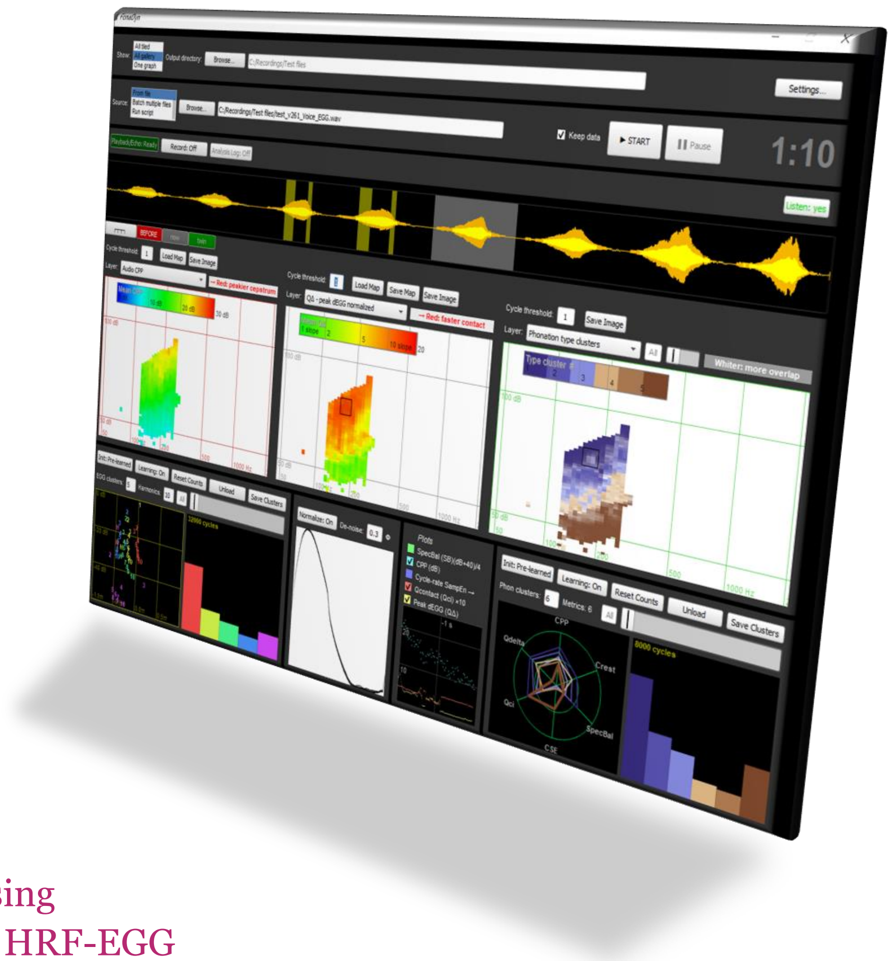
The FonaDyn Handbook

Version 3.1



NEW:

EGG de-noising
New metric: HRF-EGG
Cluster difference maps
Distributions in the colour bar
Restore-context scripts

Sten Ternström

Ternström S, Johansson D, Selamtzis A (2018). FonaDyn - a system for real-time analysis of the electroglottogram, over the voice range. *SoftwareX number 7*, 2018, pp 74-80. DOI: 10.1016/j.softx.2018.03.002

The above is an Open Access article that can be accessed at this link:
http://linkinghub.elsevier.com/retrieve/pii/S235271101830030X ,
where also major updates to FonaDyn can be found when available.
Minor updates can be found at www.kth.se/profile/stern.

FonaDyn version: 3.1.2

FonaDyn is intended primarily as a demonstrator of the principles and applications of voice mapping. Work on FonaDyn is carried out at the author's institution:

Division of Speech, Music and Hearing, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden

Declaration of contents: the FonaDyn software and its documentation are produced using only natural intelligence. ☺

Contents

0 Overview 5
1 PART ONE - Getting started 7
1.1 Hardware requirements 7
1.2 Software setup 8
1.3 Audio device configuration 14
1.4 FonaDyn configuration options 15
2 PART TWO - Theory 17
2.1 Introduction 17
2.2 Signal processing overview 17
2.3 Audio processing 19
2.4 EGG processing 24
2.5 EGG time-domain metrics 26
2.6 EGG harmonic-domain analysis 29
2.7 Sample entropy of EGG cycle data 30
2.8 EGG clustering 32
2.9 Phonation type clustering 33
2.10 Map smoothing 34
3 PART THREE - Using FonaDyn 35
3.1 Window layout 35
3.2 Recording 53
3.3 SPL calibration 57
3.4 Listening from maps 66
3.5 Files 69
3.6 Phonation Type Analysis 81
3.7 EGG Shape Analysis 84
3.8 Analyzing multiple takes or multiple files 90
3.9 Using FonaDyn with RME audio interfaces 90
3.10 Known limitations 91
Acknowledgments 92
References 92
Contents in detail 95

0 Overview

FonaDyn was first conceived in 2015 to analyze electroglottographic (EGG) signals over all or part of the voice range, thereby to visualize the great variability that exists within and between individual voices. Over time, FonaDyn is developing into a more general system for voice exploration and measurement using voice maps. It gives real-time visual feedback of EGG and acoustic metrics as well as phonatory regimes, in the paradigm of the voice range profile (VRP) [1]. ‘Phonatory regimes’ may refer to modal/falsetto, or to many other types of phonatory differences, including gradings within a single voice register. FonaDyn can be used to pursue many kinds of research questions on phonation, voice source dynamics, source-filter interaction and effects of therapy and/or training. It can be used interactively, giving visual feedback on EGG and acoustic characteristics of the voice, or as a data-acquisition front-end with extensive preview facilities. The only constraint is that the phonation must be reasonably periodic throughout.

In addition to mapping several conventional voice metrics, FonaDyn uses statistical clustering to ‘learn’ and categorise regimes of phonation, as well as EGG wave-shapes, and automatically assigns them to more layers in the voice map. The learning feature has several important advantages. The categories need not be known in advance; rather, FonaDyn helps you to find them. The classification does not initially rely on any prescribed thresholds, but rather emerges from the data. With a careful choice of the clustering parameters, it is possible to produce a classification or stratification of the voice that is specific to the research question or application at hand.

FonaDyn is not a supported product, but rather an incrementally evolving, proof-of-concept system for exploring the usefulness of voice mapping. This handbook serves to document FonaDyn’s various functions, algorithms and usage in sufficient detail for you to understand what it does and how it is done. It is up to you to apply it to your own pursuits. Separate workflow guides will no doubt be required for specific situations. Of course, inquiries and feedback are welcome!

FonaDyn is not a stand-alone application. It runs from within the SuperCollider system, which must be installed first. SuperCollider is a free open-source software development tool that runs on MacOS, Windows and Linux. It was originally developed by James McCartney for the computer music community. And “SuperCollider” happens to be an apt name for analysing vocal fold collisions. For the hardware, you will need a *fast* computer, a high-end digital audio interface, a good microphone, an EGG system with an analog line output, and a quiet room in which to record. SuperCollider can use any Windows- or Mac-compatible audio interface, but not laboratory data acquisition boards. With some optional hardware, it is nevertheless possible to acquire also non-audio signals, in parallel and in synchrony with the voice and EGG signals.

This handbook has three parts:

- Part 1 describes how to get started with FonaDyn: the installation and the hardware requirements.
- Part 2 describes the theory and design choices underlying FonaDyn.
- Part 3 describes the user interface and the most common handling procedures.

Again: this Handbook gives mostly general background information that should be sufficient as a reference for getting started, and for understanding and documenting your work with FonaDyn. It does not offer much guidance on using FonaDyn for pursuing any particular research question, or any clinical or pedagogical procedure. To meet some of those needs, a companion FonaDyn Workbook is planned. If you want to contribute to that, please get in touch. The list of references includes publications of studies in which FonaDyn has been used. If you have a particular application in mind, please feel free to send an e-mail to stern@kth.se .

If you are interested in how the SuperCollider code of FonaDyn works, go to SuperCollider's Help window and browse to the topic "FonaDyn". The code architecture is outlined in the help topic for the class "VRPMain".

1 PART ONE - Getting started

1.1 Hardware requirements

- A high-performance computer running Windows 7 or higher, or Apple Macintosh running macOS, or a Linux system (which requires more effort and expertise). The screen resolution should be at least 1400×1080 pixels, bigger is better. We recommend having two display screens, since other programs will often be used at the same time. **Important:** On laptops running Windows, click the battery icon in the task bar, and pull the performance slider to the maximum.
- A low-noise microphone at a fixed distance from the mouth. It is important to verify that no audible hum, hiss or other extraneous sound is present, by listening through the system, at high gain.
- A high-quality digital audio interface that can be configured to have a microphone signal on the first input and a line-level signal on the second input. We recommend the RME line of USB audio interfaces (www.rme-audio.com). Others may work just as well.
- An electroglottograph that has an analog output for the EGG signal. Note that this handbook does not cover the use of your EGG device as such. Please observe its instructions carefully, for how it must be powered, maintained and connected for use.
- For prompting of the subject, a pair of closed circumaural headphones.
- For parallel acquisition of non-audio signals, such as aerodynamic data, see section 3.2.8.

1.2 Software setup

IMPORTANT: If your computer is centrally managed by an IT department, there are several things that they have to know about, in order to install SuperCollider and FonaDyn in a practical way. These are listed in section 1.2.6.

If you have not already done so, you will need to install the driver and control software that came with your digital audio interface or sound card, and become reasonably familiar with it.

Then install SuperCollider on your system, as described below. Once you have installed SuperCollider itself, you should probably play around with it a bit, if only to learn how to run anything at all. You need to know at least how to evaluate a line of SuperCollider code, so that you can install and start FonaDyn. There is a good Help system built into SuperCollider. There is also a separate website at https://docs.supercollider.online/ that you can read from anywhere, with the same content. See [2] for the most gentle introduction to writing SC code.

Installing FonaDyn will also incorporate a number of FonaDyn-specific documents into SuperCollider's online Help system on your computer. These will be of interest mostly if you are interested in how the FonaDyn software works, under the hood.

1.2.1 Versions of SuperCollider

SuperCollider is maintained by an active community of voluntary developers. New versions are released about once a year. There are versions for the major operating systems, and also in some cases separate 32-bit and 64-bit versions. From the FonaDyn user's perspective, there is little or no difference between them. The versions mentioned here were the current versions in November 2023.

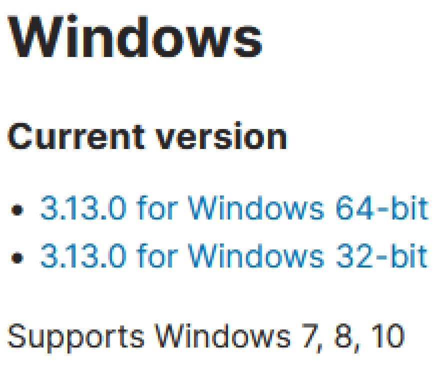
The SClang source code for FonaDyn is portable across all supported platforms. The FonaDyn 'UGen' plugins, however, come in different versions for each platform. FonaDyn version 3.1.2 is supplied with both 32-bit and 64-bit plugins on Windows, and with 'universal' binaries on MacOS. FonaDyn is developed on Windows and occasionally tested on MacOS.

Version 3.13.0 of SuperCollider, released in February 2023, contained some breaking changes and new features. The present FonaDyn version 3.1.2 relies on some of these new features, and so requires version 3.13.0 or later of SuperCollider. If you need to update either SC or FonaDyn, please follow the instructions in the file UPDATING.txt in the distribution file set.

1.2.2 Windows

Installing SuperCollider

1. Visit the SC download web page [3] and download one of the following two components (the SC version number may have changed).



Windows

Current version

- 3.13.0 for Windows 64-bit
- 3.13.0 for Windows 32-bit

Supports Windows 7, 8, 10

If you are running a 32-bit Windows, for instance on a breadboard computer, choose the 32-bit version. On the current versions of 64-bit Windows, either of these will work. The 'SuperNova' multi-processor option, where mentioned, is not used by FonaDyn.

2. Install the current version of SuperCollider, by running the downloaded .EXE file.

3. Back on the SC download web page, scroll down to **Plugins and Extensions** and click on **sc3-plugins**. On the new page, click on the button at **Download latest release**. Download the .ZIP archive that matches your choice at (1).

   sc3-plugins-3.13.0-Windows-64bit-VS.zip

   sc3-plugins-3.13.0-Windows-32bit-VS.zip

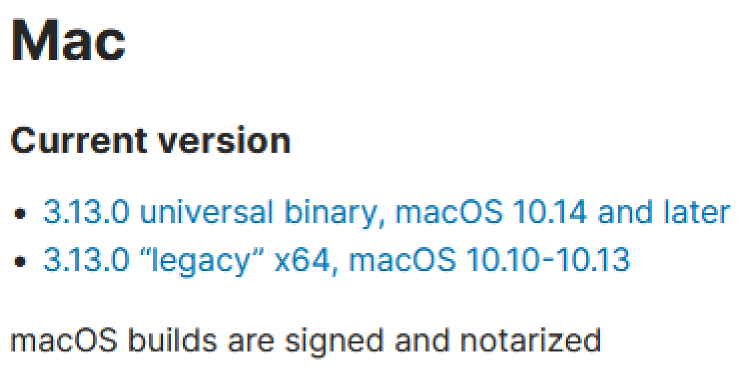
4. Open the ZIP and read the README file, to choose the right storage location. Then install the sc3-plugins, by extracting the other contents of the downloaded .ZIP file into the appropriate Extensions directory. We recommend that you install the sc3-plugins into the directory for all users. On recent versions of Windows, this will be the directory
`C:\ProgramData\SuperCollider\Extensions.`

5. See section 1.2.6 about configuring your system's anti-virus tools. We have found that some anti-virus scanning products can slow down FonaDyn too much. It is important to find the right level of protection.

1.2.3 MacOS

Installing SuperCollider

1. Visit the SC web download page [3]. Download one of the following (the SC version may have changed):



Mac

Current version

- 3.13.0 universal binary, macOS 10.14 and later
- 3.13.0 "legacy" x64, macOS 10.10-10.13

macOS builds are signed and notarized

This will download a zip file that contains the Mac installer package and some README files. Read them. (sn*), if mentioned, refers to the 'SuperNova' multi-processor option, which is not used by FonaDyn.

2. For getting the sc3-plugins, scroll down to **Plugins and Extensions** and click on **sc3-plugins**. On the new page, click on the button at **Download latest release**. Download the file sc3-plugins-3.13.0-macOS.zip.

3. In the **sc3-plugins** .ZIP archive, read the README file, to choose the right location. Then install the sc3-plugins, by extracting the other contents of the downloaded .ZIP file into the appropriate Extensions directory. We recommend that you install the sc3-plugins into the directory for all users. On recent versions of macOS, this is
`/Library/Application Support/SuperCollider/Extensions.`

1.2.4 Linux

Linux runs on many diverse platforms. At this writing, the FonaDyn-specific plugins have not yet been compiled for any Linux (older versions did run). Please contact the author at stern@kth.se if you are interested in using FonaDyn on some implementation of Linux, and/or in helping us make that build. You will then be provided with the plugin source codes, and with some guidance.

There is a Linux distribution for SuperCollider [3], but, as is customary with Linux, it is provided in source code form only. This means that you have to download the whole source and build SuperCollider itself, before you can install FonaDyn on Linux. Know-how and experience of making such builds will be necessary. You will need to find version 3.13.0 or higher in the Releases link at supercollider.github.io/download. Some of the Linux packages there are still for older versions of SuperCollider. You will need also to build the necessary sc3-plugins.

1.2.5 Installing FonaDyn into SuperCollider

1. If you are about to update to a newer version of FonaDyn, you must first uninstall your current version by evaluating the line

   ```
   FonaDyn.uninstall;
   ```

   This deletes all files in the installed folders, so do not save anything else there that you want to keep. Any local modifications to your installed FonaDyn source code will be lost.
2. Download **FonaDynInstall-*version*.zip** using the download link that you have obtained from the *SoftwareX* repository, or directly from us. The most recent version is available at https://www.kth.se/profile/stern .
3. Unzip only the contained directories **FonaDyn** and **FonaDynTools** into an Extensions directory, preferably your own SuperCollider User Extensions directory. On recent versions of Windows, this will be

   `C:\Users\<username>\AppData\Local\SuperCollider\Extensions.`

   The directory `C:\Users\<username>\AppData` is often hidden, but if you type it into the address bar in the Windows Explorer, it will open.
   On MacOS, this will be the directory

   `/Users/<`**`username`**`>/Library/Application Support/SuperCollider/Extensions.`

4. If you will be using Matlab to process FonaDyn output files, you can use the provided collection of m-files as a starting point. Unzip the folder **matlab** and move it to *the parent directory* of the Extensions directory. There is a **_readme.txt** file in the **matlab** folder with more information.
5. The remaining contents of FonaDynInstall-*version*.zip are various useful files that are not SuperCollider code. Unzip and store them wherever you like, *except* in the .../Extensions directory or any of its subdirectories.
6. Choose **File | Open user support directory**. A new file browser window will appear.
7. Unzip the file `startup.scd.example.txt` and adapt the settings to your system. Save it as `startup.scd` in the folder from (6). If that folder already contains a `startup.scd` file that you were using earlier, you can keep that file.
8. Run SuperCollider. Wait for the "Post window" to display this line:

   ```
   *** Welcome to SuperCollider 3.13.0. *** For help press Ctrl-D.
   ```

9. In a new code window (**File** | **New** or Ctrl+N), type and then evaluate the line

   ```
   FonaDyn.install;
   ```

   This will perform a few checks, copy a few additional files to their proper locations, and then recompile the class library so as to incorporate FonaDyn.
10. On Apple MacOS computers only: when you run FonaDyn for the first time, the computer must be connected to the Internet, or Apple's safety checks will not work.
11. You should now be able to start FonaDyn, by evaluating the line

    ```
    FonaDyn.run;
    ```

The `FonaDyn` directory contains source code and Help files that are specific to FonaDyn. You can read these help files using the built-in SuperCollider help system, under “Browse | FonaDyn”. The `FonaDynTools` directory contains supporting code that was written for FonaDyn, and that may be of interest also to developers of other SuperCollider programs. You can read the corresponding help files similarly, under “Browse | Tools”. The FonaDynTools directory also contains subfolders with the platform-specific dynamic-link libraries for the FonaDyn UGens.

STOP! If something goes wrong, and you see a torrent of error text being dumped in the post window, stop SuperCollider by typing Ctrl+. (Cmd+. on the Mac). That is, hold down the Ctrl/Cmd key, and press the key for the full-stop character ‘.’ (period). That is a general stop command in SuperCollider.

1.2.6 Centrally managed installation

If your computer is centrally managed, it may have access policies in effect that will stop SuperCollider from doing certain things. Please show this section 1.2 of the handbook to your IT-support department, so that they can install SuperCollider with the necessary privileges, etc. Have them note especially the following:

SuperCollider is a software development environment in which it is necessary for users to be able to create/modify files in certain privileged locations and/or functions. At run-time, it does real-time processing of audio files, which needs to be fast, while being harmless from a malware point of view.

There are three executable modules: **scsynth.exe**, **sclang.exe** and **scide.exe**, which communicate with each other using a network protocol, and they must be allowed to do so.

For performance reasons, it may also be necessary to include their *processes* in the list of processes that are excepted from anti-virus monitoring. In particular, when the DSP server, scsynth.exe, is booted for the first time in a Windows session, the Windows Defender anti-virus protection can take up to *two minutes* before allowing scsynth.exe to run (subsequent boots are quick). This is very annoying, and most users will want to except the processes “scsynth.exe” and “sclang.exe” for this reason. We have noted also that anti-virus products from F-Secure will need to be specially configured for usability with SuperCollider. They valiantly try to scan everything in real time, which usually does not work.

For its signal processing functions, SuperCollider uses a large number of small ‘plugin’ DLL’s that have the filename extension .SCX (on Linux: “.SO”) . On rare occasions it may happen that an anti-virus program will quarantine one of these, causing SuperCollider to complain that a particular module is not found. We have never known this actually to be malign, but of course your policies must be upheld. You will have to decide what to do about it.

For Windows, FonaDyn is supplied with plugins for both 64-bit and 32-bit versions of Supercollider. The latter could be appropriate for small single-board computers. The matching set of plugins is selected automatically during installation using the routine `FonaDyn.install`, as described above.

On MacOS, FonaDyn 3.1 is supplied with 64-bit plugins that work with both ARM and Intel chip archictectures. Also, the FonaDyn installation replaces the sc3-plugin `PitchDetection.scx` with one that has been recompiled to use the FFTW3F library, rather than the MacOS own vDSP library – they give different results. Running `FonaDyn.uninstall` will restore the original `.scx` file.

1.3 Audio device configuration

FonaDyn requires one stereo pair in and one stereo pair out. If your audio device has only two inputs and outputs, then these are what FonaDyn will use. If you are using an external multichannel audio interface (recommended), then your operating system's control panel for sound will probably list more than one pair of audio inputs and one pair of audio outputs. By default, FonaDyn will use the first pair of inputs and the first pair of outputs. It can be configured to record on any combination of multiple inputs and outputs, provided that they all sit on the same audio interface hardware (or on separate devices that are synchronized electrically). A list of currently available audio devices is printed in the Post window, whenever the server is booted.

If you will not be using SuperCollider for anything else, you can specify 2 ins and 2 outs in the SC startup file. For instructions on how to do this, see the SC documentation for the startup file. The startup file has its own entry in the **File** menu of SCIDE, so it is easy to find. Activating more ins and outs than are needed incurs an unnecessary CPU load.

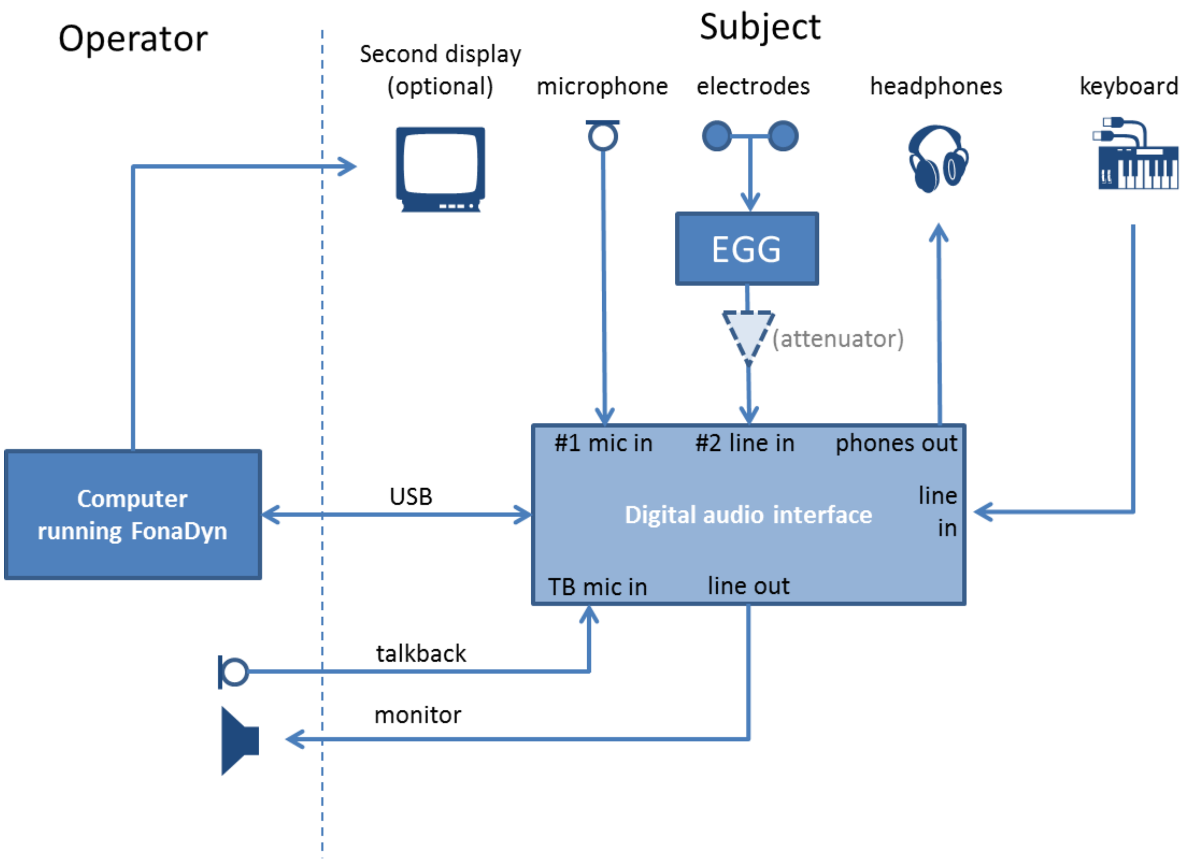


Figure 1. Typical recording setup.

A typical experimental setup is shown in Figure 1. The first (left) input channel must receive the signal from a microphone, via a microphone preamplifier. Many audio interfaces have built-in microphone preamps, on two or four inputs. For recording, you will need to calibrate the gain of the microphone channel (→3.3).

The second (right) input channel must receive the line-level signal from the electroglottograph. Unlike the microphone signal, the EGG signal level does not need to be calibrated, as it is ignored by FonaDyn (only the EGG *shape* is considered). However, the EGG gain should be adjusted so as to prevent clipping. Note that some EGG devices have a large voltage swing on their outputs, typically ±10 volts, which is too strong for most audio interfaces. If the EGG device's output level cannot be turned down enough, you may need to pass it via an attenuator to the audio interface. If your audio interface has adjustable input sensitivity, choose the *least* sensitive setting for the EGG signal.

As for outputs: if "Echo/Playback" is active, FonaDyn will play a copy of the audio signal (live or from file) to both left and right outputs.

1.4 FonaDyn configuration options

You can skip this section until you have become more familiar with FonaDyn.

In the startup.scd file (**File | Open startup file**), you can insert `FonaDyn.config` statements for some things that typically will be persistent. There can be multiple lines with `FonaDyn.config(…)` in the startup file, and it can control several options. See also the topic "FonaDyn" in the SuperCollider Help System. Note that the startup file is read only when SuperCollider starts, so the SC interpreter must be rebooted after making any changes to the startup file.

```
FonaDyn.config(inputVoice: 8, inputEGG: 9);
```

On some audio interfaces, the microphone pre-amps are not on the first two inputs. You can tell FonaDyn to use any other two inputs for voice and/or EGG. Inputs in SuperCollider are numbered from zero, so "`inputVoice: 8`" means to fetch the microphone signal on the *ninth* channel of the audio interface. You will need also to enable all the inputs up to and including the chosen ones with a line in the startup file, like this: `Server.local.options.numInputBusChannels = 10;`

```
FonaDyn.config(sampleFormat: "int16");
```

As of version 3.0, FonaDyn normally records audio and EGG into 24-bit fixed-point .WAV files. If you need interoperability with older signal file editors, such as Swell, then change to 16-bit recording, by inserting this statement into the startup file. FonaDyn will still read 16-bit files without any special action.

```
FonaDyn.config(singerMode: true);
```

FonaDyn normally has a maximum SPL of 120 dB, but this may not be enough for very strong voices (→3.3.7). Setting `singerMode` to `true` changes the maximum SPL to 140 dB, and also enforces the use of 24-bit recording. Choosing a 16-bit file as input will disable 'singer mode'.

`FonaDyn.config(fixedAspectRatio: true);`

FonaDyn normally does not constrain the aspect ratio of map cells; maps can be stretched arbitrarily. If you wish to fix the aspect ratio to 2:1 so as to follow the UEP recommendation (→3.1.4), then set `fixedAspectRatio` to `true`. FonaDyn will pad the map borders with blank margins as necessary.

You can also control the aspect ratio in this way while FonaDyn is running, by evaluating the statement in the Command Line field of the SC IDE.

`FonaDyn.config(tileMapsVertically: true);`

FonaDyn normally tiles maps side by side. In the 'Tiled' display modes, you might prefer them instead to be tiled vertically, depending on the size of your screen and on how many maps you want to display. In the 'Gallery' display modes, tiling of maps is always horizontal.

You can control this aspect of tiling maps also while FonaDyn is running, by pressing Alt+X; but that does not change the default setting given here.

`FonaDyn.config(runScript: "C:/full/path/to/text-file");`

This runs the given script on start-up. Typically you would use this to initialize FonaDyn to a state that you have customized (→3.5.8). You must use forward slashes "/" in the path, even on Microsoft Windows.

If you start FonaDyn using `FonaDyn.rerun`, though, any previously archived settings will be used instead, and the start-up script will not be run (→3.1.10).

2 PART TWO - Theory

2.1 Introduction

FonaDyn allows you to explore how the acoustic and EGG signals vary with the fundamental frequency (f_0) and the sound pressure level (SPL) of a voice. It does this by computing various features of the signals and mapping these features onto the f_0-SPL plane, also called the 'voice field'. The resulting 'voice maps' unfold in real time as the subjects vocalizes, or as an existing recording is analyzed. The particular advantages of analyzing voices by mapping them are discussed in [28].

Although the individual metrics are often of interest on their own, it is also helpful to see which *combinations* of metric values that correspond to different regimes of phonation. To facilitate this, FonaDyn can do a statistical clustering of the metrics 'on the fly' and find out which combinations are the most prevalent in the incoming signals. When you ask for a certain number of clusters, FonaDyn will classify the incoming data, on the basis of the metric values, into that many categories, and assign a colour to each category for display on the map. Clustering is done on EGG wave shapes only, and also on combinations of acoustic and EGG metrics. The clustering can be automatic, or, an experienced operator can cajole the clustering into categories that are of interest for a particular application. Normally such an analysis has these stages:

(1) record to file
(2) learn a set of cluster centroids from the recording
(3) use that set to classify other similar signals
(4) export the results for further analysis.

Here, stages (1) to (3) are done in FonaDyn. Clustering (2) will usually require several passes over the same input signal. Once you have fixed the centroids, classification (3) is a one-pass operation. The "further analysis" in (4) is done with the tools of your choice, e.g. MS Excel or Matlab. To facilitate research work, practically all signals and results from FonaDyn can be exported and imported via files in common formats (.csv, .wav, .aiff), for processing in other software.

2.2 Signal processing overview

The audio and EGG signals are processed in parallel, see Figure 2. From the microphone audio signal, six metrics are derived: the signal level, the fundamental frequency f_0, the f_0 periodicity (or "clarity"), the crest factor, the spectrum balance and the cepstrum peak prominence (CPP). The level and f_0 are taken as independent variables that specify the plot position in the voice field. The remaining metrics are taken as variables dependent on level and f_0 . The *clarity* metric is used as a periodicity gate: ranging from 0 to 1, it must be close to 1.0 for any obtained values to be considered valid. The default threshold is 0.96 . The *crest factor* gives an approximate but useful indication of the high frequency content of the voice signal [7]. For open vowels, it is somewhat similar to the MFDR. The *spectrum balance* is the ratio of acoustic powers above 2 kHz and below 1.5 kHz, as separated by two filters that slope by 24dB/octave. The *CPP* is a metric that reflects the periodicity of the audio signal *spectrum*. It reaches high values when the voice is not breathy or unstable. These audio metrics are described in greater detail in section 2.3.

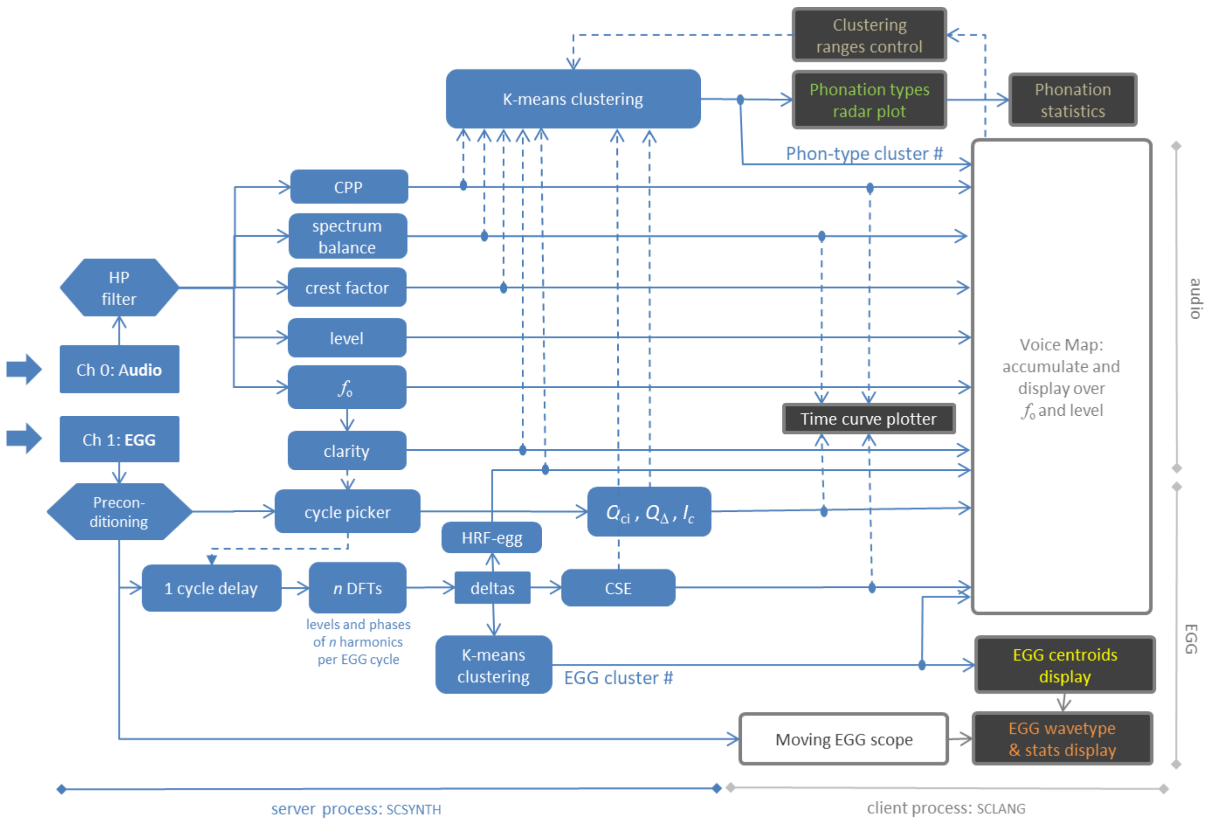


Figure 2. Block diagram overview of the signal processing in FonaDyn.

As for the EGG signal, its gain is difficult to calibrate, and also its amplitude typically varies a lot for reasons that are not relevant to phonation, such as varying larynx height, the disposition of the informant's neck, and in some EGG devices an automatic gain control (AGC). Therefore, all EGG metrics computed by FonaDyn are normalized such that *the EGG amplitude is disregarded*. Only the EGG wave *shape* is considered to be of interest, and hence only a coarse adjustment of the EGG signal gain is needed. FonaDyn computes five metrics of the EGG: the contact quotient Q_{ci}, the normalized peak EGG derivative Q_Δ , the index of contacting I_c , the harmonic richness factor *HRFegg*, and the cycle-rate sample entropy *CSE*. These metrics are described in detail in section 2.5.

The voice and EGG signals are analysed in terms of these time-varying metrics. Each metric has its own separate layer in the voice map. The cells in that layer accumulate the average of the metric, at each combination of semitone and sound level. These averages are visualised on a colour scale that is specific to each metric. This is described in more detail in section 3.1.4.

The time-varying metrics can also be statistically clustered into categories. In order to visualise the clustered categories, FonaDyn assigns colours to them, and sets the colour of each cell to that of the dominant cluster at that f_o and SPL. However, it is possible to vocalize at the same pitch and loudness in different ways, so multiple visits to one cell may result in different classifications – the categories will overlap, in places.

FonaDyn can show maps of each category separately, so that you can study this overlap, too.

Since the purpose of the voice map is to characterize the 'topography' of the metrics over the voice field, regardless of what is being said or sung, it contains no temporal information. To get at the temporal information, you can have FonaDyn produce a Log file. This is a multichannel signal file that contains f_o and SPL at full precision, all the metrics, and also resynthesis information for the EGG cycle shapes, on a time line. The Log file frame rate is nominally one frame per phonatory cycle, or it can be fixed.

2.3 Audio processing

2.3.1 Input selection

FonaDyn takes a microphone audio signal on the first A/D input and an EGG signal on the second A/D input; or, takes its input from audio files with at least those two channels. Valid input audio file formats are those supported by the 'libsndfile' library, in other words, almost any format, including some compressed audio formats such as MP3. The sampling rate must be 44100 Hz per channel. FonaDyn's own recordings of Voice+EGG are stored in 24-bit integer format WAV files by default, and this can be changed (→1.4).

When recording from the live inputs, the 'raw' microphone and EGG signals are written unchanged to the output file, with no prior processing. This is to avoid repeated preprocessing when analyzing a recording rather than the live inputs. It also enables the recordings to be used by other analysis systems, with no constraints having been imposed on the signals. In a first-generation FonaDyn recording, the signals in the file are identical to the signals that were obtained 'live' from the A/D-converters (→3.2.7).

2.3.2 Audio preprocessing

A second-order high-pass Butterworth filter at 30 Hz is always applied to the microphone signal, to suppress low-frequency rumble. The filter characteristic is essentially equivalent to a C-weighting filter for level measurements. This filter is not linear-phase, and does not need to be.

2.3.3 Representation of sound levels

FonaDyn mandates a certain calibration for sound pressure gain. In order for the SPL axis in the voice maps to be correct, the gain when recording must be set so that one of the scalings in Table 1 is achieved. There is a built-in tool for doing this (→3.3). The scaling will depend on the chosen sample format, 16 or 24 bit, and on 'normal' or 'singer mode' (→3.3.7).

In the audio track of the recorded file, the sound pressure 1 Pa of a sine wave (=94 dB SPL re 20 µPa) at a distance of 30 cm in front of the speaker must result in a signal amplitude of 0.050 RMS (0.07071 peak) relative to full scale. The recording microphone does not actually have to be placed at 30 cm in front of the speaker; for instance, a headset microphone can also be used. But the signal gain, wherever the microphone is, must be adjusted so that this scaling is achieved *as if* the microphone were at that frontal position.

Table 1. The fixed calibration options for FonaDyn-compatible signal files: normal, "singer mode" and unscaled floats.

Mode	**Normal**	**Normal**	**'Singer mode'**	**Unscaled floats**
Bits per sample	16	24	24	32
Number format	integer	integer	integer	float
Pressure at full scale (Pa)	20	20	200	3.4×10^{38}
Peak level at full scale (dB)	120 dB	120 dB	140 dB	864.6 dB
94 dB sine RMS (1 Pa)	0.05	0.05	0.005	1.0
ditto in integer representation	1 638	419 430	41 943	n/a
94 dB sine peak	±0.07071	±0.07071	±0.007071	1.414
ditto in integer representation	±2 317	±593 164	±59 316	n/a
Max SNR from quantization	98 dB	146 dB	146 dB	146 dB
±1 lowest bit SPL re. 20 µPa	30 dB	-18.5 dB	1.5 dB	-664 dB
Min. voice SPL with 10 dB margin	**40 dB**	-8.5 dB	11.5dB	-654 dB

In the bottom row of Table 1 we see that, if we wish to maintain the minimum advisable margin to the noise floor of 10 dB [29], a 16-bit recording is barely adequate for recording very soft voices in quiet surroundings, since it lies just at the bottom of the standard SPL range, which is 40-120 dB. For this reason, FonaDyn from v3.0 records signals using 24 bits per channel, by default. If you need to record at 16-bit sample width, e.g., for reasons of compatibility with other software, then you can specify 16-bit recording instead (→1.4).

FonaDyn will also read signals in unscaled floats, which are then assumed to represent instantaneous sound pressure in Pascal. FonaDyn will not record in float format, because many other audio apps refuse to handle floating-point signals larger than ±1.0. For information on how data is represented in FonaDyn's other types of output files, see section 3.5.

The reason for having a separate 'singer mode' rather than always using a fixed maximum SPL of 140 dB is that it requires a large headroom of some 40 dB, and makes the resulting audio files very low-level and awkward to listen to.

FonaDyn measures the signal level for every phonatory period, so that each period can be assigned a position in the f_0-SPL plane (the *voice field*). Hence the SPL measurement in FonaDyn corresponds most closely to a C-weighted "fast" reading; but it is faster. Calibrating the signal level to true sound pressure level (SPL) is *very* important for meaningful comparisons between across recordings; both your own and those of others (→3.2.9). As shown in Table 1, the audio signal in the `*_Voice_EGG.wav` files is expected to be scaled such that a peak-to-peak full scale sine wave represents ±20 Pa peak amplitude, which corresponds to 117 dB SPL RMS. You need to calibrate the gain in your signal chain so that this is true (→3.3), or, be prepared to correct your results in post-processing.

Some supplementary detail on how the SPL is represented internally is warranted here. As recommended by the European Union of Phoniatricians, the SPL scale in FonaDyn extends from 40 to 120 dB as measured at distance of 0.30 m. By convention in digital audio, the full-scale signal amplitude is the floating-point value ±1.0, which corresponds to the maximum *peak* amplitude that the A/D-converters can handle. In acoustics, levels are computed from the RMS of a signal rather than from the peak amplitude. However, the RMS value of a signal depends on its waveform. Only a symmetrical square wave with negligible transition times has an RMS amplitude that is equal to the peak amplitude. For all other waveforms, the RMS value is smaller than the peak amplitude. Most often we use sine waves when calibrating the level, and the RMS level of a sine wave is 3 dB below its peak amplitude. Hence the highest sine wave SPL that can be represented in FonaDyn using the nominal calibration is 117 dB. For recording louder voices, see 3.3.7.

2.3.4 Audio-based metrics

From the microphone audio signal, six metrics are derived: the signal level in dB, the fundamental frequency f_o, the f_o periodicity (or ”clarity”), the crest factor, the spectrum balance and the cepstral peak prominence (unsmoothed).

- The *level* of the audio signal serves to provide the vertical plot position in the voice maps. It is computed as follows.
  1. The audio signal is high-pass filtered as described above.
  2. The periods in the audio signal are found by double-sided peak-picking, following Dolanský [8].
  3. The RMS of the signal is computed period by period. No smoothing is applied.
  4. The RMS values are converted to decibels down, relative to full scale. These are the values that are written to the Log file (→3.5.3).

- The f_o value serves to provide the horizontal plot position in the voice map. In FonaDyn, f_o is measured in semitones, using MIDI note numbers with fractions. MIDI = 57.0 corresponds to f_o = 220.00 Hz. No calibration of f_o is needed. It is computed by an algorithm based on autocorrelation via the FFT [6] (in the SuperCollider ‘UGen’ `Tartini`). The value of f_o is updated at intervals of about 23 ms. This f_o extraction algorithm is the best that we have come across; it is good at ignoring strong overtones, for example. However, it is also quite sensitive and will find periodicity where other algorithms might not. In particular, if there is an AC mains hum in the audio signal, even a weak one, there will be a risk of locking on to that when the voice f_o is at some multiple of the mains frequency (which is 50 or 60 Hz, depending on where your lab is). Therefore it is very important not to have hum in your signal chain.

- The *clarity* metric is used as a periodicity tester. Ranging from 0 to 1, it must be above a certain threshold for the EGG processing to proceed. The default threshold is 0.96, which is a fairly strict requirement, corresponding to quite stable phonation. Mathematically, the ‘clarity’ is the value of the autocorrelation function of the *audio* signal, at a delay of one cycle at the estimated f_o. Like f_o, it is computed by the UGen `Tartini`. For running speech you may want to lower this to 0.90; for pathological voices, perhaps even lower.

- The *crest factor* gives a simple but useful indication of the high frequency content of the voice signal. It is computed as the ratio of the peak amplitude to the RMS amplitude, for every phonatory cycle. A sine wave has a crest factor of $\sqrt{2} \approx 1.414$ (≈ 3 dB), while a very bright voice signal can reach a crest factor greater than 4 (≈ 12 dB). For open vowels only, it is a close cousin of the maximum flow declination rate (MFDR), that is, of the peak value of the first time derivative of the glottal flow. However, it varies also with f_o/f_{R1} and with the glottal open quotient [7]. For male voices, the crest factor tends to be high in the habitual speech range. It would be nice, but CPU-intensive, to have also automatic inverse filtering, in which case this crest factor could be replaced by the actual MFDR.

- The *spectrum balance* SB is the ratio of the acoustic power above 2 kHz to that below 1.5 kHz, as separated by a low-pass and a high-pass filter that both slope by 24 dB/octave. The two powers are smoothed at 50 Hz and then the level difference High-Low in dB is computed. The spectrum balance is usually negative and typically increases (becomes less negative) with increasing vocal effort. In the literature, there are several variants of how spectrum balance is defined, so watch out for that. Similar metrics are for example the ‘alpha ratio’, with a crossover at 1 kHz, the ’singer power ratio’ (of opposite polarity!) and various definitions of the spectrum slope. In our experience, the exact filter cutoff frequencies have only a minor effect on SB, at least for the vowel /a/, which has little spectral energy in the filter cross-over region. The SB is indirectly related to the second time derivative of the glottal flow. SB is also very vowel-dependent. As with the long-time average spectrum (LTAS), long recordings of connected speech or song can still be compared. FonaDyn rejects unvoiced speech segments, but voiced fricatives will have a high SB, and should preferably be avoided.

- The *cepstral peak prominence* (CPP) is a metric that is higher the more regularity and the less noise there is in the audio signal. It is reported in dB and will typically range from less than +10 dB in a severely dysphonic voice to +30 dB in a very ‘clean’ voice. Like all other voice metrics, CPP varies with SPL and f_o, which is why it is instructive to see it on a voice map. In FonaDyn, the computed CPP is retained for voiced segments only, as computing the CPP for unvoiced segments is not meaningful [19].

2.3.5 More about the CPP

As for all the other metrics, voicing is detected on the basis of the 'Clarity' metric. Lowering the Clarity threshold will also somewhat lower the CPP, because a greater proportion of irregular and/or noisy phonations will be accepted.

Many parameters need to be chosen in order to compute the CPP, and these will affect the outcome. In particular, the CPP is often smoothed in one of several ways, collectively referred to as CPPS (for -smoothed). However, such smoothing can also radically change the measured value of the CPP. In FonaDyn, the default is to use the unsmoothed CPP. Its value across many frames will be averaged anyway, thanks to the mapping paradigm.

The cepstrum parameters have been set as follows, partly following Awan *et al* (2013)[20], and modified after informal experimentation to give reasonable results both with sustained vowels and running speech.

- The initial FFT is 2048 points, Hanning window.
- The cepstrum is 1024 points, resulting in 512 quefrency bins.
- The power cepstrum is converted to dB before the linear trend line is computed.
- The trend line is computed by simple linear regression. (The more “robust” methods offered by Praat require two sorting operations, which can take a long and unpredictable time to complete, and thus are unsuitable for real-time operation.)
- The frame rate is 43.07 Hz. A new estimate of CPP is produced every 23 ms. Results from unvoiced segments are discarded, by a separate mechanism.
- Voice fundamental frequencies from 60 Hz to 880 Hz are covered.
- The peak is simply the largest value in any quefrency bin; interpolation to peaks between bins is not performed. This will generate some noise in the framewise CPP estimates, but in practice, that noise will be absorbed by the per-cell averaging that occurs in the CPP layer of the voice map.

If you prefer the smoothed CPP, it is possible to obtain it simply by changing two file names. Contact us if you are interested. If you do elect to enable CPP smoothing, the following also applies:

- Temporal smoothing is implemented as one first order low-pass filter per quefrency bin. The filter coefficient is 0.3, which corresponds to a first order low-pass filter at 16 Hz for each bin. The smoothing window in time is thus an exponential decay rather than a rectangular average.
- Quefrency smoothing is a mean of 7 bins, corresponding to a quefrency window of 0.3 ms.
- CPPs values will be markedly lower than the CPP values for the same signal.

2.4 EGG processing

The EGG signal is pre-conditioned in two steps: (1) with a steep high-pass filter that removes low-frequency rumble, and (2) with a spectral thresholding scheme (→3.7.1) that can suppress some of the system noise that is very common in EGG hardware. The EGG signal is then segmented into cycles. From then on, the data rate is reduced, being proportional to the EGG cycle rate, rather than to the sampling rate. Each cycle is analyzed individually, both in the time domain and in the harmonic domain. The harmonic analysis uses a Discrete Fourier Transform (DFT) for its *N* lowest Fourier components, where *N* is typically 10 or less.

In parallel with the clustering, the time series of per-cycle DFT data are analyzed also for their 'sample entropy', or SampEn for short (→2.7). This is a metric of phonatory instability that is designed to detect abrupt changes, such as register breaks, while suppressing minor instabilities [5].

2.4.1 Preprocessing

The absolute EGG peak amplitude is monitored during recording (→3.2.4). If it comes within 0.5 % of full scale, an **EGG CLIPPING** warning is flashed on the screen. Even a slightly clipped signal would compromise the spectral analysis for the clustering, while a signal that is partially out of range is invalid.

The EGG signal is then passed through these stages:

1. A high-pass filter at 100 Hz (-3 dB @ 80 Hz), using a 1024-point linear phase FIR filter. This practically eliminates the large near-DC content that is common in EGG signals. To do so is necessary for the Q_{ci} computation. However, it also means that the lower that f_o descends *below* 100 Hz, the more the EGG waveform will be distorted. So, **interpret EGGs at low f_o with caution**.
2. Transform into the frequency domain using a 2048-point FFT with 50% overlap.
3. The signal level in each FFT frequency bin, if below an adjustable threshold, is expanded downwards using a 1:4 ratio (in decibels). This attenuates the weakest parts of the spectrum only, which are typically noise at frequencies > 3 kHz. To do so is necessary for the computation of a 'clean' Q_Δ (the normalized peak dEGG).
4. Transform the expanded signal back into the time domain.

This EGG preprocessing limits the range in f_o : downwards, to 100 Hz. Also, this EGG preprocessing introduces a 78 ms delay, which is accounted for internally. The spectral thresholding (steps 2, 3 and 4 above) is performed only when analyzing from a file, not when recording live signals. This is because the resulting delay with respect to the live sound could be annoying for visual feedback purposes.

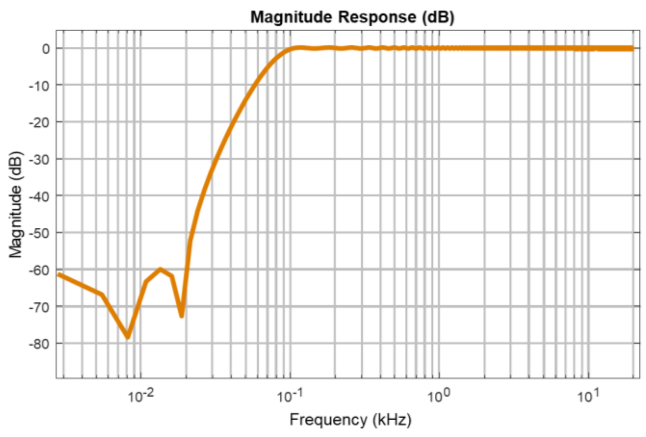


Figure 3.

Frequency response of the high-pass EGG filter. The log scale frequency labels are at 10, 100, 1000 and 10000 Hz.

2.4.2 Period segmentation

By "period segmentation", we mean the process of identifying and marking where each phonatory period, or cycle, starts. FonaDyn implements two strategies for period segmentation of the EGG signal: double-sided peak-following, after Dolanský [8]; and phase tracking. The double-sided **peak follower** is the simpler one; it is applied to the differentiated EGG signal (dEGG). The dEGG is computed as the differences between consecutive sample points. The positive peak-picker selects a point somewhere just prior to a local positive maximum of the dEGG signal. The 'circuit' then waits for a local negative maximum to occur, before allowing the next positive dEGG peak to trigger a new period. Using the dEGG rather than the EGG makes the triggering point somewhat less dependent on f_0 , but also more sensitive to noise.

Sudden changes in dEGG amplitude (which are common) may lead to one or a few lost cycles. Sudden increases in dEGG amplitude may cause the preceding cycle time to be estimated as slightly shorter.

The default method, however, is based on the **phase portrait** [4]. The EGG signal is time-integrated and then paired with the original to form a complex-valued signal of the integrated EGG. By using the integrated signal (rather than the more common Hilbert transform, which is somewhat similar to the *derivative* of the EGG), we obtain better rejection of low-level noise, and the cycle trigger point is usually very close to positive-going zero-crossings in the EGG signal. The phase of this pseudo-analytic signal is now computed as the arctangent of the complex pair; and then that *phase* signal is subjected to the Dolanský method described above. The phase wrap-around from π to $-\pi$ typically occurs very near positive zero-crossings in the EGG, giving a convenient transient in the phase signal. This method for cycle segmentation generally performs better than peak-picking with noisy signals, as well as with fluctuating amplitudes, and it seems to give more robust segmentation overall. With unusual EGG signals, however, which may have multiple deep inflexion points per cycle, this method is sensitive to multiple loops in the phase portrait, and may be less reliable than the dEGG peak-following method.

Cycles longer than 20 ms (50 Hz) or shorter than 0.23 ms (4410 Hz) are rejected at this stage. You can inspect FonaDyn's cycle segmentation in an optional, special file (→3.5.11).

2.5 EGG time-domain metrics

A somewhat more detailed discussion of these metrics is given in reference [4], which is provided in the folder 'FonaDyn Extras'. These metrics are computed directly from the conditioned EGG signal.

2.5.1 The contact quotient Q_{ci}

The contact quotient is that fraction of a whole EGG cycle for which the vocal folds can be said to be in contact; hence, it can range from zero to one. Numerous schemes have been proposed for calculating it [9]. Here, we are content to find a quantification that represents the relative *amount* of contacting over the cycle, without regard for the actual instants of opening or closing, nor for the possible complementarity with the glottal *flow* waveform.

Consider an EGG pulse normalized to length 1, with the cycle amplitude normalized to the range 0…1 . The *area under* this normalized pulse can also range from 0 to 1. This area can be estimated for arbitrary pulse shapes, so it does not require a single peak, nor well-defined opening and closing events, nor a contacting threshold that would always be in some sense arbitrary. We will call this metric the Q_{ci} , for "quotient of contact by integration". The figure below shows how it is computed.

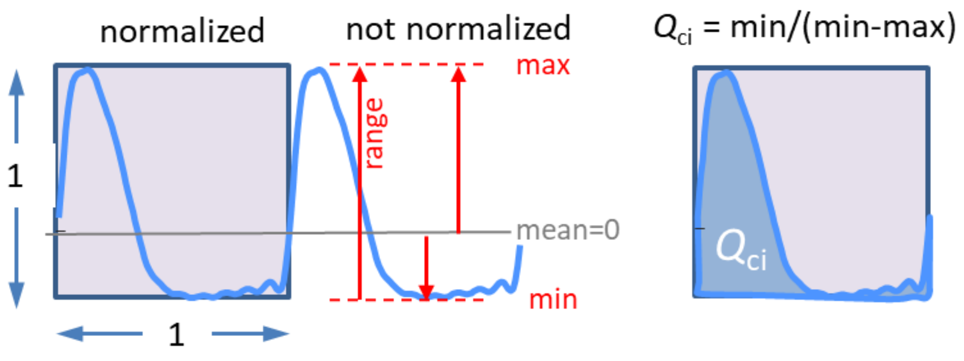


Figure 4. Computation of Q_{ci} , as the area under the normalized pulse. If the non-normalized EGG signal is DC free, with a mean of zero over the cycle, then the integral of the normalized pulse can be computed simply as min/(min - max) .

For typical pulse shapes, Q_{ci} correlates well with other metrics of the contact quotient, and hence gives much the same information. A commonly used scheme is described by David Howard (here: the CQ_H): find the positive peak in the dEGG, and then the next point on the EGG that has an amplitude of 3/7 or less of the peak-to-peak amplitude. The duration between these points is divided by the period time, to yield the CQ_H. We have found [31] that for EGG waveshapes that have a contacting phase, there is a highly linear relationship between Q_{ci} and CQ_H:

$$Q_{ci} \approx 0.75 \cdot CQ_H + 0.1$$

For the common case of CQ = 0.4, which is typical of normal speech in male and female adults, Q_{ci} was also 0.4 . The usefulness of Q_{ci} lies in that it quantifies also unusual pulse shapes, without discontinuities. Discontinuities in CQ could occur, for instance, if an EGG pulse occasionally crosses a prescribed threshold more than once per cycle. Most contact quotients are fairly insensitive to small or rare instances of clipping of the EGG signal.

Notice that the Q_{ci} metric has the same problem as other definitions of the contact quotient, in that when vocal fold contact ceases, the waveform becomes low in amplitude and nearly sinusoidal. For such a waveshape, Q_{ci} will "erroneously" tend towards 0.5 (and CQ_H toward 0.523), which would correspond to rather a large amount of contact, if interpreted as normal phonation. You need to keep this in mind when considering the Q_{ci} in the voice map display. The next metric does not have this problem.

2.5.2 The Q_Δ metric

The Q_Δ metric is particularly useful for indicating whether or not vocal fold collision is occurring at all. The Q_Δ metric is the maximum positive slope of the cycle-normalized EGG signal. This is a metric of the relative rate of change of contact area. It is *not* the same thing as the speed with which the vocal folds are moving towards each other, although usually there will be a co-variation of these two speeds. The Q_Δ metric is normalized such that a sine wave receives a minimum Q_Δ of 1 . The wave shapes and their colour mapping in the voice map are shown in Table 2.

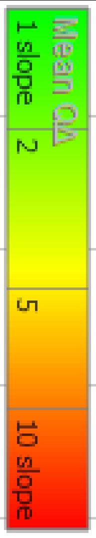
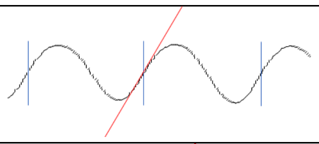
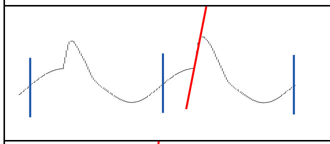
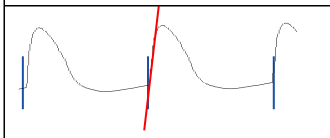
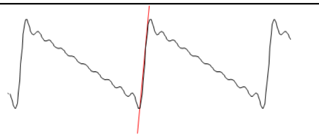
Because the EGG in FonaDyn is steeply high-pass filtered and therefore practically free of DC, the peak of the EGG derivative usually coincides closely in time with positive-going zero crossings in the EGG. An exception is for partial contacting (the second row in Table 2).

At low f_0, the Q_Δ metric converges toward the iNAQ metric proposed by C. Herbst in Enflo *et al* (2016), which in turn is similar to the inverse of the Normalized Amplitude Quotient (NAQ), introduced by P. Alku, that is often used to characterize glottal *flow*.

The Q_Δ metric, being based on a time derivative, is extra sensitive to noise in the system. White noise contains high frequencies, and when these are added to the signal, the peak derivative of the total becomes larger overall and will not descend completely down to its minimum value of 1. Even on high-quality EGG hardware, the Q_Δ of an untreated signal from soft phonation will probably not descend below 2, because of system noise. If your maps of Q_Δ are mostly in the red, even in soft phonation, then the EGG signal is far too noisy.

To deal with such noise, FonaDyn has an optional noise suppression scheme that is especially effective for low-amplitude EGG signals, as when the vocal folds are not colliding (→3.7.1). FonaDyn computes the Q_Δ metric from two different versions of the EGG pulse, with different results. The Q_Δ metric as shown in voice maps *during recording* is computed from the high-pass filtered but not yet noise-reduced signal. The reason for this is that the de-noising computations take time, increasing the total delay to about 80 ms, which could be annoying when using FonaDyn for real-time feedback. The Q_Δ metric shown in voice maps *during playback* of a pre-recorded file is computed from the filtered *and* (optionally) noise-reduced signal. The audio signal is then delayed so that the conditioned EGG and the audio remain synchronized.

Table 2. Illustration of the correspondence between EGG wave shape and the Q_Δ metric.

Mean Q_Δ	EGG shape and max slope	Interpretation
		1…1.5 Nearly sinusoidal: no vocal fold collision. This would be at very low EGG amplitudes. You will need to apply some de-noising to get Q_Δ right down to 1.
		1.5…3 Incomplete contacting: just above the threshold of collision
		3…10: normal range of contacting rates in modal voice
		N = max rate of contacting using N harmonics (synthesized with phases=0 and amplitudes 1/k)

Yet another Q_Δ is the one shown in the waveform resynthesis display of the EGG clusters panel (→3.1.7). This value is computed from the resynthesized waveform, which has a constant number N of harmonics (→2.6). It is less sensitive to noise in low-level EGG signals, and therefore it too can descend close to 1 when there is little or no vocal fold contact. Although this is the preferred variant of Q_Δ, it would take a prohibitive amount of time to compute on every cycle. (In the EGG clusters panel, it is recomputed only at the redrawing rate of the graphics.) This variant of Q_Δ is not saved by FonaDyn; but it can be obtained by post-processing FonaDyn's `Log` files in Matlab.

The Q_Δ metric is quite insensitive to small or rare instances of clipping of the EGG signal, since it is typically taken near zero crossings.

2.5.3 The I_c metric

The 'index of contacting' I_c was introduced by Ternström [4]. It is a combination of Q_{ci} and Q_Δ, namely $I_c = Q_{ci} \times \log_{10}(Q_\Delta)$. This compound metric has the useful properties of approaching zero for a pure sine waveshape, as in very weak phonation, when there is vocal fold vibration without collision; and of becoming approximately one when Q_{ci} and Q_Δ are both very high, as in very strong phonation. A physical interpretation of the I_c metric remains to be formulated, but pragmatically, it has proven to be useful.

2.5.4 The HRF$_{EGG}$ metric

The 'harmonic richness factor' is the power ratio of EGG Fourier components 2…N to the power of the fundamental, expressed in dB. HRF_{EGG} is typically very low, less than -30 dB for phonation without vocal fold contacting. It rises steeply as contacting sets in, and reaches a maximum of about +5 dB when the EGG pulses are very short (low contact quotient). It then decreases again as the EGG pulses widen with increasing VF adduction. As a contacting indicator, its information is fairly similar to that obtained from Q_{Δ} . Compared to Q_{Δ} , HRF_{EGG} is somewhat less sensitive to EGG noise, but *more* sensitive to even small or rare instances of clipping of the EGG signal.

2.6 EGG harmonic-domain analysis

The underlying ideas for the harmonic-domain analysis of the EGG were first proposed by Selamtzis and Ternström [5]. The input EGG signal is first segmented into individual cycles, as described above. Each cycle is analysed with the Discrete Fourier Transform. The resulting harmonic magnitudes and phases are then input to a so-called K-means clustering algorithm. This algorithm treats the data as point coordinates in a high-dimensional space, and allocates each point to the cluster whose centroid is closest in this space. That centroid is then updated to include the new point. This gives rise to a given number of categories, into which the cycles are classified. The clustering is 'learned' in real time, without any prior knowledge of EGG waveshapes. The clusters are colour-coded for display in the voice map, with a separate layer for each cluster. The number of categories, or clusters, must be chosen to be appropriate to the research question, usually 2...5 . Arriving at that choice usually requires some exploratory passes over the data [33]. 'Learned' cluster data can then be saved, reloaded and used to classify new signals. Section 3.7 describes how to work with clustering of EGG signals.

This analysis method emphasizes the overall periodic contacting pattern of the vocal folds. It will tend to disregard very brief or transient events that are manifest mostly at high frequencies, such as multiple contacting events that may occur very closely in time (small fractions of a cycle).

2.6.1 DFT analysis

A DFT analysis is performed of the lowest (and strongest) frequency components of the EGG signal. For N components, the program computes N value pairs (magnitude and phase) cycle-synchronously over each EGG cycle. N can be chosen as 2...20; typically it will be 6...10. Because the DFT frame length adapts to the EGG cycle length, the DFT components are equivalent to the harmonics of the EGG signal. A small quantization error arises here, in that the true EGG period time is not exactly an integer multiple of the sampling interval. In practice, though, such errors are absorbed by the subsequent statistical clustering.

The Fast Fourier Transform is *not* used, because the frame length here must be exactly one EGG cycle (to the nearest sampling interval), and because N is small. Instead, the desired DFT cosine and sine terms for the most recently completed EGG cycle are computed directly in the time domain. This also results in a constant CPU load for short and long cycles.

2.6.2 EGG waveform DFT components

Each Fourier component is a complex number with a real and an imaginary part (cartesian representation), or, equivalently, with a magnitude and a phase (polar representation). For each EGG cycle that has been judged as valid, the first Fourier component (the 'fundamental') is taken as a reference, for both magnitude and phase. The magnitudes, or absolute values, of the complex Fourier components $k \in [\,2...N\,]$

are computed, and their ratios to the magnitude of Fourier component 1 (the fundamental) are expressed as level differences ΔL_k. Using only the *relative* harmonic levels eliminates the effect of a variable gain on the EGG signal.

Similarly, the phases of the Fourier components $k \in [2...N]$ are computed relative to the phase of the fundamental, i.e., $\Delta\varphi_k = \varphi_k - \varphi_1$. The reason for computing the phases relative to φ_1 is that the cycle detection algorithm finds a cycle triggering point, whose location in the EGG cycle will depend somewhat on the current wave shape. By using the relative phases, this dependency is prevented from affecting the DFT results.

2.6.3 Level and phase of the fundamental

These values are needed in order to reconstruct the EGG waveforms from the cluster centroid values. The level of the fundamental is taken in dB down from full scale. The phase of the fundamental is taken relative to the cycle trigger point found by the cycle detection. We do not want these entities to affect the clustering, though, so they are first down-weighted by a factor of 0.001.

2.6.4 Representation of phase differences

Since the phase angles φ_k are expressed cyclically in the range $[-\pi, \pi)$, a phase greater than $\pm\pi$ will cause a 2π jump. This must be avoided, since it could give rise to falsely disjunct data clusters. One could take the absolute phase difference, folded over π, thereby avoiding jumps. The clustering would then improve, but some information is lost, and it would not be possible to reconstruct EGG waveforms from the cluster centroid values. Therefore, each relative phase $\Delta\varphi_k = \varphi_k - \varphi_1$ is instead represented by the value pair $(\cos(\Delta\varphi_k), \sin(\Delta\varphi_k))$. This eliminates the discontinuities, at the cost of an extra clustering dimension per Fourier descriptor.

From here on, the EGG harmonic-domain processing splits into two paths: DFT component clustering, and cycle-rate sample entropy estimation based on the DFT components. The entropy estimation is described first.

2.7 Sample entropy of EGG cycle data

In information theory and in physics, *entropy* is a measure of disorder. An entirely ordered, entirely predictable process has an entropy of zero. The ‘sample entropy’ of a signal [11][12][13] is an interesting metric that has found numerous biomedical applications. For brevity, it is often called SampEn. The SampEn is low for a regular, self-similar signal and high when a signal is transient, erratic, or noisy. While others [14] with some success have taken the SampEn of high-rate, isochronous signals such as sampled EGG or audio, we have found that the SampEn metric is particularly effective for phonation data that are cycle-synchronous. We call this the ‘cycle-rate sample entropy’ (CSE).

2.7.1 Behaviour

Very usefully, the CSE is zero when the vocal fold contacting pattern is stable, and it increases with decreasing stability. Unlike conventional jitter and shimmer measures, which report on changes in period time and amplitude, respectively, the CSE is fairly insensitive to such variations. This is because FonaDyn normalizes the EGG pulses in amplitude and period time, prior to computing the CSE. Also, aerodynamic turbulence in the larynx does not contribute substantially to the entropy of the EGG signal, since it is the VF contacting that is analyzed. CSE reports on cycle-to-cycle changes in the EGG waveform shape only. Typically, CSE decreases substantially at the onset of vocal fold contacting, analogously to what has been observed for other perturbation metrics such as jitter. In normal voices, it then bottoms out to zero as vocal effort increases.

2.7.2 Parameters

The SampEn analysis in FonaDyn is still somewhat experimental, so we have given you access to all the settings, to try out (→3.1.6). It has three parameters: Tolerance, Window and Length, which can be set to different values for the harmonic levels and the phases. The default values for Tolerance, Window length and sequence Length (the last two given in EGG cycles) have been found by trial and error to work well for assessing the instability of the EGG wave shape, but they may need to be adjusted in different scenarios.

Increasing the Window makes a smoother curve and reduces the temporal resolution. Increasing the Length can make the SampEn peaks more localized. The threshold or ‘tolerance’ parameter keeps the sample entropy at zero while phonation is stable, even with changing pitch, but when ‘something unusual’ happens, the SampEn peaks. This ‘something’ could be a voice break, such as those occurring between chest and falsetto voice [5], or other instabilities in phonation. The ‘Tolerance’ setting is somewhat analogous to a noise threshold. Increasing the Tolerance suppresses the influence of small variations, for instance if you want to detect only voice breaks. Setting the Tolerance to something small (<0.1, with Length=1) allows analysis of the stability of the phonated tones.

2.7.3 Computation

In FonaDyn, the CSE of the changing EGG wave shapes is computed as follows. The harmonic-domain analysis produces a vector of values that is updated on every phonatory cycle, when we obtain new values of the level and the phase, for all N harmonics. Each element in this vector provides the input to one of several SampEn estimators, running in parallel for all levels and phases of the first few harmonics. The final CSE value is simply the sum of the SampEns for the N levels and phases. The number of harmonics is selectable; typically it will be smaller than for the clustering analysis. A local SampEn is computed over a short sliding window of w glottal cycles, where the integer w can be chosen by the user (‘Window’). The window is advanced one cycle at a time, so a new value is obtained on every cycle. Another parameter is the sub-sequence length (‘Length’). The algorithm used for estimating the SampEn is given in [15].

For computing the sample entropy of the phases, we have potentially the same wrap-around problem as described in section 2.6.4. Jumps of $\pm 2\pi$ would give large but meaningless peaks in the SampEn. Here, this is avoided by substituting the phase with its absolute value. This gives a continuous signal, for all phases; and the resulting ambiguity is of little consequence for the SampEn estimation.

2.8 EGG clustering

2.8.1 Clustering dimensions

As described above, for each harmonic $k \in [\, 2 \ldots N \,]$, FonaDyn computes the relative level, and the cosine and sine of the relative phase. For each EGG cycle, this results in $3 \times (N\text{-}1)$ values. Using these values, plus the estimated level of the residual energy and the value pair $[\, \cos(\varphi_1), \sin(\varphi_1) \,]$, the statistical clustering is performed in a space with $3 \times N$ dimensions. This is done using a real-time implementation of the algorithm 'online Hartigan *k*-means' [10]. Compared to other methods for clustering, the *k*-means method has these advantages: (1) it computes quickly even in many dimensions, and (2) the *number* of points already in the clusters does not affect the classification, only the centroid updates. The latter means that, in a data set with thousands of EGG cycles, a small minority of cycles of an unusual shape can still give rise to a cluster of their own, especially if they occur early in the recording.

For the clustering to be effective, all dimensions should have values extending over roughly the same numerical range. For this reason, the level difference values as fed to the clusterer are expressed in Bels rather than decibels. This makes for a better match to the (cos, sin) values, which are always in the range [-1…1].

Note that *none* of f_o, SPL and total EGG amplitude are input as features to the clustering algorithm. Also, the positions in the clustering space of the cluster centroids are *not* related to the locations of the coloured regions in the voice map. Rather, each centroid represents a particular EGG pulse shape. This means that affinity to a cluster is *not* given by proximity in the f_o/SPL-plane. However, different EGG pulse shapes do tend to occupy connected regions in the voice map, typically with some overlap. It is these regions that make up FonaDyn's 'cluster maps' of the EGG signal.

2.8.2 Resynthesis

In order to facilitate the user's interpretation of the clustering, the cluster centroid values of levels, cos and sin are used also to resynthesize and display the approximated cycle waveform (section 3.1.7), by addition of cosines. A potential problem here is that the cluster centroid values for sin and cos, since they are not strictly paired, no longer necessarily fulfil the trigonometric identity $\cos^2(\varphi) + \sin^2(\varphi) = 1$. Another potential problem is that the probability distributions of sin and cos are very far from rectangular. In practice, though, tests with synthetic waveforms (triangle, sawtooth, square) have shown that such reconstruction works well enough.

Note that FonaDyn resynthesizes the EGG wave shape only so that the pulse shape thus approximated can be displayed, and so that its parameters can be computed, cycle by cycle. The resynthesized cycles are not concatenated back into a contiguous signal.

2.8.3 Limitations

The EGG analysis considers only the $N \leq 20$ lowest harmonics of the EGG signal, and typically 5 to 10 of them are used. This means that some high-frequency aspects, such as multiple contacting events in quick succession, might not be resolved. Usually, though, the harmonics 11…20 behave very similarly and thus would not add much information.

The preconditioning of the EGG signal means that the f_0 should be 80 Hz or greater. The further this bound is crossed, the more the clustering will start to depend on f_0.

2.9 Phonation type clustering

Different kinds of phonation, such as 'breathy', 'pressed', 'modal', 'falsetto' etc., are more accurately characterized by combinations of metrics than by individual metrics. For any combination of the acoustic and EGG metrics (2.3.4, 2.5), FonaDyn can perform a second kind of clustering that finds the most prevalent combinations. For an introduction to the reasoning behind this mode of clustering, please see [33].

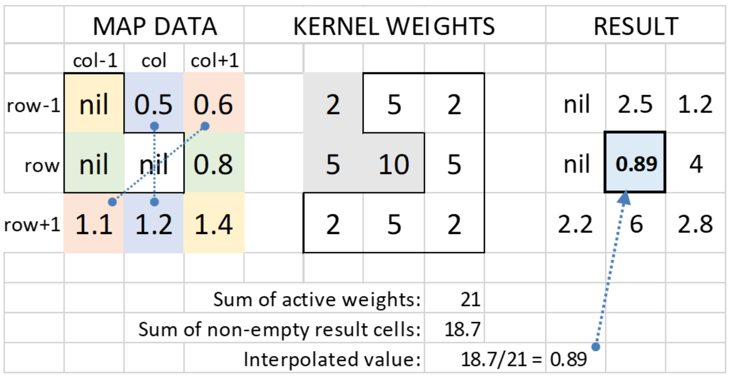
Phonation type clustering is simpler to run and more straightforward to interpret than is the EGG clustering, but it is a bit more complex to set up. Each metric needs to be normalized differently, to 0% … 100% of its expected range of variation. In other words, some population data are needed *a priori*. Section 3.5.11 describes how to work with phonation type clustering.

Most likely, the study and assessment of vocal pathologies, singing styles etc, will call for context-specific clustering, but this remains to be determined. The good news is that FonaDyn makes it possible.

2.10 Map smoothing

The cell colours in a map usually represent metric averages. Because averaging has a smoothing effect, voice maps become smoother and less ‘grainy’ the more cells that are filled, and the more repeated visits that are made to each cell. But this takes time, and the participant’s voice might even ‘warm up’ or otherwise change during the procedure. To produce a similar result in a shorter time, FonaDyn can interpolate across scattered empty cells, and smoothen the differences between adjacent cells. This gives a better-looking map that is still representative of the participant’s voice. See section 3.1.11 for how to invoke this smoothing.

The smoothing algorithm works by performing a 2D convolution of a kernel of 3×3 cells with the original cells. Each cell in the ‘destination’ map is assigned the kernel-weighted average of cells in the original ‘source’ map. Empty cells in the source map have the value ‘nil’, which when multiplied by a weight still gives a ‘nil’ cell in the destination map. However, if an empty source cell has pairs of non-empty neighbours on at least two opposing sides or corners, the destination cell is assigned the weighted average of all surrounding source cells. Figure 5 shows how this results in a smoothed *and* interpolated value.



	MAP DATA			KERNEL WEIGHTS			RESULT		
	col-1	col	col+1						
row-1	nil	0.5	0.6	2	5	2	nil	2.5	1.2
row	nil	nil	0.8	5	10	5	nil	0.89	4
row+1	1.1	1.2	1.4	2	5	2	2.2	6	2.8

Sum of active weights:	21
Sum of non-empty result cells:	18.7
Interpolated value:	18.7/21 = 0.89

Figure 5. Example of generating an interpolated and smoothed value for one cell, i.e., the one in the center. The value 0.89 is the result for the center cell in the destination map, which was empty to begin with. The colours illustrate the possible pairs.

The cell value to be smoothed is the one in the center, at [row, col]. It is originally empty, but has opposing non-empty neighbours vertically (and diagonally), and so it qualifies for interpolation. Empty or not, its value in the smoothed destination map becomes the kernel-weighted average of all non-empty cells in the source map. This is repeated for all cells, and in all layers of the map.

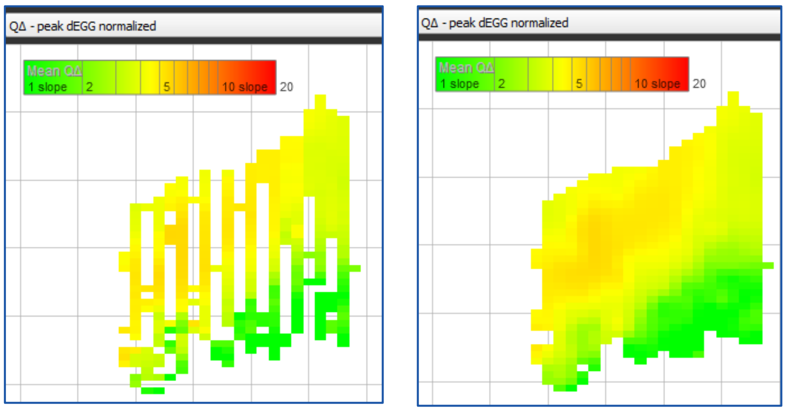


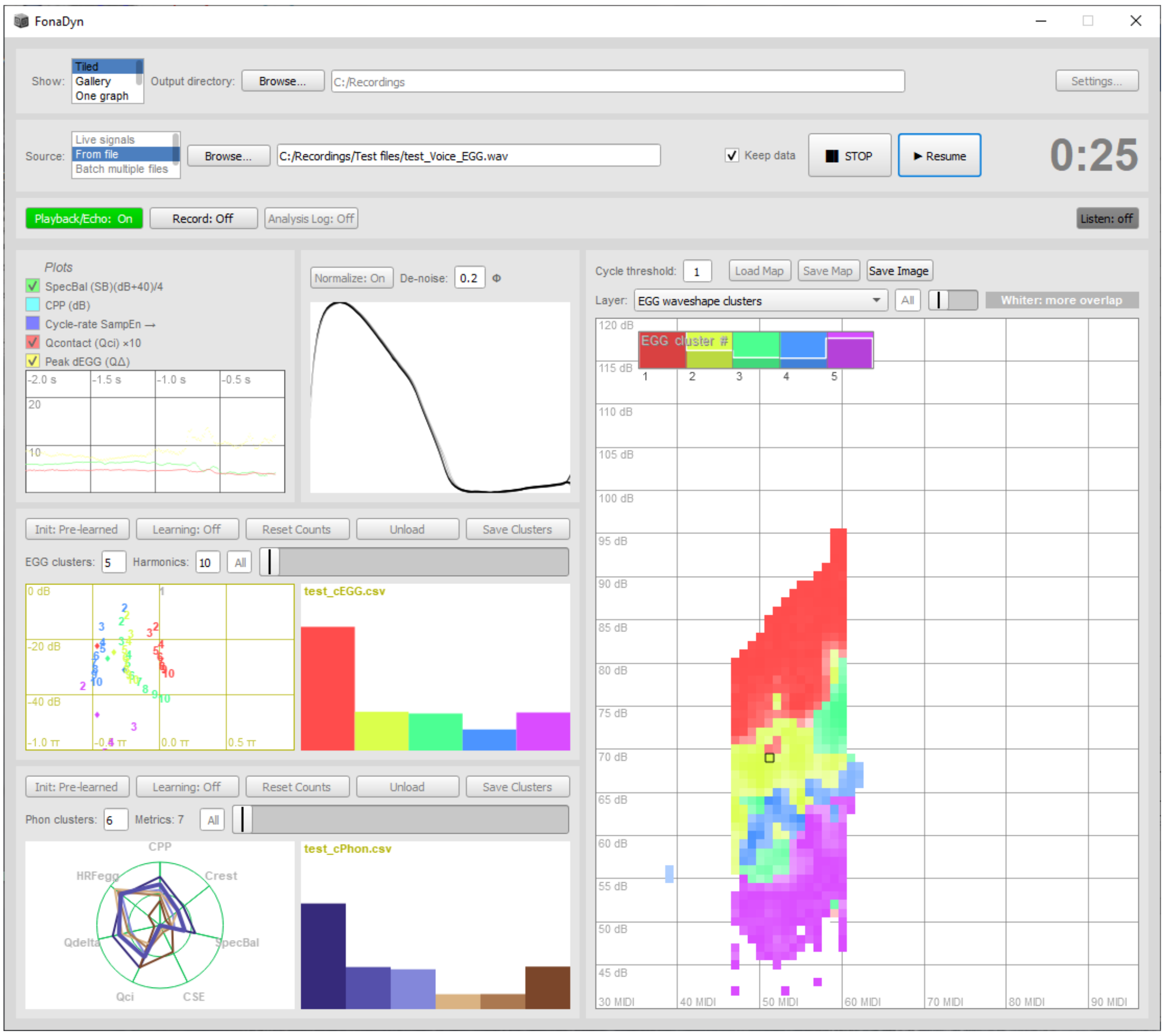
Figure 6. An example of smoothing and interpolation: left, an original map made by singing soft-loud-soft on every second semitone [31]; right, the same data after smoothing and interpolation.

3 PART THREE - Using FonaDyn

3.1 Window layout

3.1.1 Main screen

The FonaDyn main screen is a bit like the control panel of a machine. It has no pull-down menus at the top, and almost all controls are visible. It has quite a few graphs, which can be hidden or shown, as desired. In Figure 7, nearly all the possible panels are visible for reference, but normally you would hide some of them (→3.1.12).



Output directory: C:/Recordings

C:/Recordings/Test files/test_Voice_EGG.wav

✓ Keep data

0:25

Normalize: On De-noise: 0.2

Cycle threshold: 1

Layer: EGG waveshape clusters

EGG clusters: 5 Harmonics: 10

Phon clusters: 6 Metrics: 7

Figure 7. FonaDyn's main display window. The layout is described in Table 3.

The top row holds general settings. The second row selects the signal for input, controls START/STOP/PAUSE, and displays the time since START. The buttons in the third row control the output options. The subpanels with graphs show the analysis results, in real time, as described below.

Table 3. Guide to the layout of the main window.

Row 1: Graphs layout, Output directory, Settings		
Row 2: Inputs: live signals, file, file batch, or script; Keep Data; Start/Stop, Pause/Resume; clock		
Row 3: Outputs: playback, record to file, log file (+ optional time files), listen		
Selected time plots: Values during the last 1-10 s	Incoming EGG waveform oscilloscope, de-noising	Voice map panel ↑ SPL f_o →
EGG cluster centroids Harmonic levels & phases plot	Cycle counts per cluster, or EGG wave shapes	
Phonation type centroids radar plot	Cycle counts per cluster, or clustering settings	

You can use Tab or Shift+Tab to move the keyboard focus from one control to the next. *Numbers* can be edited with the keyboard, or by dragging the mouse cursor vertically, or with the up-arrow and down-arrow keys. Ctrl and Shift increase the step size.

3.1.2 Choice of Source

For visualizing and recording live signals, choose **Source: Live signals**. If you are interested only in the real-time displays of live signals, as for visual feedback of speech or singing, then continue by pressing only ►START. The displays will be active, but nothing will be recorded. For recording to take place when ►START is pressed, first press the **Record** button. See section 3.2 for more on recording.

To analyse pre-recorded files, choose **Source: From file**. 'From file' is the most common use case, so it is the default. Browse to the file in the usual Open File dialog box, or drag it from a file folder outside FonaDyn and drop it onto the file name field. Only files whose names end in `_Voice_EGG.wav` can be dropped in this way.

One often has reason to run and re-run analyses on multiple files. For small batches with the same analysis settings, you can queue a moderate number of input signal files, by choosing **Source: Batch multiple files**. This choice is useful also for a list of signal files to be conveniently accessed during a presentation or teaching lab session, for instance. The file that is selected in the list will be the first to play when ►START is pressed.

Finally, the **Source** list also contains the choice **Run script**. A script is a text file that you create, with commands for setting the many analysis parameters, and for running and saving analyses. This lets you create setups, or batches that can run for hours, while you are doing something else. Scripts are described in section 3.5.8.

3.1.3 The ‘Moving EGG’

Following an idea by Christian Herbst, this panel displays the incoming EGG cycles in real time, like a triggered oscilloscope. The signal shown is the EGG signal after pre-processing (→2.4). The period length is normalized to the width of the display frame, so it is not affected by the voice fundamental frequency. By default, the peak-to-peak amplitude is also normalized, to the height of the display frame. To see the actual EGG amplitude, relative to full scale, choose **Normalize: Off**. This is useful for checking the amplitude of the incoming EGG signal. It does not change how the signal is processed. The number in the **De-noise** field controls the spectral threshold used for de-noising the EGG signal (→3.7.1).

This panel shows the EGG waveform only when one is considered to exist, that is, when the *audio* signal exceeds the chosen clarity threshold (→2.3.4). Otherwise, the graph is blank.

Tip: you can press Alt+M to hide or show the Moving EGG panel.

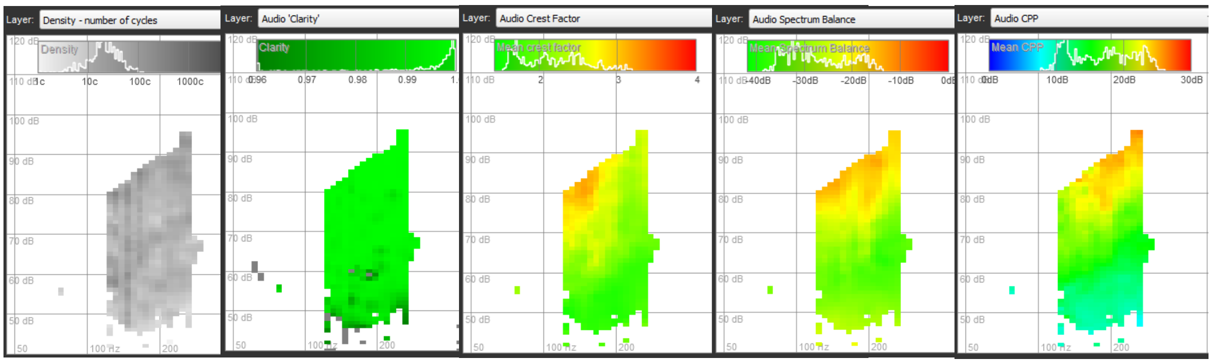
The Moving EGG display draws the n=5 most recent cycles with a fading gray scale. The cycle curves are rendered as k=80 straight line segments. To modify this rendering, press **Settings…** (→3.1.10) and check the box **Show additional diagnostic features**. This activates the display of some extra control fields. Change n by entering a different value for **Count**. Change k by entering a different value for **Samples**. Increasing Count or Samples may give a nicer image, but also increases the processing load.

The rightmost symbol at the top indicates the type of cycle segmentation (→2.4.2), with Φ for the phase tracker, and Λ for the peak follower. You can change this option, too, in the **Settings…** dialog box.

3.1.4 The Voice Map

The Voice Map panel displays a voice map of one of several acoustic and EGG metrics, each in its own ‘layer’ (Figures 8-10). The horizontal axis is the f_o in semitones (57 MIDI = 220 Hz, 60 MIDI = middle C); or, right-click to display the f_o axis in Hz. The vertical axis is the sound level in dB. This must be calibrated to correspond to the SPL at 0.3 m microphone distance (→3.2.9). The ‘pixels’ in the voice map are called cells. Each cell is 1 decibel high and one semitone wide.

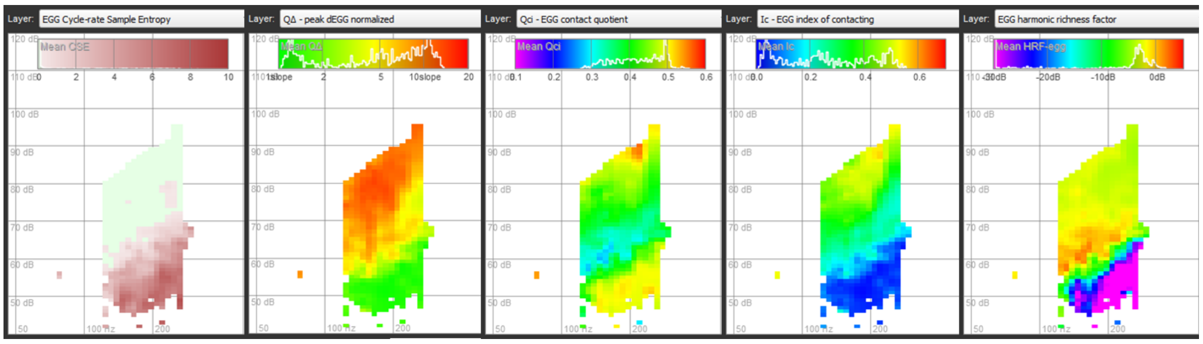
The axis extents are fixed, to facilitate comparisons between maps; however, by default, the aspect ratio is not fixed. It will change when you resize the main window, or rearrange the panel layout. Although changing the aspect ratio runs against the UEP recommendations, it maximizes the use of the display screen area. If you prefer to fix the aspect ratio to 2:1, then insert into the startup file the statement `FonaDyn.config(fixedAspectRatio: true);`


Layer: Density - number of cycles
Layer: Audio 'Clarity'
Layer: Audio Crest Factor
Layer: Audio Spectrum Balance
Layer: Audio CPP

(a) Density (b) Clarity (c) Crest factor (d) Spectrum Balance (e) CPP

Figure 8. The first five layers of the Voice Map display are derived from the audio signal.

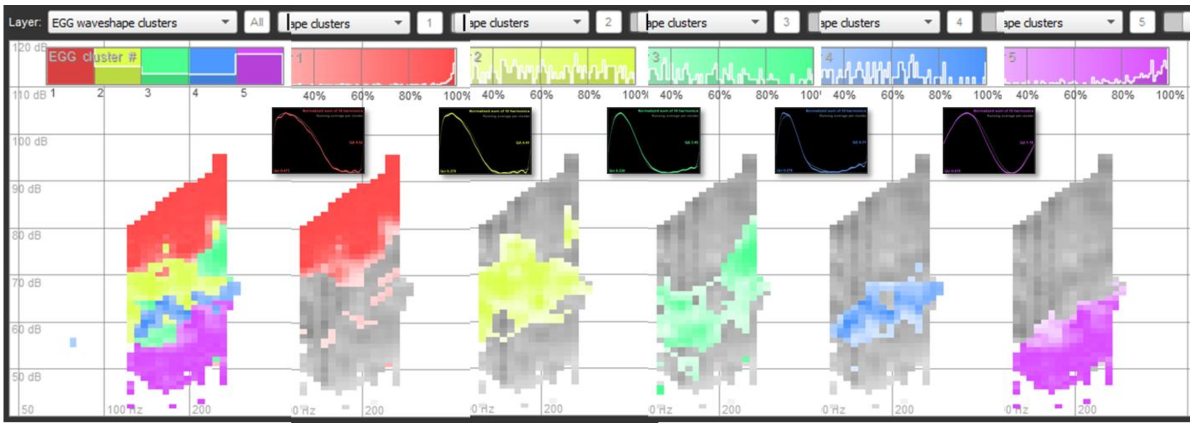
For the examples in Figures 8-10, an amateur male singer repeated soft-loud-soft /ɑ:/ vowels on several constant pitches over more than an octave. This recording (“`test_Voice_EGG.wav`”) took about 70 seconds to make. The figures show the many layers of a voice map, each of which maps a different metric. Metrics (a) to (e) are derived from the audio signal (→2.3.3), while the rest are derived from the EGG signal. Figure 8: (a) ‘Density’, where the darkest gray means >10,000 cycles in one cell. (b) ‘Clarity’ showing accepted cycles (green) and rejected cycles (gray). (c) The mean of the crest factor (peak-to-RMS ratio) of the audio signal, where red means 4 (corresponding to 12 dB). (d) The mean spectrum balance, being the ratio in dB between the signal powers above 2 kHz and below 1.5 kHz. (e) The cepstrum peak prominence (CPP) in dB.


Layer: EGG Cycle-rate Sample Entropy
Layer: QΔ - peak dEGG normalized
Layer: Qci - EGG contact quotient
Layer: Ic - EGG index of contacting
Layer: EGG harmonic richness factor

(f) Cycle-rate Sample Entropy (*CSE*) (g) Normalized peak dEGG (Q_{Δ}) (h) Quotient of contact by integ-ration (Q_{ci}) (i) Index of contacting (I_c) (j) Harmonic Richness Factor (*HRFegg*)

Figure 9. Layers 6-10 of the Voice Map display are derived from the EGG signal.

Figure 9 shows the layers for the metrics derived from the EGG signal. (f) The cycle-rate sample entropy (CSE); (g) the normalized peak dEGG Q_{Δ}; (h) the contact quotient by integration Q_{ci}; (i) the index of contacting I_c (→2.5), and (j) the HRF of the EGG signal. For making this Voice Map, the De-noise value was set to 0.2, and the map was smoothed.



(*k*) Dominant cluster regions (*l*) Regions of clusters 1-5, with corresponding EGG wave shapes

Figure 10. A compilation of Voice Map displays of EGG cluster extents (layers 11...11+N)

Figure 10 shows an example of how the automatic clustering of EGG wave shapes is mapped into the voice field (→2.7). The default is to use 5 clusters, coded by colour. Each cluster corresponds to an average EGG wave shape, and tends to form its own region in the map (Figure *10*l). A horizontal slider appears: use it to select separate sub-layers for one cluster at a time. In these sub-layers, the colour saturation indicates the relative occurrence of cycles belonging to the cluster. When the colour in a cell is fully saturated, it means that all of the cycles occurring in that cell were classified as belonging to the displayed cluster. This gives an idea of the amount of overlap between the clusters. The Density plot from Figure 8(a) is then shown in the background. Interestingly, we see that the vocal folds can vibrate without contacting at SPLs as high as 60-70 dB (or much higher, at higher pitches); as shown by the blue area in

Figure 9(*i*) or the purple area in Figure 10(*k*). For this example, the averaged wave shapes that correspond to each colour in can be seen also in

Figure *13*(*c*). After the automatic clustering, the colours have been sorted manually (→3.7.6) in order of descending vocal effort: 'red' cluster wave-shapes, strongest phonation; 'yellow' cluster wave-shapes; here, firm phonation with full vocal fold contact; 'green' and 'blue' cluster wave-shapes; here, fairly soft phonation with brief contact; 'purple' cluster: softest phonation with no vocal fold contact, as indicated by a nearly sinusoidal EGG. Figure 10 (*k*) shows the overlay of all five wave-shape regions, with the dominant EGG wave-shape cluster by colour. The 'dominant' wave-shape is the one that deposited the most cycles in a given cell. Less colour saturation (whiter colour) signifies more co-existence of wave shapes from different clusters in the same cell.

The display of *phonation type* clusters (→2.9) follows the same scheme as for the EGG wave-shape clusters, but using a different colour palette.

The **colour bar** at the top left displays a relative **distribution of the metric** on its own axis, as counted by the number of cells in the map. The toned area under the white curve corresponds to the total number of cells in the map. To reconstruct this distribution graph outside of FonaDyn, make a histogram of the corresponding column in the `_VRP.csv` file (next section).

During analysis, all the 12 layers of the voice map plus the individual cluster layers (5+5 by default) are updated in real time. Also, several layers can be displayed simultaneously (→3.1.12). It is quite instructive to watch and listen at the same time.

3.1.5 Saving and loading voice maps

All the data shown in figures 8-10 can be saved to a single .CSV text file, for further analysis in other software (→3.5). Press **Save Map** to open a standard File Save dialog box. Type in a filename (or choose an existing file to overwrite). If the filename you enter does not end in `.csv`, then FonaDyn will append `_VRP.csv` to the name. If the filename you enter *does* end in `.csv`, then FonaDyn will not change it.[1]

You can load a previously saved map by pressing **Load Map**. This is useful for inspecting maps from earlier recordings. It also lets you accumulate more data into an existing map: just check the **Keep data** box before starting. To accumulate more data, you must also continue to use the same settings and cluster data as were used to make the loaded map, or the clustering/classification results will be meaningless. To keep track of and restore such contexts when needed, you can have FonaDyn create a 'context script' (→3.5.9), and run that script instead of choosing **Load Map**.

The **Cycle threshold** field specifies the minimum number of phonatory cycles required for a cell to be displayed. It is useful for hiding spurious outliers, and for ensuring that the displayed colour corresponds to a mean of several cycles. This value is not saved with the `_VRP.csv` file; the file as saved will always contain a row for every cell that was visited at least once. If you want to remove cells with fewer cycles than the threshold from the .csv file, open it in a spreadsheet app, sort the .csv file by the column "Total", delete those rows, and save it again as .csv. FonaDyn does not care in which order the rows appear.

To save an *image* of the voice map, first resize the FonaDyn window to your preference; then press **Save Image**. A partial screen dump is generated and shown in a preview window of its own. When you press 'F', a Save File dialog appears. Also, a list of available raster image file formats is output into SuperCollider's Post window. Choose a format, and *type it in as the extension* to the image filename that you specify. The image as shown on screen will be saved to the chosen format. Or, you can leave the preview window open, for easy visual comparison with your *next* voice map. Press ESC to close it. Several image windows can be open at the same time. Press 'C' to close all of them at once. Figures 8-10 in this section were made in just a few minutes, using **Save Image** multiple times, and then mounting and cropping the image files in a slide presentation program.

For vector images with publication quality, you can create figures in Matlab® using the m-files in the 'matlab' folder.

Tip: you can press Alt+V at any time to hide or show the Voice Map panel.

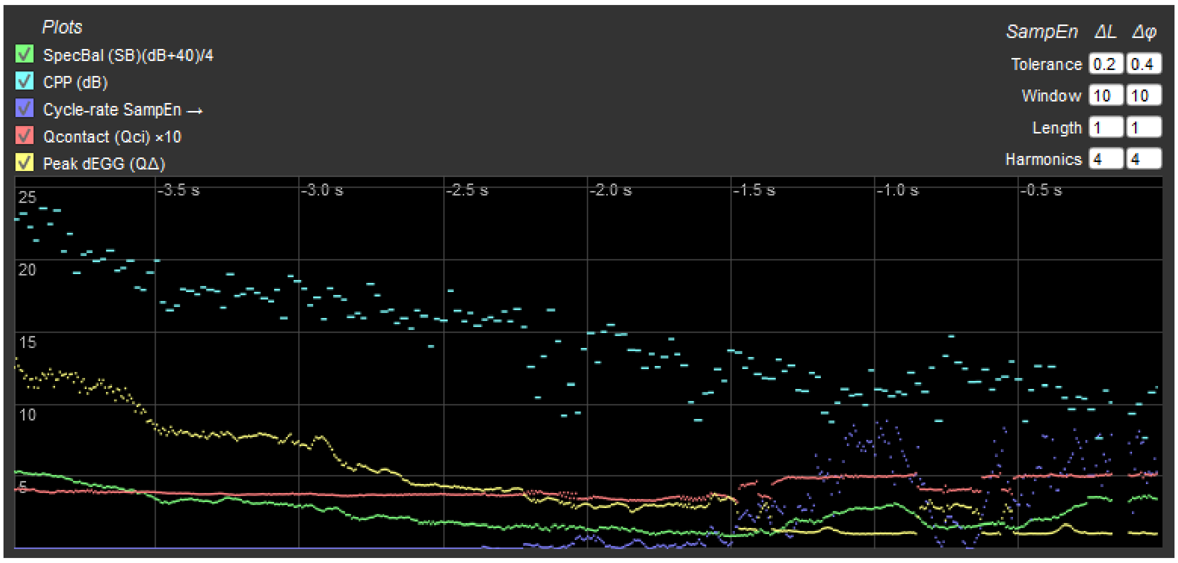
Tip: you can open a map file whose name ends in `_VRP.csv` by dragging the file from its folder and dropping it onto the map area, or onto the **Load Map** button.

[1]) The "_VRP" is a legacy label; "_VMap" would now be more appropriate. But there it is.

3.1.6 The Plots

The Plots panel can show time series of up to five metrics. The live display scrolls, showing the most recent two seconds by default. You can change the duration of the plot (1 to 10 seconds) by dragging the grid sideways with the mouse, even when the plot is running. Durations longer than 2 seconds need to be set before the ▸ START button is pressed.

Each fleck represents data from one phonatory cycle.[2] This display is the most CPU-intensive one, especially if the time axis is long; so if your computer is having problems keeping up, limit the number of curves, or hide this panel entirely (Alt-P).



SampEn	ΔL	Δφ
Tolerance	0.2	0.4
Window	10	10
Length	1	1
Harmonics	4	4

Figure 11. The Plots panel. Select the desired plots with the coloured check boxes. To change the displayed duration, drag the mouse cursor sideways on the grid.

You can choose from the two time-domain EGG metrics Q_{ci} and Q_{Δ} (→2.5), plus the audio spectrum balance, the CPP, and the CSE. Whenever the 'clarity' metric of the audio signal drops under its threshold, there will be a gap in all of these traces.

In the Plots panel, the SampEn controls at top right appear only when the Cycle SampEn box is checked (CSE). The CSE metric is computed from the cycle-time series of the EGG harmonic magnitudes and phases (→2.7). By default, only the first four harmonics (i.e., numbers 2...5, relative to the fundamental) are used. The CSE value shown by the scrolling lilac curve is the sum of those eight SampEn values.

The time history of these and all other metrics can be saved into a multichannel Log file, cycle by cycle, or at an isochronous frame rate (→3.5.3). For making and saving customized time-series plots of any metric in FonaDyn, create an Analysis Log file, and then use Matlab or some other software to customize your graphs from that. The supplied Matlab file `runDemos.m` demonstrates how to do this.

Tip: you can press Alt+P at any time to hide or show the Plots panel.

[2]) The CPP metric (light blue) is not cycle-synchronous, but is updated every 23 ms; so its flecks look longer.

3.1.7 Clustering

FonaDyn clusters both EGG wave-shapes and phonation types, independently of each other. Both types of clustering have the potential for identifying distinct modes of voice production.

The two control panels each have the same strip of five buttons along the top:

Button	**States**	**Action**
Init	Relearn	On START, clear the current cluster data, to learn afresh
	Pre-learned	On START, keep the currently loaded cluster data, for continued learning, or for classification
Learning	On	Continue learning, updating the centroids
	Off	Classify incoming data, without updating the centroids
Reset Counts	<push when running>	Set the cycle counts of all clusters to zero, now. All centroids are initialized to the current values – but they soon diverge. Useful for initializing, and for clearing spurious outlier centroids.
	<push when stopped>	Arm for an Auto Reset: clustering begins as soon as some stable phonation is detected.
Load Clusters	Load	Load a centroids data file, for initialization or for classification
	Unload	Clear all centroid data
Save Clusters	<push>	Save the centroid data to a file (`*_cEGG.csv` or `*_cPhon.csv`)

If the filename you enter for **Save** does not end in `.csv`, FonaDyn will append `_cEGG.csv` or `_cPhon.csv` to the name. If the filename that you enter *does* end in `.csv`, then FonaDyn will not change it.

A *cluster centroid* is a set of values that specifies the average location of a cluster of data points in the clustering space. In the centroids displays, each *colour* corresponds to one centroid (and, equivalently, to one cluster).

With EGG clustering, each centroid represents a commonly occurring EGG wave shape (→2.6). New points are added with every phonatory cycle.

With phonation type clustering, each centroid represents a commonly occurring combination of values of acoustic and/or EGG metrics. New points are added at a fixed rate of 44100/64 ≈ 690 Hz.

3.1.8 EGG clusters

In the left panel of Figure 12, one EGG cluster centroid (green) is shown.

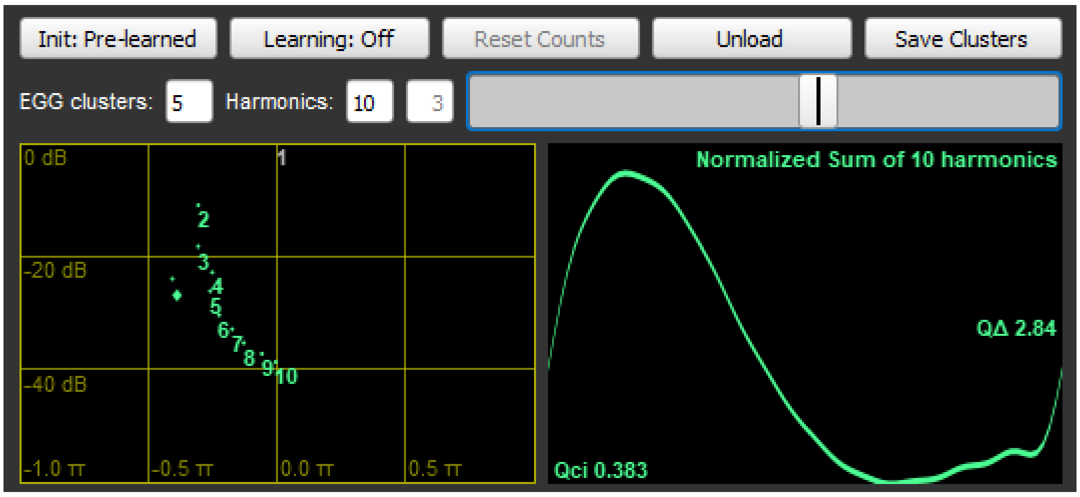
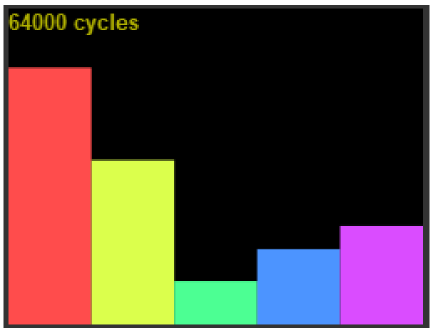


Figure 12. The graphs in the EGG Clusters panel.

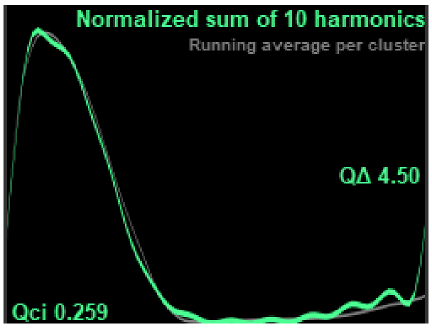
The grid to the left shows the levels and phases of the EGG harmonics, relative to the fundamental. By definition, the fundamental or first partial tone has the level 0 dB and the phase 0 radians. The gray '1' at top center (0 dB, zero phase) is thus implicit for the first partial, and is shown for visual reference only. The exact coordinates are shown by small points are at the top-left corner of the glyphs. In Figure 12, the second harmonic '2' has a *relative* level of about -10 dB and a *relative* phase of about -0.3π radians. If you were to string out the digits in sequence from left to right, keeping their heights, you would get the typical power spectrum of the EGG pulses in the given cluster. Finally, the marker ♦ shows the average level of the fundamental of this wave shape, in absolute dB relative to a full-scale signal; and also its phase, relative to the cycle trigger point. These latter level and phase do not contribute the clustering, since our philosophy is to disregard the amplitude of the EGG signal. However, they are needed for reconstructing the waveforms. For setting the fields **EGG clusters** and **Harmonics**, see section 3.7.3.

The graph to the right has four different display modes, as shown in Figure 13. By clicking on this graph, you can toggle between a bar graph, a single clustered waveform or all clustered waveforms. The horizontal scroll bar, too, selects one or all clusters. When you select a cluster from this panel, the map displays also follow suit.

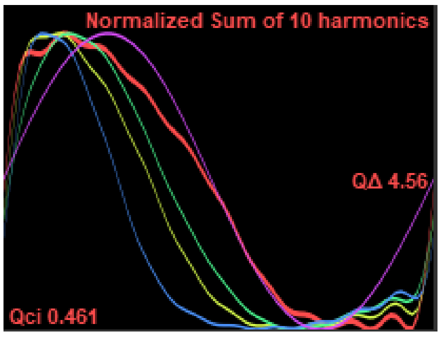
> The rippling which can be seen in the resynthesized curves is an artefact of truncating the spectrum after a few harmonics. It is known as Gibb's phenomenon, and it is not a property of the EGG signal. If it annoys you, it can be suppressed by checking the button **Settings... | Suppress Gibbs' ringing** (→3.1.10). This affects only the display, not the analysis. Because this suppression performs a kind of low-pass filtering, it will also affect the displayed value of Q_Δ next to the curve. However, that value is not saved anywhere, and the values stored in the voice map are not affected.



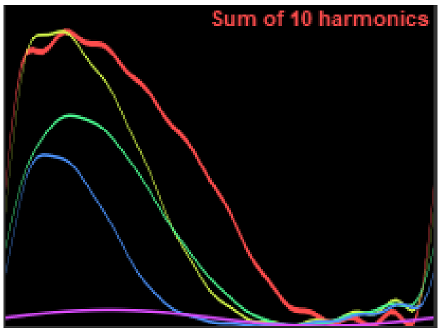
a) *Bar graph mode, with EGG cycle counts, per cluster. The number at top left is the scale of the vertical axis, which rescales automatically. Click on one of the bars to display (b).*
Click near the top of this panel to display (c).



b) *EGG wave resynthesis display, with one wave-shape selected. Time and amplitude are cycle-normalized. Vocal fold contact area increases upwards. Green line: EEG pulse resynthesized from one of the cluster centroids. Gray line: a running average of recent pulses in this cluster. The values of Q_{ci} and Q_{Δ} shown here are computed from this resynthesized waveform, and are not saved. Click on the graph to display (a).*



c) *EGG wave resynthesis display, with all wave-shapes selected. Time and amplitude are cycle-normalized. Here, for example, the sinusoidal purple signal (no vocal fold contact) is actually much weaker than the others, while the red one is the strongest. During analysis, a thin gray line shows a running average of the most recent EGG pulses that were assigned to this cluster. Click on the graph to display (a).*



d) *As in (c), but with the relative amplitudes of the wave-shapes preserved. You get this display by choosing Normalize: Off in the 'Moving EGG' panel. We see that the purple waveform, which here represents no vocal fold contact, has a much lower amplitude than the rest. The Normalize On/Off button does not affect the processing of EGG signals; only how they are displayed.*

Figure 13. The display modes of the Cluster panel.

Tip: you can press Alt+C to hide or show the Clusters panel.

Tip: you can open a clusters file whose name ends in `_cEGG.csv` by dragging the file from its folder and dropping it onto either of the cluster graphs. You'll have to **Unload** first.

3.1.9 Phonation type clusters

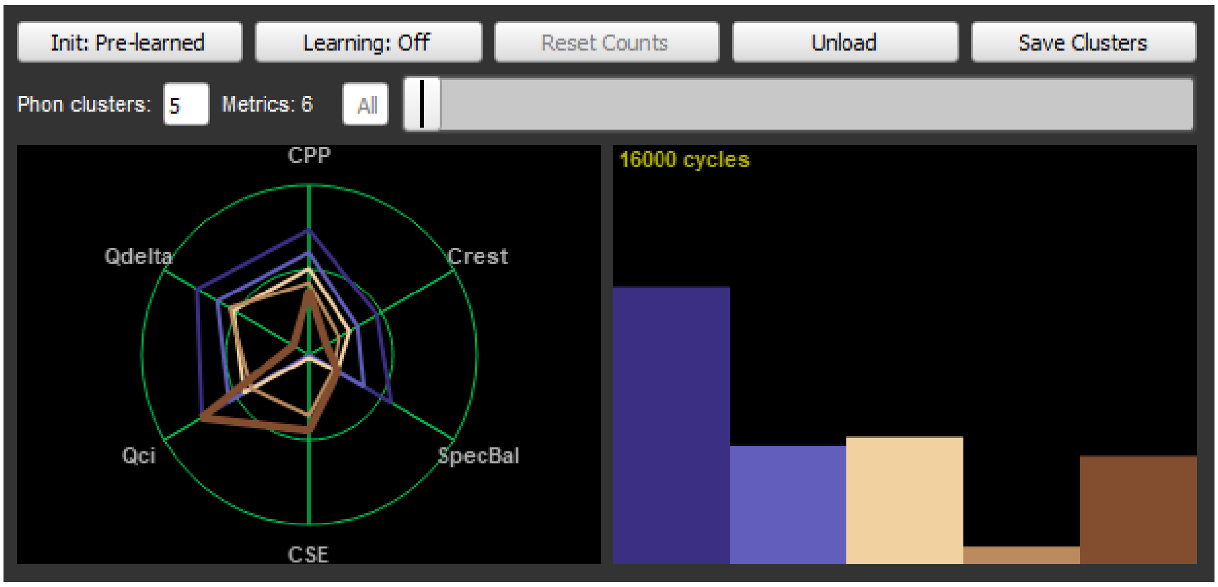


Figure 14. The graphs in the phonation type clusters panel.

Figure 14 shows an example in which six metrics have been chosen, for clustering into five clusters. The cluster centroids are shown in a ‘radar plot’ to the left, each in its own colour. Readers who are familiar with the Multi-Dimensional Voice Program® (MDVP) by KayPentax will recognize this representation, although here the selection of metrics is different. FonaDyn augments it by visualizing also how it changes across the voice range. The metrics whose values are being clustered are arranged around the perimeter of the radar plot. Of these, the CPP, Crest factor and Spectrum Balance are acoustic metrics, while CSE, Q_{ci} and Q_{Δ} are EGG metrics. The accumulated number of cycles in each cluster is shown in the bar graph to the right. You can choose and order any or all of nine metrics for clustering phonation types, by editing the corresponding `_cPhon.csv` file (→3.5.6).

Individual centroids can be selected with the horizontal slider, or by clicking on the respective bar. If there is a cluster display in the voice map, it will follow this selection. By clicking on the *left* graph, you can toggle between the bar graph and a control panel for the centroids (Figure 15).

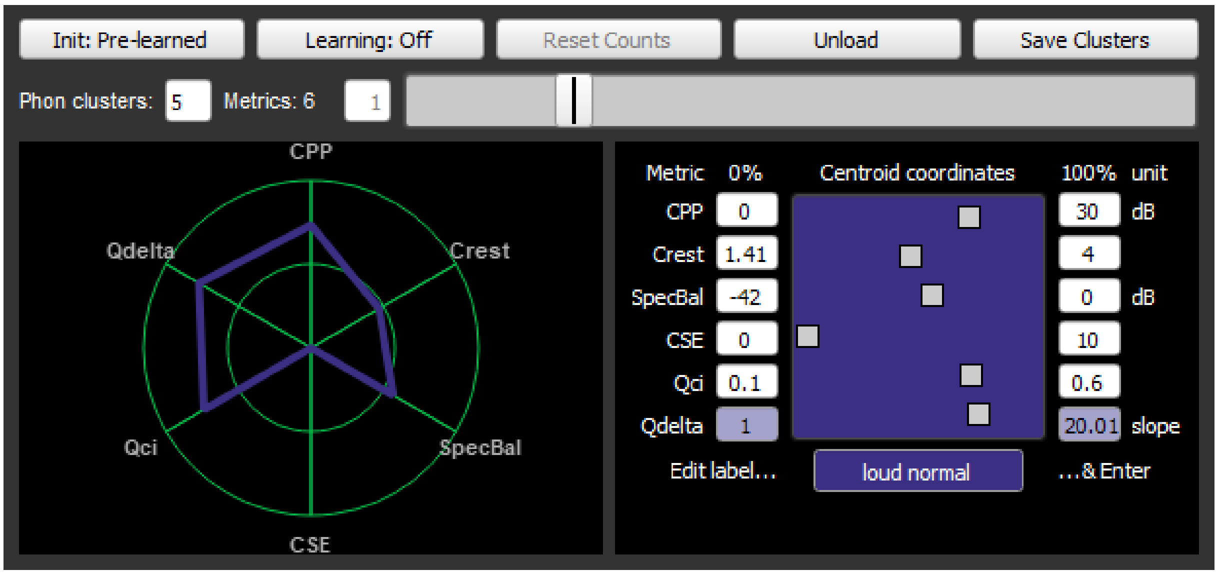


Figure 15. The phonation type clusters panel, in 'centroid editing' mode.

For the clustering to work well, the metric values have to be rescaled so that their numbers vary over a similar range, here 0% to 100%. In the radar plot, the perimeter represents 100%, the smaller circle is at 50%,[3] and the center is 0%, for each metric.

A few metrics (Q_{Δ} and Density) tend to have an exponential distribution, so their averages are transformed by the base-10 logarithm prior to clustering. This is indicated by a light blue background in their number fields (Figure 15). For instance, the cell average of Q_{Δ} varies from 1 to about 20, but is clustered as log(1) to log(20).

When **Learning** is **On**, you can drag the squares to set centroid coordinates anywhere between 0% and 100%. This would be appropriate if you already have some prior knowledge of where the desired centroids should be. Centroids can also be picked by clicking in the voice map (→3.6.6). You can add a text label describing the type of phonation, for each cluster, by typing in the field at the bottom and pressing Enter. We will return to this in section 3.6.7.

The *ranges* (0%, 100%) can be modified only by editing a `*_cPhon.csv` file and then loading it back into FonaDyn. Different ranges may be appropriate for different kinds of voices. `cPhon.csv` files for adult males, adult females and children are [in preparation for a future release], based on findings from [32].

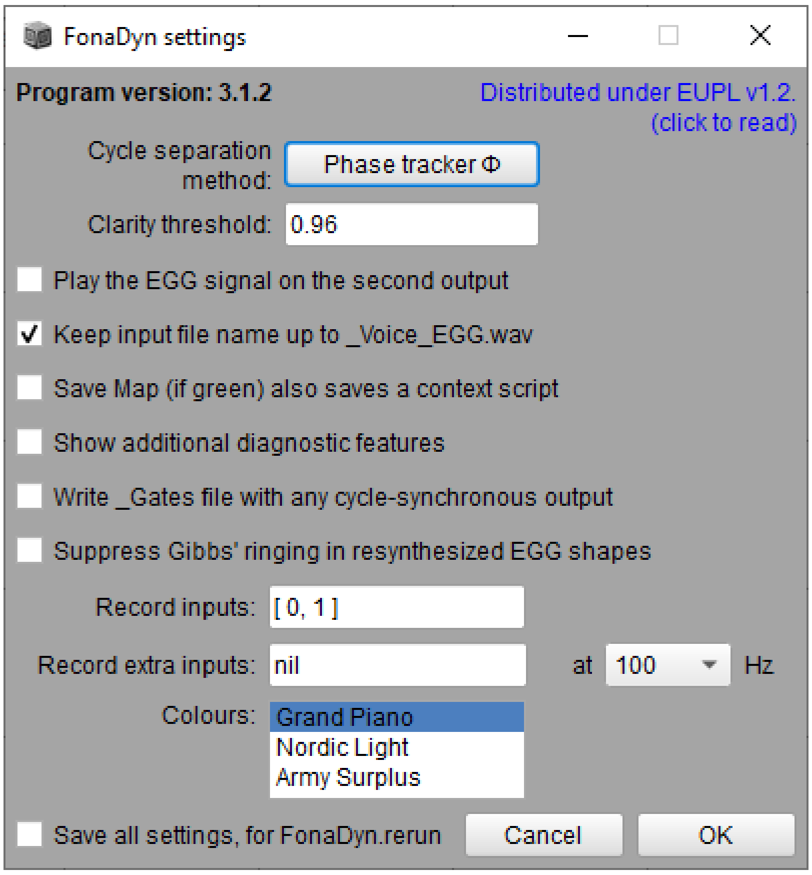
Tip: you can press Alt+F to hide or show the Phonation types clusters panel.

Tip: you can open a phonation centroids file whose name ends in `_cPhon.csv` by dragging the file from its folder and dropping it onto any of the graphs in this panel. You'll have to **Unload** first.

3) The 50% circle does not represent any particular criterion for voice quality.

3.1.10 The Settings

To reduce screen clutter, some diverse settings that are less frequently used have been placed in a separate **Settings…** dialog box.



FonaDyn settings

Program version: 3.1.2 — Distributed under EUPL v1.2. (click to read)

Cycle separation method: Phase tracker Φ

Clarity threshold: 0.96

☐ Play the EGG signal on the second output

☑ Keep input file name up to _Voice_EGG.wav

☐ Save Map (if green) also saves a context script

☐ Show additional diagnostic features

☐ Write _Gates file with any cycle-synchronous output

☐ Suppress Gibbs' ringing in resynthesized EGG shapes

Record inputs: [0, 1]

Record extra inputs: nil at 100 Hz

Colours: Grand Piano / Nordic Light / Army Surplus

☐ Save all settings, for FonaDyn.rerun

Figure 16. The Settings… box.

Click the blue text to open the license text from the Internet.

Cycle separation method selects Phase tracker or Peak follower (→2.4.2).

Clarity threshold allows you to adjust the threshold for cycle regularity (→2.3.3). This refers to the regularity of the *audio* signal, not the EGG signal. The default is 0.96. With soft or breathy voices, or with running speech, you may want to lower this setting. While recording, you can use the Clarity layer in the voice map to see which phonations are rejected from analysis. Once the voice map is saved, that information is no longer kept. Also, the Clarity threshold as such is currently *not* stored in the `_VRP.csv` file. However, the smallest value appearing in the "Clarity" column should be sufficient to know how the threshold was set when a map was made.

Play the EGG signal on the second output. This works both for playback and for listening from maps. The default is to play only the audio signal, on both outputs. Furthermore, when this option is checked, the Signal window (Figure 19) will display both the voice and the EGG tracks.

Keep input file name →3.5.2.

Save Map also saves a context script. In order to rebuild a voice map, the signal file, several settings and the two cluster data files must be the same as when the map was first created. That context is easy to forget between sessions. If this box has been checked, and a sufficient context is defined when an analysis run stops, the **Save Map** button will display its text in green. FonaDyn will then create also a script file with the name of the voice map file, with the additional suffix ".Context.txt". This script documents the necessary settings. When it is run, it will restore the context in which the map was created. This lets you rebuild maps exactly, or rerun after adjusting the settings in some way. If either of the cluster data files has been modified after the map file was saved, a warning is posted, since this might mean that a re-made map will not become the same as the original, after all.

Show additional diagnostic features. This enables a number of additional output file types, allows tweaking of the Moving EGG display, turns on extra logging to the SuperCollider post window, and maybe one or two other things.

Write _Gates file →3.5.6.

Suppress Gibbs' ringing in resynthesized EGG shapes →3.1.7. This option takes effect only upon pressing ▶ START.

Record inputs: This is a list of the hardware inputs used for recording. The first number is the input for the audio signal (the microphone), and the second number is the input for the EGG signal. Normally these are 0 and 1, respectively, but you can specify other inputs if that is more convenient. Both brackets [ ] must be present, and the numbers must be separated by commas. Also, you can record *additional* audio-rate input signals, simply by listing more hardware inputs before the end bracket. The inputs do not need to be contiguous, nor in any particular sequence. All the signals will be recorded in parallel tracks in the `_Voice_EGG.wav` file, at the same sampling rate (44100 Hz). When opening such a file for analysis, FonaDyn will read the audio signal from the first track, and the EGG signal from the second track. Any remaining tracks will be ignored.

Record extra inputs: This is similar to the previous item, except that an `_Extra.wav` output file will be created, synchronized with the `_Voice_EGG.wav` file, and containing the specified signals sampled at the rate chosen in the **Hz** box (→3.2.8). This is intended for acquiring slow physiological signals in parallel with the audio and the EGG. It is also possible to sample the `_Extra` file at full speed (44100 Hz), but that will consume a lot of disk space.

Colours: selects one of the available colour schemes. Since a black background often works poorly in print, you may prefer the 'Nordic Light' colour scheme for screen-dump illustrations that will be printed.

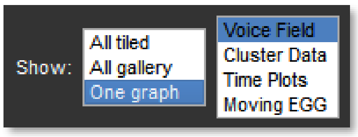
Save all settings. If checked, *all* settings, not just the ones in this dialog box, are saved when FonaDyn is closed. To resume with those saved settings in the next session, start it using `FonaDyn.rerun` instead of `FonaDyn.run` (→3.5.7).

The check box **Save all settings** works only once: it is *not* restored on `FonaDyn.rerun`, since you would then risk overwriting a good setup by mistake, when FonaDyn is closed the next time.

A more flexible, if more complex way of setting up and running FonaDyn automatically is to use a command script (→3.5.8).

3.1.11 Comparing Voice Maps

At the current state of the art, the greatest utility of voice mapping in general is probably in comparing voice maps to each other; notably, before and after an intervention with a voice patient or student. FonaDyn can help you to do so by displaying several maps simultaneously, called BEFORE, NOW and DIFF; and optionally TWIN or SMOOTH as well. When displaying more than one map, FonaDyn adds a row of up to six extra buttons that control how you interact with these maps.



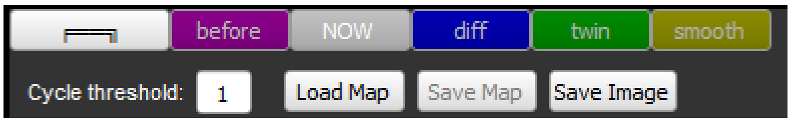


Figure 17. Controls for laying out multiple voice maps.

Multiple voice maps can be displayed as tiled next to each other, or stacked on top of each other such that only the selected one is visible. By selecting BEFORE, NOW, TWIN, SMOOTH or DIFF as the 'top of the stack', you can readily compare the maps visually to each other. The button at the top left toggles between the tiled view (▯▯▯) and the stacked view (⊓).[4]

In the stacked view, each coloured button brings the chosen map to the top. When you select a *layer* in that map, that selection is propagated to the other maps as well, except TWIN. In the tiled view, all maps are shown. Layer changes are then propagated only from the map chosen with the buttons at the top.

To save screen space with multiple maps, you may wish to choose **Show: One graph** at the very top left, as shown in Figure 17. This hides all the other graphs and so frees up space in the FonaDyn window. The **Show: All Gallery** option instead arranges the other graphs along the bottom.

NOW: The current voice map, into which FonaDyn records new data, is called the NOW map. It is the same as the single map you normally use. When comparing maps, the NOW map is assumed to represent the *post*-intervention status.

BEFORE: If you press Alt-B, a new voice map appears that initially is a copy of the current NOW map, complete with all its layers. Or, you can load a different `_VRP.csv`

4) See also the option `tileMapsVertically` (→1.4).

file into the BEFORE map. Unlike NOW, BEFORE remains unchanged when FonaDyn is running. It acts as a reference, typically for showing a *pre*-intervention status. You can now record or load post-intervention data into the NOW map. **Note**: the BEFORE map has no **Save Map** button, so you might want to save the current NOW map first. Press Alt-B again to close the view of the BEFORE map.

DIFF: Once you have two finished voice maps on display, you can press Alt-D to open a third map, DIFF, in which the cell-by-cell differences between the first two maps (usually NOW and BEFORE) are displayed.[5] The direction of comparison is map 2 minus map 1. Green colour signifies an increase in the second map, red a decrease. For each map layer, there is a corresponding comparison layer in the DIFF map. The DIFF map is computed only when it is created; it is not updated in real time. Press Alt-D again to close the DIFF map display.

When a DIFF map is present, the BEFORE and NOW maps also show a background underlay of each other, to aid the eye in interpreting the contour differences. In this situation, choosing Save Map for the NOW map does not save the underlaid map; but you can choose Save Image instead, if you need to remember what it looked like.

It is not obvious how best to visualize differences between maps of clusters. Currently, FonaDyn does this by colouring the cells whose cluster number has *changed* with the colour of the new cluster. Cells that now belong to the same cluster as before are shown as white. The sub-layers that show one cluster at a time show the *changes* in the relative percentages that appear in the BEFORE and NOW maps. Note that cluster comparisons are meaningful only if the BEFORE and NOW maps were both built using the same cluster centroids.

If you ask for a difference map of two maps whose cluster counts are different, FonaDyn will issue a warning and skip the corresponding layers.

When saved, difference maps are named `_D_VRP.csv`, unless you specify a different ending. They have the same format as `_VRP.csv` files, but the numbers therein are mostly differences (or ratios, for Q_{Δ}), as indicated in the colour bar for each map layer. It is pointless to re-Load a difference map; it is intended only for export to other software.

TWIN: Sometimes you may wish to observe how two voice metrics relate to each other, even as the map is being acquired. Press Alt-T to display a TWIN map that is updated in the same way as the NOW map, but can show some other metric (layer). The TWIN display cannot be loaded or saved, because it is not a separate map. It is just an extra window on the NOW map that can show a different layer simultaneously. Therefore, in the TWIN map, the layer selection is always independent of that in the other maps. Press Alt-T again to close the TWIN map.

SMOOTH: To fill in small gaps and to smooth out moderate graininess, press Alt-S. After a second or two, this opens a new map that is an interpolated and smoothed version of the NOW map, with all its layers. Display them side by side to see the effect of this operation. The SMOOTH map is computed only when it is created. It is not updated in real time. When you have saved it, press Alt-S again to close the SMOOTH

5) For the Q_{Δ} metric, the post/pre *ratio* is displayed instead, because it is more appropriate.

map. The smoothing algorithm is described in section 2.10. Smoothed voice maps are saved with `_S_VRP.csv` at the end of the file name, unless you specify a different ending.

If you try to smooth a map whose cluster counts do not match those in the cluster control panels, FonaDyn will issue a warning and refuse to smooth. This can happen if you have loaded a map without reloading the matching cluster files.

A DIFF map will have an empty cell wherever there is an empty cell in either the BEFORE or the NOW map. You can get more useful difference maps, with fewer holes, by first smoothing, saving and then reloading both the maps that you want to compare.

Once you have made a SMOOTH map, you can make also a DIFF map that shows the small cell-by-cell differences between the SMOOTH map and its original.

3.1.12 Controlling the window layout

Unlike many other programs, FonaDyn keeps its subwindows 'docked' within the main window. Usually, this is quicker and more convenient than having several separate floating windows which would need to be arranged on the screen. Instead, you have several options for controlling the visibility and layout of the subwindows within the main window.

The list box **Show**: at the top left contains layout shortcuts, with the following options.

Table 4. Layout options

Tiled	The default layout, with the voice map(s) at the bottom right. Individual graphs can be hidden or shown with Alt keys, see above.
Gallery	The preferred map comparison layout (→3.1.11). Individual graphs can be hidden or shown with Alt keys, see above.
One graph	Show only one graph, as selected from a second list that appears.
All Tiled	Show all graphs, in the Tiled layout.
All Gallery	Show all graphs, in the Gallery layout.
[Enter key]	Unhide all graphs.

Displaying only one or two panels at a time is especially useful for presentations, when you do not want to distract your audience with too many displays. Right-clicking on the borders of the panels will bring up a popup menu for controlling the the visibility of the various graphs; or, you can use the keyboard shortcuts listed on the next page.

Table 5. Keyboard shortcuts for controlling the visibility of graphs

Key	**Main Window**
Alt-C	Toggles the display of the EGG **c**luster graphs
Alt-F	Toggles the display of the phonation type cluster graphs
Alt-H	Toggles the display of the General and Output rows at the top (saving some screen space)
Alt-L	Toggles the display of the audio waveform for **l**istening (when enabled)
Alt-M	Toggles the display of the **m**oving EGG
Alt-P	Toggles the display of the time-curve **p**lots
Alt-V	Toggles the display of **v**oice maps (single or multiple)
	Multiple voice maps
Alt-T	Toggles the display of a Twin map that shows the same real-time data as the Now map, but in some other layer (another metric).
Alt-B	Toggles the display of a Before map that does not change, but is convenient for reference. It is a static copy of the Now map at the time when Alt-B is pressed.
Alt-D	Toggles the display of a static Diff map that shows a snapshot of the cell-by-cell **d**ifferences between Now and Before. The Diff map does not change when Now changes. Make a Before map first, or Diff will be empty. To update a Diff map, close it (maybe save it first) and open a new one.
Alt-S	Toggles the display of a static map that is a **s**moothed and interpolated version of the first map. A smoothed map does not change when the original changes. To smooth another map, maybe save the current Smoothed map first; then close it and make a new one.
Alt-X	In Tiled mode, toggles whether multiple maps are tiled horizontally or vertically.
▭	Toggles between Stacked or Tiled display of multiple voice maps.

Reducing the number of visible graphs can result in the voice map becoming very stretched out. To prevent that, you can have FonaDyn fix its aspect ratio to the standard 2:1. Evaluate the following statement (for the current session only), or insert it into the startup file (for every session): `FonaDyn.config(fixedAspectRatio: true);`

3.2 Recording

3.2.1 Overview

FonaDyn can be used for recording a minimum of 2 channels: the acoustic and EGG signals. You can record more channels, if you want to (→3.2.8). There is a built in 'wizard' for sound level calibration. All analyses are performed concurrently with recording, giving a real-time 'preview' of analysis results. The default signal format is 24-bit, 2 channels, 44100 Hz. Low-rate files are 16-bit, 100 Hz per channel by default. Only the WAV file format can be written, but FonaDyn can *read* most audio file formats.

For sample-accurate synchronization, SuperCollider supports only one multi-channel digital audio interface at a time. All signals have to arrive through the same interface.[6] For instance, you cannot simultaneously record voice via a sound card and EGG via a separate USB-based EGG device. Instead, use the analog output of the EGG device, and route it through the sound card. Most external sound cards have several input channels, so this is rarely a problem.

For stable performance, record to a local hard disk, which you specify using the field **Output directory** at the top. Then back up the new recordings to a file server or removable disk, after each session.

3.2.2 Recording environment

Use a quiet room. FonaDyn uses the 'clarity' metric as a criterion for acceptance: the EGG analysis will proceed whenever the *audio* signal from the microphone is *sufficiently periodic*, even if it is very soft. The rationale for this design is that the signal-to-noise ratio of the audio channel is typically much higher than that of an EGG device; and while a weak EGG signal is often electrically rather noisy, its lower harmonics can still be analyzed. It follows that not only the subject's voice but also other periodic sounds in the recording room may open the EGG analysis gate, including other voices, a piano, or machinery such as air conditioning, traffic outside, etc.

The recording environment should follow the established recommendations [16] for VRP acquisition, with regard to ambient noise and room absorption. If a whirring computer or any other source of tonal or soft fan noise must be present in the recording room, then the microphone should be of a cardioid type and be pointed exactly away from the noise source. In highly absorbent rooms this can be rather effective, for reducing noise from one direction only. If the subject needs prompting pitches, auditory stimuli, or a background audio track, these should be presented over closed circumaural headphones, or over earbuds.

Headphones can also be useful for restoring some hearing-of-self in anechoic or very dampened recording rooms. Some audio interfaces even have a built-in reverb effects unit, which can help, if applied in moderation. Just be sure that the reverb signal is not routed into the signal that is being recorded.

[6]) Multiple audio interfaces can be used, if they are physically synchronized to each other with a 'word clock' signal. This can be a dedicated coaxial sync connection with BNC connectors, optical ADAT/TosLink, or electrical SP-DIF or AES/EBU.

3.2.3 Normal setup

By default, the two output channels of your audio interface play a copy of what the microphone is picking up. In this way, you can monitor the incoming audio signal on headphones, and check that it is free from noise and hum. You may wish to connect a loudspeaker instead, for listening to earlier recordings together with other people.

The button **Playback/Echo** turns the audio output On or Off. If you have a loudspeaker connected, in the same room as the subject, it is advisable to prevent feedback, by choosing Off just before pressing ▸START to record. Or, monitor with headphones.

Processing and display are started when you press ▸START, but nothing is saved to disk, by default. This is useful for checking levels, and for working interactively with real-time feedback in a voice studio, for instance.

3.2.4 Check the EGG gain

Connect the EGG device and strap the electrodes onto the subject's neck. Turn on the EGG device and ask the subject to phonate. For the white 'Moving EGG' panel, choose **Normalize: Off**. Press ▸START, and wait for that button to display ■STOP. Adjust the output level of the EGG device such that you get a clearly visible waveform when the subject is phonating. Try also adjusting the electrode position vertically on the participant's neck, for maximum signal. Ideally, the displayed curve should exercise the upper half of the vertical axis, for the strongest signals.

FonaDyn monitors the EGG signal amplitude during recording. If the usable range is exceeded, the words **EGG CLIPPING** are flashed for about a second. This typically happens not so much for strong phonation, but more often if the larynx moves a lot. It is usually the near-DC drifting that pushes the EGG signal out-of-range of the A/D converter. If so, reduce the output level of your EGG device. If necessary, insert a signal attenuator between the EGG device and the input to your audio interface.

Once the EGG amplitude appears to be satisfactory, press ■STOP, and wait until that button again displays ▸START. Restore **Normalize:** to **On**.

3.2.5 Calibrate for SPL

At this stage in recording, it is time to calibrate for sound pressure level. This is such an important topic that it is presented in an entire section of its own (→3.3), rather than expanding upon it here. If you are eager just to test recording, uncalibrated, read on here. But don't skip the next section, when you come to it.

3.2.6 Recording live signals

- In the top panel of FonaDyn, check that the **Output Directory** is the one where you want new recordings to be stored. If it is not, use the adjacent **Browse...** button to select another directory.
- In the **Source** list, select **Live signals**.
- Select **Record: Ready**. An empty text field appears, where the name of the new file will be shown. You cannot choose the file name yourself, even though this field

is editable. FonaDyn creates a file name based on the current date and time (→3.3.8). The timestamp makes the filename unique.

- Press ▶ START. After a moment, the Record button becomes bright red to indicate that **Recording** is in progress.
- Have the subject perform the phonatory production procedure.
- Press ■ STOP. After a moment (possibly quite a few seconds), the Record button returns to the dull red **Ready** state**.**
- FonaDyn has now created the file and given it a name based on the current date and time. You can copy the resulting file name of the `_Voice_EGG.wav` file from the panel, and paste it into your log of the experiment, with a comment on what was recorded. You can of course also rename the file afterwards, if you wish; but let the new name end in `_Voice_EGG.wav`, if possible, because this simplifies the post-analysis with FonaDyn.
- When done, select **Record: Off**. It is easy to forget it in the **Ready** state, in which case new files will consume your disk space whenever you press ▶ START.

3.2.7 Re-recording during analysis

Even when you select a file for analysis rather than the live inputs, it is possible to re-record the input file. This would seem rather pointless, but for two things. First, the re-recorded file can receive a new time-stamped file name, with a time stamp reflecting the current time (→3.1.10). This may make it easier to track different analyses of the same file. Second, the re-recorded file contains not the raw input EGG and voice signals, but rather the pre-conditioned signals, which allows you to inspect and hear the signals as they are after conditioning, if you wish. To remind you that you are re-recording, the Record button turns **orange** rather than **red**.

3.2.8 Recording additional signals in parallel

FonaDyn itself normally records a stereo WAV file only, with voice and EGG in the left and right channels. If you want to record additional signals in synchrony with voice and EGG, FonaDyn can do that, too. To enable the recording of extra channels, first activate the total required number of hardware inputs in the SuperCollider startup file, and restart SuperCollider.

You can record additional *audio-rate* channels into the same file. In FonaDyn, open the **Settings...** dialog and find the field **Record inputs**. Here, the first two numbers (usually `[0,1]`) are reserved for the microphone and EGG signals. Record more channels simply by entering more numbers before the closing bracket; or fetch the mic and EGG signals from other hardware inputs by changing `[0,1]` to something else.

Typically, one may wish to record also *slow physiological signals* such as subglottal pressure, larynx height, or breathing-related signals. Such signals cannot be recorded by audio interfaces, because the latter block DC, and attenuate AC signals below about 20 Hz. However, some audio interfaces offer so-called ADAT optical connections, each of which adds another 8 inputs or outputs. Enthusiasts in the music community for analog synthesizers have developed DC-coupled ADAT-linked converters for slow control voltages [17].

> Be aware that such ‘consumer’ devices are not generally certified for safe use with body-contact transducers. Although malfunction is rare, a wise precaution would be to power them from batteries rather than from the mains.

FonaDyn can acquire such signals, at a much lower sampling rate (default: 100 Hz), thereby saving a lot of disk space. No anti-aliasing is performed, so the signals should be externally band-limited to half the sample rate.7 In FonaDyn, open the **Settings** dialog. In the text field **Record extra inputs**, type a list of the inputs to which you have connected the extra signals. The list must consist of comma-separated integers, and must be enclosed in square brackets, like so: `[10,11,12,13,14]`. The inputs must refer to active hardware inputs. The listed inputs do not have to be contiguous, nor need they be in any particular order. The order that you specify is the order in which the signal tracks will appear in the output file.

The output file will be a multichannel WAV file, with as many interleaved channels as you have specified in the list. The signals will be synchronized with those in the `_Voice_EGG` file. The filename will have the same timestamp as the `_Voice_EGG` file, followed by `_Extra.wav`. The sampling rate is nominally 100 Hz per channel. You can request a different sampling rate in the popup menu. Only rates that are an integer divisor of 44100 are allowed, or the extra channels could suffer from sampling jitter. To turn *off* the recording of these extra channels, clear the text field **Record extra inputs** in the **Settings** dialog box.

If you choose 44100 Hz as the sampling rate for the Extra file, it will follow the sample bit width (16 or 24) that is currently in effect for `_Voice_EGG` files. For all other rates, the Extra file will be written with 16-bit integer data.

Not all soundfile editors can display several multichannel signal files with different sampling rates at the same time. You may want to try *Sopran*, a freeware by Svante Granqvist (www.tolvan.com) that does this, and much else.

3.2.9 Using FonaDyn without an EGG device

If you are interested in mapping only acoustic metrics, or do not have an EGG device at hand, FonaDyn still wants a second periodic signal to work on. The simplest workaround is to fake an ‘EGG’ signal by copying it from the voice input. To do so, put this line into the SC startup file:

```
FonaDyn.config(inputVoice: 0, inputEGG: 0);
```

This means that *both* the voice and the ‘EGG’ will be taken from the first hardware input (0, or substitute another input number, if that is where you have connected the microphone). While all ‘EGG’ results will then be practically random and meaningless, the displays of Density, Clarity, Crest factor, Spectrum Balance and Cepstral Peak Prominence will be correct. You will probably want to hide the Moving EGG panel (press Alt+M), since its display will be irrelevant and distracting.

7) Some digital audio interfaces have an EQ section on each input that might achieve some band-limiting.

Without an EGG signal, you can still perform phonation-type clustering, based on only the acoustic metrics. To do so, edit a _cPhon.csv file (→3.5.6) and include columns only for the acoustic metrics that you want.

Conversely, if you have only an EGG and no microphone, this can be configured analogously, but the SPL axis on the voice map would need to be re-interpreted, since the EGG does not represent sound. If you are curious, you can test what FonaDyn makes of an accelerometer signal or a photoglottographic signal, instead of the EGG. We haven't yet gotten around to that.

3.3 SPL calibration

FonaDyn assumes a fixed calibration of the signal level in the .WAV files it analyzes (→2.3.3), and there is no facility for offsetting of the SPL after recording. This is by design. By calibrating accurately for SPL at the time of recording, you can be confident that all audio signals recorded in FonaDyn have the same SPL calibration. In subsequent analyses, this saves a lot of time and reduces the potential for errors. To facilitate your compliance, this section describes the logic of calibrating, and also how to use the interactive SPL calibration tool that is provided with FonaDyn. This tool can be started with the button **Calibrate**… that appears when you are about to record; or by running `FonaDyn.calibrate` .

3.3.1 Choosing a method of calibration

The choice of calibration method will depend on your choice of microphone placement, and on the equipment at your disposal. On the following pages, we describe four calibration scenarios A…D, in order of decreasing precision. They are also summarized in Table 1. Other criteria may be more important to you. These scenarios are all supported by the walk-through SPL calibration tool provided with FonaDyn.

In scenarios A and B, the front mic can be anywhere during calibration, but during recording it must be placed at a constant 0.3 m in front of the speaker. The participant must remain still. In scenarios C and D, the voice is recorded through a headset mic. In scenario C, the headset mic path gain is carefully matched to that of a calibrated front mic. In scenario D, only a headset mic and an SPL meter at 0.3 m are needed, but the calibration will not be as precise as in A, B or C. Always perform the calibration for the scenario that you will be using for the recordings. Always perform a level calibration for each new subject, and for each new placement of the recording microphone.

Table 6. SPL calibration scenarios.

	Scenario	*Pros (+) and cons (–) of this scenario*
A	A front mic, with a matching level calibrator device that is sealed to the microphone capsule, and produces 94 or 114 dB.	+ simple, stable, accurate calibration + sealed attachment eliminates ambient noise during calibration + a dB meter is not needed - voice must be recorded through the front mic, keeping a constant distance - fairly high cost of calibrator device
B	A front mic that is recorded with a tone generator playing over a near loudspeaker and a dB meter held close to the microphone	+ almost as good as A + a calibrator device is not needed (but the dB meter's calibration must be valid) - more sensitive to ambient noise - the voice must be recorded through the front mic, keeping a constant distance
C	A headset mic, the gain of which is adjusted to give the same *voice* signal strength for a sustained /a/ as that arriving through a front mic whose gain has already been calibrated as in A or B. The front mic must be at 0.3 m when adjusting the headset mic gain.	+ the exact mouth-to-mic distance of the headset need not be known + the distance is held constant, so the participant is free to move the head when recording - requires an extra microphone, and two mic inputs during calibration
D	A headset mic, the gain of which is adjusted to give the same on-screen voice sound level for a sustained /a/ as that shown by a dB meter at 0.3 m in front of the speaker.	+ low cost - with a sustained vowel, it is difficult to get a stable reading and to match accurately the on-screen SPL with the SPL on the dB meter. Not recommended.

In all scenarios, you will need to control the gain of the microphone signal – that is the objective of the calibration. Depending on your audio interface, this adjustment might be done with a physical knob that is usually located next to the microphone connector, or an on-screen 'knob' in the interface's control software, or both. Make sure that you understand and document *all* the points in your signal chain at which the microphone signal gain can be modified. Choose one of them as the primary gain control, and document and fix the settings for the others.

To run any of the scenarios, choose **Source: Live signals** and press the **Calibrate...** button. A window opens (Figure 18) with an oscilloscope, a spectrum analyzer, several buttons, and a terse version of the instructions that follow here.

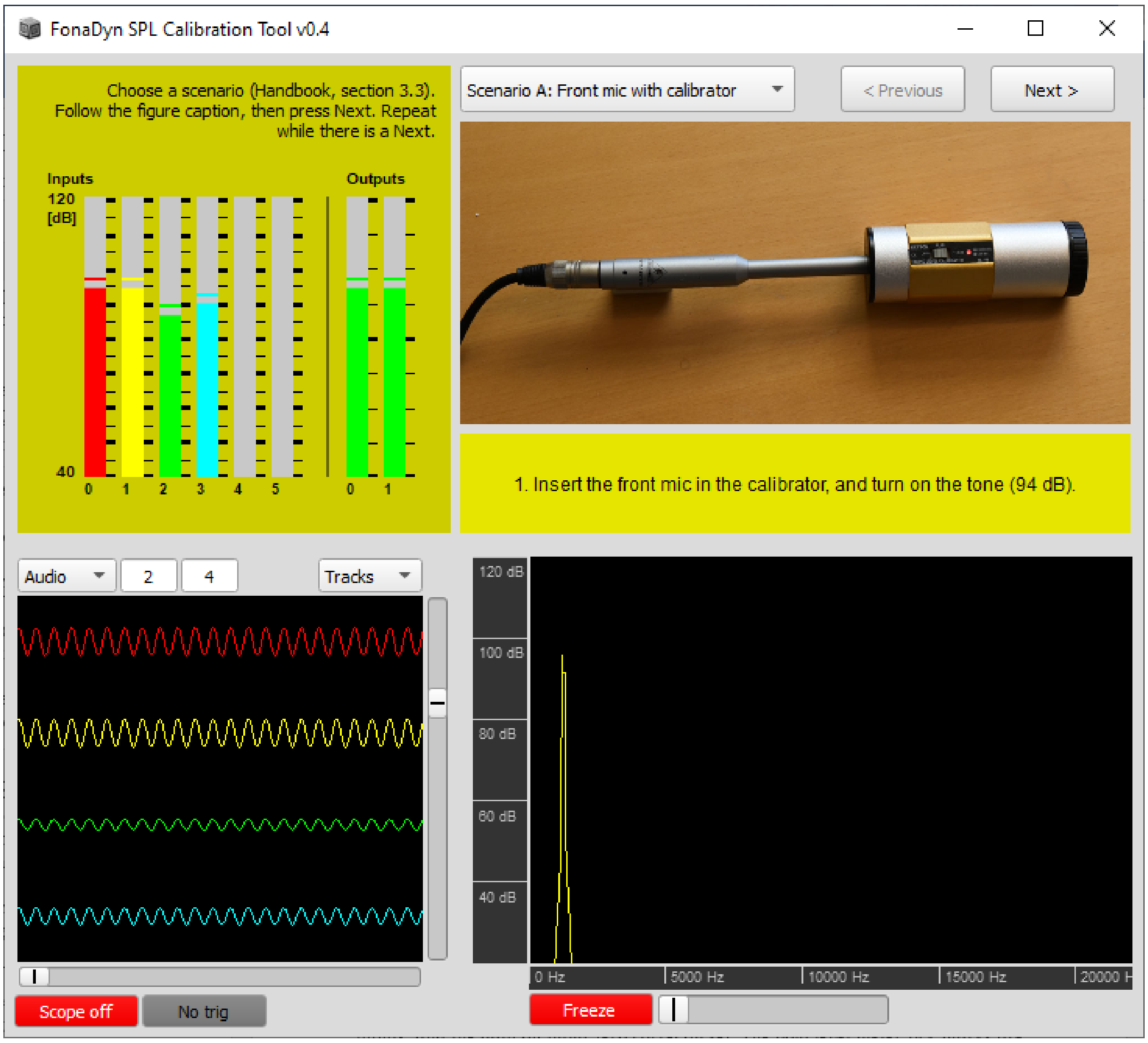

FonaDyn SPL Calibration Tool v0.4
Choose a scenario (Handbook, section 3.3). Follow the figure caption, then press Next. Repeat while there is a Next.
Scenario A: Front mic with calibrator
< Previous
Next >
Inputs
120 [dB]
40
0 1 2 3 4 5
Outputs
0 1
1. Insert the front mic in the calibrator, and turn on the tone (94 dB).
Audio
2
4
Tracks
120 dB
100 dB
80 dB
60 dB
40 dB
0 Hz
5000 Hz
10000 Hz
15000 Hz
20000 H
Scope off
No trig
Freeze

Figure 18. The calibration tool, scenario A, with the input signal copied to four of six inputs, and the gain on input zero correctly set. The bold level meter tick marks are at multiples of 10 dB. Note the sine wave coming from the calibrator device at 26 dB down = 94 dB RMS, 97 dB peak (red bar and signal).

At the top left are shown input and output level meters for all active inputs and outputs, as specified in the SuperCollider startup file. The top right panel is the 'wizard' that displays texts and photos to guide you through the chosen Scenario.

The bottom left panel is an oscilloscope that displays signals from the SuperCollider server. The two numbers control which bus signals that are displayed. The first number is the bus of the topmost signal shown in the oscilloscope, and the second number is the number of bus signals shown. Outputs start at zero, followed by inputs. In this example, there are two outputs (buses 0 and 1) and six inputs (buses 2...7).

The bottom right panel is a spectrum analyser that shows the power spectrum of the signal on the first displayed bus.

3.3.2 A: Calibrator device that fits the microphone

This scenario is applicable if you are recording voice with a fixed microphone distance of 0.3 m *and* you have a calibrator device that fits the microphone. There are SPL calibrator tone generator devices that attach hermetically to laboratory microphones. Such a device produces a 1 kHz sine wave at 114 dB (10 Pa RMS pressure) or 94 dB (1 Pa RMS pressure), when properly attached to the microphone.

1. From the drop-down list, select **Scenario A**.
2. Connect the front microphone to the first microphone input on your audio interface.
3. Set the calibrator to 94 dB, turn it on, and listen for the tone (as a battery check). Then turn it off again.
4. Attach the calibrator to the lab microphone, as exemplified in the on-screen picture.
5. Press **Next**. A level meter appears on the screen, as a green vertical bar.
6. Turn on the calibrator. The level meter should quickly rise to a constant value.
7. On your audio interface, carefully adjust the microphone signal gain so that the green level meter displays 94 dB.
8. Document the resulting gain setting(s) and then leave them untouched.
9. Stand the microphone at 30 cm in front of the participant's mouth.

You have now calibrated the gain for recording voices using that microphone at a constant 30 cm distance. With a little practice, this procedure takes less than one minute.

3.3.3 B: Front microphone and SPL meter

In this scenario, you use a SPL (dB) meter, a small powered loudspeaker, and a calibration noise that is provided by the computer.

1. From the drop-down list, select **Scenario B**.
2. Connect the second output channel of your audio interface to the powered loudspeaker, and then turn the loudspeaker on.
3. Switch the dB meter to ‘Slow’ and to ‘A-weighting’.

    Choosing ‘A’-weighting at this point reduces the influence of background noise during the calibration. At the frequency peak of the calibration noise, there is no difference between the A-weighting scale and a flat frequency response. The levels shown in FonaDyn are not A-weighted.

4. Place the microphone 0.3 m in front of where the subject’s mouth will be, and place the dB meter right next to it, as shown in the picture. The tip of the dB meter should be close to the microphone (within 3 centimeters). The dB meter does not need to point in the same direction as the microphone – dB meters are omnidirectional. Rather, you must be able to see the dB meter display and the computer screen at the same time. A stand for the dB meter will help.
5. Place the loudspeaker at least 0.5 m from, and facing the microphone+dB meter combo. The exact distance is not critical. If the microphone has an omnidirectional pickup pattern, the loudspeaker can be in any direction. If the microphone has a cardioid pickup pattern, the loudspeaker should be somewhere in front of the microphone.
6. Press **Next**. An on-screen Volume control appears, and a noise (band-passed around 1 kHz) starts playing on the loudspeaker.[8]
7. Adjust the volume of the noise to be moderately loud, but not so loud that it sounds distorted or discomforting to the ear. The actual dB level of the noise is not important.
8. Press **Next**. An on-screen level meter appears that shows the level of the incoming microphone signal.
9. Now be quiet, so that only the calibration noise reaches the microphone. On your audio interface, or on its software control panel, adjust the input gain for the mic signal so that the on-screen level meter shows the *same* value as the value displayed on the external dB meter.
10. Document the resulting gain setting(s).

You have now calibrated the gain for recording voices using that microphone at a constant 30 cm distance. The loudspeaker and the dB meter can be removed from the scene. With a little practice, this procedure takes less than three minutes.

[8]) In any room with reverberation, a stationary sine wave would create an invisible standing wave pattern, with narrow nodes (silent spots) in positions that are impossible to predict. Even 2 cm between the dB meter and the microphone can make a difference. Using a narrow-band noise instead of a sine tone minimizes the influence of such nodes, but adds a little instability to the sound level. To stabilize the reading, the on-screen level meter is very slow.

3.3.4 C: Headset microphone with a front reference microphone

A headset microphone keeps the microphone distance constant, allowing the participant to move during recording, and it improves the ratio of voice to ambient noise. However, the actual microphone distance is not the standard 0.30 m, and can be tricky to measure accurately. In this scenario, you wish to use a headset microphone, but with a level calibrated to that of a reference microphone at 0.30 m. The reference microphone will need to use an spare microphone input on the same audio interface.

1. Connect the headset microphone to the first microphone input on your audio interface, and the front reference microphone to the spare microphone input. Place the headset mic on the participant.
2. Run the command `FonaDyn.calibrate`.
3. Calibrate the gain of the front reference microphone as for scenario (A) or (B), but on the spare microphone input instead of on the first one.
4. From the drop-down list, select **Scenario C**. Two on-screen level meters appear, one for each microphone. Also, a third meter showing the level *difference* appears.
5. Set the two **Input** fields so that the headset mic signal (first input) is on meter one, and the front reference mic signal (spare input) is on meter two.
6. Have the participant vocalize on a stable, sustained /a/ vowel, keeping the participant's mouth at 0.30 m in front of the reference microphone. The participant may *not* move, but the particular voice level or pitch are not important.
7. While the participant is vocalizing, adjust the gain of the headset microphone so that the on-screen level difference becomes as small as possible. The yellow bar turns green when the level difference is smaller than 0.5 dB. This balancing operation gives greater accuracy than can be had in Scenario D.
8. Document the resulting gain setting(s).

You have now calibrated the gain for recording voice using the headset microphone as worn by this participant on this occasion. The participant is free to move the head during recording. The recording will be at a level corresponding to that of a front microphone fixed at 30 cm. With a little practice, this procedure (including step 2) takes less than five minutes. The placement of the headset on the participant's head must remain the same during the subsequent recording.

3.3.5 D: Headset microphone and dB meter only

Because head-mounted microphones are very close to the mouth, their pick-up is also very sensitive to small variations in the small distance. You will need to make a new calibration every time the headset microphone is adjusted or hung anew on the participant. The participant's own voice is used as the sound source.

1. Run the command `FonaDyn.calibrate`.
2. From the drop-down list, select **Scenario D**.
3. Have the subject wear the microphone boom and adjust the capsule so that it is at least 7 cm from the center of the mouth, and to one side.
4. Switch the dB meter to 'Slow', and to 'C-weighting' or 'Linear'. Stand the dB meter so that its tip is 30 cm in front of the participant's mouth, and orient the meter so that you can read it while you are sitting at the computer.
5. Press **Next**. An on-screen level meter appears.
6. Ask the participant to take a big breath and then sustain an /a/ vowel at a level around 80 dB, as steady as possible. This may require some prior practice.
7. On your audio interface, or in its control panel app, adjust the input gain for the mic signal such that the level displayed on-screen becomes the same as the value displayed on the dB meter (to within the nearest decibel).
8. Document the resulting gain setting(s).

You have now made an approximate calibration of the gain for recording voice using the headset microphone as worn by this participant on this occasion. With a little practice, this procedure takes less than three minutes. The placement of the headset on the participant's head must remain constant during the subsequent recording.

> *Tip*: to read the dB meter more easily while sitting at the computer, you can set up a USB webcam aimed at the dB meter, and use a webcam app such as AMCap to display temporarily an on-screen image of the dB meter, next to the on-screen level meter.

3.3.6 Verifying a level calibration

To verify a level calibration from scenario A, make a recording of the calibration tone while the calibrator is attached and active, and then inspect the tone in an audio wave editor. A 114 dB SPL sine wave should have a peak amplitude of 0.71 relative to the maximum (clipping) amplitude in the audio track, or 0.071 for 94 dB SPL. If this is the case, FonaDyn will display the audio SPL correctly in dB relative to 20 µPa, for that microphone, at 30 cm distance. If the participant maintains the distance of 0.3 m to this microphone, the standards are respected.

You can also compare your recorded calibrator tones with the supplied files `114dB-100--800Hz_Voice_EGG.wav` or `94dB-110--1760Hz_Voice_EGG.wav`, which contain sine waves in the audio channel at the correct amplitude for the indicated sound pressure levels.

3.3.7 Recording very loud voices

Very loud voices, such as operatic sopranos, can easily reach higher SPLs than 117 dB at 0.3 m. To accommodate this, FonaDyn can optionally handle a maximum SPL of 140 dB. You request this "singerMode" in the file `startup.scd`, as follows:

```
FonaDyn.config(singerMode: true);
```

> Not many microphones can handle a loud operatic soprano at close range; they report instead a signal which is weaker than the actual sound. You can check this by simultaneously using a second microphone at a greater distance, in a quiet room. If the level difference in decibels between them remains constant with changing vocal output, then this is not a problem.

When the "singer mode" setting is in effect, several things change, as follows:

1. On voice maps, the SPL scale extends to 140 dB, and the horizontal line at 120 dB SPL is plotted as fatter.
2. The voice signal amplitude as shown and heard in soundfile editors will be 10 times smaller, to give the extra 20 dB of headroom.
3. Recordings are saved to 24-bit files (`*_Voice_EGG.wav`), *even if* 16-bit is the current default sample format for recording. Otherwise, the quantization error noise floor would be too high for normal voice.
4. The full-scale amplitude in the audio track of `*_Voice_EGG.wav` files is assumed to represent ±200 Pa (137 dB SPL for a sine wave), rather than ±20 Pa.
5. The `FonaDyn.calibrate` tool automatically adjusts to the larger range.
6. When using the test files mentioned in the previous section, you will need to add 20 dB to the value given in the file name.
7. Opening a 16-bit file will disable the "singer mode". You will have to reboot the SC interpreter to restore it.

When the "singer mode" is *not* in effect (and by default it is not), then the following applies:

8. FonaDyn will refuse to open voice map files (`*_VRP.csv`) containing SPLs higher than 120 dB.

You must implement some scheme for knowing if a file was recorded in "singer mode" or not. That information is not saved into the .WAV file.

3.3.8 Examining live signals

The SuperCollider IDE (Integrated Development Environment) has several 'widgets' that can be helpful when connecting and testing your setup. On the **Server** menu, you will find the following items.

- **Show Server Meter**. This opens a live VU bar meter showing the live signal activity on all active hardware inputs and outputs. If the number of inputs is not what you expected, modify the SC startup file (on the File menu) and reboot the interpreter. All inputs must be on the same hardware device, or on multiple devices that are synchronized in hardware.
- **Show Scope**. This opens a simple oscilloscope on the outputs and inputs. The first number is the first output, the second number is the number of outputs (and inputs) shown. The numbering is zero-based. More information on the oscilloscope is available in the SuperCollider documentation. For instance, the shortcut key '`i`' sets the scope to display all available inputs. This oscilloscope does not have a triggering function.
- **Show FreqScope**. This opens a single-channel spectrum analyzer. You will need to select the desired audio bus for analysis. More information on the spectrum analyzer is available in the SuperCollider documentation.

Customized variants of these three widgets are combined in the accompanying SPL calibration tool (→3.3). To run that tool in a window of its own, execute `FonaDyn.calibrate`, typically when FonaDyn is closed. On a fast computer, you can run FonaDyn first and then the SPL calibration tool *at the same time*. This lets you see your system's input and output signals 'live', which is very useful for checking signal quality.

Internally, signals in SuperCollider run on so-called 'buses', rather like in a conventional mixing desk. The first number box at the top left of the oscilloscope selects the 'bus' number of the first signal. The second number selects the number of consecutive bus signals shown. The first buses, starting at zero, are the *output* buses. Their traces are coloured white. The following buses are the *input* buses, which, like the level meters, are coloured in a rainbow scheme that depends on how many inputs there are. The remaining buses, shown in gray, carry signals that are internal to whichever program the SCSynth server is running at the moment.

The spectrum analyzer shows the power spectrum of the topmost signal in the oscilloscope. Its dB scale is correct if calibration has been performed successfully. There is a choice of a linear or log frequency axis. The frequency axis can be zoomed using the horizontal slider, but only if the whole window is stretched wide (this is a bug in SC).

3.4 Listening from maps

3.4.1 Background

A voice map contains no temporal information. Each cell in a given layer of the map contains the value of a metric, averaged over an entire recording and colour-mapped; or, the colour of the cluster that is dominant in the cell. Still, it can be interesting to know what the voice *sounded* like when the participant was phonating at a particular place on the voice map. Uniquely, FonaDyn *can* play back from the voice map – if you have first made a matching `_Log` file.

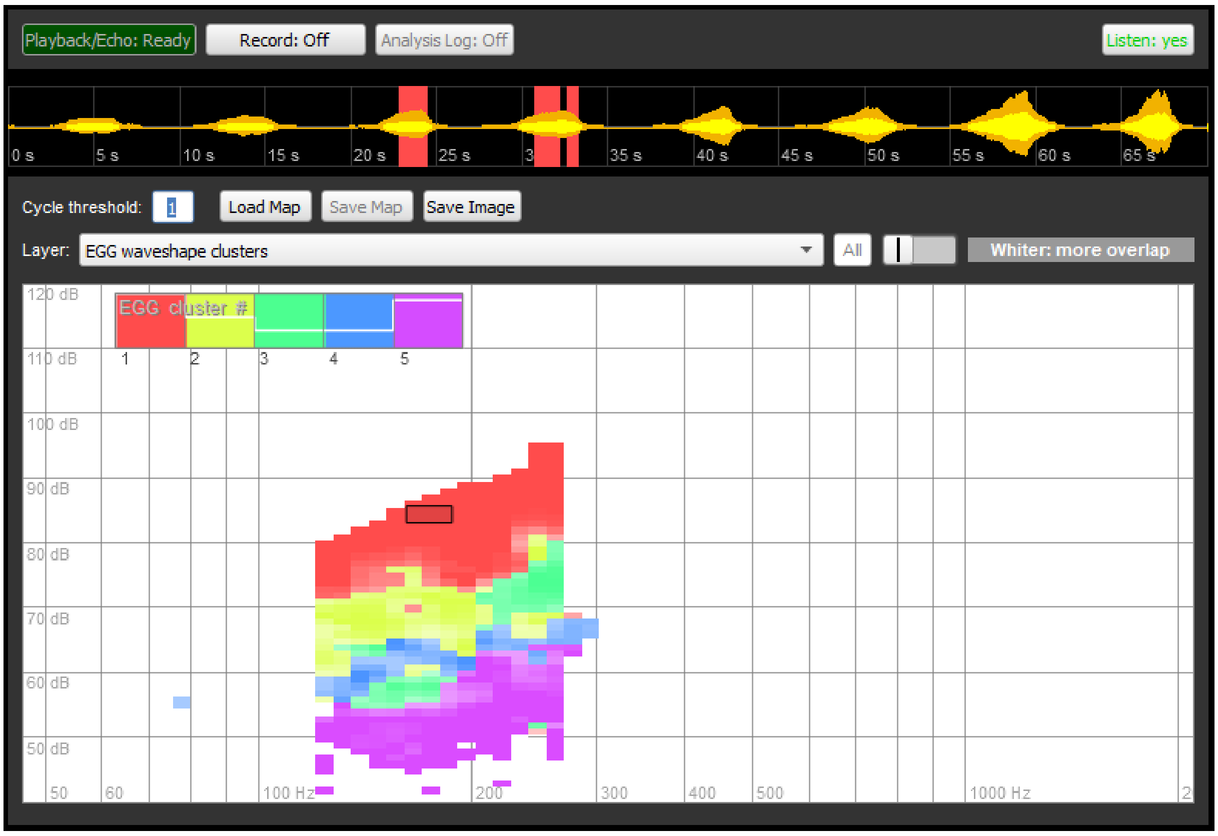


Figure 19. When 'map listening' is enabled, you can shift-left-click in the map to hear the sounds at the corresponding f_o and SPL. The locations in time of those sounds are shown in the audio signal window, and the rectangle cursor shows the parts that are selected.

A FonaDyn Log file contains cycle-by-cycle values of the time, f_o, SPL, all the metrics, the cluster number, and a few other things. With all this information, FonaDyn can use the Log file as an index, to back-track and find sounds at a specified f_o and SPL in the signal file, and even match a given cluster.

3.4.2 Creating the Log file

It is possible to make a Log file when analyzing a signal file, or simultaneously when you are recording live signals. To create a Log file, follow these steps.

1. Set the **Output Directory** to be that of the signal file. If you are recording live signals, this will always be the case – all output files are written to the same directory.
2. If you are working with an existing signal file, choose **Source: From file** and browse to it. Its file name must end in `_Voice_EGG.wav`.
3. If you will not be using the clustering feature, you can now skip to step 7. Just remember to ignore the results of the clustering, which will be performed anyway.
4. If you want to apply clustering for the particular individual, run a cluster analysis of the signal file as described in section 3.5.10, re-order the clusters if so desired (→3.7.6), and choose **Save Clusters**. Save both the EGG clusters and the phonation type clusters.
5. Choose **Load Clusters** and select the appropriate `_cEGG.csv` and `_cPhon.csv` files. Their full names do not need to match the signal file's name.
6. Set **Learning: Off**. Do this for both EGG clusters and phonation type clusters.
7. Push the button **Analysis Log** so that it displays **Log @ cycles**.
8. In the **Settings...**, *check* the option **Keep input file name up to _Voice_EGG.wav**, then choose **OK** (it is the default setting).
9. Press START and let the analysis run to completion. If you stop early, the Log file will be too short. The Log file is created and saved automatically, with a name matching that of the signal file. When the analysis has finished, the audio signal will be displayed.
10. Choose **Save Map** to save also the new voice map. The voice map name does not need to match (but it may help to pair it with the names of the cluster files).

3.4.3 Enabling listening from maps

To enable listening from maps, click the dark gray **Listen: off** button at the right side of the main window (this can be done only with the mouse). The button becomes a lighter gray. With map-playing thus enabled, FonaDyn performs a number of checks whenever you choose a new signal file for analysis, as follows.

- The name of the chosen signal file must end in `_Voice_EGG.wav.`
- There must exist in the same directory a file with the same name but ending in `_Log.aiff.`
- The _Log file must have a file system 'modified' time that is more recent than that of the signal file. (So if you edit the signal file, you must also make a new Log file.)

If any one of the above is false, the button will display **Listen: no**, and the Post window will show an explanation. If all of the above are true, the button will display **Listen: yes**, and a new graph panel will open (possibly after some delay, with long files), with a waveform display of a copy of the audio channel of the signal file (Figure 19). To disable listening, click the **Listen** button again.

In the signal window, the audio peak amplitudes are shown in orange, and the RMS amplitudes in yellow. For listening convenience, the copied audio has been normalized in amplitude, so as to remove the level overhead that is required for maintaining a calibrated SPL. You can zoom the displayed amplitude of the signal, using the mouse wheel with the cursor over the signal window.

3.4.4 Playing from the voice map

When you left-click with the mouse on the voice map, the corresponding portions of the audio waveform are highlighted. This mode of presentation was inspired by Per Fallgren's ingenious tool for browsing large amounts of audio [21]. If the voice map is showing clusters, then FonaDyn matches not only to f_0 and SPL, but also to the cluster number. If you shift-left-click, the matching portions will also be played. Press the space bar to hear them again. To stop a playback, press the space bar one more time.

3.4.5 What is played?

The Log file information is cycle-by-cycle, but it would make no sense to play back the sound of a single matching cycle. Instead, FonaDyn creates a little **sound clip of at least 200 ms** surrounding the matching cycle, and also merges consecutive matching cycles into the same clip. The 200 ms minimum duration includes 60 ms fade-in and fade-out, so as to avoid clicks; that is, onsets and offsets are gradual. Overlapping clips are merged. Non-overlapping clips are cross-faded on playback, to make the sound as smooth as possible. The clips are marked in the audio waveform display by vertical background bars (Figure 19).

The search for matching sounds takes the click position and brackets it. The default tolerance for a match is ±0.75 dB and ±0.75 semitones. This region is shown on the voice map as a rectangle cursor. The rectangle encloses the part of the voice field that is searched for playable clips. This '**search rectangle**' can be freely positioned; it does not snap to the cell boundaries. Clicking in different places even within the same voice map cell will give slightly different results. This is because the information in the Log file is not rounded to whole dB or semitones; it has higher precision. You can also enlarge or shrink the search rectangle, using Ctrl+mousewheel actions on the map.

A time window that spans at least 200 ms is likely to contain cycles also from adjacent cells, or from several clusters, so the matches might be approximate. The **better the match, the more transparent** the search rectangle will be. When the match is poor, the rectangle darkens. A clear rectangle means that the sounds that you are hearing are very representative of the sounds that created the voice map at that location. A dark rectangle means that other sounds may also be played, before or after the sound that matches.

You can listen to individual found segments one-by-one, by left-clicking on them in the signal window. You can also listen to an arbitrary part of the waveform, by selecting it with the mouse (for a grey background); and then left-clicking on that selection. To see the corresponding cells in the map, right-click on the grey selection.

A diagonal hatch pattern will show the cells that are visited by the selected part of the signal.[9]

You can **zoom the signal window** on the time axis, by doing a shift-right-click vertical drag with the mouse. A zoomed window can be panned horizontally using a right-click horizontal drag.

When displayed on its own, the Signal window has a two-way range slider at the bottom. This slider has some shortcut keys (A, N, X, C, ←, →), as described in the SC documentation for the class RangeSlider.

To see and hear also the EGG signal, open **Settings…** and check the option “Play the EGG signal on the second output”.

If you are displaying multiple voice maps (→3.1.11), listening can be invoked only from the NOW, TWIN or SMOOTH maps. The BEFORE and DIFF maps will typically refer to another signal file than the one shown in the input filename field.

When playing from a smoothed map, some cells might be silent. This can happen if an interpolated cell was empty prior to smoothing; there may not exist any corresponding sound to play.

3.5 Files

FonaDyn can export and import several types of data file that can be used for post-processing and display with software such as Matlab or Microsoft Excel. Many Matlab examples and a `runDemos.m` file are provided in the folder <userSupportDir>`/matlab`, if you copied it when installing FonaDyn. For more information on the signal file formats, see also the online documentation for the class `VRPViewMainMenuOutput`.

3.5.1 Files in general

The **Browse** buttons open a file system dialog for opening or saving files. When FonaDyn is started, the default directory for new recordings is suggested as shown in the top line. On Windows, set your file system browser so as to display the full file names with extensions, or it will be easy to confuse different kinds of file with each other.

Choosing **Load** or **Save** of files will start in the directory you last used, *for that kind of file*. Note also that the Open File dialog usually has at its top a drop-down list of recent locations, which can speed up your navigation of the file system.

You can also specify a *persistent* default directory for recordings, which is useful, because the standard location is rarely the one you want. To do so, add the following line in the SC startup file, with your desired location in double quotes:

```
thisProcess.platform.recordingsDir_("C:/Recordings");
```

In SC code such as this, always use *forward* slashes (/) in pathnames, even on Windows. Once you have restarted SuperCollider, FonaDyn will display this path in the field **Output Directory**. Recordings and log files will be saved in that directory.

9) but only in the NOW and TWIN maps

Cluster data and voice map data are saved as **text files**. Their file names are *not* given an automatic time stamp; rather, you must choose a full name yourself. You will probably want to encode the relevant settings into the file name in your own way. The same applies for image files.

As the **column separator** in CSV files, FonaDyn uses the semicolon; although files with the comma (,) as separator will also be read correctly. If you want FonaDyn to write CSV files with columns separated by commas, you can change the separator character in the source code file `VRPMain.sc`, although such a change is not recommended. It is not possible in FonaDyn to change the **decimal character**, which is always the period (.) .

3.5.2 Signal files

When analyzing, FonaDyn processes either live inputs or an existing recording. Signal files normally have a ".wav" extension, but many other sound-file formats are also supported.[10] The channel count must be at least two, with the voice audio in the first channel and the EGG in the second. The sampling rate per channel must be 44.1 kHz.

The sample format in signal files can be 16- or 24-bit integer, or 32-bit float; or a compressed format such as MP3. For all formats, FonaDyn assumes that the amplitude is calibrated, which you need to do when recording (→3.3). See Table 1 in section 2.3.3 for details of the scaling.

At 24 bits per sample, which is the default, uncompressed two-channel signal files consume about 15 MB of storage space per minute. At 16 bits per sample, this decreases to about 10 MB per minute. With 32 bits per sample, 20 MB of storage are consumed every minute.

By default, the output files that contain signals are automatically given a file name beginning with a time stamp YYMMDD_HHMMSS_ (shown as * below). This is the time at the *start* of recording. All forms of file output are optional, and all can be written in parallel. When processing from a file to new output files, the output files receive the same base name as the input file. The output files will then inherit the *original* time stamp (or other name that you have given the input file), which makes it easier to know which files are derived from which input signals. This works only for input files whose names end in `_Voice_EGG.wav`.

If instead you prefer the output files to receive a new time stamp from the time that they are created, i.e., the time of *analysis*, go to the **Settings...** box (→3.1.10) and *un*check the option **Keep input file name.** This lets you compare the outputs of several runs made using different analysis settings. However, you will need to keep track of the new time-stamped file names yourself, and of the settings used when creating them.

Recordings are made into 2-channel files at 44.1 kHz per channel and 24 bits resolution, named `*_Voice_EGG.wav`. Other formats are possible (→3.2.8, →3.3.7).

[10]) Although some features of FonaDyn do require the signal file name to end in "_Voice_EGG.wav", you can open other signal file types as well, and make voice maps of them.

Physiological signals can be saved synchronously into multichannel `*_Extra.wav` files, at up to 500 Hz per channel and 16 bits resolution (→3.2.8).

3.5.3 Log files

For how to create Log files, see section 3.4.2. This file type is a multichannel file called `*_Log.aiff`. It contains one frame of data for every EGG cycle. The data are in 32-bit floating-point format. Each frame has these tracks: time (s), f_o (semitones), signal level (dBFS), clarity (0…1), crest factor, spectrum balance (dB), CPP (dB), EGG cluster number, PhonType cluster number, SampEn (a.k.a CSE), I_c, Q_Δ, Q_{ci}, and the levels (Bels) and phases (radians) of all analyzed Fourier components of the EGG signal. See also the online documentation for the class `VRPViewMainMenuOutput`. The contents of Log files can be plotted with Matlab using the supplied m-file `FonaDynPlotLogfile.m`.

In the Log file, the harmonic levels are relative to full scale. The phases are absolute, i.e., relative not to the fundamental but to the cycle detection trigger point. The frame rate (or "sampling rate") in this file is cycle-synchronous by default, with new data for every EGG cycle. Frames of accepted cycles only (above the clarity threshold) are written to this file. That is why the time track is helpful; without it, the absolute times of each cycle would be hard to reconstruct.

It is also possible to make a log file at a fixed rate: one of 50, 100 or 300 Hz. This can simplify things when you intend to plot the log file data alongside data from other analysis programs. However, if you do use a fixed rate, remember that the data in most log-file tracks will be valid only at the time points when the value in the 'Clarity' track is above the clarity threshold value. Data from some phonatory cycles may be missing or repeated. Listening from maps using a fixed-rate log file may give strange results.

If you change any of the following, then the **Log** file must be re-created, if it is to correspond exactly with the voice map.

- the clarity threshold
- the EGG de-noising threshold
- the clustering settings
- the cluster ordering
- the SampEn settings

The data in Log files are stored as 32-bit float values. These values are *not* scaled down to ±1.0, as is the convention for normal audio files. Therefore, the signals in many of the tracks will appear to be out-of-range, if a `*_Log.aiff` file is opened in a multitrack audio editor. Also, if a file of type **_Log.aiff** is opened in other software, its sampling rate will be appear to be 44100 Hz, but this is not correct. The effective 'sampling rate' is the f_o of the analyzed signal. There is no standardized way of storing that meta-information into an .AIFF file. Instead, the first track contains the time in seconds for each frame (corresponding to each phonated cycle).

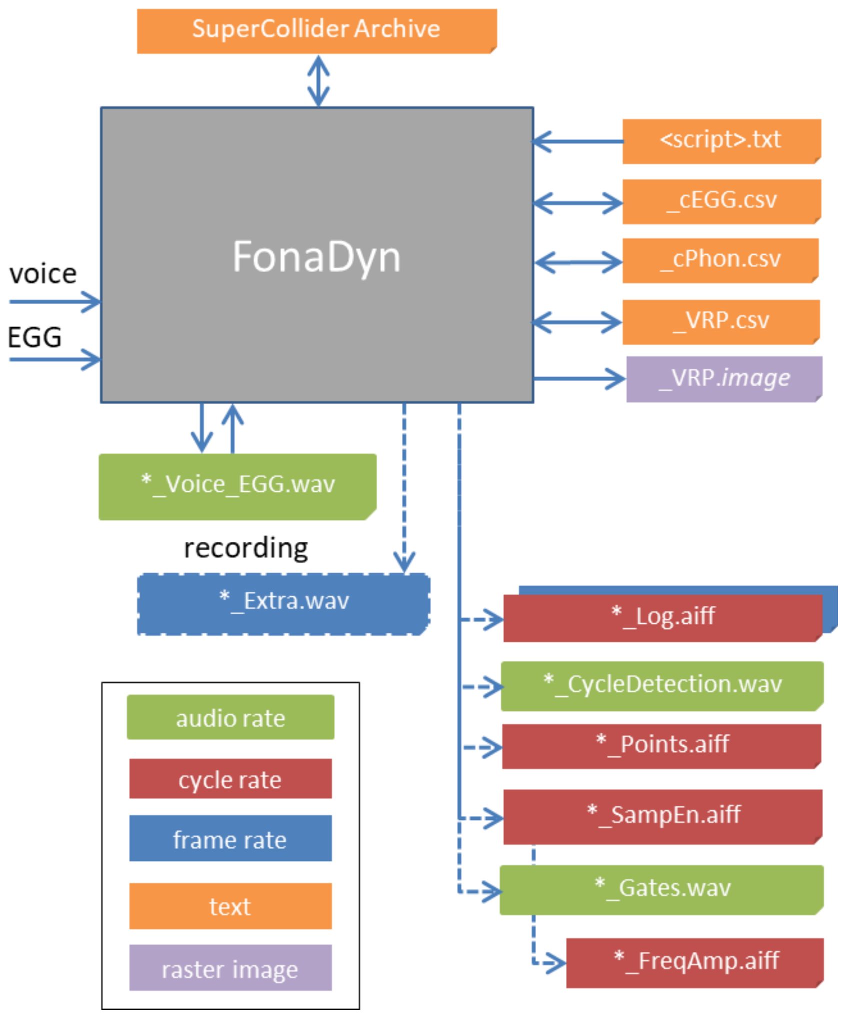
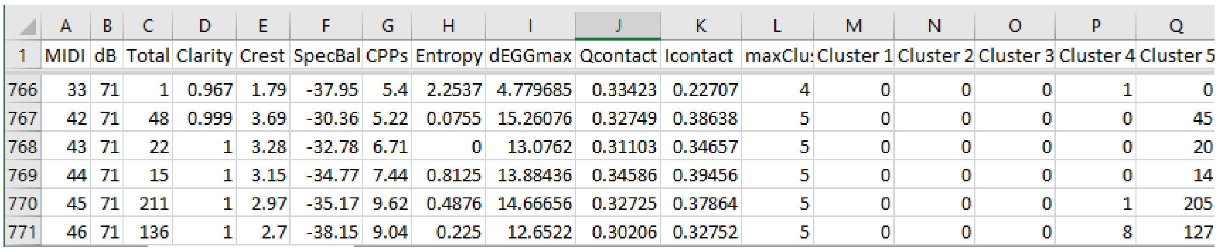


Figure 20. Summary of file types used by FonaDyn 3.1.2.

3.5.4 Voice map data

When an analysis is completed, the data underlying the voice map can be saved to a CSV file (press **Save Map**). An example of a `*_VRP.csv` file, opened as a spreadsheet, is shown in Figure 21. Each row corresponds to an occupied cell in the voice map graph. The first two columns give the cell coordinates. The following columns each correspond to one layer in the voice map. Matlab examples are provided that show how to plot customized voice map charts from this data.

In such voice map files, the order of the columns and the rows does not matter. Columns that are expected but missing will be displayed as a blank layer. Extra columns are allowed in the file, and will not be displayed by FonaDyn. This could be the case for files created with a different version of FonaDyn, or with other software. For instance, you can add other columns for your own purposes, and FonaDyn can still load and display the map data, so long as it does *not* recognize the extra column names.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	MIDI	dB	Total	Clarity	Crest	SpecBal	CPPs	Entropy	dEGGmax	Qcontact	Icontact	maxClu:	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
766	33	71	1	0.967	1.79	-37.95	5.4	2.2537	4.779685	0.33423	0.22707	4	0	0	0	1	0
767	42	71	48	0.999	3.69	-30.36	5.22	0.0755	15.26076	0.32749	0.38638	5	0	0	0	0	45
768	43	71	22	1	3.28	-32.78	6.71	0	13.0762	0.31103	0.34657	5	0	0	0	0	20
769	44	71	15	1	3.15	-34.77	7.44	0.8125	13.88436	0.34586	0.39456	5	0	0	0	0	14
770	45	71	211	1	2.97	-35.17	9.62	0.4876	14.66656	0.32725	0.37864	5	0	0	0	1	205
771	46	71	136	1	2.7	-38.15	9.04	0.225	12.6522	0.30206	0.32752	5	0	0	0	8	127

Figure 21. An example of a `_VRP.csv` *file, opened in a spreadsheet app. Each row holds data for one cell in the voice map. Columns A and B give the cell coordinates in semitones and dB, column C the total number of cycles in the cell, D the most recent value of the clarity metric, E the average of the crest factor of the audio signal, F the average level difference of the spectrum balance (dB), G the mean CPP (dB), H the mean value of the CSE metric, I, J and K the means of the metrics Q_Δ (a.k.a. dEGGmax), Q_{ci} and I_c, and L the number of the EGG cluster with the largest number of cycles. Columns "Cluster #" contain the cycle counts per EGG waveshape cluster. They are then followed by columns "maxCPhon" and "cPhon #" for the cycle counts per phonation-type cluster (not shown here). Only non-empty cells are included in this file.*

For cluster columns "Cluster *N*" or "cPhon *N*", FonaDyn stops parsing when the next higher *N* is not found. So, for example, if the column "Cluster 4" is missing, only three clusters will be read, even if "Cluster 5" is present.

The cycle counts in voice map files are absolute, that is, they are not scaled by f_o . One second of phonation results in 100 cycles at 100 Hz, but 400 cycles at 400 Hz. You can obtain the approximate phonated time by dividing these cycle counts by the corresponding f_o , where $f_o = 220 \cdot 2^{\left(\frac{MIDI-57}{12}\right)}$. Here, f_o is quantized to whole semitones (≈6% increments), so the maximum error in the duration for one cell would be ±3% .

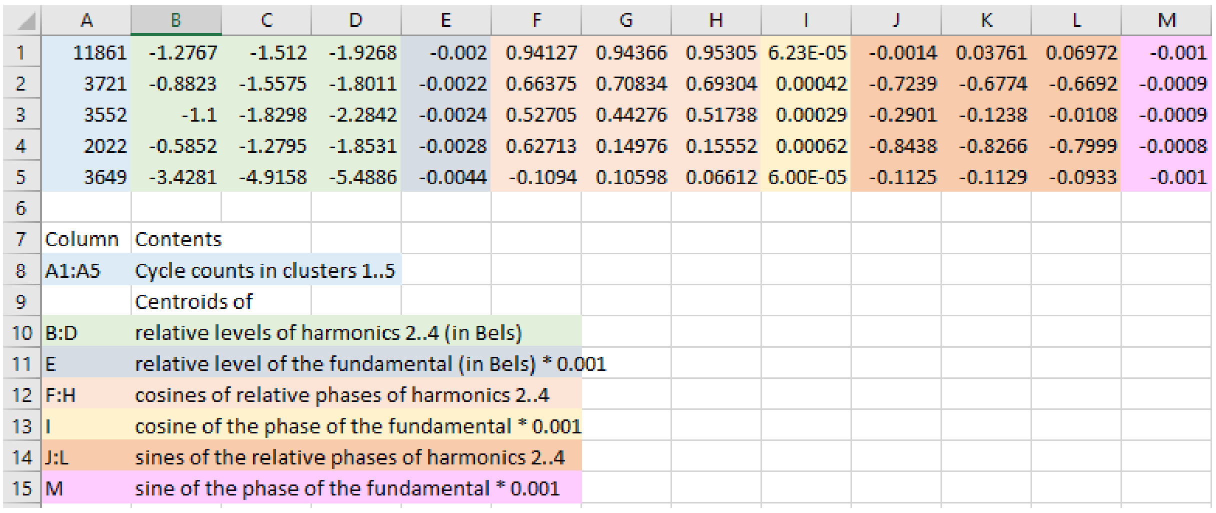
Files of smoothed maps are saved as `_S_VRP.csv` unless you specify another ending for the file name (→2.9, →3.1.11). You can also recognize a smoothed map by inspecting the columns "Total" or "Cluster *N*" or "cPhon *M*". If they contain any fractional cycle counts, the map was smoothed and interpolated. This operation "creates" new cycles, so the Total may not be exactly the sum of the cluster counts.

Files of difference maps are saved as `_D_VRP.csv` unless you specify another ending for the file name (→3.1.11). You can also recognize a difference map by inspecting the "Qcontact" or "Crest" columns in it. If some of the numbers are negative, the map is a difference map.

3.5.5 EGG cluster data

When an analysis is completed, the centroids of the resulting clusters can be saved and then reloaded. Typically, you would do this to continue learning with more input files, or to classify other signals using the same cluster data, or to use the cluster data in analyses outside of FonaDyn.

Figure 22 shows an example of a `*_cEGG.csv` file, with 5 cluster centroids for 4 harmonics, as opened in a spreadsheet, with added comments and colours.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	11861	-1.2767	-1.512	-1.9268	-0.002	0.94127	0.94366	0.95305	6.23E-05	-0.0014	0.03761	0.06972	-0.001
2	3721	-0.8823	-1.5575	-1.8011	-0.0022	0.66375	0.70834	0.69304	0.00042	-0.7239	-0.6774	-0.6692	-0.0009
3	3552	-1.1	-1.8298	-2.2842	-0.0024	0.52705	0.44276	0.51738	0.00029	-0.2901	-0.1238	-0.0108	-0.0009
4	2022	-0.5852	-1.2795	-1.8531	-0.0028	0.62713	0.14976	0.15552	0.00062	-0.8438	-0.8266	-0.7999	-0.0008
5	3649	-3.4281	-4.9158	-5.4886	-0.0044	-0.1094	0.10598	0.06612	6.00E-05	-0.1125	-0.1129	-0.0933	-0.001
6													
7	Column	Contents											
8	A1:A5	Cycle counts in clusters 1..5											
9		Centroids of											
10	B:D	relative levels of harmonics 2..4 (in Bels)											
11	E	relative level of the fundamental (in Bels) * 0.001											
12	F:H	cosines of relative phases of harmonics 2..4											
13	I	cosine of the phase of the fundamental * 0.001											
14	J:L	sines of the relative phases of harmonics 2..4											
15	M	sine of the phase of the fundamental * 0.001											

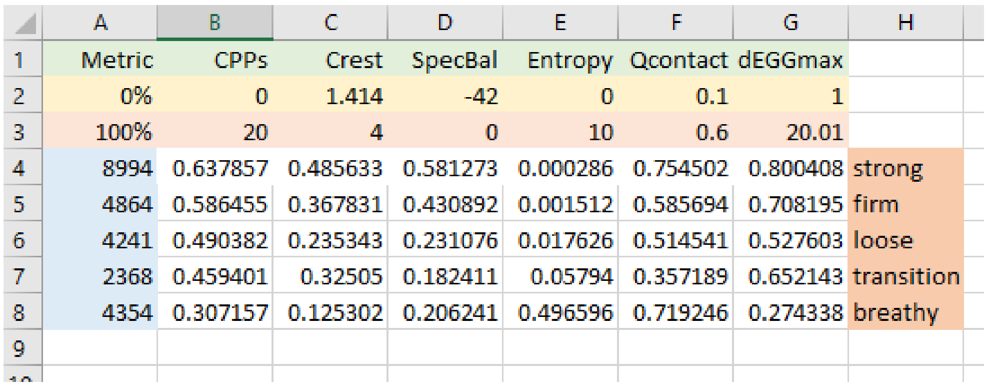
*Figure 22. Example of an EGG cluster data file (*_cEGG.csv), opened in a spreadsheet app. This example is for 5 clusters and 4 harmonics. The number of columns is 1 + 3 × N, where N is the number of EGG harmonics. The rows 7-15 and the colouring are explanatory only – they do not appear in the *_cEGG.csv file.*

From FonaDyn v3.1.0, the meaning of the column that is E in Figure 22 has changed since earlier versions. It used to contain the relative residual power level of that part of the signal that is not accounted for by the given harmonics. It now contains the relative level of the fundamental partial, scaled down by 0.001. This enables plotting of the EGG waveshapes with different relative amplitudes. When you load the earlier kind of file, FonaDyn will post a warning. If you run a re-clustering on the same signal, the result will be very similar but probably not identical.

There exists also a slightly different, much earlier type of EGG cluster files, called `*_clusters.csv`. FonaDyn will still load such files, but not save them.

3.5.6 Phonation type cluster data

When an analysis is completed, the centroids of the resulting clusters can be saved and then reloaded. Typically, you would do this to continue learning with more input files, or to classify other signals using the same cluster data, or to use the cluster data in analyses outside of FonaDyn.



	A	B	C	D	E	F	G	H
1	Metric	CPPs	Crest	SpecBal	Entropy	Qcontact	dEGGmax	
2	0%	0	1.414	-42	0	0.1	1	
3	100%	20	4	0	10	0.6	20.01	
4	8994	0.637857	0.485633	0.581273	0.000286	0.754502	0.800408	strong
5	4864	0.586455	0.367831	0.430892	0.001512	0.585694	0.708195	firm
6	4241	0.490382	0.235343	0.231076	0.017626	0.514541	0.527603	loose
7	2368	0.459401	0.32505	0.182411	0.05794	0.357189	0.652143	transition
8	4354	0.307157	0.125302	0.206241	0.496596	0.719246	0.274338	breathy
9								

*Figure 23. Example of a phonation-type cluster data file (*_cPhon.csv), opened in a spreadsheet app. This example is for 5 clusters (rows 4...8) and 6 metrics (columns B...G). Row 1 gives the metric names, which are the same as in a _VRP.csv file. Rows 2 and 3 specify the range of minimum and maximum values to which each metric will be standardized prior to clustering. Cells A4:A8 give the number of cycles found in clusters 1...5. Cells B4:G8 give the centroid locations on the respective standardized scale (0...1). In column H, a descriptive label can be added for each phonation type. The colouring is for visual clarity only – it does not appear in the *_cPhon.csv file.*

Figure 23 shows an example of a `*_cPhon.csv` file opened in a spreadsheet, with added comments. FonaDyn does not yet have a user interface for controlling the choice of metrics for this mode of clustering. Instead, you specify this in a `*_cPhon.csv` file such as the one shown here. For each metric that you want to use for the clustering, insert a column with the range to which the metric should be normalized. The column title for each metric must be the same as for that metric in voice map files (`*_VRP.csv`). The column order defines the clockwise order around the radar plot. Add rows corresponding to the number of clusters that you want. Initialize the cycle counts in column A to a small positive number. Initialize all the centroid values to something between 0 and 1. Save the file in .csv format, then **Load** it into FonaDyn's phonation type control panel (→3.1.9). If you run with **Learning:** off, then the signal will be classified using the given centroids, one per row. If you run with **Init: Pre-learned**, then the centroid values from this file will be used as seed values. If you run with **Init: Relearn**, the centroid values in this file will be discarded; only the setup of the metrics will be used.

3.5.7 Saving and restoring settings

If **Settings... | Save all settings** is checked when FonaDyn is closed, all settings in the user interface are saved into SuperCollider's internal archive. If you start FonaDyn with `FonaDyn.run`, the default settings are used. If you start FonaDyn with `FonaDyn.rerun`, the settings most recently saved from a previous session are used again. Only one set of settings is saved. If **Save all settings** is checked, any previously stored data set is overwritten when the session is ended.

Exception: the state of the **Save all settings** box itself is *not* restored on `FonaDyn.rerun`, because then you would risk overwriting a good setup by mistake when FonaDyn is closed. Hence you have to check this box before closing FonaDyn, if you want to change the archived settings.

An alternative way of auto-initializing FonaDyn, using script files, is described in the next section.

Because the internal data structures often change across versions of FonaDyn, the archived settings are tagged with the FonaDyn version number. If you update FonaDyn, then the `.rerun` command will not recognize the saved settings from an earlier version, although they may still be stored in the SuperCollider archive. In an emergency, they can be extracted, but that procedure is not described here.

3.5.8 Script files

Script files have two main uses: to bring FonaDyn's many settings into a known state, and to run automatically a sequence of steps, such as processing a whole batch of signal files. You can create a script file with any 'flat text' editor, or by using for example a spreadsheet app or Matlab®. A 'script' is a text file that FonaDyn reads line by line. Each line can contain the setting of a control variable, an action command, or commenting text. Settings can be changed between input files. Some examples are shown in the table on the next page. A more comprehensive list of control settings is given in the file **FonaDyn-Script-Syntax.pdf**.

A script that does not contain a RUN command will set FonaDyn into the specified state, without running any analysis. Scripts are executed from top to bottom. There is no syntax for breaking the order, such as jump or loop statements.

If you will be using FonaDyn repetitively, for acquiring data from many participants, for example, it can help to auto-initialize the relevant settings whenever a new FonaDyn session is started with FonaDyn.run. To do this, you can use an auto-saved state and `FonaDyn.rerun` as described in the previous section; but only one state can be saved in that way. An alternative is to create a named text script and have it execute every time FonaDyn is started. To do so, add in the startup file the statement

```
FonaDyn.config(runScript: "d:/full/path/to/the/scriptfile.txt");
```

Now, when FonaDyn is started, it will automatically run the given script. Therefore, don't include any RUN statement (which would start an analysis as well). Always use forward slashes / in pathnames, even on Windows.

A great benefit of scripts is that they document the settings that you used for a given analysis. Also, the lines in the script are echoed to the SCIDE post window as they are performed, so when a script has completed, you can copy the most recent contents of the post window to a text file.[11] This gives a record also of the dates and clock times at which each output file was created, which gives you a way of back-tracking to the settings that were used for generating a particular output file. Hopefully, such logging will be more automated in a future version.

The script syntax may seem a bit odd, because it consists of actual SuperCollider source code that directly accesses the internal variables of FonaDyn while it is running. In fact, with the statement EVAL <expression> , you can invoke *any* SuperCollider code that fits on one line. This is especially useful for using global variables to hold directory locations that may change from participant to participant, for instance. See the file "test_script.txt" for an example. But, use EVAL only if you know what you are doing; it can cause all manner of havoc.

Even when running a script, FonaDyn has to process signal files in real time. It cannot run silently at higher speed.

[11]) The number of output lines that are saved in the SC post window is 1000 by default. It can be changed in Edit > Preferences > Editor > Post window scrollback.

Script line text	**Meaning**
`// Example of a simple script file`	Double slashes start comments
`io.keepData=false`	Lines starting in lower case are interpreted as assignment statements. Here: set the state of the **Keep data** check box (→3.1.4).
`general.output_directory="C:/Recordings"`	Sets the directory for recordings and log files. Use forward slashes / to separate directory names, even on Windows.

`io.filePathInput="L:/mystudy/recs/fileA_Voice_EGG.wav"`
`// Give the absolute path (not relative),`
`// in double quotes. Long paths are OK.`
	Set the path to the next file to analyse.

`LOAD "L:/mystudy/recs/press-flow_cEGG.csv"`
`// Only files named *_cEGG.csv, *_cPhon.csv`
`// or  *_VRP.csv can be loaded.`
`// If there is an error loading a file,`
`// the script run is aborted.`
	Lines starting in upper case are interpreted as action statements: LOAD, RUN, HOLD, SAVE or EVAL. Here: Load an existing cluster centroids file, for classifying EGG wave shapes. This file will determine the number of clusters and harmonics to use.

`cluster.learn=false // classify instead of learn`	Trailing comments are allowed.

`HOLD  // Press START to continue`
`// You can interact with the user interface`
`//  while the script is running – with care`
	Pause, and check the display panel. When you press ►START, FonaDyn starts processing the first (or next) file. When this has completed, the script will regain control. If you press ■ STOP, the script run will be stopped.

`io.keepData=true`
`io.filePathInput="L:/mystudy/recs/fileB_Voice_EGG.wav"`
	Accumulate the data from the next file into the same voice map.

`RUN`	Proceed with the next file, without the user having to press ►START.

`SAVE "L:/mystudy/maps/AB_VRP.csv"`
`SAVE "L:/mystudy/maps/AB_cEGG.csv"`
`SAVE "L:/mystudy/maps/AB_cPhon.csv"`
	Save the finished voice map or cluster centroid data. If the file name ends in _S_VRP.csv, the map is smoothed and then saved. Currently, *difference* maps cannot be saved from a script.

`// Creating *_Log.aiff files`
`cluster.learn=false`
`clusterPhon.learn=false`
`LOAD "L:/mystudy/maps/AB_cEGG.csv"`
`LOAD"L:/mystudy/maps/AB_cPhon.csv"`
`io.enabledWriteLog=true`
`io.keepInputName=true  // false: time-stamp it`
`io.writeLogFrameRate=0 // 0 for cycle rate`
	How to set up for creating a *_Log.aiff file.
	
	Possible frame rates are cycle rate, 50, 100 or 300 Hz.

3.5.9 Context script files

As a convenience, FonaDyn can optionally create a ‘context script’ when you save a freshly made voice map. This script documents the relevant settings currently in effect, and when it is run, it will restore the context in which the map was created. It is given the same file name as the map, with ‘.Context.txt’ appended. This mechanism lets you rebuild maps exactly, or build another map with the same settings, or rerun after adjusting the settings in some way.

You probably don’t always want context scripts to be created, as they would clutter the folder, so the **Settings...** dialog has a box for explicitly enabling the creation of context scripts. A context script will then be created directly after an analysis run, but only if (a) the clustering panels were both initialized by loading a _cEGG.csv file and a _cPhon.csv file; (b) both panels were set to “Pre-learned” and “Learning: Off”; and (c) the analysis ran to the end of the signal file without being prematurely stopped by the user. If these conditions are all met, and no errors occurred, the **Save Map** button will display its text in green, to indicate that a context script will also be saved. If either of the cluster data files has been modified after the map file was saved, a warning is posted, since this might mean that the map will not become the same after all.

3.5.10 Origin tracking

FonaDyn is still experimental software, and new versions are published sporadically a few times per year. Currently, the files you save with FonaDyn do not contain any explicit information as to which version that created them (except ‘context scripts’, which do). When new metrics, behaviours or improved algorithms are implemented, we strive to maintain backwards compatibility, but sometimes this is not possible. We hope to develop a solution for this, but for the time being, you need to keep track of the FonaDyn version on your own. Always read the latest Release Notes carefully to see if the latest changes might impact your work. A simple strategy is to stick with the same FonaDyn version for the duration of a research study or course round.

3.5.11 Diagnostic output files

Buttons to create files of these extra types are shown only if you have checked the ‘box for nerds’, **Show additional diagnostic features** in the **Settings**. More detailed info can be found in the online help for the class `VRPViewMainMenuOutput`. These files are used mostly for closer examination of unexpected results, as follows.

Cycle Detection Log. This is a two-channel, 16-bit file (`*_CycleDetection.wav`), containing the conditioned EGG signal, and the corresponding cycle trigger pulses. This file affords post-inspection of whether or not the cycle triggering was accurate.

Output Points. This file type (`*_Points.aiff`) contains twice as many tracks as there are harmonics. It contains the cycle-synchronous level and phase *differences*, for accepted cycles only. These are the data that are input to the EGG clustering algorithm (→2.8). The last delta-level track contains the relative level of the residual high frequency energy. The last delta-phase track contains *twice* the phase of the fundamental, relative to the cycle trigger (for internal use).

Sample Entropy. This is a single-channel, cycle-synchronous file `*_SampEn.aiff`. Not scaled.

Frequency and Amplitude. This file type does not have any on/off button, but such a file is always written when either the SampEn measurement is written, or the Points are written. The reason is that you typically would want to see these metrics together. The file is a cycle-synchronous AIFF file with floats as samples. These files are called `*_FreqAmp.aiff`.

Gates Log. This file is called `*_Gates.wav`. It contains five tracks of audio-rate 16-bit samples, with the raw EGG, the conditioned EGG (delayed by 78 ms), and three trigger tracks. It shows exactly where all conditioned EGG pulses were segmented and also which of the pulses that were associated with phonation regular enough to be retained for further analysis. This enables close scrutiny of the data input to the DFT analysis, as well as the extraction (e.g., in Matlab) of cycle-by-cycle signal data with absolute times. Because it becomes big, this file is written only if you have checked that option in the **Settings...** box, and only if there is some other simultaneous cycle-rate output being written.

3.6 Phonation Type Analysis

This and the following section both apply to the analysis of both live and prerecorded signals. This section is about using the acoustic and EGG metrics for clustering different modes of phonation, which is a new feature in FonaDyn 3. The section after this one is about using the EGG *wave shapes* as a basis for the clustering. Apart from that, the two sections are very similar, since the technique of clustering is described in both of them.

At this point, you may wish first to re-visit section 3.1.9, which describes the control panel for phonation type clustering. To see phonation type cluster regions in the voice map, select the **Layer:** “Phonation type clusters”.

3.6.1 Pre-clustered sets of centroids

The easiest way of using the clustering mechanism is to classify incoming signals using pre-created cluster centroid files (`*_cEGG.csv, *_cPhon.csv`) with **Init: Pre-Learned**, **Learning: Off**. This can even be done with live signals in real time. It also allows the simultaneous creation of a Log file for in-the-map-listening. We hope to be able to supply an inventory of pre-clustered centroid files with a future version of FonaDyn, but for the time being, you will have to create them yourself.

3.6.2 Parameters for phonation clustering

Ideally, choosing the *number of clusters* should be as easy as choosing the number of phonation types that you want FonaDyn to identify. To discriminate between breathy and modal voice, for example, two clusters could suffice, in principle. But there is no way of knowing beforehand if the metrics really do change in a way that is so cleanly separable. For example, you may find that you get better discrimination with three clusters, of which two are needed to catch all the modal-type segments [5]. The topic of finding good cluster centroids for a given application is a broad one that is currently being studied at KTH [33]. For the time being, we encourage you to experiment with the default set of metrics (CPP, SB, crest factor, CSE, Q_{ci} and Q_{Δ}), using their default normalizations. If you request only two clusters, you will find that breathy, soft phonation and modal, loud phonation immediately emerge as the two dominant categories. Increasing the number of clusters will resolve more categories on the map; however, requesting more than six clusters is unlikely to give more insights [33].

If you change the number of clusters when a map is displayed, the corresponding layers of the map will be cleared, since they no longer apply.

3.6.3 Initializing the clusters

The outcome of the clustering can be sensitive to the first few seconds that are collected. A good strategy is to edit the signal files such that phonation starts immediately at medium voice amplitude. In this way, the analysis will start out ‘seeded’ with similar properties in all clusters, which soon diverge as the signals change.

Alternatively, to improve the consistency of repeated runs, FonaDyn can wait until some stable phonation is occurring and *then* automatically initialize the clusters. Before pressing ▸START, press the button **Reset Counts** so that it displays **Auto**

Reset instead. When FonaDyn has seen the first 200 ms of continuous phonation, above the 'clarity' threshold, it will clear the cluster data at that point and restart the clustering afresh.

A loud, clear voice should give high values of all metrics except the CSE (SampEn), which should be zero. A soft breathy voice will give low values of most metrics, except CSE, which will be high; while Q_{ci} will tend toward 0.5 in soft voice (which is at 80% of the default range) (→2.5.1). If the centroids radar plot looks scattered and/or random, the clustering has probably degenerated from spurious signals. Wait until phonation is stable and mid-range, and then press the button **Reset counts**. From the signals at that moment, this will generate equal 'seed' centroids that soon drift apart from one another. The end result will depend slightly on exactly *when* **Reset counts** is pressed. The final outcome will however become quite stable if you record the signals and make a second or even a third pass over the data, using the setting **Init: Pre-learned** (without pressing **Reset counts** or **Unload** in between).

3.6.4 Rearranging clusters

Another issue of initialization is that the algorithm's allocation of cluster numbers to different phonation types is necessarily arbitrary. Since each cluster number is assigned a different display colour, this means that the cluster *colours* are unlikely to be the same every time, when a recording protocol is repeated in the learning phase. Once the learning can be turned off, as for classifying signals, this is no longer a problem.

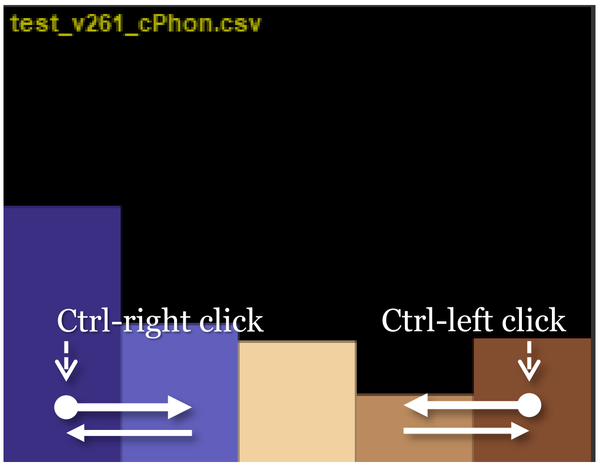


Figure 24.
You can rearrange the mapping of clusters to colours by swapping the positions of two clusters at a time. Since this is a swap, repeating a click in the same place will undo the change.
The set of colours is given by the number of clusters.

To make your figures consistent, you may wish to re-arrange the cluster order. This can be done only when FonaDyn has stopped. *Ctrl*-click left or right on the vertical cycle-count bars (Figure 24). Two adjacent clusters are then swapped, as shown here, and the cluster colours in the current voice map are also updated. Repeat this until you have achieved the desired order. The first and last clusters are treated as adjacent; the order 'wraps around.' Phonation type cluster data are saved to files of type `*_cPhon.csv`.

Another way to rearrange clusters is to edit the .csv file directly (→3.5.6). Use a spreadsheet program to rearrange the data clusters into the order that suits your application. A tip: first, colour the text on each *row*, 4th row-to-bottom, to a sequence of colours similar to those shown in FonaDyn's bar graph, left-to-right (Figure 24). It is then somewhat easier to rearrange manually those rows into the order that you want (for instance, from weak to strong phonation). With this method, you can also *remove* a cluster that you don't want, simply by deleting that line. When done, save the `_cPhon.csv` file, preserving the .csv format, and reload it into FonaDyn. If you do re-order the clusters in this way, then any maps you may have made before re-ordering will have to be re-made.

3.6.5 Clustering in multiple passes

Because FonaDyn is a real-time program, it cannot know anything at the START about what the data set as a whole is going to look like. This applies to recordings as well as to live signals. FonaDyn therefore has to adapt the clusters to the data as it goes along. This means that, on the first pass, the cluster colours early in the recording will not correspond to quite the same metric combinations as late in the recording. Over time, the clusters will become stable, but initially, it helps if *the range of variation* in the early part of the signal is representative of that in the rest. Therefore it is a good idea to try to include short excerpts from the full range of conditions as early as possible in a recording, and also to re-run a second or even third pass over the same signal, using the cluster data from the first pass for initialization.

3.6.6 Picking phonation types from the map

You can 'teach' FonaDyn about phonation types, according to what you hear, via the function for 'listening from maps'. First, prepare a Log file as described in section 3.4. To do this, you will need to create a 'placeholder' _cPhon.csv file; the resulting phonation type clusters can later be replaced with your own choices. Then choose **Listen** so that the waveform window appears. Poke around and listen in the voice map, until you think you have identified the kinds of phonation that you want to teach FonaDyn to recognize. Then follow these steps. The list looks long, but each step is very quick.

1. In the phonation type clusters panel (Figure 15), press **Unload**. The clusters associated with the map will disappear, but that's all right.
2. Choose **Learning: On**.
3. Decide on how many phonation types you want to define, and enter this number in the **Phon clusters** field. A default 'cobweb' radar plot is shown to the left, and a colour palette is shown to the right.
4. If you change the number of clusters, then the layer "Phonation type clusters" will be cleared from the current voice map. That's all right, too; just choose another layer.
5. Click on the radar plot. This displays the metrics control panel instead of the coloured bars.
6. Use the horizontal slider to select Cluster 1, giving a dark blue background.
7. Now click and listen in the voice map. Find a location where the voice production is representative of the phonation type that you want to assign to Cluster 1.

8. Press the key for ‘1’ on the numeric keypad (with NumLock on), or in the top row on the keyboard. Notice how the radar plot for cluster 1 changes, and how the little squares representing “centroid coordinates” for cluster 1 assume new positions. FonaDyn has picked the mean values from the corresponding layers in the map, at the clicked position, and has put them into a new centroid.
9. In the **Edit label...** text field at the bottom, enter a short text describing this phonation type; then press Enter.
10. Repeat steps 6...9 for the remaining phonation types, using the number corresponding to the cluster.
11. Choose **Save clusters** to save your new phonation types into a `_cPhon.csv` file.
12. Now run an analysis with **Learning: Off**, to see if your setup of phonation types is an appropriate one for your purposes. The “Phonation type clusters” layer of the voice map will be built using the cluster centroids that you specified.

3.6.7 Manual setting of phonation type clusters

If you feel confident about the role of each metric, then you can assign centroids manually. First choose **Learning: On**. Then, for each metric, drag the small square to the desired percentage. This can be as simple as thinking ‘low CPP’, ‘high SpecBal’, etc. This is instead of picking centroids from the map, as in steps 7 and 8 in the previous section. Then do steps 9...12 as in the previous section. The arrow keys, too, can be used to move the squares.

In research projects, one might typically use Matlab® to do batch clustering of many input files for groups of voice types, and then load the resulting centroids as precomputed `_cPhon.csv` files.

3.7 EGG Shape Analysis

This section applies to the analysis of both live and prerecorded signals. To see EGG waveshape cluster regions in the voice map, select the **Layer:** “EGG waveshape clusters”.

3.7.1 De-noising the EGG signal

All EGG devices currently on the market are somewhat noisy, some more than others. Generally, these devices are designed to study the EGG signal only when there is vocal fold contacting, and the signal amplitude is relatively large. But low-amplitude EGG signals are also useful, for pinpointing the threshold of collision. To do so reliably, a signal-to-noise ratio of at least 40 dB is desirable. Even then, the system noise of the device will interfere, in particular with the measurement of the Q_Δ metric. To mitigate this, FonaDyn has a built-in noise suppression scheme. This de-noising is applied only on playback, not on recording. When recording, the resulting `_Voice_EGG.wav` file contains the raw input signal. For best results, you need to set the threshold for noise suppression to suit your equipment. Note that de-noising will affect the Q_Δ (and hence also the I_c), especially in soft voice, so it is important to make a note of which setting you have chosen for any given analysis. De-noising has little effect on the other EGG metrics, however, and none on the audio metrics.

The ’Moving EGG’ panel contains a **De-noise** field, with a number that can be set between 0 and 5. Here, zero means no noise suppression at all, 0.1 – 0.3 means suppression of noise typical of a good EGG device, and 0.5 can suppress more noise as in an especially weak EGG signal (misplaced electrodes?), or in a not-so-good device. The higher this number, the more noise will be removed. Above 1.0 or so, you will start losing the higher partials in the EGG, if they are weak to begin with.

During the recording, have the participant phonate as softly as possible so that the EGG signal looks like a ragged sine wave. You cannot de-noise while recording,[12] but, while playing back the recording, gradually increase the **De-noise** number (it can be dragged with the mouse) until the sine wave looks clean. If the number needs to be more than about 1 then your EGG signal is probably too noisy to trust. On the next page are shown examples of very soft phonation, with a ‘raw’ EGG signal (Figure 25) and the same signal de-noised (Figure 26).

The de-noising in FonaDyn uses spectral thresholding, which is a textbook technique for suppressing white noise. It works as follows: the EGG signal is transformed into the frequency domain. There, any frequencies whose amplitudes are below the chosen noise threshold level will be reduced even further, using a 4:1 level expander in each of 1024 frequency bins. The expanded signal is then transformed back into the time domain. Figure 27 shows the approximate relationship between the **De-noise** value and the expander threshold level, relative to full scale.

Different EGG devices have different noise levels, and their system noise may not always be white, i.e., spectrally flat. You will have to experiment to find the best setting for your system. If you change the analog output level on your EGG device, then you may also need to adjust the de-noising threshold. The strategy is to gradually increase the **De-noise** value such that the moving EGG in the softest phonations looks like a clean sine wave, and the Q_{Δ} is mapped to clear green (=1) in the softest phonations, but does not change in the loudest phonations, as in Figure 25 and Figure 26. You can do this while an analysis is running. You can also enable listening to the de-noised EGG signal on playback (→3.1.10). In order to inspect the effect of the de-noising in detail, you can re-record an existing recording (→3.2.7). This stores the conditioned signals, rather than the raw input signals. Once you have found an adequate de-noising setting, document it with all your analyses, and do not change it during a definitive analysis run.

Awkwardly, some EGG recordings are *very* noisy. Even they can be silenced, if the **De-noise** number is set to a high value, but this will soften the EGG pulse as well, corrupting the EGG metrics. Even worse, some EGG devices contain an automatic gain control that autonomously varies the gain, thus moving the noise floor up and down during acquisition. FonaDyn’s noise threshold is fixed; it cannot automatically follow a shifting noise level. Furthermore, some EGG signals contain interfering side tones and/or AC hum that might sit stubbornly above the de-noising threshold, and thus will not be attenuated. Check the spectrum of the EGG signal to see if this is the case.

[12]) ...because the visual feedback delay would be annoying. Also, if an analysis is running with ‘context save’ enabled (→3.1.10, 3.5.9), the de-noising value is locked. That’s to ensure that the voice map can be reconstructed later.

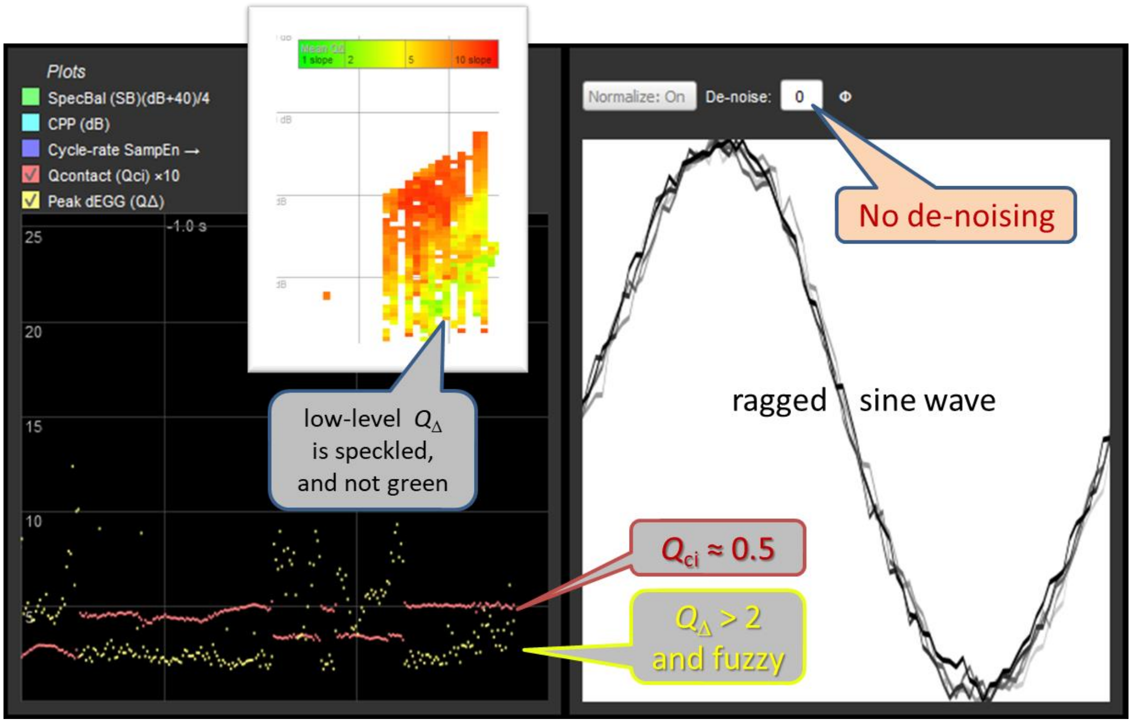


Figure 25. Appearance of an EGG signal in soft phonation, without any de-noising.

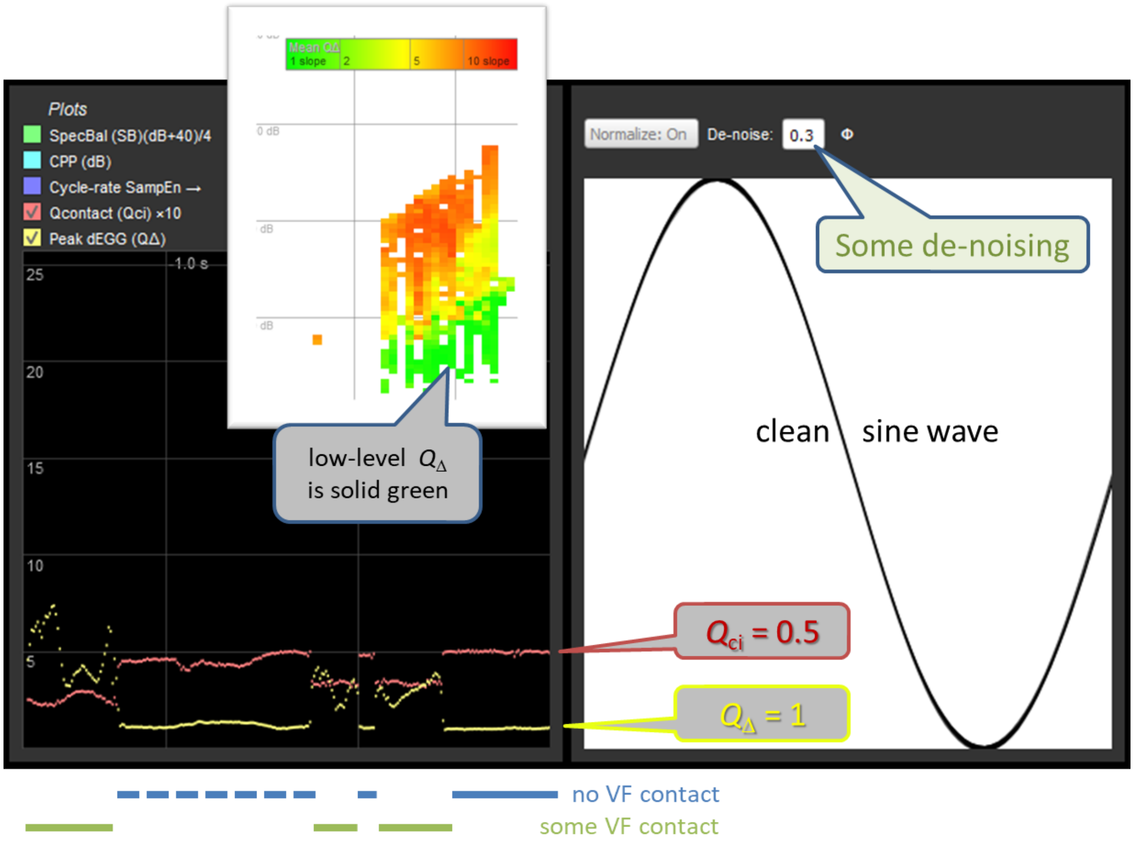


Figure 26. Appearance of the same EGG signal as in the previous figure, but with appropriate de-noising. Note how Q_Δ of the de-noised signal is lower and less erratic overall, and how it descends to 1 when there is no vocal fold contact.

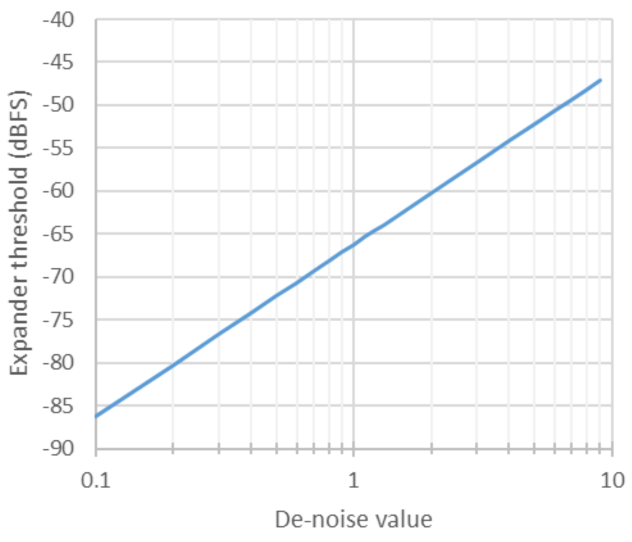


Figure 27. The de-noising expander threshold level, relative to full scale in the EGG signal, for a given de-noise value.

To get a cleaner Q_Δ in earlier versions of FonaDyn, you had to make a Log file and then resynthesize the EGG pulses from the Fourier Descriptors using the accompanying Matlab routines, in a post-processing procedure. This is still possible, with the advantage that with a fixed number of descriptors (harmonics), the Q_Δ does not depend on f_o . However, the new noise suppression mechanism (built-in from version 3.0.5) operates in real time, and also increases the accuracy of the subsequent period segmentation (→2.4.2). Both methods enable Q_Δ to descend to its theoretical minimum of 1, so as to accurately indicate non-contacting. They also result in a cleaner estimate of Q_{ci} at low voice effort levels (where Q_{ci} will be close to 0.5).

Some digital audio workstation (DAW) software apps contain various de-noising tools. FonaDyn's de-noising works in real time without manual editing, but with signals that are both particularly difficult and important, you might find it to be worth the extra effort of first running the EGG signal through such a tool.

3.7.2 Pre-clustered sets of waveshape centroids

< still in preparation >

3.7.3 Parameters for EGG clustering

Ideally, choosing the *number of clusters* should be as easy as choosing the number of phonation types that you want FonaDyn to identify. To discriminate between modal and falsetto singing, for example, two clusters could suffice, in principle. But there is no way of knowing beforehand if the EGG waveform really does change in a way that is so cleanly separable. For example, you may find that you get better discrimination with three clusters, of which two are needed to catch all the falsetto-mode cycles [5].

Therefore, a good strategy may be first to specify more clusters than you actually need, and let FonaDyn try. The most common EGG wave-shapes will tend to produce a few well-populated clusters, while the other clusters will contain much fewer cycles. It helps to edit your recordings first, so as to eliminate pauses and ambient sounds. If your signal contains pauses or other non-voice events, these may give rise to one or two 'trash' clusters, holding weird pulse shapes. You can save and edit the cluster data files to remove such trash clusters. An example of this method is given in [18].

The *number of harmonics* will depend on your research question. If you want to *see* a close representation of the actual EGG pulse shapes, 10 harmonics are usually needed. If you think that the feature you are investigating is a low-frequency phenomenon, or if you are mostly interested in the CSE metric, then fewer harmonics will usually be enough. The default of 10 harmonics seems to be appropriate for most cases.

If you change the number of clusters or harmonics when a map is displayed, the clustered layers of the map will be cleared, since they no longer apply.

3.7.4 Initializing the clusters

The outcome of the clustering is very sensitive to the first few cycles that are detected. There is usually some quiet before the subject starts phonating, which may cause some spurious 'cycles' to be detected in the background noise. Then, some of the initial centroids will be too far apart in their 3*N*-dimensional space to describe EGG waveforms, and most subsequent real EGG cycles will tend to be 'captured' by the one or two 'random' centroids that happen to best describe EGG signals.

One strategy can be to edit the signal files such that phonation starts immediately at medium voice amplitude. In this way, the analysis will start out 'seeded' with similar waveshapes in all clusters, which soon diverge as the signal changes.

Alternatively, to improve the consistency of repeated runs, FonaDyn can wait until some stable phonation is occurring and *then* automatically initialize the clusters. To use this feature, then before pressing ▸START, press the buttons **Reset Counts** so that they display **Auto Reset** instead. When FonaDyn has seen 200 ms of continuous phonation, above the 'clarity' threshold, it will clear the cluster data at that point and restart the clustering afresh.

If the cluster centroids graph looks scattered and/or random, the clustering has probably degenerated from spurious signals. Wait until phonation is stable and mid-range, and then press the button **Reset counts**. From the EGG waveform at that moment, this will generate equal 'seed' centroids that soon drift apart from one another. The end result will depend slightly on exactly *when* **Reset counts** is pressed. The final outcome will however become quite stable if you record the signals and make a second or even a third pass over the data, using the setting **Init: Pre-learned** (without pressing **Reset counts** or **Unload** in between).

3.7.5 Clustering in multiple passes

Because FonaDyn is a real-time program, it cannot know anything at the START about what the data set as a whole is going to look like. This applies to recordings as well as to live signals. FonaDyn therefore has to adapt the clusters to the data as it goes along. This means that, on the first pass, the cluster colours early in the recording will not correspond to quite the same wave-shapes as late in the recording. Over time, the clusters will become stable, but initially, it helps if *the range of variation* in the early part of the signal is representative of that in the rest. Therefore it is a good idea to try to include short excerpts from the full range of conditions as early as possible in a recording, and also to re-run a second or even third pass over the same signal, using the cluster data from the previous pass for initialization.

3.7.6 Rearranging clusters

Another issue of initialization is that the algorithm's allocation of cluster numbers to different EGG wave shapes is necessarily arbitrary. Since each cluster number is assigned a different display colour, this means that the cluster *colours* are unlikely to be the same every time, when a recording protocol is repeated in the learning phase. Once the learning can be turned off, as for classifying signals, this is no longer a problem.

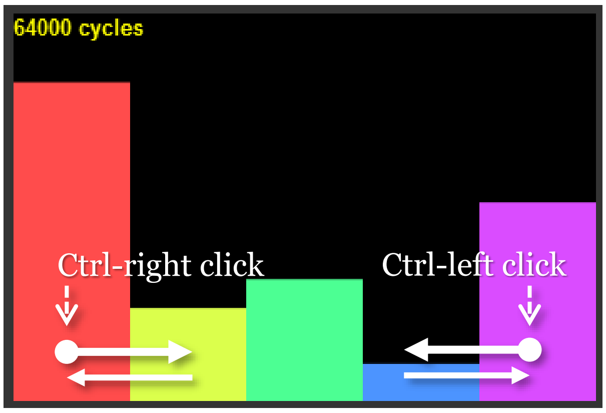


Figure 28.
You can rearrange the mapping of clusters to colours by swapping the positions of two clusters at a time. Since this is a swap, repeating a click in the same place will undo the change.
The set of colours is given by the number of clusters.

To make your figures consistent, you may wish to re-arrange the cluster order. This can be done only when FonaDyn has stopped. *Ctrl*-click left or right on the vertical cycle-count bars (Figure 28). Two adjacent clusters are then swapped, as shown here, and the cluster colours in the current voice map are also updated. Repeat this until you have achieved the desired order. The first and last clusters are also treated as adjacent; the order 'wraps around'. EGG cluster data are saved to files of type `*_cEGG.csv`.

Another way to rearrange clusters is to edit the .csv file directly. Use a spreadsheet program to rearrange the data clusters into the order that suits your application. A tip: first, colour the text on each *row*, top-to-bottom, to a sequence of colours similar to those shown in FonaDyn's bar graph, left-to-right (Figure 28). It is then somewhat easier to rearrange manually those rows into the order that you want (for instance, from weak to strong phonation). With this method, you can also *remove* a cluster that you don't want, simply by deleting that line. When done, save the `_cEGG.csv` file, preserving the .csv format, and reload it into FonaDyn. If you do re-order the clusters in this way, then any maps you may have made before re-ordering will have to be re-made.

3.7.7 Detecting specific EGG wave shapes

Editing `_cEGG.csv` files also comes in handy for constructing cluster files that capture rare but interesting pulse shapes. If you want FonaDyn to look for a specific EGG wave shape, proceed as follows. Using an audio editor, or some numerical procedure, create a `*_Voice_EGG.wav` file that contains mostly pulses of that waveshape. Train FonaDyn on that waveshape, so that it looks right when resynthesized, and then save the `*_cEGG.csv` file. Use a spreadsheet application or flat text editor to copy the row for the corresponding centroid, and append that row into another existing clusters file. Save the new spreadsheet as a new `*_cEGG.csv` file. Repeat this for several different waveshapes, if you like. Then load your new clusters file for classifying signals, with **Init: Pre-learned** and **Learning: off**. Adjust the voice map display to show only the cluster of your rare waveshape. When it occurs in the input signal, it will appear on the voice map. FonaDyn can manage up to 20 cluster centroids (wave shapes) in one file.

3.8 Analyzing multiple takes or multiple files

By default, FonaDyn clears the current cluster data and the voice map data when you press START. If instead you want to continue accumulating recordings into the current voice map, check the box **Keep data** that is to the left of the START button. The **Keep data** option could be appropriate for a long acquisition protocol, during which a subject needs to pause, or for analysing a batch of multiple input files for which the results should be combined into one voice map. If you are analyzing a batch of files, you will want the data to be cleared on START, but not for the following files. If so, then defer checking **Keep data** until after you have pressed START.

In most cases, you will also want to choose the clustering setting **Init: Pre-learned**. Choose **Learning: On** to continue the same clustering operation across input takes or files. Choose **Learning: Off** to classify all the input files using one fixed clustering (which must first be **Load**ed). When **Init: Pre-learned** is chosen, the **Auto Reset** function is disabled – otherwise it would clear the learned data.

The **Load Map** button lets you load a previously saved voice map for inspection. Then, check the box **Keep data** before starting, and you can continue accumulating voice recordings into the loaded map. The current cluster data must be consistent with the existing map, or the results will be meaningless; use **Save Cluster/Load Cluster** to ensure it.

More complex batches can be run using FonaDyn's script mechanism (→3.5.8).

3.9 Using FonaDyn with RME audio interfaces

The digital audio interfaces from RME are very versatile and ideal for multichannel acquisition. Several tips for using them with FonaDyn are given in the separate document *Using FonaDyn with RME audio interfaces*. For instance, you can create a setup that lets you build maps by browsing in a long signal file, instead of having to run through the whole file.

The models Fireface 400, UCX and UCX II as well as the Babyface all have their microphone preamps on inputs 1 and 2. This is what FonaDyn expects. The larger models Fireface 802, and the UFX models all have their microphone preamps on inputs 9-12; while those on the model 800 are on inputs 7-10. You can easily re-route FonaDyn's two input channels, using the `FonaDyn.config` function (→1.4) or the Record inputs list (→3.1.10).

3.10 Known limitations

FonaDyn version 3.1.2 has the following known limitations.

1) **Multiple instances are not supported.** Invoking `FonaDyn.run` when the FonaDyn window is already running will have no effect. Invoking `FonaDyn.run` from a second instance of SuperCollider will open a second FonaDyn window, but running the two instances simultaneously will give unpredictable results.
2) **CPU load**. The higher the fundamental frequency, the more EGG cycles have to be handled every second. In the high soprano range, FonaDyn may struggle noticeably to keep up, unless the computer is very fast. The display may update less smoothly, in fits and starts. The signal processing parts typically take less than 30% of the total. Instead, most of FonaDyn's CPU load concerns the graphics. You can reduce the load by hiding those graphic displays that are not needed, especially the Plots window. If your computer has a power-saving speed setting, make sure that it is set for highest performance, not for power saving. On Windows laptops, this setting is reached by clicking on the battery icon in the task bar. Running video and/or screen-sharing software such as Skype or Zoom at the same time may not be possible.
3) "Pure", algorithmically generated test signals that have sequences of true-zero amplitude in the time or frequency domains have been seen very occasionally to cause errors, typically caused internally by attempting to divide by zero. Live signals always contain a little noise.
4) Occasionally, with file input and/or output, FonaDyn will **fail to start or stop** properly. This can be due to a format error in the input file (did you choose the right file?), but more likely to inter-process handshaking errors, or network delays. Working with all files on a local disk is usually best.
5) It may happen that the **server process is orphaned**, that is, SCSYNTH loses its connection with SCLANG. The solution is go to the SCIDE window and invoke the command **Kill All Servers** from the **Server** menu (or evaluate `Server.killAll`), and then start again.
6) Specifying the highest numbers of clusters and harmonics (up to 20, 20) may cause SCSYNTH to complain that the connection diagram becomes too complex. (To appreciate this, you might look at the file FonaDyn-code-diagrams.pdf.) If you really want that many, this can be allowed by saving into the SC startup file the line

```
Server.local.options.numWireBufs = 256;  // the default is 64
```

   Now, restart SuperCollider, and you will probably be able to run up to the maximum.

Under the Windows Defender anti-virus system, the first time you run FonaDyn in a session, the server process SCSYNTH in SuperCollider may take a long time to boot (one or two minutes). You have to wait until the numbers at the bottom right turn green, which means that SCSYNTH is up and running. On subsequent starts, there is no delay. To avoid this delay, exclude the *processes* SCSYNTH.EXE and SCLANG.EXE from anti-virus monitoring. This will also improve performance.

Acknowledgments

FonaDyn would not exist without the long-standing collaboration, support and friendship of Peter Pabon, whose decades of experience in voice mapping are unsurpassed. The foundation version of FonaDyn was written in 2015-2016 by Dennis Johansson for his M.Sc. degree project in Computer Science [15]. He did it so well that S.T. has been able to build on it ever since.

Much of the EGG-related theory behind FonaDyn was worked out by Andreas Selamtzis, in his doctoral thesis. Isak Nilsson, in the course of *his* M.Sc. degree project [18], ran tests on MacOS, recompiled *PitchDetection* UGen for FFTW on MacOS, and contributed supporting Matlab code. Most of Ternström's doctoral students to date have been working on voice mapping in one way or another: Anick Lamarche, Peter Pabon, Andreas Selamtzis and Huanchen Cai.

FonaDyn continues to be developed by Ternström and co-workers. The work has hitherto been funded by international stipends, the Swedish Research Council (Vetenskapsrådet), project 2010-4565, and by KTH faculty grants.

FonaDyn is written in SuperCollider [3], an interactive real-time audio and music processing environment, first created by James McCarthy. SuperCollider is cross-platform (Windows, Mac, Linux, iPhone, and more) and open-source freeware. We are greatly indebted to the SuperCollider community [3] for the wonderful work that they do in maintaining and developing this brilliant language/server/IDE, and in helping its users. Please support this community, if you can.

References

[1] Ternström S, Pabon P, Södersten M (2016). The Voice Range Profile: its function, applications, pitfalls and potential. *Acta Acustica united with Acustica*, **102**(2), 268-283.

[2] Ruviaro, B. A Gentle Introduction to SuperCollider. CCRMA 2015 https://ccrma.stanford.edu/~ruviaro/texts/A_Gentle_Introduction_To_SuperCollider.pdf

[3] SuperCollider website: http://supercollider.github.io/

[4] Ternström S (2019). Normalized time-domain parameters for electroglottographic waveforms. J Acoust Soc Am.;**146**(1):EL65-EL70. doi:10.1121/1.5117174 (included in the folder FonaDynExtras)

[5] Selamtzis A, Ternström S (2014). Analysis of vibratory states in phonation using spectral features of the electroglottographic signal. *J. Acoust. Soc. Am.*, **136**(5), 2773-2783.

[6] McLeod P, Wyvill G (2005). A Smarter Way to Find Pitch. *Proc Int'l Computer Music Conf*; ICMC 2005, 138-141. [An implementation of the above, called "Tartini", is included with the 'SC3-plugins' library of signal function blocks.] Permalink: http://hdl.handle.net/2027/spo.bbp2372.2005.107.

[7] Pabon JPH (1991): Objective acoustic voice-quality parameters in the computer phonetogram. *J. Voice* **5**(3) 203-216.

[8] Dolanský LO (1955). An Instantaneous Pitch-Period Indicator. *J. Acoust. Soc. Am.* 27, 67-72 (1955); http://dx.doi.org/10.1121/1.1907499

[9] Herbst C, Ternström S. A comparison of different methods to measure the EGG contact quotient. *Logopedics Phoniatrics Vocology*, 2006; **31**(3):126-138. doi:10.1080/14015430500376580.

[10] McFee B (2012). *More like this: machine learning approaches to music similarity*. PhD thesis, University of California at San Diego, 186 p (algorithm B.1, p. 152), http://bmcfee.github.io/papers/bmcfee_dissertation.pdf. [An implementation of the above, called "KMeansRT", is included with the 'SC3-plugins' library of signal function blocks. FonaDyn supplements this with "KMeansRTv2", which provides the option of continued learning or classifying with a pre-learned vector of centroids.]

[11] Richman JS, Randall Moorman J (2000). Physiological time-series analysis using approximate entropy and sample entropy. *American Journal of Physiology*, **278** (6), 2039-2049.

[12] Yu-Hsiang Pan, Yung-Hung Wang, Sheng-Fu Liang, Kuo-Tien Lee (2011). Fast computation of sample entropy and approximate entropy in biomedicine. *Computer Methods and Programs in Biomedicine*, **104** (3), 382-396.

[13] Delgado-Bonal A, Marshak A (2019). Approximate Entropy and Sample Entropy: A Comprehensive Tutorial. *Entropy* **21**, 541; doi:10.3390/e21060541.

[14] Fabris C, De Colle W, Sparacino G (2013). Voice disorders assessed by (cross-) Sample Entropy of electroglottogram and microphone signals. *Biomedical Signal Processing and Control*, **8**(6), 920-926, ISSN 1746-8094, http://dx.doi.org/10.1016/j.bspc.2013.08.010.
(http://www.sciencedirect.com/science/article/pii/S1746809413001237)

[15] Johansson D (2016). *Real-time analysis, in SuperCollider, of spectral features of electroglottographic signals*. M.Sc. degree thesis in computer science, KTH Royal Institute of Technology, Stockholm, Sweden. Available online at this link (June 2022).

[16] Schutte HK, Seidner W (1983). Recommendation by the Union of European Phoniatricians (UEP): Standardizing Voice Area Measurement/ /Phonetography. *Folia Phoniatrica* 1983;35:286–288, (DOI:10.1159/ /000265703)

[17] http://www.expert-sleepers.co.uk/ You will need the modules ES-3, ES-6, optionally the ES-7, two TosLink optical cables, an extra 'row power' supply module and a rack or case in which to mount the modules.

[18] Nilsson I (2016). *Electroglottography in real-time feedback for healthy singing*. M.Sc. degree thesis in computer science and communication, KTH Royal Institute of Technology, Stockholm, Sweden. Available online at this link (November 2021).

[19] Fraile R, Godino-Llorente JI. Cepstral peak prominence: A comprehensive analysis. *Biomedical Signal Processing and Control*. 2014; 14(1): 42-54. doi:10.1016/j.bspc.2014.07.001

[20] Awan SN, Solomon NP, Helou LB, Stojadinovic A (2013). Spectral-cepstral estimation of dysphoria severity: External validation. *Ann Otol Rhinol Laryngol*. 2013;122(1):40-48. doi:10.1177/000348941312200108.

[21] Fallgren P, Malisz Z, Edlund J (2019). How to Annotate 100 Hours in 45 Minutes. *Proc. Interspeech 2019*, 341-345, doi:10.21437/Interspeech.2019-1648.

Some other relevant sources, including examples of research done with FonaDyn

[22] Selamtzis A (2014). *Electroglottographic analysis of phonatory dynamics and states*. Licentiate thesis in Speech and Music Communication, Stockholm: KTH Royal Institute of Technology, 2014. , vii, 31 p. Available online at http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-145692

[23] Selamtzis A, Ternström S (2017). Investigation of the relationship between the electroglottogram waveform, fundamental frequency and sound pressure level using clustering. *J. Voice,* **31** (4), July 2017, 393-400, available online at http://dx.doi.org/10.1016/j.jvoice.2016.11.003.

[24] Roubeau B, Henrich N, Castellengo M (2009). Laryngeal Vibratory Mechanisms: The Notion of Vocal Register Revisited. *J. Voice*, **23** (4), July 2009, 425–438.

[25] *Matlab* © The MathWorks, Inc. www.mathworks.com

[26] Herbst C (2004). MovingEGG. Available online at this link. (Accessed November 2021).

[27] Švec JG, Granqvist S (2010). Guidelines for selecting microphones for human voice production research. *American Journal of Speech-Language Pathology*, **19**, 356–368, November 2010.

[28] Ternström S, Pabon P. Voice Maps as a Tool for Understanding and Dealing with Variability in the Voice. *Applied Sciences* 2022;12(22):11353. https://doi.org/doi:10.3390/app122211353 , Open Access.

[29] Šrámková H, Granqvist S, Herbst CT, Švec JG. The softest sound levels of the human voice in normal subjects. *J Acoust Soc Am*. 2015;137(1):407-418. doi:10.1121/1.4904538

[30] Ternström S, D'Amario S, Selamtzis A (2018). Effects of the lung volume on the electroglottographic waveform in trained female singers. *J. Voice* **34** (3) pp. 485.e1-485.e21. (2020) https://doi.org/10.1016/j.jvoice.2018.09.006, Open Access.

[31] Lã FMB, Ternström S (2020). Flow ball-assisted voice training: Immediate effects on vocal fold contacting. *Biomed. Signal Process. Control*, **64**. 2020;**62**:102064. https://doi.org/10.1016/j.bspc.2020.102064

[32] Patel RR, Ternström S. Quantitative and Qualitative Electroglottographic Wave Shape Differences in Children and Adults Using Voice Map–Based Analysis. *J Speech, Lang Hear Res*. 2021;**64**(8):2977-2995. https://doi.org/10.1044/2021_jslhr-20-00717

[33] Cai H, Ternström S. Mapping Phonation Types by Clustering of Multiple Metrics. *Appl Sci*. 2022;12(23):12092. https://doi.org/doi:10.3390/app122312092 , Open Access.

Contents in detail

0 Overview 5
1 PART ONE - Getting started 7
1.1 Hardware requirements 7
1.2 Software setup 8
1.2.1 Versions of SuperCollider 8
1.2.2 Windows 9
1.2.3 MacOS 10
1.2.4 Linux 10
1.2.5 Installing FonaDyn into SuperCollider 11
1.2.6 Centrally managed installation 12
1.3 Audio device configuration 14
1.4 FonaDyn configuration options 15
2 PART TWO - Theory 17
2.1 Introduction 17
2.2 Signal processing overview 17
2.3 Audio processing 19
2.3.1 Input selection 19
2.3.2 Audio preprocessing 19
2.3.3 Representation of sound levels 19
2.3.4 Audio-based metrics 21
2.3.5 More about the CPP 23
2.4 EGG processing 24
2.4.1 Preprocessing 24
2.4.2 Period segmentation 25
2.5 EGG time-domain metrics 26
2.5.1 The contact quotient Q_{ci} 26
2.5.2 The Q_{Δ} metric 27
2.5.3 The I_{c} metric 28
2.5.4 The $\mathrm{HRF}_{\mathrm{EGG}}$ metric 28
2.6 EGG harmonic-domain analysis 29
2.6.1 DFT analysis 29
2.6.2 EGG waveform DFT components 29
2.6.3 Level and phase of the fundamental 30
2.6.4 Representation of phase differences 30
2.7 Sample entropy of EGG cycle data 30
2.7.1 Behaviour 31
2.7.2 Parameters 31
2.7.3 Computation 31
2.8 EGG clustering 32
2.8.1 Clustering dimensions 32
2.8.2 Resynthesis 32
2.8.3 Limitations 33
2.9 Phonation type clustering 33
2.10 Map smoothing 34
3 PART THREE - Using FonaDyn 35
3.1 Window layout 35
3.1.1 Main screen 35
3.1.2 Choice of Source 36
3.1.3 The ‘Moving EGG’ 37
3.1.4 The Voice Map 37
3.1.5 Saving and loading voice maps 40
3.1.6 The Plots 41
3.1.7 Clustering 42
3.1.8 EGG clusters 43
3.1.9 Phonation type clusters 45
3.1.10 The Settings 47
3.1.11 Comparing Voice Maps 49
3.1.12 Controlling the window layout 51

3.2 Recording 53
3.2.1 Overview 53
3.2.2 Recording environment 53
3.2.3 Normal setup 54
3.2.4 Check the EGG gain 54
3.2.5 Calibrate for SPL 54
3.2.6 Recording live signals 54
3.2.7 Re-recording during analysis 55
3.2.8 Recording additional signals in parallel 55
3.2.9 Using FonaDyn without an EGG device 56
3.3 SPL calibration 57
3.3.1 Choosing a method of calibration 57
3.3.2 A: Calibrator device that fits the microphone 60
3.3.3 B: Front microphone and SPL meter 61
3.3.4 C: Headset microphone with a front reference microphone 62
3.3.5 D: Headset microphone and dB meter only 63
3.3.6 Verifying a level calibration 63
3.3.7 Recording very loud voices 64
3.3.8 Examining live signals 65
3.4 Listening from maps 66
3.4.1 Background 66
3.4.2 Creating the Log file 67
3.4.3 Enabling listening from maps 67
3.4.4 Playing from the voice map 68
3.4.5 What is played? 68
3.5 Files 69
3.5.1 Files in general 69
3.5.2 Signal files 70
3.5.3 Log files 71
3.5.4 Voice map data 72
3.5.5 EGG cluster data 74
3.5.6 Phonation type cluster data 75
3.5.7 Saving and restoring settings 76
3.5.8 Script files 77
3.5.9 Context script files 79
3.5.10 Origin tracking 79
3.5.11 Diagnostic output files 79
3.6 Phonation Type Analysis 81
3.6.1 Pre-clustered sets of centroids 81
3.6.2 Parameters for phonation clustering 81
3.6.3 Initializing the clusters 81
3.6.4 Rearranging clusters 82
3.6.5 Clustering in multiple passes 83
3.6.6 Picking phonation types from the map 83
3.6.7 Manual setting of phonation type clusters 84
3.7 EGG Shape Analysis 84
3.7.1 De-noising the EGG signal 84
3.7.2 Pre-clustered sets of waveshape centroids 87
3.7.3 Parameters for EGG clustering 87
3.7.4 Initializing the clusters 88
3.7.5 Clustering in multiple passes 88
3.7.6 Rearranging clusters 89
3.7.7 Detecting specific EGG wave shapes 89
3.8 Analyzing multiple takes or multiple files 90
3.9 Using FonaDyn with RME audio interfaces 90
3.10 Known limitations 91
Acknowledgments 92
References 92
Contents in detail 95