
# Final Report: International Association of Oil & Gas Producers (OGP) Contract 07-18

**Mitigation & Monitoring: Passive Acoustic Monitoring (PAM) Software Development – Detection, Classification and Localisation Capabilities**

**Author:**

Joe Hood
President
Akoostix Inc. of Nova Scotia
10 Akerley Blvd, Suite 12
Dartmouth NS B3B 1J4
Phone: (902) 223-9876
Fax: (902) 405-3855

14 April 2009

**Submitted to (if required):**

John Campbell
OGP Technical Director
International Association of Oil and Gas Producers
209-215 Blackfriars Road, London
United Kingdom, SEI 8NL
Phone: 011-44-776-823-2943

**Document Number: 2008-004**

**Document Version: 1.1**

# Document Distribution and History

The following tables identify the copy distribution, document version(s), and all associated changes.

| DOCUMENT DISTRIBUTION | |
|---|---|
| Copies | Provided To |
| 1 | John Campbell |
| 1 | David Hedgeland |
| 1 | Akoostix Inc, Contract File # 07008 |
| | |
| | |
| | |
| | |

| DOCUMENT HISTORY | | | |
|---|---|---|---|
| Document Version | Date | Section | Description of Change |
| 0.9 | 19 Dec 08 | ALL | First Draft (Sections 2 & 3) |
| 1.0 | 31 Dec 08 | ALL | Original |
| 1.1 | 14 Apr 09 | 1, 4.4, 5 | Added executive summary, improved naming in detector algorithm documentation, and added more detail to performance testing from the draft paper. |
| | | | |
| | | | |
| | | | |
| | | | |

# Document Approval

| AUTHOR | | |
|---|---|---|
| Date | Name | Signature |
| | Joe Hood | |

| ENGINEERING REVIEW | | |
|---|---|---|
| Date | Name | Signature |
| | Jason McInnis | |

| CORPORATE REVIEW | | |
|---|---|---|
| Date | Name | Signature |
| | Joe Hood, President | |

# Table of Contents

# List of Figures

# List of Tables

# 1 Executive Summary

## 1.1 Project Overview

Akoostix entered into a contract with the JIP in December of 2007 to develop a PAM detection software module for PAMGUARD. The primary objective of the module was to address deficiencies in mysticetes detection using marine mammal detection experience gained through projects for Defence Research and Development Canada (DRDC) as the foundation. The project was completed in December of 2008. This report documents the contracted work.

The work was conducted in two phases. During the first phase the original algorithm was improved and the detection performance was examined using available data. In the second phase a custom PAMGUARD module was designed, developed, integrated, and tested. Beyond development of a PAMGUARD detection algorithm, Akoostix also focused on applying standard user interface design and software engineering practices to ensure the usability of the module.

## 1.2 Detection Algorithm Overview

The algorithm is a generic implementation of a standard likelihood ratio test (LRT). Incoming signals are converted to the frequency domain and a band-limited average energy is computed. An estimate of the probability of signal vs. noise being present is computed using various estimation algorithms. Two estimation methods were implemented. The first is based on an exponentially decaying average, while the second is based on a split-window block average. The module includes a secondary LRT that compares the signal estimate of in-band and out-of-band signals to assist in false alarm rejection.

Once configured, the LRT supports detection of the specified signals and rejection of:

- Out-of-band signals
- Signals that are both in-band and out-of-band
- Signals that are not the correct duration

The algorithm is not capable of examining the signal structure within the specified time-frequency limits. This is left for classification, performed downstream. Regardless, the detector provides a low processing load filter for large quantities of data, reducing the load on later processing stages.

## 1.3 Software Module Overview

The software module was designed using an object-oriented approach, ensuring that it provided a generic detector implementation. Other responsibilities such as detailed classification and localization were not coupled to the module to support reuse and modularity for a variety of sensor types and localization / classification approaches. The module input will accept any number of data channels that PAMGUARD

supports with arbitrary sample rate. The module output provides a fully populated PAMDetection object that can be used by other modules to fulfill their responsibility.

The module configuration user interface is provided in a single dialogue, using easy to understand parameters and units. The software handles creation or connection of upstream modules such as spectra generation. The interface performs bounds checking on all inputs, ensuring that a valid module can be created. The user is cued to errors and warnings using yellow and red color highlights, while valid bounds are provided by tooltips.

The software was rigorously tested using a structured test plan and predefined requirements. The interfaces and design are fully documented in the module's JavaDoc.

## 1.4 Significance

The split-window block average estimation method, developed during this project, is new to PAMGUARD, while the other estimation method is similar in some aspects to existing detectors. Akoostix was able to show that the block average estimation method is superior to others in rejecting signals that are too long in duration, while maintaining the same processing load.

The careful design of a modular component ensures that other well-designed modules will be able to connect to the detector, providing a complete system and the flexibility required to configure PAMGUARD for a wide variety of hardware and scenarios. Existing coupled streams such as the click detector, could be refactored to allow for the optional use the likelihood detector as the detection stage.

The user-friendly configuration interface reduces training requirements and user frustration, while the structured design and testing increases reliability. The detector highlights detection on the spectrogram, cueing PAM operators to potential contact.

The module is suitable for detection of a wide variety of species and vocalization types. It is ideal for short to medium duration (μs to several seconds) signals and any signal bandwidth. The exact structure of the vocalization does not have to be specified, but also cannot be exploited to improve performance. It is too difficult to specify a complete set of signals or species that could be detected, as there are no practical limits on the frequency, or bandwidth of signals. The algorithm has been used successfully during trials to detect a very wide range of broadband clicks including many beaked whales, dolphins, porpoises, and Sperm whales ranging from hundreds to many thousands of Hertz. It has also been tested for, or used during trials to detect a wide variety of moans, grunts, song, etc. This includes successful detection of Right whale, Humpback, and Bowhead vocalizations. Though it has also detected whistles, other detectors will provide better performance. This is mainly due to the relatively narrow bandwidth, but wide fluctuation in frequency of whistles that reduces the average energy of a band-limited signal. Other calls of this type would also be hard to detect using the current estimators, though a higher order estimator is likely to perform much better.

# 2 Overview

## 2.1 Introduction

This final report describes the work conducted under the International Oil & Gas Producers' (OGP) contract 07-18, titled "Mitigation & Monitoring: Passive Acoustic Monitoring (PAM) Software Development – Detection, Classification and Localisation Capabilities" [1]. The project was executed from 6 December 2007 through 31 December 2008.

Although the contract title mentions classification and localization, the focus of this project was *detection*. This project objective is formalized in the following statement from the contract, "The primary deliverable of this project will be a PAM software detection module that addresses deficiencies in mysticetes detection, while providing new options for beaked whale detection."

## 2.2 Related Effort

Prior to this contract Akoostix developed PAM software for Defence Research and Development Canada (DRDC). The lessons learned from this work, and synergistic projects at DRDC executed during this project were used to increase the efficiency and effectiveness of this work.

The same algorithm that is used for PAM at DRDC is provided in PAMGUARD. It was developed using open literature and Akoostix innovation, allowing it to be provided without restriction. The software design and interface is custom to, and tailored for, PAMGUARD.

The Acoustic Cetacean Detection Capability (ACDC) software and the Software Tools for Analysis and Research (STAR) were licensed from DRDC and used to perform the algorithm development and validation work. These tools allowed a more detailed analysis of individual detections and intermediate detector results that are not, even now, possible using the PAMGUARD module. Both ACDC and STAR continued development under DRDC funding throughout the project. Some of these enhancements enabled improved data analysis under this project.

DRDC executed a project [3] with the objective of developing automated classification of marine mammal detections during 2008. This project used the same data sets as for the subject PAMGUARD project. The manual classification results produced under the DRDC project were used to support validation by providing more detailed truth data.

## 2.3 Project Overview

The work was conducted using a two-phased approach, dividing the effort into algorithm development and PAMGUARD implementation. Algorithm development focused on requirements analysis, algorithm prototyping, algorithm testing, and software design. Implementation focused on the actual software development, testing,

and documentation. There was considerable schedule overlap between the two phases, as the Joint Industry Programme (JIP) decided to proceed with phase 2, regardless of the outcome of phase 1.

All deliverables for this project were provided as a package and included the following items. These have been organized in three groups: Contract Documentation, Developer Documentation, and User Documentation. The groups are selected to map to the contractual requirements for documentation. Many of these deliverables are also referenced within the report, giving further information about their development. Many of the deliverables are provided as a snapshot at the time of delivery. Where appropriate the source of the most current version is indicated in brackets.

1. READ_ME_PAMGUARD_Detector, Version 1.0 – A typical READ ME file, included with and describing the contents of the deliverable package [5].
2. Contract Documentation
    a. Likelihood Detector Final Report.pdf - The primary deliverable for the project, this report, describing the project's execution with references to all of the other deliverables.
    b. Likelihood Detector - PAMGUARD Module Acceptance Report.pdf - A report generated by the PAMGUARD Guardians with assistance from Akoostix [6]. It is used to demonstrate that all required deliverables have been provided to the PAMGUARD guardians and provide feedback on their perceived quality.
    c. Improved Passive Band-limited Energy Detection for Marine Mammals.pdf - A draft journal article, based on this project, describing the likelihood detector [7]. The article is in a format compatible with the Institute of Electrical and Electronics Engineers (IEEE) Journal of Oceanic Engineering. As required by the contract, a proposal for its publication is provided in Section 2.4.
3. Developer Documentation
    a. Likelihood Detector - Acceptance Test Plan.zip (available from PAMGUARD guardians) [8]. Not initially required as a deliverable, this contains the acceptance test plan (ATP) which is used to validate the expected functional operation of the likelihood detection module within PAMGUARD. The tests are also provided as baseline manual and built-in regression tests (described in the document) that can be used to test future improvements or modifications to the detection software. The zip folder also contains the signed document approval page and the file "test_sweeps_2008.12.10-12.00.00.wav" which is the data needed to run the tests.
    b. likelihood-javadoc.zip (available from PAMGUARD Sourceforge) - Contains the PAMGUARD API documentation for the likelihood detector produced from Javadoc compatible in-line documentation. This documentation would normally be produced directly from the source code, but is provided herein for convenience. The format is in keeping with the current Javadoc that already existed within PAMGUARD [11]. Implementation code also contains normal Java style comments to clarify

the intent of specific code segments [13]. Once uncompressed, the *javadoc/index.html* file can be selected to allow viewing of the documentation from an Internet browser.

c. Likelihood Detector - akoostix_mt_src.zip (available from PAMGUARD Sourceforge) - A compressed version of the latest checkout from the Akoostix_MT branch of the likelihood detector software [13]. Note: this is not a Java Archive (JAR) file and must be run from Eclipse as described in the ATP [8]. Production of a JAR file is the responsibility of the PAMGUARD Guardians and is expected in early 2009 [6].

4. User Documentation

a. Likelihood Detector - User Tutorial 2008 - Akoostix.pdf (available from PAMGUARD) - The Likelihood Detector has been included as "Exercise 3. Likelihood Detector" in the document which was derived from version 1.1.01 of the tutorial [9]. This tutorial is a basic walk through starting the first time an operator opens PAMGUARD and progressing to observation of detections on the spectrogram display. It includes a tutorial on setting up a target configuration and executing the detection processing. It encompasses any configuration of other modules required such as the sound acquisition and spectrogram displays. The tutorial is meant to be a quick start guide for users first trying to get familiar with the Likelihood Detection module.

b. likelihood-detector-docs.zip (available from PAMGUARD Sourceforge) - Contains the PAMGUARD online user documentation for the likelihood detector [12]. This documentation would normally be produced directly from the source code, but is provided herein for convenience. Once uncompressed, the *docs/LikelihoodDetector_Introduction.html* file can be selected to allow viewing of the documentation from an Internet browser.

c. Likelihood Detector - Configuration Files.zip (available from PAMGUARD) [10] - These files are provided as examples of how to configure the system and could be used as a basis for creating custom configurations and .psf files. The configurations are identical to those used to produce the results in the referenced papers [7,15]. A separate "README" file is included within this zip folder that further explains the contents.

## 2.4 Journal Article Publication – Proposal

As required by the contract, Akoostix proposes to publish the provided draft journal article in the IEEE Journal of Oceanic Engineering. This journal may be a more appropriate publication, as the focus of the paper is on practical algorithm issues more so than rigorous development of an optimal detector for a specific case. The latter would be more appropriate for a journal such as the Journal of the Acoustical Society of America (JASA).

The cost of publication is $110 per printed page, which we propose be covered by the JIP. The current color figures will be upgraded to conform to the journal standard and converted to black and white prior to submission. This will avoid additional charges

for color printing. Provided no detailed data reanalysis or changes are required by the reviewers, Akoostix proposes to bear the cost of any additional internal effort required to effect publication. This effort will be limited to twenty (20) hours. Any effort in addition to that would be negotiated with the OGP JIP prior to commencing work.

Should this proposal not meet the requirements of the OGP JIP, we are prepared to collaborate on a mutually acceptable method of publication.

## 2.5 Report Outline

This report contains three more sections:

- Section 3: Contract Execution – describes the actual contract execution
- Section 4: Likelihood Detector – describes the likelihood detector algorithm in detail, providing synthetic data examples, and configuration recommendations.
- Section 5: Performance Testing – provides an overview of performance testing results.

# 3 Contract Execution

This section provides a detailed description of how Akoostix executed the subject contract. The project was executed in two separate phases. The first phase was planned as an algorithm development phase that concentrated on designing and prototyping the likelihood detection algorithm and determining the detailed requirements. The second phase was planned as a software development phase that focused effort on implementing the algorithm and integrating it as a plug-in to the PAMGUARD application framework. This included developing configuration dialogs and integrating the detection output to PAMGUARD's database and spectrogram overlays.

## 3.1 Requirements (Phase 1)

Requirements analysis for the first phase concentrated on defining the required scenarios, initial parameters, and detection performance characteristics for the likelihood detector algorithm. Likelihood detection had been used by Akoostix in other software for transient detection purposes and much of the requirements analysis of this phase concentrated on taking the knowledge gained through previous implementation of this technique and improving it. The knowledge gained through this process was used in the software development phase to build the likelihood detector from the ground up to best suit PAMGUARD's marine mammal detection requirements.

Several requirements analysis tasks were performed during the first phase including:

- Characterization of marine mammal data: Marine mammal data was analyzed by Akoostix to determine the characteristics of the target species and associated noise.
- Algorithm development and prototyping: The algorithm was designed and prototyped using a rapid development process in order to confirm that the key concepts of the algorithm would work prior to full integration into the PAMGUARD application. See Section 3.1.1 for more information.
- Validation of algorithms: Synthetic and known real-world data was used to measure the detection performance of the algorithm and used to tweak the detection algorithm parameters for the target species. See Section 3.1.2 for more information.

The software requirements analysis took place half way through phase two. The framework for the detector was developed prior to the second requirements analysis since very little from this analysis would affect the integration of the module in the system in terms of inputs and outputs that already existed in PAMGUARD. The second requirement analysis focused on the usability features of the detector and how it would fit into the overall system. An initial interview took place between Akoostix and Doug Gillespie on July 31st, 2008, and was documented in the formal meeting minutes [2]. From this interview Akoostix further broke down the requirements into a requirements specification that was made available to the JIP on the Akoostix intranet portal and is duplicated in the formal Acceptance Test Report (ATR) where they are

mapped to individual test cases that validate each requirement [6]. During the four software development iterations (including system integration and testing) the software requirements were refined as more was understood about PAMGUARD and any constraints that were placed on the Likelihood Detector module by the plug-in framework provided by PAMGUARD.

### 3.1.1 Algorithm Development and Prototyping

The algorithm development portion of the contract proceeded as proposed. At the JIP kick-off meeting it was decided that Mobysound mysticetes data would be used as the primary data for algorithm development and performance analysis. This data is freely available for research purposes and would allow other researchers to duplicate and compare our results to other algorithms. Humpback, Bowhead and Right whale species were selected along with noise data. A secondary data set from DRDC was used to add Sperm whale and more noise data.

As proposed, the DRDC implementation of the algorithm was used to support algorithm development. This software provides detailed detection logs and tools for browsing the data to determine classification and produce statistical results. At the start of the contract the software contained the decaying average estimator and guard bands. The block average estimator was added for comparative performance analysis, producing excellent results as documented in Sections 4 and 5. Once preliminary testing was complete, the algorithm was passed to the software developers for implementation in PAMGUARD while a more thorough validation was conducted with the purpose of producing data for a scientific paper. At this stage, testing and validation was performed ad hoc to quickly identify algorithm issues and propose solutions. This ad hoc testing also helped to define a synthetic data set used for the validation task.

The algorithm is completely documented in Section 4 including recommendations for configuration. A high level description and the configuration recommendations are included in the PAMGUARD online help.

### 3.1.2 Validation

A more rigorous validation of the proposed algorithms was required to provide the data required for a scientific paper [7]. Validation was conducted using both synthetic and real data. Synthetic data provided structured signals that could be used to test and validate conceptual issues in real data, while the real data provided realistic examples for more thorough validation.

The synthetic data was synthesized based on real-world issues encountered during algorithm development, and consisted of:

- An isolated short duration pulse that emulated a signal of interest
- An isolated long duration pulse that emulated a common false alarm
- A pulse train that emulated a click train
- A rapid pulse train that emulated a false alarm case that we felt could be mitigated

- Random noise

As mentioned, the real data came from Mobysound and DRDC. A request was made to the JIP for seismic data to further test for false alarms, but none was made available.

The results of validation were presented at the OGP JIP review in Houston, Texas on 30 Oct 08 [4], and are further detailed in a draft journal article [7] and Section 5.

## 3.2 Design

A lot of the design motivation and details are provided in the Javadoc documentation provided with the Likelihood Detector module. The Javadoc discusses details about interaction and integration of the detector module. Section 4 in this report discusses the design of the algorithm. The Javadoc documentation can be built using the Eclipse IDE using the *Project->Generate Javadoc...*. A standard output directory for the documents will be *pamsoft2006/doc/index.html*. Also, the Likelihood Detector Parameters configuration (Figure 1) dialog provides built-in help and discusses most of the graphical user interface interactions and standard use cases. The following is a high-level overview of the design and will avoid repeating the details provided by the existing documentation.

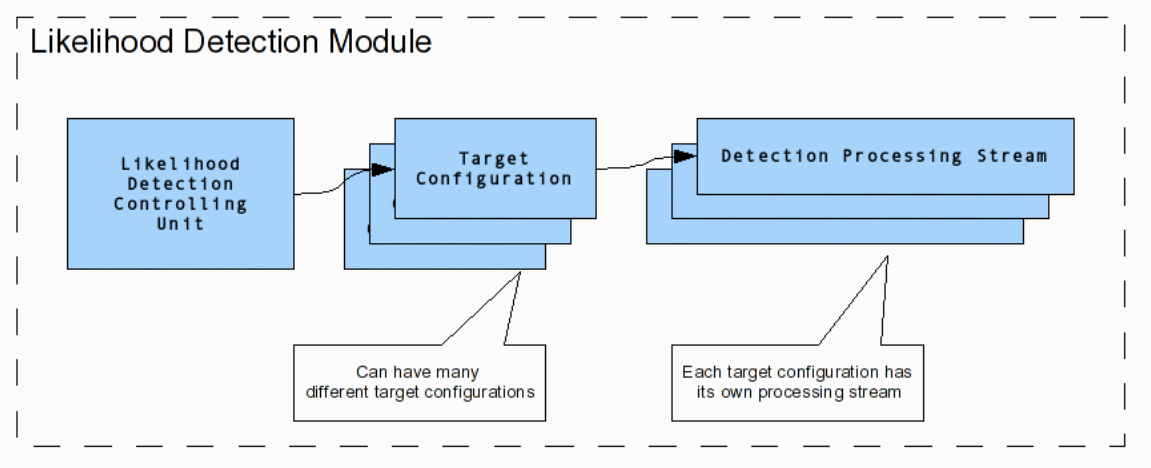


**Figure 1: High-level configuration**

The Likelihood Detection Controlling Unit implements the standard *PamControlledUnit* functionality required by PAMGUARD for a plug-in module. The controlling unit contains a process group that is used to manage one or more sets of target configuration and detection processing pairs.

The Target Configuration represents a single configuration for the likelihood detector controlling unit, which can run multiple configurations simultaneously. A configuration is represented by a number of parameters that control the scope and accuracy of the area and the characteristics to be detected. Also there are one or more signal band definitions and zero or more guard band definitions contained in each target configuration. The target configuration converts these desired operator

parameters into a set of signal processing parameters for the detection processing stream. The controlling unit uses the parameters of the target configuration that contains the parameters that were entered and validated by the user in the front-end configuration dialog and creates a corresponding processing stream. Once the processing stream is no longer required, the controlling unit disconnects any data sources as well as frees any reference to the processing stream so that garbage collection can eventually recover the resources.

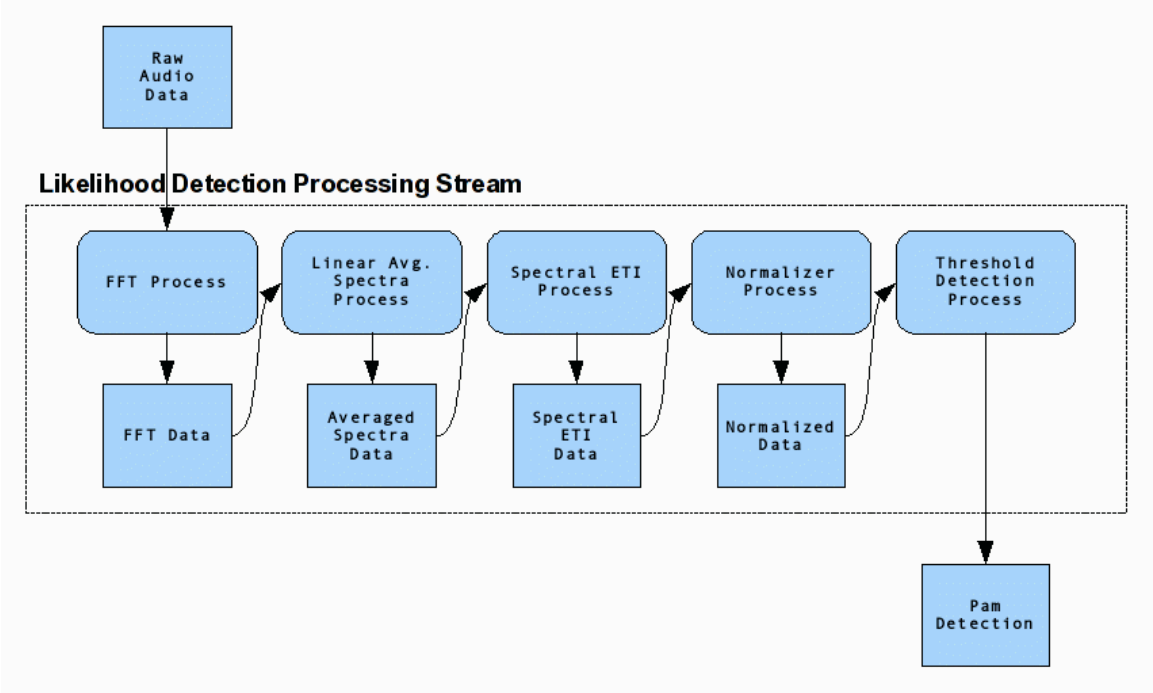


**Figure 2: The Likelihood Detection Processing Stream Design**

A Likelihood Detection Processing Stream consists of a pipeline of individual signal processing modules. Each processing module extends the PamProcess interface. This pipeline is illustrated in Figure 2.

Detections are generated at the end of the processing chain when they occur. This detection is used to generate an overlay on the Spectrogram over the user-defined frequency range defined by the detection band and the duration of the detection (a rectangle). The detection is also logged to the PamDatabase and could be subscribed to by other modules such as Ishmael localization. The fields used in the PamDetection were discussed with the PAMGUARD development team and agreed on. For information on the different detection fields refer to the Javadoc.

## 3.3 Implementation

The implementation of the software occurred over four planned development iterations:

- Plug-in Framework: Developed the skeleton of the Likelihood Detection Controlling Unit along with the first process within that module, Fast Fourier Transform (FFT) Process.
- Detection Processing Development: Included developing the signal processing modules that where unlikely to change prior to detailed requirements analysis including Linear Average Spectra Process, Spectral Energy Time Indicator (ETI) Process, and the some of the Normalizer Process. The baseline Target Configuration module was also developed during this iteration.
- Detector Development and Integration: Included finalizing the detector implementation by finishing the Normalizer Process and Threshold Detection Process. At this point, requirements analysis was finalized and the Target Configuration module as well as the graphical user interface dialog was completed.
- System Integration and Testing: See Section 3.4

Developing the software in this order allowed portions of the algorithm design tasks (first phase) to run concurrently with the software development without risking change to the software in the final stages of the first phase. Akoostix had been directed to proceed with software development before completing validation, as it was clear that the algorithm would be useful on some level.

### 3.3.1 Detection Algorithm

The detection algorithm is described in Section 4. The detection algorithm was built a module at a time in the order shown in Figure 2. Each module was developed independent of the PAMGUARD system. Unit tests were developed for each processing step to ensure that the signal processing performed as expected. This included mathematical tests. During the System Integration and Testing phase the controlling unit integrated the responsibility for the signal processing chain as well as passing the correct parameters to configure the processing. Integration went smoothly and was used less effort than the integration of the tree view in the graphical user interface, discussed later.

### 3.3.2 Graphical User Interface

The Graphic User Interface (GUI) was developed to provide a simple integrated window for detector configuration. This includes creating the FFT process, though the data source must be configured externally. This section describes the GUI at a high level. A detailed description of its use is found in the online help provided with PAMGUARD.

At a high level, the input device and channels are selected at the top of the dialogue and one or many configurations can be created within the *tree view* window. These configurations can also be separately exported and imported to facilitate transfer to other users. The configuration options themselves are in standard units and are not dependent on the input data source sampling rate or sensitivity, except that the entire frequency range for a particular band must be available for that band to be enabled.

Appropriate default values are populated as configurations are added and parameter range checking is enforced. Hovering over a box will provide the legal range limits as a tool-tip. If a parameter is entered that is out of bounds, the box will turn red. These checks ensure that the module configuration will not cause a software exception, but do not guarantee that the configuration is appropriate for the target species. The default values were selected to promote effective configuration.

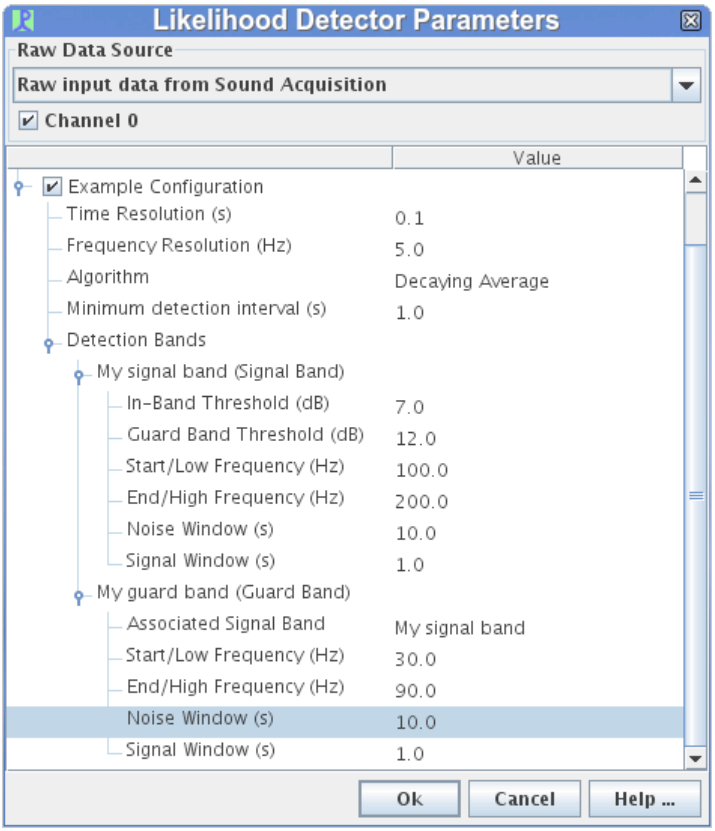


**Figure 3: A Likelihood Configuration with one Signal Band and one Guard Band**

The user interface (UI) used for PamGuard is based on the Java Foundation Classes' standard GUI Application Program Interface (API) called Swing. Swing provides a series of standard graphical widgets for building user interfaces, and its design revolves around the Model-View-Controller pattern. Simple user interfaces are easily possible in Swing, however the design of the Swing APIs make more complex interaces such as those found in scientific applications very difficult. A few of the issues encountered while using Swing to create the configuration interface for the Likelihood Detection module are described below.

Swing doesn't support many modern, complex widget components, such as the tree-based table (i.e., multi-column tree) used in the configuration dialog. While Swing provides the ability to custom-build your own widgets, this is a time-consuming process with a steep learning curve of the internals of the library.

The user interface components use a hodge-podge of methods to handle user events and callbacks. In order to handle all of the standard user-generated events from a widget, such as editing, validation, resizing, etc., it is necessary to navigate an unintuitive series of direct callbacks on widgets, immediate system events, and asynchronous system events. Often it was either impractical or impossible to manage events in a user-intuitive manner.

The model-view-controller paradigm that forms the basis of the Swing library is imposed upon the developer whether it is required or not. It is not possible to create an interface that has a direct relationship with its data. The ramifications of this are that the developer must create abstractions and code that isn't required, and often duplicate functionality in order to produce a working model from a set of existing data. For example, while the configuration dialog doesn't require a complex Model View Controller (MVC) setup to change simple values in a dedicated structure, a tree-like pseudo-database was created to act as a model for the tree-table because the Swing widget can not operate on anything else.

Issues related to development, testing, and debugging of the GUI were the primary cause of schedule delays related to the software. For future scalability, there are other open-source friendly toolkits that can be considered and that provide for a cheaper and more effective development process. For example, the Standard Widget Toolkit, developed by International Business Machines (IBM), or Qt Jambi, developed by TrollTech.

## 3.4 Testing

Akoostix elected to use two levels of regression tests to demonstrate the required functionality and to permit future developers to quickly assess the health of the Likelihood Detector module.

First, unit testing was performed on the critical signal processing components of the system. Section 3.4.1 further describes this effort.

Integration tests were developed later, during the System Integration and Testing phase of the project, as a last step towards validating and delivering the software. Section 3.4.2 further describes this effort.

Other testing included direct comparison of output with prior implementations of the algorithm (DRDC software logs) and use of the debugger to trace GUI parameters and ensure that they were properly passed to the required processing modules. The DRDC software is not provided as a deliverable and manual debugging is time consuming, so they are not proposed as standard regression tests. The debugger could be used if a specific symptom or problem were observed.

A complete description of the tests performed for this contract is provided in the ATP [6].

### 3.4.1 Unit Tests

Unit tests were integrated into the Likelihood Detection module directly and are designed to be reused to perform regression tests. These test require no manual input

from a developer and pass back a simple PASS / FAIL on execution. They can provide some confidence that future maintenance or enhancement of the software did not change the fundamental behavior of the detector or its performance. Descriptions of the unit tests are provided in the ATP [6]. It is highly recommended that if any future development occurs in the signal processing modules that these tests are updated to maintain their usefulness.

### 3.4.2 Integration Tests

Integration tests were performed to test the end-to-end functionality of the detection module. These tests require a quality assurance person or software developer to run them manually following the steps described in the ATP [6]. The manual tests validate correct system integration via user interaction with the system by providing expected results given a controlled environment with a known set of test data.

## 3.5 Documentation

A moderate level of documentation was generated for the Likelihood Detector module. Each documentation type is designed for a particular stakeholder in the PAMGUARD software or OGP JIP. This documentation is included by reference and was delivered separately as direct inclusion in the final report would be cumbersome and not serve any purpose. This also applies to the source code.

A complete list - and brief descriptions of each item - has been provided in Section 2.3.

# 4 Likelihood Detector

## 4.1 Introduction

This section describes the likelihood detector module's detection algorithm. First, an overview of the algorithm's processing stream is provided. Next the algorithm's configuration options are presented using the PAMGUARD user interface as a reference. The following section describes the relationship between dialog parameters and the signal processing blocks that each parameter affects. Subsequent sections explain the detailed processing performed in each processing block, referencing back to the configuration dialogue to explain the effect of relevant settings on the module's behaviour. Finally a guideline to selecting configuration parameters is presented.

## 4.2 Processing Block Diagram

Figure 4 shows an overview of the likelihood detector algorithm's processing stream. Raw time series data is processed to generate spectra and later an average band-limited energy over time. The ETI is used to compute an estimate of the probability that either a signal or just noise are present, which is converted to a likelihood ratio in the normalizer process. This ratio is analyzed in the threshold detection process to produce one PamDetection each time that detection is declared.

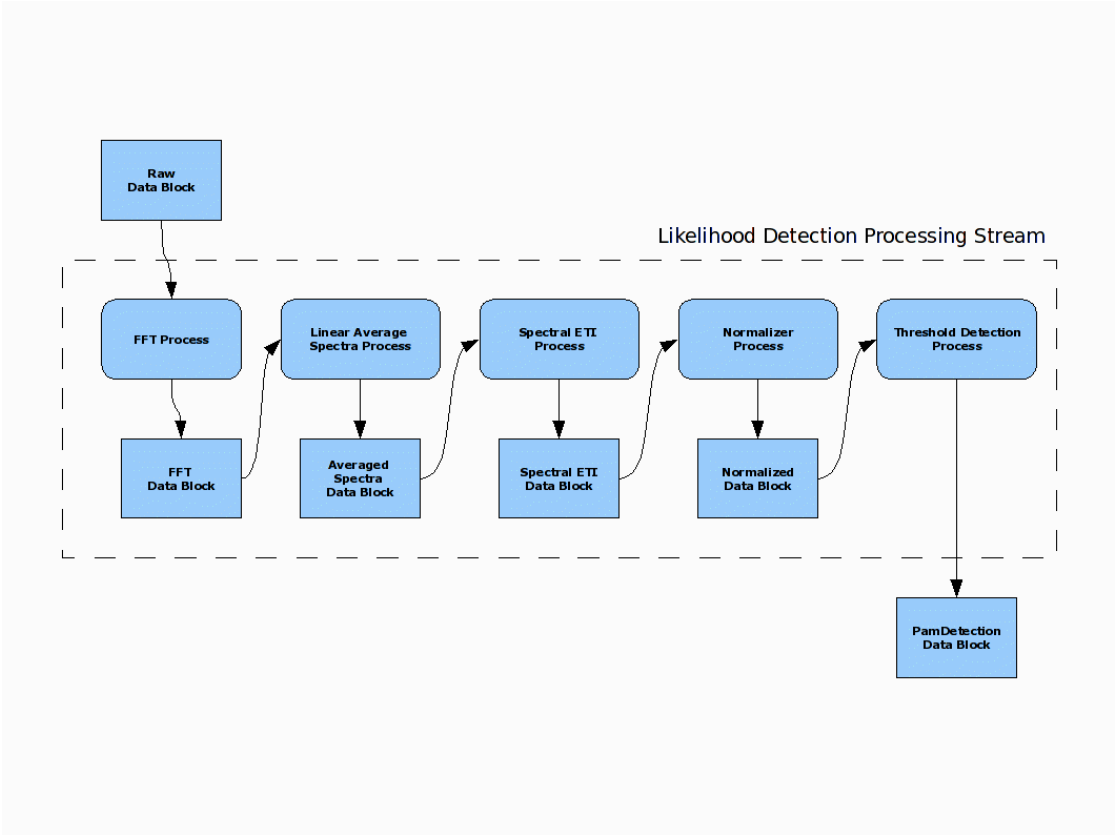

**Figure 4: Block Diagram of the Likelihood Detection Algorithm**

## 4.3 Configuration Dialog

Figure 5 shows an example of the likelihood detector's configuration dialog. The various parameters are included for reference as each of the processing blocks are described in detail. For a detailed description of how to add and configure a likelihood detection module from within PAMGUARD, please refer to the online help, or PAMGUARD tutorial.

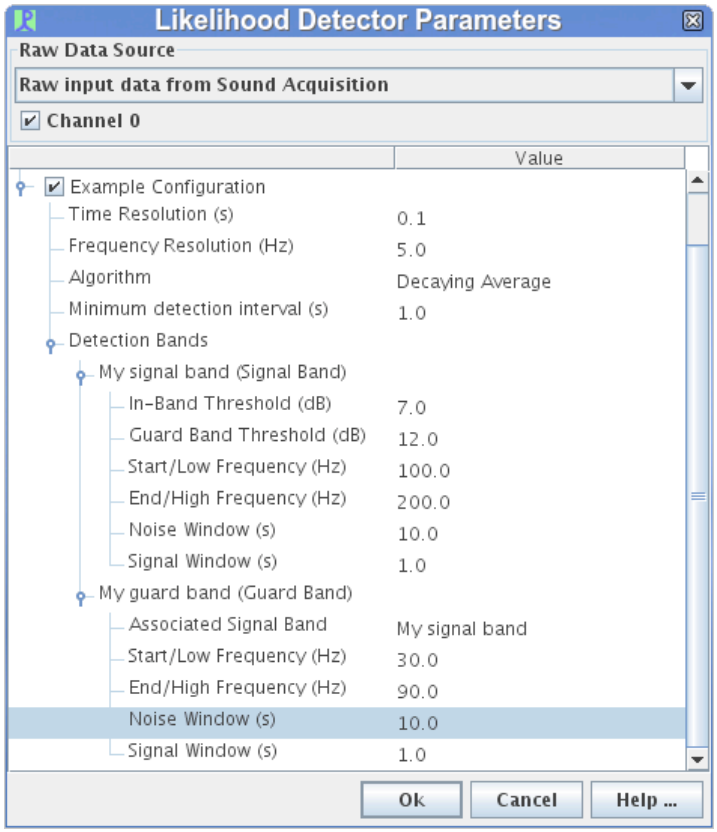


**Figure 5: A Likelihood Configuration with one Signal Band and one Guard Band**

## 4.4 Configuration Parameter to Processing Block Mapping

This section specifies the relationship between configuration dialog parameters (Figure 5) and the signal processing blocks (Figure 4), prior to explaining each processing block in detail. Where appropriate, the configuration parameter name is used and highlighted with *italics* in the processing block descriptions.

- *Dialog: Time Resolution (s) & Frequency Resolution (Hz)*

    - These parameters are used to determine settings for both the **FFT Process** and the **Linear Average Spectra Process** blocks (see Section 4.5 for details). This is not a 1-1 mapping as Time Resolution can affect both processing blocks, while Frequency Resolution only affects the FFT Process block.
- *Dialog: Start/Low Frequency (Hz) & End/High Frequency (Hz)*
    - These parameters (unique to each defined signal and guard band) define the spectral bins that are summed in the **Spectral ETI Process block** (see Section 4.6 for details).
- *Dialog: Algorithm, Noise Window (s) & Signal Window (s)*
    - The Algorithm (a common setting for each detector configuration) as well as each Noise and Signal window defined once for each signal and guard band determine the settings of the **Normalizer Process block** (see Section 4.7 for details).
- *Dialog: Minimum detection interval (s), In-Band Threshold (dB), Guard Band Threshold (dB) & Associated Signal Band*
    - These parameters are used to control how the **Threshold Detection Process block** behaves (see Section 4.8 and 4.9 for details). The Minimum detection interval is a common setting for each detector configuration. Each signal band has both an In-Band and Guard Band threshold. Finally, each guard band (if any are used) has an Associated Signal Band.

## 4.5 FFT Process / Linear Average Spectra Process

The spectral processing portion of the algorithm was performed using PAMGUARD's *FFTDataSource* interface. The user indicates their desired *Frequency Resolution* via the Likelihood Detector Parameters dialog (see Figure 5). Since the Fourier Transform is calculated most efficiently when operating on data blocks that are powers of 2, the detector chooses an FFT size that will result in the resolution that best matches the user-requested *frequency resolution*. The initial FFT order, $N$, is computed using

$$N = fl\ \ oo\left(\log 2\left(\frac{F_s}{F_{res}}\right)\right) \tag{1}$$

where $F_s$ is the raw data sample rate and $F_{res}$ is the user requested frequency resolution. The actual FFT size is then

$$N_{FFT} = 2^N \tag{2}$$

and the resulting resolution is

$$F'_{res} = \frac{F_s}{2^N} \tag{3}$$

with an associated error of

$$error = \frac{|F_{res} - F'_{res}|}{F_{res}} \quad (4)$$

If *error* is less than 25%, then N is used for the FFT order, otherwise N+1 is used.

Similarly the user may select their desired processing *Time Resolution*. Once the FFT size has been determined, the likelihood detector attempts to satisfy this time resolution. This is achieved by a combination of FFT overlap, and spectral averaging. The methodology is described in the following two sub-sections.

### 4.5.1 FFT Overlap (related to Hop Size)

By default, the spectral processing starts with an overlap of 50%. This means that for every FFT operation, only half of the input raw data samples are new, while the other 50% are taken from the end of the previous FFT data as shown in Figure 6.

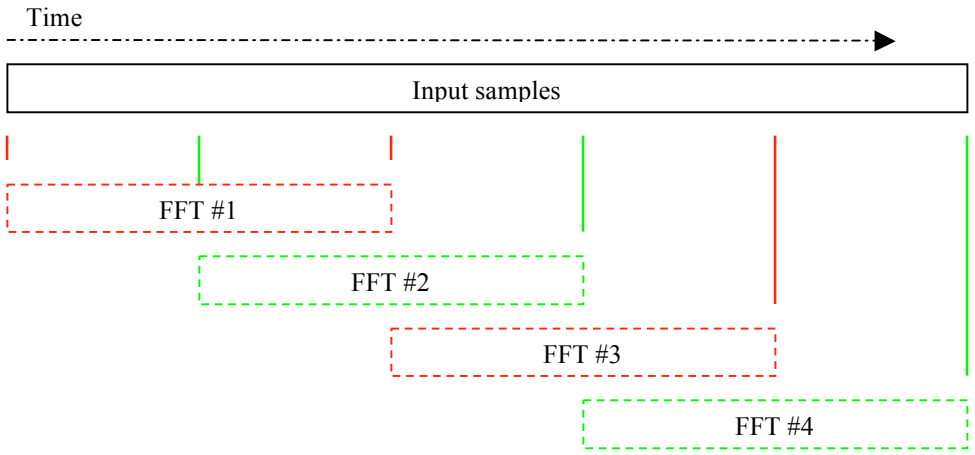


**Figure 6: Overlapping FFT blocks in relation to input samples**

In PAMGUARD, overlap is represented using a related parameter: FFT hop size. Table 1 shows the relationship between percent overlap, hop size, and time resolution.

**Table 1: FFT Table, assuming 8kHz input sample rate and FFT size of 512. We assume no spectral averaging.**

| % OVERLAP | HOP | TIME RESOLUTION |
|---|---|---|
| 0 | 512 | 64 ms |
| 25 | 384 | 32 ms |
| 50 | 256 | 16 ms |
| 75 | 128 | 8 ms |
| 99.8 | 1 | 0.125 ms |
| 100 | 0 | Cannot hop by 0 samples |

Hop is calculated from percent overlap using

$$hop = fl\ \ od\left((100 - overlap)N_{FFT}\right) \tag{5}$$

Actual time resolution is then

$$t'_{res} = \frac{hop}{F_s} N_{avg} \tag{6}$$

where $N_{avg}$ is the integer number of averages. Initially no averaging is used ( $N_{avg} = 1$).

As shown in Table 1, decreasing the hop size also decreases the time resolution. It also reduces the processing efficiency, as more FFT operations must be performed on the same length of input data. A hop size of 1 sample is the minimum possible, but is also extremely inefficient, as an FFT operation would be performed for each new input sample. Overlap is therefore limited to 0.90 (90%), since overlap greater than that would drastically reduce the processing performance of the algorithm.

The likelihood module assumes 50% overlap to start. The resulting time resolution is then calculated. If it is greater than the user requested time resolution, the overlap is increased using

$$overlap = 1 - \frac{F_s \cdot t_{res}}{N_{FFT}} \tag{7}$$

where $t_{res}$ is the user's requested time resolution.

If the calculated time resolution is less than the user requested time resolution, averaging is employed as explained in section 4.5.2.

### 4.5.2 Spectral Magnitude Averaging

As Section 4.5.1 states, the algorithm begins with a 50% overlap. This intermediate time resolution $t'_{res}$ is calculated using (6) with $N_{avg}$ initially set to 1. If the resulting time resolution is less than the user requested, averaging is employed. Time resolution can be increased by an integer factor by averaging the spectra $N_{avg}$ times. $N_{avg}$ is then computed using

$$N_{avg} = f\ loo\left(\frac{t_{res}}{t'_{res}} + 0.5\right) \tag{8}$$

where $t_{res}$ is the user requested time resolution. The resulting *time resolution* is computed using (6) with this new value of $N_{avg}$.

Thus the data rate has been reduced by a factor of $N_{avg}$ and the time resolution has been increased by a factor of $N_{avg}$.

## 4.6 Spectral ETI Process

This module converts spectral magnitude data from the first module to a series of average band-limited energy levels based on user defined frequency band(s) of interest. These are called energy bands and are specified in the dialog as *Start/Low Frequency* and *End/High Frequency*.

The likelihood detector allows the user to define an arbitrary number of energy bands. These bands are defined in terms of a *Start/Low Frequency* and *End/High Frequency*. Frequencies must be between 0 Hz and the Nyquist frequency (half of the data source sample rate).

Energy bands are separated into two classes:

- Signal band – Energy in this band can trigger detection if the likelihood ratio exceeds the specified In-Band Threshold.
- Guard band – Each guard band is associated with a signal band. A signal band may have one or several guard bands associated with it. Guard bands can prevent a signal band from triggering a detection. If the signal band's "signal window" estimate divided by the average "signal window" estimate of all associated guard bands is less than the defined Guard Band Threshold then a detection is not declared.
- The mathematical equations related to detection are explained in detail in Section 4.8

A Noise Window and Signal Window are defined for each band. These are used to compute the associated probability estimates that are used to estimate the likelihood ratio for each time step. Appropriate selection of noise and signal window length is explained in Section 4.10.

## 4.7 Normalizer Process

The computed energy bands are passed into the normalizer process where noise and signal estimates are computed followed by the corresponding likelihood ratio. These estimates are computed using the selected *Algorithm* (decaying average or block average). This section describes the mathematical formulae behind each algorithm and describes a processing optimization that allows the block average to be computed as efficiently as the decaying average.

### 4.7.1 Decaying Average Estimator

The decaying average is a type of infinite impulse response (IIR) filter. It's mathematically represented by

$$y[n] = \alpha y[n-1] + (1-\alpha)x[n] \tag{9}$$

where $\alpha$ is a weighting constant that must be less than 1.0 and is related to the *Noise Window* and *Signal Window* using (10). *x* represents an input band-energy sample for sample *n* and *y* represents the estimate at samples *n* and *n-1* respectively.

The previous output sample $y[n-1]$ is not known for the first data sample, so it is set using the first input sample $x[0]$.

An $\alpha$ of 0.0 produces output samples *y* identical to the input samples *x* (i.e. no averaging). The closer $\alpha$ gets to the value 1.0, the more weight the previous estimate output samples are given and thus the higher the averaging. An $\alpha$ of 1.0 is not allowed as it would prevent the input from affecting the output.

In PAMGUARD the user defines the averaging time constant in seconds, which is converted to $\alpha$ using

$$\alpha = 1 - \frac{t_{res}}{\tau_c} \tag{10}$$

where $t_{res} \le \tau_c \le 600$ to ensure practical constraints and $\tau_c$ is the user defined *Signal Window* or *Noise Window* size. This is an approximation as the true decay time constant is dependant on the correlation between samples, but the approximation is adequate for configuring the detector.

#### 4.7.1.1 Two Speed Decaying Average Estimator

A very long time constant (or equivalently, an $\alpha$ value very close to 1.0) is often used to estimate the noise level when using the decaying average algorithm. If the real background level decreases (such as at the end of a long rainstorm), it may take quite some time for the estimator to lower its background estimate to the new level. This can result in missed detections during this time since the likelihood ratio is underestimated. This overestimation of the background level can also occur after a long series of detections that will cause the noise estimate to slowly drift up during detection.

The decaying averager used for background estimation is modified to use two values for $\alpha$ to allow it to recover more quickly after a significant drop in background level. The value of $\alpha$ is chosen for each input sample, based upon

$$\alpha = \begin{cases} \alpha_S, & y_N[n-1] \le \lambda \cdot y_S[n-1] \\ \alpha_N, & otherwise \end{cases} \tag{11}$$

Basically, $a_N$ is used when the noise estimate is lower or on the same order of the signal estimated (the normal case), but if the last signal estimate, multiplied by the scaling factor $\lambda$ is less than the last noise estimate, $\alpha_S$ is used. $\lambda = 1.5$ works well for this detector and is set as a constant inside the algorithm.

#### 4.7.1.2 Pulse Response of the Decaying Average Estimator and Likelihood Estimator

The step response of the decaying average estimator and resulting likelihood estimate is illustrated in this section. A time resolution of 1 second is used for all examples. A test pulse (target) is defined as a 30 decibels (dB) energy spike with duration of 3 samples (3 seconds). The pulse is centered 50 seconds into the data with all other input

data set to a constant level of 10 dB. This artificial example serves to demonstrate the behavior of the associated estimators.

The decaying average is configured to use a signal window length of 3 seconds, and a noise window length of 60 seconds and later 9 seconds to illustrate an effect. The results of processing are shown in Figure 7. The input signal produces a detection likelihood ratio exceeding 10 dB (bottom panel). The dotted line represents a detection threshold of 3 dB, which would result in detection beginning at 49 seconds and continuing until 56 seconds; about 7 seconds in duration and noticeably longer than the actual pulse.

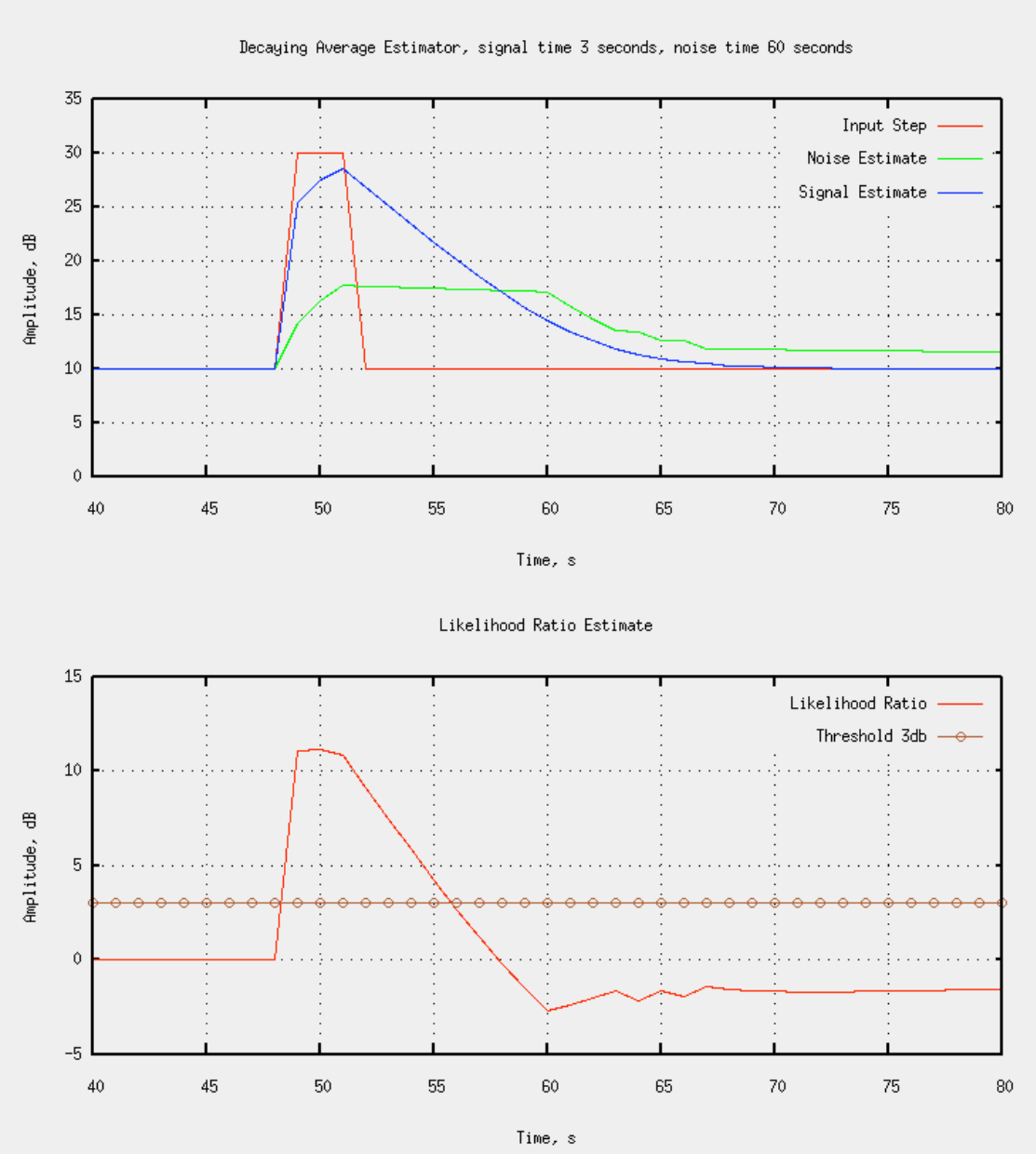


**Figure 7: 3 second pulse response of the decaying average estimator**

The noise estimate is noticeably disturbed for ~20 seconds after the pulse (top panel). Note the change in decay of the noise estimate once the signal estimate is noticeably lower than it. This is the result of the two-speed decaying average.

It is also worth noting that after detection the likelihood ratio estimate's steady state value is below 0 dB, due to residual energy in the background estimate. This will decay exponentially, however if a weak signal were introduced after the strong signal it may not be detected.

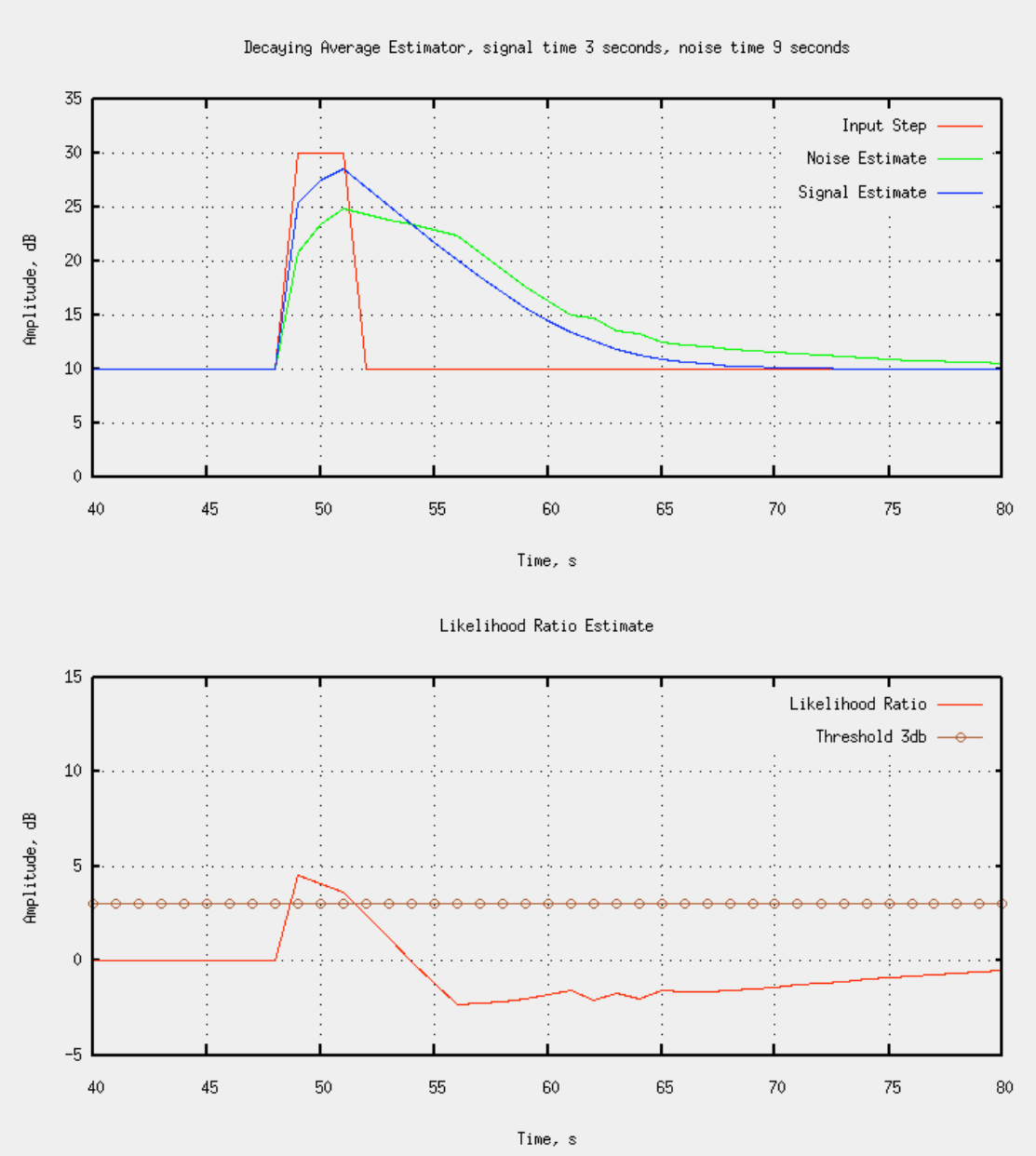


**Figure 8: 3 second pulse response of the decaying average estimator, 9 second noise window**

Choosing a much shorter noise window length (9 seconds) results in Figure 8. This window length was chosen to illustrate the different results obtained when identical

window lengths are chosen for block average and decaying average estimators. As shown, the noise estimate tracks the input signal more closely, resulting in a much weaker detection. This result is in sharp contrast to the result obtained when using the block average (Section 4.7.2.1). These issues are further discussed in Section 4.7.3.

Next a much longer pulse is used to stimulate the same processing stream, again using a noise window length of 60 seconds. This simulates a non-target signal that abruptly starts (i.e. rain storm, ship power up, etc). The pulse is still 30 dB magnitude, but is

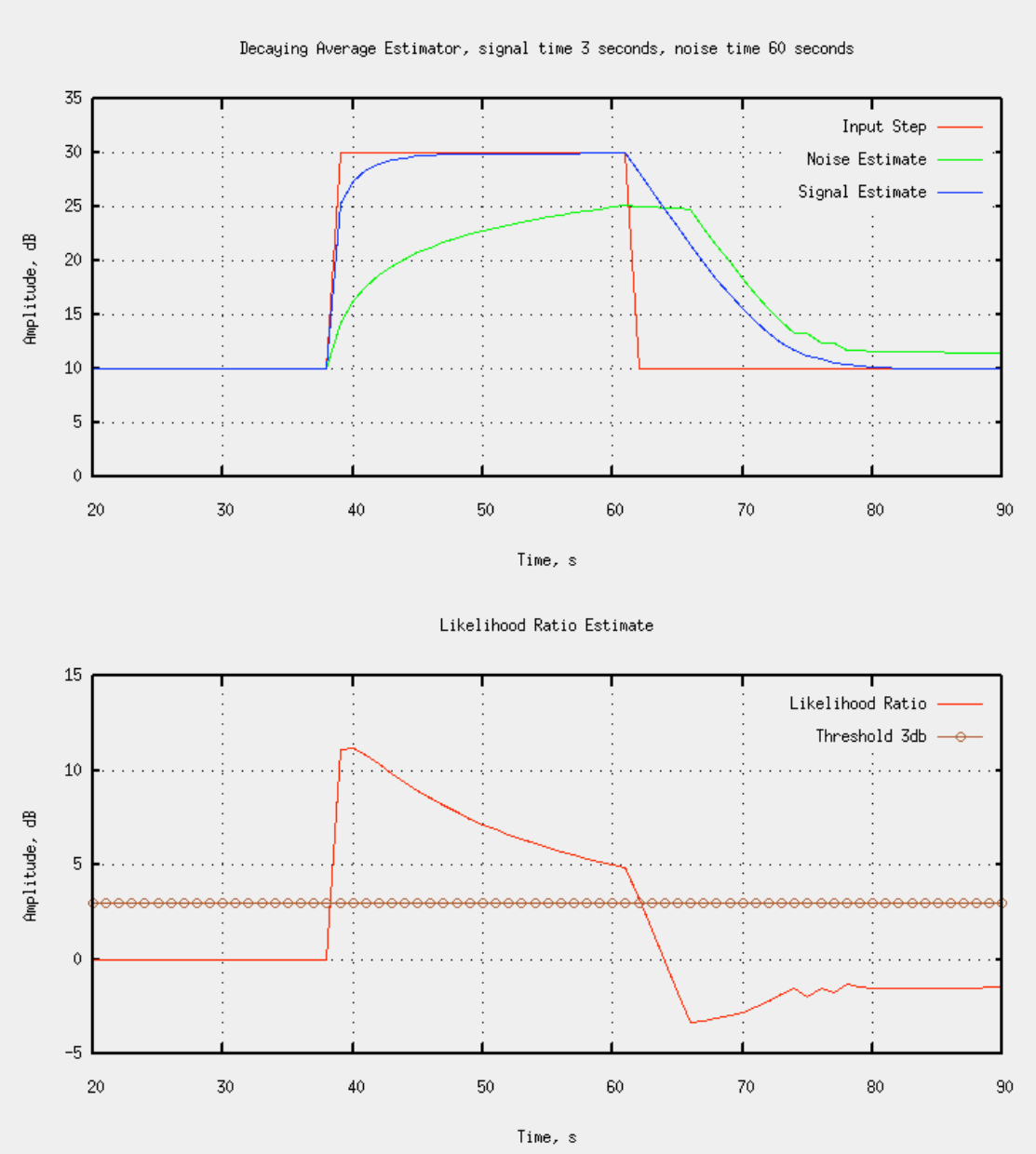


**Figure 9: 23 second pulse response of decaying average estimator**

now 23 seconds in duration and centred at 50 seconds. The length of the pulse could be much longer and would still produce the same effect. The results of processing are shown in Figure 9.

The longer duration pulse produces detection, an undesirable result. This demonstrates a weakness in the decaying average algorithm. It cannot discern between short or long duration signals because it only processes prior data points with no awareness of data beyond the current sample. Section 4.7.2.1 demonstrates that the block average estimation algorithm performs better in this case. This is important since PAMGUARD should trigger on a whale call which is short duration, while it should not trigger on longer-duration events such as an abrupt change in ship speed, which is much longer in duration, but may fall within the same frequency band.

### 4.7.2 Block Average (Split Window) Estimator

The split window estimator uses two rectangular windows of different widths to estimate the signal and noise levels. The windows are always an odd number of samples to allow precise definition of the window centre. The *Noise Window*, $W_N$, must be longer than the *Signal Window*, $W_S$. The signal for each time step *n* is estimated using

$$y_S[n] = \frac{1}{W_s} \sum_{i=-(W_S-1)/2}^{(W_S-1)/2} x[n+i] \tag{12}$$

where *x* is the band energy. The noise is then estimated using

$$y_N[n] = \frac{1}{W_N - W_S} \left[ \sum_{i=-(W_N-1)/2}^{(W_N-1)/2} x[n+i] - y_S[n] W_S \right] \tag{13}$$

The split window estimator is non-causal, that is, to calculate output sample $y[n]$ it must have access to input sample $x[n + (W_N - 1)/2]$. In a real time processing system, this means that the output will lag the input by approximately half of the length of the noise window. The algorithm is implemented by buffering, which introduces a lag in the processing stream. Depending on the time resolution and the length of the estimator's windows, the lag could be several seconds long, resulting in a delay before detections are presented to the operator. This is usually acceptable as other delays including operator reporting and team mitigation are often much longer resulting in an insignificant change in overall reaction time. If this is not acceptable then the decaying average estimator should be used.

A graphical representation of the windows is shown in Figure 10. The average of the samples contained in the red is the signal estimate, while the average of the green colored samples becomes the noise estimate. This window slides along with the data moving one sample at a time.

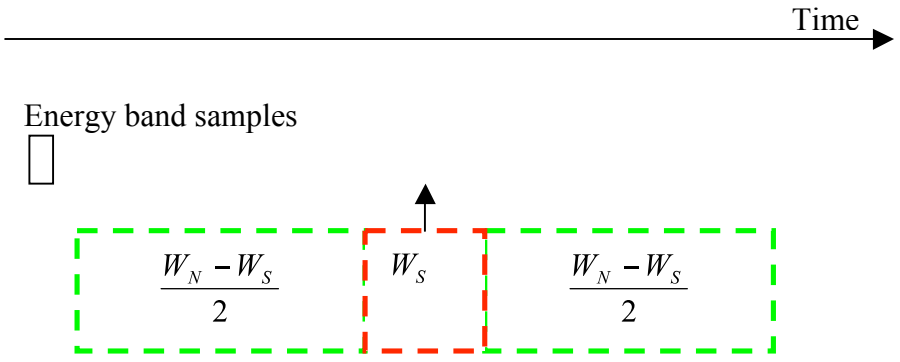


**Figure 10: Graphical representation of split window estimator**

The split window estimator has finite impulse response (FIR) and can be represented by two FIR filters. Implementation of this estimator occurs in the time domain and uses a processing optimization. Results are computed using a processor load that is very similar to that of the decaying average. The optimization exploits the sliding window by simple adding one sample and removing another. This requires management of a circular buffer containing all active samples, but saves considerable processing time.

#### 4.7.2.1 Pulse Response of Block Average Estimator and Likelihood Estimator

As in Section 4.7.1.2, the pulse response of the block average algorithm is demonstrated herein. The same 30 dB, 3-second duration pulse is passed through the processing stream resulting in the output plotted in Figure 11. A signal window length of 3 seconds and a noise window length of 9 seconds are used. The preferred noise window is shorter than that of the decaying average due to the estimator's characteristics. A 60 second window could be used, but strong medium-duration pulses may be detected, which is not desirable.

The same signal produces a likelihood ratio peak level of 20 dB, almost 10 dB higher than the decaying average and an exact match to the actual signal to noise ratio (SNR). This match only occurs because the signal window length is identical to the synthetic signal duration. The noise estimate (green) is a bimodal signal, with a depression corresponding to the peak of the signal estimate, where none of the signal is present in the noise estimate. The noise estimate is clearly affected by the signal when the windows are not centered on the signal, resulting in large dips in the likelihood ratio on either side of a pulse. This limits how close signals can be in time and still be detected. This could be mitigated using a 2-pass algorithm that would estimate signal locations and remove them from consideration in the noise estimate, a standard signal processing technique.

Using a detection threshold of 3 dB, the detection would begin at 49 seconds and continue until 51 seconds; about 3 seconds in duration. This is a better match to the true signal duration than the decaying average.

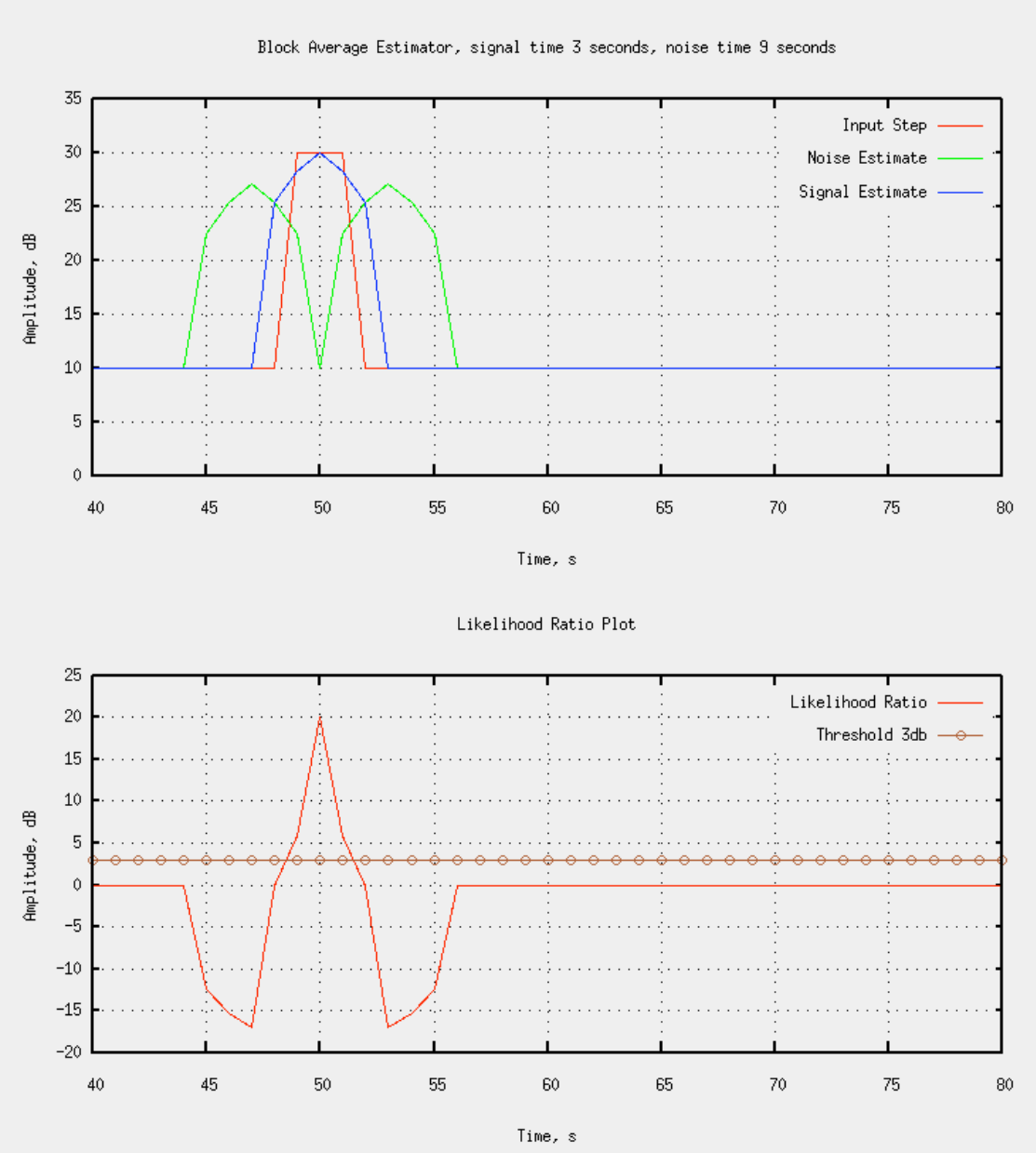


**Figure 11: 3 second pulse response of the block average estimator**

As in Section 4.7.1.2, the processors response to a much longer pulse is now demonstrated. A 23 second pulse with the same amplitude is used. The window lengths used for the short pulse are duplicated to ensure that the processing is identical. The processor's output is plotted in Figure 12 illustrating that detection would not result in this case as long as the threshold is at least 3 dB. This can be

proven mathematically for a square pulse. Generally the block average estimation algorithm can be used to limit the signal duration that will result in detection, whereas the decaying average estimation algorithm is unable to use signal duration as a discriminator.

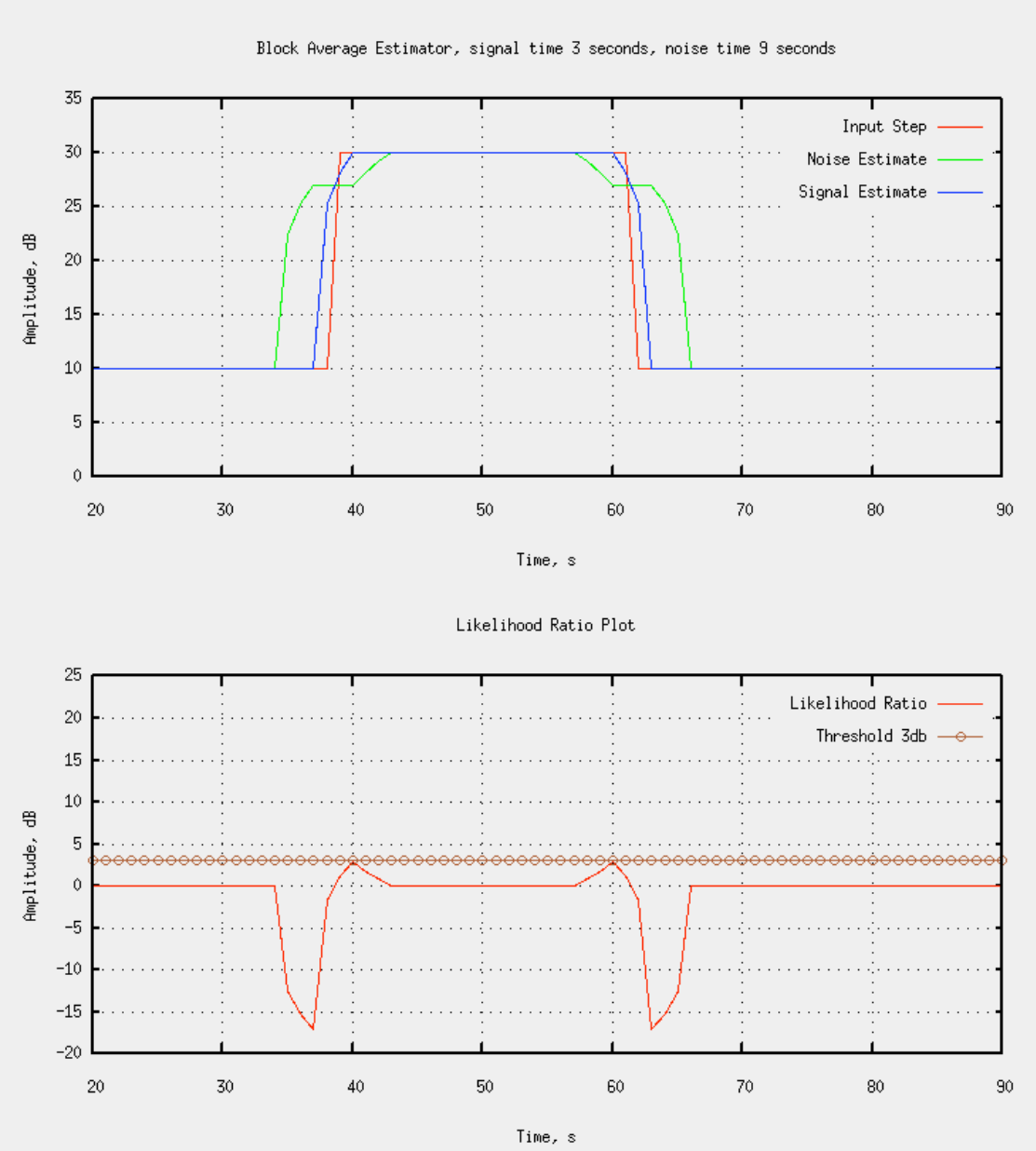


**Figure 12: 23 second pulse response of the block average estimator**

### 4.7.3 Estimator Selection Conclusions

The previous sections provided the formulas and examples of the processing for both likelihood estimation algorithms. This section summarizes the conclusions:

- Both algorithms require the same order of processing power.
- The block average introduces a fixed lag equal to one half the noise window length, while the decaying average has no lag.
- The block average produces a better estimate of the true SNR for signals that match the signal window length.
- The block average will filter detections that are longer than the signal window length. The cutoff duration is dependent on the signal energy envelope, noise window length, and detection threshold.
- The decaying average takes time to recover after loud signals even with the 2-speed averaging method.
- The block average produces minima in the likelihood ratio before and after a pulse that could be mitigated using a 2-pass split window technique.
- Different guidelines are required in setting signal and noise window lengths depending on the selected algorithm.

## 4.8 Threshold Detection Process

This section defines the LRT implemented by the subject detector. The LRT differs from the classical approach in that it uses guard bands as part of a secondary test to reduce false detections.

The detection decision is taken in two stages. First the in-band likelihood is computed using

$$L_t = \frac{y_S}{y_N} > \tau_t \tag{14}$$

where $L_t$ represents the estimated likelihood ratio for the *main* or signal band and $\tau_t$ is the in-band threshold in linear units vice decibels. The user specifies this as *In-Band Threshold* in Figure 5. If this test passes, and guard bands are defined, the secondary LRT is also performed using

$$L_b = \frac{y_S}{\overline{y_{SG}}} > \tau_b \tag{15}$$

where $\overline{y_{SG}}$ is the average signal estimate of all associated guard bands and $\tau_b$ is the guard band threshold in linear units vice decibels. The user specifies this as *Guard-Band Threshold* in Figure 5. The second test ensures that the detected signal is appropriately band limited and has proven effective at reducing false alarms.

## 4.9 Minimum Time Between Detections

The detector algorithm filters detection events by enforcing a minimum time between detections (*Minimum detection interval* in Figure 5). This can help reduce the amount of information presented to the operator, or downstream processing. In most situations operators are most interested in the simple presence of marine mammals, not exhaustively collecting information on each and every click or moan, so this setting may be appropriate.

Setting it to zero effectively turns it off, but if the operator chooses a non-zero value for the minimum time between detections, they will see, at most, one detection within that interval. This filtering is performed on a channel-by-channel basis. Thus a detection on sensor channel 1 will not prevent a detection from appearing on sensor channel 2 even if it occurs before the minimum time has been reached.

Furthermore, this filtering is performed independently for each target configuration. For example, if the operator has added two detector configurations, one for beaked whale clicks and another for Right whale moans, both configurations may have a different value defined for minimum time between detections. Furthermore, detections caused by beaked whale clicks cannot prevent detections caused by Right whale moans even if they occur on the same sensor channel.

## 4.10 Selecting Detector Configuration parameters

This section provides *rules of thumb* that can be used to select algorithm configuration parameters until users become familiar with the subtleties of the algorithm and can select parameters specific to their intended use.

### 4.10.1 Time Resolution

For very short signals (<50ms) the time resolution should be on the order of the signal duration, or up to twice that. Setting it to a value much larger than the signal duration will reduce the detectors ability to detect low SNR (weak) signals as the signal energy will be averaged over the FFT window length (considering the Hann window is applied).

For longer signals, a time resolution less than the signal is appropriate, and helps to achieve better detection timing. The default of 0.1 seconds is often acceptable. The data is still averaged over the length of a signal using the signal window parameter. The signal window is a sliding window, which is more appropriate for producing band-energy estimates over the expected signal duration.

### 4.10.2 Frequency Resolution

The frequency resolution is normally set to

$$f_{res} = \frac{1}{2} t_{res} \tag{16}$$

that results in 50% overlap.

### 4.10.3 Band Frequency Limits

It is normal to define the band low and high frequency to contain the entire expected signal, or at least that portion of the signal that contains the most energy.

Guard bands are often defined both above and below a signal, if both are expected to be undisturbed during a vocalization. Defining one guard band (above or below a signal) is also appropriate especially to avoid strong harmonics. Where the spectrum is colored (not flat) due to sensor response or ambient noise, the relative spectral energy

of ambient noise in the signal and guard band should be considered when setting threshold levels.

### 4.10.4 Algorithm and Signal / Noise Window Lengths

For the block average estimator, a good choice for the signal window is the duration of the click or call being detected, while the noise window should be at least 3 times longer. If the actual signal's duration is variable, the signal window can be biased to the longer cases. If the difference between short and long duration signals is significant (more than 2x), another option is to create multiple bands each tuned to the different durations. The noise window shouldn't be so long as to cause neighbouring signals to be in the noise window during detection. For example, if the real signal's duration is 1.0 seconds and the inter-call interval (ICI), defined as the time between start of one call and the start of the next call, is 3.0 seconds a noise window of 5 seconds would be appropriate. In this case, a noise window of say 10.0 seconds would be too long. Generally

$$W_N \leq 2 \cdot ICI - W_S \tag{17}$$

to avoid corrupting the noise estimate with other signals of interest. Longer noise windows will result in more stable estimates and the effect of loud signals on the average will be less, but it will also allow false alarms from medium duration signals (between the signal window length and approximately half the noise window length).

For the decaying average estimator that uses feedback to generate estimates similar to the processing used in the PAMGAURD click detector, a different strategy for selecting signal and noise windows is appropriate. The signal window can still be on the order of the duration of the vocalization of interest, though it will effectively smear out the signal if it is also many more times greater than the time resolution. As a general rule of thumb, it should not be more than 4 times greater than the time resolution. The noise window needs to be longer than for the block average estimator, especially if loud calls are occurring frequently, or the noise estimate will develop a bias due to the feedback as illustrated in Section 4.7.1.2.

In general, the block average estimator is a better algorithm choice because it doesn't have a long-term memory.

# 5 Performance Testing

Performance testing on real data was an integral part of this project. It was conducted early on to improve Akoostix understanding of current detector limitations, to validate prototype improvements, and finally to demonstrate performance of the detection algorithms for a variety of practical cases.

Rigorous, detailed performance testing across the spectrum of species and environments is not practical at this stage of development. Continued performance evaluation is likely as various users employ the detection algorithm, record the configuration used, and report results. This information can be used to suggest adjustments to their configuration, suggest algorithm improvements, and to adjust the circumstances under which this algorithm is employed.

## 5.1 Overview

Much of the analysis work was conducted ad hoc at Akoostix, involving detailed examination of the various stages of detection, and the recorded data that produced both valid and false detection. The results were used to iterate the detector design. Formally documenting the informal analysis is beyond the scope of this contract.

Once the detector was complete, final performance testing was conducted with preliminary results presented at the OGP JIP Program review [4]. A more detailed analysis is provided in the proposed journal article [7] and summarized below. The detector configurations used to produce the journal article were also provided to PAMGUARD as examples [10] to support user evaluation and employment.

## 5.2 Data and Configuration

This section provides information on the data that was used for performance testing and how the detector was configured.

Mobysound mysticetes data were used as the primary data for performance analysis [14]. This data is freely available for research purposes and would allow other researchers to duplicate and compare these results to other algorithms. Humpback, Bowhead and Right whale species were selected along with noise data. A secondary data set from DRDC was used to add Sperm whale and more noise data, as Mobysound noise samples were not considered sufficiently challenging.

Noise samples borrowed from DRDC included one data set with loud shipping and another that contained low-frequency active sonar pings. Samples of seismic exploration were also requested but none were available.

Table 2 documents the amount of data provided by type. Data sets containing vocalization typically included high SNR calls with some other noise and recording artifacts. The call frequency is typically very high. The noise data is not known to contain any marine mammal vocalizations.

All data were resampled to 8 kHz using sox, an open source sound utility, with the quadratic (–q) resampling option. Most data were not significantly affected by the resampling, but Sperm whale clicks were significantly band limited. This is equivalent to using a hydrophone with limited bandwidth and considered a valid use case. Resampling was required to ensure that all frequency band parameters would remain valid for all data.

**Table 2: Amount of data processed**

| | Duration (minutes) |
|---|---|
| *Bowhead Data* | 66.1 |
| *Humpback Data* | 127.3 |
| *Right Whale Data* | 112.8 |
| *Sperm Whale Data* | 62.5 |
| *Noise Data* | 250.8 |

A total of four detector configurations were created for each estimation algorithm (e.g. decaying average and split-window average) to facilitate comparison of the two algorithms across species.

One detector configuration was created for each of Bowhead and Sperm Whale data. Two configurations were developed for Humpback calls, accounting for the greater variety of calls. Right Whale data was included in the tests, but a specific detector was not created for this species.

Table 3 and Table 4 contain the precise processing parameters used within the algorithm to match the formulas provided in Section 4. Equivalent (recommended) processing settings in PAMGUARD format (i.e. as would be required to fill in the configuration shown in Figure 1) were provided to the PAMGUARD team and should be available from their website. These configurations were also provided as part of the project deliverables [10]. Here the decaying average detector's thresholds were adjusted to match the detection rate of the split window detector for each target species. The enables comparison of the false detection rate and ensures validity of algorithm comparisons.

**Table 3: Detector parameters for decaying average based detection**

| Param. | Bowhead | Humpback 1 | Humpback 2 | Sperm whale |
|---|---|---|---|---|
| *N FFT* | 2048 | 2048 | 2048 | 64 |
| *Overlap* | 61% | 61% | 61% | 68.8% |
| *Band Min* | 100 Hz | 200 Hz | 625 Hz | 1000 Hz |
| *Band Max* | 700 Hz | 500 Hz | 1500 Hz | 3900 Hz |
| $\alpha_S$ | 0.9 | 0.9 | 0.9 | 0.000 |
| $\alpha_N$ | 0.998 | 0.998 | 0.998 | 0.988 |
| $\tau_t$ | 4.6 | 6.5 | 6.5 | 5.1 |
| $t_{min}$ | 3.0 s | 3.0 s | 3.0 s | 0.5 s |

$t_{min}$ denotes the minimum allowed time between detections.

**Table 4: Detector parameters for split window average based detection**

| Param. | Bowhead | Humpback 1 | Humpback 2 | Sperm whale |
|---|---|---|---|---|
| *N FFT* | 2048 | 2048 | 2048 | 64 |
| *Overlap* | 61% | 61% | 61% | 68.8% |
| *Band Min* | 100 Hz | 200 Hz | 625 Hz | 1000 Hz |
| *Band Max* | 700 Hz | 500 Hz | 1500 Hz | 3900 Hz |
| *Guard Min* | 1000 Hz | 600 Hz | 200 Hz | 1000 Hz |
| *Guard Max* | 1500 Hz | 1000 Hz | 500 Hz | 3900 Hz |
| $W_S$ | 25 | 25 | 25 | 1 |
| $W_N$ | 55 | 45 | 45 | 11 |
| $\tau_t$ | 4.0 | 4.0 | 4.0 | 3.0 |
| $\tau_b$ | 16.0 | 16.0 | 16.0 | 4.5 |
| $t_{min}$ | 3.0 s | 3.0 s | 3.0 s | 0.5 s |

$W_S$ for guard bands is identical to signal bands expect for the Sperm whale detector where $W_S=200$. $W_N$ for guard bands does not affect detection. $t_{min}$ denotes the minimum allowed time between detections.

## 5.3 Analysis Methodology

Performance was examined by counting the number of detections for all combinations of target configuration and species. Examining performance across this wide spectrum of data provides a better indication of potential real-world performance. A simple analysis of detections for the species of interest and any noise that occurs during those species recordings would be less rigorous. (Note: Results from both Humpback configurations were combined into a single set of results.)

The detection rate for correct detections was matched across algorithms by adjusting the exponential average threshold. This allows performance comparison between detection methods for one point on a ROC curve. The algorithm that produces a lower false alarm rate, for the same detection rate would be considered superior.

No effort was made to classify between marine mammal detections and other noise events when the detector configuration did not match the species contained within the data, though further classification of detections was performed when the detector configuration matched the data (i.e. when the Bowhead configuration was used on known Bowhead calls). This was done to determine if the detector triggered on non-specified calls. Analysis was performed manually by viewing a spectrogram, time series, and listening as required.

The typical Bowhead endnote was specified for that species. Only three Humpback units[1] were specified with two falling into one band, all other Humpback unit detections were not included in the final statistic (shown in brackets). A standard Sperm Whale click was specified for that species. After analyzing 20% of the sperm whale detections and only finding sperm whale clicks, it was assumed that all detections in that data were of sperm whales.

## 5.4 Results

This section provides an overview of performance testing on real data, where the data and configurations described in Section 5.2 were used. Analysis followed the methodology described in Section 5.3. Detection performance was examined for all combinations of detector and data.

The results of the detector analysis are provided in Table 5 and Table 6. The species-specific classification results are provided in brackets. The results demonstrate that detection rate can be maintained with a significant reduction in false alarms when the split-window estimator is used on this data using the specified configurations.

There is clear overlap between the Bowhead and Humpback calls with many cross-species detections. This was expected as the frequency and duration of the Bowhead endnote and many Humpback units overlap. There is some improvement in false alarm rate for these cases with the proposed detector, but improvements are more significant for other cases such as for the Right whale data.

Sperm whale detection is clearly a problem, in part due to the reduced recording bandwidth. Many of the recordings also contained impulsive signals from the hydrophone bumping into objects and from the environment that were difficult for the detector to eliminate. Increasing the bandwidth and using more advanced classification at later processing stages would improve system performance for signals of this type. Regardless the proposed detector would reduce the volume of false detections requiring more complex processing. Here the sonar related signals were easily removed from the noise data.

The results are presented in terms of the raw number of detections instead of detection and false alarm probabilities, as there is less room for interpretation. The results are valid for the data sets used. Every attempt was made to use the best available data, and more challenging noise data was added to improve the relevance of the results. Regardless, different performance will be realized with different sensing hardware in a

[1] A Humpback unit is a short continuous vocalization with a break before and after similar to a spoken word.

different environment. On a different day or in a different area, the species of interest may produce more or less variable vocalizations and the sensed data may have different SNR. Any attempt to extend these performance figures to those scenarios would be difficult if not impossible.

**Table 5: Detector results for decaying average based detection**

| Data | Bowhead | Humpback | Sperm whale |
|---|---|---|---|
| *Bowhead Data* | *(290)* 374 | 371 | 1668 |
| *Humpback Data* | 1065 | *(339)* 1130 | 3597 |
| *Right Whale Data* | 111 | 58 | 3880 |
| *Sperm Whale Data* | 0 | 0 | *(1622)* 1622 |
| *Noise Data* | 4 | 3 | 375 |

Numbers in brackets indicate the number of correct detections for matching species and detector configuration.

**Table 6: Detector results for split window average based detection**

| | Bowhead | Humpback | Sperm whale |
|---|---|---|---|
| *Bowhead Data* | *(289)* 321 | 299 | 696 |
| *Humpback Data* | 754 | *(335)* 744 | 1064 |
| *Right Whale Data* | 28 | 3 | 3404 |
| *Sperm Whale Data* | 0 | 0 | *(1618)* 1618 |
| *Noise Data* | 0 | 0 | 59 |

Numbers in brackets indicate the number of correct detections for matching species and detector configuration.

For this dataset and the selected configuration, the detector performance was very high, especially where the split-window estimator was employed. The cumulative probability of detection very quickly reaches 100%, while over four hours of challenging pure noise data produces no detections in two of the three test scenarios. The third test scenario is very likely to improve with increased bandwidth as the false alarms occur on cavitating blade pops that are more limited in bandwidth than sperm whale clicks. Though seismic data was not available the detector should be able to be configured to reject seismic noise for many target species. This detection algorithm coupled with operator monitoring, localization, and classification tools could prove very effective for PAM.

## 5.5 Performance Summary

A more general summary of the performance analysis is provided herein, with more detail found in the previously referenced presentation and article:

- The use of a split-window average and guard bands can reduce the false alarm rate, for a given detection rate.

- When implemented using the suggested optimization, processing load does not change significantly over other energy-based detectors.
- Processor load scales well with number of detector configurations, provided the initial FFT processing parameters match.
- Waiting for more complex classification processing stages to filter false detections increases processing load and system complexity.
- The detector is suitable for variable calls and many species. (Further analysis of the detector's performance for beaked whale click detection, including a more in-depth receiver operating characteristic (ROC) analysis, can be found in [15].)
- The general principal of requiring an absence of non-target signal features should be attempted with other detection techniques such as spectrogram correlation. It is believed that a similar reduction in false alarm rate can be achieved once the same general principals are applied.
- Sperm whale clicks are difficult to differentiate from impulsive signals using the proposed detector, and likely any simple detector. Initial classification of species is inherent to the detector, but some species will be difficult to segregate. Subsequent processing stages that use perception based classifiers, such as that proposed by Young and Hines [16], are being explored to improve system performance with promising results. Another approach would be to couple this detector with the click train analysis used in other PAMGUARD modules.
- There is clear overlap between the Bowhead and Humpback calls with many cross-species detections. This was expected as the frequency and duration of the Bowhead endnote and many Humpback units overlap. There is some improvement in false alarm rate for these cases with the proposed detector, but improvements are more significant for other cases such as for the Right whale data.

# Annex A Acronyms

| | |
|---|---|
| ACDC | Acoustic Cetacean Detection Capability |
| API | Application Program Interface |
| ATP | Acceptance Test Plan |
| ATR | Acceptance Test Report |
| dB | decibel |
| DRDC | Defence Research and Development Canada |
| ETI | Energy Time Indicator |
| FFT | Fast Fourier Transform |
| FIR | Finite Impulse Response |
| GUI | Graphic User Interface |
| Hz | Hertz |
| IBM | International Business Machines |
| ICI | Inter-Call Interval |
| IEEE | Institute of Electrical and Electronics Engineers |
| IIR | Infinite Inpulse Response |
| JAR | Java ARchive |
| JIP | Joint Industry Programme |
| LRT | Likelihood Ratio Tracker |
| MVC | Model View Controller |
| O&G | Oil and Gas |
| OGP | Oil and Gas Producers |
| PAM | Passive Acoustic Monitoring |
| PAMGUARD | Passive Acoustic Monitoring Guard |
| ROC | Receiver Operating Characteristic |
| STAR | Software Tools for Analysis and Research |
| UI | User Interface |

# Annex B References


1. Mitigation and Monitoring: Passive Acoustic Monitoring (PAM) Software Development – Detection Classification and Localization Capabilities, International Association of Oil and Gas Producers, Contract 07-18, Dec 2007.
2. Meeting Minutes, PAMGUARD Integration Discussion, Akoostix Inc. & Douglas Gillespie, 31 July 2008.
3. Hood, J., Burnett, D. (2008) Final Report for Compilation of Marine Mammal Passive Transients for Aural Classification, (DRDC Atlantic CR 2008-XXX), Akoostix Inc.
4. Hood, J. (2008), PAM Mysticete Detection Algorithms and Performance – An Improvement for PAMGUARD, presented at the Joint Industrial Programme Review Meeting 28-30 Oct 08, Houston, Texas.
5. READ ME Likelihood Detector Version 1.0, (2008), Akoostix Inc.
6. Likelihood Detector - Standard Acceptance Report, (2008), PAMGUARD.
7. Hood, J., Flogeras, D., (unpublished), Improved Passive Band-limited Energy Detection for Marine Mammals.
8. McInnis, J. (2008), Test Plan for the Likelihood Detector Module in PAMGUARD, Version 1.1, (2008-006), Akoostix Inc.
9. PAMGUARD (2008), A User's Introduction to PAMGUARD, Tutorial (Draft from Version 1.1.01, Oct 2008)
10. Likelihood Detector - Configuration Files (2008), zip file provided with contract deliverables, Akoostix Inc.
11. likelihood-javadoc (2008), zip file provided with contract deliverables, Akoostix Inc.
12. likelihood-detector-docs (2008), zip file provided with contract deliverables, Akoostix Inc.
13. Likelihood Detector – akoostix_mt_src, (2008), software zip file provided with contract deliverables, Akoostix Inc.
14. Mellinger, D., Clark, C. (2006), MobySound: A reference archive for studying automatic recognition of marine mammal sounds, *Applied Acoustics*, vol. 67, pp. 1226-1242.
15. Theriault, J., Hood, J., Moretti, D., DiMarzio, N., Mosher, D., Murphy, T. (unpublished) Detection of Blainville's Beaked Whale (Mesoplodon densirostris) using Autonomous Underwater Gliders.
16. Young, V., Hines, P. (2007), Perception-based automatic classification of impulsive-source active sonar echoes, *J. Acoust. Soc. Am*., vol. 122(3), pp. 1502-1517.