CABLE: Cachalot Automatic Body Length Estimator

User Manual

Written by Wilfried Beslin
Updated December 16, 2018
Software version: 1.0

TABLE OF CONTENTS

1. INTRODUCTION

1.1 What is CABLE?

CABLE is a tool capable of automatically estimating how many sperm whales are present in an audio recording, and how large they are. It does this by compiling and clustering stable inter-pulse interval (IPI) distributions from sperm whale clicks it has detected in the recording. The program was written with MATLAB version R2015a (The MathWorks, Inc.), and uses the Signal Processing Toolbox, the Statistics and Machine Learning Toolbox, the Curve Fitting Toolbox, and the Parallel Computing Toolbox.

The program was developed with two important goals in mind:

- **Flexibility**
  CABLE was designed to be usable even with very limited recording equipment and field protocols. It only requires a single audio channel, and it does not rely on knowledge of how many whales are present, or what their orientations are with respect to the receiver.

- **Automation**
  CABLE is completely automatic. The only user input needed is the initial specification of data files and routine parameters.

1.2 Background: sperm whale IPIs and acoustic length estimation

Sperm whale clicks are unique in that an individual click contains multiple pulses (Backus & Schevill 1966). These pulses are the result of a single sound pulse reverberating through the spermaceti organ (Norris & Harvey 1972; Møhl 2001). Consequently, the inter-pulse interval is directly related to the size of the spermaceti organ. Since there is a relationship between spermaceti organ size and total body length (Nishiwaki *et al.* 1963), it is possible to estimate a whale's body length simply by measuring the IPI of its clicks. This relationship has been well established in the literature (e.g. Norris & Harvey 1972; Møhl *et al.* 1981; Adler-Fenchel 1980; Gordon 1991; Rhinelander & Dawson 2004; Growcott et al. 2011).

1.3 How does CABLE work?

It is often difficult to obtain reliable IPI measures automatically. This is because the "true" IPI (that is, the IPI that corresponds to spermaceti organ length and ultimately body length,) is usually obscured by undesirable pulses when clicks are recorded off the whale's body axis (Zimmer *et al.* 2005). CABLE works around this problem by attempting to use IPIs only from clicks that appear to be on-axis. This approach makes it possible to compile reliable IPI measures from multiple whales clicking simultaneously, without the need to localize individuals or separate their click trains.

1.3.1 Filtering of IPIs

CABLE filters out unsuitable IPIs through a series of steps, which include automatic click classification, IPI precision testing, and IPI repetition testing.

- **Click classification**
  CABLE uses a machine learning technique to classify clicks. This involves the application of a classifier that has been trained to distinguish on-axis sperm whale clicks ("Good") from other transients ("Bad") based on a series of temporal and spectral features.

- **IPI precision**
  There are two common methods for computing IPIs automatically: autocorrelation analysis, and cepstral analysis (Goold 1996). CABLE uses both methods simultaneously, and computes the average IPI for each click. To reinforce confidence in the reliability of an IPI, it must be precise; that is, both methods must agree on the same value, within a certain tolerance.

- **IPI repetition**
  Sperm whales regularly produce about 1-2 clicks per second while foraging (Backus & Schevill 1966; Whitehead & Weilgart 1990). Since sperm whales also move relatively slowly, changes in click structure due to recording aspect are gradual. In other words, neighboring clicks in a time series are structurally very similar. Because of this, individual IPI measures are expected to be repeated locally in time. CABLE uses this property to further validate the reliability of each IPI.

1.3.2 Clustering of IPIs

After all IPIs have been filtered and compiled, CABLE examines the resulting distributions to infer how many sperm whales are present, and what their body sizes are. This is done by clustering IPIs using Gaussian mixture models (GMMs). Each cluster is considered to represent an individual whale, where the cluster mean is an estimate of the whale's "true" IPI. Cluster means are used to infer body length.

Gaussian mixture modelling often yields several plausible solutions. These solutions can be compared based on the Bayesian Information Criterion (BIC). The model with lowest BIC is the one that is most supported by the data. CABLE outputs each model ranked by lowest BIC. Each model has a reported ΔBIC value, where ΔBIC = 0 indicates the "best" model.

1.4 Limitations

CABLE does has a few issues that users should be aware of. This includes limitations to the IPI compilation method itself, and also minor inconveniences in the implementation.

1.4.1 Filter selectiveness

The quality control procedure that CABLE uses to isolate reliable IPIs is highly aggressive. With default parameters, **it is common for over 99% of clicks to be rejected**. There are two reasons for this. The first reason is that on-axis clicks are usually much rarer than off-axis clicks. The second reason is because the filter was deliberately designed to be stringent. This compromise was made because false positives are a much greater concern than false negatives in this case. The clustering algorithm is sensitive to noise, so a few false positives can lead to erroneous conclusions. At the same time, IPIs are precise enough that only a few true positives are needed to form a pattern. Since sperm whales click very frequently, it is generally not an issue to obtain enough samples for *something* to be detected.

Longer recordings are more likely to return useful results, but this is not a guarantee. The number of IPIs measured depends greatly on how much time a whale spends in alignment with the hydrophone, and this can vary considerably between recordings. When tested using 4-minute recordings with sperm whales present, CABLE yielded IPIs in several cases, but there were many others where it did not.

Resist the temptation to set filtration parameters to very low thresholds. This can result in spurious IPI distributions.

1.4.2 Interpretation of IPI clusters

As noted previously, the clustering algorithm is sensitive to noise: it will attempt to include every IPI it is given into a cluster. With default parameters, the IPI filtration routine is very good at eliminating spurious values, so this is generally not a problem. However, incorrect values do make it into the filtered distributions from time to time. For this reason, be cautious about drawing conclusions from very small clusters. As a general rule, clusters that contain very few IPIs (e.g., < 5) should not be trusted. If an IPI distribution contains many small clusters, a good course of action is to increase the filtration parameter values (discussed in section ***3.2 Parameters***). Clusters that persist under very strict parameter values are more likely to be real.

In regards to the selection of clustering models, BIC is quite good at finding the most correct model, but it is also not perfect. There are some cases where slightly less favoured models are actually more appropriate. When this does happen, it is often when there are two or more very narrow clusters (e.g., SD ≤ 0.02 ms) that are very close or overlapping with one another. In these cases, the multiple clusters could potentially be a single cluster. This uncertainty becomes greater when one or more of the clusters in question have very few IPIs. Therefore, a certain amount of judgement should be used when selecting models for ambiguous IPI distributions.

1.4.3 Recording quality

Reliable IPIs can only be obtained from high quality clicks where the multi-pulse structure is clear. Therefore, CABLE is unlikely to detect anything if recording quality is very poor. Factors

that complicate IPI calculation include low signal-to-noise ratio, surface reflections, and significant reverberation.

If recording from the surface, ensure that the hydrophone is as deep as possible. Surface echoes with a time delay of arrival shorter than an IPI will always result in contaminated clicks that will likely be classified as “Bad”. When this is the case, no amount of recording time will yield a useful IPI distribution.

1.4.4 Training dataset

The classifier CABLE uses to recognize on-axis sperm whale clicks was created using a “training” dataset. This dataset consisted of several thousand click samples, where each was manually labelled as being “Good” or “Bad”. All clicks in this dataset were recorded near the surface off the coast of Dominica, where only female and juvenile sperm whales were present. Because of this, it is possible that CABLE could have a bit more difficulty identifying “Good” clicks from different scenarios (e.g., large males, bottom-mounted recorders, etc.) This remains to be seen.

1.4.5 Other issues: software quirks and inconveniences

CABLE has a few issues in usability that unfortunately cannot be resolved. These are a consequence of CABLE being a compiled MATLAB application.

- **Loading and closing time**
  CABLE usually takes a very long time to start (sometimes more than one minute). This occurs because MATLAB Runtime needs to be loaded, and MATLAB Runtime is a very large application. If parallel computing is enabled (the default), then CABLE may also take a couple of seconds to close, making it appear unresponsive. This occurs because MATLAB needs time to close the "pool" it has created on the parallel computing cluster (which is just the local cores in this case).

- **Firewall warnings**
  If CABLE is using parallel computing for the first time, you may receive firewall warnings about "cable.exe" and "ctfxlauncher.exe". This is likely because MATLAB's parallel computing functions automatically request network access, should they need to find a particular computing cluster. Note that CABLE does not actually support this feature; only local CPU cores are used. Thus, these warnings do not really apply. CABLE will function normally whether or not you allow network access.

- **Console warnings**
  If you happen to run CABLE from the command prompt, you may see a few warnings. Some of these are simply updates returned by CABLE as it processes a file, such as "GMM invalid... Insignificant clusters". Others are internal to MATLAB and may be bugs. This includes "Could not launch SMPD process manager" and "Objects of fdopts.sosscaling class exist – not clearing this class or any of its superclasses". Neither of these seem to have any adverse effects on CABLE.

1.5 Feedback and bug reports

The latest version of CABLE is 1.0. This is the first public release. The algorithm has now been peer-reviewed, and development and beta tests have found no major bugs in the software.

While bugs are unlikely, it cannot be guaranteed that the software is completely bug-free. In the case of unexpected crashes or failures, CABLE will attempt to record the error details in a text file. If you do encounter any bugs while using CABLE, or if you have suggestions for improving its usability, please contact the developer at wilfried.beslin@dal.ca. In the case of bug reports, please include the error files, if any.

Since this version was built entirely based on data from Dominica, I would also be very interested to know how well it performs in other regions – especially if you are recording mature males, or if you use bottom-mounted hydrophones.

1.6 Authors and Citation

This program was developed by Wilfried Beslin as part of a Master of Science thesis at Dalhousie University, Halifax, Nova Scotia, Canada. It was developed under the supervision of Hal Whitehead, with the collaboration of Shane Gero. This work emanates from The Dominica Sperm Whale Project: http://www.thespermwhaleproject.org Follow: @DomWhale

The algorithm used by CABLE is described in:

> Beslin, W. A. M., Whitehead, H., and Gero, S. (in press). "Automatic acoustic estimation of sperm whale size distributions achieved through machine recognition of on-axis clicks". *The Journal of the Acoustical Society of America*.

Please cite this article if you use CABLE.

2. USAGE OVERVIEW

2.1 Requirements and installation

CABLE is a standalone application built for the Windows operating system using MATLAB Compiler and MATLAB version R2015a. It will only work on Windows operating systems (64-bit).

MATLAB itself is not needed to run CABLE. However, unless you happen to have a full installation of MATLAB R2015a and MATLAB Compiler SDK, **you will need MATLAB Runtime**. MATLAB Runtime is made freely available by The Mathworks, Inc. at:

https://www.mathworks.com/products/compiler/matlab-runtime.html.

Before running CABLE, **make sure you have downloaded and installed MATLAB Runtime R2015a (8.5)**.

CABLE itself comes in a "zip" folder and does not require installation. Simply place the program directory anywhere where you have write and execute permissions. The program directory should contain each of the following files and folders:

- ***CABLE.exe***
  The main executable. Use this to run CABLE.

- ***Examples***
  Directory containing sample input files for demonstration (see section ***3.1.3 Example input files***).

- ***LengthEquations***
  Directory used for storing data to convert IPIs to body length (see section ***3.3.4 Length estimation***).

- ***Output***
  An initially empty directory, provided as the default for storing output files (see section ***3.3 Data output***). Files inside this directory may be deleted without harm.

- ***UserManual.pdf***
  This document

- ***splash.png***
  An image displayed while the application is loading.

2.2 Running the program

Run CABLE by clicking *CABLE.exe* in the program directory. Note that it may take a (very) long time to load. Unfortunately, there is no simple way around this – it is a limitation of MATLAB applications (see section ***1.4.5 Other issues: software quirks and inconveniences***).

The program is controlled via a (non-resizable) graphical user interface (GUI). The GUI window contains three tabs: *Main*, *Plotting*, and *Advanced Settings*.

- ***Main***
  This tab is where IPI compilation is set up and run. It also displays the current status of the application. See section ***3. IPI Compilation*** for details.

- ***Plotting***
  This tab controls the display, appearance, and saving of IPI cluster plots. See section ***4. Cluster Plotting*** for details.

- ***Advanced Settings***
  This tab is where advanced parameters affecting the IPI compilation routine can be set. It is generally not necessary to access this tab. See section ***5. Advanced Settings*** for details.

All tabs provide options for entering data, including editable text fields. If an entry in a text field is invalid, the field will turn red and may be accompanied by a warning message. Certain functions will not execute until all their dependent fields contain valid entries.

If you are using CABLE for the first time, it is recommended that you try running some of the example files before doing anything else. One of these files, a 4 min and 45 sec WAV file of sperm whales from Dominica, is pre-loaded on start-up. To see what CABLE can do, just press the large "Run" button on the *Main* tab. An IPI distribution will be compiled from the pre-loaded file. On a modest laptop computer, this file takes about 4 minutes to process. For more information on the other example files, see section ***3.1 Data input***.

3. IPI COMPILATION

IPI compilation is controlled in the *Main* tab (**Figure 1**). Use this window to specify input data, the types of output desired, and run the routine.

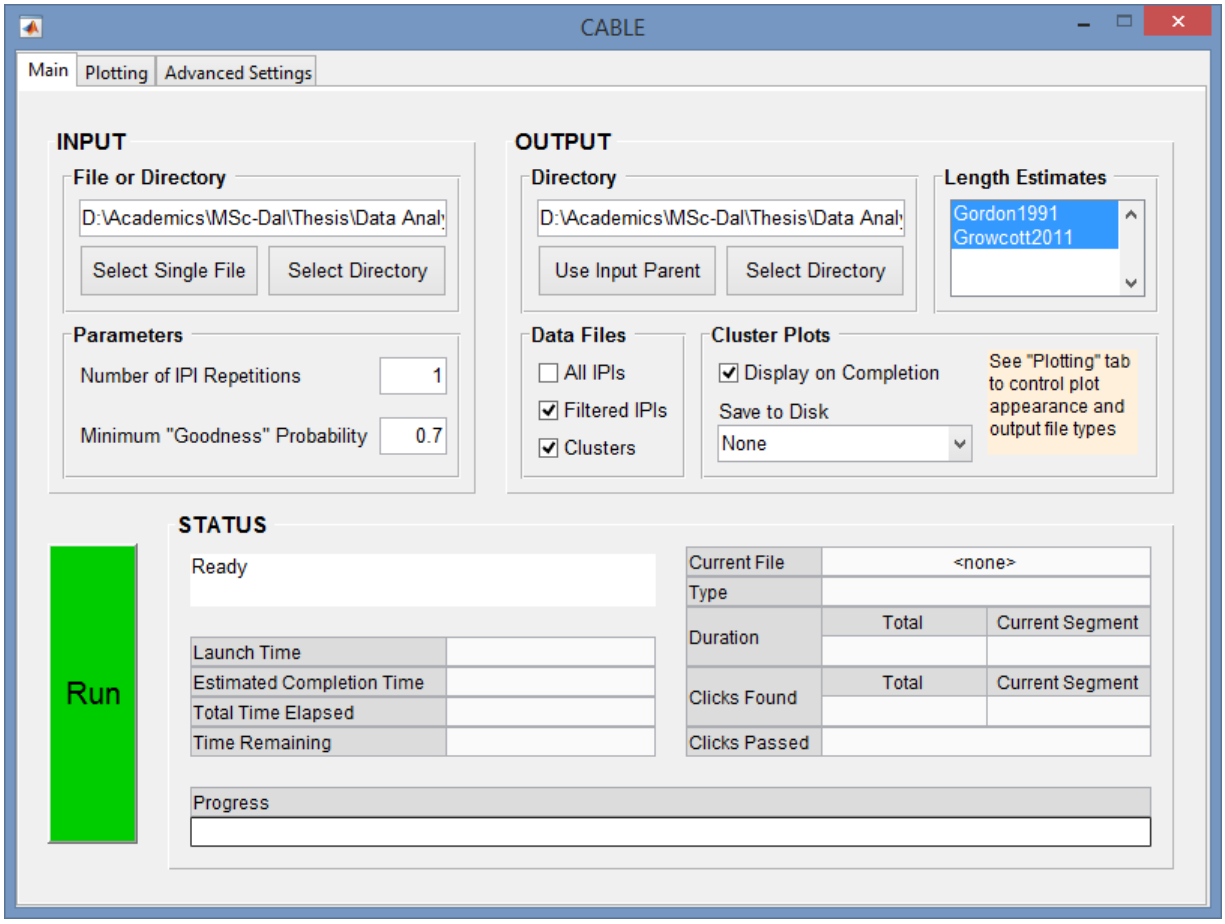

Figure 1: The *Main* tab

3.1 Data input

Data to input for IPI compilation is specified under the *INPUT > File or Directory* panel at the top left. Input is specified in the form of a path string to a file on your hard drive. When IPI compilation starts, the program will attempt to read the contents of the specified path.

3.1.1 Types of input

The program is capable of processing various types of input. The input path may point to an individual file, or to a folder containing multiple files. The folder option is designed to facilitate automation: each file within will be processed independently, effectively the same as if they had been specified manually one after the other. Any corrupted or unrecognized files will be skipped. The accepted file types are listed below:

- **Audio file**
  The most basic input option is a WAV sound file* containing sperm whale clicks recorded in the field. CABLE will scan these files and detect clicks automatically. These

clicks will be subsequently classified as being either on-axis sperm whale clicks or not. Audio inputs can have any number of channels, but only one channel is processed (usually the first, but this can be changed; see section ***5. Advanced Settings***). Audio inputs can also have any sampling rate, but note that IPI compilation is always done at 48 kHz. If a file's sampling rate is different from 48 kHz, it will be resampled automatically.
* The program has only been tested using WAV files. It is possible that other sound file types will work too, but these are not officially supported.

- **Full IPI file**
  One of the outputs that can be created by CABLE after processing an audio file is a full IPI file (see section ***3.3.2 Output data files***). A full IPI file consists of a list of IPIs for every click that has been detected in an audio file, along with their time of occurrence and classification score. This type of file can also be used as input. When a full IPI file is input, the routine will filter those IPIs based on the filtration parameters. Full IPI files have the advantage that they are very quick to process, since clicks have already been detected and classified.

- **Filtered IPI file**
  Another type of output created by CABLE is a filtered IPI file (see section ***3.3.2 Output data files***). A filtered IPI file consists of a list of IPIs that have already been validated. These can be used as input for clustering. Filtered IPI files are the quickest to process, but also the least flexible, since their filtration parameters cannot be changed.

- **Audio Directory**
  When the input path points to a directory, the files within are processed separately. However, using this option, it is also possible to group audio files together. To do this, create a nested directory within the main input folder. All files within this subdirectory will be pieced together as if they were one file. Note that this is only supported for audio files; full and filtered IPI files cannot be grouped.

3.1.2 Specifying the input path

There are three ways to specify the input path:

- **Direct Entry**
  The input path can be entered directly in the edit field inside the *INPUT > File or Directory* panel.

- ***Select Single File*** **button**
  Use this to browse for a single file to input.

- ***Select Directory* button**
  Use this to browse for a folder to input.

3.1.3 Example input files

Included in the program directory are several example files intended to demonstrate every type of input. These files consist of sperm whale recording data taken off the coast of Dominica in March and April 2015, as part of the Dominica Sperm Whale Project (DSWP). The files are located in the *Examples* folder, and include the following:

- *DOM_20150302_AllIPIs.csv* (Full IPI file)
- *DOM_20150302_FilteredIPIs.csv* (Filtered IPI file)
- *DOM_20150319.wav* (Audio file)
- *DOM_201504* (Audio directory; contents listed below)
  - *DOM_20150406.wav*
  - *DOM_20150407a.wav*
  - *DOM_20150407b.wav*

You can run the WAV and CSV files individually, or you can set the whole *Examples* folder as the input path. If you run the *Examples* folder, all the files in the *DOM_201504* subfolder will be treated as a single recording. You could also run the *DOM_201504* folder directly, but in this case, all the files within will be treated separately.

To keep the *Examples* and program directories clean, it is recommended that you do not use the *Use Input Parent* button (discussed in section ***3.3.1 Specifying the output path***).

3.2 Parameters

IPI filtration depends primarily on two parameters, which can be set in the *INPUT > Parameters* panel. These parameters include number of IPI repetitions, and minimum "Goodness" probability.

- ***Number of IPI Repetitions***
  The IPI repetition parameter dictates how many times an IPI must be locally repeated for it to be considered valid. A value of 0 indicates that IPIs do not need to be repeated. This parameter must be a non-negative integer.

- ***Minimum "Goodness" Probability***
  Detected clicks are automatically classified as being "Good" (i.e. on-axis sperm whale clicks) or "Bad" (i.e. off-axis clicks, dolphin clicks, noise, etc.). To do this, the classifier assigns a probability to each click indicating how "Good" it is. The minimum "Goodness" probability is the threshold that determines which label each click will receive. For example, if set at 0.7, then all clicks with a probability greater than or equal to 0.7 will

be classified as “Good”, and everything else will be “Bad”. This parameter must be a number between 0 and 1 inclusive.
NOTE: some features used by the classifier depend on pulse detection, and can only be calculated if a click has at least two pulses. All clicks with fewer than two pulses are immediately considered “Bad”. Therefore, setting the minimum “Goodness” probability to 0 does not disable classification completely, because those clicks with too few pulses will still be filtered out.

Of these two parameters, *Number of IPI Repetitions* has the strongest impact. *Minimum “Goodness” probability* usually has subtle effects, unless set to extreme values (< 0.1 or > 0.9).

3.3 Data output

An output directory is created during each IPI compilation run. The name of this directory is given automatically: it is equal to the name of the input file or folder, followed by “_IPIData#”, where # is a positive integer. The output structure depends on the type of input (file or directory), the output path, and whether or not outputs already exist.

- **Differences based on input type**
  If the input path points to a single file, each output file will be written directly to the output folder. However, if the input path points to a folder, then the output folder will contain a subdirectory for each input file. These subdirectories will have exactly the same name as their corresponding inputs.

- **Output path**
  The main output directory will be created inside the folder pointed to by the output path specified by the user.

- **Presence of existing output**
  Existing output is not overwritten. If the output path already contains a folder for a given input, then a new folder will be created each time the input is reprocessed. These multiple outputs are differentiated by the numeric suffix at the end of the folder name. So for example, if some input file “example.wav” is processed for the first time, the name of its output directory will be “example_IPIData1”. If it is run again with the same output path, then a new folder will be created called “example_IPIData2”.

Several output files will be created for each input file. These can be selected in the output settings (discussed in later sections). However, in all cases, a *log.txt* file will be created for each input file. This log file contains information on the input file, any problems that were encountered, and the input parameters specified. If CABLE runs into any problems while processing input, please include this log file in your report.

3.3.1 Specifying the output path

Output folders will be created inside the directory pointed to by the output path. There are three ways to specify this path:

- **Direct Entry**
  The output path can be entered directly in the edit field inside the *OUTPUT > Directory* panel.

- ***Use Input Parent* button**
  This will automatically assign the output path to point to the directory in which the input file or folder is located.

- ***Select Directory* button**
  Use this to browse for a folder to assign as the output path.

Note that input and output should be kept separate. If the input is a folder, the output path will be considered invalid if it points to that same folder, or any folder within. In other words, output may not be nested inside an input folder.

3.3.2 Output data files

CABLE can produce three different types of files as output. Whether these are created or not depends on whether or not they are checked in the *OUTPUT > Data Files* panel. The options are described below:

- ***All IPIs***
  This will create a CSV file containing IPI values (in milliseconds) for each click that was detected. Along with this is the time of occurrence of each IPI (in seconds), its classification probability, and whether or not it is precise. This type of file can be used as input to quickly examine the effects of different filtration criteria (see section ***3.1 Data input***). However, note that specifying this output will increase execution time, because IPIs will need to be calculated for all clicks.

- ***Filtered IPIs***
  This will create a CSV file containing only those IPI values that have been validated by all steps in the filter (i.e. classification, precision, and repetition). IPIs are reported in milliseconds. It also includes occurrence times in seconds. This type of file can be used as input for quick clustering.

- ***Clusters***
  This will create one CSV file for every plausible Gaussian mixture model, as well as an extra file listing the ΔBIC values for each model. The model files include the means (*mu*), standard deviations (*sigma*), and component proportions (*p*) for every cluster. Means

and standard deviations correspond to the IPI scale and are measured in milliseconds. If length estimates were also specified, those will be included in these files too (see section ***3.3.4 Length estimation***).

3.3.3 Output cluster plots

Once CABLE has finished compiling and clustering IPIs for a file, the results can be displayed in a series of plots. These plots can be displayed immediately on completion, and also saved to disk as image files. Plot appearance, image file types, and other options are controlled under the Plotting tab. If one of the Save to Disk options is enabled, image files will be created in the output directory for each of the plots specified. Note however that plots will only be created if there exists data (i.e., IPIs or valid models) to produce them. Cluster plots can also be recreated at any time from cluster output files. Details are explained in section ***4. Cluster Plotting***.

Note that CABLE can only plot data for one input file at a time. If the input is a directory and cluster plots are set to be displayed on completion, a plot will be displayed only for the most recently completed file. It is also not possible to manipulate plots while the IPI compiler is running. For these reasons, there is little advantage to enabling the *Display on Completion* option when the input is a folder.

3.3.4 Length estimation

Body length estimates can be included in the cluster output files. These estimates are made based on the mean IPI (*mu*) of each cluster. Length estimation equations can be selected in the *OUTPUT > Length Estimates* listbox. It is possible to include more than one equation, or none at all (press *Ctrl + left click* to deselect entries).

Length estimation equations are limited to polynomials. They are stored as lists of polynomial coefficients, in order of decreasing power, inside text files within the *LengthEquations* folder. One file exists for each equation, where the file name dictates the equation name. With this system, it is possible to extend CABLE with your own length polynomials. By default, CABLE comes with two equations that convert IPIs (in milliseconds) to meters. These are published equations that were derived from regression analysis of IPI measures and photogrammetrically estimated body lengths:

- Gordon (1991): $y = -0.001x^2 + 1.453x + 4.833$
  Based on 11 female sperm whales off Sri Lanka

- Growcott *et al.* (2011): $y = 1.258x + 5.736$
  Based on 33 large male sperm whales off Kaikoura, New Zealand

3.4 Running IPI compilation

Once input and output files have been specified, run CABLE by clicking the large green *Run* button. While the routine is running, the *Plotting* and *Advanced Settings* tabs will be disabled. The *Run* button will also be replaced with a red *Abort* button. Click this button to cancel the run at any time. Depending on the current step of the routine, it may take a few seconds for cancellation to take effect. **Figure 2** shows an example of what CABLE looks like while compiling IPIs.

Audio files can be quite large, which requires a lot of memory to process. If memory becomes limited, the IPI compilation routine can become extremely slow and even fail. To avoid this problem, CABLE analyzes audio files in separate segments. The results are pieced back together after IPIs have been calculated.

The status of the routine is displayed in the white message box at the top left of the *STATUS* panel. The *STATUS* panel also includes information on execution time, and the file currently being worked on.

- **Execution time**
  The routine will display the time at which it started, and the amount of time it has been running for. If more than one file is being processed, it will also display an estimated time of completion, which gets updated as each file finishes. However, note that the amount of time the routine takes to process a file depends very heavily on the number of clicks found in that file. Therefore, if you are processing a small number of files that vary greatly in click density, estimated completion time can be inaccurate.

- **File info**
  CABLE will display information on the current file being processed. This includes its name, type (audio, full IPI file, filtered IPI file, etc.), duration, number of clicks detected, and number of clicks that passed every filtration step. Duration and number of clicks found is displayed for both the whole file and the segment being analysed.

- **Progress bar**
  IPI compilation progress is also displayed visually with a progress bar. The bar is divided based on how many files are being processed, and how many segments they are broken into. The bar is coloured according to the status of each file. **Figure 2** explains this in further detail.

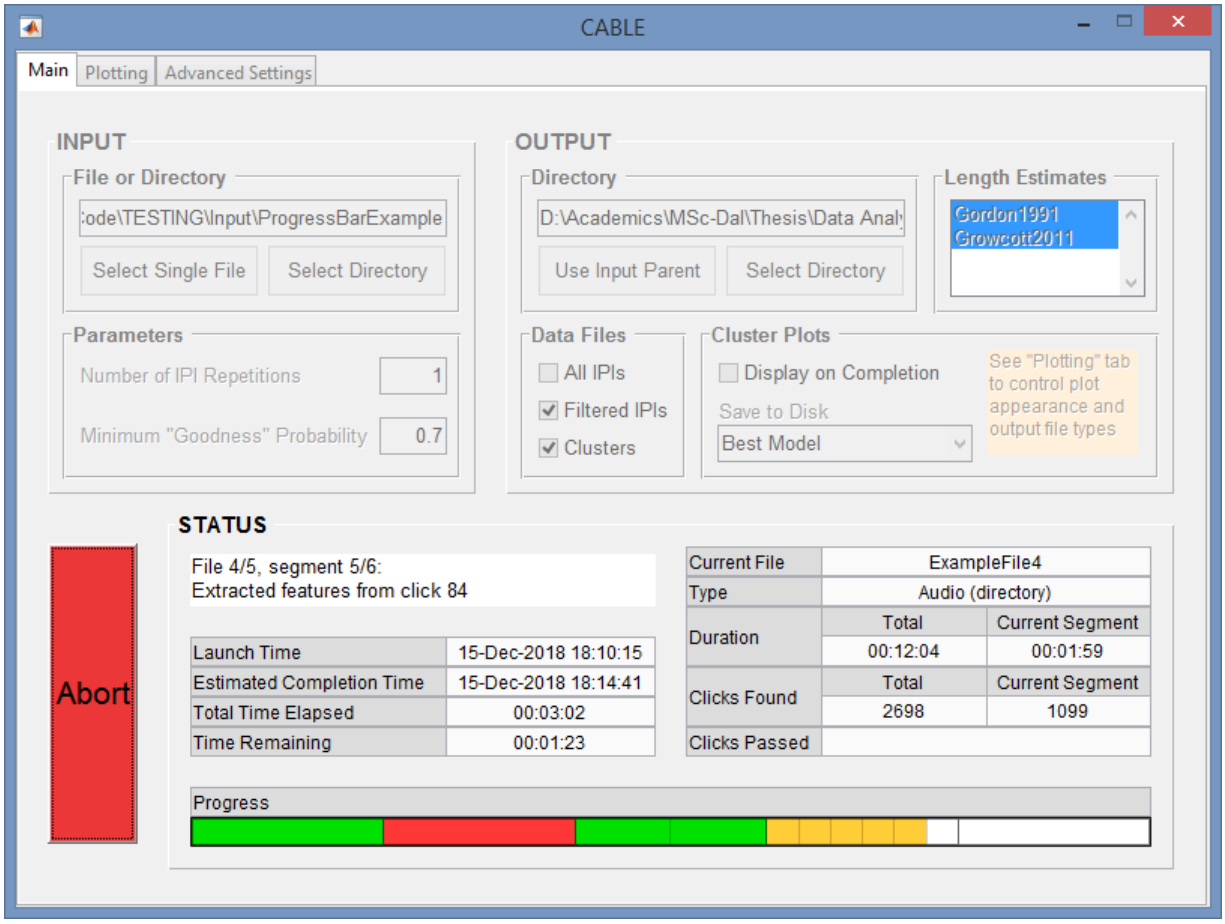

Figure 2: The *Main* tab during IPI compilation. In this example, the input path points to a folder containing 5 files. Dark lines on the progress bar indicate file divisions, while light lines indicate segment divisions. The progress bar colour code is as follows:

- Green: finished file
- Red: failed or unsupported file
- Yellow: file in progress

4. CLUSTER PLOTTING

CABLE can show the IPIs it has compiled and clustered as a series of plots. These plots include a histogram of the filtered IPI distribution, and continuous lines depicting the probability distributions of clusters found by Gaussian mixture modelling (see **Figure 3**). Cluster plotting is controlled via the *Plotting* tab (**Figure 4**).

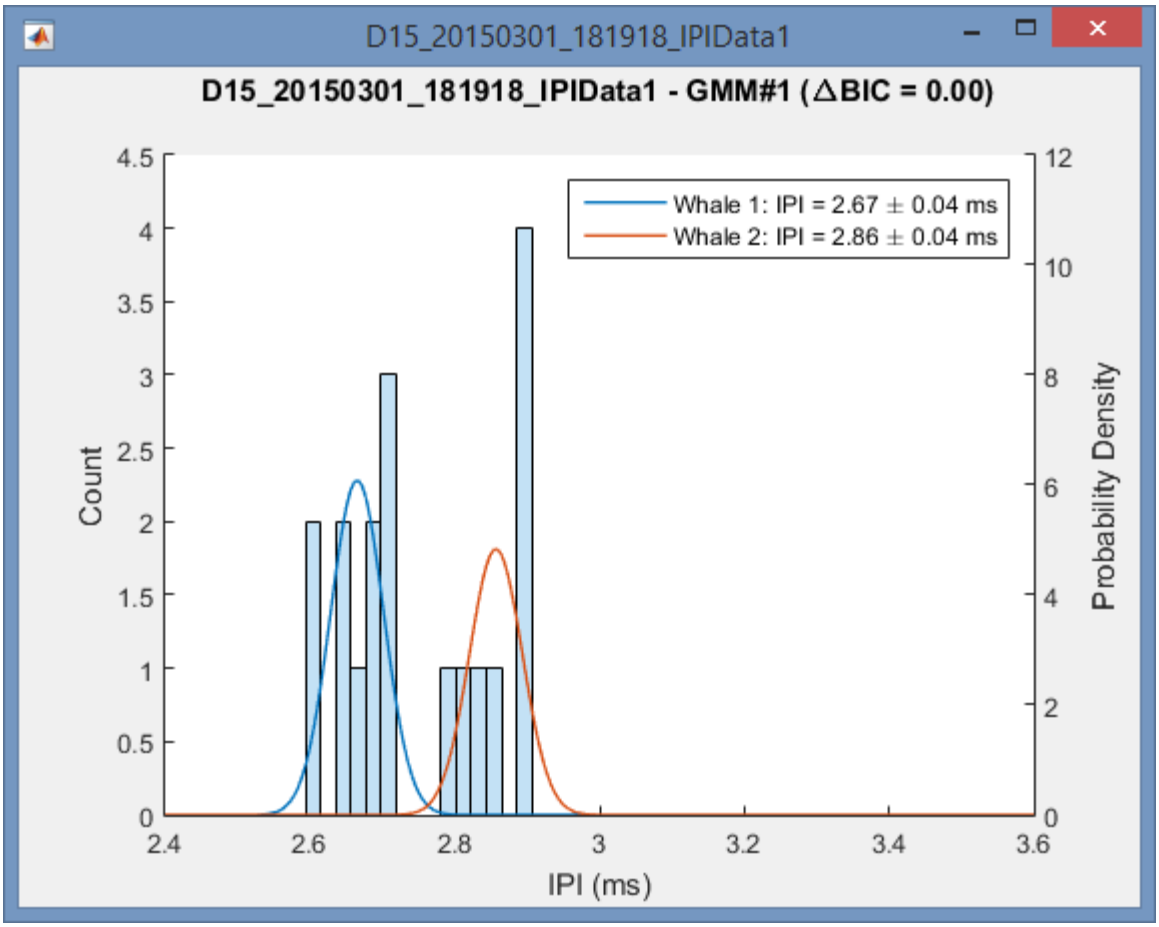

Figure 3: Example cluster plot. The histogram corresponds to the distribution of IPIs that passed every filtration criterion. Bin width is equal to the sampling resolution of IPI analysis (that is, 1/48 ms). The coloured lines correspond to the probability distributions of individual clusters in a Gaussian mixture model. Only one model is plotted at a time.

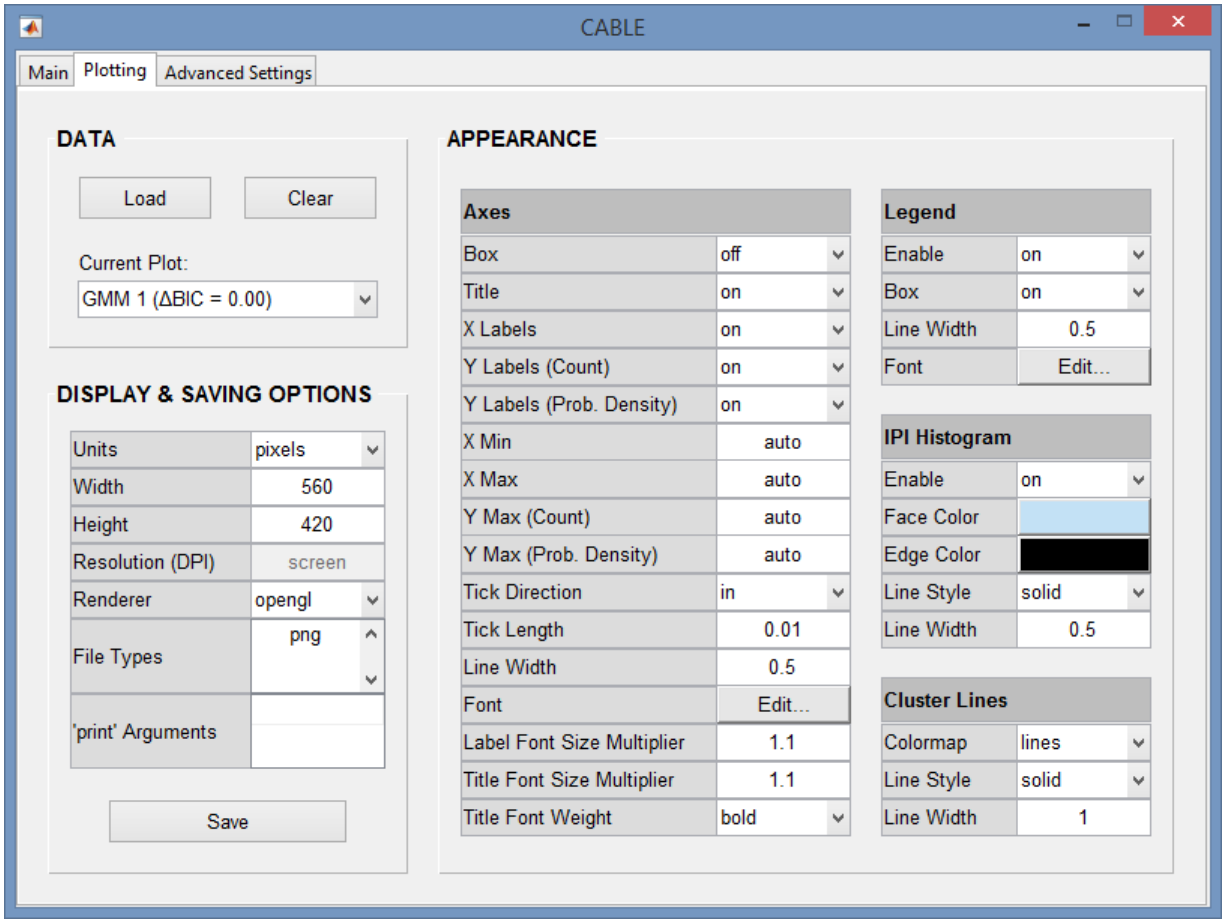

Figure 4: The *Plotting* tab

4.1 Loading and displaying plots

Plots can be set to display automatically after CABLE has finished processing a file, but they can also be created at any time. To do this, press the Load button in the *DATA* panel of the *Plotting* tab, and select an output directory created by CABLE. For plots to be created, this directory must contain a Filtered IPI file and associated cluster files. To delete a plot, press the *Clear* button in the *DATA* panel, or the X button on the plot window. Data can only be plotted for one file at a time.

Gaussian mixture modelling may yield more than one solution. ΔBIC indicates the relative likelihood of a model, where a value of 0 corresponds to the "best" model. CABLE's cluster plotter displays each solution as a separate plot. You can select the model being displayed using the popup menu in the *DATA* panel. The filtered IPI distribution alone (with no model fits) is also a display option.

4.2 Plot display and saving options

The *DISPLAY & SAVING OPTIONS* panel allows you to control things such as the size of the plot window, how the plot is rendered, and how to save it. Internally, figure saving is done using the MATLAB 'print' command, which is a very flexible function. CABLE's options essentially provide an interface to 'print'. For casual save operations, it is not necessary to understand how this function works. However, if you need to use the more advanced options, then it helps to know 'print'. The documentation for 'print' is available here:

https://www.mathworks.com/help/matlab/ref/print.html.

Note however that documentation is publically available for the latest version of MATLAB only. CABLE 0.3 uses MATLAB version R2015a. Changes to 'print' in newer MATLAB versions are probably unlikely, but this cannot be guaranteed. CABLE's plot display and saving options are described below.

- ***Units***
  The units in which the figure dimensions are specified. Options are "pixels", "centimeters", and "inches".

- ***Width***
  The figure width, in the units specified by the *Units* option.

- ***Height***
  The figure height, in the units specified by the *Units* option.

- ***Resolution (DPI)***
  The resolution of the figure when saved to an image file in DPI, or dots (pixels) per inch. This can be any integer greater than 0, or the word 'screen' to use screen resolution. If the figure units are in pixels, this is limited to screen resolution. Higher resolution results in better quality figures, but larger file sizes. For publication-quality figures, MATLAB suggests using 200 or 300 DPI.

- ***Renderer***
  The rendering method to use for both displaying and saving images. Options are 'opengl' and 'painters'. In general, 'opengl' should be used for raster images, and 'painters' should be used for vector images. Other than this, it is usually not necessary to change the renderer, unless you encounter rendering artefacts.

- ***File Types***
  This is a list of file types in which to save the figure. Note that these are not just file extensions (so for example, for JPEG, use 'jpeg' instead of 'jpg'). Note also that these are not validated until runtime. Any invalid entries will simply be ignored, so be sure that your entries are correct. Some common formats are listed below:
  - Raster formats
    - 'png' (24-bit PNG)
    - 'jpeg' (24-bit JPEG)
    - 'bmp' (24-bit BMP)
  - Vector formats
    - 'tiff' (compressed 24-bit TIFF)
    - 'pdf' (full-page colour PDF)
    - 'svg' (Scalable Vector Graphics)
    - 'epsc2' (Encapsulated PostScript Level 2 with colour)

  For the full list of supported file types, see the MATLAB documentation for 'print' (the list is under the *formattype* parameter).

- ***'print' Arguments***
  This is an advanced option that allows you to specify any additional arguments you wish to MATLAB's 'print' function. Enter each option in sequence, separated by a space These options must also include the '-' prefix. The rows in this field correspond to file types. As an example, say for some reason you wanted to save two files: a PNG, and a black-and-white EPS that included a TIFF preview and used a loose bounding box. The *File Types* and *'print' Arguments* fields would look as shown in **Figure 5**.

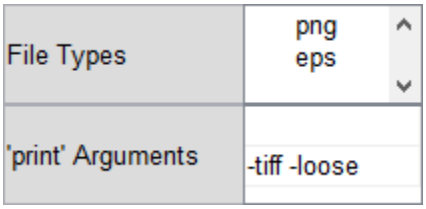

Figure 5: Example inputs for the *File Types* and *'print' Arguments* fields in the cluster plotter's *DISPLAY & SAVING OPTIONS* panel.

To save a figure from the *Plotting* tab, press the *Save* button at the bottom of the *DISPLAY & SAVING OPTIONS* panel. You can choose to save images for all models, or only select models. The files will be saved inside the same output directory containing the data used to create the plots.

4.3 Adjusting plot appearance

Plot appearance is highly customizable. Adjust the values in the *APPEARANCE* panel to make the plot look the way you want. The options are described below.

- ***Axes Properties***

    - ***Box***
      Specifies if the axes should have a border or not.

    - ***Title***
      Specifies if the title should be displayed or not.

    - ***X Labels***
      Specifies if the X-axis label and tick values should be displayed or not.

    - ***Y Labels (Count)***
      Specifies if the count axis and its tick values should be displayed or not. The count axis always appears on the left.

    - ***Y Labels (Prob. Density)***
      Specifies if the probability density axis and its tick values should be displayed or not. If both the count and probability density axes exist, then the probability density axis is displayed on the right. If only the probability density axis exists, it is on the left.

    - ***X Min***
      Determines the lower limit of the X-axis. This may be any non-negative number smaller than *X Max*, or the strings 'auto' or 'full'. 'auto' sets automatically determined limits appropriate for the data. 'full' sets the limits equal to the *IPIRange* parameter in the *Advanced Settings* tab.

    - ***X Max***
      Determines the upper limit of the X-axis. This may be any nonnegative number greater than *X Min*, or the strings 'auto' or 'full'. 'auto' sets automatically determined limits appropriate for the data. 'full' sets the limits equal to the *IPIRange* parameter in the *Advanced Settings* tab.

    - ***Y Max (Count)***
      Determines the upper Y limit based on the count axis. This may be any positive number, or the string 'auto'. 'auto' sets an automatically determined limit appropriate for the data. If *Y Max (Count)* is adjusted, then *Y Max (Prob. Density)* will automatically be adjusted accordingly.

- ***Y Max (Prob. Density)***
  Determines the upper Y limit based on the probability density axis. This may be any positive number, or the string 'auto'. 'auto' sets an automatically determined limit appropriate for the data. If *Y Max (Prob. Density)* is adjusted, then *Y Max (Count)* will automatically be adjusted accordingly.

- ***Tick Direction***
  Specifies if tick marks should point inside the plot area, or outside.

- ***Tick Length***
  Determines the length of all marks.

- ***Line Width***
  Determines the thickness of axes lines and tick marks.

- ***Font***
  Determines the fonts to be used for the plot title, labels, and tick values. Press the *Edit…* button to change the font type, weight, and size.

- ***Label Font Size Multiplier***
  Determines how much larger (or smaller) the axis labels should be relative to the base font size.

- ***Title Font Size Multiplier***
  Determines how much larger (or smaller) the title should be relative to the base font size.

- ***Title Font Weight***
  Specifies if the title should be displayed in bold or normal font weight.

- ***Legend***

  - ***Enable***
    Specifies if the legend should be displayed or not.

  - ***Box***
    Specifies if the legend should be enclosed in a border or not.

  - ***Line Width***
    Determines the thickness of the legend border lines.

- ***Font***
  Determines the fonts to be used for the legend labels. Press the *Edit…* button to change the font type, weight, and size.

- ***IPI Histogram***

  - ***Enable***
    Specifies if the IPI histogram should be displayed or not.

  - ***Face Color***
    Determines the colour of the histogram. Click the coloured button to change colour.

  - ***Edge Color***
    Determines the colour of the histogram outline. Click the coloured button to change colour.

  - ***Line Style***
    Determines the appearance of the histogram outline (e.g. solid or dotted). It is not possible to remove lines, but if you want to achieve this effect, you could make *Face Color* and *Edge Color* the same.

  - ***Line Width***
    Determines the thickness of the histogram outline.

- ***Cluster Lines***

  - ***Colormap***
    Specifies the colour scheme used for plotting cluster lines. Options are limited to select built-in MATLAB colour maps, which are listed within this documentation page: https://www.mathworks.com/help/matlab/ref/colormap.html.
    Note that MATLAB occasionally modifies the colours in newer versions, so the colours displayed in the documentation page might not be exactly the same as the ones in CABLE.

  - ***Line Style***
    Determines the appearance of the cluster lines (e.g. solid or dotted).

  - ***Line Width***
    Determines the thickness of the cluster lines.

5. ADVANCED SETTINGS

In addition to the two basic parameters presented in the *Main* tab, there are several other factors that control how IPI compilation is run. These are accessible via the *Advanced Settings* tab (see **Figure 6**). Advanced parameters are grouped into categories, which are viewable via subtabs.

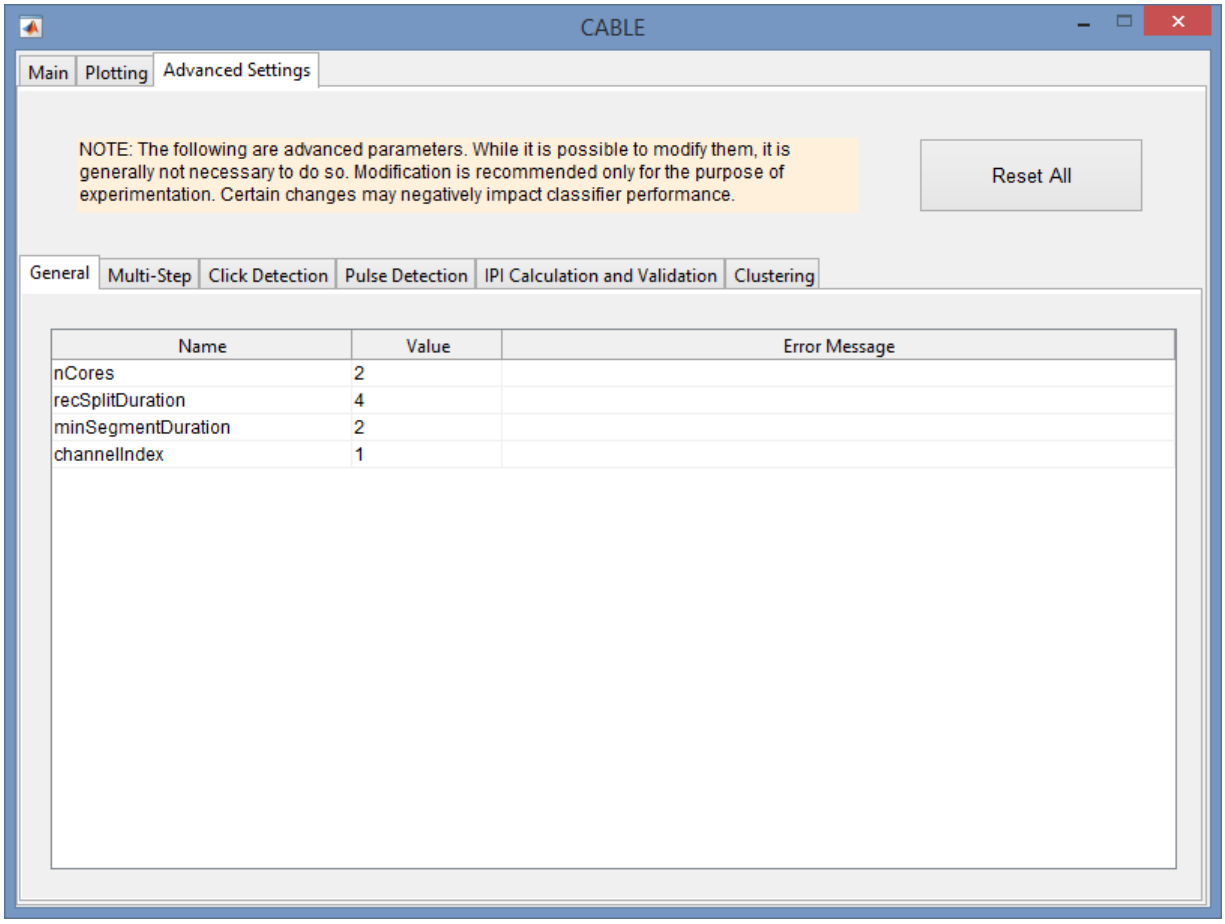

Figure 6: The *Advanced Settings* tab

With the exception of the *General* parameters, it is discouraged to modify advanced parameter values during casual use. The default values for these parameters have been worked out thoroughly, and should be robust to most scenarios. They should be modified only for the sake of experimentation. To restore the default values, click on the *Reset All* button.

Advanced parameters are presented in a table with three fields: *Name*, *Value*, and *Error Message*. To change a parameter value, edit the *Value* field. If an edition is invalid, the reason will be printed in the corresponding *Error Message* field. IPI compilation will not be able to run if any parameter is invalid.

Each parameter is described below.

- ***General***

    - ***nCores***
      The number of CPU cores to use for parallel processing. Note that this corresponds to the number of physical cores, not logical cores. Setting this to 1 disables parallel computing.

    - ***recSplitDuration***
      The duration at which to split audio files into segments, in minutes. For example, if set to 4, then audio files will be split into 4-minute intervals. Must be larger than *minSegmentDuration*.

    - ***minSegmentDuration***
      The minimum allowed duration of an audio segment, in minutes. If audio splitting based on *recSplitDuration* would result in a segment that is shorter than *minSegmentDuration*, then that split is not performed. For example, if *recSplitDuration* is 4 and *minSegmentDuration* is 2, and a 9-minute file is being processed, then that file will be split into 2 segments: one 4 minutes long, and one 5 minutes long. It is not split into 3 segments, because then the last segment would be 1-minute long, which is shorter than *minSegmentDuration*. This parameter cannot be larger than *recSplitDuration*.

    - ***channelIndex***
      This controls which channel of an audio file to use. If this number is larger than the number of channels in an audio file, then that file will not be processed successfully.

- ***Multi-Step***

    - ***IPIRange***
      The minimum and maximum limits of the range within which IPIs are calculated, in milliseconds. Note that if there exist IPIs outside this range, then those IPIs might either be missed (if too short), or have incorrect values (if too large). The default range is that recommended by Marcoux *et al.* (2006).

    - ***pulseDurationRange***
      The minimum and maximum expected duration of individual pulses within sperm whale clicks. Units are in milliseconds. This parameter affects both click detection and pulse detection.

- ***tukeyFalloffDuration***
  To minimize ringing artefacts in click spectra, a Tukey window is applied to each click before performing the FFT, where the flat region encompasses the entire range of a detected click. This parameter controls the extent of the window's falloff regions, measured in milliseconds. Setting this to 0 is equivalent to using a rectangular window.

- ***Click Detection***

  - ***threshOn***
    The linear SNR threshold which controls the onset of click detection events.

  - ***threshOff***
    The linear SNR threshold which dictates the limits of a click range.

  - ***alphaSignal***
    Smoothing factor for the exponential signal power estimation filter.

  - ***alphaNoiseOn***
    Smoothing factor for the exponential noise power estimation filter, used while a click has been detected.

  - ***alphaNoiseOff***
    Smoothing factor for the exponential noise power estimation filter, used while only noise is present.

  - ***minEchoProp***
    The minimum proportion of a click envelope peak beyond which the next click will be considered an echo. Setting this too low may break up multi-pulsed clicks, while setting it too high may cause clicks to merge with their reflections.

  - ***maxClickDuration***
    The maximum allowed duration of a click, in milliseconds.

  - ***minClickSep***
    The minimum expected separation between clicks, in milliseconds. This parameter may help reduce the detection of echoes, but it only makes sense to use if a single whale is present. By default, it is set to 0.

- ***Pulse Detection***

    - ***smoothBandwidths***
      A vector of bandwidths to use for waveform envelope smoothing, from narrowest (small) to widest (large). Units are in milliseconds.

    - ***nSmoothRuns***
      The number of times to run the envelope smoothing filter in succession.

    - ***peakBaseHeightProp***
      The proportion of the peak-to-base height of a pulse to use as the reference for estimating pulse durations.

    - ***promThreshProp***
      A parameter used to determine the minimum prominence that a peak in the smoothed envelope may have to indicate the presence of a pulse. It is a proportion of the difference between the prominences of the most and least prominent peaks in the smoothed envelope.

    - ***minPromThreshScale***
      A scale factor relative to the RMS of the absolute value of the difference between successive samples in the unsmoothed envelope. This is meant to provide a measure of peak prominences for peaks which arise only from noise, and is thus the minimum allowable prominence threshold.

- ***IPI Calculation and Validation***

    - ***doMethod_Autocorrelation***
      A Boolean value that determines if IPIs should be computed using the autocorrelation method or not. Valid entries are ‘true’ or ‘false’.

    - ***doMethod_Cepstrum***
      A Boolean value that determines if IPIs should be computed using the cepstrum method or not. Valid entries are ‘true’ or ‘false’.

    - ***useChiSquared_Autocorrelation***
      A Boolean value that determines if χ^2 windowing should be used before computing IPIs from autocorrelation. χ^2 windowing was proposed by Goold (1996) as a method to amplify the signal of subsequent pulses in sperm whale clicks.

- ***useChiSquared_Cepstrum***
  A Boolean value that determines if χ^2 windowing should be used before computing IPIs from the cepstrum. χ^2 windowing was proposed by Goold (1996) as a method to amplify the signal of subsequent pulses in sperm whale clicks.

- ***maxIPIDeviation***
  The maximum acceptable difference between the IPI values of a click that were computed using different methods (i.e. autocorrelation and cepstrum). This parameter is what controls IPI precision. Any clicks whose individual IPI estimates differ by a larger amount than *maxIPIDeviation* are rejected by the IPI filtration routine. Units are in milliseconds.

- ***ICIRange***
  The minimum and maximum inter-click interval (ICI) limits within which successive clicks are searched for. This is used when checking for IPI repetitions. Units are in seconds.

- ***ICITol***
  The maximum expected difference in ICI between successive clicks in a click train. This is used only when checking for 2 or more IPI repetitions. Units are in seconds.

- ***IPITol***
  The maximum expected difference in IPI between successive clicks in a click train. This is used when checking for IPI repetitions. Units are in milliseconds.

- ***Clustering***

  - ***KDEBandwidths***
    Narrow and wide bandwidths used during Gaussian kernel density estimation (KDE). KDE is used to get an initial estimate of the minimum and maximum number of IPI clusters that might be present. Units are in milliseconds.

  - ***nkExtra***
    The number of extra clusters to test for beyond the KDE estimates. For example, if set to 1, and a KDE suggests that 3 – 5 clusters are present, then GMMs will be run for 2 – 6 clusters.

  - ***shareSigma***
    A Boolean value indicating if every cluster in a GMM must have the same standard deviation or not. Valid entries are ‘true’ or ‘false’.

- ***sigma2RegVal***
  Cluster variance regularization value. This is a small number added to the variance of each cluster during the EM process. Valid entries are a single real number, or the string ‘auto’. If set to ‘auto’, an appropriate regularization value is set automatically based on the working sampling rate (48 kHz).

- ***EMTol***
  The EM algorithm estimates maximum likelihood through iterative computation. *EMTol* is the tolerance value used to decide when the log-likelihood has converged to a maximum.

- ***maxEMIterations***
  The maximum number of EM iterations allowed. GMMs that fail to converge within *maxEMIterations* will not be counted as valid models.

- ***maxEMTries***
  The maximum number of times to try running the EM algorithm. In a few cases (presumably caused by unusual initial conditions or atypical IPI distributions), the EM algorithm can encounter errors. Sometimes, retrying with new initial conditions fixes the problem.

6. REFERENCES

Adler-Fenchel, H. S. (**1980**). "Acoustically derived estimate of the size distribution for a sample of sperm whales (*Physeter catodon*) in the Western North Atlantic," Can. J. Fish. Aquat. Sci. **37**, 2358–2361.

Backus, R. H., and Schevill, W. E. (**1966**). "*Physeter* clicks," in *Whales, Dolphins, and Porpoises,* edited by K. S. Norris (University of California Press, Berkeley), pp. 510–527.

Goold, J. C. (**1996**). "Signal processing techniques for acoustic measurement of sperm whale body lengths," J. Acoust. Soc. Am. **100**, 3431–3441.

Gordon, J. C. D. (**1991**). "Evaluation of a method for determining the length of sperm whales (*Physeter catodon*) from their vocalizations," J. Zool. Lond. **224**, 301–314.

Growcott, A., Miller, B., Sirguey, P., Slooten, E., and Dawson, S. (2011). "Measuring body length of male sperm whales from their clicks: The relationship between inter-pulse intervals and photogrammetrically measured lengths," J. Acoust. Soc. Am. **130**, 568–573.

Marcoux, M., Whitehead, H., and Rendell, L. (**2006**). "Coda vocalizations recorded in breeding areas are almost entirely produced by mature female sperm whales (*Physeter macrocephalus*)," Can. J. Zool. **84**, 609–614.

Møhl, B. (**2001**). "Sound transmission in the nose of the sperm whale *Physeter catodon*. A post mortem study," J. Comp. Physiol. [A] **187**, 335–340.

Møhl, B., Larsen, E., and Amundin, M. (**1981**). ''Sperm whale size determination: Outlines of an acoustic approach," FAO Fisheries Ser. **5**, 327–332.

Nishiwaki, N., Oshumi, S., and Maeda, Y. (**1963**). "Changes in form of the sperm whale accompanied with growth," Sci. Rep. Wh. Res. Inst. Tokyo **17**, 1–13.

Norris, K. S., and Harvey, G. W. (**1972**). "A theory for the function of the spermaceti organ of the sperm whale (*Physter catodon L.*)," in *Animal Orientation and Navigation*, edited by S. R. Galler, K. Schmidt-Koenig, G. J. Jacobs, and R. E. Belleville, SP-262 (NASA, Washington, DC), pp. 397–417.

Rhinelander, M. Q., and Dawson, S. M. (**2004**). "Measuring sperm whales from their clicks: Stability of interpulse intervals and validation that they indicate whale length," J. Acoust. Soc. Am. **115**, 1826–1831.

Whitehead, H., and Weilgart, L. (**1990**). "Click rates from sperm whales," J. Acoust. Soc. Am. **87**, 1798–1806.

Zimmer, W. M. X., Madsen, P. T., Teloni, V., Johnson, M. P., and Tyack, P. L. (**2005**). “Off-axis effects on the multipulse structure of sperm whale usual clicks with implications for sound production,” J. Acoust. Soc. Am. **118**, 3337–3345.