IEEE Instrumentation and Measurement
Technology Conference, Venice
Italy, May 24-26, 1999

On-line Frequency Domain System Identification based on a Virtual Instrument

József G. Németh, Balázs Vargha and István Kollár

Department of Measurement and Information Systems
Technical University of Budapest
Budapest, Hungary, H-1521
E-mails: nemeth@mit.bme.hu, vargha@mit.bme.hu, kollar@mit.bme.hu
WWW: http://www.mit.bme.hu/[~nemeth, ~vargha, ~kollar]
Phone: +36 1 463-2057, fax: +36 1 463-4112

Abstract — **System identification is an inherently iterative process. Yet, limited attempts have been made so far to implement the whole identification loop in a single device. This paper discusses the difficulties of the task and presents a solution based on a Matlab toolbox and a set of virtual instruments.**

During the identification session the obtained models may call for refinement or validation by new experiments. Using this integrated software - hardware tool these subsequent experiments can be accomplished on-line with the identification procedure. As a demonstration, the modeling of a notch filter is described.

The universal applicability of this solution is believed to be guaranteed by the modular architecture of virtual instrumentation and the general definition of the software interface developed. The interface allows combining Matlab-based identification packages with virtual instruments or pure hardware interfaced to Matlab.

Keywords — **Virtual instrument, system identification, on-line, Matlab, SigLab, recursive measurement**

1. Introduction

System identification is used in various fields of engineering to determine the model that best corresponds to our observations about a system. A parametric model extracts the important aspects of the dynamic behavior of the system under test. Linear time invariant systems (LTI) are often applied as models for their relative simplicity and well-established theory. In fact, many systems are almost linear within their range of operation.

The procedure of identification of linear systems is often quite involved. It comprises the design of the excitation signal, the measurement of the excitation and the response and the estimation and validation of parametric models. Finding the best solution implies several iterations through these steps that can also include repeated measurements.

In spite of this *iterative nature of the system identification task* [1], limited attempts have been made so far to implement *the whole identification session in a single instrument*. This paper discusses the reasons for this and presents a solution.

For practical reasons, we considered a *study case* based on a frequency domain system identification package and a commercially available instrument. Yet, we would like to stress the possibility to apply this *solution in a more general context:*

- The underlying concepts of frequency domain system identification are equivalent to those of the time domain methodology despite all the differences in the methods they use. (In the frequency domain approach instead of using the measured data samples directly, their FFTs are taken, which has far-reaching implications on the whole procedure.) Altogether, it is clear that instead of using frequency domain identification, in this study we could use time domain system identification as well.
- The use of an advanced soft-instrument, as it will be seen, spared us the burden of hardware interfacing and building a complete graphical user interface, but this measurement setup is not considered as optimal or the only possible one. By making the necessary extensions, the implemented solution is applicable to a different instrument or device as well.

2. Current implementations of identification

System identification is used more often than one would think. In some respects adaptive filtering and control, channel equalization and many other applications belong to this topic since they seek to determine a set of model parameters based on a noisy input and output (or at least some assumptions about the input). The available knowledge about the input signal, the time and precision requirements, the applied model class and parameter

estimation algorithm and other circumstances cause the differences among these techniques and implementations.

There are just a few sophisticated *universal tools* that make use of *all the mathematical apparatus of modeling*. These are characterized by a strong dependence on the user as they leave choices open to him. Their first concern is interactivity and they concentrate less on doing things in real-time or running automatically adjusted to the needs of a specific task. Most importantly, data related to the actual measurements are typically transferred on an export / import basis within these tools. Therefore, they can be best described as *off-line processing tools*.

3. Difficulties of on-line identification

There are several questions and problems that need to be tackled before developing an on-line identification tool.

Firstly, the elaborated frequency domain methods [1,2] are currently non-recursive, i.e. they assume the availability of the whole data record needed to perform the necessary calculations. This implies that the measurement and the parameter estimation stages are inherently separated in time. From another point of view, physical system modeling usually does not require quick reactions. *Hence, no direct need* arises for a real-time identification tool.

Secondly, *specifying the requirements* for such an instrument is also cumbersome. Calculations for *s*-domain modeling are quite involved, especially for orders over 10-15. Thus, depending on the task, the requirements can vary in a wide range. An additional difficulty in creating an integrated tool is that *the hardware is almost always non-standard*. Even if it is commercially available, so in principle technical data are available, a lot of different data acquisition boards / equipment are in use. Con-sequently, it is extremely difficult to write standardized interface code that provides seamless data transfer from any hardware to a standard identification package.

Thirdly, even if the interface is given to handle several different kinds of hardware, the *user interface* is expected to *hide the irrelevant particularities* of each type of hardware and allow the *logical setting of parameters*.

Finally, many users do not know about the details of the measurement setup necessary for effective and precise system identification (e.g. time domain methods inherently assume that piecewise constant excitation is used, and the system response is measured without using an anti-alias filter [3,4,5], whereas frequency domain system identification in the *s*-domain requires anti-alias filtered measurement of both the input and the output signals.) It is also often overlooked that periodic excitations and measurement of full periods in steady-state can significantly improve the accuracy of the result [6]. Therefore, the interface of the instrument should make the best choices natural for the *user* and *guide their activity* in the right direction.

The solution for the above problems presented here is a *virtual instrument*, i.e. an instrument composed of layers of software and hardware having a virtual control panel that only appears on a computer display.

4. The advantages of virtual instrumentation

Virtual instrumentation is a fast growing area of the industrial and laboratory measurement technology. Modern instruments have so complex data processing units and user interfaces that they are similar to a PC from this point of view. On the other hand, in many cases engineers want to use their PC as a controllable and flexible measuring instrument. This demand resulted in the birth of virtual instrumentation.

Virtuality means that there are no real buttons or switches on the data acquisition hardware of the instrument: the user interface and most of the high-level data processing is realized by software on the host computer and its display.

The traditional instrument usually has fixed number of channels, fixed data processing abilities and user interface. The manufacturer-defined parameters of the instrument cannot be easily modified.

Meanwhile, the virtual instrument's *hardware* is typically *modular* and its software can be easily modified or extended by a *software upgrade*. In some cases the user himself can make his *own modifications* to match his specific needs. From this point of view, virtual instruments may be divided into two big groups:

- Virtual instruments that are built for end-users have special data acquisition software supplied by the vendors. For these softwares the source code is not distributed and they cannot be modified by the user.
- The other type of products supports the development of the software under *high-level programming languages* such as C, Pascal or Matlab and the vendors provide the instrument driver libraries.

There are many I/O interfaces available for virtual instruments:

- PC plug-in cards fit into the PC motherboard and mostly use PCI or ISA buses for communication. One of the main advantages of plug-in cards is that the measurement data are available in the RAM of the PC. However, these cards may suffer from the noise of the PC switching power supply and it is difficult to move the virtual instrument because it cannot be separated from the PC.
- Remote data acquisition means that the measuring instrument is not in the PC. Special buses were developed to support remote control measurement in the last 20 years, for example IEEE 488 and VXI, but traditional communication interfaces can also be used,

such as USB or SCSI. The remote access solution allows to place the data acquisition part of the instrument the nearest to the place of the measurement, or to leave it running autonomously in a field experiement.

The *scalable architecture* of computers and bus systems provides virtual instrumentation with flexibility at the lowest possible cost. The *development* and *maintenance* costs are also low because of the use of high-level programming languages and standardized, large-scale hardware components. The widespread use of *portable PCs* makes this line of instrumentation even more appealing.

5. Bases for an integrated virtual instrument

5.1. Frequency Domain System Identification Toolbox of Matlab

Matlab is a high level programming language that is well suited to the task of matrix computations exceedingly used in identification.

Its *Frequency Domain System Identification Toolbox* [2] is a universal off-line tool that contains sophisticated methods and services. The graphical interface of the toolbox offers a route through each step of the identification loop [1], except for accomplishing the on-line measurement. Data can be exported from and imported into the toolbox.

5.2. SigLab virtual instruments

Meanwhile, another Matlab based tool, *SigLab* (DSP Technologies) [7], is *a collection of virtual instruments* based on a common data acquisition hardware.

The hardware comprises 18-bit sigma delta converters for high-precision measurements in the audio range and three DSPs for real-time computing and communication. It is accessible via SCSI bus from a host computer. This high-speed bus allows using more SigLabs together.

SigLab provides virtual instruments for both time-domain and frequency-domain analysis. The *virtual function generator* can produce common excitation signals such as sine, square, chirp or random waveform and arbitrary waveforms such as a multisine. (In this later case the waveform has to be designed previously and saved in Matlab.)

The output can be analyzed in the time-domain with a *virtual oscilloscope*. The time base, filtering, (pre- / post-) triggering, averaging option can be set similarly to a digital oscilloscope.

For *frequency domain analysis* SigLab contains a stepped sine spectrum analyzer and a Fourier analyser supporting the non-parametric estimation of frequency response function.

It also contains a *virtual identification tool*, which provides satisfactory parametric models for a certain range of applications.

Altogether SigLab addresses successfully the problem of making *multi-channel experiments* with *real-time preprocessing* (averaging, FFT, statistical estimates) but offers limited services as far as data processing and analysis is concerned.

5.3. Combining the Toolbox with SigLab

The great advantage of SigLab compared to other virtual instruments is that the user can manipulate the measurement data directly in Matlab without the overhead of transferring and converting data between programs. Together with the Frequency Domain System Identification Toolbox, they provide a flexible environment for measurement and identification. The idea to combine these two tools into an integrated virtual instrument followed naturally. In order to do so, we needed to build an *'intelligent' interface* that handles some previously mentioned difficulties of on-line tools.

6. Interface allowing on-line identification

6.1. Guidelines applied

We had to keep in mind several criteria in developing this interface between the FDIdent toolbox and SigLab:

- First of all, SigLab is an independent product, so it is risky to override any m-files in the SigLab package. The interface ought to be *compatible* with next versions of SigLab.
- It must *allow* users of *different hardware* to build their own module into it.
- The *specification* of the interface should be *general* enough to support other kind of identification packages (e.g. the time domain *System Identification Toolbox of Matlab*).

6.2. Implemented features

The interface is added to the existing tools as a new entity through new visual controls and the actions behind these controls. It is not merely a means to launch the measurement, transfer and convert the data, but offers complex services in a user-friendly fashion.

The interface contains, first of all, the description of the hardware with function calls that operate on this data. It is through these calls, for instance, that consistency checks and necessary adjustments can be carried out during the design of the excitation signal in the identification package.

The technical data about the hardware as well as the properties of the virtual control panel are mostly collected dynamically in the interface code, so that the modifications in subsequent versions of the virtual instrument do not necessarily lead to inconsistencies.

HW description contents
List of available devices and associated virtual instruments
Freq./time domain?
Pre-defined excitation needed?
Max. amplitude peak
Legal values for:
excitation clock frequency
excitation record length
analysis sampling frequency
excitation record length
Options for:
reconstruction filter
anti-alias filter
with available BWs for both
All values can be given in analytical or explicit form, as they are only available through parametric calls, offering automatic adjustments, etc.

With a custom designed instrument this would not be a problem, but the virtual instruments in SigLab, as stand-alone instruments, offer restricted interfacing options. They give no guidance to the user in adjusting the measurement settings according to our purposes, so we had to provide automatic assistance in the parameter settings and structured help as part of the interface.

Some difficulties we encountered stemmed from the fact that SigLab supports, first of all, non-parametric identification. It does not calculate variances of obtained Fourier amplitudes and in some cases, it even returns coherence values over 1 because of 32-bit floating-point processing. It is arguable how well variances can be approximated from coherence values (even supposing that the system is linear), but erroneous coherence values close to one can completely corrupt the parameter estimation, because these correspond to noiseless measured values allowing no freedom at the model fitting.

As a consequence of these facts, we had to add some new controls to the existing virtual instruments in order to allow easy execution of repeated measurements. All these extensions are also made dynamically without making any modifications to the original code of SigLab.

6.3. General applicability of the solution

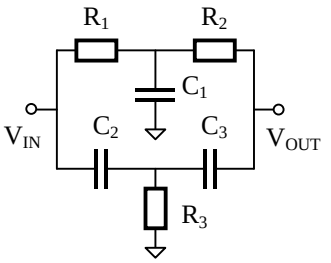
This interface is built up of separate modules, with clear definitions of services. On the instrumentation side we established a hardware description in an encapsulated form, and functions to call the virtual instrument. On the identification package side we designed a graphical interface and extensions to the virtual instrument.

Thanks to the modular design, components on both sides can be technically interchanged for a different tool of the same functionality. Beyond the implementation issues, the definition of the interface is believed to be general enough to support the use of different hardware or different identification package.

7. Experimental results

7.1.Measurement set-up

The presented on-line solution works, and we made tests to validate the abilities of the whole environment. We have built a simple passive circuit, a typical 50 Hz notch filter. The filter is a so-called double T filter, as it is shown in Fig. 1.



$R_1 = R$ $C_1 = 2C$
$R_2 = R$ $C_2 = C$
$R_3 = R/2$ $C_3 = C$

Figure 1. Notch filter.

The frequency response function of the filter is

$$\frac{V_{OUT}}{V_{IN}} = \frac{1 + R^2C^2s^2}{1 + 4RCs + R^2C^2s^2} \ . \qquad \text{(Eq.1.)}$$

The test circuit was built with $R = 330\,\text{k}\Omega$ and $C = 10\,\text{nF}$. The transfer function in Eq.1. represents the ideal case, when the parameters are precisely proportional to each other. The frequency response function of the system in Fig. 2. is also computed with the ideal parameter values and plotted from MicroSim PSpice.

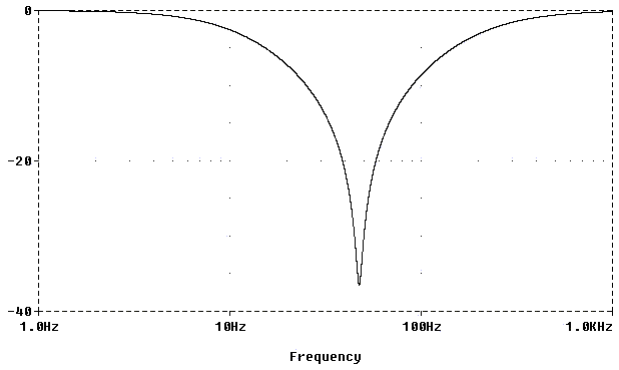


Figure 2. Transfer function of the filter with ideal parameters ($R = 330\,\text{k}\Omega$, $C = 10\,\text{nF}$)

Of course, the resistor and capacitor values have tolerances, so the transfer function will vary in each implementation of the filter. In order to illustrate its effect the Monte Carlo analysis of the filter is shown in Fig. 3.

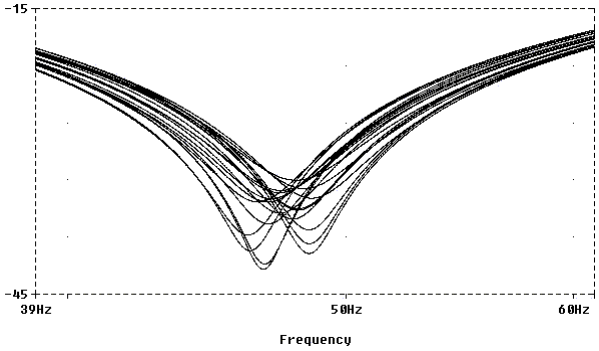


Figure 3. Modeling with tolerances: Monte-Carlo simulation

The Monte Carlo analysis is a way to determine the sensitivity of the transfer function with respect to the circuit parameters. This simulation was also made by MicroSim PSpice with 20 runs. It is clear from the simulation that the center frequency of the notch filter varies because of the tolerances. In a control application we might want to consider the parametric model of the real system.

In the ideal case we have a 2/2 order system according to Eq. 1. But in the general case a circuit containing three energy storage elements can be described by a 3rd order model. The apparent contradiction is explained by the degeneration of the model because of the symmetries in the parameters (i.e. $C_1 = 2C_2 = 2C_3$, $R_1 = R_2 = 2R_3$).

7.2. Identified models

We accomplished several experiments and identified and validated models of different orders on-line.

The excitation signal was a multi-sine containing 40 linearly spaced frequencies from 5 to 200 Hz. The variances were estimated from the results of 5 experiments. The measurements had about 90 dB signal-to-noise (SNR) ratios.

We identified models of orders between 2/2 and 4/4. Figure 4. shows the chart of the cost function values for each model.

Except the 4/4 model, the cost function values are extremely high. This is due to the practically noiseless measurements, which cause the heavy weighting of small model-data errors in the cost function [1].

As it was expected the 2/2 model could not describe the system because of the circuit parameter tolerances. Interestingly, the third order models still fail.

The frequency response function of the 4/4 model and the model-data error terms (residuals) are plotted in Fig. 5. This model performed well on measurement data coming from *independent experiments* with different excitation signal and noise levels, too.

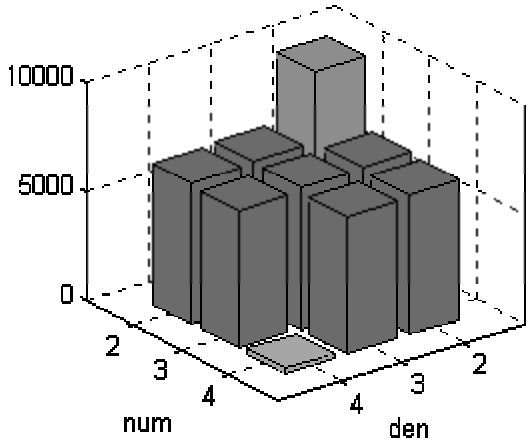


Figure 4. Cost functions of identified models with different order

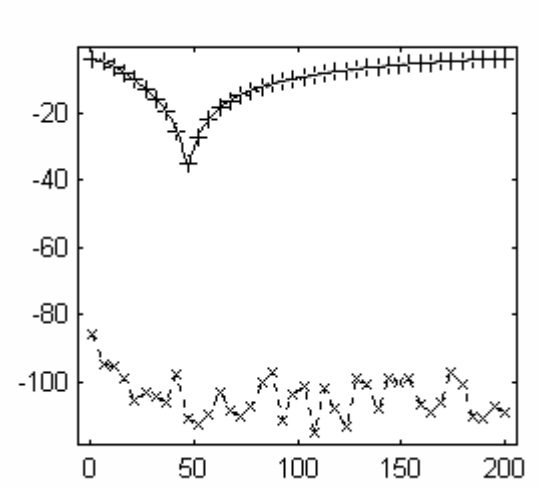


Figure 5. Frequency response function of the 4/4 model with model-data errors

Clearly the 4/4 model cannot be accepted as the pure model of the filter unless we investigate the reasons for the higher model order by making *further experiments*. The extra pole and zero in the model can be due to parasitic effects or small nonlinearities of the realized filter but it may well be caused by imperfections of the measurement channels:

- a slight input/output channel mismatch,
- the frequency dependant gain of the preamplifiers, or
- small nonlinearities.

In this situation *possessing an on-line tool is a great advantage*. Further experiments need to be done to evaluate the models with different excitation signals and to determine the model of the measurement channels. Once the channel is modeled, the measurements can be compensated for its effect.

8. Future goals

8.1. Simulator

A main inconvenience of SigLab is that the software cannot be tested if the hardware is not present. However, for development purposes it would be advantageous if the software could be used alone.

The best solution to this problem would be a *hardware simulator program*. The simulator is an ideal tool for verifying the theoretical considerations before using them in the measurement process. Without the simulator, each developer must have a device, even if measurement will only be made on one site.

The other benefit of developing the simulator would be the use of this integrated tool in the education. It could be very useful for teachers to check the students' knowledge, who are studying system identification or digital signal processing. The built-in recorder tool in the identification package can record the student's steps in solving the problem. This tool enables the teacher to get an overview about the knowledge of the class (e.g. is he going too fast ?, etc.).

In practice the simulator means that the hardware dependent parts of the software should be simulated by new hardware-independent Matlab files. SigLab's hardware is handled by a dll file, which must be replaced by a new m-file implementing a virtual hardware under Matlab.

8.2. Extensions to different hardware

In developing this interface we tried to keep the very flexible architecture of virtual instrumentation. On one hand, the software is written under an interpreter language, thus it is *portable between different platforms*. On the other hand, flexibility, in our interpretation, also requires the independence of the hardware. The easy interfacing to another hardware makes possible to use the appropriate data acquisition hardware or the device already in use for a specific measurement problem.

Although SigLab has a high-precision data acquisition device, it has some limitations, too. The main limitation is its small bandwidth. Currently the bandwidth is adequate for making measurements in the audio bandwidth. One of our future goals is to build an *automated impedance analyzer tool*, which gives a more complex model than a simple parallel or serial model. A real capacitor, for instance, can be modeled by four parameters: serial inductance and resistance of the lead wires, capacitance, leakage resistance. In order to get enough information about all the poles and zeros we need a bandwidth of about some megahertz.

9. Conclusion

We built *an 'intelligent' software interface* allowing the integration of SigLab under the Frequency Domain System Identification Toolbox of Matlab.

The *functioning of the resulting integrated virtual instrument* has been demonstrated. It allows to carry out experiments *on-line* with the system identification procedure. During the identification session the user can use the modeling results based on the previous experiment to modify the experiment setup and refine or validate their model by making further measurements.

The integrated virtual instrument is organized in layers from the measurement hardware up to the system identification software. Thus the solution is *open to* application specific *extensions* and modular development.

The availability of an integrated environment for on-line system identification can, in our belief, significantly contribute to the development of universal automated modeling tools.

References

[1] Schoukens, J. and Pintelon, R. , Identification of Linear Systems: A Practical Guide to Accurate Modeling. Pergamon Press, 1991

[2] Kollár I. , Frequency Domain System Identification Toolbox. The MathWorks, 1994

[3] Franklin, G. F., Powel, J. D. and Workman, M. L. , Digital Control of Dynamic Systems. Addison-Wesley, 1990

[4] Ljung, L. System Identification: Theory for the User. Prentice-Hall, 1987

[5] Schoukens, J., Pintelon, R. and Van hamme, H. , "Identification of Linear Dynamic Systems using Piecewise Constant Excitations: Use, Misuse and Alternatives" Automatica. Vol.30, No.7, 1994, pp. 1153-69

[6] Godfrey, K., Perturbation signals for System Identification. Prentice-Hall, 1993

[7] SigLab 2.0 Manual., DSP Technology Inc., Fremont, CA, 1995