Design and Implementation of A Data-Display MATLAB GUI for Cochlear Model Simulator

Yatong Bai

ybai70@gatech.edu

Georgia Institute of Technology

George W. Woodruff School of Mechanical Engineering

ME4699 Final Report

December 10th, 2018

Table of Contents

1 Background and Purpose

1.1 Purpose of Documentation

This software design report explains how to use the Data Display Graphical User Interface (GUI) for cochlear simulation and documents the implementation of the GUI. The targeted audiences are the group members in the research group and other researchers interested in simulation data obtained by the research group.

1.2 Purpose of the GUI

The research group performs simulations with physiologically-based models of cochlea in order to better understand and analyze the behaviors of cochlea. The simulations are done with MATLAB. The multi-physics models have mechanical, electrical and acoustic components and are based on Finite Element Method [1].

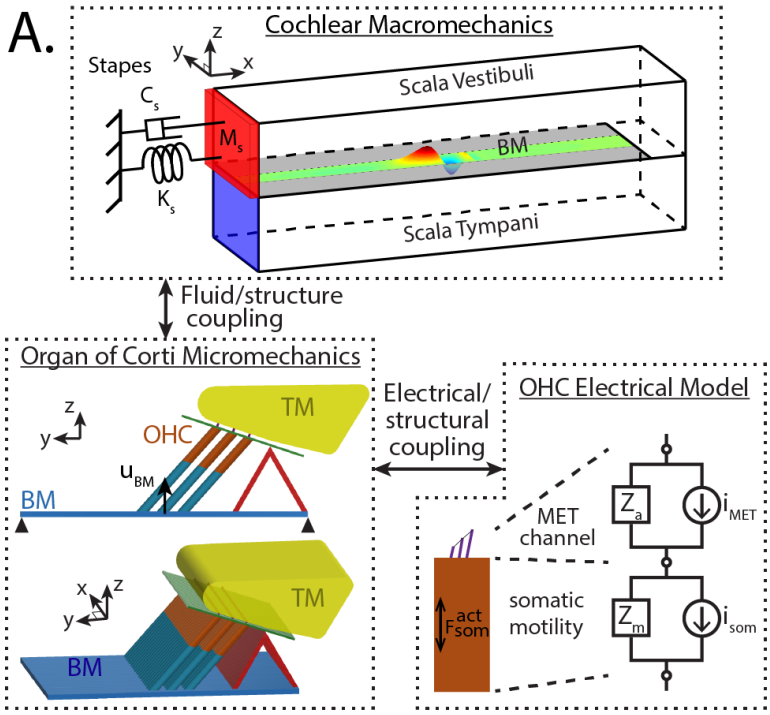


Figure 1. Structure of the cochlear model.

Currently, researchers have to create code in MATLAB in order to display simulation data. The data includes cochlear fluid pressure, outer hair cell forces, velocities, potentials and currents as functions of stimulus frequency, level, and the x location along the cochlea. Manually processing this complicated data into 3-D or 2-D arrays and plotting can be very tedious and requires extensive experience and the structure of the data.

The Data Display Graphical User Interface (GUI) is designed to make this process easier and faster. Its goals include facilitating researchers in this research group to display simulation data and sharing the data with other researchers. One of the crucial advantages of the GUI is that it is friendly to people with minimal knowledge about MATLAB programming or the structure of the simulation data. Also, since the GUI is planned to be uploaded to the website along with the simulation data, the size of the data is restricted. Currently the two data files that the GUI is using is approximately 3GB

in total. The GUI includes the functionality of plotting 1D curves of all simulation data with respective to position or frequencies at various levels, as well as the functionality of creating pressure colormaps.

Due to its relative low implementing difficulty, clear code structure, and good compatibility with other MATLAB components, MATLAB App Designer was chosen as the implementing tool of the GUI. The GUI is implemented and tested on both Windows and Mac using MATLAB R2017a. The user needs to have App Designer installed as MATLAB add-on to run the GUI. Tested screen resolutions range from 1600*900 to 3840*2160, with the tested aspect ratio of 16:9 and 16:10. Figures may not display correctly on lower-resolution screens.

2 Using the GUI

2.1 How to use the GUI

The GUI is divided into two sections: the left section controls general plotting options (Frequency, Level, Simulations, Plot vs, etc.) and graphical options (font sizes, x-axis limit, etc.); the right section has three tabs, which control parameters associated with pressure, velocity and potential data respectively.

To add or modify figures, check / uncheck corresponding checkboxes and modify parameters. Figures are automatically updated if "Update Figures Real Time" is checked, but the user can always click "Plot" to plot again.

The GUI creates at most 4 MATLAB figures: one for 3D pressure colormap (Fig. 1), one for 2D pressure colormap (Fig. 2), one for 1D pressure curves and velocity curves (Fig. 3), and one for potential, current and force curves (Fig. 4). Each of the figure has its default location and size. The number of subplots of Fig. 2, Fig. 3, and Fig. 4 varies according to the options that the user selects, and these three figures therefore changes their sizes as needed. After the user changes the size of Fig. 3 or Fig. 4, they will not automatically change sizes anymore (Fig. 2 will still change size according to the number of slices). After the user moves a figure, the figure will preserve its position when updated.

The GUI has the ability to plot multiple simulations. Each simulation can have different level-frequency combination and therefore different responses. 1D curves vs location / frequency of a particular data of all simulations are plotted in the same subfigure.

Colormaps handle multiple simulations in a different manner. The 3D colormap option is disabled for multiple-simulation plotting. 2D colormaps are plotted as subplots in a Fig. 2 in the sequence shown in Figure 4.

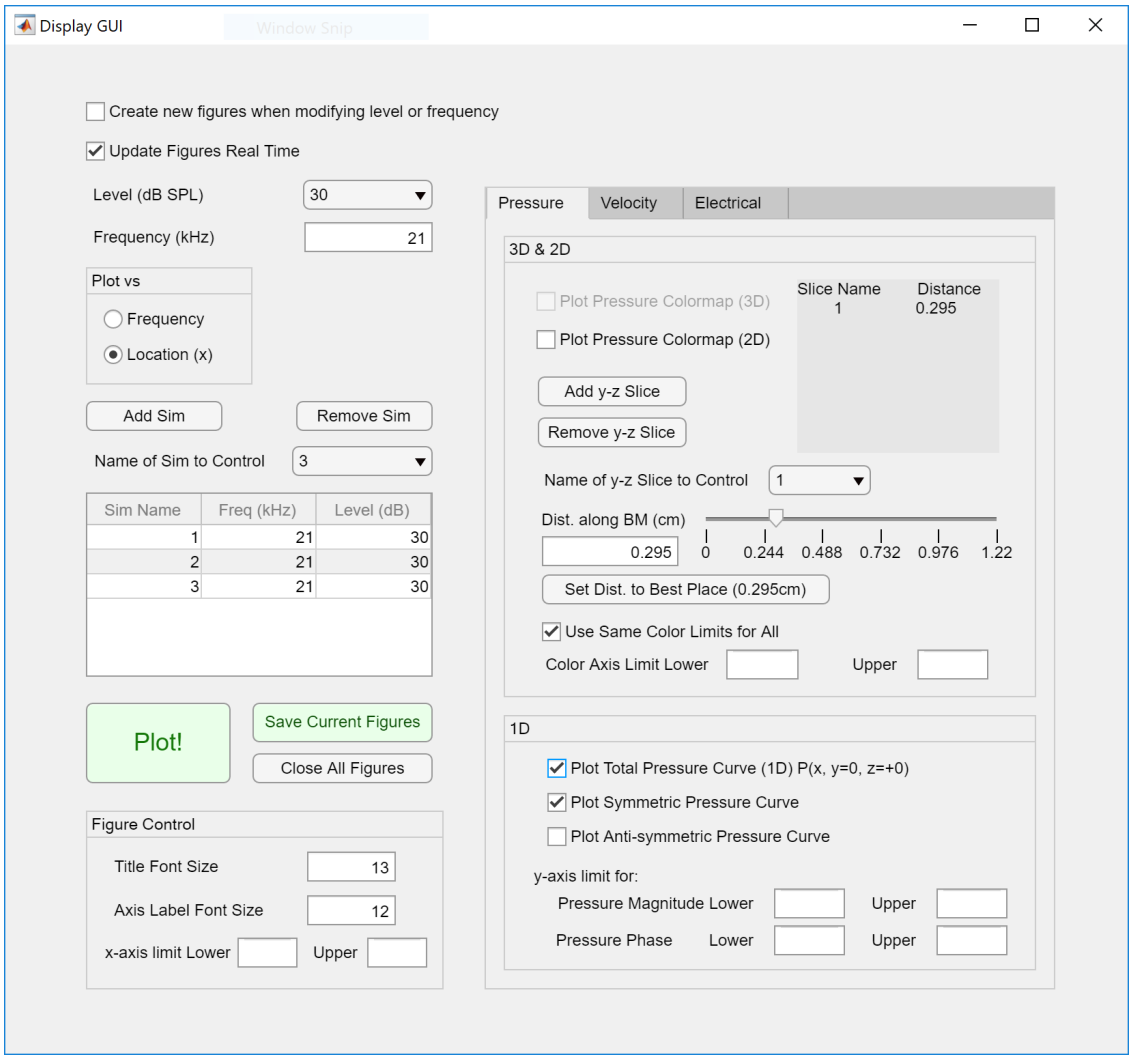


Figure 2. The GUI Layout.

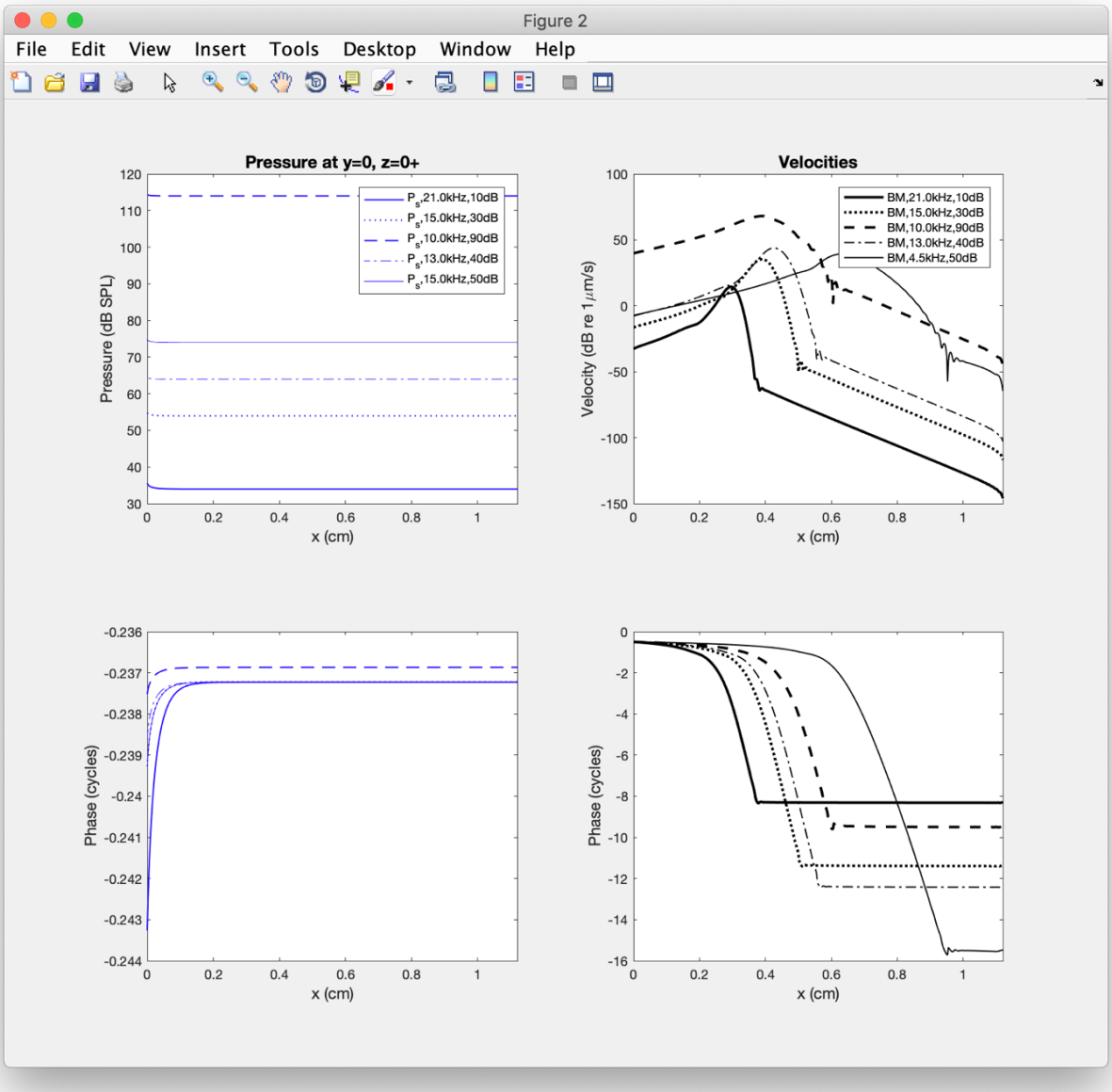


Figure 3. 1D curves for different simulations of one data are plotted on the same subplot.

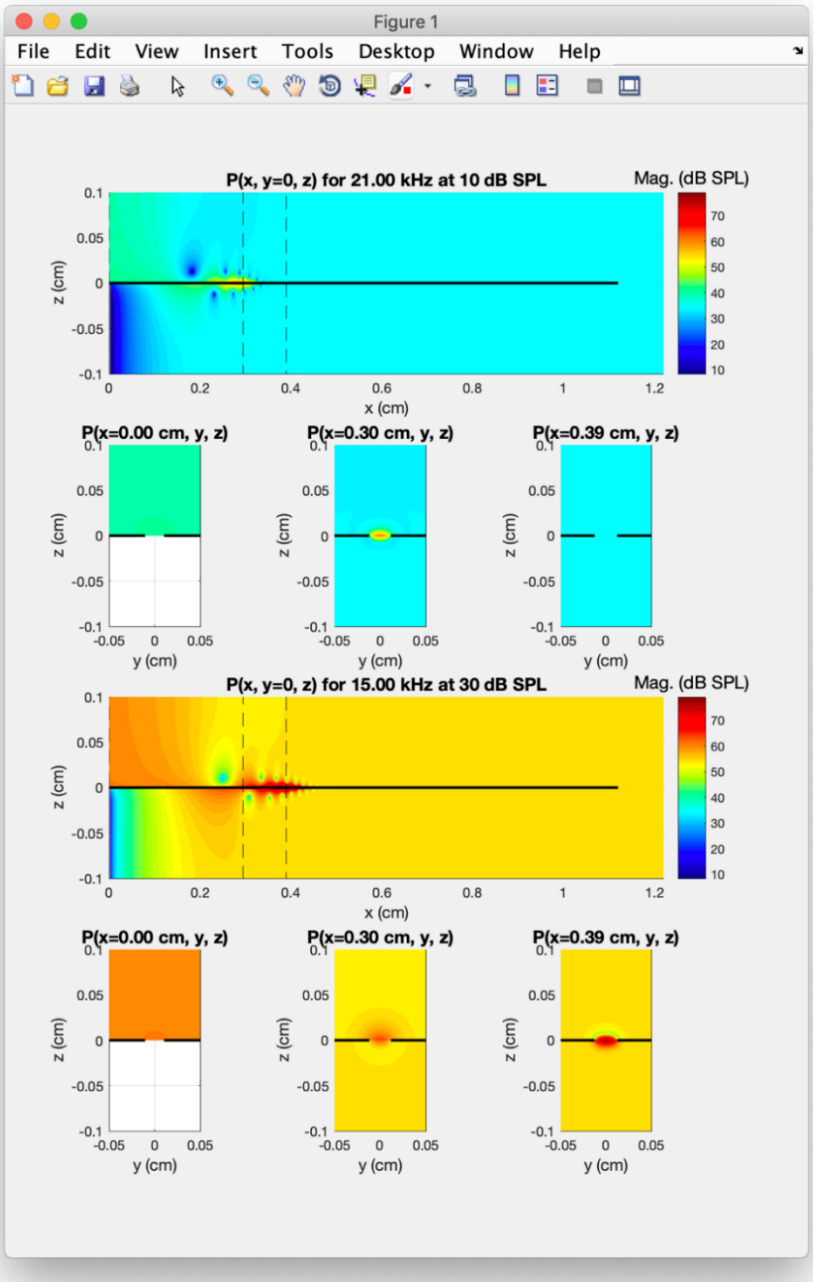


Figure 4. 2D colormaps of multiple simulations. The colormaps are arranged in the sequence of: x-z slice of simulation 1; all y-z slices of simulation 1; x-z slice of simulation 2; all y-z slices of simulation 2, etc. Y-z slices are sorted in increasing x location.

To add / remove a simulation, click the “Add Sim” / “Remove Sim” button, which are located in the left of the GUI. To modify a simulation, select the simulation to be modified using the “Name of Sim to Control” drop down menu. Then, use the Level dropdown menu and the Frequency edit field to modify it. It is also possible to modify the frequencies by directly editing the table. However, levels can not be edited using the table and can only be modified using the dropdown menu.

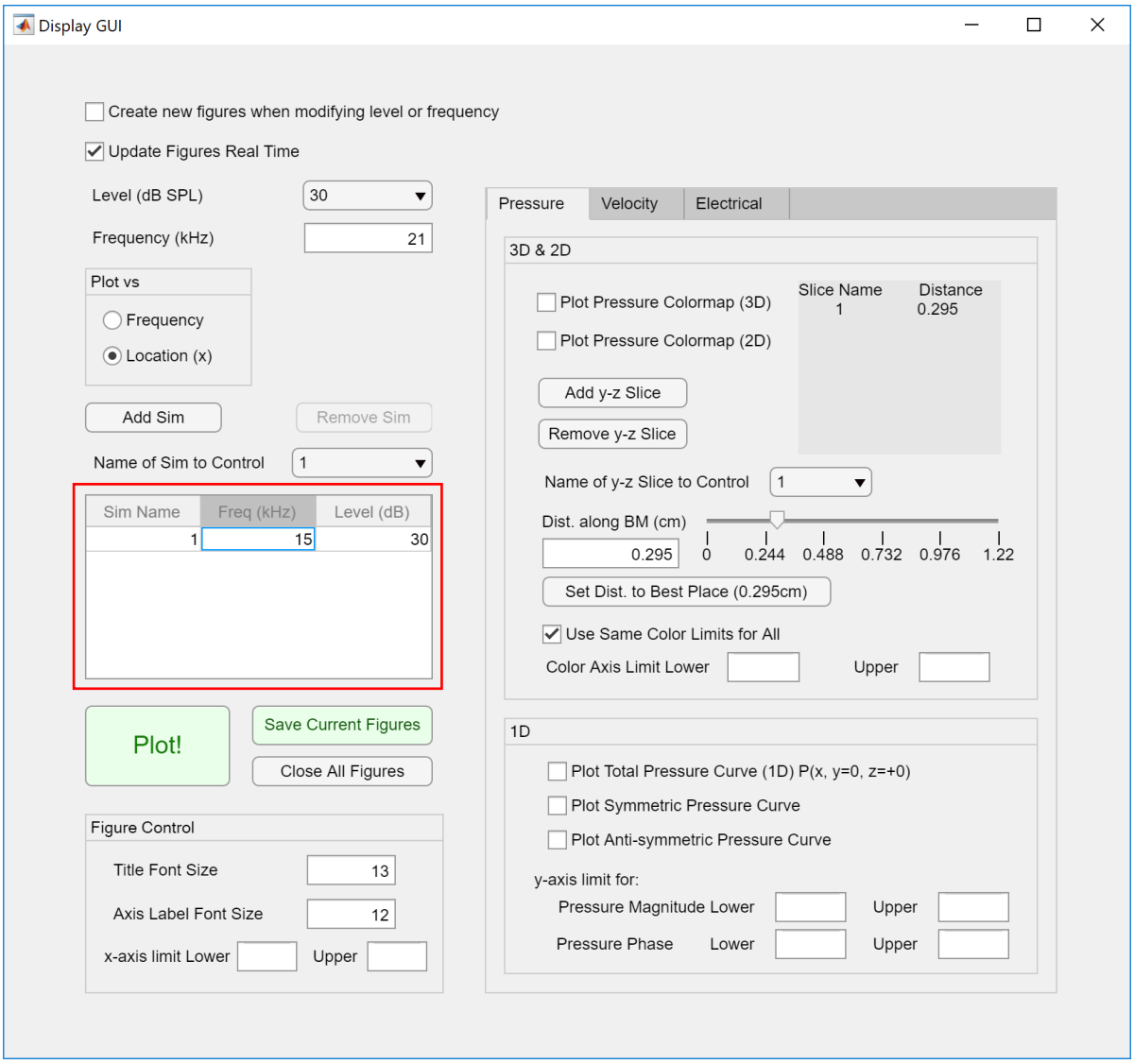


Figure 5. The "Freq" column of the simulation table is interactive.

Similarly, to add / remove an y-z slice in the colormaps, click the "Add y-z Slice" / "Remove y-z Slice" button. To modify the x location of a y-z slice, choose the slide to control and use either the slider or the edit field to control its location along the cochlea. The "Set Dist. to Best Place" button sets the location of the selected slice to the best place of that level-frequency combination. The slice control applies to all simulations in the 2D colormap. Although there is also a table of slices, it is not editable.

The GUI has the ability to plot 1D curves of pressure (P at y=0, z=0), velocities, potentials, forces and currents versus either x location along the cochlea, or stimulus frequency. To change the option of plotting versus x or frequency, use the "Plot vs" button group. When plotting versus frequency, the "x-axis Scale" button group controls whether the x axis is in logarithmic scale or linear scale. If "Plot vs Frequency" is chosen, the pressure colormaps will be disabled. Refer to section 2.3.1 and section 2.3.2 for detailed descriptions.

To save the figures, click "Save Current Figures". The GUI will create a new folder in the current directory and save the figures in both ".jpg" and ".fig" format in the new folder. Refer to section 2.3.11 for a detailed description.

"Close All Figures" button closes all figures. If the user clicks this button and plots the figures again, the figures will be defaulted to their original locations and sizes. This may also be helpful if figures are not displayed correctly.

2.2 GUI Performance

The time that is required to make or update plots varies depending on the number of simulations, and the number of plots the user chooses to generate. Pressure data, force data, velocity data, potential data and current data need to be loaded, processed and plotted individually. Therefore, when many of them are selected by the user, the GUI may respond slower, especially when there are more than one simulation.

To avoid internal instabilities, it is advisable to not let the GUI plot again while the previous plotting process hasn't finished. When "Update Figures Real Time" option is checked, modifying options and parameters on the GUI will automatically trigger a new plotting process. When the GUI is making plots, a green "Plotting…" label will be shown on the GUI. Trying to update the plots while they haven't been finished plotting may cause functions to be called again before it finishes execution. When this happens, the values stored in the global variables may not be correct, and may thus generate errors.

If multiple simulations and multiple plots are selected (the GUI responds slower) and the user desires to change multiple parameters (or select/unselect multiple options), it is advisable to uncheck "Update Figures Real Time" checkbox at the top-left of the GUI. This is to avoid updating the plots multiple times, which may be slow in this case. After changing all parameters, click the "Plot!" button to update the figures.

When encountering errors that are resulted from changing parameters while figures are still being modified, click "Close All Figures" and then click "Plot!" should clear the errors.

2.3 Plotting Options

2.3.1 Plot vs Location (x) / Frequency

The GUI has the functionality of plotting versus location (x along the cochlea) and plotting versus frequency. When plotting versus location, stimulus level and frequency are both fixed. X locations are the x-axis and the data are the y axis. Pressure colormaps can also be created. The 2D colormap plot consists x-z slices and y-z slices and is organized as shown in Figure 4. When plotting versus frequency, location (x) is fixed to the best place of the specified frequency.

2.3.2 Frequency

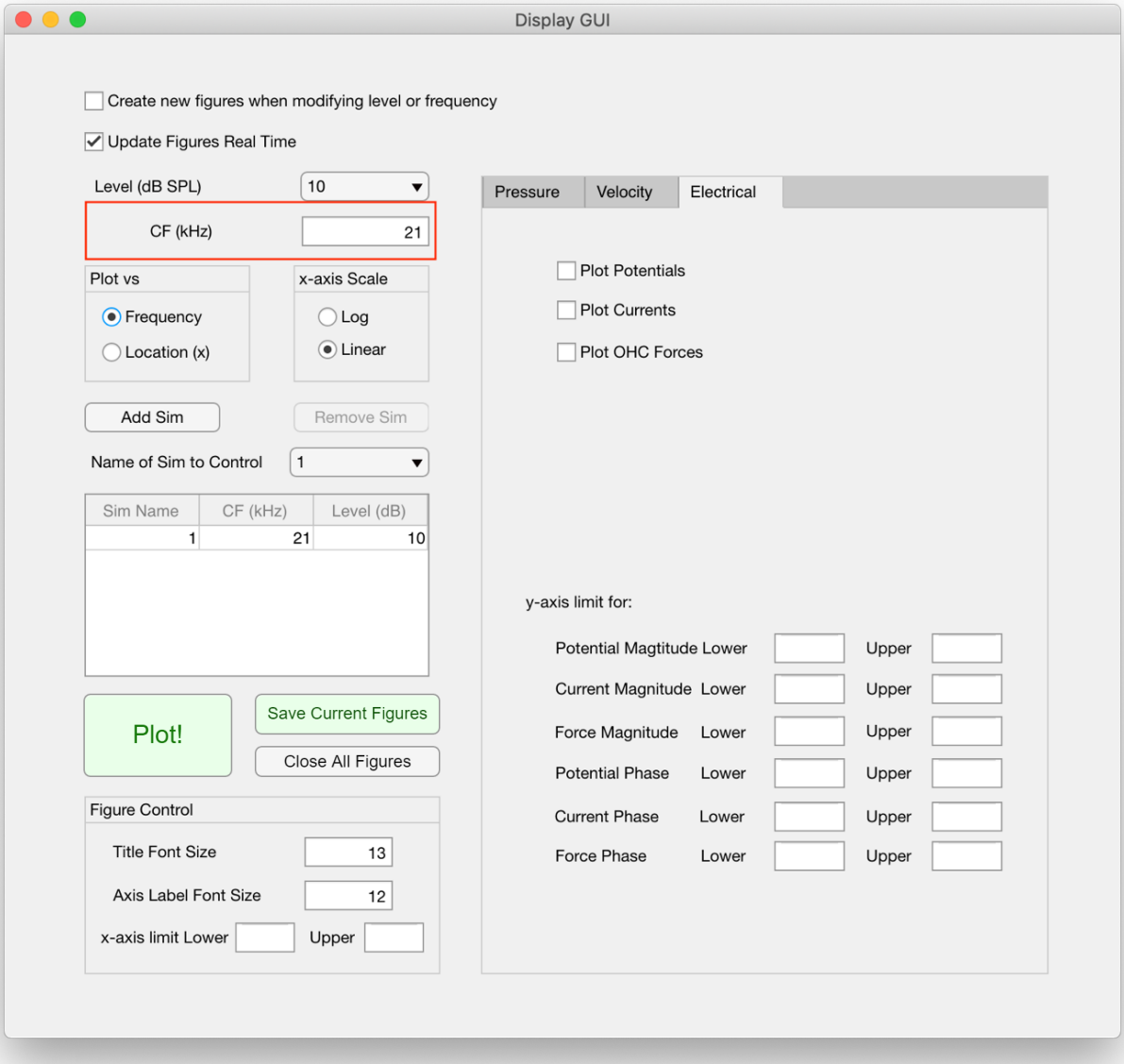


Figure 6. Frequency / CF edit field. Since "Plot vs Frequency" is selected, the edit field is now labeled "CF".

As described in section 2.3.1, user-entered frequencies are used differently in "Plot versus x" and "Plot versus frequency. Therefore, when "Plot with frequency" is selected, the "Frequency" edit field changes its label to "CF".

Pressure data, which is contained in the file "PureToneResults_FullP.mat", has a frequency resolution of 1000Hz, which is more coarse than other data, which are contained in "PureToneResults_2018BJ_AllButP.mat". These data has a frequency resolution of 100Hz. The reason for this difference is that velocity/potential data only depend on x location and frequency, while pressure depends on x, z, mode, and frequency. As a result, pressure data consumes significantly more space. The size of the velocity/potential data is about 73MB for 100Hz resolution, while that of pressure data at 100Hz resolution is more than 26GB. With 1000Hz resolution, the pressure data is now about 3GB. Since one of the design purposes of this GUI to share experiment results with other researchers, the research group plans to post the GUI

online along with the simulation data. Using a very fine resolution for pressure data makes it very difficult to be posted online. Therefore, pressure data has more coarse frequency resolution. Other data have finer frequency resolution for optimal precision.

When the frequency that the user enters does not exist in both data files, the closest available frequency will be chosen. A label will be displayed on the GUI to inform the user about the approximation, and the GUI will display the updated frequency value. If the specified frequency only exists in "AllButP", the GUI will use that frequency for everything but pressure, and look for the closest frequency that exists in "FullP". The informing label won't be shown, but the legends of the plots will show the actual frequency that is being plotted.

2.3.3 Plot symmetric / anti-symmetric pressure curve (1D Pressure)

If "Plot Symmetric Pressure Curve" option is checked, then symmetric component of pressure at y=0, z=0 will be plotted in the same graph of 1D total pressure. If "Plot Anti-symmetric Pressure Curve" option is checked, then anti-symmetric component of pressure at y=0, z=0 will be plotted in the same graph as well. The total pressure is the sum of symmetric and anti-symmetric components.

As in de Boer [?], the pressure is written as the sum of the symmetric and antisymmetric part:

$$P(x,y,z) = P_d(x,y,z) + P_c(x,y,z) \tag{5}$$

where P_d is the antisymmetric part and P_c is the symmetric part:

$$P_d(x,y,z) = \frac{1}{2}\Big[P(x,y,z) - P(x,y,-z)\Big] \tag{6}$$

$$P_c(x,y,z) = \frac{1}{2}\Big[P(x,y,z) + P(x,y,-z)\Big] \tag{7}$$

Figure 7. Relationship between symmetric, anti-symmetric, and total pressure.

2.3.4 Relative to Stapes (Velocities)

This option is located in the "Velocities" tab.

If this option is checked, the velocities will be divided by stapes velocity (which are loaded from the data file).

If this option is not checked, the velocities will be divided by reference velocity, which is 10^{-4} cm/s.

2.3.5 Update Figures Real-time

This checkbox is located at the top-left of the GUI.

If this option is checked, then every time when the user modifies one of the parameters or check/uncheck an option, the GUI re-load the data and replots every figure. This make the GUI easy to use when there are only one or two simulations and not many data selected. However, if the user wishes to modify several parameters or change several options at one time, checking this option makes the GUI slower to respond. Since the GUI may generate errors if the user changes options while the GUI is still plotting, it is advisable to uncheck this option in this case.

When this option is unchecked, the GUI will not plot or update the figures until "Plot!" button is clicked.

2.3.6 Create New Figures for New Plots

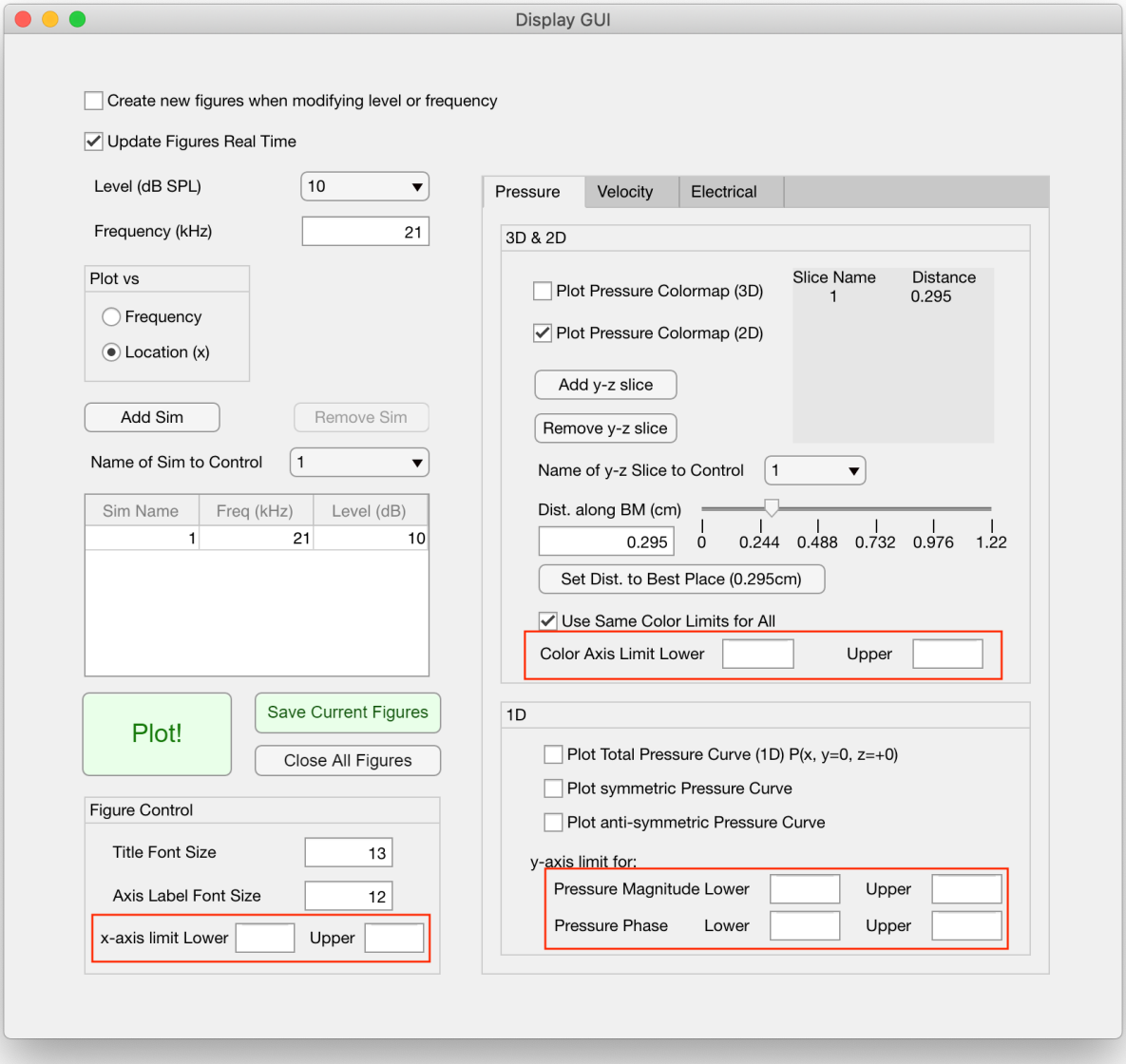
This checkbox is located at the top-left of the GUI.

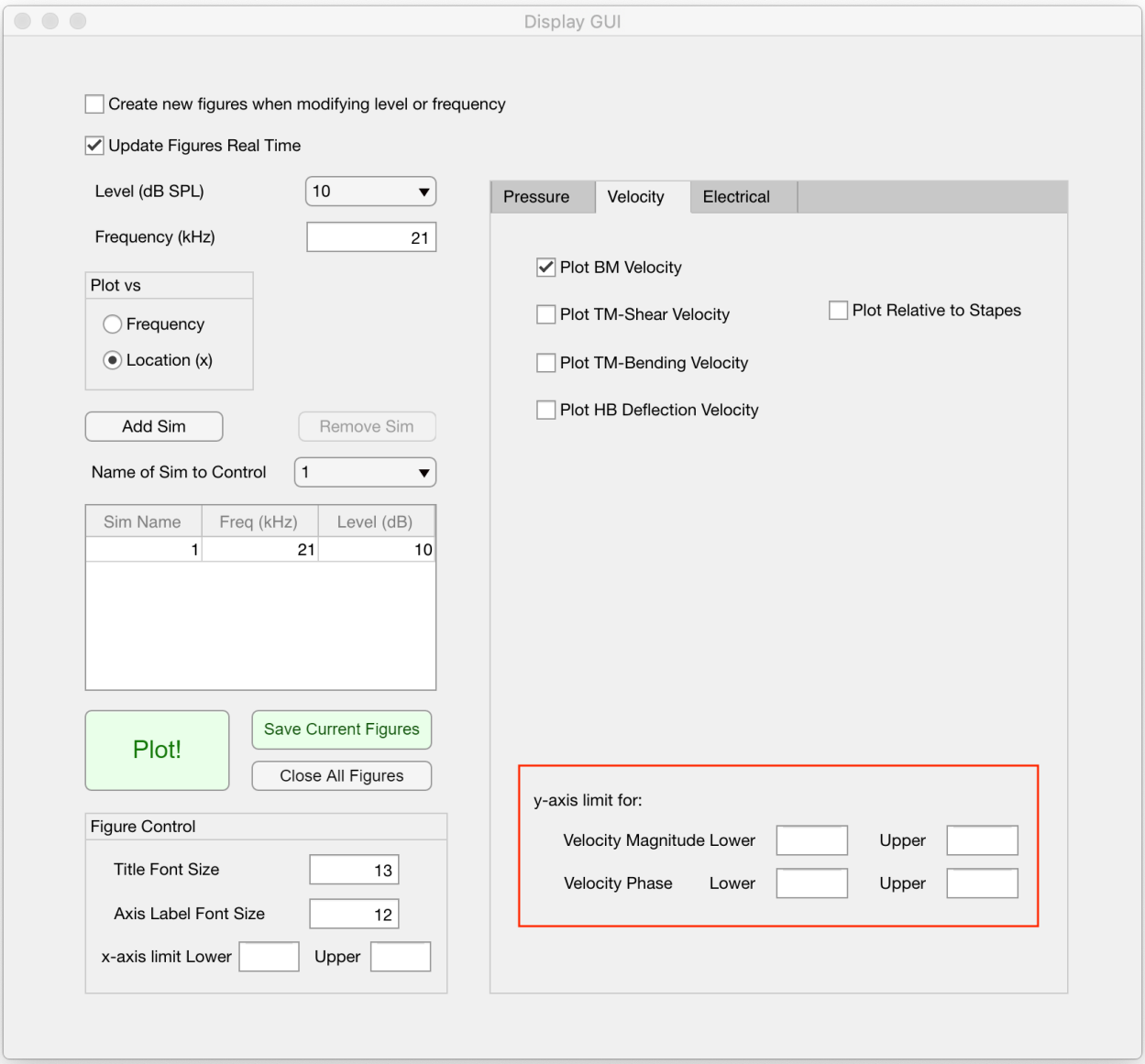
When this option is checked, the GUI opens new MATLAB figures every time it plots.

When this option is not checked, the GUI updates the plots on current open figures.

2.3.7 Axis Limits

Axis limits are adjustable for x-axis, y-axis (1D graphs) and color axis (colormaps). X-axis limit affects every 1D plot. However, y-axes adjustment for different data are separate. Y-axes adjustment for magnitude and phase are also separate. The locations of the adjustments in the GUI is shown in Figure 5. User-specified color axis limit applies to every colormap. However, whether different colormaps share default color axis limits depends on whether "Use Same Color Limits for All" is checked. Refer to section 2.3.10.




Display GUI
Create new figures when modifying level or frequency
Update Figures Real Time
Level (dB SPL)
10
Frequency (kHz)
21
Plot vs
Frequency
Location (x)
Add Sim
Remove Sim
Name of Sim to Control
1
Sim Name
Freq (kHz)
Level (dB)
1
21
10
Plot!
Save Current Figures
Close All Figures
Figure Control
Title Font Size
13
Axis Label Font Size
12
x-axis limit Lower
Upper
Pressure
Velocity
Electrical
Plot BM Velocity
Plot TM-Shear Velocity
Plot Relative to Stapes
Plot TM-Bending Velocity
Plot HB Deflection Velocity
y-axis limit for:
Velocity Magnitude Lower
Upper
Velocity Phase
Lower
Upper

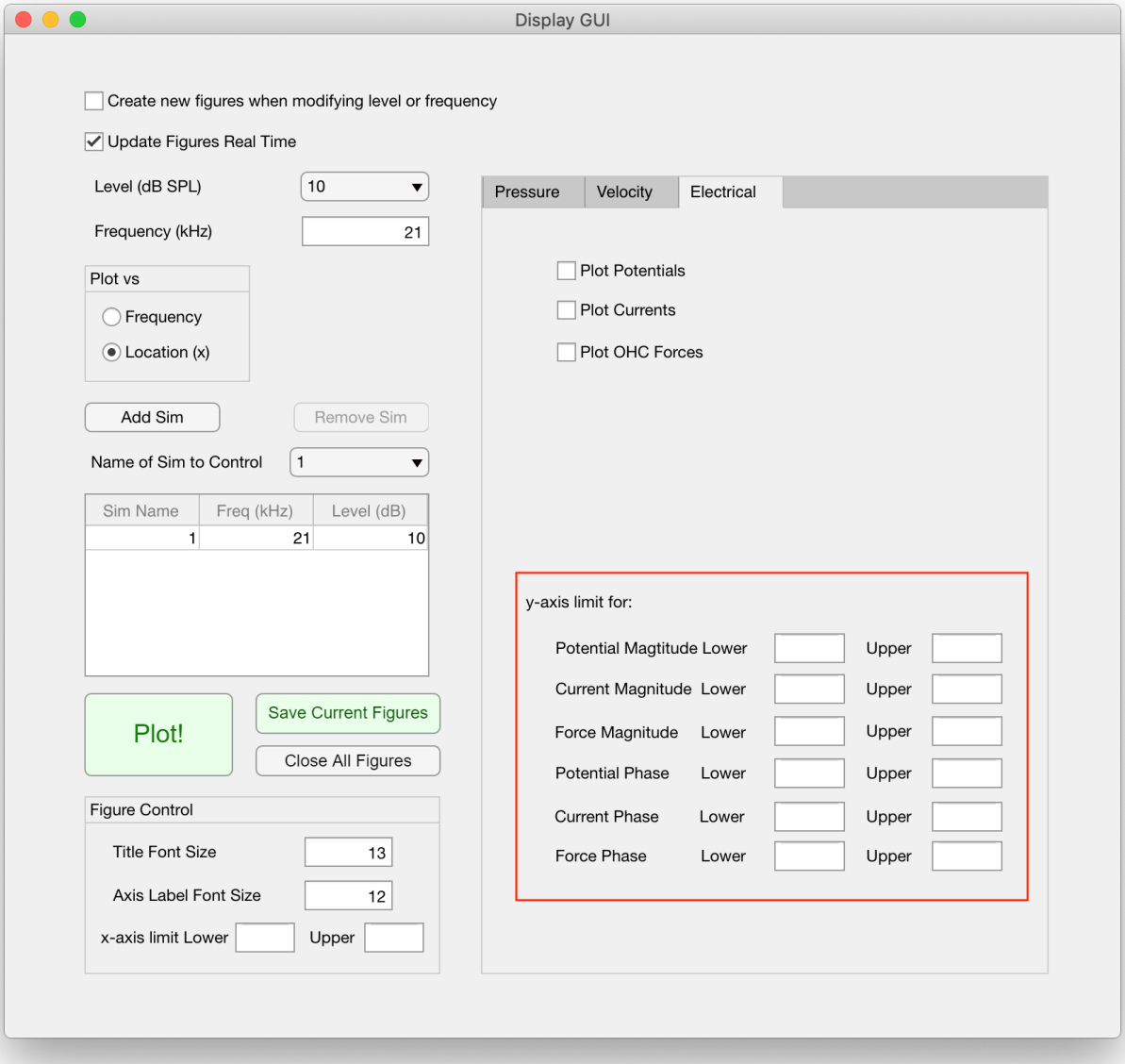


Figure 8. Locations of axis limit adjustments.

When both boundaries (lower / upper limit) are not specified, default axis limits will be used. Default x-axis limit is the length of the basilar membrane (1.12 cm for the gerbil); default y-axis limits and color axis limit depend on the range of the data. When only one boundary is specified, the other boundary will be the default value. When both boundaries are specified, they will both be used in the corresponding plot. To return to default values, please clear the edit fields.

If log / linear option is changed, x-axis limit will be cleared and default limit will be used.

2.3.8 Log / Linear

Log / Linear radio button group is only visible when "Plot vs Frequency" is selected. In the "Plot vs Frequency" case, the user can control whether to use log scale for the x-axis (frequency) or to use linear scale.

2.3.9 Font Size

This option is located at the bottom-left of the GUI.

The "Font Size" panel offers the adjustability of figure title font sizes and figure axes label font sizes. When the user enters a value, it applies to all figures, including the color bars.

2.3.10 Use Same Color Limits for All

This option is located in the "3D & 2D" panel in the "Pressure" tab.

This option controls whether all pressure colormaps share the same default color limits. When this option is checked, the default color limits are determined based on the total range of all the pressure data being plotted. When this option is unchecked, each simulation has its own color limits, based on the data range of that particular simulation.

However, this option doesn't affect user-specified color axis limits. For example, if the user is plotting two simulations with different F-L combination and sets the upper limit of the color axis, leaving the lower limit blank, and uncheck this option, then all the 2D colormaps (3D colormap will be disabled in this case) share the same upper color axis limit that the user sets. However, their lower limits are determined based on the data they are plotting, which are likely different.

3D colormap, 2D x-z slice, and 2D y-z slices of one simulation always share the same color axis limit, regardless whether this option is checked.

2.3.11 Save Current Figures

When the "Save Current Figures" button is clicked, all currently open figures created by the GUI will be saved.

The GUI creates a new folder in the current directory. The name of the folder is determined by the current date and time, as well as the frequency and level of each simulation. The format is: "`date(yyyymmdd)_time(hhmmss)_(sim1F%L)_(sim2F%L)_...`".

Then, the GUI saves all the figures in both `.fig` and `.png` formats in the new folder. The names of the files also follow the same format, but starting with the content of the figure (e. g. Pressure1D&Velocities).

3 GUI Implementation

3.1 Software Structure

The front-end objects (Edit Fields, Dropdown Menus, Tables, etc.) shown on the GUI are linked (by callback functions) with backend variables that controls what data need to be plotted. When the user enters values in the GUI, these variables are also updated. After these variables are ready, they are fed into the function "plotFigs.m". "plotFigs.m" loads the files that contain all the data, and assigns only the specified data to variables, which are arrays ready to be plotted. To optimize readability, most of the actual plotting are done in plot functions (e.g. plotBMVelo.m). Stimulus frequency and best place location are returned to the GUI from "plotFigs.m". Refer to section 3.3.1 for detailed description about "plotFigs.m".

3.2 Code Layout of GUIs in MATLAB App Designer

The code of a GUI starts with declarations of public properties. As described before, public properties consist front-end objects and user-defined "static" variables. After a public property is defined under "`properties (Access = public)`", it can be referenced anywhere in the GUI by using "`app.PROPERTY_NAME`" (e. g. "`app.createFigs`").

Within this part of the code, front-end objects are declared first. This is automatically done by the App Designer after the user creates a front-end object. App Designer protects this part from being edited by the user, avoiding the possibility of creating bugs.

Next, the variables are declared. They are declared in a similar way of declaring a front-end object.

The helper functions are defined and implemented as "methods", after properties are defined. They are defined under "`methods (Access = private)`", with the similar format as ordinary functions are defined. However, "app" is required to be an input argument. The actual format would be "`function FUNC_NAME (app, ARG1, ARG2, …)`".

The callback functions are declared as methods after the helper functions. The declarations of the callback functions are automatically done by App Designer once the user chooses to create a callback function. The declarations are in the format of "`function FIELD_NAMEValueChanged(app, event)`". The declarations of callback functions are not editable by the programmer. However, the user can change all the contents of callback functions.

Callback functions are automatically called when the user modifies the corresponding front-end object. However, one special callback function is the startup function, which is "`startupFcn(app)`". This function is called when the GUI starts.

In the "Code View" of the App Designer, initialization of the app is implemented after the callback functions. Implemented as function "`createComponents(app)`", it sets the default values, editabilities, visibilities and other fields of the front-end objects. This part of code is not editable by the programmer. If the programmer wishes to change the default fields of the front-end objects, he/she must modify them using the "Design View" and the properties window, instead of changing the code. This will be further explained in section 2.2.4.

In a nutshell, the layout of the code is:

- Public Properties Declaration
- Helper Functions
- Callback Functions
- Front End Objects Initialization (non-editable)

3.3 Key Variables and Functions

This section introduces some of the important functions and variables to ease possible modifications to the code.

Although several helper functions are written in the GUI file in App Designer, most of the data processing and plotting code are written outside the GUI. The reasons are:

- Debugging the code written in the GUI is not as easy;
- Cannot separate to different files;

- App Designer responses slower than MATLAB.

On the other hand, helper functions written in App Designer can access public properties. Therefore, functions that uses many public properties are written in the GUI, while other functions are written as normal MATLAB script files outside of the GUI.

3.3.1 plotFigs.m

"PlotFigs.m" is one of the functions written outside the GUI. Its main functionalities are loading data, processing pressure data, plotting pressure colormaps, and feeding other data (potentials, velocities, etc.) to their corresponding plotting functions.

The first part of "PlotFigs.m" loads pressure data from "PureToneResults_FullP.mat" to "`matObj`" and loads all other data from "PureToneResults_FullP.mat" to "`matObj2`". The data tables are subsequently loaded into "`T`" and "`T2`". The tables contain lists of available frequencies and levels. They are used to determine the corresponding row numbers of the selected frequencies and levels in the data arrays. Because the pressure data array has more dimensions and therefore consume much more storage space, the frequency resolution of pressure data is less fine then other data (refer to section 2.3.2). As mentioned in section 2.3.1, the function loads different data into the data variables depending whether the user chooses to plot vs frequency or vs location along the cochlea (x). However, the basic implementation is same for both cases: after loading tables, this function looks for the location that corresponds to the specified F and L, and then load data that are ready to be plotted into variables (`vBM`, `deltaPhiA`, `P_3D`, etc.). In order to handle multiple simulations, these variables are cell arrays and each cell consists data of one simulation.

```
matObj = matfile('PureToneResults_FullP.mat', 'Writable', false);
matObj2 = matfile('PureToneResults_2018BJ_AllButP.mat', 'Writable', false);
T = matObj.T; % table of data
T2 = matObj2.T;
```

Figure 9. Code snippet that loads `matObj` and `T`.

```
% Corresponding location of specified F and L
iRow{i, 2} = find(T2.F==F(i, 2) & T2.L==L(i));
```

Figure 10. Code snippet that looks for the row of data that corresponds to the specified stimulusfrequency and level.

```
vStapes{i}    = matObj2.vStapes(1,iRow{i, 2}); % cm/s, stapes velocity
vBM{i}        = matObj2.vBM(1:length(xMesh), iRow{i, 2}); % cm/s, BM velocity
vTMR{i}       = matObj2.vTMR(1:length(xMesh), iRow{i, 2});
vTMB{i}       = matObj2.vTMB(1:length(xMesh), iRow{i, 2});
deltaPhiA{i}  = matObj2.deltaPhiA(1:length(xMesh), iRow{i, 2});
deltaPhiB{i}  = matObj2.deltaPhiB(1:length(xMesh), iRow{i, 2});
P_box{i}      = matObj.P(1:length(xMesh_Total), 1:length(zMesh),...
    1:Nmodes, iRow{i, 1}); % dyne/cm^2
```

Figure 11. Code snippet that assign the data to variables. The variables are cell arrays to enable multiple-simulation plotting.

"PlotFigs.m" function also processes and plots pressure colormaps. The 3D data is first "squeezed" into 2D arrays. "Squeeze" function deletes redundant array dimensions that only has one entry. In this GUI, this function converts the

multi-dimensional data arrays into 2D arrays or 1D vectors. Then, the data is converted to dB scale. Next, the 3D colormap is made by using the built-in "slice" function, while 2D colormaps are created by using "Plot3" function.

3.3.2 createFigs

"`createFigs`" is a structure array that consists of all the plotting options associate with data processing. It contains a list of Boolean variables that flags whether certain data should be plotted. 3D colormap option can only be checked when there is only one simulation. In addition, there are several other Booleans that control what to plot, namely "`clearFigs`", "`StartUp`", "`withX`", "`log`".

```
app.createFigs.StartUp =          true;
app.createFigs.NewFigsForLF =     false;
app.createFigs.Pressure3D =       false;
app.createFigs.Pressure2D =       true;
app.createFigs.Pressure1D =       false;
app.createFigs.BMvelo =           true;
app.createFigs.TMshearVelo =      false;
app.createFigs.TMbendVelo =       false;
app.createFigs.RelativeToStapes = false;
app.createFigs.PlotSymAndAnti =   false;
app.createFigs.HBDeflect =        false;
app.createFigs.Potentials =       false;
app.createFigs.Currents =         false;
app.createFigs.OHCforces =        false;
app.createFigs.withX =            true;
app.createFigs.clearFigs =        true;
app.createFigs.log =              false;
```

Figure 12. A list of fields of "`createFigs`" and their default values.

The "`clearFigs`" field is a Boolean that specifies figure variables should be cleared. This is set to true "Close All Figures" is clicked, or at startup. The "`StartUp`" field is a Boolean that is only true when the GUI is starts. When this field is true, the GUI clear several necessary variables. The "`withX`" Boolean field controls plotting vs whether location or frequency. When this field is true, data are plotted vs location. The "`log`" Boolean field controls whether the x-axis is in linear scale or log scale. This is only effective when plotting vs frequency.

3.3.3 graphicOption

Similar to "`createFigs`", "`graphicOption`" is also a structure array. This variable controls graphic options such as axis limits (refer to section 2.3.7) and font sizes (refer to section 2.3.9).

```
app.graphicOption.Label =              12;
app.graphicOption.Title =              13;
app.graphicOption.xLim =               [NaN, NaN];
app.graphicOption.yLimP =              [NaN, NaN];
app.graphicOption.yLimF =              [NaN, NaN];
app.graphicOption.yLimI =              [NaN, NaN];
app.graphicOption.yLimV =              [NaN, NaN];
app.graphicOption.yLimPhi =            [NaN, NaN];
app.graphicOption.yLimPpha =           [NaN, NaN];
app.graphicOption.yLimFpha =           [NaN, NaN];
app.graphicOption.yLimIpha =           [NaN, NaN];
app.graphicOption.yLimVpha =           [NaN, NaN];
app.graphicOption.yLimPhipha =         [NaN, NaN];
app.graphicOption.cLim =               [NaN, NaN];
app.graphicOption.sameCL =             true;
```

Figure 13. A list of fields of "`graphicOption`" and their default values.

One of the fields of this variable is "sameCL". This variable controls if different 2D colormaps slices share the same default color limits.

3.3.4 getLineStyle.m

This function returns the line style and thickness of the certain simulation. The line styles of multiple simulations are by the sequence of "solid line", "dot line", "dashed line", "dot-dash line". The first four simulations (i=1:4) will use these line styles and share the same line thickness of 2. Starting from the first line, it restarts the cycle by using a solid line. However, the line thickness is reduced to half: simulation 5-8 will follow the same sequence and have thickness of 1. By the same token, simulation 9-12 follow the line style sequence and have 0.5 as thicknesses and so on.

4 Incomplete Functionalities and Known Bugs

4.1 Potential Improvements

4.1.1 Efficiency and Performance

Currently, the code is not written in a very efficient way. During the implementing process, functionality has always been prioritized over efficiency.

The code does not check whether a certain plot need to be updated. Instead, it loads all data and replot every subplots, even when the user changes parameters that only affect a few plots.

Currently, the code may take several seconds to finish plotting. The actual time depends on the amount of simulations and the amount of sublots the user is plotting. This performance issue may slow down the process if the user wishes to make multiple changes at one time and may even create errors if the user updates the GUI before it finishes plotting. Therefore, in that case, please try unselecting "Update Figures Real Time".

To solve this problem, it is advisable to modify the code by skipping unnecessary loading and plotting of data that have not been changed. Another approach to speed up the loading process to load the whole "AllButP" file into the memory in advance, so that the file doesn't have to be accessed every time.

4.1.2 Compatibility with Lower Screen Resolutions

The GUI is tested on screen resolutions ranging from 1600*900 to 3840*2160, with screen aspect ratio of 16:9 or 16:10. The windows may not display correctly on screens with lower resolutions.

Currently, figures of 1D curves are sized according to the screen resolution. 2D colormap has the protection of not exceeding the top edge of the screen. Whereas the sizes 3D colormap and the GUI itself are fixed.

Although most of the computer screens that researchers are using have resolutions within the supported range, many projectors have lower resolutions. Therefore, if the user plan to do a presentation, it is recommended to check the projector resolution in advance. If the resolution is too low, please consider using the saved figures or other ways to present.

4.1.3 Legends

Currently, the locations of the legends may not be optimal. Although the GUI tries to optimize the locations of the legends by setting them to "best" (`legend.Location='best'` makes MATLAB optimize the locations automatically). However, when multiple simulations are plotted and the legends become large, they may still block the curves. Even after the user moves the legends, they will return to their default places if the figures are re-plotted. Preserving the locations of the legends was considered difficult.

Moreover, some of the contents of the legends are not optimal. For example, potential "Δφ" is written as "Phi" in the GUI due to that Greek letters cannot be displayed correctly in some MATLAB versions.

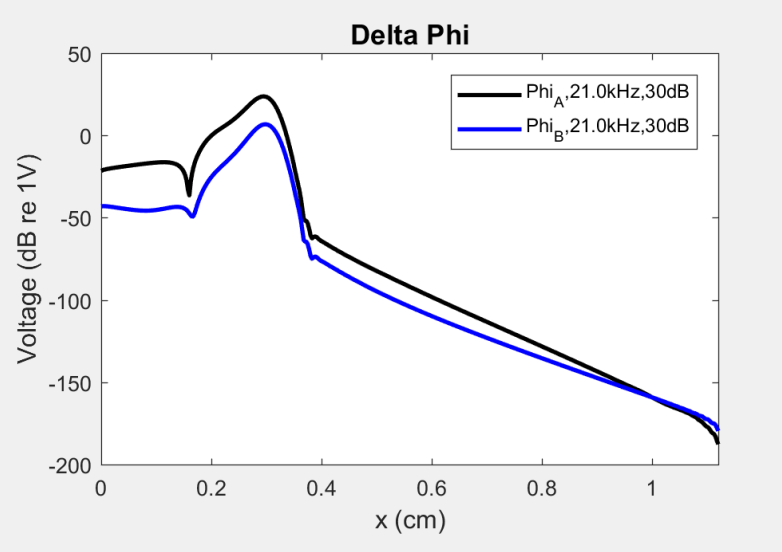


Figure 14. "Δφ" is written as "Phi" in the GUI legends, which are defaulted at the top-right of the figure.

4.2 Known Bugs

4.2.1 Unstable when Parameters are Updated Too Frequently

As mentioned in section 2.2, the GUI takes several seconds to plot the figures. The actual time varies depending on the number of simulations and the number of plots the user selected. If the user tries to modify the parameters or check/uncheck options before the GUI finishes plotting, Errors may be generated. Try clicking "Close All Figures" and clicking "Plot!" again to resolve the issue. Please refer to section 2.2 for detailed descriptions.

4.2.2 Y-z Slices of 2D Colormap

The y-z slices of the 2D colormap doesn't change their sizes when the user changes the overall size of Fig. 2. However, x-z slices adjust their sizes accordingly. Therefore, the visual effect of Fig. 2 is the best when its size is kept at the default value.

In addition, although very rarely, sometimes the y-z slices of the 2D colormap may not align with each other very well. The possibility for this to happen is generally very low. But this misalignment may happen more frequently if the performance of the computer is weaker. If this happens, click "Plot!" again and the slices should align perfectly.

References:

[1] Bowling and Meaud. "Forward and Reverse Waves: Modeling Distortion Products in the Intracochlear Fluid Pressure". Biophysical Journal, 2018.