GOLDEN SAMPLE IDENTIFICATION USING CLIO AND SCILAB

by Daniele Ponteggia - dp@audiomatica.com

INTRODUCTION

The efficiency and quality of a manufacturing process can be kept under control through measurements on production items. Every production process is unique and reflects the history and philosophy of a company. This asks for flexible analysis tools tailored to the specific production needs. In this application note we will show some practical examples of statistical analysis on sets of measurements that can be useful in QC management, such as the identification of the golden sample. The analysis is usually done on a statistically relevant number of samples. This means a sufficiently large set of electrical or acoustical measurements, as an example frequency responses, to be analyzed. While it is possible to include such kind of tools into a QC measurement software, this approach may be not sufficiently flexible. We found out that it is possible to create powerful analysis tools by complementing a reliable and programmable QC system (CLIO) with a general purpose numerical computation software (Scilab).

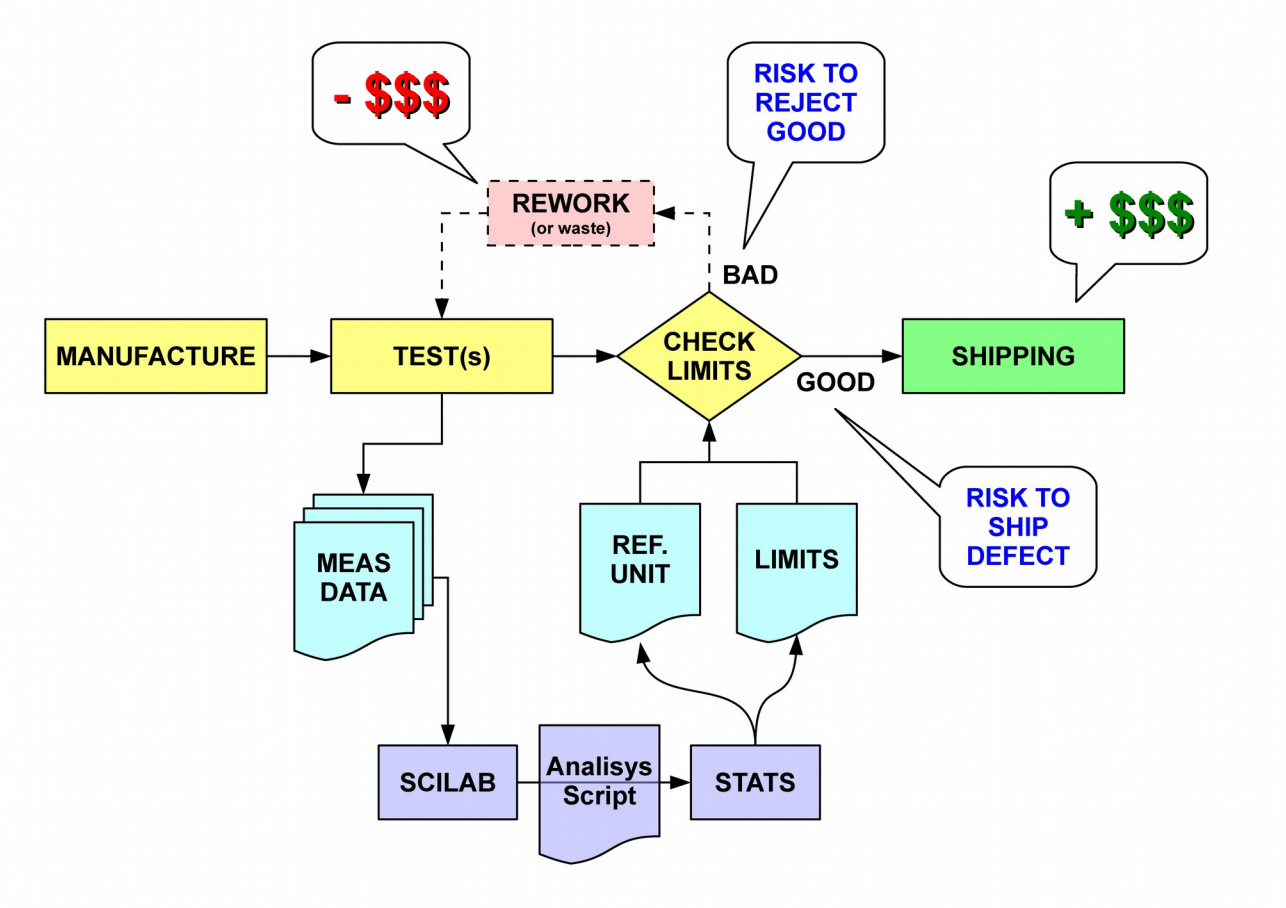

Figure 1: Typical production process with Quality Control

This application note is divided in three sections. First we will show how to use Scilab to import CLIO frequency responses saved in .txt file format, then two practical applications are presented in detail: how to search the golden sample in a set of measured items, and how to find if in a production there are items that match the golden sample properties and that can be used as substitutes for the actual golden sample.

Together with the application note, there are two compressed .zip folders (goldensample.zip and goldenmatch.zip) with the example Scilab scripts and sample data in order to experience with the scripting process. The files should available for download in the www.audiomatica.com Tech Support section web site.

SCILAB AS ANALYSIS TOOL

Scilab (www.scilab.org) is an open source numerical computation software. Scilab syntax is simple and similar to the "industry standard" Matlab software. The learning curve of the software is not steep and plenty of examples are available through Internet and on-line documentation. Scilab is very powerful in the analysis of sets of measurements: data is stored and manipulated in vector/matrices, statistical and processing functions are already present as high-level commands and a powerful plotting library is available.

Files created by the CLIO system are either in binary or text format. In the following examples we will use frequency response CLIO .txt files. Such kind of files are simple three column text with frequency, level and phase.

We show briefly a simple example of Scilab commands to load a CLIO .txt frequency response file and create a log-lin plot. Suppose that we start with a CLIO .txt file (which in this example is named "response.txt") that contains a frequency response with 64 frequency points, one on each row of the file with a triplet: frequency, sound pressure level and phase.

```
Freq [Hz]       dBSPL           Phase [Deg]
10.00           61.05           91.76
11.28           61.47           86.60
12.73           62.33           76.61
14.35           63.35           66.21
16.19           64.54           55.57

[…]

10841.56        95.10           -56.17
12229.84        94.69           15.37
13795.90        93.65           17.48
15562.49        95.66           -69.29
17555.30        96.60           77.42
19803.29        95.16           109.52
```

Figure 2: CLIO frequency response .txt file

Using the Scilab command `fscanfMat` it is possible to read the file into a 64x3 matrix. Please consult the Scilab help to get detailed information on Scilab commands syntax.

The command can be executed directly from the Scilab console (please check that the file is located in the current Scilab working directory):

```
-->M=fscanfMat('response.txt');
```

The command parse the "response.txt" file stripping the first line, which is the file text header, and stores the data in a matrix `M` of dimensions 64x3 (rows x columns). The first column of the `M` matrix contains the frequency points, the second column the sound pressure level and the third the phase.

Here is the matrix as imported by Scilab:

```
-->M
 M =

    10.          61.05     91.76
    11.28        61.47     86.6
    12.73        62.33     76.61
    14.35        63.35     66.21
    16.19        64.54     55.57
    [...]
    10841.56     95.1    - 56.17
    12229.84     94.69     15.37
    13795.9      93.65     17.48
    15562.49     95.66   - 69.29
    17555.3      96.6      77.42
    19803.29     95.16     109.52
```

The command loads into the M matrix the three columns of data from the .txt file. Frequency, level and phase are stored in the matrix columns and are accessible as vectors:

Frequency	$f_i \quad i=1\ldots64$	`M(1:64,1)`
Sound Pressure Level	$\left\|H\left(f_i\right)\right\| \quad i=1\ldots64$	`M(1:64,2)`
Phase	$arg\left\{H\left(f_i\right)\right\} \quad i=1\ldots64$	`M(1:64,3)`

Plotting the data with Scilab is also very simple. Here is a basic sequence of commands to plot the frequency response log-lin graphic in figure 3.

First of all the plot is created with the command:

```
-->plot2d(Mread(:,1),Mread(:,2),2);
```

Then some properties of the graphics are handled in order to get the desired result:

```
-->a=gca();
-->a.log_flags="lnn";
-->a.grid=[3,3];
-->a.box="on";
```

For a detailed description of Scilab graphics, please consult the Scilab documentation.

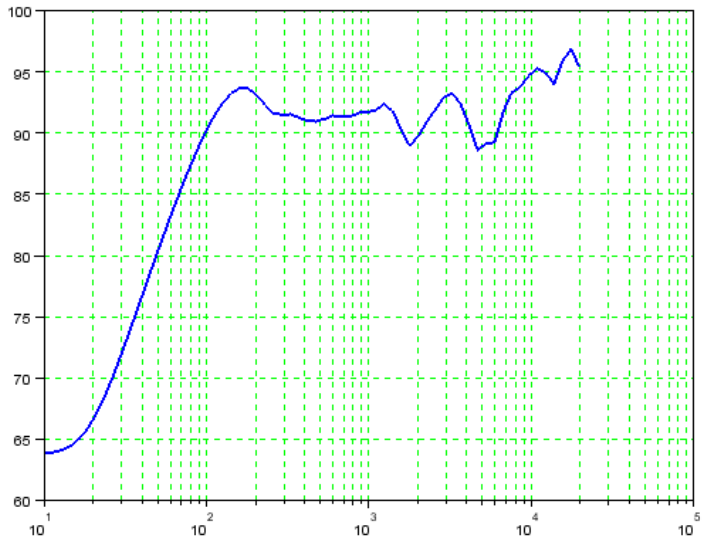

Figure 3: Frequency response plotted with Scilab

Besides the possibility to execute commands from the console, Scilab has an editor for creating and executing script commands. Therefore it is possible to create analysis procedures in a simple way, while having an high degree of customization.

Files in CLIO binary format can also be easily handled with Scilab, please check our Application Note 001 "CLIO 10 sinusoidal file structure with import examples in Scilab" which is available on our website www.audiomatica.com.

GOLDEN SAMPLE SEARCH

The search for the golden sample is a critical issue in the quality control process. The golden sample can be defined in product testing as:

"[...] a sample that has all test results in the middle of the nominal range".

Every manufacturing process can have a different strategy for this search, here we show only a possible solution using CLIO and Scilab.

In our example we deal with a production batch of microphones, we try to find the golden sample among this batch by searching the item which is nearest the average frequency response (magnitude only) in a specified range.

First the frequency response of every microphone of the batch is collected by placing it in front of a reference loudspeaker driver, taking great care of the repeatability of the positioning against the transducer[1]. Since every microphone capsule can have a different sensitivity, we calculate the sensitivity of each microphone in a given frequency range (500 Hz – 2 kHz) and then apply a correction in order to align the frequency response to a reference value. The entire set of collected responses is shown in figure 4.

1 The repeatability and reproducibility of a measurement setup it is a topic in itself. We would not enter here into the details, but it must be noted that if the repeatability of the measurement is not sufficiently granted, most of the statistical analysis that is described here is meaningless.

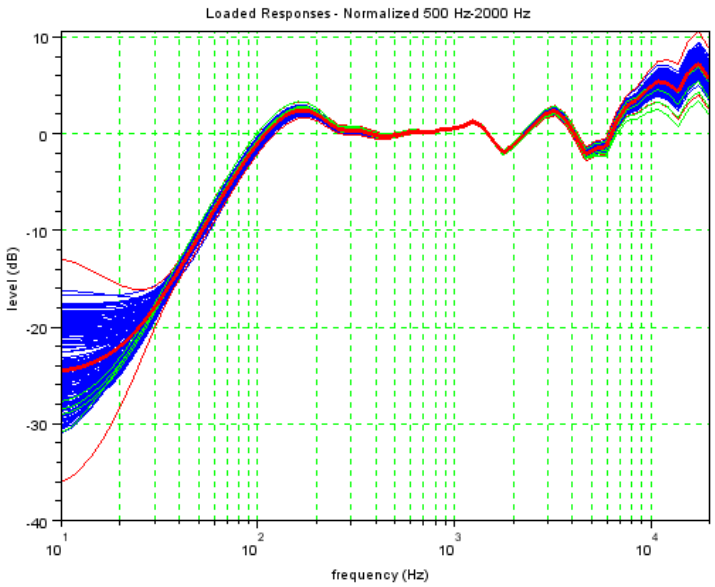

Figure 4: Original set of measured data

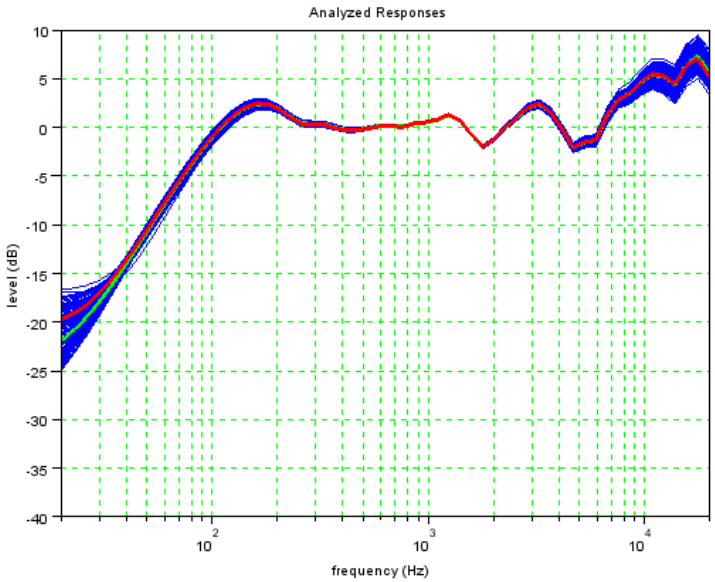

Figure 5: Reduced (purged) set of data

Next we calculate the average value and standard deviation of the magnitude frequency response. Our research policy requires, at this point, to discard items whose response exceed the average response (bold red curve) plus/minus two times the standard deviation (thin red curves).

We end up with a new (reduced) set of microphones where the items with a frequency response which deviates too much from the average (outliers) are discarded (Figure 5).

At this stage the average frequency response is calculated again on the new set

(shown in bold red curve) and a research of the item with minimum deviation (in a frequency range 200 Hz – 16 kHz) from the average is carried out.

The item that has less deviation from the average is the golden sample (bold green in figure 5, not easily visible because overlaps with the average red curve).

The golden sample search process can be implemented with a fairly simple Scilab script. We analyze here the code in depth.

The head of the script has only comments and a brief description of the script itself:

```
// goldensample.sce
//
// Load responses in CLIO .txt format from a folder
// and search for the Golden Sample
//
// fming,fmaxg - analysis frequency range
// normsens    - =1 sensitivity normalization
// fmins,fmaxs - sensitivity frequency range
// purgeout    - =1 purge outliers
// stdpurge    - std purge multiplier
```

Memory and already present graphic windows needed to be cleared:

```
clear; //clear memory
lines(0); //avoid console output halt

//closes all open graphic windows
wins=winsid();
for w=wins
    xdel(w);
end
```

Then the settings must be edited to meet the golden sample search needs:

```
//Analysis settings
fming=200;
fmaxg=16000;
normsens=1;
fmins=500;
fmaxs=2000;
purgeout=1;
stdpurge=3;
```

The response .txt files that are present in the current active Scilab folder[2] are loaded in memory:

```
//Load frequency responses from CLIO .txt files
S=dir('*.txt');
filelist=S(2);
for i=1:size(filelist,1) do
    measname(i)=filelist(i);
    Mread=fscanfMat(measname(i));
    measfreq(i,:)=Mread(:,1)';
```

2 If the script is loaded double clicking on it in the Windows Explorer, the active Scilab folder becomes automatically the script folder.

```
    measresp(i,:)=Mread(:,2)';
end
```

Please note that in this case only the first two columns of each imported file are used, phase data is discarded. The data is stored in two matrices `measfreq` and `measresp` with as many rows as the number of measurements m, and with a number of columns equal to the number of points n on each measurement[3].

Frequency	$f_i \quad i=1\ldots n$	`measfreq()`
Sound Pressure Level	$\left\|H_j(f_i)\right\| \quad i=1\ldots 64$	`M(1:64,2)`

If normalization is set the following code is executed. Thanks to the native matrix processing of Scilab the operation requires a very simple code.

```
//if normsens then apply normalization
if normsens==1 then
    [errval,fmini]=min(abs(measfreq(1,:)-fmins));
    [errval,fmaxi]=min(abs(measfreq(1,:)-fmaxs));
    meassens=mean(measresp(:,fmini:fmaxi),2);
    measresp=measresp-meassens.*.ones(1,size(measresp,2));
end
```

`fmini` and `fmaxi` are the indexes of the frequency vector `measfreq` bounded by the `fmins` and `fmaxs` sensitivity range. `meassens` is a m size vector with the calculated sensitivities S_j of each measured response `measresp` as the average sound pressure level in the sensitivity range.

$$S_j = \frac{\sum_{i=f_{mini}}^{i=f_{maxi}} \left|H_j(f_i)\right|}{f_{maxi} - f_{mini} + 1} \qquad \forall j$$

Once the sensitivities are calculated, the responses are normalized.

The following code plot the set of loaded responses:

```
//plot frequency response set (after normalization)
f=scf();
drawlater();
for i=1:size(measname,1) do
    plot2d(measfreq(i,:),measresp(i,:),2);
end
a=gca();
a.box="on";
atmax=ceil(max(measresp)/10).*10;
a.data_bounds=[20,atmax-50;20000,atmax];
a.tight_limits="on";
a.log_flags="lnn";
```

3 Due to this simplified approach, every measurement .txt file that is loaded must have the same number of points. The script do not provide any check on dimensions, failing to provide .txt files with a consistent number of frequency point will lead to errors.

```
a.grid=[3,3];
if normsens==1 then
    titlestring="Loaded Responses - Normalized "+string(fmins)
+"Hz-"+string(fmaxs)+" Hz";
else
    titlestring="Loaded Responses";
end
title(titlestring);
xlabel("frequency (Hz)");
ylabel("level (dB)");
drawnow();
```

The following script computes the statistics of the data set:

```
//compute statistics
measmean=mean(measresp,1);
measdstd=stdev(measresp,1);
```

Frequency limits of the golden sample search are converted to indexes of the `measfreq` arrays:

```
//golden sample search frequency range
[errval,fmini]=min(abs(measfreq(1,:)-fming));
[errval,fmaxi]=min(abs(measfreq(1,:)-fmaxg));
```

If purgeout is selected, the responses outside the average plus/minus given times of the standard deviation are removed from the data set. The curves of the outliers are plotted in green and the measurement names are displayed on the Scilab console:

```
//if purgeout then find and purge outliers
drawlater();
measOK=[];
if purgeout==1 then
    plusstd=measmean+stdpurge.*measdstd;
    minustd=measmean-stdpurge.*measdstd;
    plot2d(measfreq(1,:),plusstd,3);
    plot2d(measfreq(1,:),minustd,3);
    j=1;
    for i=1:size(measname,1) do
          if (sum(measresp(i,fmini:fmaxi)>plusstd(fmini:fmaxi))+sum(meas-
resp(i,fmini:fmaxi)<minustd(fmini:fm
axi)))==0 then
            measOK(j)=i;
            j=j+1;
        else
            plot2d(measfreq(1,:),measresp(i,:),1);
            disp(measname(i)+" DISCARDED")
        end
    end
else
    measOK=(1:size(measfreq,1))';
end
drawnow();
```

```
if measOK==[] then
    disp("Purge excess!!!");
    abort;
end
```

The purge code returns a vector `measOK` which contains the indexes of the "good" curves, the vector will be used later on to extract a reduced set of measurements.

Finally the average value is plot in bold red, this curve is plotted as last to stay on top of the other curves (see figure 4):

```
//plot statistics
plot2d(measfreq(1,:),measmean,3);
e=gce();
e.children(1).thickness=3
```

The statistics are calculated again on the reduced data set:

```
//recompute statistics
measmean=mean(measresp(measOK,:),1);
measdstd=stdev(measresp(measOK,:),1);
```

And finally the search of the golden sample is carried out:

```
//golden sample research
//compute difference between average and whole set of measurements
measerro=measresp(:,:)-ones(size(measresp(:,:),1),1).*.measmean;
//compute sum of squared errors and find minimum between valid items
(golden)
[errval,igolden]=min(sum(measerro(measOK,fmini:fmaxi).^2),2);
igolden=measOK(igolden);
//show golden
disp(measname(igolden)+" GOLDEN SAMPLE");
```

For each item and for each frequency point the algorithm computes the difference between measured data and the set average and store it in the `measerro` matrix. Then a squared sum of the `measerro` matrix along the frequency points dimension is carried out, this results in a vector with dimension equal to the number of items of the set.

$$ErrVal_j = \sum_{i=f_{mini}}^{i=f_{maxi}} \left[\left| H_j(f_i) \right| - H_{mean}(f_i) \right]^2 \qquad \forall\, j \in measOK$$

A research of the index with the minimum of this vector returns directly the index `igolden` of the golden sample, and the name of the item measurement is displayed in the Scilab console.

The response of the golden sample and the data set is plot with the following commands:

```
//plot frequency response set with golden sample
f=scf();
drawlater();
for i=1:size(measOK,1) do
```

```
    plot2d(measfreq(1,:),measresp(measOK(i),:),2);
end
plot2d(measfreq(1,:),measmean,3);
e=gce();
e.children(1).thickness=3
plot2d(measfreq(igolden,:),measresp(igolden,:),5);
e=gce();
e.children(1).thickness=3
a=gca();
a.box="on";
atmax=ceil(max(measresp)/10).*10;
a.data_bounds=[20,atmax-50;20000,atmax];
a.tight_limits="on";
a.log_flags="lnn";
a.grid=[3,3];
title("Analyzed Responses");
xlabel("frequency (Hz)");
ylabel("level (dB)");
drawnow();
```

The error between the items of the data set and the item choose as the golden sample can be easily calculated and plotted:

```
//compute error from golden
golderro=measresp(:,:)-ones(size(measresp(:,:),1),1).*.measresp(igolden,:
);
//plot error against golden sample
f=scf();
drawlater();
for i=1:size(measOK,1) do
    plot2d(measfreq(1,:),golderro(measOK(i),:),2);
end
plot2d(measfreq(igolden,:),golderro(igolden,:),5);
e=gce();
e.children(1).thickness=3
a=gca();
a.box="on";
atmax=ceil(max(golderro)/5).*5;
a.data_bounds=[20,-atmax;20000,atmax];
a.tight_limits="on";
a.log_flags="lnn";
a.grid=[3,3];
title("Error Against Golden Sample");
xlabel("frequency (Hz)");
ylabel("level (dB)");
drawnow();
```

An example of this latest plot is shown in figure 6. The relative error of each item response against the golden sample can be useful to identify limit curves in a QC procedure.

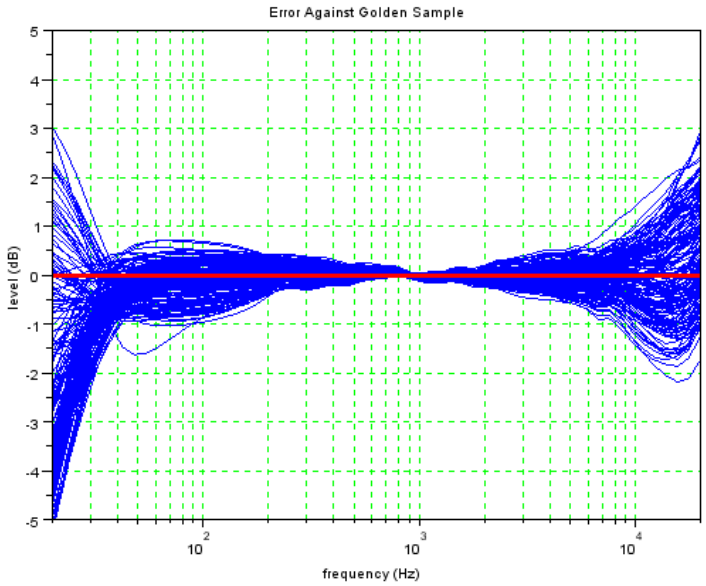

Figure 6: Relative error against golden sample

GOLDEN SAMPLE MATCH

One of the problems of using a physical golden sample is the conservation of the item. Due to aging or any accidental damage the golden sample may become unusable. Identification of possible substitutes for the golden sample during production it is a very interesting feature. Using Scilab it is possible to create a script which seeks for golden sample candidates in a production batch.

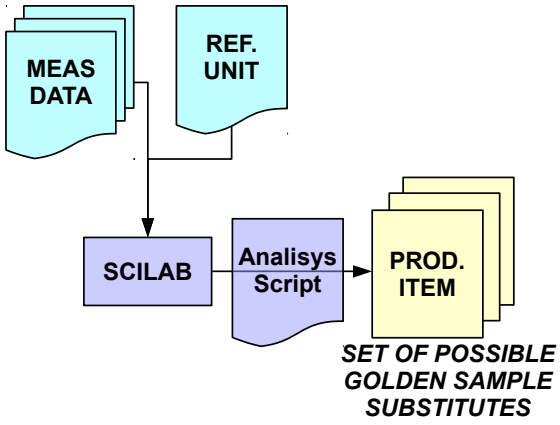

Figure 7: Golden Sample Match Process

The golden sample match process act as in figure 7: a batch of items is tested during the QC execution, then the measured data is compared against the reference (golden sample) unit.

The Scilab script calculates the error between each sample of the batch and the reference, then finds the candidates according to a specified criteria. In this example the criteria is the sum of the absolute error against the response of the reference, normalized by the number of frequency points.

$$SSnorm_j = \frac{\sum_{i=f_{mini}}^{i=f_{maxi}} \sqrt{\left[\left|H_j(f_i)\right| - H_{GS}(f_i)\right]^2}}{fmaxi - fmini + 1}$$

As in the previous example the head of the script has only comments and a brief description of the script itself:

```
// goldenmatch.sce
//
// Load responses in CLIO .txt format from a folder
// and search for substitutes to the Golden Sample
// response stored in a specified .txt file
//
// fming,fmaxg - analysis frequency range
// normsens    - =1 sensitivity normalization
// fmins,fmaxs - sensitivity frequency range
// goldthres   - cumulative error threshold
// goldfile    - golden sample response text file
```

Memory and graphics cleanup:

```
clear;    //clear memory
lines(0); //avoid console output halt

//closes all open graphic windows
wins=winsid();
for w=wins
    xdel(w);
end
```

The following settings should be edited, description of the variables is in the script head.

```
//Analysis settings
fming=200;
fmaxg=16000;
normsens=1;
fmins=500;
fmaxs=2000;

//cumulative error threshold
goldthres=0.075;

//golden sample file setting
goldfile="golden_sample.txt";
```

The golden sample response is loaded and if requested aligned to sensitivity:

```
Mread=fscanfMat(goldfile);
goldfreq=Mread(:,1)';
goldresp=Mread(:,2)';

//if normsens then apply normalization
if normsens==1 then
```

```
    [errval,fmini]=min(abs(goldfreq-fmins));
    [errval,fmaxi]=min(abs(goldfreq-fmaxs));
    goldsens=mean(goldresp(fmini:fmaxi),2);  //average on SPL
    goldresp=goldresp-goldsens.*.ones(1,size(goldresp,2));
end
```

Response of the other items on the same folder are loaded and normalized to sensitivity:

```
//load other responses and apply normalization if requested
S=dir('*.txt');
filelist=S(2);
i=1;
for j=1:size(filelist,1) do
    if filelist(j)<>goldfile then
        measname(i)=filelist(i);
        Mread=fscanfMat(measname(i));
        measfreq(i,:)=Mread(:,1)';
        measresp(i,:)=Mread(:,2)';
        i=i+1;
    end
end

if normsens==1 then
    [errval,fmini]=min(abs(measfreq(1,:)-fmins));
    [errval,fmaxi]=min(abs(measfreq(1,:)-fmaxs));
    meassens=mean(measresp(:,fmini:fmaxi),2);
    measresp=measresp-meassens.*.ones(1,size(measresp,2));
end
```

Errors against the golden sample are computed and plotted (Figure 8):

```
//errors against golden
golderro=measresp-goldresp.*.ones(size(measresp,1),1);

//create plot
f=scf();
drawlater();
for i=1:size(measresp,1) do
    plot2d(measfreq(1,:),golderro(i,:),2);
end
a=gca();
a.box="on";
a.log_flags="lnn";
a.grid=[3,3];
title("Responses Error Against Golden Sample");
xlabel("frequency (Hz)");
ylabel("level (dB)");
drawnow();
```

Then the rating criteria is evaluated:

```
//analysis frequency range indexes search
[errval,fmini]=min(abs(measfreq(1,:)-fming));
[errval,fmaxi]=min(abs(measfreq(1,:)-fmaxg));
```

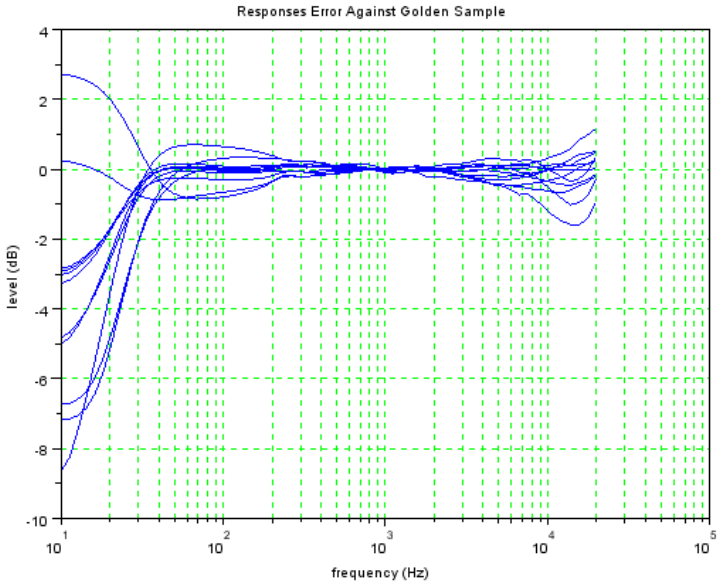

Figure 8: Errors against golden sample

```
//compute cumulative error
ssnorm=sum(sqrt(golderro(1:$,fmini:fmaxi).^2),2)./size(golderro(1:$,fmini
:fmaxi),2);
```

Candidates are identified, response are plotted and name of the items displayed on the Scilab console:

```
//find golden sample candidates
goldcandidates=ssnorm<=goldthres;
goldcandidatesindex=find(goldcandidates);

//output names on console
disp(measname(goldcandidatesindex));

//plot candidates and golden sample responses
f=scf();
drawlater();
plot2d(goldfreq,goldresp,5);
for i=1:size(measresp(goldcandidatesindex),1) do
    plot2d(measfreq(1,:),measresp(goldcandidatesindex(i),:),2);
end
a=gca();
a.box="on";
a.log_flags="lnn";
a.grid=[3,3];
title("Golden Sample and Candidates Responses");
xlabel("frequency (Hz)");
ylabel("level (dB)");
l=legend(["Golden Sample" measname(goldcandidatesindex)']);
l.legend_location="in_lower_right";
drawnow();
```

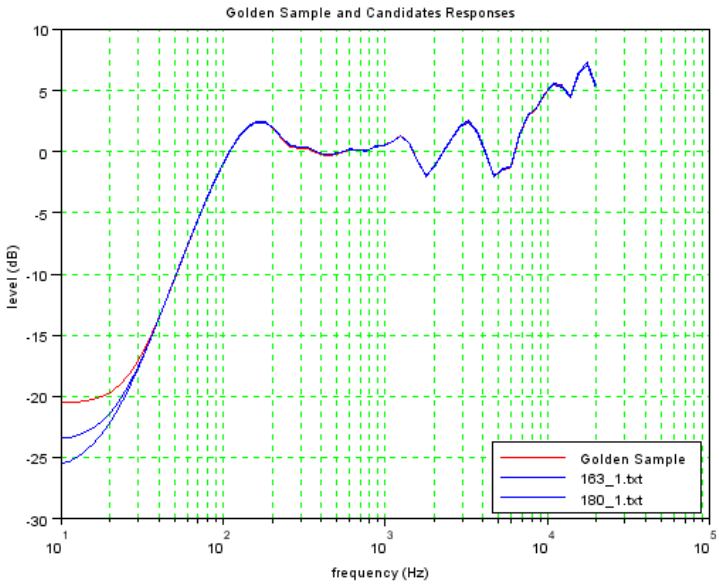

Figure 9: Golden sample and golden candidates response

CONCLUSIONS

Using Scilab it is possible to easily create flexible post-processing scripts that are able to handle very large sets of measurements. This approach is well suited to deal with typical QC applications that often require statistical analysis tools over large sets of data.

The Scilab software can be freely downloaded from the www.scilab.org website, if you are interested in the scripts presented in this application note, please search on Audiomatica web site www.audiomatica.com or write an email to dp@audiomatica.com.

REFERENCES

[1] Ponteggia, D., *"Statistical Analysis of Electro-Acoustic Measurements Sets Using Scilab"*, E-brief presented at the 131th AES Convention, New York, NY, USA, 2011

[2] Audiomatica, *"CLIO 10 QC Software Extension User's Manual"*, http://www.audiomatica.com/download/qcmanual10.pdf

[2] Baudin, M., *"Introduction to Scilab"*, Scilab Consortium, 2010, http://www.scilab.org/content/download/1754/19024/file/introscilab.pdf

[3] Rietsch, E. *"An Introduction to Scilab from a Matlab User's Point of View"*, 2010, http://wiki.scilab.org/Tutorials?action=AttachFile&do=get&target=Scilab4Matlab.pdf

[4] Ponteggia, D., *"CLIO 10 Sinusoidal File Structure With Import Examples In SCILAB"*, http://www.audiomatica.com/download/appnote_001.pdf

[5] http://en.wikipedia.org/wiki/Golden_sample (Accessed August 30, 2011)