Design of the Auditory Modelling Toolbox

January 17, 2013

1 Goals of the software

- High quality code: Easy to read and maintain
- High quality documentation in the code: This is the best place to keep the documentation, as it is easier to keep up to date when the code changes. The documentation can be extracted in HTML and TEX formats.
- Simplicity: Eliminate unneccesary configuration options. Always choose the simplest possible solution. This makes it much easier to read and modify the code.

2 Directory structure

`general` This directory contains general functions that cannot be placed elsewhere.

`monaural` Monaural models

`binaural` Binaural models

`humandata` Function returning recorded (and published) human data

`experiments` Descriptions of experiments that reproduce a figure in a paper.

`demos` Simple demos describing how to use the toolbox. The demos usually generates some plot, which are automatically put on the homepage. The are prefixed by demo_

`mex` All Matlab `.c` mex files

`oct` All Octave `.cc` C++ files

`comp` Computational routines. They are all prefixed by `comp_` Computational subroutines does not need to check input arguments, they never take a variable number of input arguments, and in many cases they are shadowed by a C/Mex/Oct implementation.

`src` C source code

`testing` Simple test scripts to verify that new and old code produce the same result, and to perform other tests. Each test function is prefixed by `test_` A test function takes no input, and returns a single variable indicating how many tests failed.

`reference` The `reference` directory are used to keep old, unmodified code that is known to work or simple (but slow) implementations of algorithms. They are used to compare other functions against. Reference function are all prefixed by `ref_`

2.1 The init file

The `mydirinit.m` gets executed when the main upstart routine `amtstart` is run. The upstart routine adds the directory `mydir` to Matlab's search path before executing the routine.

The `mydirinit` script must define a variable called `status` and set it to 1 (one) if the toolbox should be loaded. Any other value of status will case the `mydir` directory to be removed again from the search path.

This simple mechanism enables you to check if everything it correct, and refuse loading the directory if something is not.

Similarly, it is possible to set a variable `module_version` to any number, and this will be the version number of the module. Otherwise, the global version number will be used.

So in the most common case, the `mytoolboxinit.m` file will contain just a single line:

```
status=1;
```

3 Input and output

3.1 Data in general

It is generally assumed that all data is a regular sampling of a continuous phenomena at a certain sampling frequency `fs`. Many routines will therefore ask for both a signal and a sampling frequency. By the same reasoning, frequencies are usually specified in Hz because we assume the sampling frequency to be known (so don't do normalized frequencies between 0 and 0.5 as Matlab sometimes does).

3.2 Definition of the input to the model

If nothing else is noted, this should be the description of sound input to any model:

An acoustical signal is representated by a column vector of numbers. The numbers are obtained by sampling the air pressure of the acoustical signal at a

constant sampling rate. The numbers are scaled such that an acoustical signal with a level of 100 dB SPL corresponds to a digital signal with an RMS value of 1.

3.3 General Data structures

It is probably impossible to list the relevant data structures before developing the software, but the few ones listed here should always be used.

- A single channel signal is a column vector.
- A multi channel signal is a column matrix.
- A multi-channel signal stemming from a filterbank has time as first dimension, channel number as second, and original signal channel (left/right) as third dimension.

3.4 Specific data strucures

- The output from the modulation filterbank has time as first dimension, frequency channel number as second, modulation channel number as third and original signal channel (left/right) as fourth.

4 General coding standards

- The same parameter should have the same name across all files. See the section on common variable names.

4.1 Matlab specific coding standards

4.2 C and Fortran specific coding standards

- Variables name are allowed to be both lower and upper case. This convention is called *camel casing*, see http://en.wikipedia.org/wiki/CamelCase. The general rule is that the word starts with a lower case letter, and then the following words are starts will upper case letters.

5 Function names

- All function names in Matlab must be lowercase. This avoids a lot of confusion because some computer architectures respect upper/lower casing and others do not. Furthermore, function names are traditionally written in uppercase in Matlab documentation.
- Models are named after the paper in which they first appeared, as in dau96, or after their commonly accepted name if theres is no doubt as to which model is the name refers.

- *As much as possible*, functions are named after the function they perform, rather than the algorithm they use, or the person who invented it.
- It is not allowed to use underscores in function names. They are reserved for structural purposes, i.e. as in `demo_gammatone` or `test_dau96`.
- If a model consist of several files, the must all be prefixed by the model name, as in `takanen2013mso` and `takanen2013lso`.

6 Variable names

- Variable should not have the same name as an already existing function is Matlab. This makes the code easier to read and less prone to errors. However, as almost all short names are taken by a Matlab function, this can be hard to obtain in practice.
- No global variables. Global variables makes it harder to debug, and the code cannot be parallized.
- Never use `i` or `j` as a variable name in Matlab, as they are used for the imaginary unit. This creates a great deal of confusion when reading other peoples code. Please use `ii` and `jj` instead, or something completely different. Using `i` and `j` are allowed in C, which does not have an imaginary unit.

The following is a list of common variables.

`insig`	Input signal
`outsig`	Output signal
`inoutsig`	Some simple functions modify the signal in place, so the signal is both input and output. This may save some memory and processing time. Please write 'insig' and 'outsig' in the documentation, as the user should not care about this detail.
`fs`	Sampling frequency.
`siglen`	Length of signal
`nsigs`	Number of signals
`fc`	Center frequency/frequencies of filter/filter-bank.
`flow`,`fhigh`	Generic low, high frequencies determining a range of frequencies.
`a,b`	Filter coefficients to IIR filters.

7 Caching files

Some models take a very long time to complete, and it is therefore necessay to cache the result after a run in order to not need to recompute everything if a simple plotting parameter is changed. In order to provide a uniform mechanism for handling this problem, AMToolbox provides the `amtredofile` function. `amtredofile` returns true if a file should be recalculated, and false if it is safe to load the contents of the file. It accepts the following flags which can be inherited by the function using `amtredofile:`

`'autorefresh'` Re-calculate the file if it does not exist. Return 1 if the file exist, otherwise 0. This is the default.

`'refresh'` Always recalculate the file.

`'cached'` Always use the cached version. Throws an error if the file does not exist.

To use `amtredofile` in your functions and scripts, please use the following code structure:

- Add the following to your help section to document the flags. Modify as necessary:

```
%     'auto '    Redo the experiment if a cached dataset
%                does not exist. This is the default.
%
%     'refresh'  Always recalculate the experiment.
%
%     'cached'   Always use the cached version. This
%                throws an error if the file does not exist.
```

- Before you call ltfatarghelper, add the following line to the list of input parameter definitions. This will import the flags.

```
definput.import={'amtredofile'};
```

- Enclose you code in if-statements of the following type:

```
save_format='-v6';
s = [mfilename('fullpath'),'_myfigure.mat'];
if amtredofile(s,flags.redomode)
  ... run your expensive computation here,
  ... output is in out1 and out2 as an example ...
  save(s,'out1','out2',save_format);
else
  s = load(s);
  % Unload the structure, you could also use it directly
```

```
  out1 = s.out1;
  out2 = s.out2;
end;
```

8 Rules for writing demos and experiments

- Demo must run without requiring user input, neither from the keyboard or from calling them as functions. This is because the documenation system must be able to run them in the background.
- A demo can be a function, as long as the function can be run without taking any inputs, see above.
- A demo, experiment or example in a function may not use the Matlab GUI or the waitbar function: The documentation system cannot handle this in the background.
- You cannot load data like "load mydata", because you cannot assume anything about the path. All loads must use the full path to the function. This can easily be done by using "mfilename" (see the code for examples) or "amtbasepath"

9 Structure of a human data file

```
function [outx,outy]  = data_soendergaard2099(varargin)
%DATA_SOENDERGAARD2099  This is the headline that describes the function
%   Usage: [outx,outy] = data_soendergaard2099()
%
%   `[outx,outy]=data_soendergaard2099(flag) returns data points from the Soendergaard
%   of:
%
%     'noplot'        Don't plot, only return data. This is the default.
%
%     'plot'          Plot the data.
%
%     'figXXX'        Put in a description of the output data.
%
%     'figYYY'        Put in a description of the output data.
%
%   Examples:
%   ---------
%
%   Figure 5 can be displayed using :::
%
%     [out1,out2] = data_soendergaard2099('fig5','plot');
%
```

```
%   Figure 7 can be displayed using :::
%
%     [out1,out2] = data_soendergaard2099('fig7','plot');
%
%   References: soendergaard2099universe
definput.flags.plot = {'noplot','plot'};
definput.flags.type={'figXXX,'figYYY'};
[flags,keyvals]  = ltfatarghelper({},definput,varargin);
if flags.do_figXXX
  outx = [0, 1.3750, 1.3750, 1.6250, 1.8750, 2.7083];
  outy = [0.4513, 0.7500, 0.2500, 0.4590, 0.3436, 0.5159];
  if flags.do_plot

    % Put the visualization of the data here
  end;
end;
% Do the same for the next figure
if flags.do_figYYY
  ...
end;
```

10 Anchors

An anchor is a capitalized word in the code, that can be extracted by a script.
Each file should contain the following anchors:

AUTHOR The line following this anchor indicates who the authors are.

For debugging it is possible to insert XXX, FIXME, BUG, TODO into the code, followed by a description of the problem.