ODEON APPLICATION NOTE
Plotting the results of a multipoint response calculation in Octave or Matlab

MAK – April 2024

1. Introduction

This note provides a short introduction to the program `PlotMultipointResponse.m`. The program can be executed in both Octave (tested in version 7.2.0) and Matlab (tested in version R2023b) and can be used to customize figures from data provided by ODEON's multipoint responses. By default, the figures produced by `PlotMultipointResponse.m` are similar to those produced in the "Simulated vs. Measured and Targets" tab of the multipoint response within ODEON, but can be customized by adjusting parameters within the code.

2. Downloading and opening the program

The program is contained in a ZIP file downloadable from odeon.dk > Learn > Application Notes, in the description for this application note. The ZIP file contains various .m files, however the only one that needs to be opened and run is `PlotMultipointResponse.m`, while the rest of the files are functions. The code within the main file is commented for clarity, and a description of the code is also given in the next section.

In order to run `PlotMultipointResponse.m` to produce the figures, the user will first need to calculate multipoint responses within ODEON and export the results to a .txt file. Please note that this must be done in the Multipoint response, by pressing Ctrl+A in the "Energy Parameters" tab, any of the "Energy parameter bars" tabs, or the "Simulated versus Measured and Targets" tab.

3. Program structure

`PlotMultipointResponse.m` consists of three sections. To use the program, it is only necessary for the user to specify the values of a few variables in the first section. However, if the user wants a different output than the default output of the program, he/she will have to modify the second and/or third section of the program. For that reason, descriptions of all three sections are given in the following.

1) *Define basic information*:
   Here the user should first specify the variables `PlotReceivers` and `PlotFrequencies`. These determine the receivers and frequencies (in Hz) for which the results of the multipoint calculation will be plotted. Since these variables are lists, they should be defined with square brackets as, say,

```
PlotReceivers   = [1, 5, 2];
PlotFrequencies = [125, 500, 1000];
```

Please note that the lists do not have to be in ascending order, although this is usually preferred.

Then, the user should specify the value of `GraphFraction`. To explain what this variable means, suppose that we want to plot the parameter `Param` and denote the difference between its maximum and minimum value by `a`. When plotting `Param`, the program chooses the y-axis to have the extent `b = a/GraphFraction`. In other words, `GraphFraction` is the size of the part of the y-axis that contains the graph of `Param` relative to the size of the displayed part of the y-axis. The role played by `GraphFraction` is easily revealed by trying different values, e.g., 0.1 and 0.9.

Next, the user should specify the location (`FileLocation`) and the name (`FileName`) of the .txt file containing the results of the multipoint response calculation. These variables are strings and should be defined using single quotation marks (‘ ’).

Finally, the user should specify the variable `ParameterName`. This variable is a string and holds the name of the parameter for which the results are plotted. In the program, the comment next to this variable lists all parameter names that can be chosen.

2) *Read data from file*:

In this section, the results of the multipoint response calculation are read from file. This requires no action from the user, but for modification purposes he/she should be aware of the following. The program reads the active sources (`SourceList`), the active receivers (`ReceiverList`), the list of frequencies for which values of the chosen parameter exist (`FrequencyList`), the simulated values of the chosen parameter (`ParamSim`), the measured values of the chosen parameter (`ParamMeas`), the difference between the simulated and measured values of the chosen parameter in units of JND (`ParamJND`) and, finally, a string (`ParamExists`) which is `'Yes'` if the chosen parameter exists and `'No'` if it does not. While `SourceList` and `ReceiverList` both are lists, `ParamSim`, `ParamMeas` and `ParamJND` are matrices with the number of rows being `length(ReceiverList)` and the number of columns being `length(FrequencyList)`. The results for the *n*th receiver in `ReceiverList` are stored in the *n*th row of these matrices. A `NaN`-entry in these matrices signifies that the value is not available.

Having read the data, the program goes on to determine the indices of the receivers and frequencies chosen for plotting. If, for example, the multipoint response has been calculated with receivers 1, 2, 4 and 5, such that `ReceiverList = [1, 2, 4, 5]`, and the user has chosen that results should be plotted for receivers 1, 5 and 2 (by choosing `PlotReceivers = [1, 5, 2]`), we will have `ReceiverIndices = [1, 4, 2]`. If `PlotReceivers` contains at least one receiver that is not contained in `ReceiverList`, the program sets the variable `ReceiversExist` to `'No'`, and to `'Yes'` otherwise. The situation is completely analogous for the frequencies.

At the end of this section, the program checks if `ParamExists`, `ReceiversExist` and `FrequenciesExist` are all `'Yes'`. If this is the case, the program proceeds to the third section. If at least one of the variables is `'No'`, the program terminates and no plots are produced.

3) *Plot data*:
   In this section, plots of the data are produced. The program distinguishes between STI parameters (STI, STI(Female) and STI(Male)) and all other parameters. The reason for this is that the STI parameters do not depend on frequency.
   For the non-STI parameters, the program produces three types of plots. First, it plots the chosen parameter as a function of frequency at the values 63 Hz, 125 Hz, 250 Hz, 500 Hz, 1000 Hz, 2000 Hz, 4000 Hz and 8000 Hz for each of the chosen receivers. Secondly, it plots the difference between the simulated and measured values of the chosen parameter in units of JNDs as a function of frequency at the values 63 Hz, 125 Hz, 250 Hz, 500 Hz, 1000 Hz, 2000 Hz, 4000 Hz and 8000 Hz for each of the chosen receivers. Thirdly, it plots the chosen parameter as a function of the receiver position for each of the chosen frequencies (i.e. the values in `PlotFrequencies`).
   For the STI parameters, the program plots the chosen parameter as a function of the receiver position.

4. Example output

As an example, we now consider the room “Auditorium21 at DTU”, which is available in the rooms-folder of ODEON. Choosing the early decay time (EDT) as the parameter to be plotted and setting

```
PlotReceivers   = [1, 2];
PlotFrequencies = [125, 500, 2000];
```

gives the following figures 1-5.

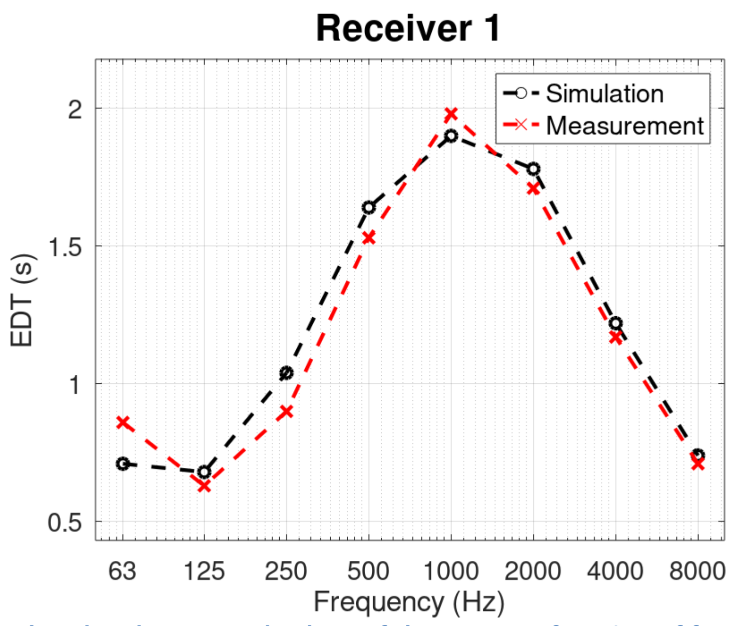

Figure 1 The simulated and measured values of the EDT as a function of frequency at receiver 1.

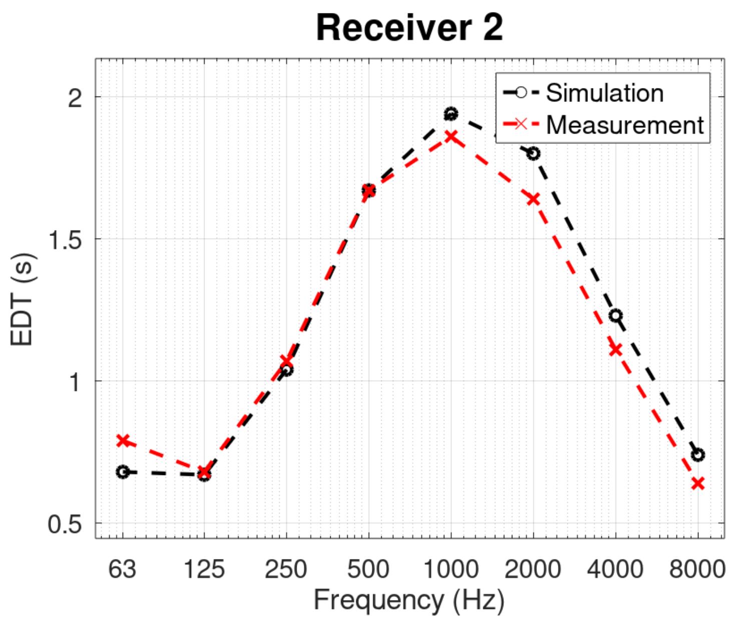

Figure 2 The simulated and measured values of the EDT as a function of frequency at receiver 2.

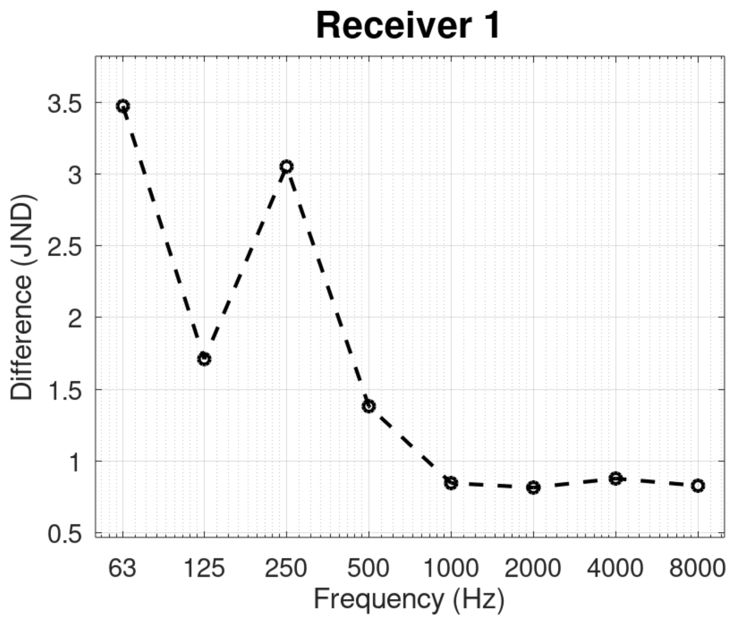

Figure 3 The difference between the simulated and measured values of the EDT in units of JNDs at receiver 1.

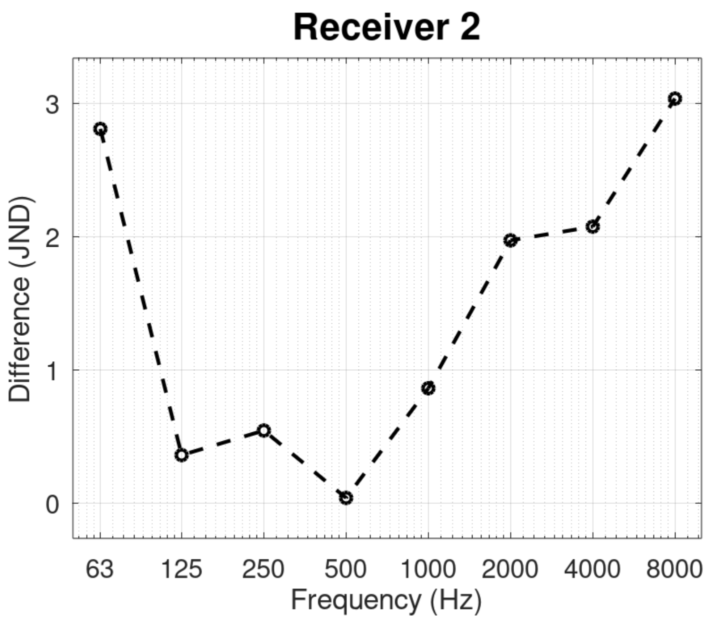

Figure 4 The difference between the simulated and measured values of the EDT in units of JNDs at receiver 2.

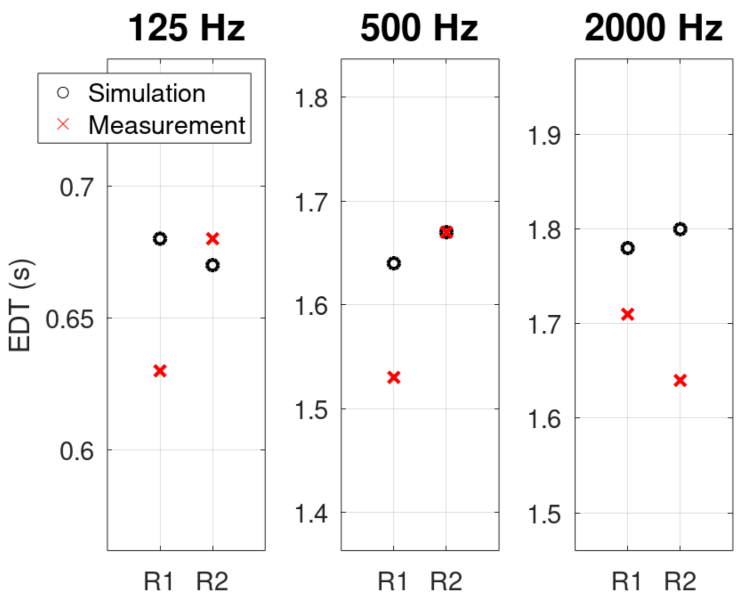

Figure 5 The simulated and measured values of the EDT vs the receiver position for different, fixed frequencies.