Quick-Auditory-Filter (qAF) Toolbox: User Manual

Hearing Lab, University of California, Irvine

Applied Hearing Science Lab, Indiana University, Bloomington

January 29, 2015

Contents

1 GENERAL INFORMATION

1.1 Toolbox Overview

One of the fundamental features of the auditory system is its tonotopic organization. The auditory periphery acts as a frequency analyzer, mapping different frequency components of sounds to specific locations along the basilar membrane. Functionally, this process can be modeled as a bank of band-pass filters, namely auditory filters. The shape of the auditory filter, in particular its bandwidth, is highly predictive of perceptual phenomena such as masking.

While the estimation of the auditory filter plays a fundamental role in the study of auditory perception, it is traditionally a time-consuming process that may require up to two hours to complete data collection. Shen and Richards (2013) and Shen et al. (2014) described a Bayesian adaptive procedure for the efficient estimation of the auditory-filter shape, the Quick-Auditory-Filter (qAF) Procedure. The toolbox described in this user manual provides an efficient implementation of the qAF Procedure in Matlab.

The qAF Toolbox provides the following features:

- Various parameterizations of the auditory filter are possible
- Data are organized intuitively
- Parameter-space configurations are flexible
- Implementation of simulations is straightforward

1.2 Authorized Use Permission

2 INSTALLATION

2.1 Software Requirements

The qAF Toolbox is designed to be used with Matlab version 2008a or newer, which supports an object-oriented framework and syntax with the statistics toolbox installed.

2.2 Hardware Requirements

There are no specific hardware requirements for the qAF Toolbox.

2.3 Installing the qAF Toolbox

1. Download the `qAF_Toolbox.zip` file to your computer. Unzip the file and place the contents in a directory that is available to Matlab.
2. In Matlab, select the directory containing the files from the `qAF_Toolbox.zip` as the current directory.
3. The qAF Toolbox is ready to use.

2.4 Files Included in the qAF_Toolbox.zip File

- `qAF.m`
- `exp_config.m`
- `template.m`
- `example_qAF_roex.m`

2.5 Quick Start

To construct a new experiment, open the `template.m` file from the unzipped folder using the Matlab editor. Follow the instructions in the file. It will lead you through the steps to modify a template experiment. The user must provide code for stimulus generation and presentation, for collecting responses, and for saving the resulting data.

3 HOW DOES THE qAF TOOLBOX WORK?

3.1 The qAF Algorithm

The qAF algorithm is a computational algorithm that adaptively determines the stimuli to be presented in a notched-noise masking experiment in order to efficiently estimate the auditory-filter shape. The notched-noise masking experiment has been the main experimental paradigm for the estimation of the auditory-filter shape (e.g., Patterson, 1976; Patterson et al., 1982).

In a typical experiment, listeners detect the presence of a pure-tone target presented in noise. The noise, i.e. the masker, contains two frequency bands, one on each side of the target frequency, forming a spectral notch (see Fig. 1a). The masker can then be fully described by the upper notchwidth (the distance from the lower edge of the upper masker band to the target frequency), the lower notchwidth (the distance from the upper edge of the lower masker band to the target frequency), and the masker spectrum level (L_m, in dB SPL). The upper and lower notchwidths (g_u and g_l, respectively) are often expressed on a normalized frequency scale:

$$g = \frac{\Delta f}{f_0}, \tag{1}$$

where Δf is the distance from the target frequency f_0.

According to the power spectrum model of masking, the detection threshold is defined at a certain target-to-masker intensity ratio K at the output of the auditory filter centered at f_0. Therefore, as the masker notches move away from the target frequency, higher masker levels are needed to mask a fixed-level target. By measuring the masker level needed to mask the target for various combinations of the upper and lower masker notchwidths, it is possible to reverse-engineer the shape of the auditory filter with a few assumptions. However, the drawback of this procedure is that the masker level at threshold has to be measured repeatedly at each of the notchwidth combinations, leading to time-consuming data collection. For a typical experiment with more than 10 notchwidth combinations to be tested, the total testing time could be more than two hours.

The qAF algorithm estimates the auditory-filter shape in a single experimental track, in which the masker properties (g_u, g_l, and L_m) are adaptively manipulated on a trial-by-trial basis (see Shen et al., 2014, for a more detailed description of the qAF algorithm). Following Rosen et al. (1998) and Oxenham and Shera (2003), the qAF algorithm uses a roex($pwtp$) model (Patterson et al., 1982) for the auditory-filter shape (see Fig. 1b). The filter shape is made of three components: a high-frequency skirt (described by a slope parameter p_u), a low-frequency tip (described by a slope parameter p_l), and a low-frequency tail (described by a slope parameter t). On the low-frequency side, the relative contributions of the tip and tail components are determined by a weight parameter w. Therefore, to fully describe the auditory-filter shape, all four parameters (p_u, p_l, t, and w) and the detection efficiency parameter K must be estimated.

The estimation of the model parameters is achieved by the qAF algorithm in the following way (see Fig. 1c). First, a prior distribution is specified for each of the parameters by the experimenter, providing an "initial guess" to the parameter values. As the experiment progresses, the posterior distributions of these parameters are updated iteratively. On each trial, the interim model estimate (based on the posterior parameter distributions) is used to

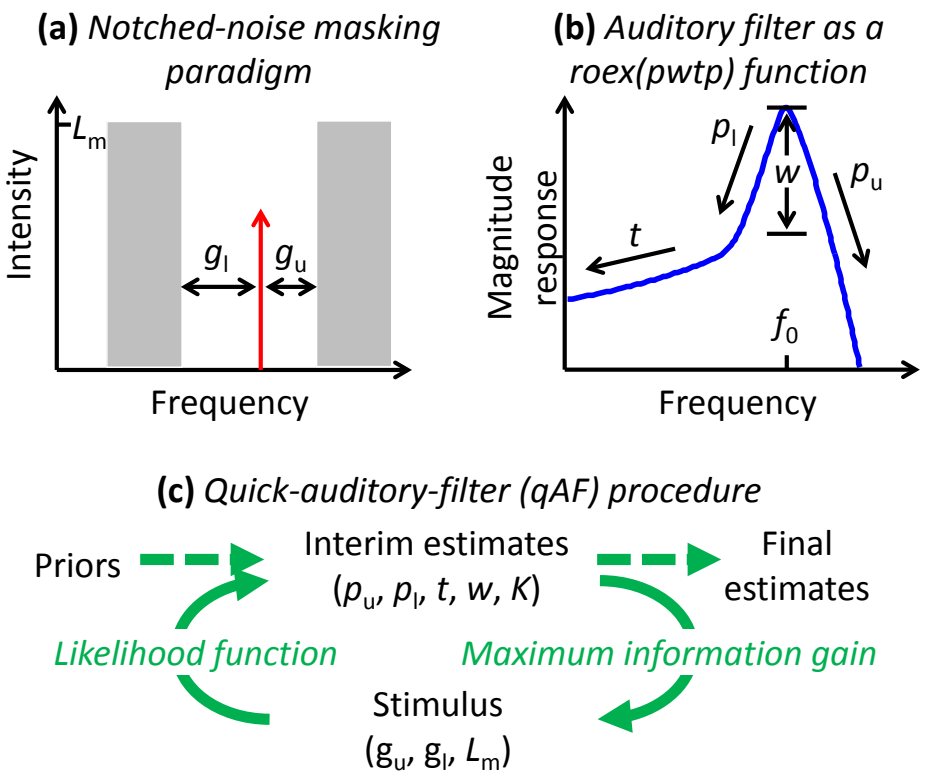

Figure 1: The qAF Procedure for the estimation of auditory-filter shapes. (a): Stimulus design in a notch-noise masking experiment showing the target tone as a red line and the upper and lower masker bands as two gray areas. (b): Formulation of the auditory-filter shape using the roex(*pwtp*) function. (c): Iterative computations in the qAF procedure. During each iteration, the five-dimensional posterior parameter distribution is updated based on the most recently collected response. The stimulus to be presented in the following trial is then determined by searching for the combination of the stimulus parameters that minimize the expected entropy of the posterior distribution. The final estimates correspond to the mean of the posterior distribution.

search for the stimulus parameters (g_u, g_l, and L_m) that maximized the potential information gain from the trial to be tested (Kontsevich and Tyler, 1999). Then, an experimental trial is run using the optimized stimulus and a response is collected from the subject. The posterior parameter distributions are then updated using an extended-Kalman filtering algorithm (Fahmeir, 1992). Each qAF track terminates when a preset number of trials is reached or when a user-specified termination criterion is met.

Compared to the traditional procedure, the qAF procedure is more efficient because all three stimulus parameters are manipulated adaptively according to previous responses collected rather than repeating threshold measurements for a series of preselected notchwidth combinations. It has been demonstrated that the qAF procedure can achieve satisfactory reliability using 200 trials for normal-hearing listeners who are naive to any psychoacoustical experiments (Shen et al., 2014).

3.2 Organization of the qAF Toolbox

The qAF Toolbox contains software routines that enable the efficient implementation of the qAF procedure in Matlab. To allow a flexible and intuitive organization of the experimental variables and the computational algorithms, the routines and relevant variables are organized into a `qAF` class. To create an experiment, the user creates an object in the `qAF` class (e.g., `qaf = qAF(par)`, see the following section for details). Each `qAF` object is a data structure that holds several *variables* and *methods*. The *variables* store parameters such as the auditory filter parameters, trial-by-trial data (e.g., signal strength and observer's responses), and internal parameters (e.g., the counter for the trial number). The *methods* conduct operations on the *variables*, including setting and returning the *variable* values, providing the Bayesian estimates of the auditory filter, and updating the signal strength. Tables 1 and 2 list the *variables* and *methods*, respectively. Figure. 2 illustrates how the *variables* and *methods* are organized in a `qAF` object and their role in the (1) Initialization and (2) Iteration phases of an experiment.

In the initialization phase of an experiment, a `qAF` object is constructed. The constructor `qaf = qAF(par)` sets the initial stimulus parameters `qaf.x0` (i.e. g_u, g_l, and L_m for the first trial) and initializes the prior parameter distribution (i.e. the initial guesses for the model parameters). In the iteration phase of the experiment, the *variable* `qaf.x` is a matrix that holds information regarding the stimulus parameters on each trial. The values of g_u, g_l, and L_m are arranged in the three columns in `qaf.x`, in that order; and the stimuli on various experimental trials correspond to rows in `qaf.x`. Following each trial, a new row is added to `qaf.x`. The *variables* `qaf.phi` and `qaf.P` hold the mean and covariance matrix of the posterior parameter distribution, which are updated following each trial. The *variable* `qaf.phi` is a matrix that holds the means of the posterior distributions following every trial, arranged in columns. The values in the ith column are considered as the interim parameter estimates following the ith trial. The *variabl* `qaf.P` is a three-dimensional matrix. The third dimension of the matrix is associated with different trials, while the first two dimensions are used to store the covariance matrix of the posterior parameter distribution following each trial. The values of the *variables* `qaf.x`, `qaf.phi`, and `qaf.P` are updated after each trial by the *method* `qaf.update(r)` according to the collected response `r` from that trial. As part of the updating process in `qaf.update(r)`, the current percent correct `qaf.PC` and the optimal

Table 1: List of the variables in the qAF Toolbox.

Variables	Comments
`qaf.par`	Data structure containing the configurations of the parameter space
`qaf.n`	Current trial number
`qaf.x`	Stimulus parameter (i.e. g_u, g_l, and L_m)
`qaf.xnext`	Next stimulus parameter based on previous trials
`qaf.phi`	Mean of the posterior parameter distribution
`qaf.P`	Covariance matrix of the posterior parameter distribution
`qaf.r`	Listeners binary responses in terms of correctness
`qaf.PC`	Percent correct
`qaf.phi0`	"True" parameter values for a virtual listener used for the purpose of simulations

stimulus to be used on the next trial `qaf.xnext` are also calculated.

4 HOW TO USE THE qAF TOOLBOX

4.1 Configuring an Experiment

Before creating a `"qAF"`track, relevant parameters of the experiment must be set. The routines included in the qAF Toolbox assume a specific data structure for these parameters. Users should follow the example given in the qAF package and modify the `exp_config.m` function. This function constructs a structure `par` that includes experiment configurations, initial status of the adaptive track, and the prior parameter distributions.

To modifiy `exp_config.m`, first set the model name in the `par.model` parameter. Currently, only the roex model is supported.

```
par.model = 'roex';
```

The parameter `npar` indicates the number of parameters used to describe the auditory filter. This is one of the input arguments passed to the `exp_config.m` function when it is called. Accepted values are 3, 4, and 5. If `npar` $= 5$, then parameters p_u, p_l, t, w, and K are estimated. If `npar` $= 4$, then parameters p_u, p_l, w, and K are estimated, and parameter t is set to a value of 9. If `npar` $= 3$, then parameters p_l, w, and K are used, parameters p_u and p_l are assumed to be equal, and t is set to a value of 9.

```
par.npar = npar;
```

The parameter γ refers to the chance proportion correct for the experiment. A γ value of 0.5 is appropriate for a two-alternative forced-choice (2AFC) design. In case of a 3AFC design, the γ parameter should be set to 1/3, etc.

Table 2: List of the methods in the qAF Toolbox.

Methods	Comments
Initialization:	
`qaf = qAF(par)`	Constructs a qAF object.
Iteration:	
`qaf.update(r)`	Updates posterior and xnext based on the previous posterior and the new response `r`.
Simulation:	
`qaf.setPhi0(phi0)`	Sets the parameters in the psychometric function of the virtual observer for simulation.
`r = qaf.simulateResponse(x)`	Returns a response from the virtual observer.
Other:	
`qaf.plotPhi()`	Plots the parameter estimates.
`qaf.plotP()`	Plots the posterior parameter distribution.
`qaf.plotAF()`	Plots the auditory filter.
`qaf.plotStim()`	Plots the stimulus.

```
par.pf_par.gamma = 1/3;
```

The parameter β refers to the assumed slope of the auditory filter. The choice of assumed β value only influences the result to a very limited degree (Shen and Richards, 2013).

```
par.pf_par.beta = 1;
```

The parameter `stim_par.fo` is the stimulus signal frequency. This is one of the input arguments passed to the `exp_config.m` function when it is called.

```
par.stim_par.fo = fo;
```

The parameter `stim_par.siglev` indicates the stimulus signal level in dB SPL. This is one of the input arguments passed to the `exp_config.m` function when it is called.

```
par.stim_par.siglev = siglev_spl;
```

The parameter `stim_par.BW` is the bandwidth of the masker, measured in hertz. In the following example, the bandwidth is set to one quarter of the stimulus signal frequency.

```
par.stim_par.BW = 0.25*par.stim_par.fo
```

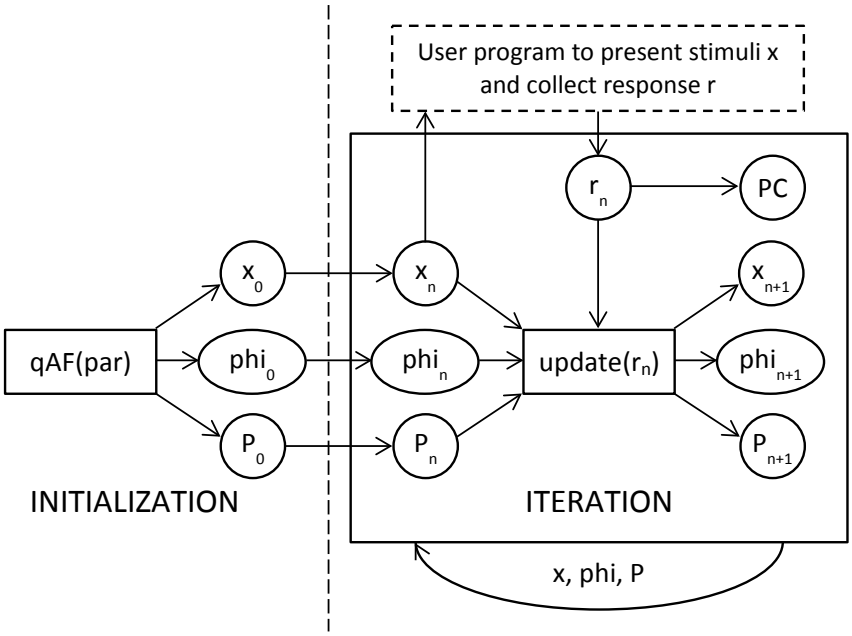

Figure 2: Schematic of an experiment constructed using the qAF Toolbox.

The parameter `x_lim` describes the masker. The three columns in this parameter are the upper notchwidth g_u, the lower notchwidth g_l, and the masker spectrum level L_m. The two rows in this parameter describe the range boundaries of g_u, g_l, and L_m. The parameter `par.x_n` represents the number of points along the three axes.

```
par.x_lim = [0 0 -10;0.4 0.6 55];
par.x_n = [3 13 20];
```

In the above example, the potential values of g_u are three linearly spaced values between 0 and 0.4; the potential values of g_l are 13 linearly spaced values between 0 and 0.6; and the potential values of L_m are 20 linearly space values between -10 and 55 dB SPL.

The prior parameter distribution is established by seting two parameters, the parameters `par.prior_phi` and `par.prior_P`, corresponding to the mean and the covariance matrix of the prior distribution, respectively. The parameter `par.prior_phi` has to be a column vector with a size of `npar` by 1, while the parameter `par.prior_P` has to be a diagonal matrix with a size of `npar` by `npar`.

```
par.prior_phi = [40; 40; 5; -30; 5];
par.prior_P = diag([40 40 5 40 20].^2)
```

In the above example, the prior distribution is set for a five-parameter model. The prior distribution for each parameter takes the form of a Gaussian distribution. The prior means for p_u, p_l, t, $10 \log w$, and $10 \log K$ are 40, 40, 5, -30, and 5, respectively. The prior standard deviations for p_u, p_l, t, $10 \log w$, and $10 \log K$ are 40, 40, 5, 40, and 20, respectively.

4.2 Running an Experiment

Once the configuration for the experiment has been established in the `exp_config.m` file, three steps are needed to carry out the experiment. An example of setting up the experiment is provided in `template.m` and copied below.

```
% step 1, configure the experiment parameters
configpar = exp_config_roex(5, 50, 2000);

% step 2, create a qAF object
qaf = qAF(configpar);

% step 3, create an experimental loop (one trial per loop), and update
% signal level using the update method
N = 80; % total number of trials to run
for i = 1:N

    %-------------------------------------
    % For an actual psychophysical experiment, a user-defined function is
    % needed here, in the format of:
    %
    % r = UserFunction(qaf.xnext);
    %
    % It takes the stimulus parameter as the input argument and returns,
    % presents the stimulus, obtains the response, and return the response
    % correctness as the output argument.
    %-------------------------------------

    % update the model parameters
    qaf.update(r);

end
```

First, the data structure `configpar` is created by the function `exp_config_roex()`. In this example, a five-parameter model is used and the signal level and frequency are set to 50 dB SPL and 2000 Hz, respectively. Second, a qAF object is constructed. Finally, stimuli are presented to the test subject and responses are collected iteratively using a loop structure. Stimulus generation, presentation, and recording the subject's responses are acheived through a user-defined function appropriate for research of interest. Within each loop, the *method* `qaf.update(r)` updates the signal strength `x` according to the response, `r`, on that trial. The *method* `x = qaf.update(r)` also updates the *variables* held within the qAF object, including the stimulus parameters `qaf.x` and response `qaf.r`, the posterior parameter distribution `qaf.phi` and `qaf.P`, the percent correct `qaf.PC`, and the signal level to be used for the stimulus presentation on the next trial `qaf.xnext`. These *variables* are accessible both within the experimental loop (for online data peeking) and after the experiment is finished (for data storage).

4.3 Simulating Experiments

Using the qAF toolbox, simulations are easy to carry out. For simulations a virtual observer provides responses to the stimuli in accordance with a "true" set of auditory-filter parameters `phi0`. Simulations are useful, for example, to study the effects of different prior distributions on the convergence of the parameter estimates, determine the appropriate ranges for the parameters, etc. Below is an example (provided in the file `example_qAF_roex.m`).

```
% initialize the qAF procedure using the five-parameter roex model with
% a signal level of 50 dB SPL and a signal frequency of 2000 Hz.
configpar = exp_config_roex(5, 50, 2000);
qaf = qAF(configpar);
% setting the true parameter for a virtual observer
phi0 = [44.7281   45.8076    8.0470  -41.3365   10.8181];
qaf.setPhi0(phi0);
ntrials = 200; % total number of trials to run

for i = 1:ntrials
    % simulating a response from the virtual observer
    r = qaf.simulateResponse(qaf.xnext);
    % Update model parameters
    qaf.update(r);
end

% Plot the stimuli sampled and the auditory filter estimated.
figure(1);
qaf.plotStim();
% Plot the parameter estimates as functions of trial number.
figure(2);
qaf.plotPhi();
% Plot the posterior distribution
figure(3);
qaf.plotP();
```

5 ACKNOWLEDGMENTS AND REFERENCES

This work was supported by NIH Grant No. R21 DC013406 (co-PIs: V.M.R. and Y.S.). Individuals contributing to the qAF Toolbox and this user manual include Yi Shen, Rajeswari Sivakumar, Virginia Richards, and Eric Brown.

References

Fahmeir, L. (1992). Posterior mode estimation by extended kalman filtering for multivariate dynamic generalised linear models. *J. Amer. Statist. Assoc*, 87:501–509.

Kontsevich, L. L. and Tyler, C. W. (1999). Bayesian adaptive estimation of psychometric slope and threshold. *Vision Res*, 39(16):2729–2737.

Oxenham, A. J. and Shera, C. A. (2003). Estimates of human cochlear tuning at low levels using forward and simultaneous masking. *Journal of the Association for Research in Otolaryngology*, 4(4):541–554.

Patterson, R. D. (1976). Auditory filter shapes derived with noise stimuli. *J Acoust Soc Am*, 59(3):640–654.

Patterson, R. D., Nimmo-Smith, I., Weber, D. L., and Milroy, R. (1982). The deterioration of hearing with age: frequency selectivity, the critical ratio, the audiogram, and speech threshold. *J Acoust Soc Am*, 72(6):1788–1803.

Rosen, S., Baker, R. J., and Darling, A. (1998). Auditory filter nonlinearity at 2 khz in normal hearing listeners. *J Acoust Soc Am*, 103(5 Pt 1):2539–2550.

Shen, Y. and Richards, V. M. (2013). Bayesian adaptive estimation of the auditory filter. *J Acoust Soc Am*, 134(2):1134–1145.

Shen, Y., Sivakumar, R., and Richards, V. M. (2014). Rapid estimation of high-parameter auditory-filter shapes. *J Acoust Soc Am*, 136(4):1857.