AMS Tutorials (2.3)

(Auditory Modelling System)

Ray Meddis
Lowel P. O'Mard

Centre for the Neural Basis of Hearing
Department of Psychology
University of Essex
Colchester
Essex, UK
CO4 3SQ

1 First steps with Auditory Modelling system (AMS)

This tutorial describes AMS and how to use it. It is divided into two sections:

- a formal description of AMS (sections to)
- worked examples using auditory models(sections to)

The tutorial assumes that

- AMS has already been installed.
- You currently have a folder at the top level C:\DSAM containing all relevant folders and resources.

If you have located the DSAM folder elsewhere, you will need to take care when interpreting path names in what follows.

The tutorial materials are all found in C:\DSAM\AMS\tutorials. Copy this folder to your normal workspace. Do not work inside the DSAM workspace.

You may find it helpful to create a shortcut to AMS.exe and place it on your desktop. You will find it in *C:\DSAM\AMS*. The phrase 'launch AMS' can then be interpreted as 'double click on the AMS shortcut'.

1.1 Getting started

Launch AMS by double-clicking on the desktop icon 'Shortcut to AMS.exe'.

The program should now display a window with a blue title bar containing pull-down menus and a 'GO' button. We call this window the 'AMS window'. You can resize the AMS window by dragging on the corners in the usual way. The current simulation is named in the title bar of the AMS window.

Do not click on 'GO'. If you do, AMS will run the program that was most recently loaded (or a default program set by the installer). This could give you a long wait or other complications.

1.1.1 Load the model

- In the AMS window, select the 'File' pull-down menu and select 'Load script file (*.sim)'
- Navigate to the *Tutorial* folder. (You should have your own copy in your workspace).
- In the *Autocorrelation* folder, select the *acf.sim* file and click on 'Open'.

AMS will now load the *acf.sim* file. Nothing will happen until you click on GO.

This operation should be trouble free if you have been given the correct files. However, if there are problems (such as missing files or incorrectly prepared .par files), these will be reported immediately in the AMS window.

1.1.2 Go!

- Click on 'GO'

Your screen should fill with display windows showing the results of the various stages of the model computations. A detailed description of these will be given in section 5.

Your screen should look something like this.

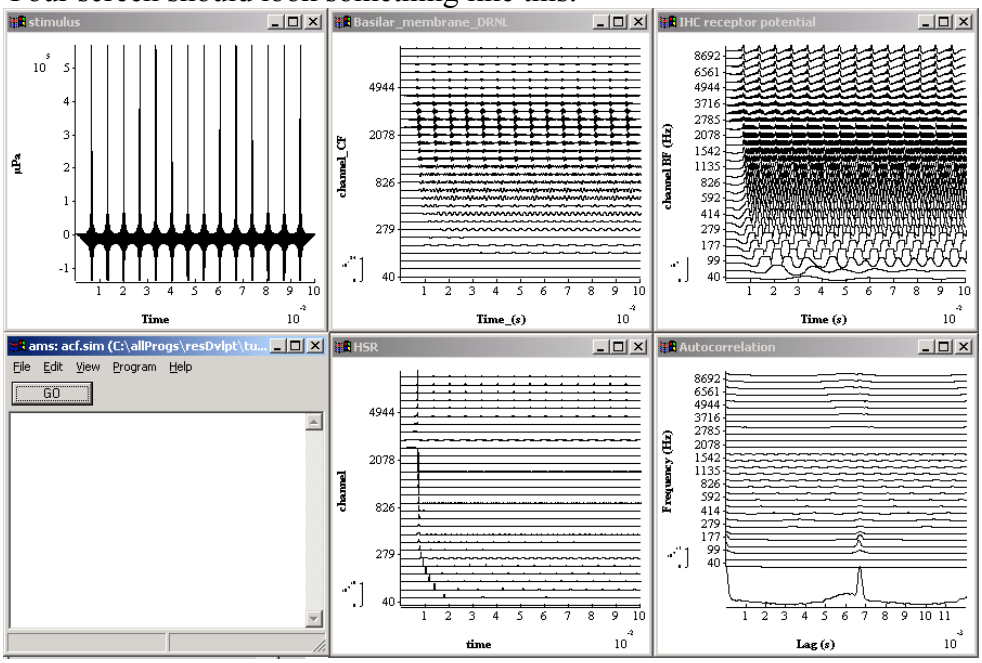


Figure 1. Sample screen display for acf.sim

1.1.3 Stop!

If a program is taking too long, select the AMS window and type CTRL/c. It might work! PCs are getting better at interrupting programs but are still not perfect.

If this does not work, kill the program by invoking the windows 'Task Manager' and 'ending' the AMS process. You can find the Task Manager by right-clicking on the taskbar (normally the bar at the bottom of the screen).

If the program locks up (nobody is perfect!), you should use the Task Manager to kill it.

1.1.4 Quit.

If the AMS window is open but the program is not running, you can quit the program using the File menu; select File->Quit from the AMS window.

1.1.5 Re-launching AMS

When you next launch AMS, it should start with the model most-recently loaded. This assumes that you quit your last session in an orderly way (by selecting 'Quit' from the File menu.

1.2 Trouble shooting.

Nothing should go wrong at this stage. If there are problems at this stage, repeat the whole installation process from the beginning.

If that does not work, email lowel@essex.ac.uk or email rmeddis@essex.ac.uk.

1.2.1 Simple things that can go wrong.

- AMS may be looking at a different simulation file from the one expected. To be absolutely sure, load the file explicitly using File->Load Script file (*.sim). This is particularly a problem if you have been using .spf files. These can look very similar to the corresponding .sim file. If you change a .sim file and it has no effect, this probably means that you are running the .spf file!

- .par file has some comments that do not have comment marks (#). If these are missing, AMS will try to read the comments and become confused.

- Case sensitivity sometimes arises, for example with stimulus file names. Mostly, however, AMS is not case sensitive.

- A file required by AMS might be in use by another application. Close the application or otherwise release the file.

2 About AMS

2.1 A shared resource

A note on the philosophy of AMS may well be appropriate here. The DSAM (Development System for Auditory Modelling) and the AMS application are all part of an auditory community-wide project. While it is hosted by the CNBH (Centre for Neural Basis of Hearing), DSAM has been designed as a platform for common use. Anyone can contribute modules to the core routine library as long as they are willing to adopt the conventions of DSAM code. A software tool is available that generates the basic code for routines. A separate tutorial will soon be available for users who wish to incorporate C-code routines into the DSAM system. Alternatively, users can make suggestions for modules that might be included or directions that might be explored. Indeed, active users have suggested all of the modules presently in use.

2.2 Organic nature

DSAM is growing continuously and consists of a fixed library of tried and tested routines with a minority of routines that are still in development. Some routines are very stable, a minority are still being debated and moulded to suit the needs of users. At any given time, both a stable and a BETA version of the system are available. The stable is relatively problem free and is supported by the manual. The beta version includes the latest improvements and extensions with no guarantee that they have been thoroughly tested; BETA versions will have undergone functional testing, however it is never possible to know the behaviour of the system in every foreign situation.

This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Please bear the above in mind. We are not a commercial concern, though effort is made to present as professional a package as possible. This software is a shared effort. If you are having difficulty, get in touch and we shall try to help. If you have a comment or suggestion, let us have it. If we can use it, we shall.

Documentation is a nightmare for us. Not only is it time-consuming to produce, it is also rapidly out-dated by the continual developments that are taking place. The authors are also scientists and science has first priority, so the documentation is always lagging behind. The very best way to find out about DSAM and to learn its subtle ways is to visit Essex University and spend a day or two with us. You will be made very welcome. This is how most serious users got started.

2.3 Levels of user

Low level. Originally, all users were C-savvy programmers who borrowed or cannibalised C-code from the core-routines library.

High level. For some purposes, complete models have been prepared for end-users who did not wish to become involved in the detailed process of constructing models. Discuss your requirements with us, if you fall into this category.

Flexible modellers. Currently, our emphasis is on providing a system that allows users to build up models from fixed but flexible building blocks. This is AMS (the Auditory Modelling System). The rest of this document is aimed at flexible modellers. . Many different models can be produced. The flexible modeller can combine an AMS model with MATLAB or UNIX programming.

2.4 Free to University users

University researchers engaged on purely scientific studies can download and use this software freely. They are also welcome to consult the authors for help when required.

The software is not free to commercial users or users engaged on commercial contracts. We expect such users to contact us to discuss their requirements and any applicable charges.

3 Understanding AMS files (.sim, .par and .spf)

AMS is an application that allows the user to build auditory models using a simple scripting language. The script that defines a model is contained in a text file called a simulation file or a '.sim' file. The simulation file specifies a list of modules and specifies the order in which they are to be run.

Each module name may be accompanied by the name of a parameter or '.par' file. Parameter files are text files that specify the value of the parameters used in the accompanying module. If no parameter file is specified, AMS will supply default

parameters. Parameters can also be set and changed while AMS is running. Parameter files merely specify the parameter values at start-up.

Parameter and simulation files are created and modified using the text editor of your choice.

There is another kind of file called a 'simulation parameter file (or '.spf' file). AMS itself normally creates this file. It is a text file that can be edited in a text editor. It is a complete description of a simulation in a single file. It incorporates both simulation and parameter file specifications.

Examples of many simulation and parameter files can be found in the folders that accompany these tutorial notes.

3.1.1 Overview and example of .sim and .par files

Here is a simple but complete and viable simulation file. It requests AMS to generate a click and show it in a display window.

```
#           generateClick.sim
begin {
#  Stimulus generation
            Stim_Click        < stimClick1.par
            Display_Signal
      }
```

The important part lies between the left and right curly brackets. It consists of the names of two process modules; Stim_Click and Display_Signal. A parameter file is specified for the click generation module; stimClick1.par.

```
#           Stim_Click  < stimClick1.par

TIME        0.001       Time for the delta-function click (s).
AMPLITUDE   40          Amplitude (uPa).
DURATION    0.01        Duration (s).
DT          1e-05       Sampling interval, dt (s).
```

This parameter file requests a 40 μPa click after 1 msec in a total stimulus whose duration is 10 msec. The sample rate of the model is 100 kHz and the width of the click is a single sample (10 μsec).

3.2 The .sim file

3.2.1 Simulation file structure

This section gives a formal statement of the structure of a simulation file. You can skip this on first reading but it may prove useful later as a source of reference.

The simulation file is made up of a series of separate lines:

- Continuation across successive lines is not permitted.
- Blank lines are ignored.

- Lines in the file beginning with a '#' symbol are ignored by DSAM. These lines are used for comments.
- Anything occurring after the final curly bracket is ignored.

3.2.2 Header.

Scripts can begin with optional header information consisting of two parameters. The parameter name is the first item on each (optional) line, for example:

```
Diag_mode               on          Diagnostics mode - \off\ or \on\
Par_file_path_mode      relative    Parameter file path mode -
\relative\ or \absolute\
```

Diagnostics mode: on/off. . This instructs AMS whether or not to show run-time diagnostics in the AMS window. There are two possible values: 'off' and 'on'. This default setting is 'off'.

Parameter file path mode: relative/absolute. Parameter file paths are either *relative* to the folder containing the .sim file location or *absolute*. The default setting is 'relative'.

If a line is not added for either parameter, then the respective default settings will be used.

3.2.3 Process statements.

All process statements must lie inside the curly brackets of a *begin { }* sequence. Everything after the curly brackets is ignored.

Each line within the curly brackets must obey the following syntax

<switch> <label%> process name <path indicators> <parameter file name>

3.2.3.1 Switch.

An '@' can be used to switch off a module. This module can be enabled at run time, if required, using the edit\ simulation parameters window. To enable the module, open this window and find the module name in the list. It should have a'@' character beside it. Remove the '@' sign by double-clicking on the module name. It may be necessary to single click first (to open the parameter window) and then double click on the module name, this is a bug in the system.)

3.2.3.2 Labels

Labels are useful when a line will be referenced from elsewhere; for example in a path indicator or from a MATLAB program.

- Labels are optional.
- All labels consist of an ASCII string terminated with a '%'.
- The string may not contain a space.

3.2.3.3 Process names.

A list of process names can be found in the manual (help files) accessed from the AMS window: Help-> DSAM Help->Process modules reference library.

A process can, itself be a simulation (Util_SimScript). If this happens, the named simulation file is run at this point.

3.2.3.4 Path indicators

These define the source of the input to the process and the destination of the output.

The syntax of a path indicator is:

(input list -> output list)

- Path indicators follow immediately after the process name
- Path indicators are optional. If none is given, defaults are used
  - The default *input* to a process comes from the preceding process.
  - The *output* goes by default to the following process.
- The input list and the output list consist exclusively of labels defined elsewhere in the script. The % label indicator is omitted inside a list.
- If a line contains a path indicator, it must also have a label

For example:

acc% accumulate (a, b -> c, d)

This shows that the accumulate process will take its input from the two processes labelled *a* and *b*. The output will go to processes labelled *c* and *d*. The output will not go to the following process.

3.2.3.5 Parameter file names

These are preceded by a '<' character. Parameter files are optional. If they are not present, AMS supplies default parameters. The default parameters can be found in the manual (help files).

- The parameter file name is, in fact, the *address* of the parameter file.
- If the Parameter file path mode is set to 'absolute', this will be taken to represent the full file path.
- If the Parameter file path mode is set to 'relative', this will be taken to represent the file path starting with the location of the simulation file.

3.2.4 Repeat loops

A repeat loop can be placed anywhere within a *begin {...}* group.

```
repeat N {
      ......
```

```
                }
```

A group of process statements inside the curly brackets of a repeat loop will be computed N times, where N must be a positive integer.

Some process modules accumulate results (for example, ANA-histogram) inside a repeat loop unless actively reset. To reset a module use the *reset* command. For example

```
repeat 20   {
      ...
reset myHist
myHist%     Ana_Histogram     <myHistogram.par
            }
```

3.2.5 Editing .sim files

Changes can be made to .sim and .par files using a text editor. When a change is made (and the new text saved), it is necessary to reload the files into AMS because AMS cannot see that you have edited the files.

- Reload using the AMS window File menu 'File->Reload simulation file'.

3.3 Parameter files

The .par files contain parameters that are used by their calling process.

Each process requires its own unique set of parameters. A list of parameters can be found in the manual (help files) associated with its respective process. A list of process names and their parameters can be found in the manual (help files) in Help\ DSAM Help\ Process modules reference library.

Parameter files are optional. If no file is specified, the default parameters will be used. If a parameter file contains less than the full set of parameters, the missing parameters will be assigned default values. The values associated with the parameters in the manual are the default values.

A parameter file consists of a list of parameter name\value pairs; one pair per line.

- The parameter name is the first item on each line
- The parameter value is the second item on each line
- All subsequent material on the line will be ignored.
- Blank lines are ignored.
- All characters following the comment character '#' character will be ignored.

3.3.1 Changing parameters

Permanent changes. Change parameters by editing .par files using a text editor. When a change is made (and the new text saved), it is necessary to reload the files into AMS because AMS cannot see that you have edited the files.

- Reload using the AMS window File menu 'File->Reload simulation file'.

Temporary changes. Parameters can also be changed temporarily while AMS is open by using the parameter windows.

- Select 'Edit->Simulation parameters' from the AMS pull-down menus. This will open another window listing all the processes. Select a process by clicking on its name.
- When the parameter window for a given process is open, you can select and change any visible parameter. These changes will remain in force until the end of the session when they will be forgotten unless they are saved in a .spf file.
- It is not necessary to close the parameter window before running the program again. Just click on 'GO'.
- Parameters affecting displays can be changed in the same way. However, they can also be changed by right-clicking on the display and selecting 'preferences'. If you change the parameters in this way, you must close the parameter window before attempting any other action. The display will be changed without the need to rerun the program

3.4 .spf files

A simulation parameter file (.spf file) describes the complete model and *all* its parameters in a single file.

Its advantages include

- Compact and easily transportable.
- Gives a *complete* specification of the model including the process sequence and all parameters.
- It shows full process names including its label.

Its disadvantages include

- It is not easy to add new process modules to a model described by a .spf file.

3.4.1 Creating .spf files

- Run AMS
- Load the model of choice using the .sim file
- Select the option 'Save simulation pars' from the AMS window File menu.
- Name it. The file name must have the file extension '.spf'.
- Save it.

The .spf file is a text file that can be inspected using any text editor.

AMS is able to run the model using this file alone. To run a model using a .spf file, use the 'File' pull-down menu on the AMS window and select 'Load parameter file'. In some circumstances, this is the preferred mode of operation (only one file needed). It is particularly useful when you wish to share a model with a colleague who has no desire to manage or to delve into a whole collection of separate files.

After making temporary changes to parameters using the parameter windows in AMS, you may wish to preserve the changes. The best way is to immediately save the simulation parameters as an .spf file with its own name. You still have the original parameters in the .sim and .par files. These are unaffected by saving an .spf file.

Warning! If you save an .spf file, AMS will assume that this is now your current file and will store this assumption when you quit. When launching AMS at a later date, the system will automatically load the last file in use. If you saved a .spf file, AMS will automatically reload the .spf file. If you really want the original .sim file, you must reload it explicitly. This can be a trap for the unwary. If you find that changes to your .par files are not having any effect, this may be because the system is reading your most recent .spf file.

3.4.2 Example of simulation specification file

This ,spf file is based on the example given in 3.1.1.

```
##--------------- Simulation script ----------------------##
DIAG_MODE                       ON              Diagnostics operation mode
PAR_FILE_PATH_MODE              "RELATIVE"      Parameter file path mode

begin  {
       Stim_Click     <       stimClick1.par
       Display_Signal
}

##----- Stim_Click.0             (stimClick1.par) -----##
TIME.Stim_Click.0               0.001           Time for the delta-function click
AMPLITUDE.Stim_Click.0          40              Amplitude (uPa).
DURATION.Stim_Click.0           0.01            Duration (s).
DT.Stim_Click.0                 1e-005          Sampling interval, dt (s).

##----- Display_Signal.1                (not_set) -----##
MAGNIFICATION.Display_Signal.1        1               Signal magnification.
NORMALISATION.Display_Signal.1        "MIDDLE"
CHANNEL_STEP.Display_Signal.1 1               Channel stepping mode.
NUMGREYSCALES.Display_Signal.1        10              Number of grey scales.
X_RESOLUTION.Display_Signal.1 0.01            Resolution of X scale
WIDTH.Display_Signal.1        -1              Displayed signal width
Y_AXIS_TITLE.Display_Signal.1 ""              Y-axis title.
Y_AXIS_MODE.Display_Signal.1  "CHANNEL"       Y-axis mode
AUTO_SCALING.Display_Signal.1 ON              Automatic scaling ('on' or 'off').
AUTO_Y_SCALE.Display_Signal.1 ON              Automatic y-axis scale
MAXY.Display_Signal.1         0               Maximum Y value
MINY.Display_Signal.1         0               Minimum Y Value
Y_NUMBER_FORMAT.Display_Signal.1      "y"             Y axis scale number
Y_DEC_PLACES.Display_Signal.1 0               Y axis scale decimal places.
Y_TICKS.Display_Signal.1      15              Y axis tick marks.
Y_INSET_SCALE.Display_Signal.1        ON              Y inset scale mode
X_AXIS_TITLE.Display_Signal.1 ""              X axis title.
AUTO_X_SCALE.Display_Signal.1 ON              Autoscale option for x-axis
X_NUMBER_FORMAT.Display_Signal.1      "xe-3"          X axis scale number
X_DEC_PLACES.Display_Signal.1 0               X axis scale decimal places.
X_TICKS.Display_Signal.1      6               X axis tick marks.
X_OFFSET.Display_Signal.1     0               X offset for display in zoom mode
X_EXTENT.Display_Signal.1     -1              X extent for display in zoom mode
WIN_TITLE.Display_Signal.1    "Display_Signal.1"      Display window title.
MODE.Display_Signal.1         "LINE"          Display mode
SUMMARYDISPLAY.Display_Signal.1       OFF        Summary display mode
FRAMEDELAY.Display_Signal.1   0               Delay between display frames (s)
TOPMARGIN.Display_Signal.1    5               Top margin for display
WIN_HEIGHT.Display_Signal.1   500             Display frame height
WIN_WIDTH.Display_Signal.1    440             Display frame width (pixel units).
WIN_X_POS.Display_Signal.1    0               Display frame X position
WIN_Y_POS.Display_Signal.1    0               Display frame Y position

<<>> Simulation parameter file divider.
```

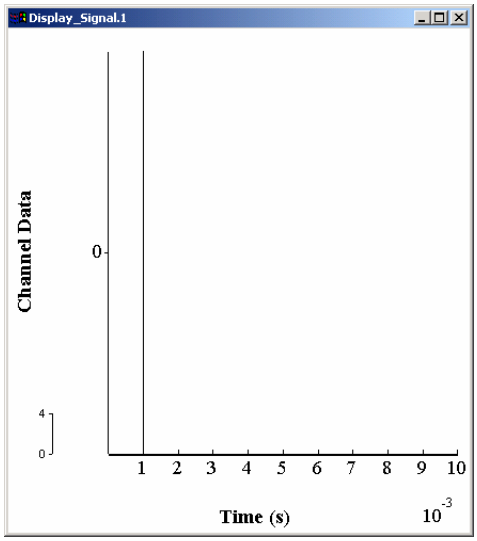
```
        DIAG_MODE.ams.0                 "OFF"           Diagnostics mode specifier
        SIM_FILE.ams.0                  "C:\Program Files\DSAM\AMS\resDvlpt\tutorials
\generateStimulus\generateClick.sim"                    Simulation file.
        SEGMENT_MODE.ams.0              ON              Segmented or frame-base processing
        # Sub-parameter list:
          FILELOCKING_MODE.ams.0        OFF             File locking mode ('on' or 'off').
          NUM_RUNS.ams.0                1               Number of repeat runs
```

3.5 Running a simulation

The files described above are located in a folder
'Tutorials\GenerateStimulus'

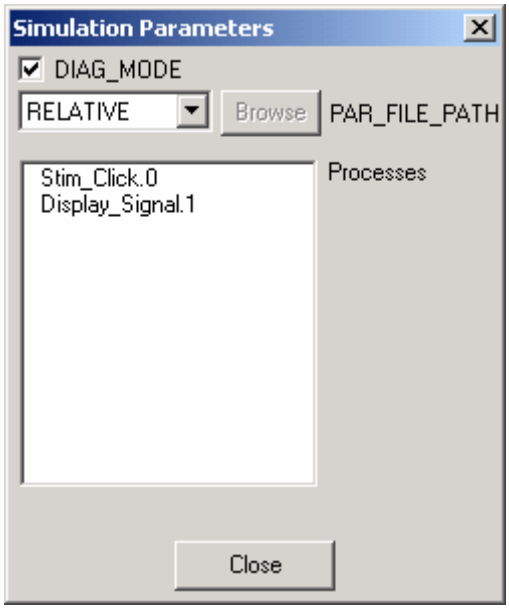
You can run this program in the following way

1. Launch AMS. The AMS window will appear on the screen.
2. Select from the pulldown menus- File -> Load script file (*.sim)...
3. Open file 'generateClick.sim'
4. Click on 'GO'
5. You should then see the display screen below.

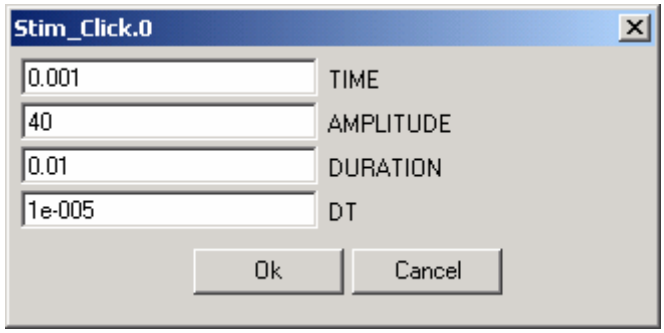


You can change the parameters while AMS is running. In the followiing example we shall change the time of occurrence of the click from 1 to 5 msec.

1. Select from the pulldown menus- File -> edit\ simulation parameters. You should see a display like this



2. Click on the process Stim_Click. This should produce a display like this



3. Change the click time (TIME) from .001 to .005.
    1. Click on 'GO' in the AMS window. The click in the display should move to the later time.

4 Using AMS with MATLAB

***** new section on runDSAMsim*********

Use the folder MATLABrunDSAMsim for the programs and models described below.

A special MATLAB call (RunDSAMSim) allows you to control AMS from MATLAB.

This function is stored in c:\dsam\matlab. If you want to use it you should include this folder in your MATLAB list of paths using 'setpath'.

The help file for runDSAM sim is as follows

```
>> help RunDSAMSim

  RunDSAMSim:

  [data, info] = RunDSAMSim(<sim file>, [<parameter settings>, [<diag. mode>,...
                            [<signal>, [<signal info.>]]]])

  <sim file>           Simulation file name (string).
  <parameter settings> Parameter '<name> <value> ...' pairs (string).
  <diag. mode>         Diagnostic mode 'screen', 'off' or '<file name>'.
  <signal>             Data signal ([chan, samples] real matrix).
  <signal info.>       Signal information (structure).
```

Example
The simplest way of running an AMS file is to set the MATLAB path to the folder containing the .sim file and then call RunDSAMSim

```
>> cd myFolder
>> [data, info] = RunDSAMSim('mySimFile.sim');
```

MATLAB will then run the AMS model specified in my 'simFile.sim'. The result of the AMS run is stored in 'data' and details of the model are stored in the cell array 'info'. The result of the AMS run is the state of the model after the execution of the final module. Typically this will be a two-dimensional matrix of BF against time but other possibilities exist.

Notes

1. You can specify either a .sim file or a .spf file
2. Any calls in the .sim file to Display_Signal will be ignored. RunDSAMSim uses a special form of AMS called 'ams_ng'. 'ng' stands for 'no graphics'.
3. It is not necessary to write any data to file using the 'dataFile_out' module. However, if you do write to files, these will be available for reading as soon as computation is complete. A routine to help read files into MATLAB is described below.

Trouble shooting

1. Does your current directory contain the .sim file you are referring to?

2. Does your .sim file work independently of MATLAB? To check this, load up AMS and call the .sim file directly. If it does not work with AMS, it will not work with MATLAB!
3. Is c:\dsam\matlab on your MATLAB filepath?
4. Was RunDSMSim properly installed? 'C:\DSAM\Matlab' should contain a file called 'RunDSAMSim.dll'.
5. Was 'ams_ng.exe' installed as an executable file? It should be found in c:\dsam\ams .

4.1 Example Program

The following example shows how MATLAB can pass a signal directly to AMS, run a model and collect the results directly without writing any data to file.

The sim file is

```
# gammatoneDemo.sim

diag_mode      on

begin {

# stimulus will be supplied by MATLAB calling program


# Outer-/middler-ear filter model.
               Filt_MultiBPass       < filtMultiBandpassGP.par

#convert to stapes velocity
               Util_mathOp           < mathOpStapes.par

# Basilar membrane filter model.
gammaT%        BM_gammaT             < BM_gammatone.par


# Result will be collected directly by the MATLAB calling program

       }
```

This model applies a bandpass filter (to simulate the onuter-middle ear), a scalar (to convert the pressure to stapes velocity and a set of gammatone filters.

The MATLAB program prepares the stimulus, passes it to AMS using the runDSAMsim function and then collects the output in the info.data vector.

```
function matlabDemoRunDSAMsim
% matlabDemo is a skeleton script for running a modle using AMS_ng

% Set the directory to the location of the .sim file
% If the file is not in the current directory, specify the full file path
simFilePath= 'gammatoneDemo.sim';

% Specify any parameters to be changed.
% This is a single continuous string of pairs of parameters:
%    <name> <value> <name> <value>    etc.
% To find the correct form of each parameter name, create a .spf file from
% the .sim file
lowestBF=1000;  % Hz
highestBF=1500; % Hz
numChannels= 2; % #

pars=[ ...
```

```
         ' MIN_CF.BM_gammaT.gammaT '  num2str(lowestBF) ...
         ' MAX_CF.BM_gammaT.gammaT '    num2str(highestBF) ...
         ' CHANNELS.BM_gammaT.gammaT '  num2str(numChannels) ...
     ];

% Specify diagnostic mode
diagMode='ON';

% Create the input signal
frequency=1000;     % Hz
duration= 0.1;      % s
sampleRate=100000;  % Hz
dt=1/sampleRate;
t=dt:dt:duration;   % NB first time is dt (not 0).
signal= sin(2*pi*frequency*t);

plot(t, signal)

% Store details for AMS
info.dt=dt;
info.length=length(signal)

% Now run AMS
[data info]=runDSAMsim (simFilePath, pars, 'ON', signal, info);

info
plot(t, data)
```

The program is mainly self explanatory but some points are worth noting.

1. runDSAMsim allows you to change individual parameters in the model. The changes are coded into a string consisting of successive pair of strings separated by spaces. Each pair consists of the name of the parameter to be changes and the new value. The name can be found by inspecting the corresponding .spf file for the model.
2. The name can change between sessions if the .sim file is altered. To prevent this, make sure that a module has a label if it is intended to change one of its parameters. In the example, the module BM_gammaT contains parameters to be changed. For this reason a label gammaT% is given to the module. The parameter names are have the affix gammaT% and the name will not change even if new modules are added at a later date.
3. Take care to use the correct and full name of the variables and have spaces between each string. The following layout is useful for keeping track of what is happening, especially if there are many parameters to be changed.

```
pars=[ ...
         ' MIN_CF.BM_gammaT.gammaT '    num2str(lowestBF) ...
         ' MAX_CF.BM_gammaT.gammaT '     num2str(highestBF) ...
         ' CHANNELS.BM_gammaT.gammaT '  num2str(numChannels) ...
     ];
```

4. The signal is a one dimensional vector of values for a mono signal. AMS needs to know the length of the vector and dt, the time interval between samples. This information is placed in info;

```
info.dt=dt;
info.length=length(signal)
```

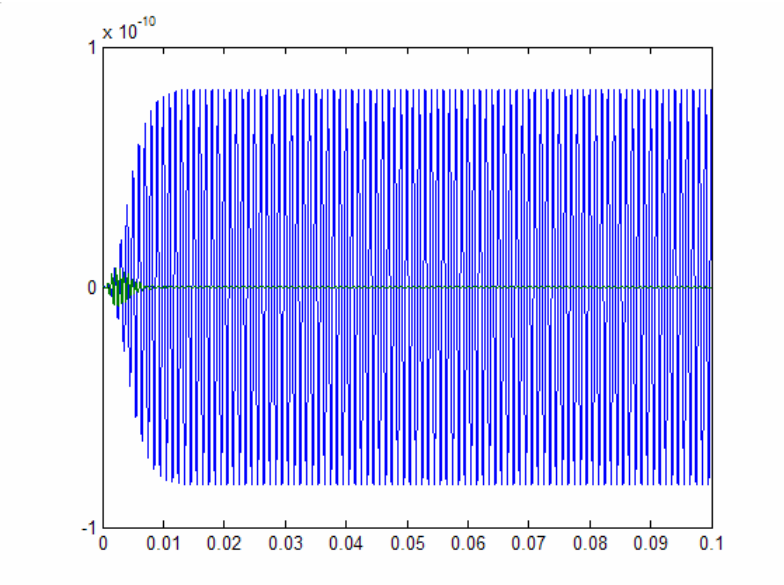
5. All of this information is used in the runDSAMsim call. The string ‘ON’ specifies that the model is run in diagnostic mode (recommended).

```
[data info]=runDSAMsim (simFilePath, pars, 'ON', signal, info);
```

6. The output from the model is stored in data. The sampling interval is returned in info.dt. info.labels gives the channel centre frequencies.

```
[data info]=runDSAMsim (simFilePath, pars, 'ON', signal, info);
```

The output from the program consists of two superimposed waveforms; one large and one small (the output from the two filters).



4.2 Reading .dat files

The .sim file can request files to be written using dataFile_Out. If you intend to read these into MATLAB, they should be specified as text files with names ***.dat.

readDatFile is a routine that can be used to read most text files generated by AMS. A copy of this program is given in the folder.

```
[results, column1Values, columnHeadingsValues, errorMsg]=readDatFile(datFileName)
```

The routine reads a .dat file and decomposes it into

Results	the values in the body of the matrix
Column1Values	the values in column (normally time)
columnHeadingValues	the values in the first row of the .dat file (normally channel best frequencies)
errorMsg	empty or bad news

5 File input/output in AMS

AMS uses the processes DataFile_In and DataFile_Out to handles files.

5.1 File types

The file types that can be used are identified by their file extension

- .DAT ASCII files produced by and readable by a text editor. Values are successive values of a waveform. For binaural signals, pairs of values are used.
- .WAV signal files using the *.wav* format and having values between +1 and –1. MATLAB is a useful source of .wav files
- .AIF signal files using the *AIFF* format. These files preserve the absolute value of the signal. Unfortunately, MATLAB does not create or read this type of file. (A ReadAIFF.m MATLAB is part of the AMS installation. It can be found in the C:\DSAM\AMS\Matlab directory with its supporting files. The directory can be added to your matlab path.)
- .raw binary files.

5.2 Parameters

The parameter files for *DataFile_In* and *DataFile_Out* have the same format. There are nine parameters. Not all are required all of the time. For example, when using *DataFile_Out* it is possible to omit the parameter file altogether (if you are happy with the default file name *output.dat*)

FILENAME	Make sure that the filename extension is included. This must be one of the above types. (*default* 'output.dat'). AMS assumes that the file to be read is in the same folder as the .sim (or .spf) file. If it is somewhere else, the full filepath must be specified.
GAIN	This gain will be applied to the values read in; (*default* 0). Setting the gain can be tricky and is described in more detail below.
STARTTIME	To ignore the initial part of the signal, use this parameter to specify how much to ignore; (*default* 0).
DURATION	Specifies how much of the signal is to be used. A value of –1 indicates that the signal is to be used to its end; (*default* –1). DURATION has a special meaning in SEGMENT mode (section 5.4 below).
SAMPLERATE	If the file contains its own sample rate (e.g. .wav and .aif), this parameter is ignored. For .dat and .raw, this value should be set; (*default* 8000).
CHANNELS	Distinguishes monaural (*value* 1) from binaural (*value* 2). This parameter is ignored when using .wav and .aif files; (*default* 1).

WORDSIZE	Number of words to be used for each point in the waveform; (*default* 2). Used only for binary files.
ENDIAN_MODE	Values are BIG, BIG_UNSIGNED, LITTLE, LITTLE_UNSIGNED and DEFAULT. This parameter is used when transferring files from computers with different operating systems (e.g. Mac to IBM PC). If a file cannot be read using the default, try all the others in turn and hope for the best!
NORM_MODE	Applies a correction factor. Used only for binary files.

AMS output files can be used for reinput to AMS. If the aim is to stage the AMS analysis by producing an output file to be used for restarting the process at a later date, a *.aif* file type should be used as this preserves all aspects of the signal used by AMS.

5.3 File input

Any of the above file types can be used for reading in a new signal that has been produced using other software such as audio capture software or user programs.

5.3.1 Text or ASCII files

When signals at input using text or ASCII files, you should observe the following rules:

- It must have the file extension '.dat'
- Make sure that this is a genuine text file (not a word-processed file)
- The signal values are a column of figures one value wide.
- SAMPLERATE must be set in the .par file (AMS cannot guess this)
- Specify DURATION (AMS will not work this out for you)

5.3.2 Gain setting

This can be a major headache for beginners, so listen up.

AMS assumes that all stimulus input files are given in micro Pascals (μPa). If they are not in this form, then the GAIN parameter can be used to set matters aright. Most auditory filter systems are nonlinear in nature and the nonlinearity is applied differently at different levels. It is important that the level of the signal is specified correctly.

5.3.3 Gain setting for WAV files

WAV is a common file format use by much audio capture software. All *.wav* files are compressed to have values within the limits of +1 to –1. A .wav file contains no

information as to the original amplitude of the signal. The user must specify a GAIN function that will restore a signal to its original intensity. After restoration, the waveform will be specified in μPa. Choosing an appropriate GAIN function can be tricky.

When AMS reads a .wav file, it reads the values as if they were between +0.5 and -0.5 μPa . There are complex historical reasons for this. If you forget this (and think that the AMS sees the input as between +/- 1 μPa) your level will be reduced by 6 dB.

To specify the GAIN you need to specify the peak amplitude in μPa. If you do not know the peak value, you cannot specify the GAIN. If you do, we can attempt the necessary calculations. Table 1 may be helpful.

peak level dB	Peak mPa	AMS GAIN
0	28	35
20	283	55
40	2828	75
60	28284	95
80	282843	115
100	2828427	135

Table 1 GAIN to be applied to a .wav file to achieve specified peak amplitudes. This table assumes that the peak value in the .wav file is +1. Note that the peak value dB will normally be greater than the rms value in dB SPL.

As a rule of thumb, speech-level sounds should have a gain of around 95 dB to achieve an rms level between 50 and 60 dB SPL.

If you are creating your own .wav files using MATLAB, make sure that the data are normalized before using the *wavwrite* function.

Example 1
Our first example is very simple but illustrates the principles.

Create a .wav file using a pure tone (sinusoid). The wav file will have a peak value of 1. We wish to convert this to a pure tone stimulus at 0 dB SPL.

Solution: A 0 dB pure tone has an rms of 20 microPascals and a peak value of approximately 28 microPascals (you just have to know that). To convert our .wav file peak of +1 to 28 we need to apply a scalar of 28. This is equivalent to a gain of

20*log10(28/0.5) = 35 dB.

Remember that AMS sees the .wav file as numbers in the range +0.5 to -0.5 (see above).

Set the GAIN parameter in the *DataFile_In* process to 35.

Example 2
Well, actually, I wanted 56 dB SPL.

Solution: add 56 to 35 to get the desired GAIN of 91 dB.

Example 3
I don't care about the peak value. I want to set an overall rms value for my *wav* file (dBrmsDesired).

Solution: Measure the rms value of the .wav file (dBrmsMyFile).

Remember that AMS sees the .wav file as numbers in the range +0.5 to -0.5 (see above).

Your required GAIN is

GAIN= dBrmsDesired – dBrmsMyFile

Example 4
How do I get AMS to measure my rms value? Not all .wav files are properly normalized – especially those created by user programs in MATLAB(!).

Use the module Ana_Intensity. Create a .sim file like this:

```
# intensity.sim

begin  {
      DataFile_In       < fileIn.par
      DisplaySignal
      Ana_Intensity
      DataFile_Out
       }
```

and a .par file (called *fileIn.par*) like this

```
#     DataFile_In       < fileIn.par

FILENAME    myfile.wav
```

This program will display the whole wav file and compute its level (rms, re 0 dB SPL).

The level will appear in a file called 'output.dat' in the current directory. Both the left and right channels are assessed. Only one value is given for monaural signals.

```
Time (s)                  0           1
0.00000e+000       -39.3021    -39.3021
```

In this case the signal would require a GAIN of 39.3 dB to restore it to a 0 dB SPL signal and a GAIN of 89.3 to restore it to a 50 dB SPL signal.

This useful little program can be found in the *Tutorials\MeasureIntensity* folder.

5.4 File use in SEGMENT mode

In SEGMENT mode AMS analyses equal-length sections of the input file one at a time. The parameter DURATION specifies the length of each segment. The analysis is repeated for each segment of the input file. When a dataFile_Out module is encountered, the current signal is appended to the output file.

SEGMENT MODE and NUM_RUNS are specified at run time using the AMS window pull-down menus Edit\Main parameters and the General and Specific tabs respectively.

If NUM_RUNS is set to -1, AMS will compute the number of segments for you and reset NUM_RUNS to the appropriate value when you click on 'GO'.

Example
Compute the rms level of the signal over 5 msec segments for the first 5 segments using the intensity.sim program given in section 5.3.2 above.

Load intensity.sim
Set DURATION to .005
Set SEGMENT mode ON
Set NUM_RUNS to 5
GO

AMS now computes the intensity of the first five 5-ms segments of the signal. The output file looks like this

Time (s)	0	1
0.00000e+000	-52.0259	-52.0259
1.00000e+000	-36.2996	-36.2996
2.00000e+000	-52.0259	-52.0259
3.00000e+000	-36.2996	-36.2996
4.00000e+000	-52.0259	-52.0259

6 Gammatone filterbank

This model can be found in the folder *Tutorials\AuditoryPeriphery*. The original version of this folder can be found in *DSAM\AMS\ Tutorials\AuditoryPeriphery*. If you have not already done so, copy this folder into your regular workspace. Please do not work in the *AMS* workspace.

Launch AMS and load 'filterbankGammatone.sim'. Hit 'GO'. You should see the displays shown in Figure 2

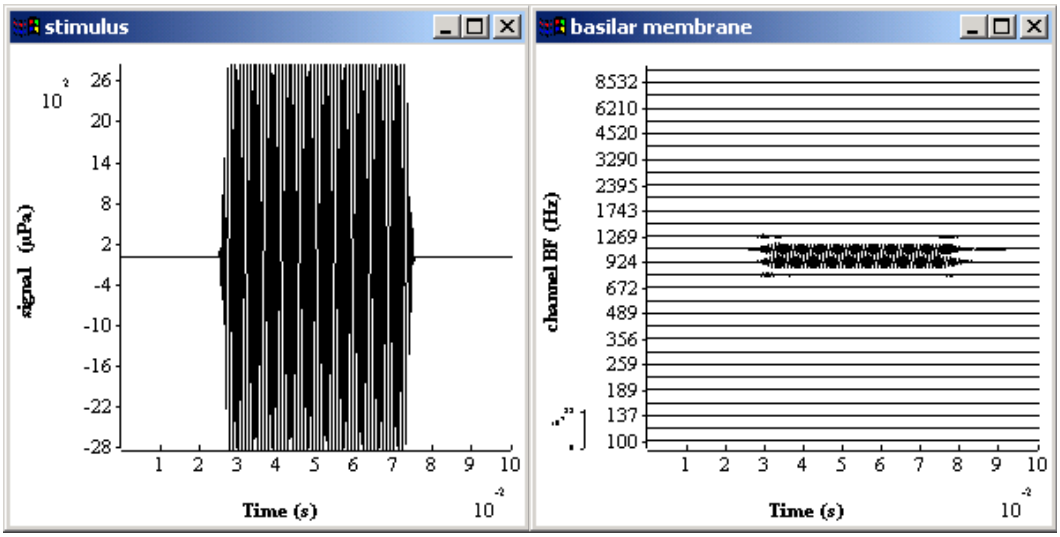


Figure 2 Output from filterbankGammatone.sim.

The stimulus display shows a 50-msec, 40-dB SPL, 1-kHz signal, shaped by a 2.5-msec raised cosine onset and offset ramp. The basilar membrane window shows the response of a gammatone filterbank consisting of 30 filters with BFs between 100 and 10000 Hz equally spaced on a ERB scale. The bandwidths of the filters are based on a proposal by Moore and Glasberg (??).

6.1 Simulation file.

```
# filterbankGammatone.sim

begin {

# generate stimulus
            stim_pureTone_2    < stimTone.par

# Ramps
            Trans_Gate         < rampUp.par
            Trans_Gate         < rampDown.par
            Display_Signal     < displaySignal.par

# Outer-/middler-ear filter model.
            Filt_MultiBPass    < filtMultiBandpassGP.par
```

```
#convert to stapes velocity
            Util_mathOp       < mathOPstapes.par

# Basilar membrane filter model.
            BM_gammaT         < BM_gammatone.par
            Display_Signal    < displayBM.par
            DataFile_out

      }
```

The sequence of operations in the simulation file is

- Generation of the stimulus (stim_pureTone_2)
- Application of onset and offset ramps (Trans_Gate).
- Outer\middle ear is simulated using two parallel bandpass filters (Filt_MultiBPass) and a scalar (Util_mathOp)
- Computation of the gammatone filterbank (BM_gamma_T)
- Writing the result to a file (DataFile_out).

A more detailed account of each stage is given along with the individual parameter files below.

6.2 processes and parameters

6.2.1 Stimulus generation

The parameter file stimTone.par reads as follows:

```
#           stim_pureTone_2   < stimTone.par

frequency         1000        #Hz
intensity         40          #dB SPL
duration          .05         #seconds
dt                .00001      #100 kHz
begin_silence     .025        #seconds
end_silence       .025        #seconds
```

The first line is a comment. It is good practice to paste the calling line from the .sim file as the first line of the parameter file. This shows the name of the .par file in a printout and offers a reminder of the name of the calling process. This practice can be a great help to a third party advising on any problems you may be experiencing.

DURATION is the duration of the pure tone. Silences are <u>additional</u> to the tone. The total duration is 0.1 s.

DT sets the sample rate for the whole model, including all succeeding processes. DT will be checked by AMS to make sure that it is small enough to satisfy the requirements of all filters that appear later in the simulation script.

6.2.2 Ramp on/off

Two parameter files; one for ramping up and one for ramping down.

```
#           Trans_Gate        < rampUp.par

OP_MODE     RAMP
POS_MODE    RELATIVE
DURATION    2.5E-3

#           Trans_Gate        < rampDown.par

OP_MODE     DAMP
POS_MODE    RELATIVE
DURATION    2.5E-3
```

TRANS_GATE is a process that transforms the stimulus by applying a gating function. It takes a number of parameters but only three key parameters are shown here. The other parameters are supplied by default.

OP_MODE specifies whether the ramp is up (RAMP) or down (DAMP).

POS_MODE specifies whether the ramp is applied at the beginning of the signal (ABSOLUTE) or at the onset of the tone (RELATIVE).

DURATION specifies the length of the ramp.

Tip: Ramps must be included when using internally generated signals (except clicks). A warning will be issued if no ramps are set.

6.2.3 Display signal

```
#           Display_Signal    < displaySignal.par

WIN_TITLE   stimulus

WIN_HEIGHT  300
WIN_WIDTH   300
WIN_X_POS   0
WIN_Y_POS   0

Y_AXIS_MODE LINEAR
Y_AXIS_TITLE      "signal   (µPa)"
```

Display_signal is a very complex function that takes many parameters. Only a few are required for this display, however.

WIN_TITLE allows the user to specify text in the (blue) band at the top of the display. If the text requires spaces, it should be given inside double quotes.

WIN_HEIGHT, WIN_WIDTH are specifications in pixel units of the height and width of the display windows.

WIN_X_POS, WIN_Y_POS specify the location (in pixel units) of the top left hand corner of the display relative to the screen. (0, 0) specifies the top left of the screen.

In general, the location and size of the display will vary with different screen sizes and different resolutions.

Y_AXIS_MODE defines the meaning of the numbers alongside the y_axis.
CHANNEL defines the numbers as representing the best frequency (BF) of the individual channels. LINEAR defines the numbers as representing the numerical value of the function, as in a normal graph. The signal has only one channel, so LINEAR is the appropriate choice.

Y_AXIS_TITLE specifies the text to be written alongside the y axis. The X_AXIS_TITLE is not given because the default value for this is 'time', and that is appropriate for this display.

6.2.4 Outer/middle ear filter

The outer and middle ear attenuate high and low frequency sounds. Here we have chosen to simulate a guinea pig outer/middle ear function using two parallel bandpass filter following Sumner et al. (2003)[1].

```
#           Filt_MultiBPass   < filtMultiBandpassGP.par

NUM_FILTERS       2

CASCADE     0:2
ATTENUATION 0:0
LOWER_FREQ  0:4000
UPPER_FREQ  0:25000

CASCADE     1:3
ATTENUATION 1:0
LOWER_FREQ  1:700
UPPER_FREQ  1:30000
```

The parameter NUM_FILTERS specifies the number of filters. Each filter must be fully described in terms of its parameters. The notation *i:param* specifies the parameter for the i^{th} filter. Filter numbers are sequenced 0,1.. N-1.

Each filter is composed of a cascade of first-order filters where the 3-dB down points of the first-order filters are specified by the parameters LOW_FREQ and UPPER_FREQ. The parameter CASCADE specifies the number of cascaded filters.

A cascade of 3 filters, for example, is similar to but not the same thing as a third order filter. A filter consisting of a cascade of three first-order filters has high and low frequency skirts with the same slope as a third order filter but the upper and lower cut-offs will be narrower than those specified in the parameters. In this case the LOW_FREQ and UPPER_FREQ parameters specify the 9-dB down points of the filter.

[1] In fact, Sumner recommends a cascade of two filters but parallel filters appear to give a better fit to the animal data.

ATTENUATION is the reduction in level between the input and output at BF. It is specified in dB.

6.2.5 Conversion to stapes velocity

AMS uses realistic physical units whenever possible. The input signal is in micro Pascals. The next stage requires that the pressure representation be changed to stapes velocity. The outer/middle ear is linear in operation. Therefore, this can be achieved by applying a scalar using Util_Mathop. The scalar used is 1.4e-10 following the suggestion of Sumner et al (2003).

```
#           Util_mathOp       < mathOPstapes.par
OPERATOR    SCALE
OPERAND     1.4E-10
```

6.2.6 Gammatone filter

The BM_gammaT process computes the gammatone filter output at a number of different BFs. It is a process with many parameters. A small number of key parameters are shown in the parameter file.

```
#           BM_gammaT         < BM_gammatone.par

CHANNELS    20
MIN_CF      100
MAX_CF      10000
CASCADE     4
CF_MODE     HUMAN
B_MODE      ERB
```

CHANNELS specifies the number of channels to be used. CHANNELS must be a positive integer greater than 1 (see 8.1.2.1 below for example of single channel operation).

MIN_CF and MAX_CF specify the range of CFs to be used.

CASCADE indicates the number of cascaded first-order gammatone filters that are used to make a single filter. The greater the cascade value, the steeper the slope of the filter skirts.

CF_MODE refers to the method of spacing the channel CFs. When set to 'HUMAN, the BFs are equally spaced on a Greenwood scale based on the dimensions of the human basilar membrane.

B_MODE specifies the method for computing bandwidths. ERB is based on the formula suggested by Moore and Glasberg
If a single filter is required, set the CF_MODE to 'single' and use the parameter SINGLE_CF to set its CF.

6.2.7 BM display

```
#           Display_Signal    < displayBM.par

WIN_TITLE   "basilar membrane"

WIN_HEIGHT  300
WIN_WIDTH   300
WIN_X_POS   300
WIN_Y_POS   0

Y_AXIS_MODE CHANNEL
Y_AXIS_TITLE      "channel BF (Hz)"
```

Only new parameters will be explained here. For an explanation of other parameters, see 6.2.3.

Y_AXIS_MODE is set to 'CHANNEL' to request that channel CFs be identified along the y-axis. A scale is shown in the bottom left-hand corner of the display to help interpret the positive and negative deflections of the individual functions. This scale can be small and may be necessary to maximise the display to read this clearly.

6.2.8 File output

The process *DataFile_Out* automatically writes the result to a file called, by default, 'output.dat'.

In this case, the output file consists of a matrix with 30 columns (1 for each channel) and 10000 rows (1 for each time sample). In addition, the first column indicates the time of each row and the first row indicates the CF of each channel.

Tabs separate the columns. This makes the files suitable for reading directly into Excel and other spreadsheets.

Tip: If you want to give your output file a different name, use a parameter file with a single parameter:

```
FILENAME          myfileName.dat
```

It is important to use .dat as the file ending if you want to look at the file in a text editor.

6.3 Things to do

6.3.1 Switch off diagnostics

When you are happy that the model is running satisfactorily, switch off the diagnostics by adding a line to the top of the .sim file.

```
Diag_mode   off         Diagnostics mode ("off" or "on").
```

6.3.2 Change the BF range

- Run the filterbankGammatone model
- Select Edit\ Simulation parameters.. from the AMS window pull down menus
- Click once on BM_gammaT.6 in the Simulation Parameters window.
- Change MIN_CF and MAX_CF to new values (say 500 and 3000)
- Click on GO (no need to close the parameter window).

7 Nonlinear filterbank

This model can be found in the folder *Tutorials\AuditoryPeriphery*. The original version of this folder can be found in *DSAM\AMS\|Tutorials\AuditoryPeriphery*. If you have not already done so, copy this folder into your regular workspace. Please do not work in the *AMS* workspace.

Gammatone filters are linear in the sense that their properties do not change with signal level. However, the auditory periphery is known to be nonlinear. As the signal level increases, the bandwidth of the filter increases and the frequency that elicits the biggest response (the best frequency, or BF) shifts. To meet the requirements of nonlinearity we need a nonlinear filterbank. The method adopted here is the dual resonance nonlinear (DRNL) filter, described in detail elsewhere: Meddis et al, (2001) and Lopez-Poveda and Meddis R (2001a).

The .sim file below contains a new model that is exactly the same as that given in *filterbankGammatone.sim* above except for two changes.

- The line

  ```
  @           BM_gammaT          < BM_gammatone.par
  ```

  now contains a '@' character. This disables the BM_gammaT process but leaves it in the model so that it can be enabled at run time. This will allow comparisons to be made with the nonlinear filterbank.

- A new basilar process has been introduced to implement the nonlinear filters

  ```
              BM_DRNL            < BM_drnlGP.par
  ```

7.1 Simulation file

```
# filterbankDRNL.sim

begin {

# generate stimulus
            stim_pureTone_2    < stimTone.par

# Ramps
            Trans_Gate         < rampUp.par
            Trans_Gate         < rampDown.par
            Display_Signal     < displaySignal.par

# Outer-/middler-ear filter model.
```

```
            Filt_MultiBPass    < filtMultiBandpassGP.par

#convert to stapes velocity
            Util_mathOp        < mathOpStapes.par

# Basilar membrane filter model.
@           BM_gammaT          < BM_gammatone.par
            BM_DRNL            < BM_drnlGP.par
            Display_Signal     < displayBM.par
            DataFile_out

      }
```

7.2 DRNL parameter file

The parameter file for the DRNL filter is complex but necessarily so. Normally, this need not concern most users because standard sets of parameters are becoming available for animal and human applications. Default parameters will also be useful while setting up models. This means that you can omit the parameter file altogether. However, you will normally need to use a tailored .par file for publishable studies.

The .par file shown below is based on the filterbank proposed for the guinea pig by Sumner et al. (2003). The parameters are based directly on table I in that paper. Their table is reproduced below the .par file for comparison purposes.

The nonlinear system consists of two separate pathways, a linear and a nonlinear pathway. Parameters are given separately for each pathway and tagged as NL or NONLIN for the nonlinear path and LIN or LINEAR for the linear path.

Many key parameters vary as a function of CF. By default these parameters are specified as a logarithmic function of CF

$$\log(\text{param}) = p(0) + p(1)\log(CF)$$

For example, in the case of the bandwidths of the filters in the nonlinear path, p(0) and p(1) are set to 0.8 and 0.58 using

```
NONLINBWIDTH_PARAMETER  0:0.8
NONLINBWIDTH_PARAMETER  1:0.58
```

At 1 kHz, the nonlinear bandwidth can be computed:

$$NLBW = 10^{(0.8 + 0.58 \log_{10}(1000))} = 347 \text{ Hz}$$

With nonlinear filters, the distinction between BF (best frequency) and CF (centre frequency) becomes important. We try to use the term CF to refer to a parameter of a component filter while BF refers to the frequency that yields the greatest response at the output of the system as a whole. For example, BF changes with level (it is an emergent property) while CF (the description of a passive component) does not change. Notice that, in the DRNL model, the CF of the nonlinear path is not normally the same as that for the linear path.

When computing model parameters, CF is normally taken to be the CF of the filters in the nonlinear path. CF_MODE controls the spacing of the CFs. When it is set to 'GUINEA_PIG', the CFs are equally spaced along a Greenwood (1990) scale whose parameters are based on guinea pig measurements.

```
#           BM_DRNL           < BMdrnlGP.par

# nonlinear path
NL_GT_CASCADE           3     Nonlinear gammatone filter cascade.
NL_LP_CASCADE           4     Nonlinear low-pass filter cascade.

NONLINBWIDTH_PARAMETER  0:0.8
NONLINBWIDTH_PARAMETER  1:0.58

COMPRSCALEA_PARAMETER   0:1.87
COMPRSCALEA_PARAMETER   1:0.45

COMPRSCALEB_PARAMETER   0:-5.65
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.


# linear path
LINCF_PARAMETER         0:0.339
LINCF_PARAMETER         1:0.895

LINBWIDTH_PARAMETER     0:1.3
LINBWIDTH_PARAMETER     1:0.57

LINSCALEG_PARAMETER     0:5.68
LINSCALEG_PARAMETER     1:-0.97

# CF List Parameters:-
DIAG_MODE               PARAMETERS
CF_MODE                 GUINEA_PIG
MIN_CF                  300   Minimum centre frequency (Hz).
MAX_CF                  15000 Maximum centre frequency (Hz).
CHANNELS                20    No. of centre frequencies.
```

Table 2 ***Basilar membrane filtering (DRNL) parameters taken from sumner et al ( ) Table I..***

Parameters that are fixed across all BFs			
Compression exponent, v (dB/dB)	0.1		
Gammatone cascade of non-linear path	3		
Low-pass filter cascade of non-linear path	4		
Center frequency of non-linear path, CF_{NL}	Set equal to BF		
Low-pass cut off of non-linear path, LP_{NL},	Set equal to BF		
Gammatone cascade of linear path	3		
Low-pass filter cascade of linear path	4		
Low-pass cut-off of linear path, LP_{lin}	Set equal to CF_{lin}		
Parameters that vary with BF: $p(BF) = 10^{p_0 + m\log_{10}(BF)}$	Filter-bank coefficients		Single filter at 6kHz BF (in Figure 4B). Filterbank values shown in brackets.
	p_0	m	
Bandwidth (Hz) of non-linear path, BW_{NL} (Hz).	0.8	0.58	980 (unchanged)
Compression parameter, a	1.87	0.45	251 (3716)
Compression parameter, b	-5.65	0.875	4.52×10^{-3} (unchanged)
Center Frequency of linear path, CF_{lin} (Hz).	0.339	0.895	2961 (5253)
Bandwidth of linear path, BW_{lin} (Hz).	1.3	0.53	634 (2006)
Linear path gain, G_{lin}.	5.68	-0.97	103 (unchanged)

7.3 Model output

The output should look like Figure 3.

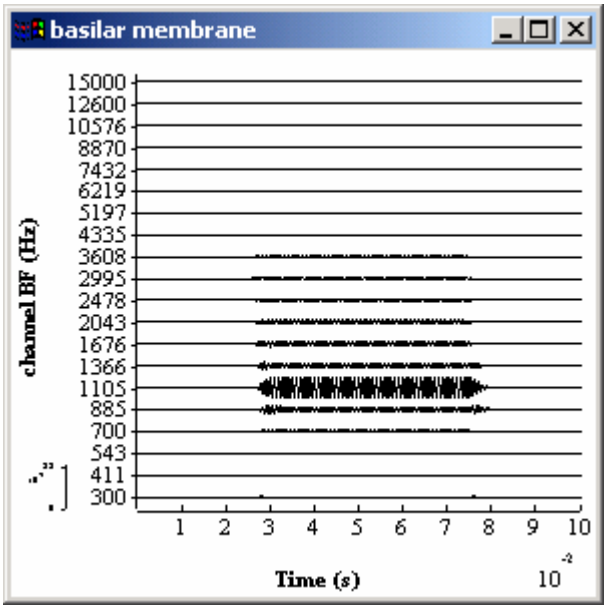


Figure 3 DRNL output

7.4 Things to do

7.4.1 Compare the output of the DRNL with the gammatone filter.

1. Select Edit\ Simulation parameters from the AMS window pull-down menus
2. Double click on BM_drnl.7 to disable the nonlinear computations. A '@' character should appear beside the process name. It may be necessary to single-click to open the window and then double-click to disable the process.
3. Double-click on BM_gammaT.6, to enable the gammatone filter. The '@' character should disappear.
4. Make sure that only one of the two filters is enabled and then click on 'GO'.

Differences should be small at low signal levels but greater at high signal levels.

Tip: A problem with this method is that the display is set to autoscale. After running the program check the inset scale at the bottom left of the display. You may need to maximize the display to read this well.

8 Auditory nerve

This model can be found in the folder *Tutorials\AuditoryPeriphery*. The original version of this folder can be found in *DSAM\AMS\Tutorials\AuditoryPeriphery*. If you have not already done so, copy this folder into your regular workspace. Please do not work in the *AMS* workspace.

This section extends the model to include a simulation of a stream of spikes in a single auditory nerve fiber.

8.1 Single auditory nerve fiber

8.1.1 Simulation file

```
# auditoryNerve.sim

begin {

# generate stimulus
            stim_pureTone_2   < stimTone.par

# Ramps
            Trans_Gate        < rampUp.par
            Trans_Gate        < rampDown.par
            Display_Signal    < displaySignal.par

# Outer-/middler-ear filter model.
            Filt_MultiBPass   < filtMultiBandpassGP.par

#convert to stapes velocity
            Util_mathOp       < mathOpStapes.par

# Basilar membrane filter model.
            BM_DRNL           < BM_drnlGP1.par
            Display_Signal    < displayBM.par

# IHC receptor potential
            IHCRP_Shamma3StateVelIn < IHCRP_VelIn_GP.par
            Display_Signal    < displayRP.par

#IHC synaptic response
            IHC_Meddis2000    < IHChsr2000spike.par

# apply a refractory period
            AN_SG_Meddis02    < ANrefractory.par
            Display_Signal    < displaySynapse.par

            DataFile_Out

}
```

The first part of the *auditoryNerve.sim* file is the same as that for the DRNL filterbank. Only one change has been made; the DRNL filter has been changed to show only one auditory nerve fiber. This makes the demonstration more like a real physiological experiment.

The new processes in the .sim file are concerned with the response of the

- Inner hair cell (IHC)
- IHC/auditory nerve (AN) synapse
- Spike activity of the auditory nerve.

The output should look like Figure 4.

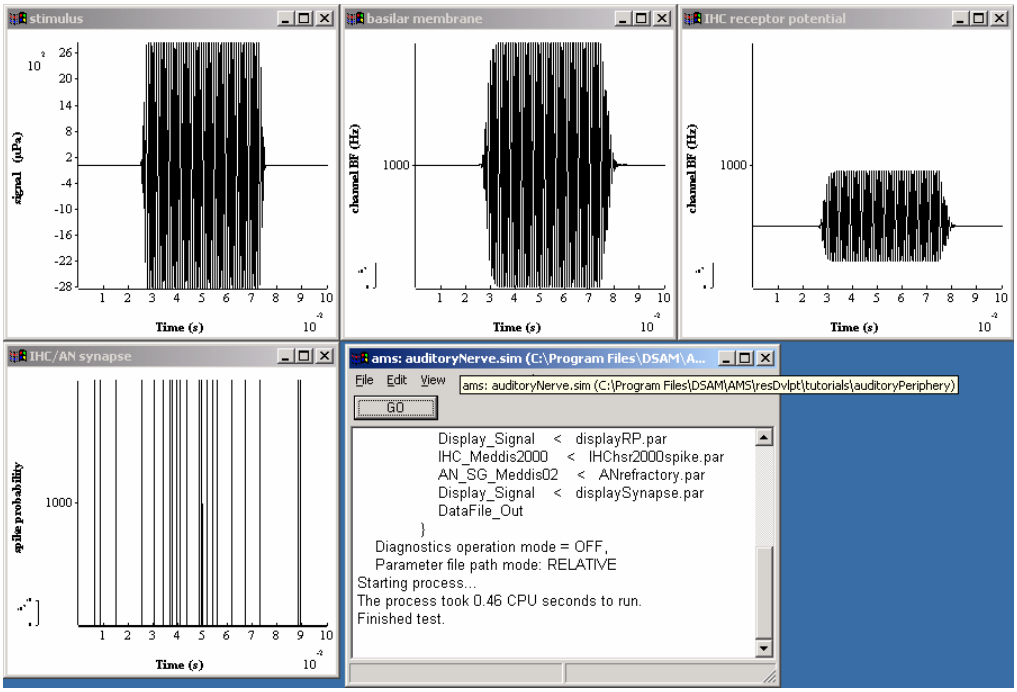


Figure 4 Auditory nerve model

8.1.2 Processes and parameters

8.1.2.1 Filterbank

```
#            BM_DRNL            < BMdrnlGP1.par
```

(.....as in previous section)

```
CF_MODE                  SINGLE
SINGLE_CF                1000
```

This .par file is the same as BMdrnlGP.par with the number of channels set to single and its CF specified to be 1000 Hz. Any number of channels could have been used but a single channel is more appropriate for the examples that follow.

8.1.2.2 Receptor potential

```
#            IHCRP_Shamma3StateVelIn < IHCRP_VelIn_GP.par

# Sumner, C,  Lopez-Poveda, E.A.,  O'Mard, L.P. and Ray Meddis,
# "A revised model of the inner-hair cell and auditory nerve complex"

E_T          0.1          Endocochlear potential, Et (V).
E_K          -70.45E-3    Reversal potential, Ek (V).
G0           1.97e-09     Resting conductance, G0 (S).
G_K          1.8e-08      Potassium conductance, Gk (S = Siemens).
RP_CORRECTION      0.04   Reversal potential correction, Rp/(Rt+Rp).
G_MAXC       8e-09        Maximum mechanical conductance, Gmax (S).
```

```
S0          85e-9       Sensitivity constant, S0 (/m).
U0          7e-9        Offset constant, U0 (m).
S1          5e-7        Sensitivity constant, S1 (/m).
U1          7e-9        Offset constant, U1 (m).
C_TOTAL     6e-12       Total capacitance, C = Ca + Cb (F).
T_C         2.13e-3     Cilia/BM time constant (s).
GAIN_C      16          Cilia/BM coupling gain, C (dB).
```

This module computes the IHC receptor potential on the basis of BM velocity.

These parameters are based on Sumner et al. (2002a). This parameter file can be omitted for casual work. The default values will work well for most purposes. Nevertheless, publishable work should use an explicit parameter file that can be quickly checked against other published sources.

Tip: Most of these parameters should not need changing. However, GAIN_C may be usefully changed in some circumstances. It represents the coupling gain between the basilar membrane and the IHC. Adjusting the gain can be used to adjust the threshold of the subsequent auditory nerve fiber.

8.1.2.3 Display receptor potential

```
#           Display_Signal    < displayRP.par

WIN_TITLE   "IHC receptor potential"

WIN_HEIGHT  300
WIN_WIDTH   300
WIN_X_POS   600
WIN_Y_POS   0

Y_AXIS_MODE LINEAR
Y_AXIS_TITLE      "receptor potential (mV"
AUTO_Y_SCALE      OFF
MINY        -60E-3
MAXY        -20E-3
```

The AUTO_Y_SCALE is set to OFF because we know that the receptor potential only exists within a limited voltage range. Accordingly, MINY and MAXY have been set to –60 mV and –20 mV respectively.

8.1.2.4 IHC synapse

```
#           IHC_Meddis2000    < IHChsr2000spike.par

# Sumner, C,  Lopez-Poveda, E.A.,  O'Mard, L.P. and Ray Meddis,
# "A revised model of the inner-hair cell and auditory nerve complex
# Table II


OP_MODE     SPIKE Output mode: 'spike' or probability, 'prob'
DIAG_MODE   OFF   Diagnostic mode:('off', 'screen' or <file name>).
RAN_SEED    0     Random number seed (0 for different each run).
```

```
PERM_Z      2e+32 Transmitter release permeability, Z (unitless gain)
REV_POT_ECA 0.066 Calcium reversal potential, E_Ca (Volts).
BETA_CA     400   Calcium Boltzmann function parameter, beta.
GAMMA_CA    130   Calcium Boltzmann function parameter, gamma.
TAU_M       1e-4  Calcium current time constant (s).
TAU_CA      1e-4  Calcium diffusion (accumulation) time constant (s).

REPLENISH_Y 10    Replenishment rate (units per second).
LOSS_L      2580  Loss rate (units per second).
REPROCESS_X 66.3  Reprocessing rate (units per second).
RECOVERY_R  6580  Recovery rate (units per second).

POWER_CA    3     Calcium transmitter release exponent (power).

# These parameters used to set fiber type
GMAX_CA           8E-9  Maximum calcium conductance (Siemens).
CONC_THRESH_CA    4.48e-11    Calcium threshold Concentration.
MAX_FREE_POOL_M   10    Max. no. of transmitter packets in free pool
(integer).
```

A recommended .par file is given even though the default values will give a reasonable result. When publishing results using a model of this kind it is important to specify these parameters. The values given here have been taken from Sumner et al (2002a) table II

OP_MODE is an important parameter. When set to 'SPIKE' the output signifies a stochastic transmitter release event. The model assumes that a single release event is enough to generate a post-synaptic spike in the auditory nerve fiber if it is not already in a refractory state. The output is, therefore, binary. A release event lasts for only one sample period.

Tip: An alternative is to set the OP_MODE parameter to 'PROB'. In this case the output is a continuous function between 0 and 1 representing the *probability* of observing the onset of a spike during that sample period. This gives a smoother representation of the IHC response.

8.1.2.5 Auditory nerve refractory effects

```
#           AN_SG_Meddis02    < ANrefractory.par

RAN_SEED          0
PULSE_DURATION    -1
MAGNITUDE         1
REFRAC_PERIOD     0.75E-3
RECOVERY_TAU      .6E-3
```

This process computes the refractory state of one or more nerve fibres. A release event triggers an action potential in the AN fiber if it is not in a refractory state. The process introduces an absolute refractory state for a period of 0.75 msec (REFRAC_PERIOD) following an action potential. It then enters a relative refractory state in which the probability of a successful conversion of a release event into an

action potential is governed by an exponential recovery process. The time constant of this process is defined by RECOVERY_TAU.

The output from this process is a stream of pulses whose height and width are defined by PULSE_MAGNITUDE and PULSE_DURATION respectively. If PULSE_DURATION is set to –1, the width of the pulse will be set automatically to DT.

If RAN_SEED is set to a negative integer, the random number process used to make the stochastic decisions, will be the same on each run. If it is set to zero, it will be different for each run.

Tip: This process requires an input consisting of a random binary process (transmitter release events). The IHC synapse process parameter OP_MODE must therefore be set to deliver SPIKE events. Probabilities are not appropriate as input.

8.2 Generating a post stimulus time histogram (PSTH)

A PSTH can be computed using the REPEAT function. All processes within the REPEAT curly brackets are repeated a number of times (specified after the REPEAT command. The number of repeats must be set in the .sim file. It cannot be changed at run time.

Processes up to the generation of the receptor potential are deterministic. Repeating these processes would not serve any purpose, as they would give the same result each time. The REPEAT block, therefore, begins with the IHC synapse, the first stochastic process.

8.2.1 Simulation file

This .sim file produces a PSTH based on 200 presentations of a pure tone stimulus to a single auditory nerve fiber. It is the same as the auditoryNerve.sim file except for two new processes at the end.

```
# ANpsth.sim

begin {

# generate stimulus
            stim_pureTone_2   < stimTone.par

# Ramps
            Trans_Gate        < rampUp.par
            Trans_Gate        < rampDown.par
            Display_Signal    < displaySignal.par

# Outer-/middler-ear filter model.
            Filt_MultiBPass   < filtMultiBandpassGP.par

#convert to stapes velocity
            Util_mathOp       < mathOPstapes.par
```

```
# Basilar membrane filter model.
            BM_DRNL           < BM_drnlGP1.par
            Display_Signal    < displayBM.par

# IHC receptor potential
            IHCRP_Shamma3StateVelIn < IHCRP_VelIn_GP.par
            Display_Signal    < displayRP.par

repeat 200  {

#IHC synaptic response
            IHC_Meddis2000    < IHChsr2000spike.par

# apply a refractory period
            AN_SG_Meddis02    < ANrefractory.par
            Display_Signal    < displaySynapse.par

            Ana_histogram     < histogramPSTH.par
            Display_Signal    < displayPSTH.par

            }

            DataFile_Out
}
```

When run, the output should look like Figure 5.

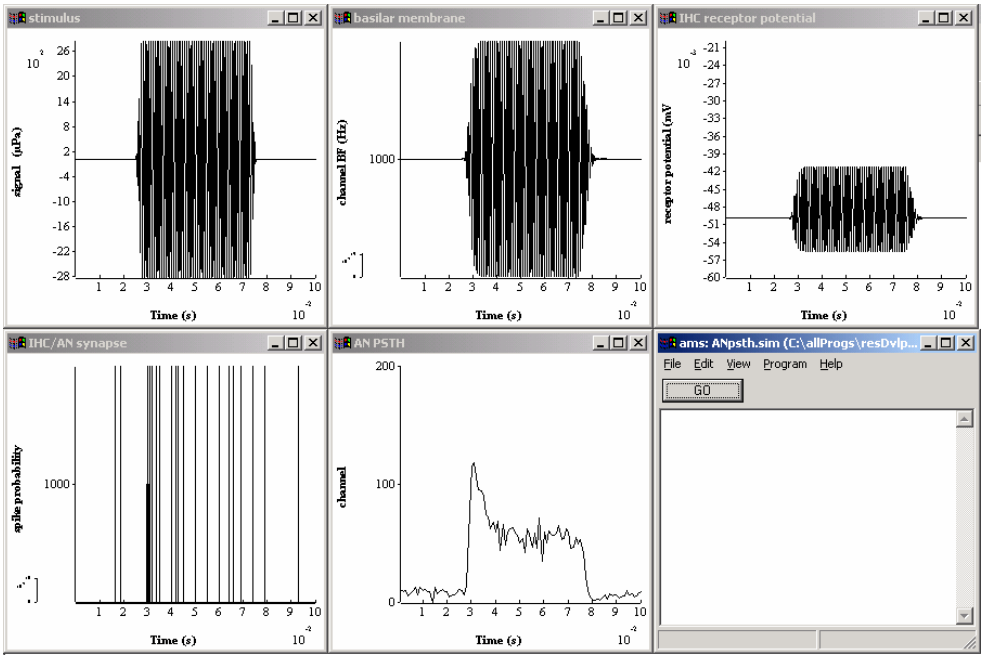


Figure 5 PSTH computed using the auditory nerve model.

8.3 Create a PSTH

The histogram process can be used to bin and count events. It is an accumulating process. Every time it is activated it adds the new results to the old results.

```
#           Ana_histogram     < histogram1.par
```

```
DETECT_MODE     SPIKE
BIN_WIDTH       1E-3
THRESHOLD       0.5
```

Ana_Histogram accumulates results across all the repeat trials.

DETECT_MODE when set to SPIKE an event occurs whenever the input function rises above the THRESHOLD value. The time of the event is the time of the positive-going zero-crossing.

BIN_WIDTH defines the width of the bin. If not set, the default is DT and this is normally too small for most purposes.

THRESHOLD The input from the AN_SG_Meddis02 process is in the form of narrow pulses with a height of 1 unit. By setting the THRESHOLD to 0.5, all events will be detected as they pass form zero to one.

8.3.1 Display the PSTH

```
#           Display_Signal    < displayPSTH.par

WIN_TITLE   "AN PSTH"

WIN_HEIGHT  300
WIN_WIDTH   300
WIN_X_POS   300
WIN_Y_POS   300

Y_AXIS_MODE LINEAR
Y_AXIS_TITLE      "spike count"
AUTO_Y_SCALE      off
MINY        0
MAXY        200
Y_NUMBER_FORMAT   Ye0
Y_TICKS     3
```

The display has been created using a fixed y-axis with a maximum of 200 (spikes). This will guarantee that the PSTH will grow (before your very eyes) with each REPEAT loop.

Y_NUMBER_FORMAT defines the format of the numbers in the y-axis. Ye0 keeps the numbers simple.

Y_TICKS specifies the number of numbers along the y-scale.

8.3.2 Things to do

Try different signal levels and notice how the PSTH changes.

- Load and run the program
- Edit\simulation parameters

- select stim_pureTone_2 and open parameter window
- change INTENSITY
- GO

8.4 Generate a Period histogram

ANpsth.sim can be adapted to produce a period histogram (rather than a PSTH) by changing the histogram parameter file

```
 # ANperiodHistogram.sim

begin {

# generate stimulus
            stim_pureTone_2   < stimTone.par

# Ramps
            Trans_Gate        < rampUp.par
            Trans_Gate        < rampDown.par
            Display_Signal    < displaySignal.par

# Outer-/middler-ear filter model.
            Filt_MultiBPass   < filtMultiBandpassGP.par

#convert to stapes velocity
            Util_mathOp       < mathOPstapes.par

# Basilar membrane filter model.
            BM_DRNL           < BM_drnlGP1.par
            Display_Signal    < displayBM.par

# IHC receptor potential
            IHCRP_Shamma3StateVelIn < IHCRP_VelIn_GP.par
            Display_Signal    < displayRP.par

repeat 200  {

#IHC synaptic response
            IHC_Meddis2000    < IHChsr2000spike.par

# apply a refractory period
            AN_SG_Meddis02    < ANrefractory.par
            Display_Signal    < displayFiberResponse.par

            Ana_histogram     < histogramPH.par
            Display_Signal    < displayPH.par

            }
            DataFile_Out

}
```

The results should look like this

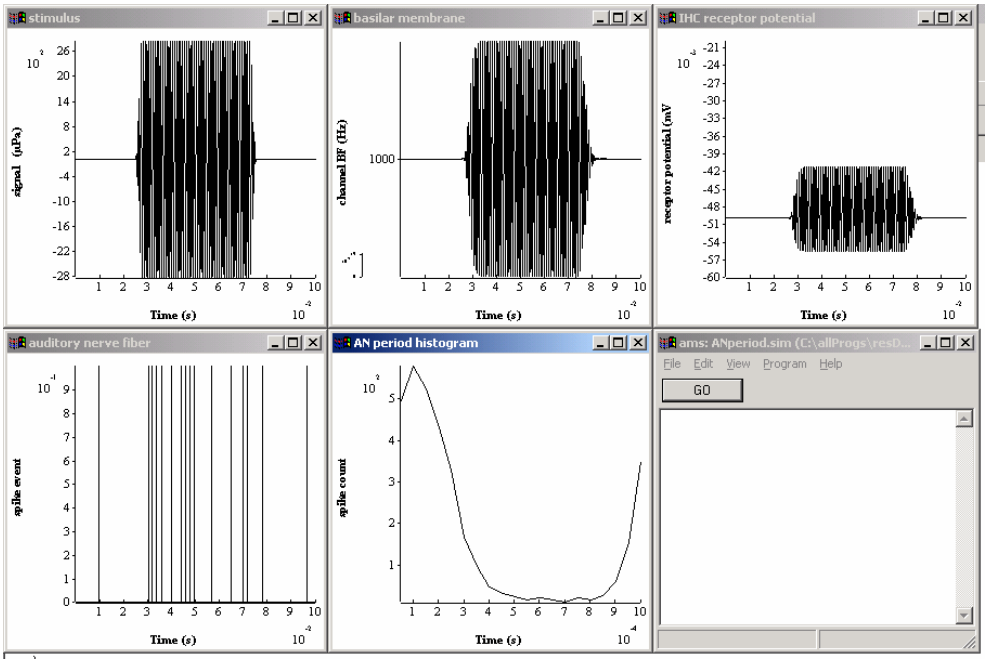


Figure 6 Periodhsitogram based on a single auditory nerve fiber, 200 repeats

8.4.1 period histogram

```
#           Ana_histogram     < histogramPH.par

BIN_WIDTH         50E-6       # 20 bins for 1 kHz period

DETECT_MODE       SPIKE
THRESHOLD         0.5

TYPE_MODE         PH
PERIOD            0.001
OFFSET            0.0025
OFFSET            0.0025
```

TYPE_MODE is defined as PH for period histogram (use PSTH for post stimulus time histogram)

PERIOD specifies the period of the signal. The stimulus is a 1-kHz tone and it period is 0.001s.

BIN_WIDTH is now 50 microseconds. This will create 20 bins in a single 1 msec cycle.

OFFSET specifies the duration of the initial section of the stimulus that is to be ignored (the silent period).

Tip: The binwidth must be carefully calculated to give an exact number of bins

8.4.2 Display period histogram

```
#           Display_Signal    < displayPH.par

WIN_TITLE   "AN period histogram"
```

```
WIN_HEIGHT  300
WIN_WIDTH   300
WIN_X_POS   300
WIN_Y_POS   300

Y_AXIS_MODE LINEAR
Y_AXIS_TITLE      "spike count"
AUTO_Y_SCALE      on
```

The AUTO_SCALE has been switched on for this example.

8.4.3 Things to do

Try changing the frequency of the stimulus. Note that you will need to change the period and the binwidth in the histogram module.

9 The 'Loudness' model

This model can be found in the folder *Tutorials\Loudness*. The original version of this folder can be found in *DSAM\AMS\Tutorials\Loudness*. If you have not already done so, copy this folder into your regular workspace. Please do not work in the *AMS* workspace.

9.1 General introduction

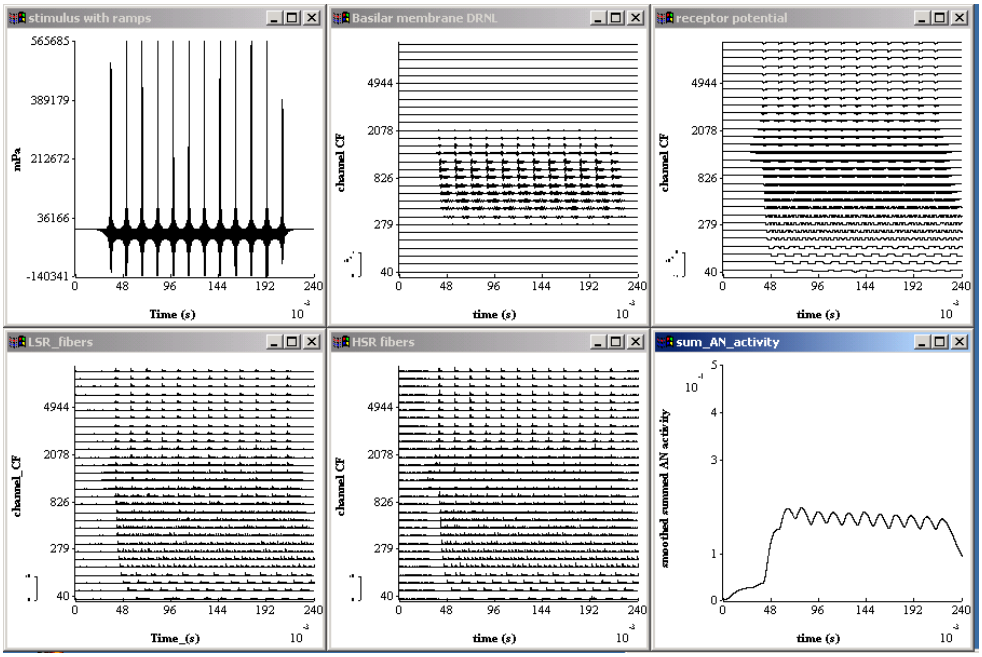
The model is based on the idea that the loudness of a stimulus might be related to the sum total of auditory nerve activity in response to that stimulus. The model consists mainly of a sequence of computations representing different stages in physiological signal processing between the acoustic stimulus and the auditory nerve (AN) response. Two types of fibers are represented (LSR, HSR) [2]. The model finishes by summing all AN spike activity (across channels) and applying a 'temporal window function'. Finally, the model creates an output file, 'output.dat' that contains the output from the temporal window as a function of time. The research question is whether or not the output from the model can predict the loudness property of stimuli.

For an approachable overview of the topic of loudness I recommend Plack and Carlyon (1995). Their coverage suggests lots of experiments that can be tried with the model.

To run the model, launch 'AMS'. Consult section 2.5 on help with getting started. The output of the model should look something like this. The temporal sequence of screens goes from left to right from top to bottom.

The use of the model here is not an endorsement of the theory but an opportunity to test the theory,

[2] While three types are commonly quoted, some research suggest that some animals have only two distinct types; Relkin,E.M. and Doucet,J.R. (1991). Two makes for an easier model, but the .sim file could be easily extended to include three if required.



9.2 The loudness simulation file

The model is specified by a .sim file (loudness.sim) that should be present in your *tutorials\loudness* folder:

Simulation files are discussed in general in section 3.1 above.

```
#loudness.sim

diag_mode on            Diagnostics mode ("off" or "on").

begin {

# stimulus generation
            Stim_Harmonic           < stimHarmonic.par

# apply ramps to signal
            Trans_Gate              < rampUp.par
            Trans_Gate              < rampDown.par
            Util_PadSignal          < padSignal.par
            Display_Signal          < displayStimulus.par

# outer/middle ear simulation
            Filt_MultiBPass         < filtIIRMoore.par

# convert to stapes velocity
            Util_mathOp             < mathOPstapes.par

# nonlinear DRNL filterbank
            BM_DRNL                 < BM_DRNLHuman.par
            Display_Signal          < displayBM.par

# receptor potential
z%          IHCRP_Shamma3StateVelIn (->a,b,d)   < IHCRP_VelIn_GP.par
a%          Display_Signal    (z->) < displayRP.par
```

```
# IHC synaptic response and spike generation in three fiber types
# LSR
b%          IHC_Meddis2000    (z->) < IHC_Meddis2000LSR.par
            An_SG_Binomial          < binomialLSR.par
b1%         Display_Signal    (->x) < displayLSR.par


# HSR
d%          IHC_Meddis2000    (z->) < IHC_Meddis2000HSR.par
            An_SG_Binomial          < binomialHSR.par
d1%         Display_Signal    (->x) < displayHSR.par

# combine LSR, MSR and HSR fiber outputs
x%          Util_Accumulate (b1, d1->)

# add across channels
            Util_ReduceChannels           < reduceChannels.par

# apply temporal window
            Filt_MultiBPass         < filtMBtemporalWindow.par
            Display_Signal          < displaySumAN.par

# write results to a data file
            DataFile_out            < dataFileOut.par
            }
```

9.3 Processes and parameter files

9.3.1 Stimulus generation

```
#           Stim_Harmonic           < stimHarmonic.par

FUND_FREQ         64    Fundamental frequency (Hz).
INTENSITY         60    Intensity (dB SPL).
DURATION          0.2   Duration (s).
DT                2e-05 Sampling interval, dt (s).

LOW_HARMONIC      1     Lowest harmonic number.
HIGH_HARMONIC     20    Highest harmonic number.

PHASE_MODE        COSINE
```

This process generates a harmonic stimulus. The .par file can be used to define

- the fundamental frequency
- the first and last harmonic number
- the intensity and duration of the signal.
- the phase relationships among the harmonics

The process can be used to create even more complex harmonic stimuli. See the manual.

Signal level is specified as dB SPL per component. The signal is represented as instantaneous pressure in micro Pascals.

Sample rate. DT, the stimulus sampling interval, is a very important parameter. This not only defines the sample rate for generating the stimulus but also defines the

sample rate for all the model computations. This value must be set with care. A useful value is 2e-5s (equivalent to 50-kHz sample rate). For definitive computations, DT should be even shorter.

Large values of DT (lower sampling rates) may give rise to false results. AMS may even complain if the value is so large as to be predictably problematic. For example some IIR filters become unstable if the sample rate is too low. On the other hand, very small values of DT (e.g. 1e-6s) may result in long computation times with very little improvement in the quality of the result. If two values of dt give almost identical results, use the larger value and enjoy shorter computation times.

9.3.2 Stimulus ramps

```
            Trans_Gate              < rampUp.par
            Trans_Gate              < rampDown.par
```

See 6.2.2

9.3.3 PadSignal

This process adds silence to the beginning and end of the stimulus. It is important to do this for the 'Loudness' model because it integrates information over time. This integration needs a period of silence before and after the end of the stimulus proper to allow the integrator to stabilise.

```
#           Util_PadSignal          < padSignal.par

BEGIN_DURATION    0.02  initial silence to be added
END_DURATION      0.02  final silence to be added
```

9.3.4 Display signal

```
            Display_Signal          < displayStimulus.par

            Display_Signal          < displayStimulus.par
```

See section 6.2.3.

9.3.5 Human outer/middle ear

The outer middle ear (pre-emphasis) filter is a set of four parallel IIR bandpass filters. The overall effect of the outer/middle ear was calculated by combining the outer ear transfer function as published in Moore et al. (1997) Fig. 2 with the middle ear function published as figure 3 of the same paper. The overall filter was published in Glasberg and Moore (2002).

```
#           Filt_MultiBPass   < filtIIRMoore.par

NUM_FILTERS       4

CASCADE     0:2
ATTENUATION 0:-12
LOWER_FREQ  0:100
UPPER_FREQ  0:1300
```

```
CASCADE     1:2
ATTENUATION 1:1.5
LOWER_FREQ  1:350
UPPER_FREQ  1:6500

CASCADE     2:2
ATTENUATION 2:5
LOWER_FREQ  3:1800
UPPER_FREQ  3:5200

CASCADE     3:2
ATTENUATION 3:-11
LOWER_FREQ  3:7500
UPPER_FREQ  3:14000
```

The parameter NUM_FILTERS specifies the number of filters. Each filter must be fully described in terms of its parameters. The notation *i:param* specifies the parameter for the i^{th} filter. Filter numbers are sequenced 0,1.. N-1.

Each filter is composed of a cascade of first-order filters where the 3-dB down points of the first-order filters are specified by the parameters LOW_FREQ and UPPER_FREQ. The parameter CASCADE specifies the number of cascaded filters.

A cascade of 3 filters, for example, is similar to but not the same thing as a third order filter. A filter consisting of a cascade of three first-order filters has high and low frequency skirts with the same slope as a third order filter but the upper and lower cut-offs will be narrower than those specified in the parameters. In this case the LOW_FREQ and UPPER_FREQ parameters specify the 9-dB down points of the filter.

ATTENUATION is the reduction in level between the input and output at BF. It is specified in dB.

The resulting filter is shown in Figure 7.

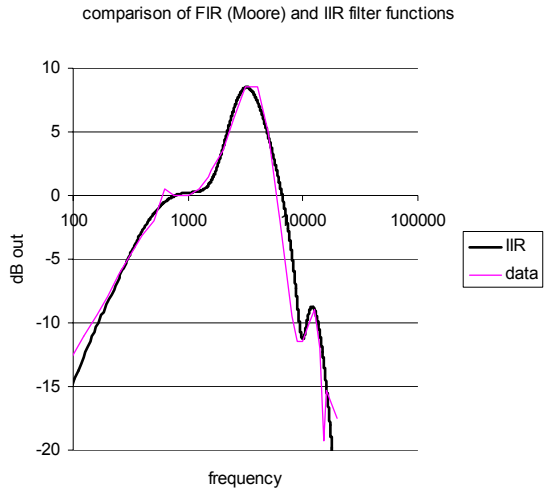


Figure 7 Comparison of multiBandPass approximation with original human data.

9.3.6 stapes motion

The next stage requires that we simulate the conversion of the input signal from sound pressure (μPa) to stapes motion (m/s). This is a linear conversion requiring nothing more than the application of a scalar.

```
#           Util_mathOp       < mathOpStapes.par
# converts microPascals to stapes velocity (m/s)

OPERATOR    SCALE
OPERAND     3E-10
```

The scalar we have used is 3e-10. This was chosen to give an output of 1e-8 m/s for a 1 kHz signal at 0 dB SPL (Lopez-Poveda and Meddis, 2001).

9.3.7 BM_DRNL < BM_DRNLHuman.par

This process simulates the nonlinear response of the basilar membrane at a number of points along the length of the cochlea. It accepts stapes motion as input and returns basilar membrane (BM) velocity at 40 different points with characteristic frequency (CF) varying between 40 Hz and 10 kHz and spaced according to Greenwood's function for the distribution of CFs along the BM.

The rationale of the DRNL filter is explained elsewhere, Meddis et al. (2001). The particular coefficients used for the filters are taken from Lopez-Poveda and Meddis (2001). These have been optimized for human listeners. Expect these filter coefficients to change as new data is obtained and better fits are established.

This is a filterbank with many filters. The filter parameters for any given filter depend on the CF of that filter. They are computed in DSAM using a log regression equations fit to psycho-acoustic data. The regression equations look like this

log(parameter)= a + b log(CF)

where *a* and b need to be specified for each parameter. Unless you are specifically interested in the nuances of DRNL filtering, we advise that these parameters are not changed. These parameters are liable to change in the light of new data and new methods of estimating the parameters. If you want the latest set of parameters, contact Ray Meddis.

However, users may want to specify the number of filters and the range of their CFs. This can be done at the bottom of the file.

```
#           BM_DRNL           < BMdrnlGP.par

# Parameters are based on
# Lopez-Poveda, E.A.,   and Ray Meddis.
#"A human non-linear cochlear filterbank"
# JASA 110, 3107-3118, (2001)

#nonlinear path
NL_GT_CASCADE           3     Nonlinear gammatone filter cascade.
```

```
NL_LP_CASCADE           3     Nonlinear low-pass filter cascade.

NONLINBWIDTH_PARAMETER  0:-0.032
NONLINBWIDTH_PARAMETER  1:0.774

COMPRSCALEA_PARAMETER   0:1.4
COMPRSCALEA_PARAMETER   1:0.82

COMPRSCALEB_PARAMETER   0:1.62
COMPRSCALEB_PARAMETER   1:-0.82

COMP_N_EXPON            0.25  Compression exponent, n (units).

L_GT_CASCADE            3     Linear gammatone filter cascade.
L_LP_CASCADE            4     Linear low-pass filter cascade.


#linear path
LINCF_PARAMETER   0:-0.068
LINCF_PARAMETER   1:1.02

LINBWIDTH_PARAMETER     0:0.037
LINBWIDTH_PARAMETER     1:0.79

LINSCALEG_PARAMETER     0:4.2
LINSCALEG_PARAMETER     1:-0.48

#BF List Parameters:-
CF_MODE                 HUMAN
MIN_CF                  40    Minimum centre frequency (Hz).
MAX_CF                  10000 Maximum centre frequency (Hz).
CHANNELS                10    No. of centre frequencies.
```

9.3.8 Display BM

see section 6.2.7.

9.3.9 Receptor potential

See section 8.1.2.2 for a general description of this process.

In this example the output of the receptor potential process is used three times; as an input to the LSR and HSR fiber computations and also to the display process. To signal this, the three destinations are indicated in the parentheses.

```
z%    IHCRP_Shamma3StateVelIn (->a,b,d)   < IHCRP_VelIn_GP.par
```

9.3.10 Display Receptor potential

See section 8.1.2.3.

9.3.11 IHC synapse; LSR fibers

See section 8.1.2.4 for a general description of this process.

In this example the input comes from the receptor potential process labelled z%.

```
b%          IHC_Meddis2000    (z->) < IHC_Meddis2000LSR.par
```

This process generates a stream of probabilities that reflect the likelihood that an action potential will occur in an auditory nerve fiber. The parameters that define this as a low spontaneous rate synapse can be found at the end of the .par file.

```
#           IHC_Meddis2000    < IHC_Meddis2000HSR.par

# Sumner, et al (2002); Table II


OP_MODE     PROB  Output mode: 'spike' or probability, 'prob'
DIAG_MODE   OFF   Diagnostic mode:('off', 'screen' or <file name>).
RAN_SEED    0     Random number seed (0 for different each run).

PERM_Z      2e+32 Transmitter release permeability, Z (unitless gain)
REV_POT_ECA 0.066 Calcium reversal potential, E_Ca (Volts).
BETA_CA     400   Calcium Boltzmann function parameter, beta.
GAMMA_CA    130   Calcium Boltzmann function parameter, gamma.
TAU_M       1e-4  Calcium current time constant (s).
TAU_CA      1e-4  Calcium diffusion (accumulation) time constant (s).

REPLENISH_Y 10    Replenishment rate (units per second).
LOSS_L      2580  Loss rate (units per second).
REPROCESS_X 66.3  Reprocessing rate (units per second).
RECOVERY_R  6580  Recovery rate (units per second).

POWER_CA    3     Calcium transmitter release exponent (power).

# These parameters used to set fiber type
# based on HSR (1) in Sumner table II
GMAX_CA           8E-9  Maximum calcium conductance (Siemens).
CONC_THRESH_CA    4.48e-11    Calcium threshold Concentration.
MAX_FREE_POOL_M   10    Max. no. of transmitter packets in free pool
```

9.3.12 Auditory nerve spike generation; LSR fibers

```
#           An_SG_Binomial          < binomialLSR.par

NUM_FIBRES          36    Number of fibres.
RAN_SEED            0     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       0.00075     Refractory period (s).
```

This process takes spike probabilities from the synapse process and converts them to spike activity in a specified number of fibers. Spikes take the form of pulses whose height and width can be specified as parameters. For this particular example, the height and duration are arbitrary, so long as both HSR and LSR pulses are the same. 36 LSR fibers are used compared to 64 HSR fibers. This preserves the ratio of HSR/LSR fibers found in (Relkin, Hear. Res., 1991, 215-222). The output is the sum of all the pulses.

The method for computing the spike events is based on a rapid approximation using the binomial distribution (see Pearson). Refractory effects are simulated (somewhat imprecisely) using a formula to be found in Meddis?.

9.3.13 Display spike activity

There are two displays displayHSR.par and displayLSR.par.. Autoscale is switched off and all three displays are set to have the same y-scale so that comparisons can be made between displays.

```
#           Display_Signal    < displayLSR.par

WIN_TITLE         "LSR_fibers"

WIN_HEIGHT        300
WIN_WIDTH         300
WIN_X_POS         0
WIN_Y_POS         300

Y_AXIS_TITLE      "channel BF"
Y_AXIS_MODE       CHANNEL
AUTO_Y_SCALE      OFF
MAXY              10
MINY              0

X_AXIS_TITLE      Time_(s)
```

Note that the output from the display process is directed toward the accumulate process below labeled x%.

```
b1%         Display_Signal    (->x) < displayLSR.par
```

9.3.14 HSR fibers

```
# HSR
d%          IHC_Meddis2000    (z->) < IHC_Meddis2000HSR.par
            An_SG_Binomial          < binomialLSR.par
d1%         Display_Signal    (->x) < displayHSR.par
```

see LSR fibers above

9.3.15 Util_Accumulate

The two sets of fibers are now combined using this process. Util_Accumulate simply sums both of the input signals; b1 and d1. The output of this process is a single multi-channel representation.

```
x%        Util_Accumulate (b1, d1->)

This process needs no parameter file.
```

9.3.16 Util_ReduceChannels < reduceChannels.par

This process adds together the activity in all of the channels, which is across BF channels.

It can be configured to combine channels together in different ways such as across two or three equal sets but on this occasion only one grand total is required.

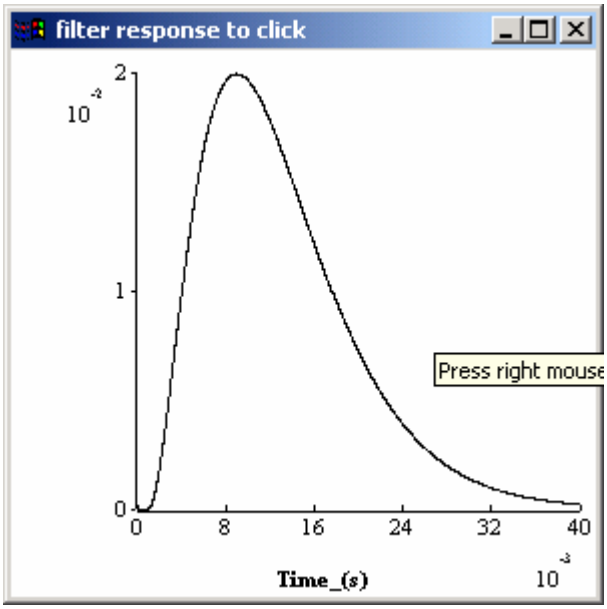
We have a choice of whether to sum the channels or average them. Summing the channels is more in the spirit of the loudness model. However, averaging has the advantage of keeping the 'loudness' estimate roughly invariant when the number of channels is changed. This means that we can turn the auto-scale off in the subsequent display.

```
#           Util_ReduceChannels           < reduceChannels.par

MODE  AVERAGE     Mode - 'average' or simple 'sum'.
NUM_CHANNELS      1 (resulting number of channels)
```

9.3.17 Filt_MultiBPass < filtMBtemporalWindow.par

The temporal window is implemented here using a third-order bandpass filter with a very zero Hz lower cut-off frequency and a 40 Hz upper cutoff frequency. The parameters have been chosen to make the filter similar in operation to the temporal widow specified in Plack et al. (?) Here is the impulse response of the filter.




```
#           Filt_MultiBPass         < filtMBtemporalWindow.par

NUM_FILTERS 1     No. of parallel band pass filters.
CASCADE     0:3
ATTENUATION 0:0
LOWER_FREQ  0:0
UPPER_FREQ  0:40
```

9.3.18 Display_Signal < displaySumAN.par

The autoscale has been set to off. This makes it easier to see changes in "loudness" when the stimulus is changed.

'linear' has been chosen for the y-axis mode because we want to know the actual level of the output rather than the channel number.

```
#           Display_Signal          < displaySumAN.par

WIN_TITLE   "sum AN activity: loudness?"

WIN_HEIGHT  300
WIN_WIDTH   300
WIN_X_POS   600
WIN_Y_POS   300

Y_AXIS_TITLE      "smoothed summed AN activity"
Y_AXIS_MODE       LINEAR
AUTO_Y_SCALE      OFF
MAXY              0.5
MINY              0

X_AXIS_TITLE      "time (s)"
```

9.3.19 DataFile_out < dataFileOut.par

The summed auditory nerve activity shown in the display is written to an ASCII .dat file. The first value in each row is the time.

```
Time (s)  0
0.00000e+000  0
2.00000e-005  0
4.00000e-005  0
6.00000e-005  0
8.00000e-005  0

   --        --
   --        --

2.80000e-004  0
3.00000e-004  0
3.20000e-004  0
3.40000e-004  0.0859397
3.60000e-004  0.171963
3.80000e-004  0.42995
4.00000e-004  0.602249
4.20000e-004  0.774716
4.40000e-004  0.775472
4.60000e-004  0.77623

The file name is specified as a '.dat' file. this guarantees that the output
will be in printable form.


#           DataFile_out            < dataFileOut.par

FILENAME    output.dat
```

9.4 Things to do

9.4.1 Stimulus intensity

Increase the intensity of the sound and monitor the effect on the loudness estimate in the final display. Does it agree with your intuition?

9.4.2 Effect of phase on loudness

The stimulus is a harmonic complex with each component in cosine phase. Does the phase affect the loudness estimate? The data that inspired the model came from Gockel et al. They found that cosine phase stimuli were louder than random phase stimuli.

10 Autocorrelation

This model can be found in the folder *Tutorials\Autocorrelation.* The original version of this folder can be found in *DSAM\AMS\Tutorials\Autocorrelation.* If you have not already done so, copy this folder into your regular workspace. Please do not work in the *AMS* workspace.

10.1 Model implementation (.sim and .par files)

10.1.1 .sim file

The autocorrelation model is broadly similar to the model presented in the original JASA articles: Meddis and Hewit (1991 a, b), Meddis and O'Mard(1997)

There is, however, one important difference. It has been updated to include the latest nonlinear peripheral model and the latest hair cell model from the Essex stable. There is no reason to persist with the original linear models now that we have useful nonlinear models.

The model has a number of stages

- Stimulus Input; A harmonic complex.
- Auditory periphery: The output from the periphery is a stream of *probabilities* of action potentials in fibers of the AN.
- Autocorrelation. A separate autocorrelation is computed at each stimulus frequency.
- Summary ACFs are computed by adding vertically all the rows in the ACF matrix (i.e. across all BF channels).
- Write to file

```
#ACF.sim

begin {

#  Stimulus generation
stim%       Stim_Harmonic           < stimHarmonic.par
```

```
#apply ramps to signal
            Trans_Gate              < rampUp.par
            Trans_Gate              < rampDown.par
            Display_Signal          < displayStimulus.par

#outer/middle ear simulation
            Filt_MultiBPass         < filtMultiBandpass.par

#convert to stapes velocity
            filt_LowPass            < filtScaleToStapes.par

#nonlinear DRNL filterbank
            BM_DRNL                 < BM_DRNLHuman.par
            Display_Signal          < displayBM.par

#receptor potential
            Filt_MultiBPass         < filtStereocilia.par
            IHCRP_Shamma            < IHCreceptorPotential.par
            Display_Signal          < displayReceptorPotential.par

# HSR
            IHC_Meddis2000          < IHC_Meddis2000HSR.par
            Display_Signal          < displayHSR.par

            Ana_Acf                 < acf.par
            Display_Signal          < displayACF.par

            Util_ReduceChannels     <reduceChannels.par


#write results to a data file
            DataFile_out            < dataFileOut.par
            }
```

10.1.2 Example

When the model is run, the screen should look something like this.

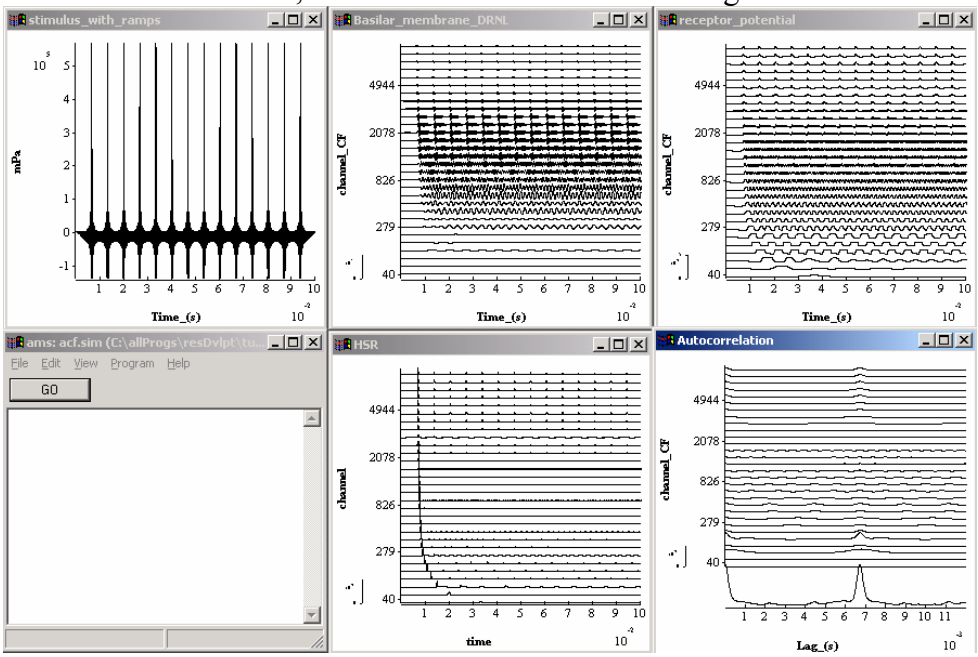


Figure 8 Autocorrelation model

10.1.3 Generate harmonic stimulus

```
#           Stim_Harmonic           < stimHarmonic.par

LOW_HARMONIC      1     Lowest harmonic number.
HIGH_HARMONIC     20    Highest harmonic number.
PHASE_MODE        COSINE

FUND_FREQ         150   Fundamental frequency (Hz).
INTENSITY         60    Intensity (dB SPL).
DURATION          0.1   Duration (s).
DT                2e-05 Sampling interval, dt (s).
```

Stim_Harmonic is capable of generating a wide range of harmonic stimuli. Here it creates a harmonic complex. The output is in microPascals.

DT is the sampling period

10.1.4 Auditory periphery

This is described fully in sections 9.3.2 to 9.3.10.

```
#apply ramps to signal
            Trans_Gate              < rampUp.par
            Trans_Gate              < rampDown.par
            Display_Signal          < displayStimulus.par

#outer/middle ear simulation
            Filt_MultiBPass         < filtMultiBandpass.par

#convert to stapes velocity
            filt_LowPass            < filtScaleToStapes.par

#nonlinear DRNL filterbank
            BM_DRNL                 < BM_DRNLHuman.par
            Display_Signal          < displayBM.par

#receptor potential
            Filt_MultiBPass         < filtStereocilia.par
            IHCRP_Shamma            < IHCreceptorPotential.par
            Display_Signal          < displayReceptorPotential.par

# HSR
            IHC_Meddis2000          < IHC_Meddis2000HSR.par
            Display_Signal          < displayHSR.par
```

10.1.5 Autocorrelation

This module performs a within-channel running autocorrelation function and displays the final state of the function at the end of the signal.

```
#           Ana_Acf                 < acf.par

TIME_CONST  0.0025      Time constant, tw (s).
MAX_LAG     0.012       Maximum autocorrelation lag, tau (s).
```

TIME_CONST specifies the time constant of the running autocorrelation function. MAX_LAG specifies the longest period to be computed.

10.1.6 Display_Signal < displayACF.par

We want to see the summary autocorrelation coefficient (SACF) as well as the individual ACFs. This can be achieved by setting the subsequent display module parameter 'summary_display' to 'ON'. Note that this is a display facility only. The SACF is not passed to the next module. The output from Ana_Acf is the matrix of autocorrelation function (channel x lag).

It is also important to label the x-axis correctly ('Lag') to avoid confusion.

```
#           Display_Signal          < displayACF.par

Y_AXIS_MODE CHANNEL     Y-axis mode ('channel' (No.) or 'scale').
AUTO_Y_SCALE      ON    Automatic y-axis scale ('on' or 'off').

X_AXIS_TITLE      "Lag (s)"

WIN_TITLE   Autocorrelation   Display window title.
SUMMARYDISPLAY    ON    Summary display mode ('on' or 'off').

WIN_HEIGHT  300   Display frame height (pixel units).
WIN_WIDTH   300   Display frame width (pixel units).
WIN_X_POS   600   Display frame X position (pixel units).
WIN_Y_POS   300   Display frame Y position (pixel units).
```

10.1.7 Computing the SACF

This module performs a summation (or averaging) across channels of the signal. The user specifies how many output signals he wants. If two are selected, the top half of the signals are combined to produce the first output signal and the bottom half used to produce the second output signal.

```
#     Util_ReduceChannels           < reduceChannels.par

MODE              AVERAGE     Mode - 'average' or 'sum'.
NUM_CHANNELS      1           (resulting number of channels)
```

10.1.8 Writing the results to file

The result of the preceding module is written to an output file. The default is 'output.dat'.

The output has the following format

```
Time (s)  0
2.00000e-005  0.000820702
4.00000e-005  0.000817631
6.00000e-005  0.000803369
8.00000e-005  0.000778897
1.00000e-004  0.000745795
1.20000e-004  0.000706055
1.40000e-004  0.000661867
1.60000e-004  0.000615415
```

```
1.80000e-004   0.000568703
     -              -           -
```

11 Binaural pitch extraction

This model can be found in the folder *tutorials\binauralPitch.* The original version of this folder can be found in *DSAM\AMS\tutorials\ binauralPitch.* If you have not already done so, copy this folder into your regular workspace. Please do not work in the *AMS* workspace.

This is a complex example. It is included to show how to handle binaural models. It also illustrates some fancy connections between modules.

When all harmonics of a fundamental frequency are presented to both ears (a diotic pitch stimulus), the percept has a pitch at a frequency close to the spacing of the harmonics. However, a binaural pitch stimulus consisting of a series of consecutive harmonics organised so that the even numbered harmonics are presented to one ear and odd numbered harmonics presented to the other ear (a dichotic pitch stimulus) also produces a pitch percept at the fundamental frequency (Houtsma and Goldstein, 1972). If pitch is processed monaurally, one might argue that the dichotic pitch stimulus should have a pitch at twice the frequency of the monaural stimulus. The fact that it does not has been used to argue that pitch is processed centrally, that is after the combination of left and right ear inputs.

However, this leaves us with two possibilities. The first is that raw stimulus information is combined and then pitch is extracted. The second is that periodicity information is extracted separately from each ear and that information is combined later to give the pitch percept.

We need to decide between two hypotheses:

- Early cross-over hypothesis: Left and right signals are combined before periodicity processing takes place
- Late cross-over hypothesis: Periodicity is processed separately in each ear and the results shared

To make specific predictions about the outcome of such experiments, we need to develop suitable computational models. The models to be described below are developments of monaural autocorrelation models.

11.1 Model outline

The 'binauralPitch' folder contains two models of binaural pitch extraction 'early-crossover' and 'late-crossover' (see Figure 9). Both models work on the basis that all auditory processes are duplicated on the left and the right sides. There is sharing of information of the results of these processes between the left and the right systems. The two models differ in terms of the level at which the information is shared (before or after periodicity analysis).

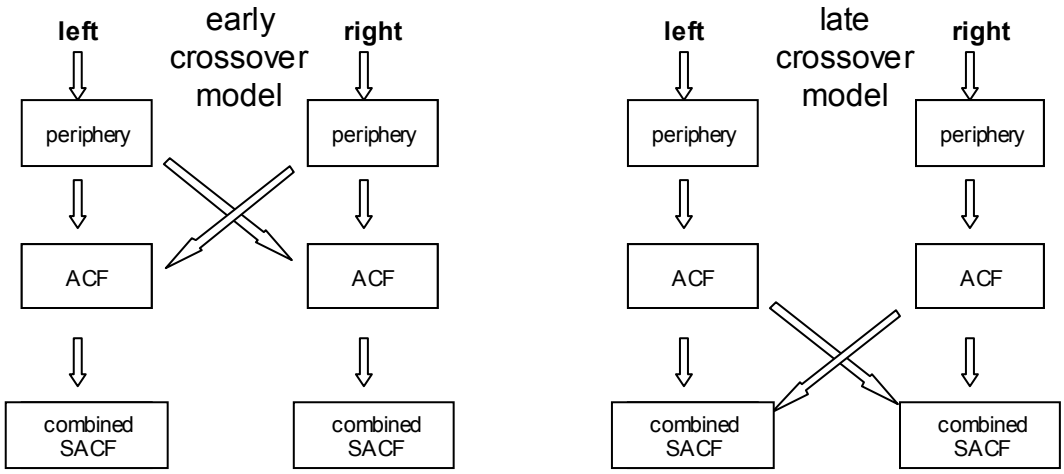


Figure 9 Comparison of the structure of the early-crossover and late-crossover models.

In the *early-crossover* version, the input from the left and right auditory nerve (AN) is combined before the autocorrelation analysis is performed.

In the *late-crossover* version, a separate autocorrelation analysis is performed on the left and right input from the auditory nerve. The results of these two autocorrelation analyses are then combined across the left/right divide.

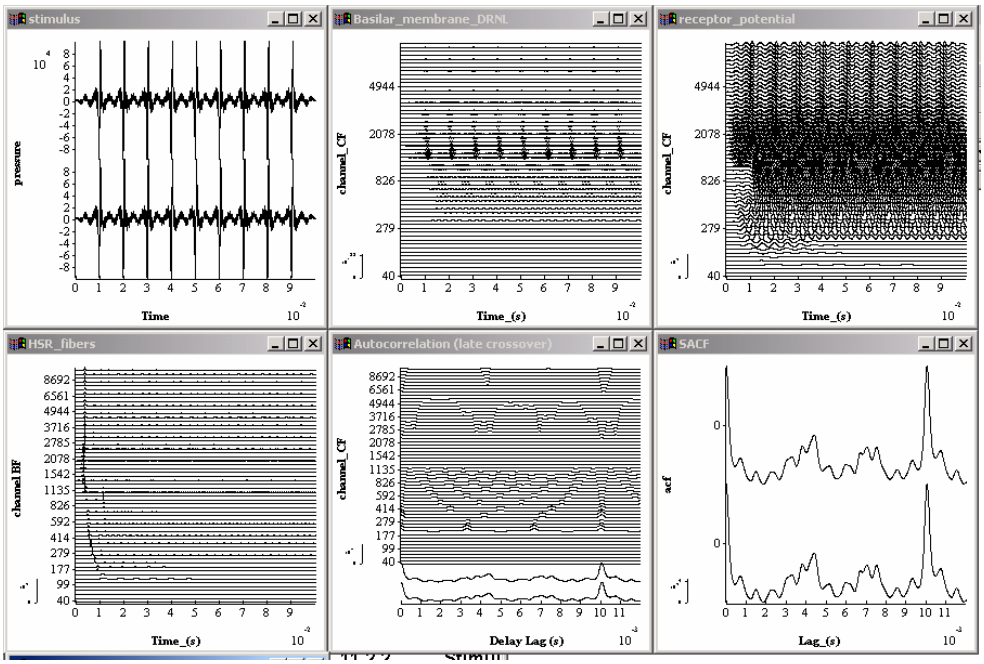


Figure 10 Output from early cross-over model for the diotic pitch stimulus (all harmonics to both ears (dioticF0100H3_12.wav).

The models are binaural and require a binaural input signal. Signal processing is separate for the left and right auditory nerves for both early-crossover and late-crossover models. In the displays, the left and right channels are interleaved so that corresponding channels can be compared.

Both models have the same three processing stages

- periphery (see sections 9.3.2 to 9.3.10)
- autocorrelation
- cross-ear combined summary autocorrelation (SACF)

Processing of binaural signals requires no special programming. DSAM\AMS automatically detects and processes binaural signals appropriately. For a binaural signal, each channel occupies two adjacent lines (left and right) in each display.

11.2 Early-crossover model

11.2.1 .sim file

```
#pitchEarlyCrossover.sim

# Simulation accepts a binaural input
#  combines the AN spike probability input from both ears
#  performs an acf on the combined input.

begin {

#  Stimulus generation
            DataFile_In             < dataFileIn.par

#apply ramps to signal
            Trans_Gate              < rampUp.par
            Trans_Gate              < rampDown.par
            Display_Signal          < displayStimulus.par

# outer/middle ear simulation
            Filt_MultiBPass         < filtIIRMoore.par

# convert to stapes velocity
            Util_mathOp             < mathOPstapes.par

# nonlinear DRNL filterbank
            BM_DRNL                 < BM_DRNLHuman.par
            Display_Signal          < displayBM.par

# receptor potential
            IHCRP_Shamma3StateVelIn < IHCRP_VelIn_GP.par
            Display_Signal          < displayReceptorPotential.par

# HSR auditory nerve synapse
ihcrp% IHC_Meddis2000 (->swap, acc, dispHR) < IHC_Meddis2000HSR.par
dispHR%     Display_Signal    (ihcrp->)     < displayHSR.par

#create left right mirror image
swap%       Util_swapLR       (ihcrp->acc)

#average left/right ihc probabilities together
acc%        Util_Accumulate   (ihcrp, swap-> )

#autocorrelate
            Ana_Acf                 < acf.par
            Display_Signal          < displayACFearly.par

#create summary acf by collapsing across channels
```

```
a%          Util_ReduceChannels     < UtRedChans.par
            Display_Signal          < displayACFearly.par
#write results to a data file
            DataFile_out            < dataFileOut.par
            }
```

11.2.2 Stimulus input

```
#           DataFile_In             < dataFileIn.par

FILENAME    dioticF0100H3_12.wav    Stimulus file name.
GAIN        100               Relative signal gain (dB).
```

The input file is a binaural *.wav* file. The file must be binaural or an error will be generated. The binauralPitch folder contains a number of .wav files, both dichotic and diotic harmonic stimuli. For example, *dioticF0100H3_12.wav* is a diotic stimulus with F0=100 Hz and harmonics 3 to 12.

The .wav files were generated using MATLAB programs also given in the binauralPitch folder; *dioticHarmonicStimulus.m* and *dichoticHarmonicStimulus.m*

The GAIN has been set to 100. This yields signal peak of 74 dB SPL. Of course, the effect of the gain depends upon the values in the *.wav* file. However, if the peak value in the *.wav* file is 1, a GAIN of 100 dB will yield a peak at 74 dB SPL.

11.2.3 The peripheral model

This includes all modules between the outer/middle ear simulation (Filt_MultiBPass) and the generation of AN spike probabilities (IHC_Meddis2000). See sections 9.3.2 to 9.3.10 for a complete description.

11.2.4 Crossover

```
#create left right mirror image
swap%       Util_swapLR       (ihcrp->acc)

#average left/right ihc probabilities together
acc%        Util_Accumulate   (ihcrp, swap-> )
```

The combination of information of left and right AN activity takes place using the swap module (Util_swap). This module takes a copy of the binaural AN response and swaps left with right to create a mirror image. *Util_Accumulate* then adds the original AN response to the mirror image to create a binaural combination of the left and right AN response. This might be represented algebraically,

AN response = [AN_left, AN_right]
swapped response = [AN_right, AN_left]
combination = AN response+ swapped response
=[AN_left +AN_right, AN_right+AN_left]

Note that the left and right are now the same.

The *Util_Accumulate* module adds together two or more inputs. The inputs are defined in the path indicator *(ihcrp, swap->)*These indicators refer to row labels (e.g. *ihcrp* % and *acc*%). These labels are then used to define 'connectors' that are shown in parentheses. For example

- *(-> swap, acc)* means 'take the output of this module and send it to the two modules labelled swap% and acc%'.
- *(ihcrp, swap->)* means
    - use as **input** to this module the output from the modules labelled *ihrcp*% and *swap*%

11.2.5 Autocorrelation and SACF

The process Util_Acf performs the autocorrelation. A separate ACF is computed for each channel. The left and right channels are interleaved in the display.

The summary ACF is computed using the Util_ReduceChannels module. This adds across channels to produce a single channel, the SACF.

See section 10.1.5 and 10.1.7 for more information

11.3 Late-crossover model

11.3.1 .sim file

The late-crossover model is similar to the early-crossover model in most respects. However, the sharing of information between left and right takes place after the computation of the ACFs.

```
# pitchLateCrossover.sim

#  simulation accepts a binaural input
#  computes an acf for each ear independently
#  and then averages the two acfs.

begin {

#  Stimulus generation
            DataFile_In             < dataFileIn.par

# apply ramps to signal
            Trans_Gate              < rampUp.par
            Trans_Gate              < rampDown.par
            Display_Signal          < displayStimulus.par

# outer/middle ear simulation
            Filt_MultiBPass         < filtIIRMoore.par

# convert to stapes velocity
            Util_mathOp             < mathOPstapes.par

# nonlinear DRNL filterbank
            BM_DRNL                 < BM_DRNLHuman.par
            Display_Signal          < displayBM.par

# receptor potential
            IHCRP_Shamma3StateVelIn < IHCRP_VelIn_GP.par
```

```
            Display_Signal          < displayReceptorPotential.par
# HSR
            IHC_Meddis2000          < IHC_Meddis2000HSR.par
            Display_Signal          < displayHSR.par

# autocorrelate
            Ana_Acf                 < acf.par
            Display_Signal          < displayACFlate.par

# create summary acf
redch%      Util_ReduceChannels (->swap, acc)   < UtRedChans.par

# create left right mirror image
swap%       util_swapLR       (redch->acc)

# add left/right acfs together
acc%        Util_Accumulate   (redch, swap-> )
            Display_Signal          < displayACF2.par

#write results to a data file
            DataFile_out            < dataFileOut.par
            }
```

This model is the same as the early crossover model up to the point where the autocorrelation is performed. In this model the autocorrelation is performed separately on the left and right channels before the left and right channels are combined.

11.3.2 Example

The two models can now be tested with diotic and dichotic stimuli:
Diotic

- Both left and right channels have 10 harmonics of a 100-Hz fundamental starting with the 3rd harmonic. (dioticF0100H3_12.wav)

Dichotic

- The left channel consists of 5 odd-numbered harmonics of a 100 Hz fundamental starting with the 3rd harmonic.
- The right channel consists of 5 even numbered harmonics of a 100 Hz fundamental starting with the 4th. (dichoticF0100H3_12.wav).

Diotic stimulus. Figure 11 shows the response of the early cross-over model to the diotic stimulus. The combined SACF shows a substantial peak at a lag of 10 msec. This pattern would normally be associated with a percept of a 100-Hz pitch

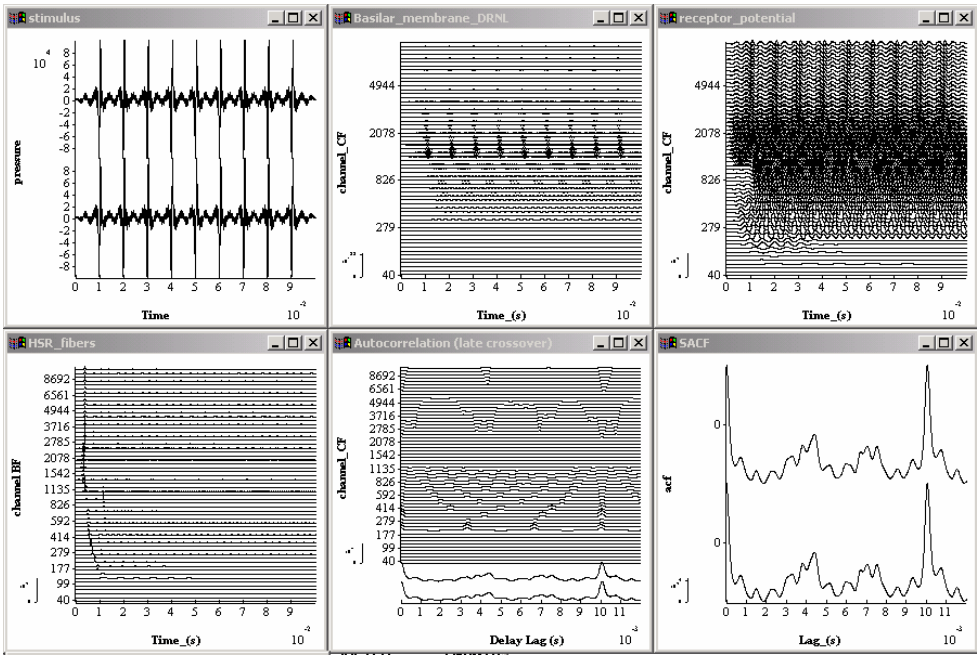


Figure 11. Early-crossover model with dichotic stimulus (see text)

Figure 12 shows the screen display for both the late-crossover model for exactly the same stimulus. The output is the same as Figure 11. Both early and late crossover models predict a pitch percept of 100 Hz.

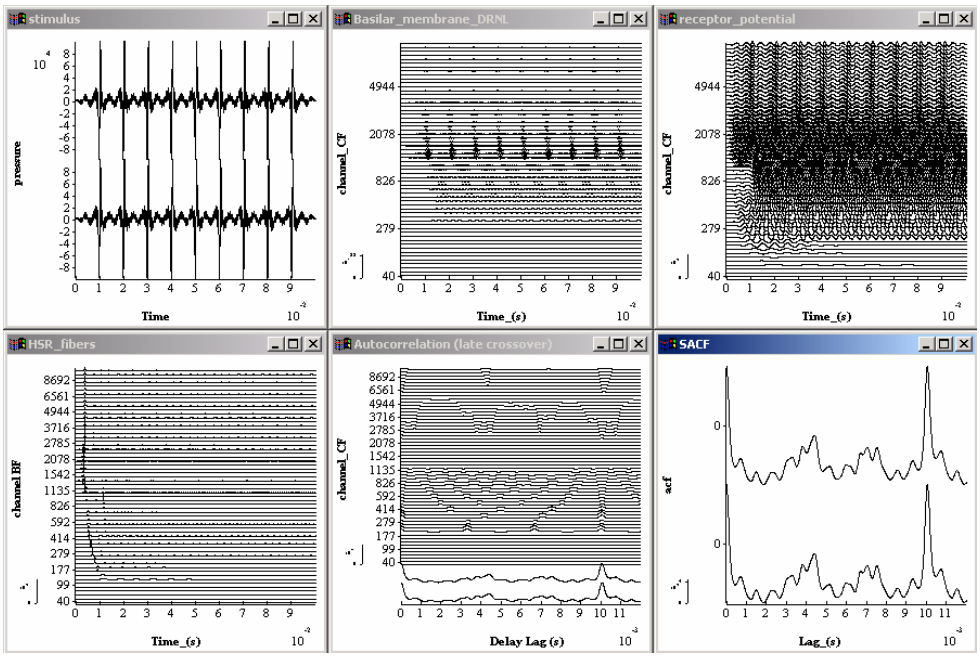


Figure 12. Late-crossover model with dichotic stimulus (see text)

Dichotic stimulus. Figure 13 shows the response of the early cross-over model to the dichotic stimulus. The combined SACF shows a substantial peak at a lag of 10 msec.

This pattern would normally be associated with a percept of a 100-Hz pitch, that is no change from the diotic condition.

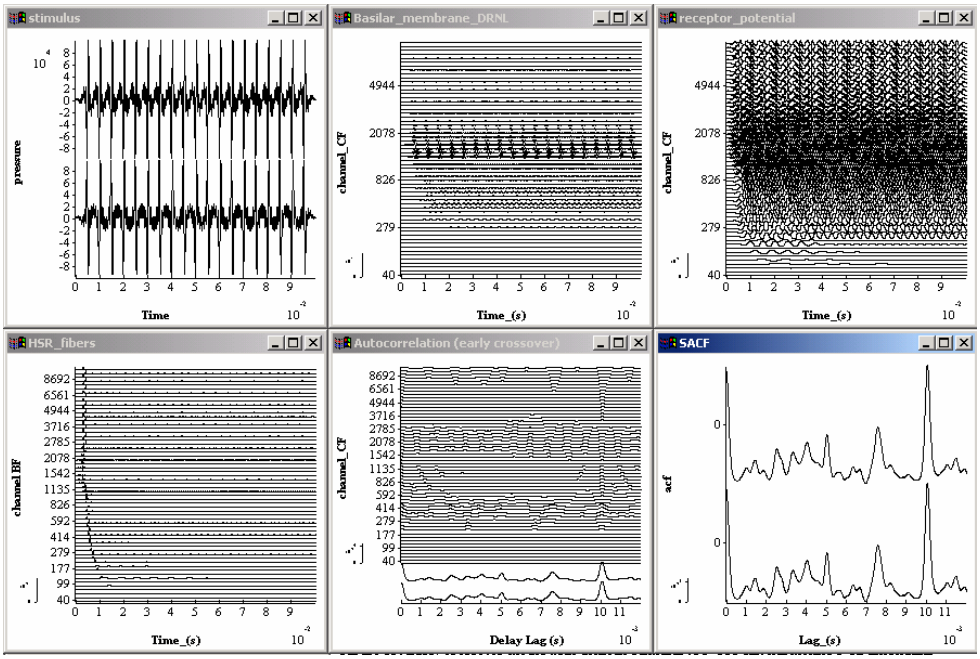


Figure 13. Early-crossover model with dichotic stimulus (F0=100 Hz, harmonics 3-12 alternating between ears).

Figure 14 shows the late cross-over model's prediction for the diotic stimulus. It is the same as the early cross-over model. Both models predict that both diotic and dichotic stimuli will give rise to a 100 Hz pitch percept.

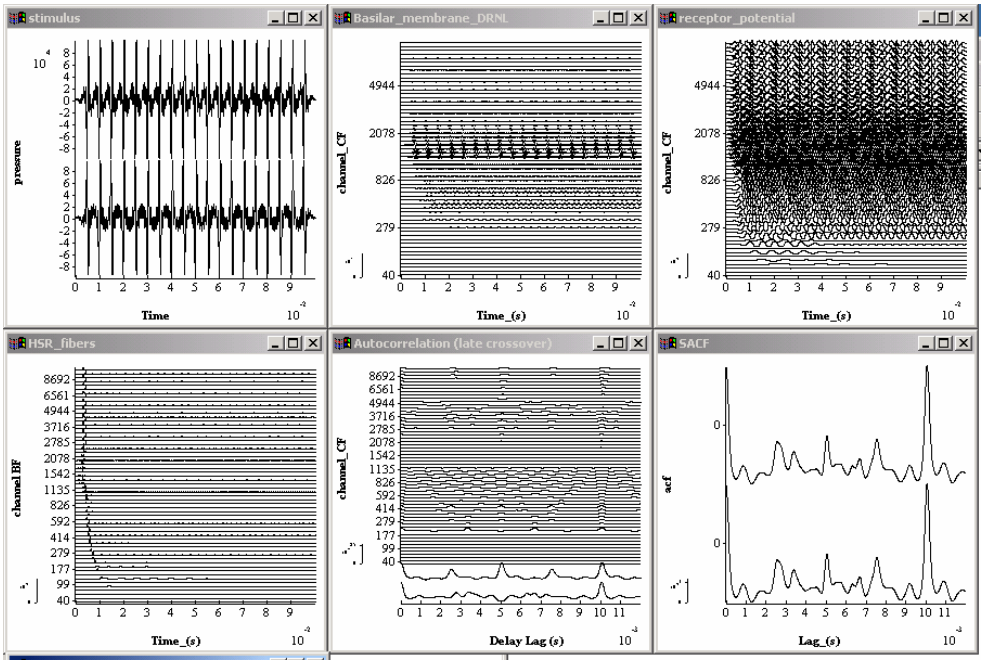


Figure 14. Late-crossover model with dichotic stimulus (F0=100 Hz, harmonics 3-12 alternating between ears).

11.4 Discussion

Both models (early and late crossover models) predict that both stimuli (dichotic and diotic) will be perceived as having a 100-Hz pitch. The fact that diotic and dichotic stimuli both give the same pitch cannot, therefore, be used to support the conclusion that the pitch detection mechanism occurs after combination of binaural information, Houtsma and Goldstein (1972).

At first sight the result looks trivial; after all why do we need to explain a null result? However, close examination of the autocorrelation figures shows that it is more complex than can be easily intuited. Figure 15 shows that the analysis gives different summary SACFs for all four conditions. We might expect this because the diotic and dichotic stimuli are very different. It is not surprising that the early crossover model gives a similar pitch prediction for both stimuli. This is because the AN effects of the separated harmonics are brought together before the autocorrelation process.

However, in the late crossover model, one might expect that 200-Hz periodicities would be more prominent for the dichotic stimulus. This is certainly true for the left SACF (upper trace, even harmonics) where a clear peak can be seen at 5-msec lag. However, (and here is the surprise), the lower trace (odd harmonics) shows a dip at 5 msec. When these two traces are combined to give the final pitch percept, the 5-msec peak and trough cancel leaving the main peak firmly at 10 msec with a resulting prediction of 100-Hz pitch percept.

diotic (same to both ears) dichotic

early

late

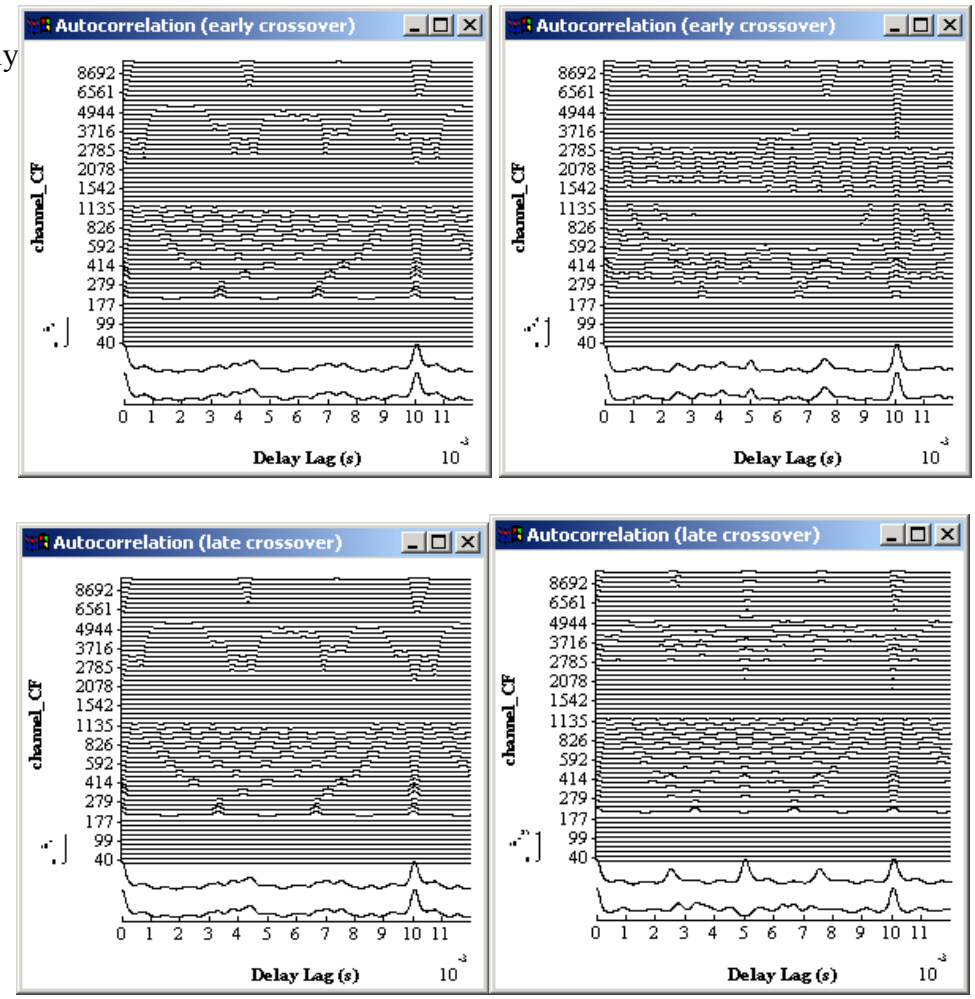


Figure 15 Comparison of the autocorrelation functions for diotic and dichotic stimuli presented to the early and late crossover models.

12 Precedence

This model can be found in the folder *Tutorials\Precedence.* The original version of this folder can be found in *DSAM\AMS\Tutorials* This model can be found in the folder *Tutorials\Precedence.* If you wish to change the program, copy this folder into your regular workspace. Please, do not make changes in the *AMS* workspace.

12.1 Model implementation (.sim and .par files)

12.1.1 .sim file

The precedence model is based as closely as possible on the precedence model of Hartung and Trahiotis (2001). In their model, they show that many effects associated with the title 'precedence effect' can be explained in terms of the activity of the auditory periphery (i.e. do not require neural processes such as inhibition).

Precedence concerns the phenomenon where the ITD of the first of a pair of binaural clicks determines the location of the clicks. In other words, the second pair of clicks is largely ignored. It is taken into account but is given a much lower weighting in the decision-making process.

This model is slightly different from the published version.. It has been updated to include the latest nonlinear peripheral model and the latest hair cell model from the Essex stable. There is no reason to persist with the original linear models now that we have useful nonlinear models. It does, however, raise the question of whether it still works with a nonlinear periphery.

There is another, less important difference. Hartung and Trahiotis did not use a running cross correlation but integrated across the whole 30 msec waveform. This simplifies the modelling process by avoiding some difficult decisions. In this implementation, however, we have used a 50-msec stimulus period starting with a 20 msec silence. The result is then integrated with a time constant of 15 msec.

The model has a number of stages

- Stimulus Input; A pair of binaural clicks.
- Auditory periphery: This section uses the same auditory periphery as described in the Loudness tutorial. The output from the periphery is a stream of *probabilities* of action potentials in fibers of the AN.
- Cross-correlation. A separate cross-correlation (CCF) is computed at each stimulus frequency.
- Summary CCFs are computed by adding vertically all the rows in the CCF matrix (i.e. across all BF channels).
- Write to file

```
#precedence.sim

off               Diagnostics mode ("off" or "on").
relative          Parameter file path mode - \relative\ or \absolute\
```

```
begin {

#  Stimulus generation
# left clicks
L1%         stim_click        (->Lacc)    <click_L1.par
L2%         stim_click        (->Lacc)    <click_L2.par
Lacc%       Util_accumulate   (L1, L2->bin)

# right clicks
R1%         stim_click        (->Racc)    <click_R1.par
R2%         stim_click        (->Racc)    <click_R2.par
Racc%       Util_accumulate   (R1, R2->bin)

#combine left and right clicks into binaural stimulus
bin%        Util_createBinaural     (Lacc, Racc->)

# allow screen control of level
            Util_mathOp             < mathOpClickLevel.par
            Display_Signal          < displayStimulus.par

#outer/middle ear simulation
            Filt_MultiBPass         < filtIIRMoore.par

#convert to stapes velocity
            Util_mathOp             < mathOpstapes.par


#nonlinear DRNL filterbank
            BM_DRNL                 < BM_DRNLHuman.par
            Display_Signal          < displayBM.par

#receptor potential
            IHCRP_Shamma3StateVelIn < IHCRP_VelIn_GP.par
            Display_Signal          < displayReceptorPotential.par

# HSR spike probability
            IHC_Meddis2000          < IHC_Meddis2000HSR.par
            Display_Signal          < displaySpikeProb.par


# cross correlate
            Ana_ccf                 < ccf.par
            Display_Signal          < displayCCF.par

#create summary acf
            Util_ReduceChannels     < UtRedChans.par
            Display_Signal          < displayCCFsummary.par


#write results to a data file
            DataFile_out
            }
```

12.1.2 Example

When the model is run, the screen should look something like this.

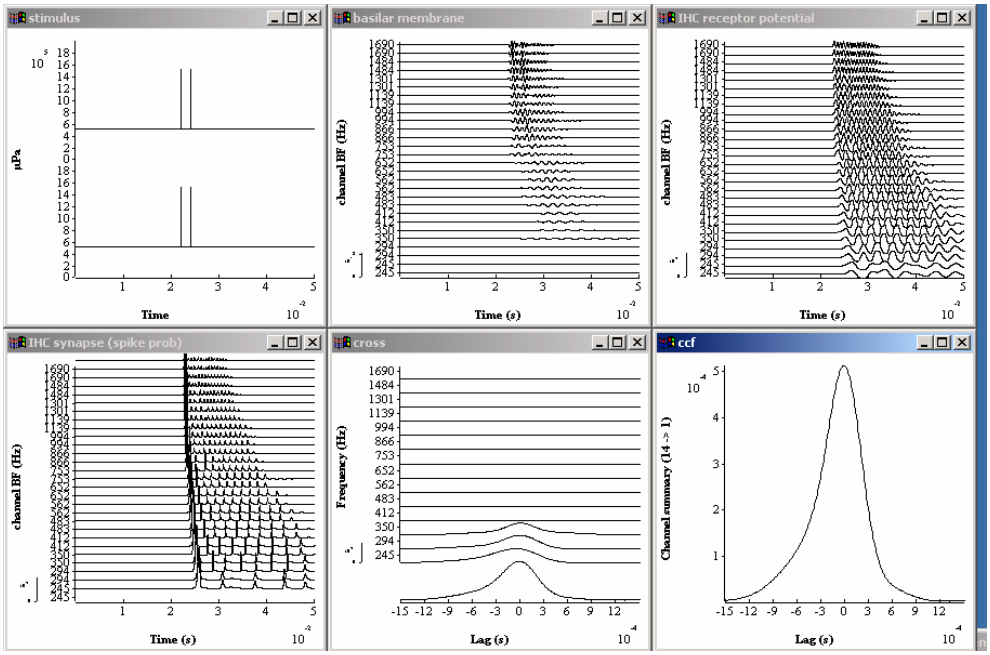


Figure 16 Precedence model

12.1.3 Generate click stimuli

We have four clicks; two on the left (L1, L2) and two on the right (R1, R2). We compute them all separately so that we can have individual control over their timing. The current set up creates a pair of clicks 2 msec apart with 0-msec ITD. The image should be central (last display in Fig. 17) .

Each click parameter file looks like this.

```
#           stim_click        <click_L1.par

TIME        22e-3
AMPLITUDE   1e6
DURATION    50e-3
DT            1e-5
```

TIME is when the click occurs (after 22 msec).
AMPLITUDE is the peak pressure in microPascals. These clicks are as wide as the sampling interval (DT). Note that many psychophysical experiments have wider clicks.
DURATION is the duration of the total stimulus including silence.
DT is the sampling period

First we create the two clicks on the left and then add them together

```
Lacc%       Util_accumulate    (L1, L2->bin)
```

Util_accumulate requires no parameter file. It just adds all its inputs together. The inputs are specified in the brackets (L1, L2 -> bin) that also specify that the output goes to the module labelled ‘bin%’.

Then we create the two clicks on the right and then add them together

```
Racc%       Util_accumulate   (R1, R2->bin)
```

12.1.4 Create Binaural stimulus

Combine the left and right stimulus into a binaural signal

```
bin%        Util_createBinaural     (Lacc, Racc->)
```

Util_createBinaural requires two inputs labelled Lacc and Racc to form the left and right components of the binaural signal.

The overall level of the stimuli can be controlled by a scalar applied by mathOp

```
            Util_mathOp             < mathOpClickLevel.par
```

The scalar is set to 1, so does not influence anything but is there for the user should he wish to explore very low or high signal levels without having to reset each click level independently.

12.1.5 Auditory periphery

This is the same as that used in the *Loudness* model and is described fully in sections 9.3.5 to 9.3.14.

```
            Filt_MultiBPass         < filtIIRMoore.par

#convert to stapes velocity
            Util_mathOp             < mathOpstapes.par


#nonlinear DRNL filterbank
            BM_DRNL                 < BM_DRNLHuman.par
            Display_Signal          < displayBM.par

#receptor potential
            IHCRP_Shamma3StateVelIn < IHCRP_VelIn_GP.par
            Display_Signal          < displayReceptorPotential.par

# HSR spike probability
            IHC_Meddis2000          < IHC_Meddis2000HSR.par
            Display_Signal          < displaySpikeProb.par
```

Note: The BM_DRNLHuman.par file has been adjusted to use the same range of 14 best frequencies (224.7 to 1690 Hz) as used by Hartung and Trahiotis.

12.1.6 Cross correlation

This module performs a within-channel running cross correlation function and displays the final state of the function at the end of the signal.

```
            Ana_ccf                 < ccf.par
```

```
OFFSET      48e-3
TIME_CONST   1e30
MAX_LAG     1.5e-3
```

OFFSET specifies the time at which the running cross correlation is to be displayed. The OFFSET must be shorter than the duration of the stimulus. As a rule of thumb, OFFSET+MAX_LAG <= DURATION.
(DURATION is specified in the click parameter files).

TIME_CONST specifies the time constant of a running cross-correlation function. Here we have set the time constant to a very large number (1e30). This is effectively an infinite time constant.

MAX_LAG specifies the longest ITD to be computed (H&T used +/- 1.5 msec).

12.1.7 Display_Signal < displayCCF.par

We want to see the summary cross correlation coefficient (SCCF) as well as the individual CCFs. This can be achieved by setting the subsequent display module parameter 'summary_display' to 'ON'. Note that this is a display facility only. The SACF is not passed to the next module. The output from Ana_ccf is the matrix of cross correlation functions (channel *x* lag).

It is also important to label the x-axis correctly ('Lag') to avoid confusion.

```
#           Display_Signal          < displayCCF.par

Y_AXIS_MODE CHANNEL     Y-axis mode ('channel' (No.) or 'scale').

AUTO_Y_SCALE      ON    Automatic y-axis scale ('on' or 'off').

X_AXIS_TITLE      "Lag (s)"

WIN_TITLE   cross correlation       Display window title.
SUMMARYDISPLAY    ON    Summary display mode ('on' or 'off').

WIN_HEIGHT  300   Display frame height (pixel units).
WIN_WIDTH   300   Display frame width (pixel units).
WIN_X_POS   300   Display frame X position (pixel units).
WIN_Y_POS   300   Display frame Y position (pixel units).
```

12.1.8 Computing the SCCF

This module performs a summation (or averaging) across channels of the signal. The user specifies how many output signals he wants. If two are selected, the top half of the signals are combined to produce the first output signal and the bottom half used to produce the second output signal. We want only one.

```
#           Util_ReduceChannels     < UtRedChans.par

MODE              SUM   Mode  'average' or simple 'sum'.
NUM_CHANNELS      1     Number of resulting channels.
```

12.1.9 Writing the results to file

The result of the preceding module is written to an output file. The default is 'output.dat'.

The output has the following format

```
Delay period (s)    0
-1.50000e-003 2.27253e-006
-1.49000e-003 2.39382e-006
-1.48000e-003 2.52502e-006
-1.47000e-003 2.66667e-006
-1.46000e-003 2.8193e-006
-1.45000e-003 2.98337e-006
.............
```

12.2 Fun things to do

12.2.1 Move to the left using the first click

Advance the time of the first left click (L1) by, say, 300 microSec. The CCF peak should move to the left.

12.2.2 Move to the left using the second click

Restore the time of the L1 and now advance the time of L2 by the same amount, say, 300 microSec. The peak should move to left of centre but by a smaller margin than observed when L1 was moved

12.2.3 Do some real modelling

Hartung and Trahiotis (2001) show a number of results using their implementation of the model (using linear filters and the old 86 hair cell model). You could try to replicate their results using the lates nonlinear model.

.

13 Bibliography

Hartung, K. and Trahiotis, C. (2001) “peripheral auditory processing and investigations of the ‘precedence effect’ which utilize successive transient stimuli”, J Acoust Soc Am, 110, 1505-1513.

Glasberg, B. R., and Moore, B. C. J. (2002). "A model of loudness applicable to time-varying sounds," J. Audio Eng. Soc. 50, 331-342.

Moore BCJ, Glasberg BR, Baer T. (1997). “A model for the prediction of thresholds, loudness, and partial loudness,” J Audio Eng Soc 45 (4): 224-240.

Greenwood Donald D. (1990) "A cochlear frequency-position function for several species 29 years later", J Acoust Soc Am, 87, 6, Cochlear Mechanics.

Lopez-Poveda E.A., and Meddis R. (2001a). "A human nonlinear cochlear filterbank," J. Acoust. Soc. Am. 110, 3107-3118.

Meddis, R. and O'Mard, (1997) "A unitary theory of pitch perception," Journal of the Acoustical Society of America. 102, 1811-1820.

Meddis, R., O'Mard, L.P., and Lopez-Poveda, E.A. (2001). "A computational algorithm for computing nonlinear auditory frequency selectivity," J. Acoust. Soc. Am. 109, 2852-2861.

Moore, B. C. J., Glasberg, B. R., and Baer, T. (1997). "A model for the prediction of thresholds, loudness and partial loudness," J. Audio Eng. Soc. 45, 224-240.

Moore, B.C.J. and Glasberg, B.R. (1987) "Formulae describing frequency selectivity in the perception of loudness, pitch and time," in *Frequency Selectivity in Hearing*, edited by B.C.J.Moore (Academic, London)

Plack , C.P. and Carlyon,R.P. (1995) 'Loudness perception and Intensity coding' in Hearing, ed. B.C.J.Moore (Academic, New York).

Relkin,E.M. and Doucet,J.R. (1991) Hearing Research, 55, 215-222.

Sumner, C., Lopez-Poveda, E.A., O'Mard, L.P. and Meddis, R. (2002). 'Adaptation in a revised inner-hair cell model,' submitted to Journal of Acoustical Society of America. (accepted subject to revision).

Sumner, C., O'Mard, L.P., Lopez-Poveda, E.A., and Meddis, R. (2003). 'A non-linear filter-bank model of the guinea-pig cochlea,', submitted to Journal of Acoustical Society of America (accepted subject to revision).

Sumner, C.J., O'Mard, L.P., Lopez-Poveda, E.A., and Meddis, R. (2002a). "A revised model of the inner-hair cell and auditory nerve complex," Journal of the Acoustical Society of America, 111, 2189-2199.

Houtsma, A.J.M. And Goldstein, J.L. (1972) "The central origin of the pitch of complex tones: evidence from musical interval recognition,", J Acoust Soc Am, 51, 520-529