.17. A stimulus generated by the PureTone_Multi module. This stimulus consists of individually specified pure tones which are added together to form a composite signal.



Stim_PureTone_MultiPulse: Multiple Pure Tone Pulse Stimulus Module

File name: StMPPTone.[ch]

Description

The module generates a signal that contains multiple pure tone pulses at different frequencies.

Inputs	None.
Outputs	Single-channelled output.
Reference	

Module Parameters

Example 12.11. Stim_PureTone_MultiPulse Parameter File

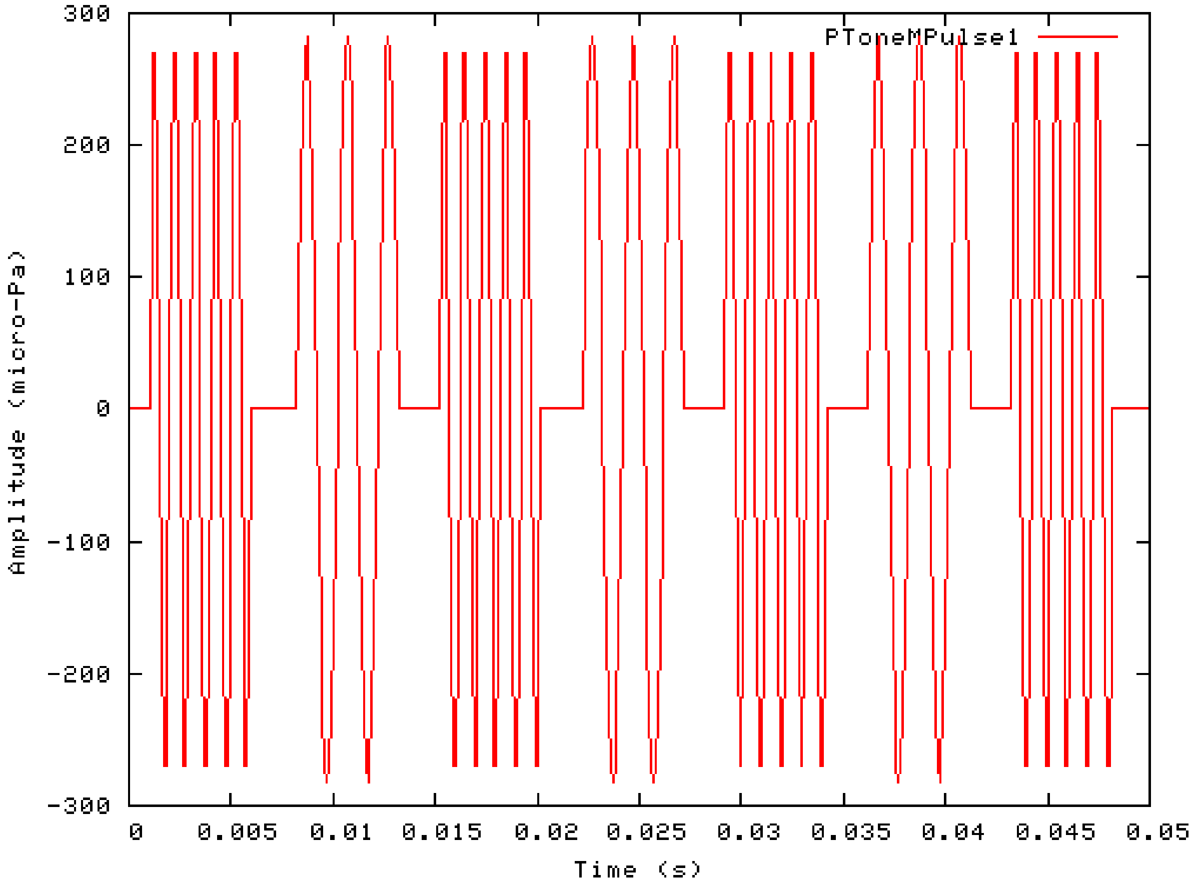
NUM_PULSES	2	Number of Pulse Frequencies
PULSE_FREQ	0:1000	
PULSE_FREQ	1:2000	
INTENSITY	56	Intensity (dB SPL).
SILENCE	0.01	Silent period, before first pulse (s).
PULSE_DURATION	0.005	Pulse duration (s).
REP_PERIOD	0.007	Repetition period (s).
DURATION	0.08	Duration (s).
DT	0.0001	Stimulus sampling interval, dt (s).

Examples Using the ’Stim_PureTone_MultiPulse’ Process Module

Example: AutoTest/St/PToneMPulse1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/PToneMPulse1.sim
begin  {

 Stim_PureTone_MultiPulse < MPPTone1.par
 Display_Signal

}
```

Figure 12.18. A stimulus generated by the PureTone_MultiPulse module. The stimulus consists of a series of pulses, indivudually specified, which are repeated. The figure illustrates a stimulus which has a series of two pulses of different frequency.



Stim_StepFun: Step function Stimulus Module

File name: StStepFun.[ch]

Description

T his module generates a step function stimulus.

Inputs	None.
Outputs	Single channelled output.
Reference	

Module Parameters

Example 12.12. Stim_StepFun Parameter File

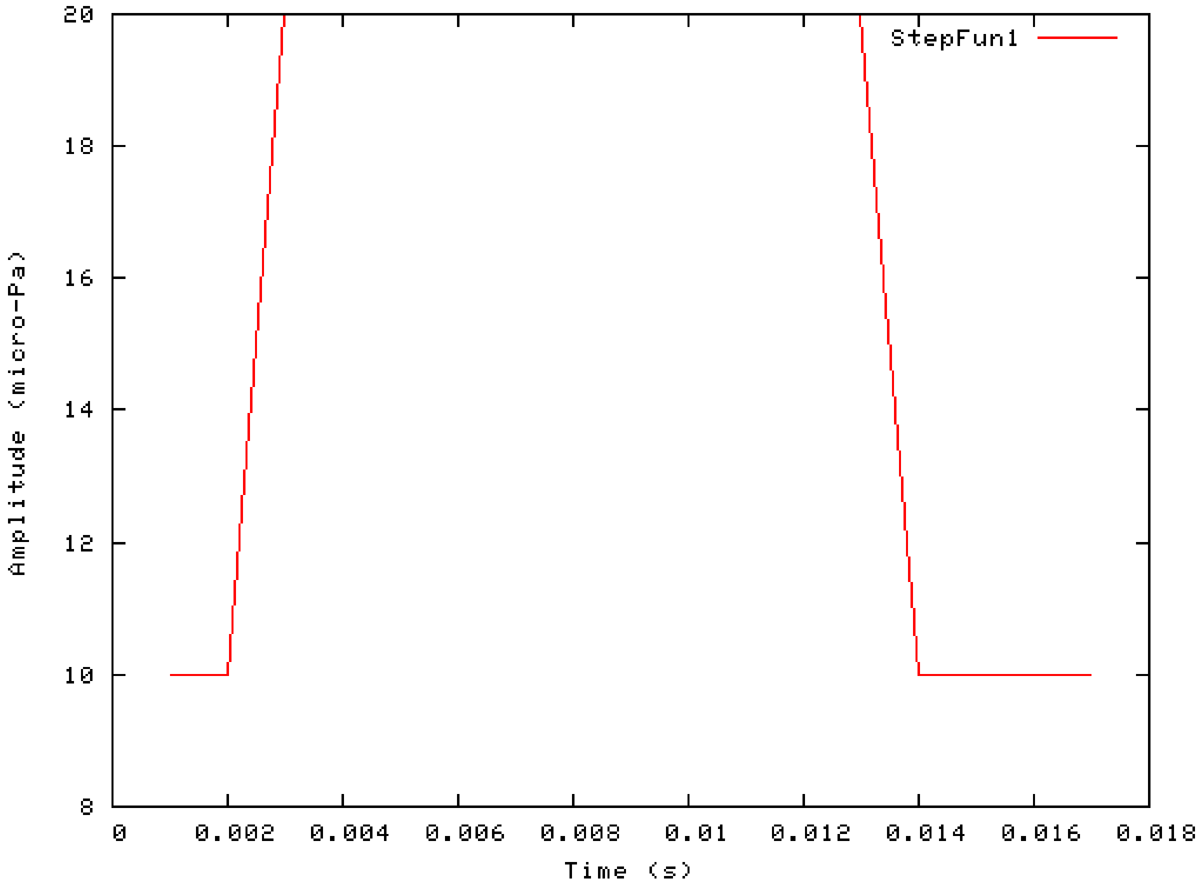
AMPLITUDE	30	Amplitude (arbitrary units).
B_E_AMP	0	Begin-end period amplitude (arbitrary units).
BEGIN_PERIOD	0.01	Period before the signal begins (s).
END_PERIOD	0.01	Period after the signal ends (s).
DURATION	0.08	Duration (s).
DT	1e-05	Sampling interval, dt (s).

Examples Using the ’Stim_StepFun’ Process Module

Example: AutoTest/St/StepFun1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/StepFun1.sim
begin  {

 Stim_StepFun    < StepFun1.par
 Display_Signal

}
```

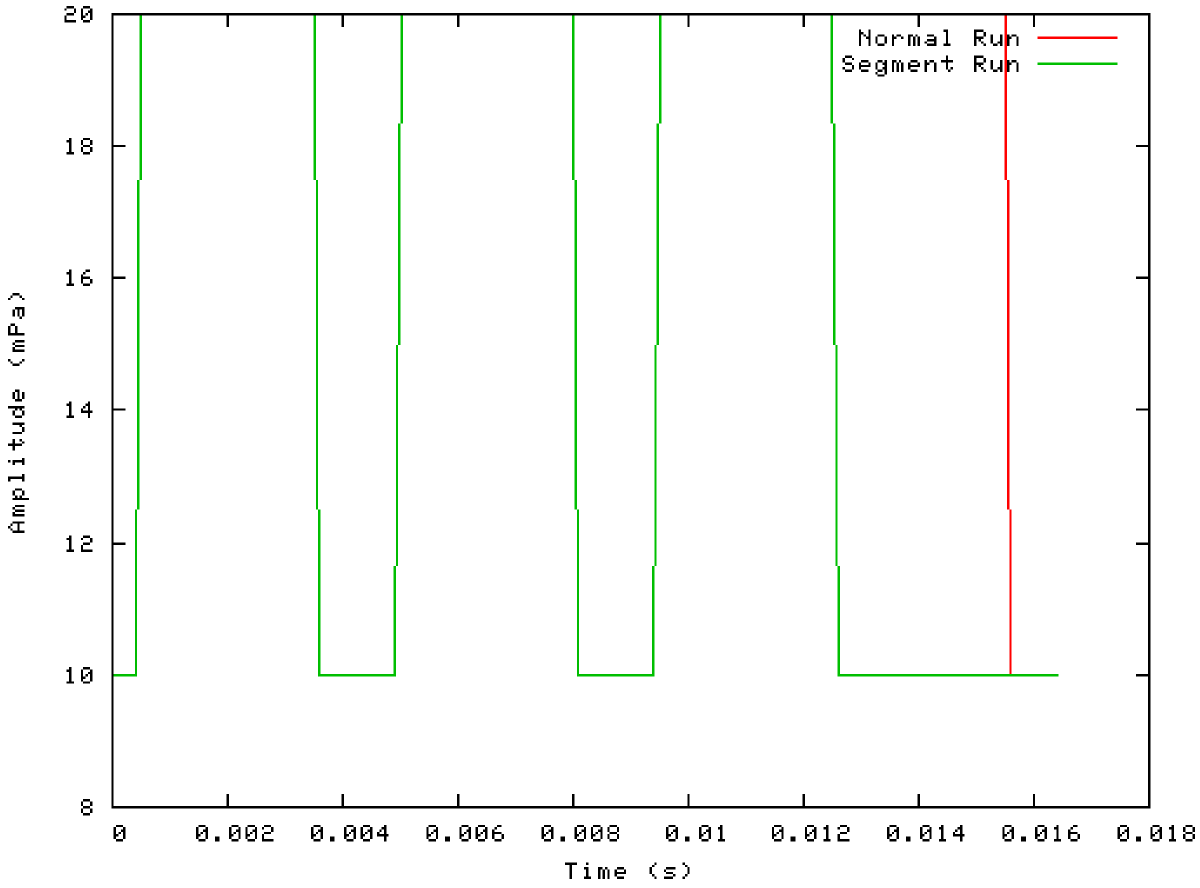
Figure 12.19. A stimulus generated by the Stim_StepFun module. This stimulus has been used where a current pulse is required, though it can be applied to other cases.



Segmented Mode Example: AutoTest/St/StepFun2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/StepFun2.sim
begin  {

 Stim_StepFun     < StepFun2.par
 Display_Signal

}
```

Figure 12.20. StepFun stimulus behaviour in *segmented rocessing* mode. The full stimulus is generated once in normal mode, then the stimulus is generated in segments in *segment-processing* mode. Note that in *segmented-processing* mode the stimulus is repeated.



Stim_WhiteNoise: White Noise Stimulus Module

File name: StWhiteNoise.[ch]

Description

This module generates a monaural or binaural white noise stimulus.

Inputs	None.
Outputs	Single channelled output.
Reference	

Module Parameters

Example 12.13. Stim_WhiteNoise Parameter File

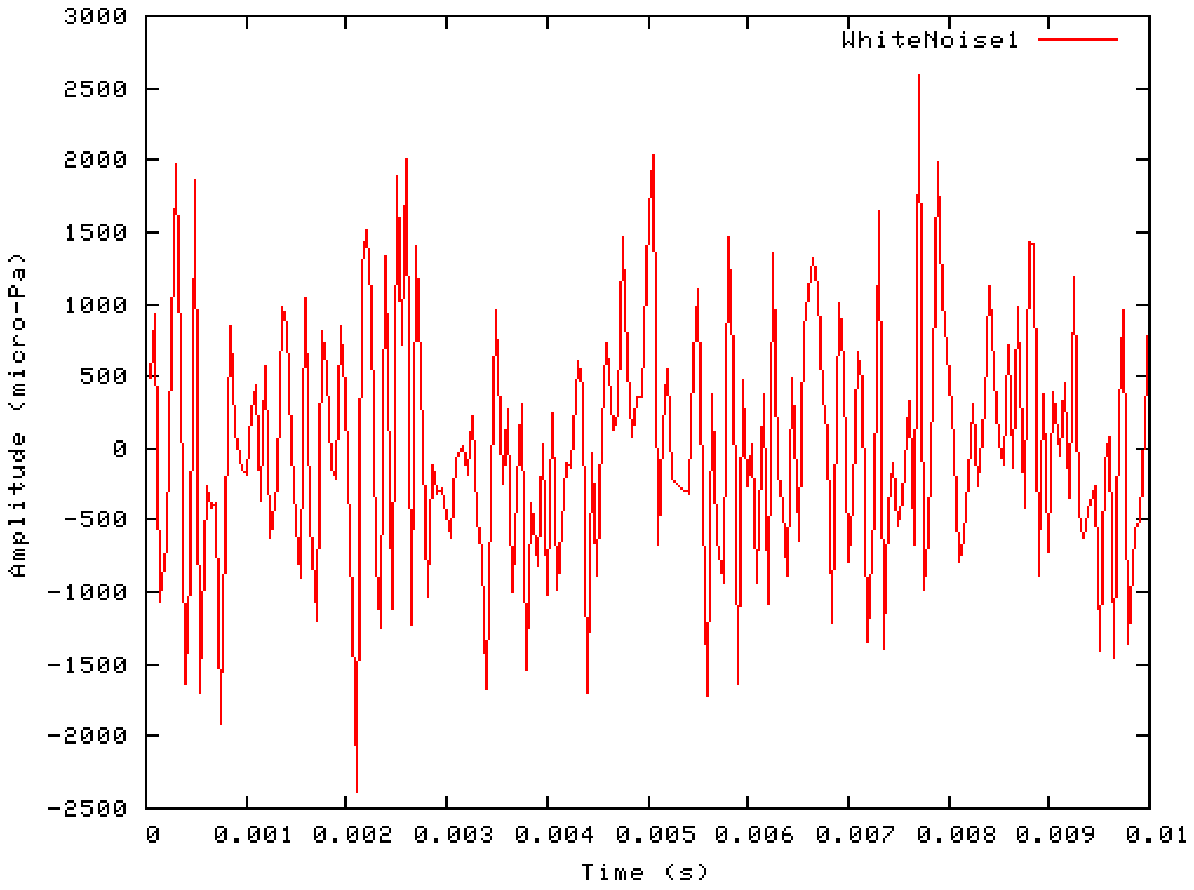
NUM_CHANNELS	1	Number of Sound Channels (Binaural = 2; Monaural = 1).
DEGREE_CORR	1	Degree of Binaural noise correlation (Correlated = 1; Uncorrelated = -1).
RAN_INDEX	36	Noise Randomization Index (greater than 12).
RAN_SEED	0	Random Number Seed.
INTENSITY	56	Intensity (dB SPL).
DURATION	0.1	Duration (s).
DT	1e-05	Sampling Interval, dt (s).

Examples Using the ’Stim_WhiteNoise’ Process Module

Example: AutoTest/St/WhiteNoise1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./St/WhiteNoise1.sim
begin  {

 Stim_WhiteNoise  < WNoise1.par
 Display_Signal

}
```

Figure 12.21. A stimulus generated by the Stim_WhiteNoise module.


WhiteNoise1
Amplitude (micro-Pa)
3000
2500
2000
1500
1000
500
0
-500
-1000
-1500
-2000
-2500
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
Time (s)

Chapter 13. Transform Modules

Introduction

The *transform process* modules are a later addition to the DSAM library. Unlike the other process modules, transform modules do not create a signal but change or *transform* an existing one. For example, if a stimulus is ramped using the Trans_Gate module, then unless the stimulus generation is repeated, any process connected to the stimulus process will use the transformed, ramped signal. The use of a transform modules makes no significant difference when using simulation scripts.

Trans_CollectSignals: Channel collection Transform Module

File name: TrCollectSignals.[ch]

Description

This process module collects the channels from multiple input signals. The output is a single multi-channel signal with the number of channels equal to the sum of all the input signal channels.

Inputs	Multi-signal input.
Outputs	The input signal channels.are rearranged to appear as a single multi-channel signal.
Reference	

Module Parameters

Example 13.1. Trans_CollectSignals Parameter File

```
MODE    CHAN_INDEX    Channel labelling mode ('chan_index', 'input_labels' or 'user').
```

Examples Using the 'Trans_CollectSignals' Process Module

Example: AutoTest/Tr/CollectSignals1.sim

```
begin {

s1% Stim_PureTone (->cc) < PTone2.par
s2% Stim_Click (->bm1) < Click1.par
bm1% BM_GammaT (s2->cc) < GammaTLog1.par
s3% Stim_Harmonic (->cc) < Harmonic1.par
cc% Trans_CollectSignals (s1, bm1, s3->)
 Display_Signal

}
```

Example: AutoTest/Tr/CollectSignals2.sim

```
begin {

s1% Stim_PureTone (->cc) < PTone2.par
s2% Stim_Click (->bm1) < Click1.par
bm1% BM_GammaT (s2->cc) < GammaTLog1.par
s3% Stim_Harmonic (->cc) < Harmonic1.par
cc% Trans_CollectSignals (s1, bm1, s3->) < CollectSigs1.par
 Display_Signal

}
```

Example: AutoTest/Tr/CollectSignals3.sim

```
begin {

s1% Stim_PureTone (->cc) < PTone2.par
s2% Stim_Click (->bm1) < Click1.par
bm1% BM_GammaT (s2->cc) < GammaTLog1.par
s3% Stim_Harmonic (->cc) < Harmonic1.par
cc% Trans_CollectSignals (s1, bm1, s3->) < CollectSigs2.par
 Display_Signal

}
```

Trans_Damp: Damping Transform Module

File name: TrGate.[ch]

Description

When this process module is used the Trans_Gate process module is called in reduced operation mode.

Inputs	Arbitrary signal input.
Outputs	Transforms the input signal and passes on.
Reference	

Module Parameters

Example 13.2. Trans_Damp Parameter File

POS_MODE	RELATIVE	Position mode (’absolute’ or ’relative’)
OP_MODE	RAMP	Operation mode (’ramp’ or ’damp’)
TYPE_MODE	RAISED_COS	Type mode (’linear’, ’sine’, ’raised_cos’ or ’exp_decay’)
OFFSET	0	Time offset (s)
DURATION	0.0025	Ramp duration - negative assumes to the end of the signal (s)
SLOPE	0.016	Slope parameter or half-life (in ’decay_par’ mode only)

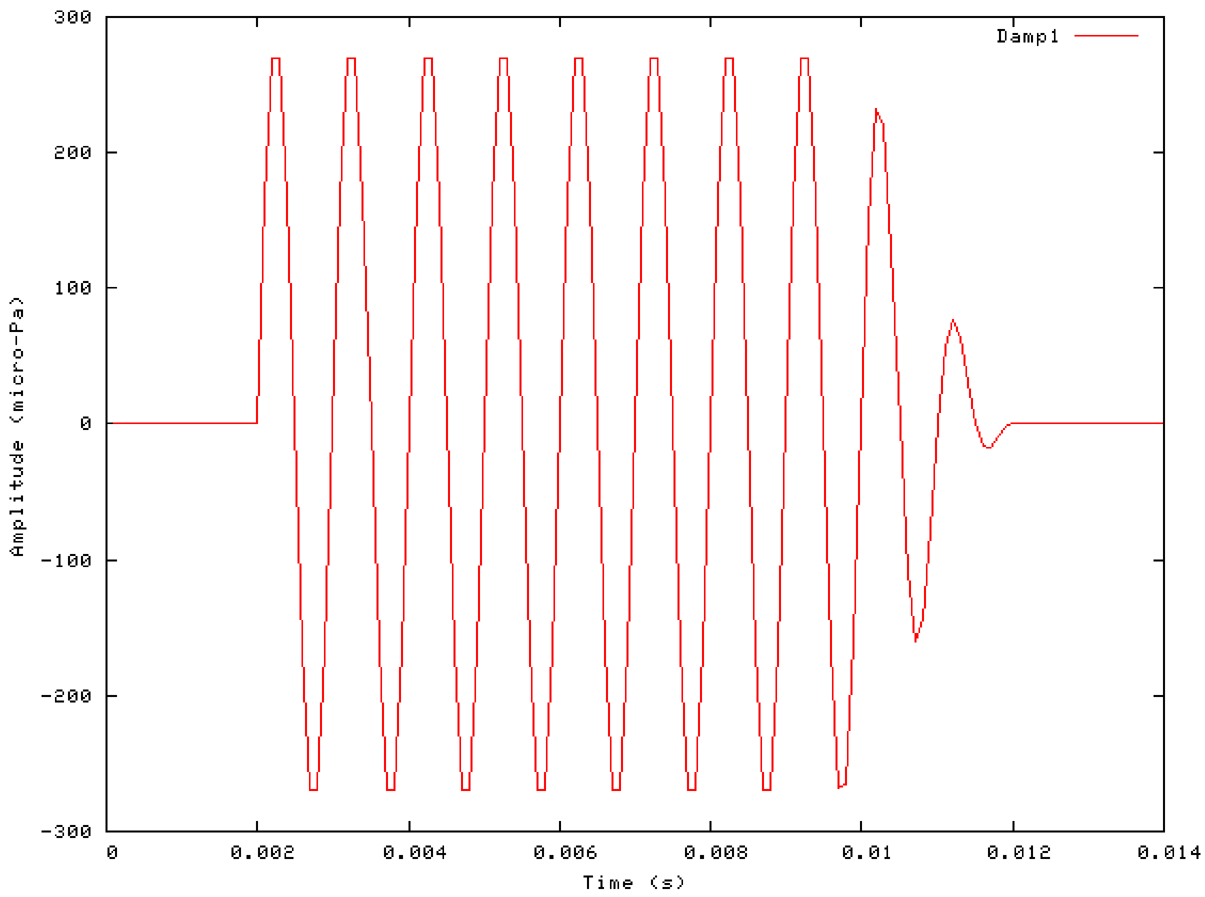
See the section called “Trans_Gate: Gating Transform Module” for the parameter options. Note that the use of the process module implies that the *op_mode* parameter is set to *damp* and cannot be changed.

Examples Using the ’Trans_Damp’ Process Module

Damped Example: AutoTest/Tr/Damp1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Tr/Damp1.sim
begin {

 Stim_PureTone_2  < P2Tone1.par
 Trans_Damp     < Damp4.par
 Display_Signal

}
```

Figure 13.1. Damped Sinusoid



Trans_Gate: Gating Transform Module

File name: TrGate.[ch]

Description

This is the general signal gating transformation module. It changes the start or end envelope of a signal using a selection of ramping, or damping functions. The ramp functions are defined as a function that can vary from 0 to 1 as a function of the ramp interval.

Impulse signals, such as clicks, should not be ramped; if a warning message is given, in this case ignore it. If a signal is ramped, then its ramp flag will be set. This process will only operate on the first signal section in segmented processing mode.

Inputs	Arbitrary signal input.
Outputs	Transforms the input signal and passes on.
Reference	

Module Parameters

Example 13.3. Trans_Gate Parameter File

POS_MODE	RELATIVE	Position mode ('absolute' or 'relative')
OP_MODE	RAMP	Operation mode ('ramp' or 'damp')
TYPE_MODE	RAISED_COS	Type mode ('linear', 'sine', 'raised_cos' or 'exp_decay')
OFFSET	0	Time offset (s)
DURATION	0.0025	Ramp duration - negative assumes to the end of the signal (s)
SLOPE	0.016	Slope parameter or half-life (in 'decay_par' mode only)

Table 13.1. Trans_Gate: Position mode options.

Pos_Mode	Description
absolute	In this mode the signal is 'ramped' forwards from the **offset** time and 'damped' forwards to the **offset** time (effectively the damped option is a ramp backwards from the **offset** time).
relative	

Table 13.2. Trans_Gate: Op_Mode options.

Op_mode	Description
ramp	Ramp (raise) the signal forwards in time using the specified **type_mode** for the period specified by the **duration** parameter.
damp	Damp (lower) the signal forwards in time using the specified **type_mode** for the period specified by the **duration** parameter (effectively the signal is ramped backwards for the specified **duration**).

Table 13.3. Trans_Gate: Type_mode options.

Type_Mode	Function	Description
Linear	$\frac{t}{\text{interval}}$	This is a simple linear gating function. It varies from 0 to 1 as **t** ranges from 0 to the **interval**.
Exp_decay	$\exp\left(\frac{-dt \cdot t \cdot \ln 2}{slope}\right)$	This is the exponential gating function. It varies from 1 to a lower value, dependent upon the **slope** parameter (half-life), as the **t** ranges from 0 to the **interval**..
Raised_cos	$\frac{1}{2}\cos\left(\pi\frac{t}{\text{interval}}+1\right)$	This is the commonly used raised-cosine gating function. It varies from 0 to 1 as the **t** ranges from 0 to the **interval** length.
Sine	$\sin\left(\frac{t}{2\cdot\text{interval}}\right)$	This is the sine gating function. It varies from 0 to 1 as the **t** ranges from 0 to the **interval** length.

Examples Using the ’Trans_Gate’ Process Module

Absolute Ramp and Damp Examples: AutoTest/Tr/GateAbsRamp1.sim & AutoTest/Tr/GateAbsDamp1.sim

```
# This example simulation script
with its associated parameter files
is in the
# AutoTest package, which can be
downloaded from the DSAM web site.

# ./Tr/GateAbsRamp1.sim
begin  {

 Stim_PureTone_2  < P2Tone1.par
 repeat 10 {
 Trans_Gate       < Ramp1.par
 Display_Signal
 }

}
```

```
# This example simulation script
with its associated parameter files
is in the
# AutoTest package, which can be
downloaded from the DSAM web site.

# ./Tr/GateAbsDamp1.sim
begin  {

 Stim_PureTone_2  < P2Tone1.par
 repeat 10 {
 Trans_Gate       < Damp1.par
 Display_Signal
 }

}
```

Figure 13.2. Absolute Sine Ramp

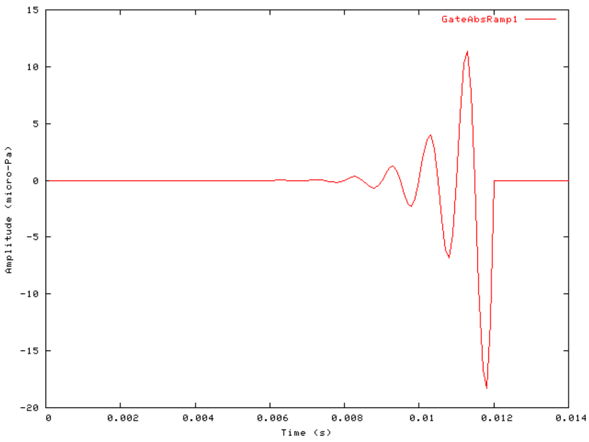
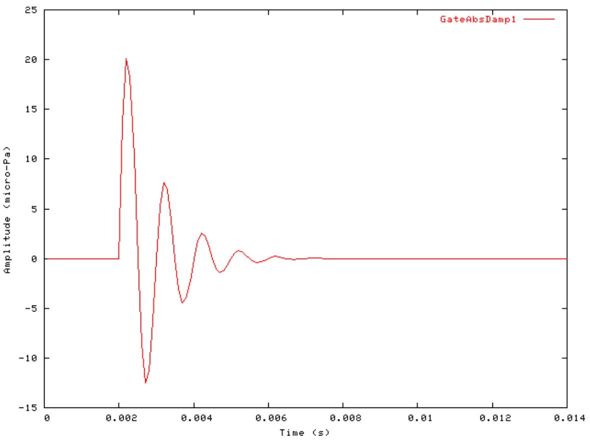


Figure 13.3. Absolute Sine Damp



Relative Ramp and Damp Examples: AutoTest/Tr/GateRelRamp1.sim & AutoTest/Tr/GateRelDamp1.sim

```
# This example simulation script
with its associated parameter files
is in the
# AutoTest package, which can be
downloaded from the DSAM web site.

# ./Tr/GateRelRamp1.sim
begin  {

 Stim_PureTone_2   < P2Tone1.par
 Trans_Gate        < Ramp2.par
 Display_Signal

}
```

```
# This example simulation script
with its associated parameter files
is in the
# AutoTest package, which can be
downloaded from the DSAM web site.

# ./Tr/GateRelDamp1.sim
begin  {

 Stim_PureTone_2   < P2Tone1.par
 Trans_Gate        < Damp2.par
 Display_Signal

}
```

Figure 13.4. Relative Sine Ramp

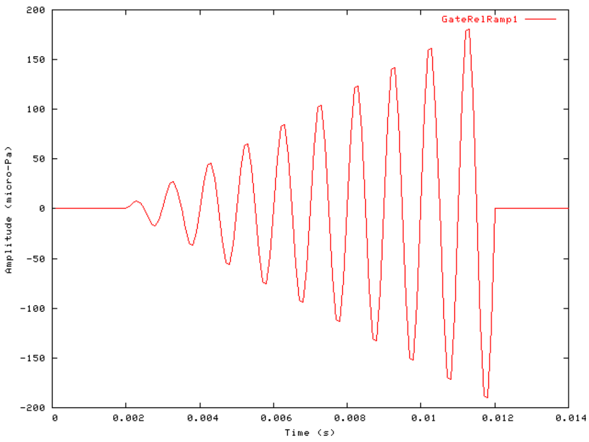
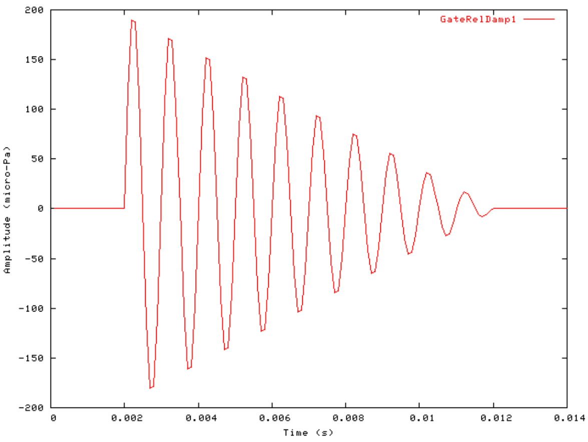


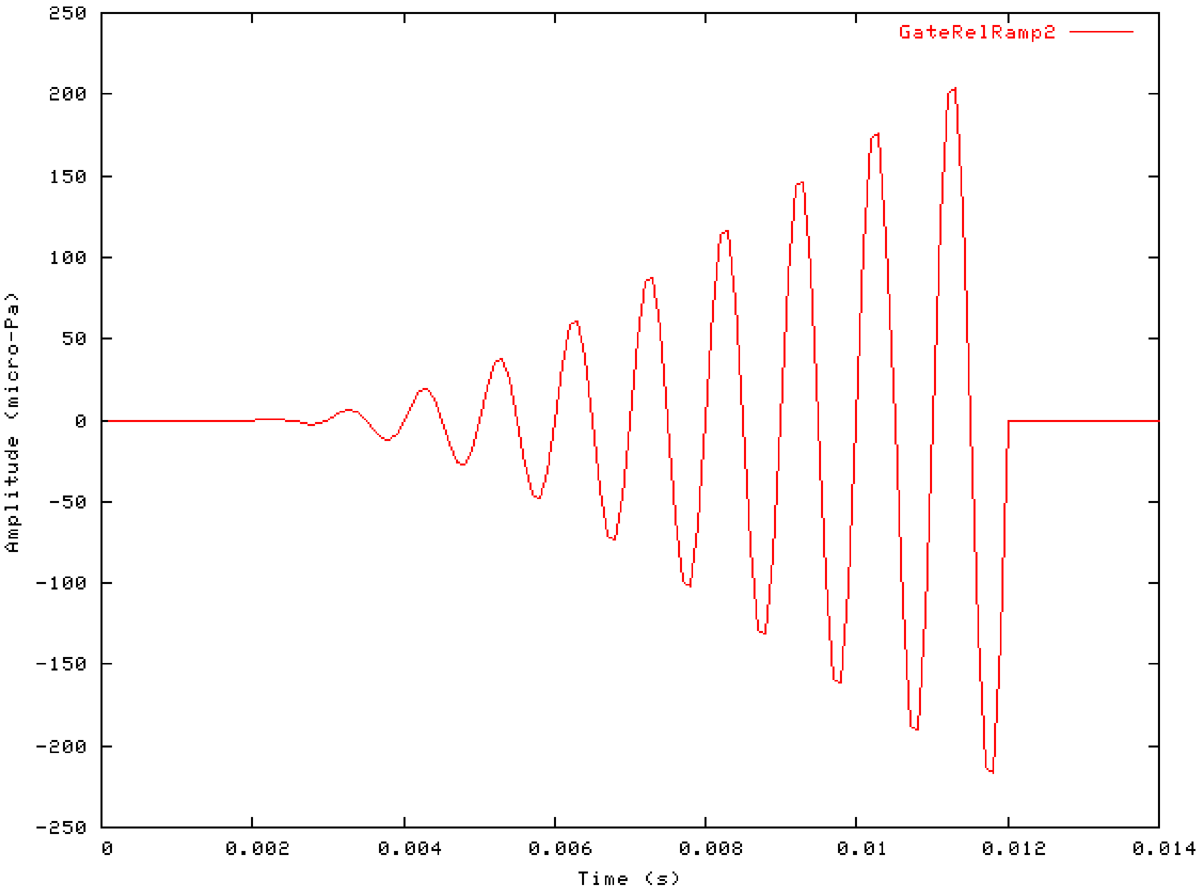
Figure 13.5. Relative Sine Damp



Raised Cosine Example: AutoTest/Tr/GateRelRamp2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Tr/GateRelRamp2.sim
begin  {

 Stim_PureTone_2  < P2Tone1.par
 Trans_Gate      < Ramp3.par
 Display_Signal
}
```

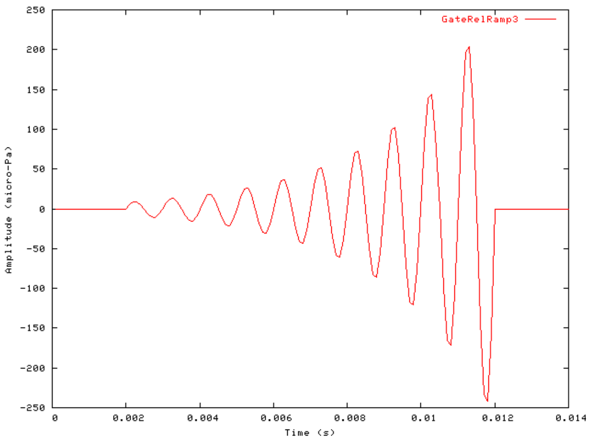
Figure 13.6. Gating module results using *raised_cos* function.



Exponential Decay Example: AutoTest/Tr/GateRelRamp3.sim & AutoTest/Tr/GateRelDamp2.sim

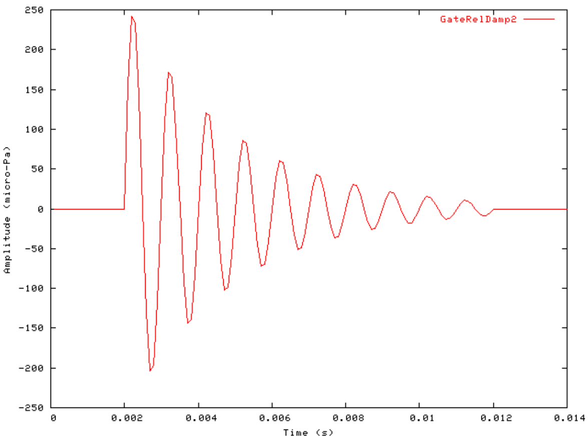
```
# This example simulation script
with its associated parameter files
is in the
# AutoTest package, which can be
downloaded from the DSAM web site.

# ./Tr/GateRelRamp3.sim
begin  {

 Stim_PureTone_2  < P2Tone1.par
 Trans_Gate       < Ramp4.par
 Display_Signal

}
```

Figure 13.7. Relative Exponential Ramp



```
# This example simulation script
with its associated parameter files
is in the
# AutoTest package, which can be
downloaded from the DSAM web site.

# ./Tr/GateRelDamp2.sim
begin  {

 Stim_PureTone_2  < P2Tone1.par
 Trans_Gate       < Damp3.par
 Display_Signal

}
```

Figure 13.8. Relative Exponential Damp



Comments:

All the tests use a pure tone stimulus surround by silence.

Trans_Ramp: Ramping Transform Module

File name: TrGate.[ch]

Description

When this process module is used the Trans_Gate process module is called in reduced operation mode.

Note that the use of the process module implies that the *op_mode* parameter is set to *ramp* and cannot be changed.

Inputs	Arbitrary signal input.
Outputs	Transforms the input signal and passes on.
Reference	

Module Parameters

Example 13.4. Trans_Ramp Parameter File

POS_MODE	RELATIVE	Position mode ('absolute' or 'relative')
OP_MODE	RAMP	Operation mode ('ramp' or 'damp')
TYPE_MODE	RAISED_COS	Type mode ('linear', 'sine', 'raised_cos' or 'exp_decay')
OFFSET	0	Time offset (s)
DURATION	0.0025	Ramp duration - negative assumes to the end of the signal (s)
SLOPE	0.016	Slope parameter or half-life (in 'decay_par' mode only)

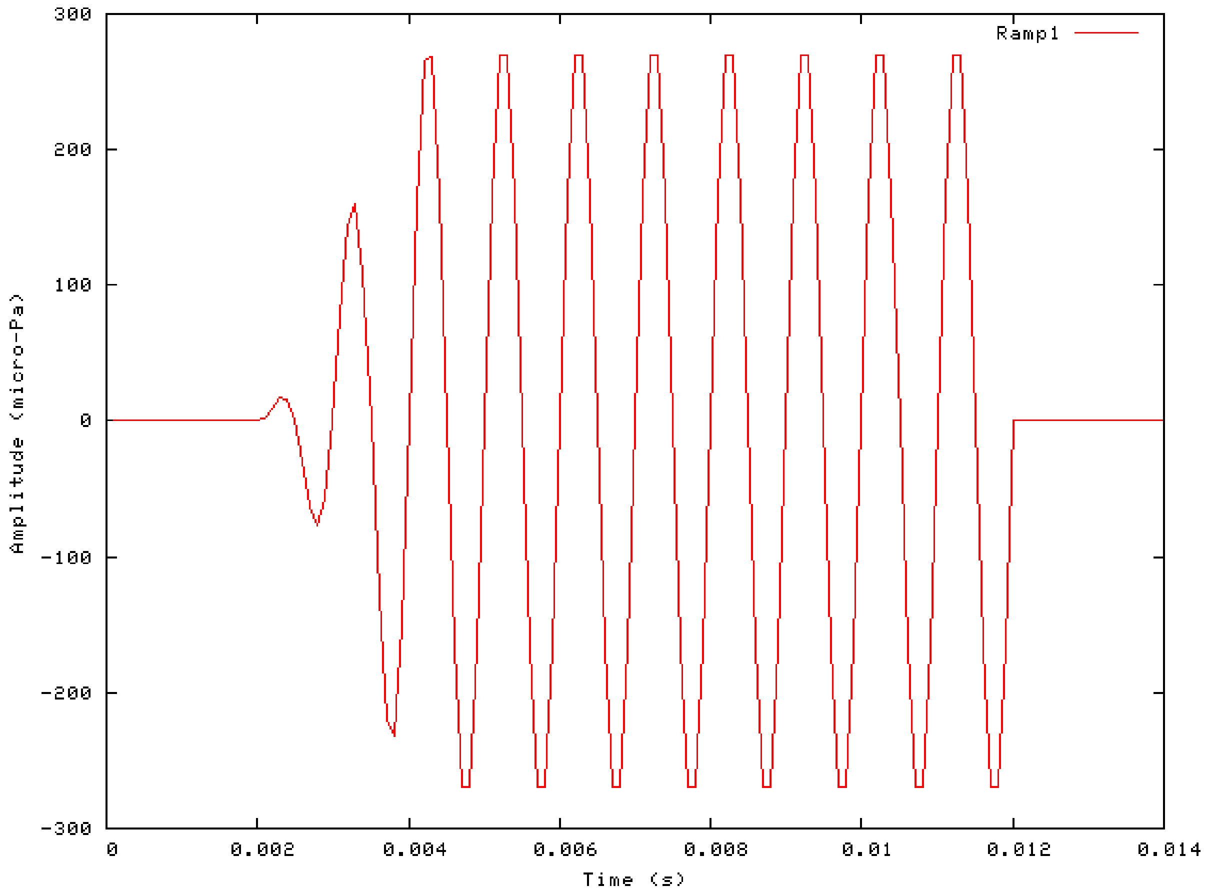
See the section called “Trans_Gate: Gating Transform Module” for the parameter options. Note that the use of the process module implies that the *op_mode* parameter is set to *ramp* and cannot be changed.

Examples Using the 'Trans_Ramp' Process Module

Ramped Example: AutoTest/Tr/Ramp1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Tr/Ramp1.sim
begin {

 Stim_PureTone_2  < P2Tone1.par
 Trans_Ramp     < Ramp5.par
 Display_Signal

}
```

Figure 13.9. Ramped Sinusoid



Trans_SetDBSPL: Level transform module

File name: TrSetDBSPL.[ch]

Description

This program calculates a signal’s current intensity and adjusts it to a specified intensity. An offset is specified for the start of the intensity calculation. **It is not advisable using this module in segment processing mode, as it will set the gain for the entire signal from the first processed segment only**. Therefore the appropriate gain will only be correct for the rest of the signal if the first segment contains a reference pure tone.

Inputs	Arbitrary signal input.
Outputs	Transforms the input signal and passes on.
Reference	

Module Parameters

Example 13.5. Trans_SetDBSPL Parameter File

```
OFFSET   0   Time offset for intensity calculation (s).
DBSPL    0   Required intensity setting (dB SPL).
```

Example: AutoTest/Tr/SetDBSPL1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Tr/SetDBSPL1.sim
begin  {

 Stim_PureTone    < PTone1.par
 repeat 10 {
 Trans_SetDBSPL   < DBSPL1.par
 Display_Signal
 }
 Ana_Intensity    < Intensty1.par

}
```

This test outputs a single number result.

Chapter 14. Utility Modules

Util_Accumulate: Accumulate Utility Module

File name: UtAccumulate.[ch]

Description

This utility accumulates data from its input process. It adds to previous data if the process has already been used. **It will continue adding to the previous data unless the process is reset -** see the section called “Accumulating Data with Process Modules”.

This utility can also accumulate output from more than one connected process. This allows the output from more than one process to be accumulated as a single process signal.

Inputs	Arbitrary. If input is being supplied from multiple processes, then they must all have the same signal dimensions.
Outputs	Produces an output signal with the same dimensions as the input signal, though the output.
Reference	

Module Parameters

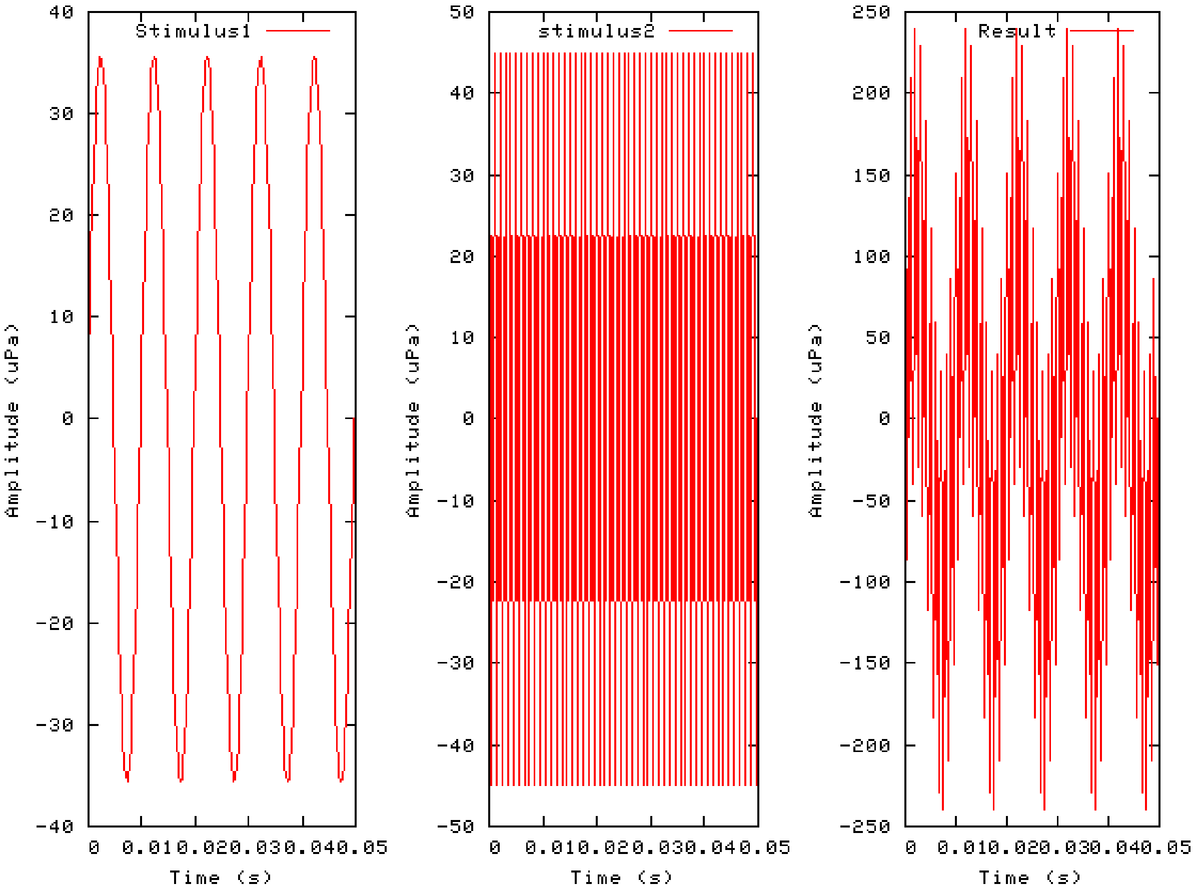
This utility has no parameters.

Examples Using the ’Ana_ACF’ Process Module

Example: AutoTest/Ut/Accumulate1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/Accumulate1.sim
begin  {

s1% Stim_PureTone  (->a) < PTone2.par
s2% Stim_PureTone  (->a) < PTone3.par
 repeat 3 {
 a% Util_Accumulate (s1, s2->)
 Display_Signal
 }

}
```

Figure 14.1. Two pure tones added together using the accumulator module.



Util_AmpMod: Amplitude Modulation Utility Module

File name: UtAmpMod.[ch]

Description

This module generates an amplitude modulated output signal from the connected input signal. A number of amplitude modulations can be specified.

Inputs	Arbitrary single input.
Outputs	Produces output signal with the same dimensions as the input.
Reference	

Module Parameters

Example 14.1. Util_AmpMod Parameter File

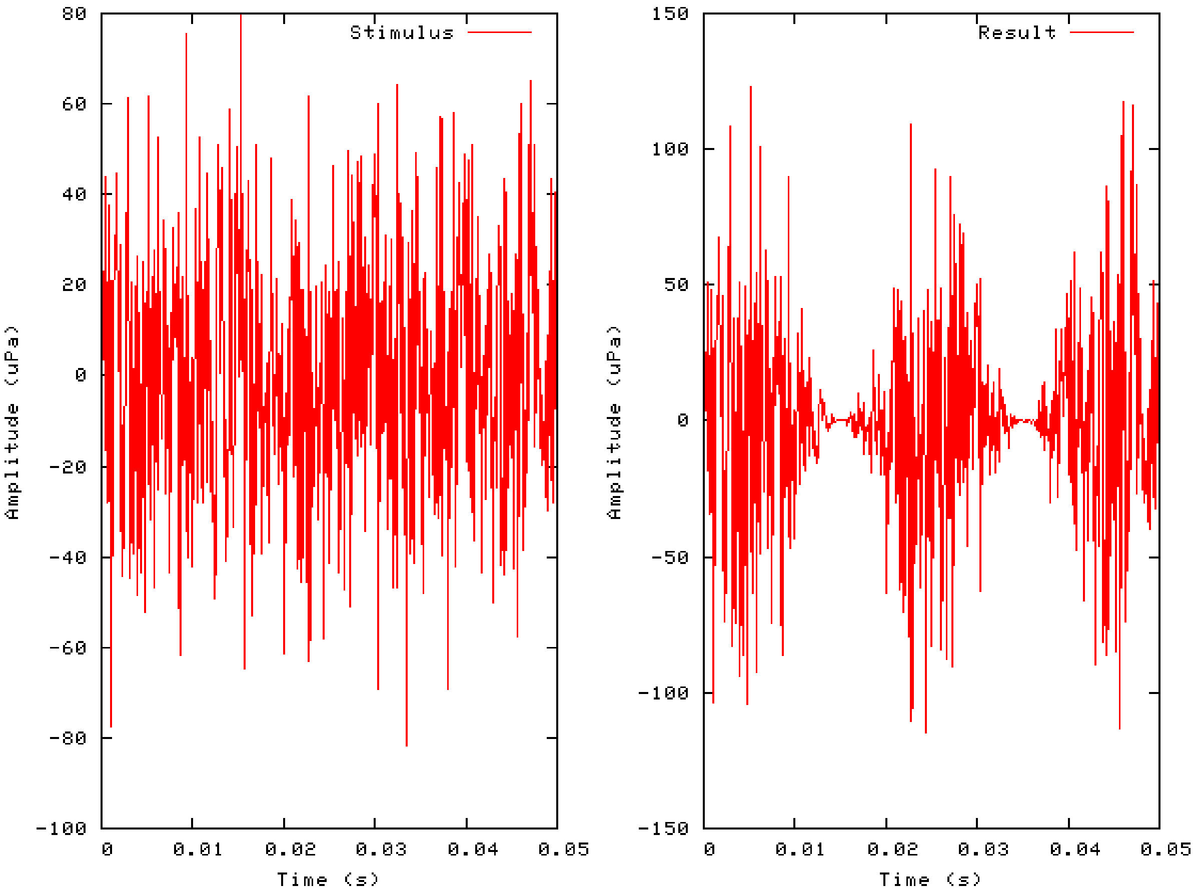
NUM_FREQS	1	Number of modulation frequencies.
DEPTH	0:50	
FREQUENCY	0:100	
PHASE	0:0	

Examples Using the ’Util_AmpMod’ Process Module

Example: AutoTest/Ut/AmpModulation1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/AmpModulation1.sim
begin  {

st% Stim_WhiteNoise  < WNoise2.par
 Util_AmpMod     < AmpMod1.par
 Display_Signal

}
```

Figure 14.2. An amplitude modulated pure tone using the Amplitude modulation utility.



Util_BinSignal: Bin Collection Utility Module

File name: UtBinSignal.[ch]

Description

This module generates a binned histogram from the input process. It adds to previous data if it has already been used. **It will continue adding to the previous data unless the process is reset -** see the section called “Accumulating Data with Process Modules”.

Inputs	Arbitrary. This module accepts multiple inputs.
Outputs	Produces an output signal with each channel length defined by the *bin_width* parameter and the length of the original signal. If the *bin_width* parameter is negative then it will be set to the input signal duration by default.
Reference	

Module Parameters

Example 14.2. Util_BinSignal Parameter File

MODE	SUM	Bining mode (’average’ or ’sum’).
BIN_WIDTH	-1	Bin width for binned signal (s) (-ve: prev. signal duration).

Table 14.1. Util_BinSignal: Mode options

Mode	Description
average	Each output bin contains the average of the input binned samples, i.e. if the bin consists of 10 input samples the total bin sum will be divided by 10.
sum	Each output bin contains the sum of the input binned samples.

Examples Using the ’Util_AmpMod’ Process Module

Example: AutoTest/Ut/BinSignal1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/BinSignal1.sim
begin {

st% Stim_PureTone   < PTone2.par
 repeat 4 {
 Util_BinSignal  < BinSig1.par
 Display_Signal
 }

}
```

Figure 14.3. Output from the BinSignal utility module using the*sum* binning mode. The bin width is set to the stimulus sampling interval.

Example: AutoTest/Ut/BinSignal2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/BinSignal2.sim
begin  {

 Stim_PureTone    < PTone2.par
 repeat 4 {
 Util_BinSignal   < BinSig2.par
 Display_Signal
 }

}
```

Figure 14.4. Output from the BinSignal utility module using the*sum* binning mode. The bin width is set to half the stimulus sampling interval.

Example: AutoTest/Ut/BinSignal3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/BinSignal3.sim
begin  {

 Stim_PureTone    < PTone2.par
 repeat 4 {
 Util_BinSignal   < BinSig3.par
 Display_Signal
 }

}
```

Figure 14.5. Output from the BinSignal utility module using the*average* binning mode. The bin width is set to half the stimulus sampling interval.

Util_Compression: Compression Utility Module

File name: UtCompress.[ch]

Description

This module generates compressed output from the input process. It offers several compression functions.

Inputs	Arbitrary single input
Outputs	Produces signal with same dimensions as input signal.
Reference	
See also	Util_HalfWaveRectify

Module Parameters

Example 14.3. Util_Compression Parameter File

```
MODE          LOG  Compression mode ('log' or 'power').
MULTIPLIER    1    Signal multiplier (arbitrary units).
EXPONENT      1    Power exponent ('power' mode only).
MIN_RESPONSE  0    Minimum response from module (arbitrary units).
```

Table 14.2. Util_Compression: Mode options

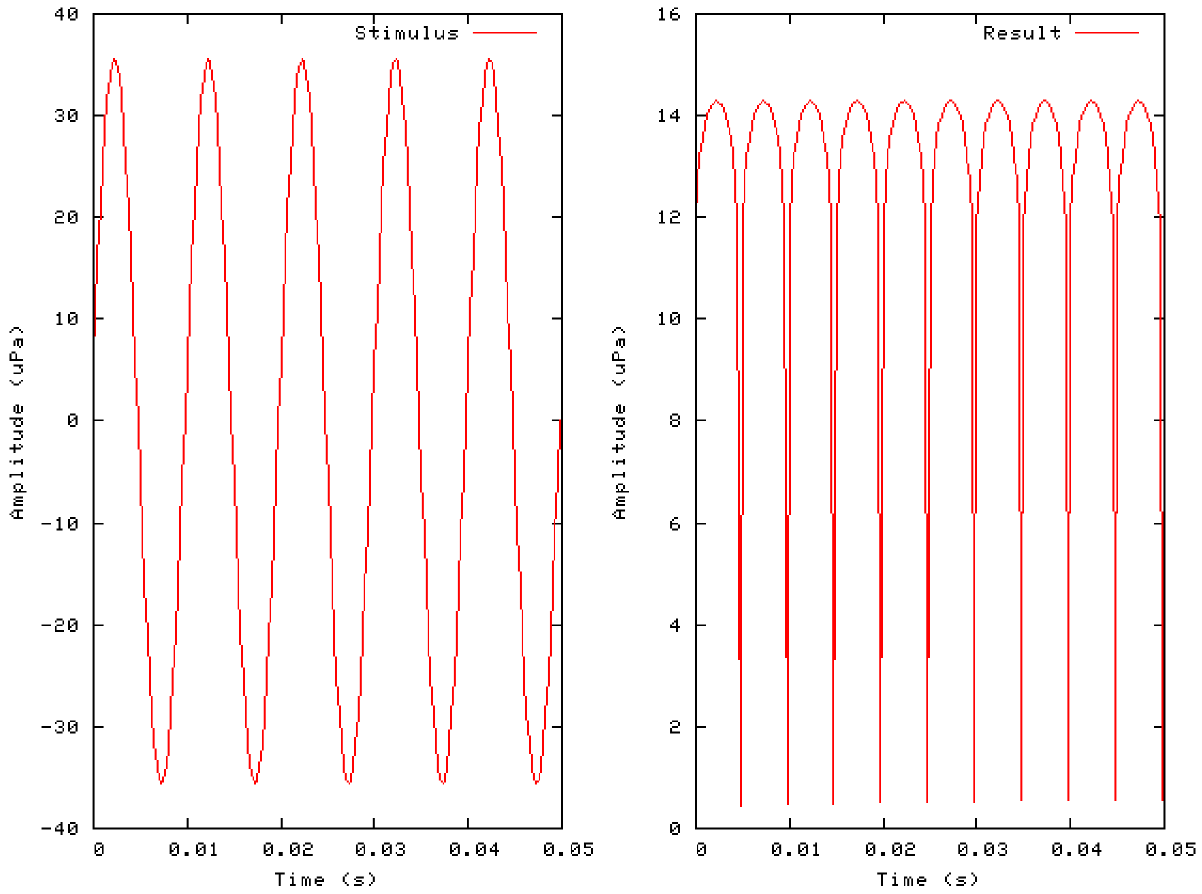
Mode	Description
log	Logarithmic compression: y = log10(|x|). In this mode the output is not allowed to fall below the *MIN_RESPONSE*parameter value.
power	Power compression: y = a.|x|^z where z is the **exponent** parameter and a is the **MULTIPLIER**.

Examples Using the ’Util_Compression’ Process Module

Example: AutoTest/Ut/Compress1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/Compress1.sim
begin  {

st% Stim_PureTone   < PTone2.par
 Util_Compression < Compres1.par
 Display_Signal

}
```

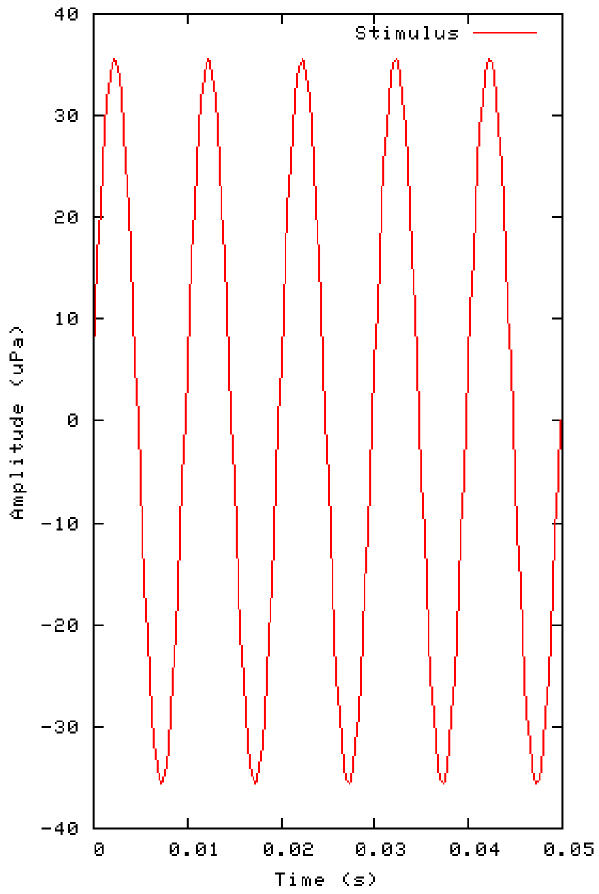
Figure 14.6. Output from the Compression utility module, showing *power* compression (see Patterson et al. [27]). The stimulus input is also shown.

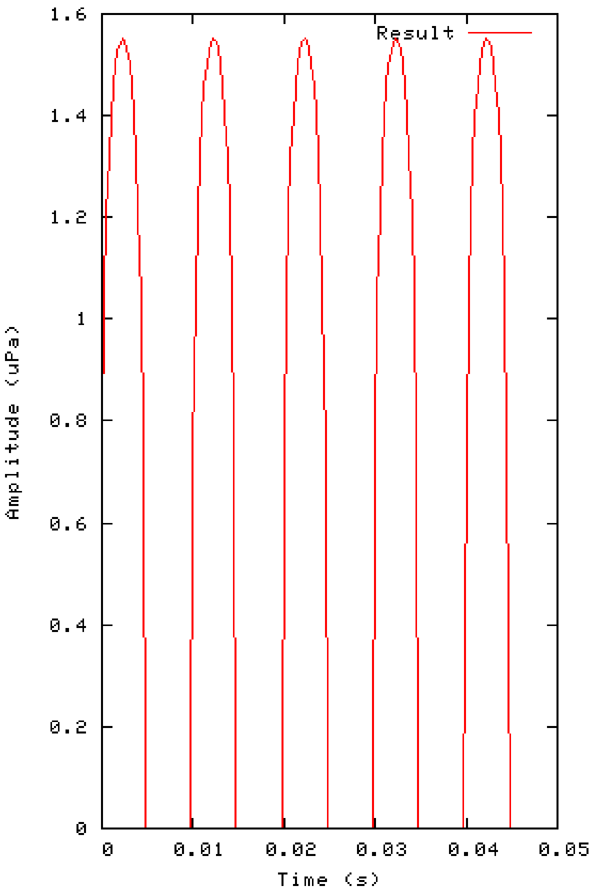


Example: AutoTest/Ut/Compress2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/Compress2.sim
begin  {

st% Stim_PureTone    < PTone2.par
 Util_Compression < Compres2.par
 Display_Signal

}
```

Figure 14.7. Output from the Compression utility module, showing *log* compression (see Patterson et al. [27]). The stimulus input is also shown.





Util_ConvMonaural: Monaural Conversion Utility Module

File name: UtConvMonaural.[ch]

Description

This routine creates a monaural signal from a binaural signal. The resulting output is the sum of the left and right channels.

Inputs	Binaural signal.
Outputs	Produces a monaural signal with the other signal dimensions remaining the same.
Reference	

Module Parameters

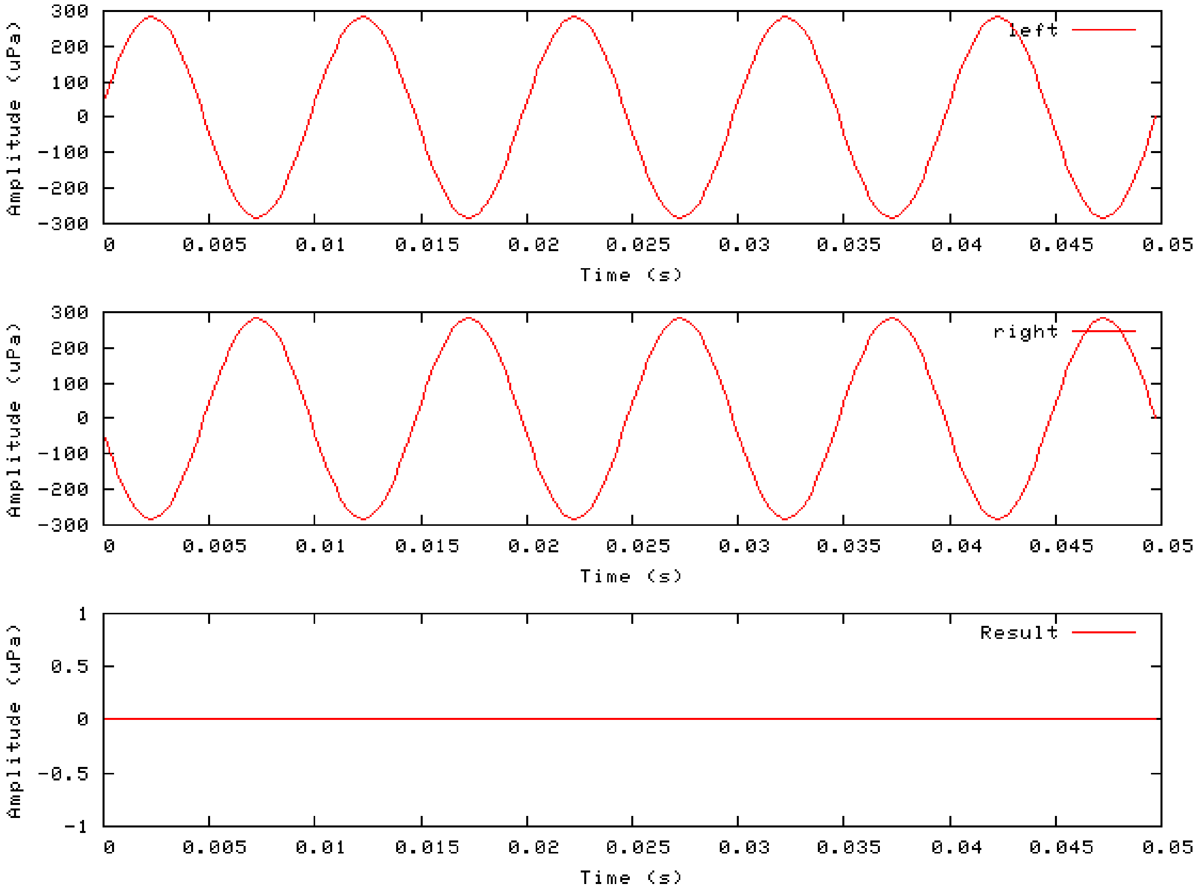
This utility has no parameters.

Examples Using the ’Util_ConvMonauralion’ Process Module

Example: AutoTest/Ut/ConvMonaural1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/ConvMonaural1.sim
begin  {

st% Stim_PureTone_Binaural < BPTone2.par
 repeat 10 {
 Util_ConvMonaural
 Display_Signal
 }

}
```

Figure 14.8. A binaural signal with the left and right waveforms (top two panels respectively) 180 degress out of phase converted to a monaural signal. The result should be a zero signal (subject to rounding errors).



Util_CreateBinaural: Binaural Output Creation Utility Module

File name: UtCreateBinaural.[ch]

Description

This routine creates an interleaved binaural signal from two connected input processes.

Inputs	Expects two process inputs.
Outputs	Produces a binaural output.
Reference	

Module Parameters

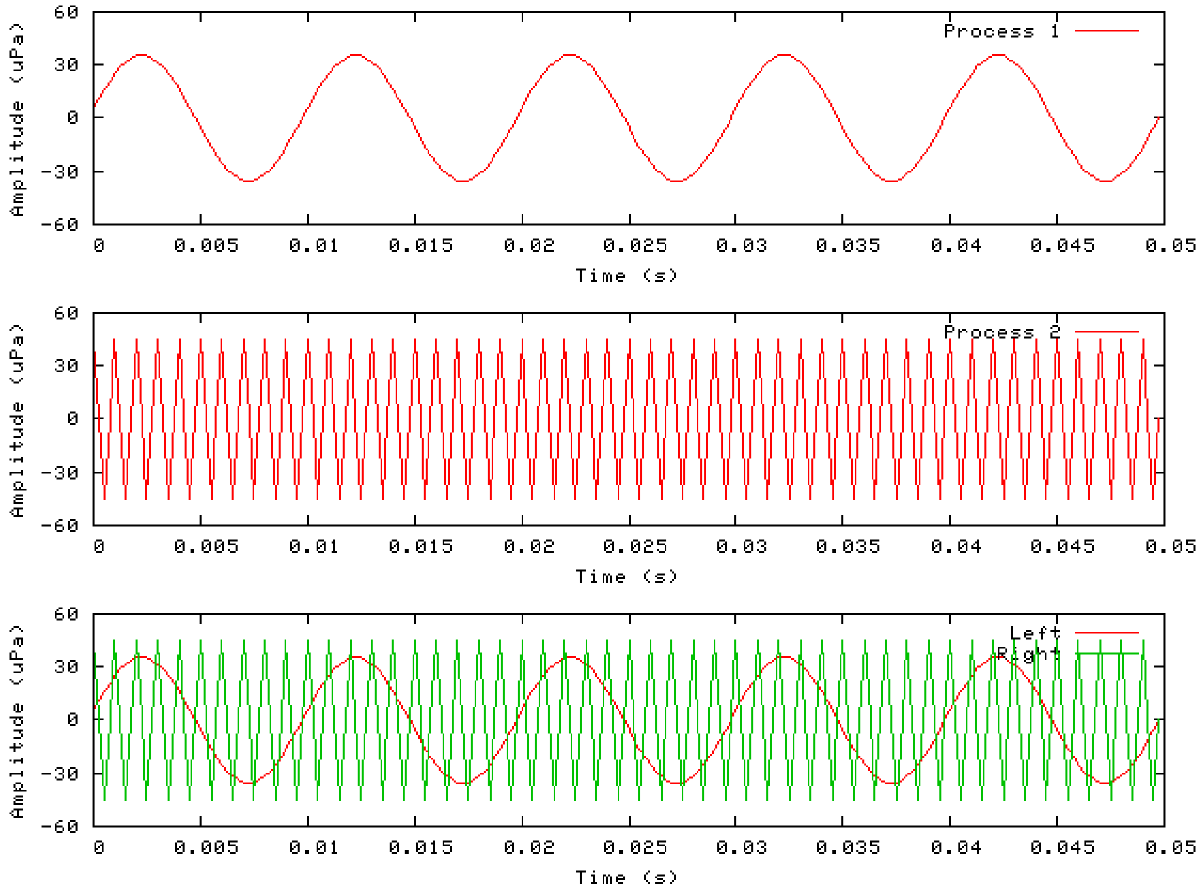
This utility has no parameters.

Examples Using the ’Util_CreateBinaural’ Process Module

Example: AutoTest/Ut/CreateBinaural1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/CreateBinaural1.sim
begin  {

s1% Stim_PureTone  (->cb) < PTone2.par
s2% Stim_PureTone  (->cb) < PTone3.par
cb% Util_CreateBinaural(s1, s2->)
 Display_Signal

}
```

Figure 14.9. Two pure tones of different frequency and amplitude are combined to create a binaural signal using theUtil_CreateBinaural.



Util_CreateJoined: Signal Join Utility Module

File name: UtCreateJoined.[ch]

Description

This routine creates an output by joining the connected process inputs.

Inputs	Expects any number of process inputs.
Outputs	Produces an output signal that has a length which is the sum of the input process signals.
Reference	

Module Parameters

This utility has no parameters.

Examples Using the 'Util_CreateJoined' Process Module

Example: AutoTest/Ut/CreateJoined1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.
# ./Ut/CreateJoined1.sim
begin  {

s1% Stim_PureTone  (->cb) < PTone2.par
s2% Stim_PureTone  (->cb) < PTone3.par
s3% Stim_PureTone  (->cb) < PTone2.par
cb% Util_CreateJoined(s1, s2, s3->)
 Display_Signal

}
```

Figure 14.10. Three pure tones with two of different frequency have been joined using the *Util_CreateJoined* utility.

Util_Delay: Signal Delay Utility Module

File name: UtDelay.[ch]

Description

This module introduces an initial delay in a monaural signal or an interaural time difference (ITD) in a binaural signal. For binaural signals if the delay is positive, then the right (second) channel is delayed relative to the left (first) channel, and vice versus for negative delays. Monaural signals always treat delays as positive values.

Inputs	Arbitrary
Outputs	Produces delayed signal, with an unchanged signal length, i.e. the end of the signal is ’pushed off the end’.
Reference	-

Module Parameters

Example 14.4. Util_Delay Parameter File

MODE	SINGLE	Operation mode (’single’ or ’linear’).
INITIAL_DELAY	0	Initial time delay (s).
FINAL_DELAY	0	Final time delay (not used with ’single’ mode).

Table 14.3. Util_Delay: Mode options

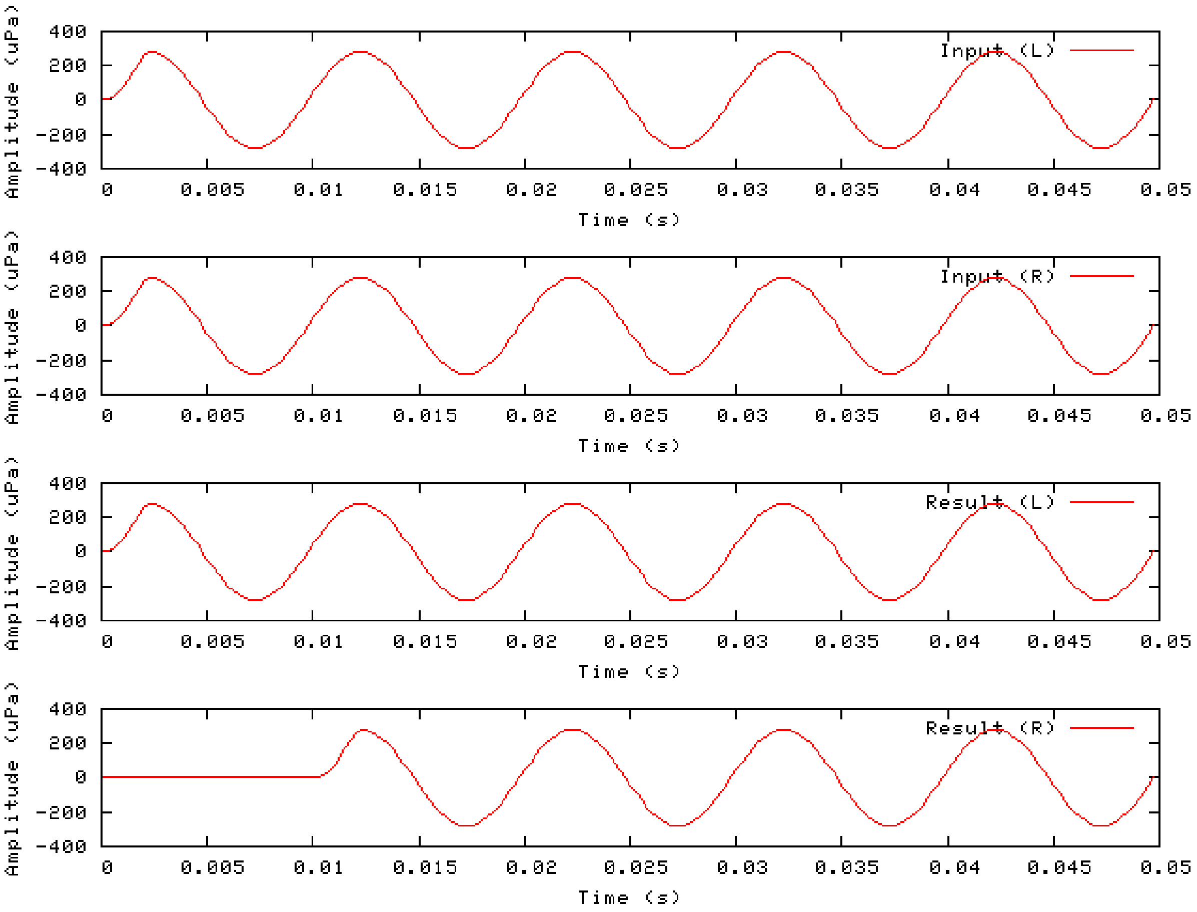
Mode	Description
Single	In this mode only the **INITIAL_DELAY** parameter is used, and this same delay is applied to all channels.
Linear	In this mode the delay is different for each channel. Starting from the lowest channel (channel index 0) the delay varies linearly from the **INITIAL_DELAY** value to the **FINAL_DELAY** value for the highest channel.

Examples Using the ’Util_Delay’ Process Module

Example: AutoTest/Ut/Delay1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/Delay1.sim
begin {

st% Stim_PureTone_Binaural < BPTone1.par
 Trans_Gate      < Ramp4.par
 Util_Delay      < Delay1.par
 Display_Signal

}
```

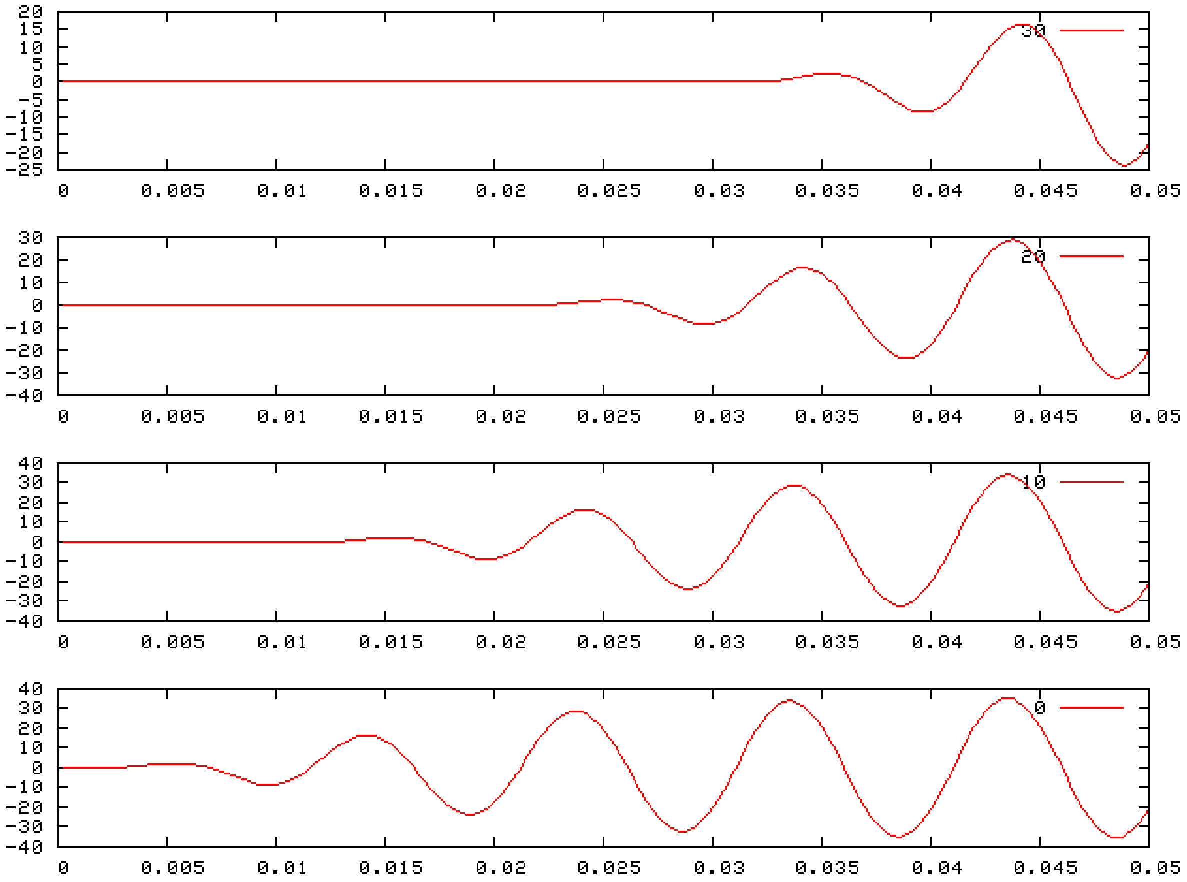
Figure 14.11. The "Result" panels show a binaural signal that has had an ITD introduced using.



Example: AutoTest/Ut/Delay2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/Delay2.sim
begin  {

 Stim_PureTone    < PTone7.par
 Trans_Gate      < Ramp4.par
 BM_GammaT       < GammaTL4.par
 repeat 4 {
 Util_Delay      < Delay2.par
 Display_Signal
 }

}
```

Figure 14.12. The response of a gamma tone filter after it has been processed using *Util_Delay* in the *linear* mode. The legend shows the delay value for each channel.



Util_HalfWaveRectify: Half-wave Rectification Utility Module

File name: UtHalfWRectify.[ch]

Description

This module uses the input signal from its process EarObject to generated an output signal that is half-wave rectified.

Inputs	Arbitrary single input.
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	

Module Parameters

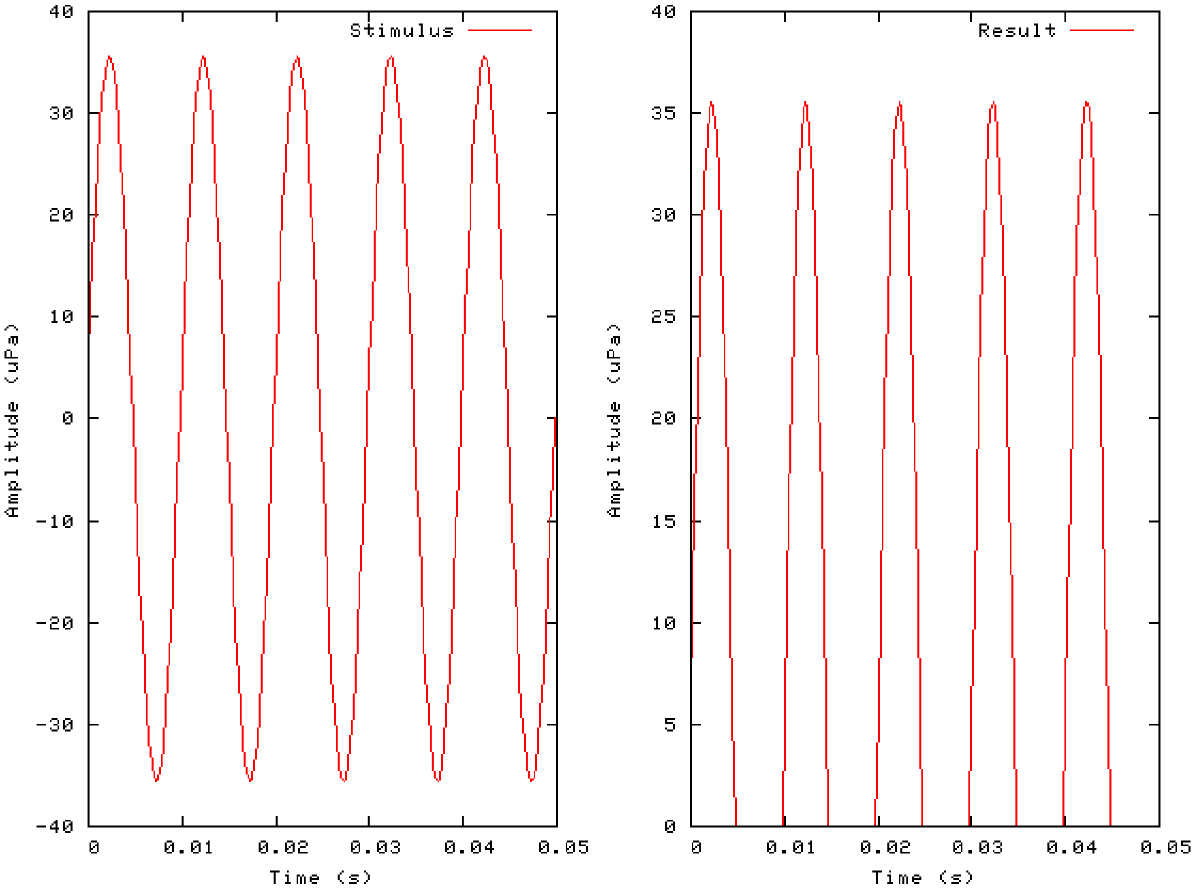
This utility has no parameters.

Examples Using the ’Util_HalfWaveRectify’ Process Module

Example: AutoTest/Ut/HalfWRectify1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/HalfWRectify1.sim
begin  {

st% Stim_PureTone    < PTone2.par
 Util_HalfWaveRectify
 Display_Signal

}
```

Figure 14.13. A pure tone processed using the *Util_HalfWaveRectify* utility.



Util_IteratedRipple: Iterated Ripple Utility Module

File name: UtIterRipple.[ch]

Description

This module can be used to produce iterated ripple noise (IRN) stimuli. The module processes its input to generate an output signal using the Iterated Ripple Algorithm as described in [33].

To produce IRN stimuli the input should be the Stim_WhiteNoise process module. The **NUM_IT** parameter sets the number of times the noise segment is added. The **GAIN** and **DELAY** parameters respectively set the attenuation of the added noise and the delay before it is added. The **MODE** parameter sets whether the original sample of white noise is added at every iteration (IRSO), or whether the noise that results from each iteration is used as the next sample to be added (IRSS).

This process utility can also be used to apply the Iterated Ripple Algorithm to other signals.

Inputs	Arbitrary single input.
Outputs	Produces an output signal with the same dimensions as the input.
Reference	[33].
See also	Stim_WhiteNoise

Module Parameters

Example 14.5. Util_IteratedRipple Parameter File

NUM_IT	2	Number of iterations.
MODE	IRSO	Ripple signal mode: ’IRSO’ - add original, ’IRSS’ - add same.
DELAY	0	Delay (s).
GAIN	1	Gain (scalar).

Table 14.4. Util_IteratedRipple: Mode options

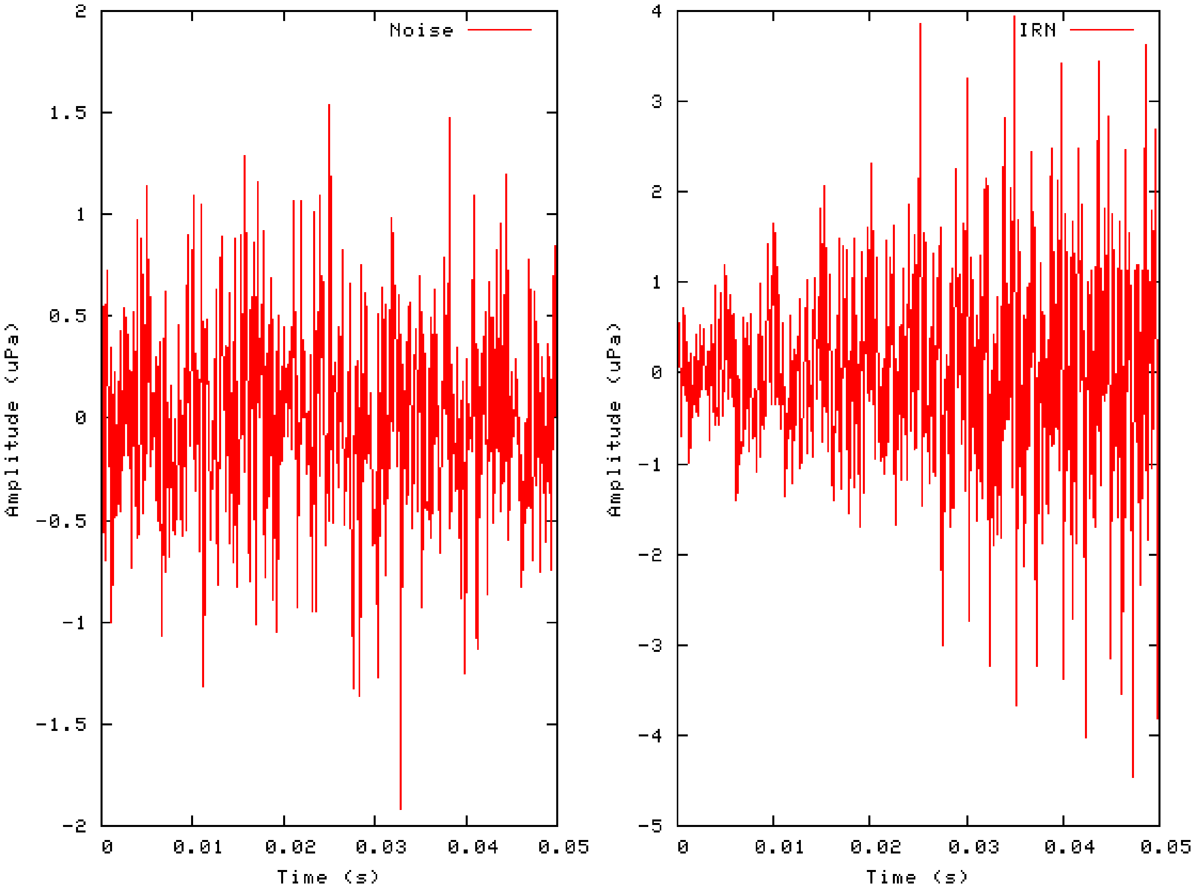
Mode	**Description**
IRSO	Iterated Rippled Signal Add-Original
IRSS	Iterated Rippled signal Add-Same.

Examples Using the ’Util_IteratedRipple’ Process Module

Example: AutoTest/Ut/AmpModulation1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/IteratedRipple1.sim
begin {

st% Stim_WhiteNoise  < WNoise1.par
 Util_IteratedRipple < IterRip1.par
 Display_Signal

}
```

Figure 14.14. The autocorrelation of the above IRN stimulus, showing a pitch percept 200 Hz (5 ms lag). With IRN the pitch percept frequency occurs at the reciprical of the delay.



Util_MathOp: Mathematical Operation Utility Module

File name: UtMathOp.[ch]

Description

This utility carries out simple mathematical operations upon it inputs. The operations are done on a sample-by-sample basis.

Inputs	This process requires input signal or multiple processes, depending upon the chosen mathematical operation. For multiple input operations the inputs are expected to similar.
Outputs	Produces an output signal that has been process according to the chosen operator. The output signal has the same dimensions (length number of channels, etc.) as the input signal.
Reference	-

Module Parameters

Example 14.6. Util_MathOp Parameter File

OPERATOR	SCALE	Mathematical operator ('add', 'modulus', 'scale', 'sqr' or 'subtract').
OPERAND	1	Operand (only used in scale mode at present).

Table 14.5. Util_MathOp: Operator options.

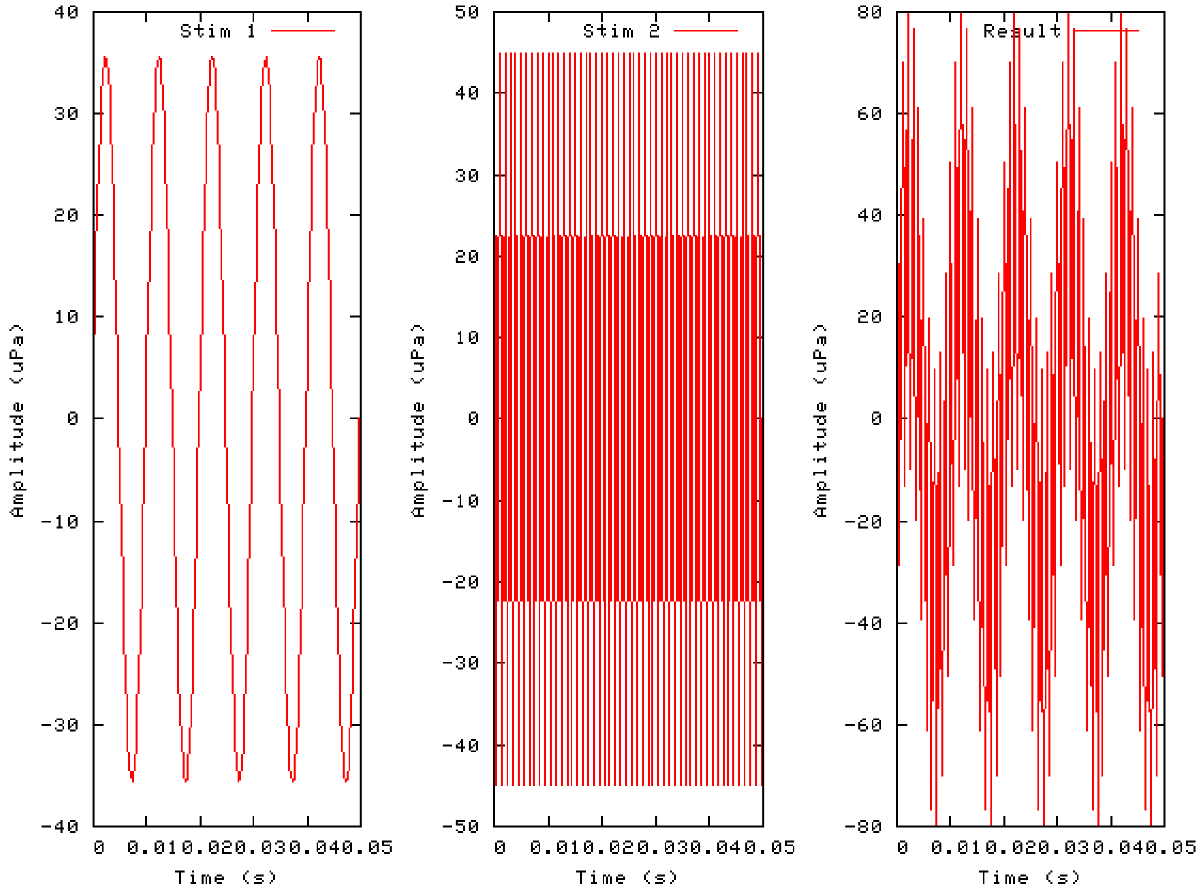
Operation	Description
Add	This operator adds each sample of two processes: {process 1} + {process 2}.
Modulus	This operator finds the modulus of each sample of a process: |{process}|.
Scale	This operator multiplies each sample of a process by a scaler: {process} * operand.
Sqr	This operator squares each sample of a process: {process} * {process}.
substract	This operator subtracts the samples of two processes: {process 1} - {process 2}.

Examples Using the 'Util_MathOp' Process Module

Example: AutoTest/Ut/MathOp1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/MathOp1.sim
begin  {

s1% Stim_PureTone  (->mo) < PTone2.par
s2% Stim_PureTone  (->mo) < PTone3.par
mo% Util_MathOp    (s1, s2->) < MathOp1.par
 Display_Signal

}
```

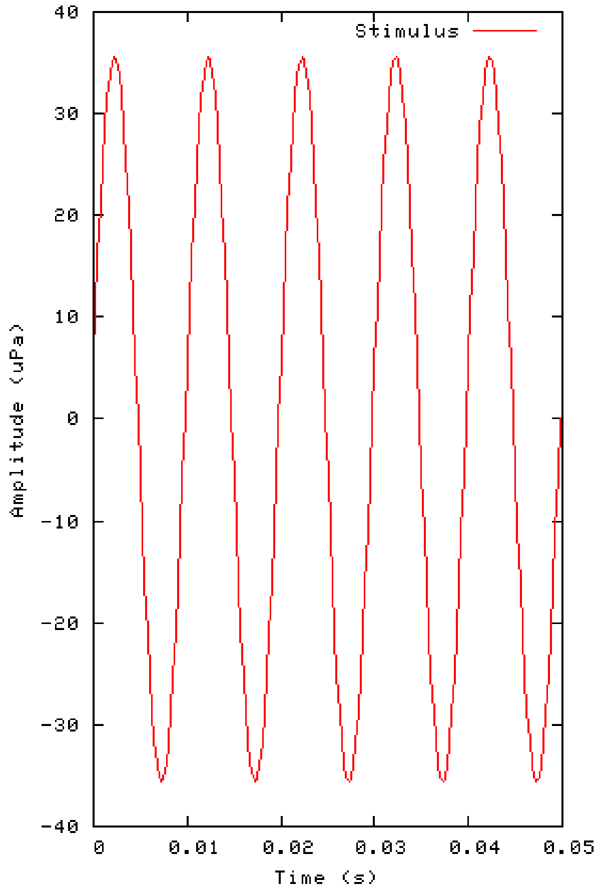
Figure 14.15. Two pure tones with different frequencies have been added using the Util_MathOp process.

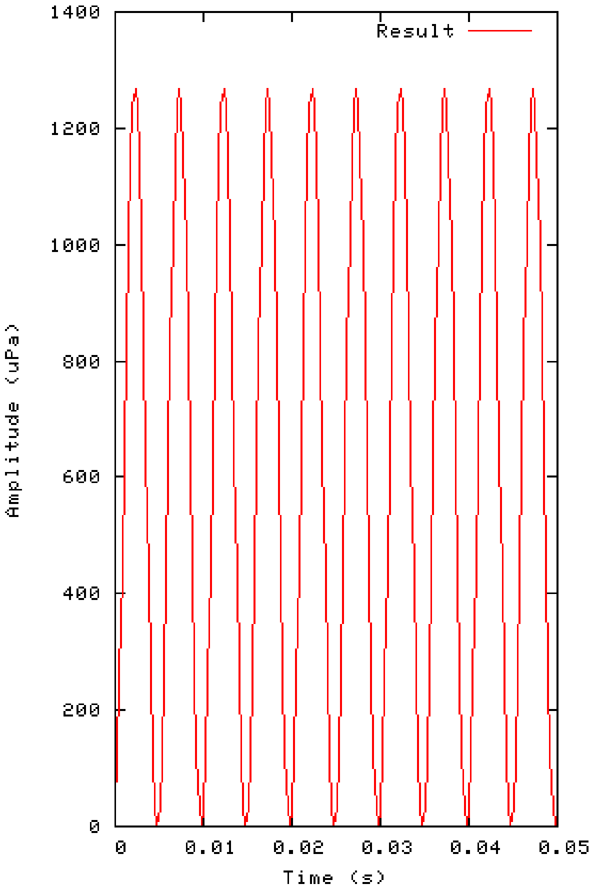


Example: AutoTest/Ut/MathOp4.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/MathOp4.sim
begin  {

st% Stim_PureTone    < PTone2.par
 Util_MathOp     < MathOp4.par
 Display_Signal

}
```

Figure 14.16. A pure tone has been squared using the Util_MathOp process.





Util_PadSignal: Signal Padding Utility Module

File name: UtPadSignal.[ch]

Description

This module produces an output signal that has been padded at the beginning and end of the signal with specified pad values, e.g. silence. Different pad value can be used for the beginning and end of the signal.

Inputs	Arbitrary
Outputs	Produces an output signal, with the addition length as specified by the padding.
Reference	-

Module Parameters

Example 14.7. Util_PadSignal Parameter File

BEGIN_DURATION	0	The pad duration for the beginning of the signal (s).
BEGIN_VALUE	0	The pad value for the beginning of the signal (units).
END_DURATION	0	The pad duration for the end of the signal (s).
END_VALUE	0	The pad value for the end of the signal (units).

Examples Using the ’Util_PadSignal’ Process Module

Example: AutoTest/Ut/PadSignal1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/PadSignal1.sim
begin  {

st% Stim_PureTone    < PTone6.par
 Display_Signal
 Util_PadSignal   < PadSignal1.par
 Display_Signal

}
```

Figure 14.17. A pure tone padded using the *Util_PadSignal* process.

Util_Pause: Process Pause Utility Module

File name: UtPause.[ch]

Description

This module allows the processing of a simulation to be paused for a period of time **DELAY** rounded to the nearest second, producing a specified message. The alert mode behaves differently for GUI and non-GUI applications: applications compiled with or without graphics support. For GUI applications the alert mode will cause the program to halt and produce an alert window when the **DELAY** is set to zero. Processing will then wait until the *Ok* button is pressed. For non-GUI applications, the alert mode will cause an audible bell.

Inputs	Arbitrary
Outputs	The input signal is passed as the output signal unchanged.
Reference	-

Module Parameters

Example 14.8. Util_Pause Parameter File

ALERT_MODE	ON	Bell mode ('on' or 'off').
DELAY	-1	Delay time: negative values mean indefinite period (s).
MESSAGE	"Processing paused"	Pause diagnostic message.

Examples Using the 'Util_Pause' Process Module

Example: AutoTest/Ut/Pause1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/Pause1.sim
begin  {

 Stim_PureTone    < PTone6.par
 Util_Pause      < Pause1.par
 Display_Signal

}
```

Util_ReduceChannels: Channel Reduction Utility Module

File name: UtRedceChans.[ch]

Description

This module reduces multiple channel data to a lower number of channels. The input process channels are divided into the process' output channels. Each sample of the appropriate number of channels of the input signal is summed, and then the average is calculated by dividing by the number of channels summed.

The input channels are divided into the output channels. Therefore the number of output channels must be a multiple of the number of input channels.

Inputs	Arbitrary single input.
Outputs	Produces an output signal with a reduced number of channels, but other dimensions remain the same.
Reference	

Module Parameters

Example 14.9. Util_ReduceChannels Parameter File

MODE	SUM	Mode - ’average’ or simple ’sum’.
NUM_CHANNELS	1	Number of channels signal is to be averaged to.

Table 14.6. Util_ReduceChannels: Mode options.

Mode	Description
average	When combining channels the resulting combination channel is averaged across channel.
Sum	When combining channels the resulting combination channel is simply summed across channel.

Examples Using the ’Util_ReduceChannels’ Process Module

Example: AutoTest/Ut/ReduceChannels1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/ReduceChannels1.sim
begin {

 Stim_PureTone    < PTone2.par
 Trans_Gate      < Ramp2.par
 BM_GammaT      < GammaTL1.par
 Util_ReduceChannels < RedceCh1.par
 Display_Signal

}
```

Figure 14.18. ACF input to the Util_ReduceChannels process module.

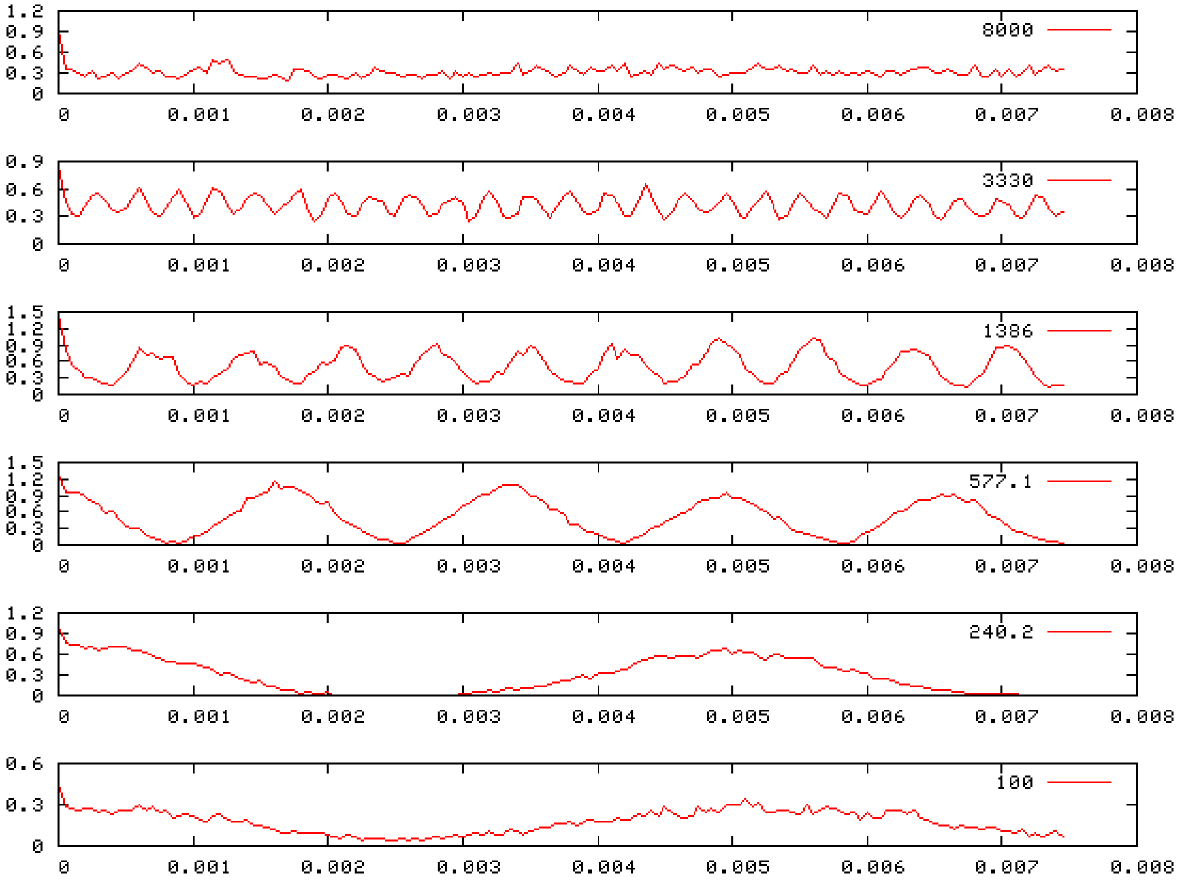


Figure 14.19. Reduce channels output for above ACF input.

Util_ReduceDT: Sampling Interval/Rate Conversion Utility Module

File name: UtReduceDt.[ch]

Description

This module produces an output signal whose sampling rate is increased by reducing the sampling interval of the input process signal by a specified denominator, i.e. dt->dt/denominator.

Inputs	Arbitrary signal input.
Outputs	Produces a signal with a smaller sampling interval, hence longer sample length.
Reference	

Module Parameters

Example 14.10. Util_ReduceDT Parameter File

```
DENOMINATOR   1   Reduction denominator (integer).
```

Examples Using the ’Util_ReduceDT’ Process Module

Example: AutoTest/Ut/ReduceDT1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/ReduceDT1.sim
begin  {

 Stim_PureTone    < PTone2.par
 Util_ReduceDT    < RedceDt1.par
 Display_Signal

}
```

Pure tone stimulus

The pure tone stimulus is not visibly changed by processing with Util_ReduceDT, however, more sample points have been added.

Util_RefractoryAdjust: Refractory Adjustment Utility Module

File name: UtRefractory.[ch]

Description

This module applies a refractory effect adjustment for auditory nerve output. It is normally used to operate on spike probabilities - see [22]. A fixed dead time following a spike is set with length **PERIOD**, during which no spike events can occur.

Inputs	Arbitrary signal input. This process has the most meaning, however, when used with proabability output, such as from the IHC_Meddis86 process module.
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	
See also	AN_SG_Carney, AN_SG_Simple.

Module Parameters

Example 14.11. Util_RefractoryAdjust Parameter File

```
PERIOD    0   Refractory period (s).
```

Examples Using the ’Util_RefractoryAdjust’ Process Module

Example: AutoTest/Ut/RefractoryAdj1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/RefractoryAdj1.sim
begin  {

 Stim_PureTone    < PTone4.par
 Trans_Gate      < Ramp2.par
 IHC_Meddis86    < Meddis86.par
 repeat 10 {
 Util_RefractoryAdjust < Refract1.par
 }
 Display_Signal

}
```

Figure 14.20. Output from a Meddis ’86 IHC, adjusted for refractory effects using the *Util_RefractoryAdjust* utility module.

Util_Sample: Signal Sampling Utility Module

File name: UtSample.[ch]

Description

This module samples a signal at fixed intervals specified by **DT**. It could be used as a crude method to reduce the sampling rate of the input signal.

Inputs	Arbitrary signal input.
Outputs	Produces an output signal where each channel is sampled at the given sampling interval rate, **DT.**
Reference	

Module Parameters

Example 14.12. Util_Sample Parameter File

```
TIMEOFFSET   0    Time offset (s).
DT          -1    Sampling interval, dt (s) (-ve assumes prev. signal dt).
```

Examples Using the ’Util_Sample’ Process Module

Example: AutoTest/Ut/Sample1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/Sample1.sim
begin  {

 Stim_PureTone    < PTone2.par
 Util_Sample     < Sample1.par
 Display_Signal

}
```

Figure 14.21. This is a pure tone after processing by the ’Util_Sample’ utility module. Note the uneven curve of the sinusoid due to missing intermediary points.

Util_SelectChannels: Channel Selection Utility Module

File name: UtSelectChans.[ch]

Description

This module copies the input signal to the output signal, but copies only the specified input channels or sets the other channels to zero, depending upon the mode: *remove*, *zero* or *expand*. The *selection mode* setting defines which channels are chosen.

Inputs	All inputs allowed
Outputs	The number of output channels depends on the mode used.
Reference	Used in the DvowelIdent application.

Module Parameters

Example 14.13. Util_SelectChannels Parameter File

```
MODE              REMOVE  Selection mode - ’zero’, ’remove’ or ’expand’.
SELECTION_MODE    ALL     Channel selection mode (’all’, ’middle’, ’top’, ’botton’ or ’user’.
NUM_CHANNELS      0       No. of channels in selection field (This is no longer).
```

Table 14.7. Util_SelectChans: Mode options

Mode	**Description**
Remove	Using the selection array, unselected (zero) channels are not included in the produced output signal, and the selected channel’s samples are multiplied by the value in the selection array. The number of output channels produced by this process module will be equal to or less than the number of input channels of the supplying process module.
Zero	The number of output channels for this process is the same as the number of input channels as supplied by the previous process. However, using the selection array, the samples of the output channels that have not been selected will be set to zero. The selected channel’s samples are multiplied by the value in the selection array.
Expand	Using the selection array unselected (zero) channels are not in included in the produced output signal. For each selected channel the corresponding number copies in the output signal is defined by the value of the selection variable. For example, a value of 6 in the selection array for channel 0 will produce six copies of channel 0. Using this mode the number of output channels produced can be greater or less then the number of input channels of the supplying process module.

Table 14.8. Util_SelectChans: Selection Mode options

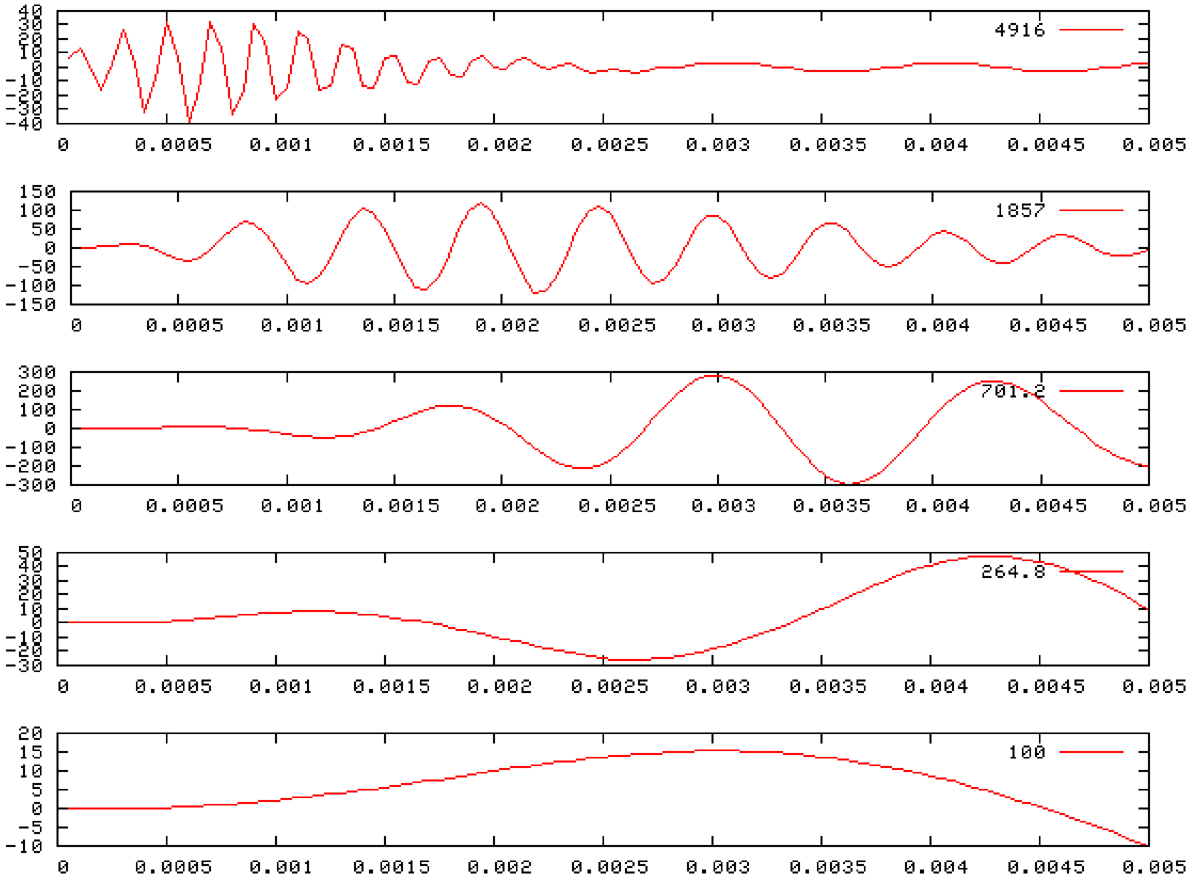
Selection Mode	**Description**
All	This is the default value. All channels are selected, and this means the signal is passed on unchanged.
Highest	Only the highest indexed channel is selected.
Lowest	Only the zero channel is selected.
Middle	Only the middle channel is selected.
User	In this mode an array of real values is supplied as a parameter to the module. Each value in the array corresponds to an input signal channel. Zero values mark a channel as "off". Positive values mark a channel as "on", and in *expand* operation mode the magnitude of the value defines the number of copies of the channel made.

Examples Using the 'Util_SelectChannels' Process Module

Example: AutoTest/Ut/SelectChannels1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/SelectChannels1.sim
begin  {

 Stim_PureTone    < PTone5.par
 Trans_Gate      < Ramp3.par
 BM_GammaT      < GammaTL3.par
 Util_SelectChannels < SelecCh1.par
 Display_Signal

}
```

Figure 14.22. This shows the output from a 10 channel gammatone filter bank with alternate channels removed using *Util_SelectChannels*.



Util_ShapePulse: Pulse Shaping Utility Module

File name: UtShapePulse.[ch]

Description

This module turns a train of detected spikes into a pulse train with set **MAGNITUDE** and **DURATION**. For continuous signals, a **THRESHOLD** can be set which will convert positive-going crossings of the threshold into spike events.

Inputs	Arbitrary signal input.
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	

Module Parameters

Example 14.14. Util_ShapePulse Parameter File

```
THRESHOLD  0       Event threshold (arbitrary units).
DURATION   0.0002  Pulse duration (s).
MAGNITUDE  3.8     Pulse magnitude (arbitrary units).
```

Examples Using the ’Util_ShapePulse’ Process Module

Example: AutoTest/Ut/ShapePulse1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/ShapePulse1.sim
begin  {

 Stim_PulseTrain  < PulseTr4.par
 repeat 2 {
 Util_ShapePulse  < ShapePulse1.par
 }
 Display_Signal

}
```

Figure 14.23. This routine shows a pulse train stimulus processed using *Util_ShapePulse*. The process is repeated twice: once with the module accumulating the results, then once with the module process being reset just before each time it is called. The two runs are output to the files ’output1.dat’ and ’output2.dat’ respectively.

Util_SimScript: Simulation Script Interface

File name: UtSimScript.[ch]

Description

This module is used for running simulation processes using the *Simulation script* programming mode see the section called “Using simulation scripts ”.

When the diagnostics mode is *on*, each time the *simulation* is executed, the process diagnostics will be printed out. However, it is important to note that the diagnostics mode for simulation scripts is overridden by the diagnostics parameter for the main application. This means that when the diagnostics mode for the main application is *off* than even if the simulation script’s diagnostic mode is *on* there will no diagnostics output from the simulation run. The parameter file path mode defines the file path for the simulation’s parameter files.

Inputs	Arbitrary. This process accepts multiple inputs.
Outputs	The output produced is that of the last process in the simulation
Reference	
See also	Using simulation scripts

Examples Using the ’Util_ReduceChannels’ Process Module

Example: AutoTest/Ut/SimScript1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/SimScript1.sim
begin  {

 Stim_Harmonic   < Hrmonic1.par
gstim% Trans_Gate    (->sim1, sim2) < Ramp2.par
sim1% Util_SimScript (gstim->cbin) < SimScript2.sim
sim2% Util_SimScript (gstim->cbin) < SimScript2.sim
cbin% Util_CreateBinaural(sim1, sim2->)
 reset fibres
 repeat 500 {
 AN_SG_Simple    < SpikeGn1.par
 fibres% Util_Accumulate
 }
 Ana_ACF       < AutoCorr2.par
 Display_Signal  < Display1.par

}
```

Figure 14.24. Output using the *Util_SimScript* utility module.

Util_Standardise: Signal Standardisation Utility Module

File name: UtStddise.[ch]

Description

This module standardises an input (x_I) with N samples to produce a output with:

$$\sum_{i=1}^{N} x_i = 0, \qquad \sum_{i=1}^{N} \frac{x_i^2}{N} = 1$$

The method used uses the standard deviation and mean of the signal i.e.

$$\text{s.d.} = \sum_{i=1}^{N} \frac{\left(x_i - \bar{x}\right)^2}{N}$$

The standardised values are calculated for each sample x_i ,

$$S_i = \frac{x_i - \bar{x}}{\text{s.d.}}$$

Inputs	Arbitrary signal input.
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	

Module Parameters

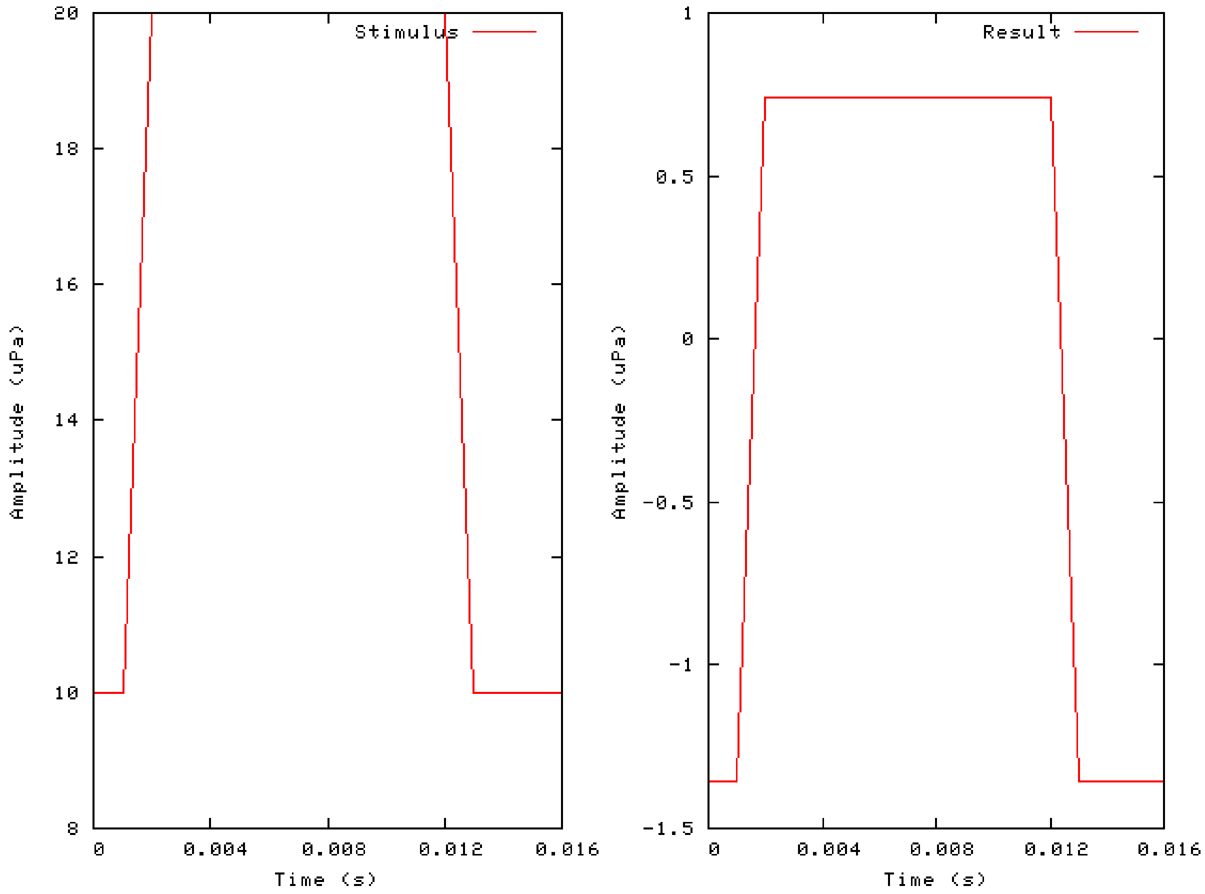
This utility has no parameters.

Examples Using the ’Util_Standardise’ Process Module

Example: AutoTest/Ut/Standardise1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/Standardise1.sim
begin  {

st% Stim_StepFun    < StepFun1.par
 Util_Standardise
 Display_Signal
}
```

Figure 14.25. A step function stimulus, transformed using *Util_Standardise*.



Util_Strobe: Signal Strobe Utility Module

File name: UtStrobe.[ch]

Description

This module implements strobe criteria for the AIM stabilised auditory image (SAI). The Ana_SAI calls this routine (See [27]). The process monitors activity in each channel of the process' input signal producing strobe pulses according to the required strobe mechanism. The different strobe mechanism modes, set using the **CRITERION** parameter, are described below in Table 14.9, “Util_Strobe: Strobe modes.”.

The **STROBE_DIAGNOSTICS** parameter can be ’off’, ’screen’ or sent to a ’file name’. At present this module’s diagnostics consists of the strobe threshold value at each time interval.

Inputs	Arbitrary signal input.
Outputs	Produces an output signal with the same dimensions as the input signal.
Reference	[27]
See also	Ana_SAI

Module Parameters

Example 14.15. Ana_SAI Parameter File

```
DIAGNOSTICS          OFF             Diagnostic mode (’off’, ’screen’, ’error’ or <file
                                     name>).
INT_MODE             STI             Integration mode: ’STI’ - stabilised temporal
                                     integration, ’AC’ - autocorrelation.
CRITERION            PEAK_SHADOW+    Strobe criterion (mode): ’user ’, ’threshold’ (0),
                                     ’peak’ (1), ’peak_shadow-’ (3), or ’peak_shadow+’
                                     (4/5).
STROBE_DIAGNOSTICS   OFF             Diagnostic mode (’off’, ’screen’, ’error’ or <file
                                     name>).
THRESHOLD            0               Threshold for strobing: ’user’, ’threshold’ and ’peak’
                                     modes only.
THRESHOLD_DECAY      5000            Threshold decay rate (%/s).
STROBE_LAG           0.005           Strobe lag (delay) time (s).
TIMEOUT              0.01            Strobe lag (delay) timeout (s).
NWIDTH               -0.035          Negative width of auditory image (s).
PWIDTH               0.005           Positive width of auditory image (s).
NAP_DECAY            2500            Neural activity pattern (input) decay rate (%/s)
IMAGE_DECAY          0.03            Auditory image decay half-life (s).
```

Table 14.9. Util_Strobe: Strobe modes.

Mode	Description	Original AIM Criteria
User	This mode expects input from another process. For this mode only the **threshold** parameter is used, to define the input value above which strobing occurs.	(-)
Threshold	Strobe on every sample above the specified **threshold**.	1
Peak	Strobe on each peak of the input signal which is above the specified **threshold**	2
Peak_Shadow-	Avoid strobing on input signal peaks preceded by a larger peak. An adaptive strobe threshold is set up for each channel of the input signal and its value is set to the height of the first input signal peak when it occurs. Thereafter, the strobe threshold decays exponentially with time in the absence of supra threshold input peaks. The rate of decay is defined by the **thresholdDecayRate**, which is specified as the half-life period of the process. When an input signal peak next exceeds the strobe threshold, the level of the strobe threshold is reset to the height of the new peak, and the strobe pulse is set at the peak time.	3
Peak_Shadow+	Avoid strobing on peaks followed by larger peaks. This strobe mode uses an adaptive threshold similar to the *Peak_Shadow* mode above. However, instead of assigning the strobe pulse straight away it is ’remembered’ as a potential strobe pulse. If another, larger peak occurs within the specified **delay** period, then that new peak is set as the potential strobe point. If no peaks occur within the **delay** period, then the strobe pulse is recorded. As a refinement of this process the **delayTimeout** parameter can be set after which the potential strobe pulse is recorded anyway. If the **delayTimeout** parameter is set to zero, this strobe mode operates as the original AIM strobe criteria 4, and with non-zero values it operates as strobe criteria 5.	4/5
Delata_Gamma	Not yet implemented	6

Table 14.10. Util_Strobe: Diagnostics mode options

Mode	Description
off	No diagnostics are output.;
screen	Diagnostics from the module are output to the screen.
error	Diagnostics are output to the standard error output stream (stderr on Unix)
<file name>	Diagnostics are output to the specified file.

Examples Using the ’Util_Strobe’ Process Module

Example: AutoTest/Ut/Strobe1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/Strobe1.sim
begin  {

 Util_SimScript   < StrobeTest1.sim
 repeat 1 {
 Util_Strobe     < Strobe1.par
 }
 Display_Signal

}
```

Figure 14.26. The stimulus and the strobe process response in *user* mode.

Example: AutoTest/Ut/Strobe2.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/Strobe2.sim
begin  {

 Util_SimScript   < StrobeTest1.sim
 repeat 1 {
 Util_Strobe     < Strobe2.par
 }
 Display_Signal

}
```

Figure 14.27. The stimulus and the strobe process response in *threshold* mode (Strobe criteria 1 under the old AIM system).

Example: AutoTest/Ut/Strobe3.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/Strobe3.sim
begin  {

 Util_SimScript   < StrobeTest1.sim
 repeat 1 {
 Util_Strobe     < Strobe3.par
 }
 Display_Signal

}
```

Figure 14.28. The stimulus and the strobe process response in *peak* mode (Strobe criteria 2 under the old AIM system).

Example: AutoTest/Ut/Strobe4.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/StrobeT4.sim
begin  {

 Util_SimScript   < StrobeTest2.sim
 repeat 1 {
 Util_Strobe     < Strobe4.par
 }
 Display_Signal

}
```

Figure 14.29. The stimulus and the strobe process response in *peak_shadow*- mode (Strobe criteria 3 under the old AIM system).

Example: AutoTest/Ut/Strobe5.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/Strobe5.sim
begin  {

 Util_SimScript   < StrobeTest1.sim
 repeat 1 {
 Util_Strobe     < Strobe5.par
 }
 Display_Signal

}
```

Figure 14.30. The stimulus and the strobe process response in *peak_shadow+* mode (Strobe criteria 4 under the old AIM system).

Example: AutoTest/Ut/Strobe6.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/Strobe6.sim
begin  {

 Util_SimScript   < StrobeTest1.sim
 repeat 1 {
 Util_Strobe     < Strobe6.par
 }
 Display_Signal

}
```

Figure 14.31. The stimulus and the strobe process response in *peak_shadow+* mode, with the time-out set to a non-zero value (Strobe criteria 5 under the old AIM system).

Comments

In this test program, the routine is run six times, using a different *strobe mode* for each run. The data is stored in the files 'output1.dat', 'output2.dat',..., 'output6.dat'. The program also outputs the test stimulus to the file 'stimulus.dat'

Util_SwapLR: Channel Swap Utility Module

File name: UtSwapLR.[ch]

Description

This utility expects binaural input. It produces output with the left and right channels swapped.

Inputs	This process requires input from a binaural signal.
Outputs	Produces swapped binaural output signal.
Reference	-

Module Parameters

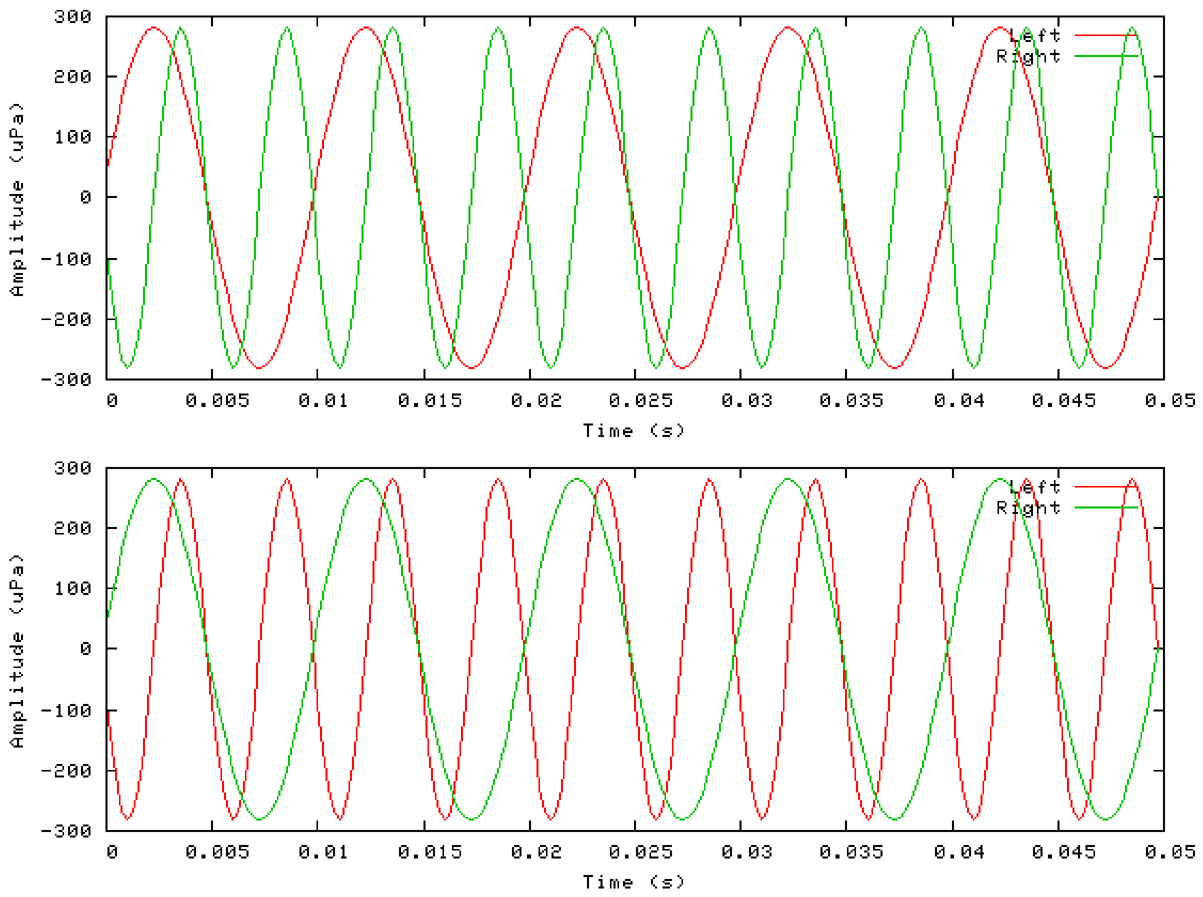
This utility has no parameters.

Examples Using the 'Util_SwapLR' Process Module

Example: AutoTest/Ut/SwapLR1.sim

```
# This example simulation script with its associated parameter files is in the
# AutoTest package, which can be downloaded from the DSAM web site.

# ./Ut/SwapLR1.sim
begin  {

st% Stim_PureTone_Binaural < BPTone3.par
 Display_Signal
 Util_SwapLR
 Display_Signal

}
```

Figure 14.32. A binaural signal before (top panel) and after (bottom panel) applying the *Util_SwapLR* process.



Bibliography

[1] Augustine G.J., Charlton M.P., Smith S. J. (**1985a**) “Calcium entry into voltage-clamped pre-synaptic terminals of squid” *J. Physiol. (London)* [367], 143-162.

[2] Arle J. E., Kim D. O., (**1991**) “Neural Modeling of Intrinsic and Spike-Discharge Properties of Cochlear Nucleus Neurons” *Biological Cybernetics* [64], 273-283.

[3] Beauchamp K., Yuen C., (**1979**) *Digital Methods for Signal Analysis*, George Allen & Unwin, London. 256-257.

[4] Beauvois Michael W. , Meddis R., (**1991**) “A Computer Model of Auditory Stream Segregation” *The Quarterly Journal of Experimental Psychology* [43A], 517-541.

[5] Carney L. H., (**1993**) “A model for the responses of low-frequency auditory-nerve fibers in cat” *J. Acoust. Soc. Am.* [93], 401-417.

[6] Dallos P., Cheatham M. A., (**??**) “Non-Linearities in Cochlea Receptor Potentials and their Origins$^{a)}$” *J. Acoust. Soc. Am.* [86], 1790-1796.

[7] Davis H., (**1958**) “Transmission and Transduction in the Cochlea” *Laryngoscope* [68], 359-382.

[8] Fay Richard R., (**1988**) *Hearing in Vertebrates: a Psychophysics Databook* Fay Richard R., Hill-Fay Associates, Winnetka, Illinois, 443.

[9] Greenwood D. D., (**1990**) “A cochlear frequency-position function for several species -- 29 years later” *J. Acoust. Soc. Am.* [87], 2592-2605.

[10] Hewitt M. J., Meddis R., (**1991a**) “An Evaluation of Eight Computer Models of Mammalian Inner Hair-Cell Function” *J. Acoust. Soc. Am.* [90], 904-917.

[11] Hewitt M. J., Meddis R., (**1993**) “Regularity of cochlear nucleus stellate cells: A computational Modeling study” *J. of the Acoust. Soc. Am.* [93], 3390-3399.

[12] Geisler C. D., (**1987**) “Coding of Acoustic Signals on the Auditory Nerve” *IEEE Engineering in Medicine and Biology Magazine* [June], 22-28.

[13] Glasberg B. R., Moore B. C. J., (**1990**) “Derivation of Auditory Filter Shapes from Notched-Noise Data” *Hearing Research* [47], 103-147.

[14] Hodgkin A. L., Huxley A. F., (**1952**) “A quantitative description of membrane current and its application to conduction and excitation in nerve” *J. Physiol.* [117], 500-544.

[15] Hudspeth A. J., Lewis R. S., (**1988**) “Kinetic analysis of voltage- and ion-dependant conductances in saccular hair cells of the bull-frog, rana catesbeneiana,” *J. Physiol.* [400], 237-274.

[16] Kiang N. Y.-S., Watanabe T., Thomas E. C., Clark L. F., (**1965**) “Discharge Patterns of Single Fibers in the Cat” *Research monograph* [Np. 35], MIT Press, Cambridge, MA, page 26.

[17] Kidd R. C., Weiss T. F., (**1990**) “Mechanism that degrade timing information in the cochlea” *Hearing Research* [49], 181-208.

[18] Lopez-Poveda E. A., O’Mard L. P., Meddis R., (**1998**) “A revised computational inner hair cell model” 102-108. *Psychophysical and Physiological Advances in Hearing* Palmer A. R., Rees A., Summerfield A. Q., Meddis R., Whurr Publishers Ltd., London.

[19] McGregor R. J., Yuen C., (**1987**) *Neural and Brain Modeling*, Academic, San Diego.

[20] Meddis R., (**1988**) “Simulation of mechanical to neural transduction in the auditory receptor” *J. Acoust. Soc. Am.* [79], 702-710.

[21] Meddis R., (**1988**) “Simulation of Auditory-Neural Transduction: Further Studies” *J. Acoust. Soc. Am.* [83], 1056-1063.

[22] Meddis R., Hewitt M. J., (**1991b**) “Virtual Pitch and Phase Sensitivity of a Computer Model of the Auditory Periphery: I. Pitch Identification” *J. Acoust. Soc. Am.* [89], 2866-2882.

[23] Meddis R., O’Mard L. P., Lopez-Poveda E. A., (**2001**) “A computational algorithm for computing nonlinear auditory frequency selectivity” *J. Acoust. Soc. Am.* [109], 2852-2861.

[24] Mountain D. C., Hubbard A. E., (**1996**) “Computational analysis of hair cell and auditory nerve processes” *Auditory Computation* Hawkins H. L., McMullen T. A., Fay R. R., Springer, New York.

[25] Palmer A. R., Russel I. J., (**1986**) “Phase-Locking in the Cochlear Nerve of the Guinea-Pig, and ist Relation to the Receptor Potential of Inner Hair-Cells” *Hearing Research* [24], 1-15.

[26] Patterson R. D., (**1994**) “The sound of s sinusoid: Spectral Models” *J. Acoust. Soc. Am.* [96], 1409-1418.

[27] Patterson R. D., Allerhand M. H., (**1995**) “Time-domain modeling of peripheral auditory processing: A modular architecture and a software platform” *J. Acoust. Soc. Am.* [98], 1890-1894.

[28] Pickles J. O., (**1988**) *An Introduction to the Physiology of Hearing*, Academic, London.

[29] Rhode W. S., Cooper N. P., (**1993**) “Two-Tone Suppression and Distortion on the Basilar Membrane in the Hook Region of Cat and Guinea Pig Cochleae” *Hearing Research* [66], 31-45.

[30] Rothman J. S., Young E. D., Manis P. B., (**1993**) “Convergence of Auditory Nerve Fibers Onto Bushy Cells in the Ventral Cochlear Nucleus: Implications of a Computational Model” *J. of Neurophysiology* [70], 2562-2583.

[31] Shamma S. A., Chadwick R. S., Wilbur W. J., Morrish K. A., Rinzel J., (**1986**) “A biophysical model oc cochlear processing: Intensity dependence of pure tone responses” *J. Acoust. Soc. Am.* [80], 133-145.

[32] Sumner C., Lopez-Poveda E. A., O’Mard L. P., Meddis R., (**2002**) “A revised model of the inner-hair cell and auditory-nerve complex” *J. Acoust. Soc. Am.* [111], 2178-2188.

[33] Yost W. A., Patterson R. D., Sheft S., (**1996**) “A time domain description for the pitch strength of iterated rippled noise” *J. Acoust. Soc. Am.* [99], 1066-1078.

[34] Zhang X., Heinz M. G., Bruce I. C., Carney L. H., (**2001**) “A phenomenological model for the responses of auditory-nerve fibers: I. Nonlinear tuning with compression and suppression” *J. Acoust. Soc. Am.* [109], 648-670.

Index