

A Course in Computer-based Instrumentation: learning LabVIEW™ with Case Studies*

A. BRUCE BUCKMAN

Department of Electrical and Computer Engineering, The University of Texas, Austin TX 78712, USA.

E-mail: buckman@uts.cc.utexas.edu

Case studies in computer-based instrumentation were used as a vehicle to teach LabVIEW programming to a class of graduate and advanced undergraduate students. Selection of cases was heavily student-driven. Much of the case study method's efficiency appears to arise from the way it mimics, and provides instructor guidance in the way nearly everyone independently learns to use software.

INTRODUCTION AND BACKGROUND

ENGINEERING FACULTY continually face the problem of introducing new technology into a curriculum that has no room for any additions. Virtual instrumentation is one of those new technologies that must find its place in future curricula. This paper is a report on a new undergraduate elective course on computer-based instrumentation that was forced to start with an assumption of no student background in LabVIEW™ programming. The course addressed this problem by approaching the task of learning LabVIEW the same way most software purchasers learn a new acquisition: choosing a small but useful task to do, learning how to do that task, and then branching out into new tasks of ever-increasing complexity. This approach usually does *not* involve reading a manual from cover to cover, or working through examples that are either oversimplified or unrelated to the first tasks the user wants to accomplish with the software.

A case study approach, patterned after the case studies used in many business schools, seems appropriate. The main task of the organizer of a case studies course is a careful selection of studies with a goal of introducing all the necessary skills and functions, coupled with leading the students through the process of developing the end product, which is in this case a LabVIEW virtual instrument (VI).

STUDENT-DRIVEN CASE SELECTION

At the outset of the course, the students were polled concerning what instrumentation, control, and signal-processing functions they would like to

be able to accomplish with LabVIEW by the end of the course. Many had some idea of what LabVIEW could do in these areas, but several did not. Some were graduate students with very specific goals in mind, which related directly to their thesis or dissertation projects. Others were undergraduates with little detailed knowledge beyond a professed interest in instrumentation or signal processing as a career field.

The initial student poll results could be classified into the following areas:

1. Rapid frequency acquisition.
2. Rapid phase acquisition.
3. Image processing.
4. Control of an experiment from a computer.

The mutual overlap among these applications was clearly not sufficient to cover everything in a one-semester course, so an instruction and negotiation phase followed. During this time, the instructor's goal was to reduce the expressed areas of interest to a set of cases that allow, or even require, coverage of the underlying LabVIEW programming concepts that need to be covered in the course, regardless of the specific cases used. The final selection of cases is governed by time constraints, instructor expertise, and the feasibility of getting a reasonable result in the allotted time, as well as by the extent of student interest. Three cases were settled on for coverage in our course:

- **Case 1.** Rapid determination (in a time much less than one cycle) of the phase difference between a noisy, nominally sine wave, signal and a clean reference.
- **Case 2.** Rapid determination (in a time much less than one cycle) of the frequency of a noisy, nominally sinusoidal, signal.
- **Case 3.** A virtual vector voltmeter for magnitude and phase comparisons of two noisy sine waves.

* Accepted 9 September 1999.

THE PROCESS OF CASE DEVELOPMENT

Each case gets developed in similar fashion, but draws on a different set of LabVIEW programming operations. In this paper, Case 1, above, will be used to illustrate the steps in case development. The sequence of steps is as follows.

Step 1: Refine the problem definition

The starting point of this step is the short descriptive problem statement given above. Most likely, several approaches to solving the problem will be investigated, so consideration of how the solutions are to be compared is critical. The class decided that responses of the candidate systems to step-wise, discontinuous changes in the phase shift should be compared on the basis of standard system response parameters such as rise time, settling time, and steady-state error, for a set of different noise levels relative to a fixed signal amplitude. Since amplification could equalize signal and reference amplitudes, no generality is lost by choosing unit amplitudes for reference and signal in the absence of noise. Uniform white noise was used in the simulation, since it provided a more severe test of the candidate systems than $1/f$ noise.

Step 2: Build a simulation test-bed in LabVIEW

This step included generating sampled-data simulations of reference and noisy signal with user-controlled noise amplitude. These waveforms are the inputs to the various solutions (sub-VI's or just sections of diagram) which are wired into the test-bed for evaluation. For Case 1, the test bed provided a 1024-sample waveform at a selected frequency, for which the phase changed discontinuously at the midpoint, sample 512. This waveform is subjected to additive uniform white noise. Figure 1 shows the wiring diagram of one of the VI's developed for Case 1. The first For Loop at the upper left of the diagram generates the test signals. The unfiltered signal waveform is brought to the front panel as a waveform graph. The phi sig sub-VI generates the phase-modulated sine wave. Its diagram is shown in Fig. 2. At the far right of Fig. 1 is another waveform graph called phase(t). It plots the imposed phase modulation (a step discontinuity at sample 512) and the resulting phase versus time as decoded by the phase acquisition algorithm between the unfiltered signal and phase(t). The upper right corner of the diagram is a tool for evaluating the noise in the measured phase once the system reaches steady state. It displays the mean and standard deviation of the subset of samples of the output waveform phase(t) which

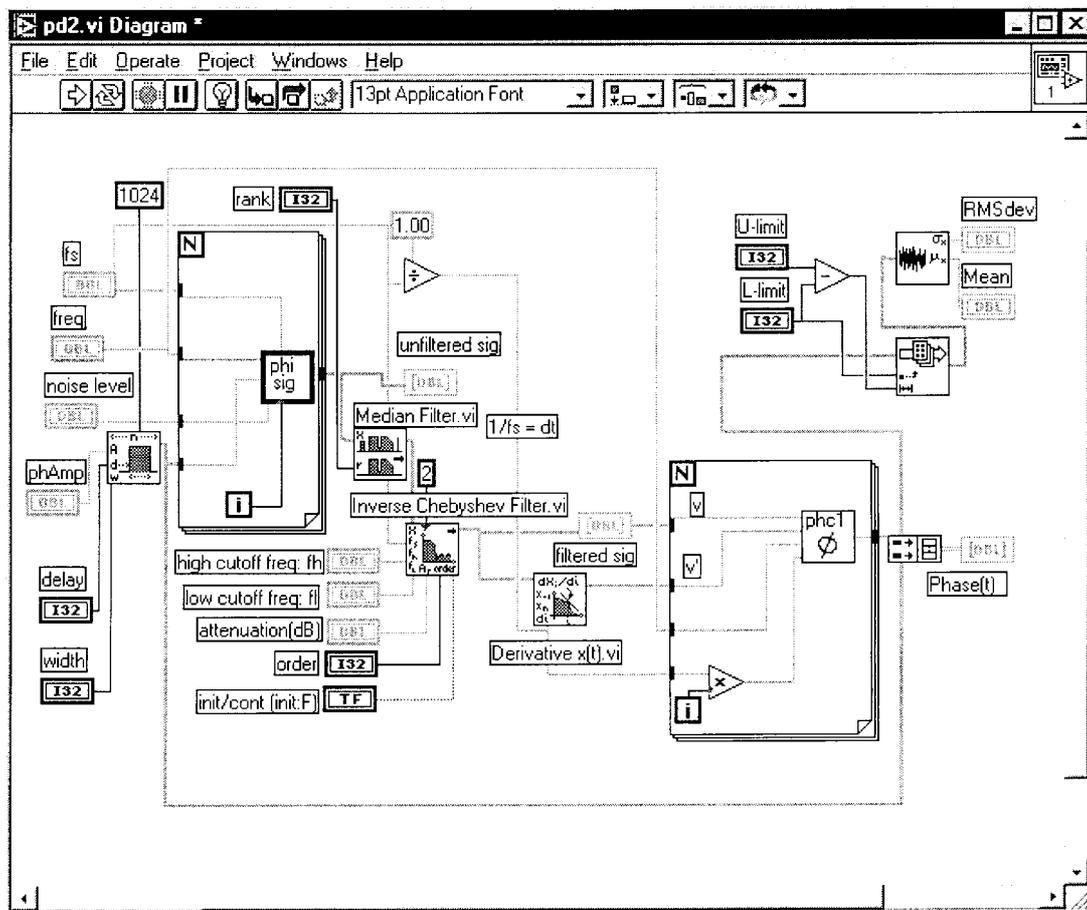


Fig. 1. Test-bed wired with signal and derivative-based algorithm.

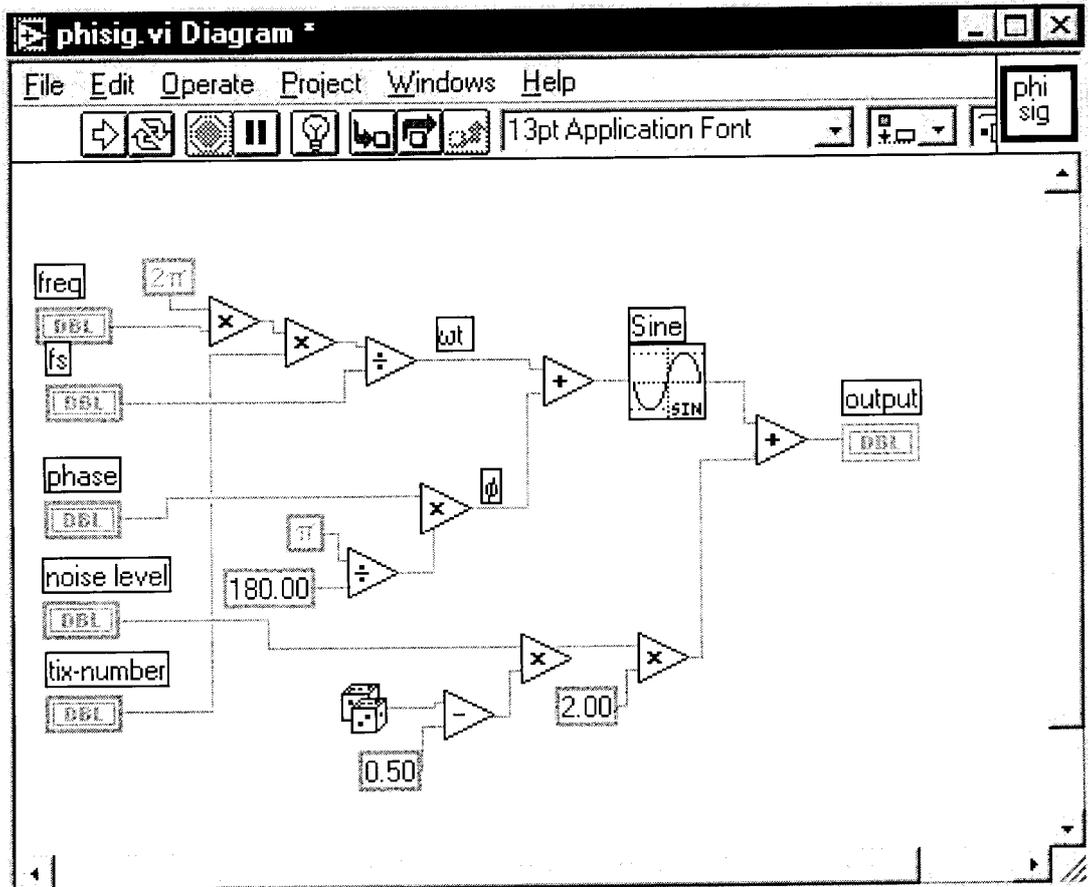


Fig. 2. Diagram of the phi sig sub-VI.

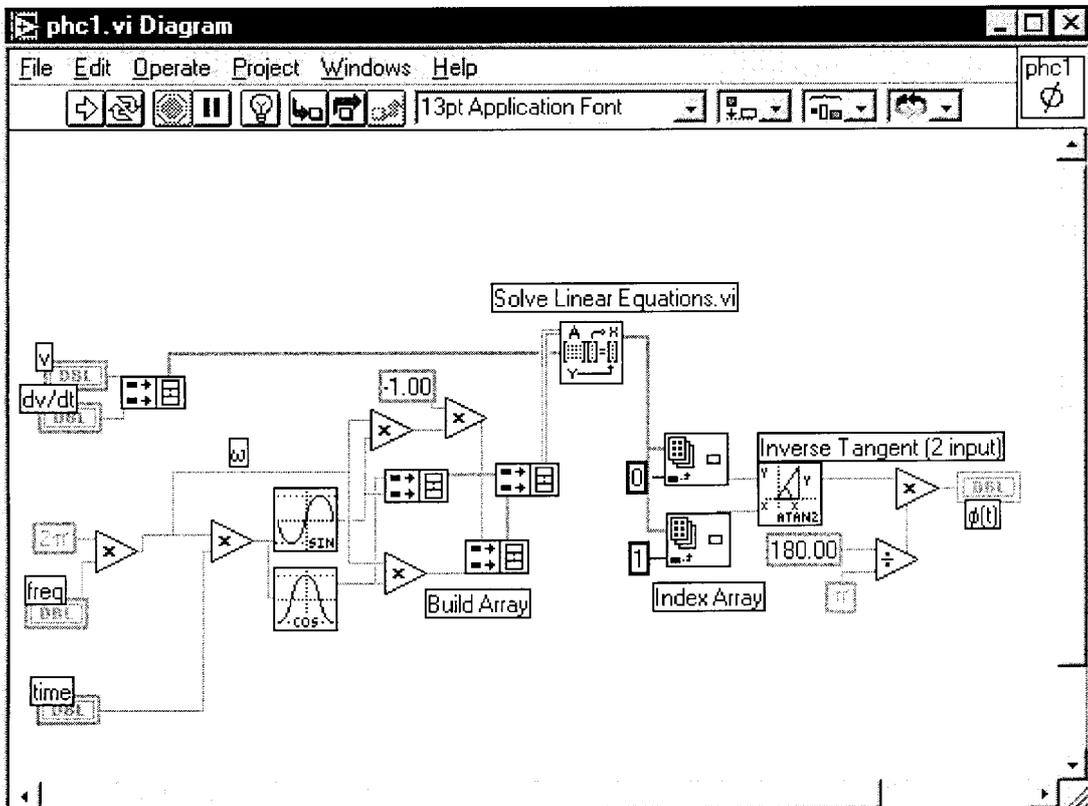


Fig. 3. Diagram of the phc1 sub-VI.

lie between sample number L-limit and sample number U-limit. The user of the simulation program must set these limits so that they define a region where the output has reached steady state after the phase discontinuity at sample 512. Then the mean calculated in this part of the VI compared with the value of the phase input after the discontinuity is a measure of the steady-state error, while the standard deviation is a measure of the effects of noise on the decoded phase.

Step 3: Wire the candidate algorithms into the test-bed and evaluate the results

The final, refined problem definition for this case study allowed evaluation of the candidate systems on the basis of several criteria, without specifying weights for each. This is generally a more instructive problem definition to use for a case studies course, since the various approaches tried will inevitably have different strengths and weakness with respect to the list of criteria. As the case study developed, the first approach tried was based on calculating the phase instantaneously from the ratio of the signal to its first derivative. This is the algorithm shown in Fig. 1. After filtering by a Median Filter.vi of user-adjustable

rank and by a bandpass Inverse Chebyshev Filter.vi user-adjustable parameters, the signal is differentiated. The signal waveform, $v(t)$, and its time derivative, $v'(t)$, are inputs to the sub-VI, phc1, whose diagram is shown in Fig. 3. This sub-VI solves for phase as a function of time, given the signal, its derivative, and the nominal frequency.

At this point, the case study clearly revealed the extreme noise sensitivity of this method in the absence of any filtering of the signal. This led to a consideration of the various filters available in LabVIEW, including combinations of these filters. The filtering in Fig. 1 was not proven to be optimum, but it did produce useful settling time, steady-state error, and steady-state noise levels, as shown by the front panel in Fig. 4.

Figure 5 shows the test bed evaluation of a competitive approach that reduces rise and settling times at the cost of increased steady-state noise. Figure 6 shows this VI's diagram. It performs a linear least squares linear fit to a window of samples of adjustable width. The parameters fitted are the coefficients, A and B in:

$$f(t) = A \cos(\omega t) + B \sin(\omega t)$$

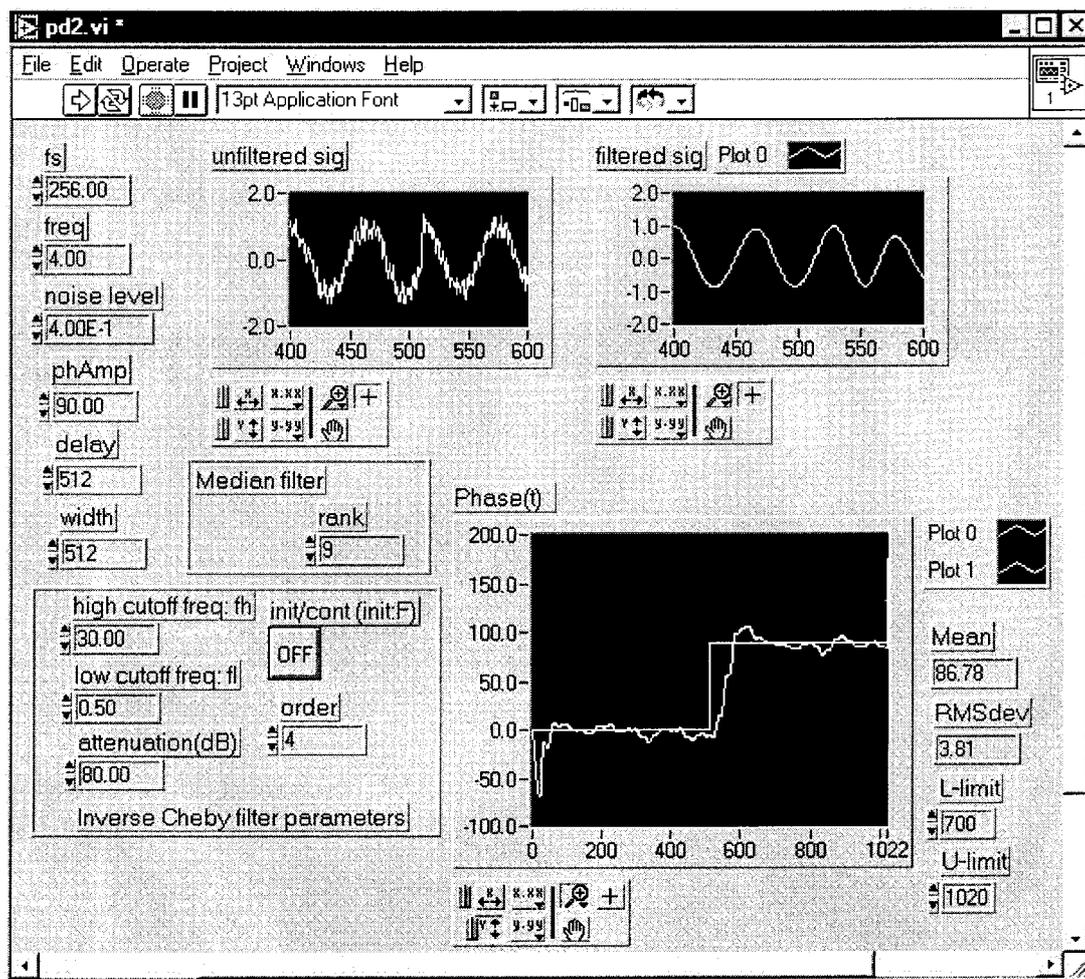


Fig. 4. Test-bed front panel, showing a typical result from the signal and derivative-based algorithm.

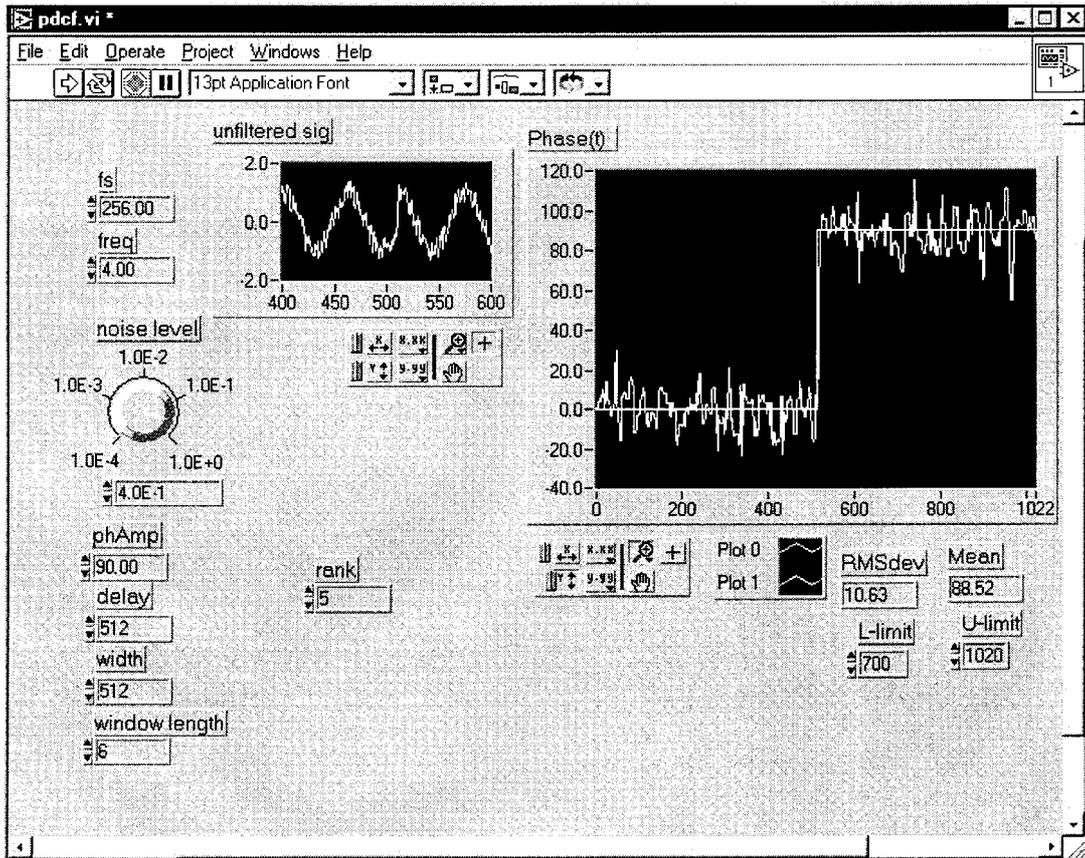


Fig. 5. Test-bed front panel, showing a typical result from the least squares-based algorithm.

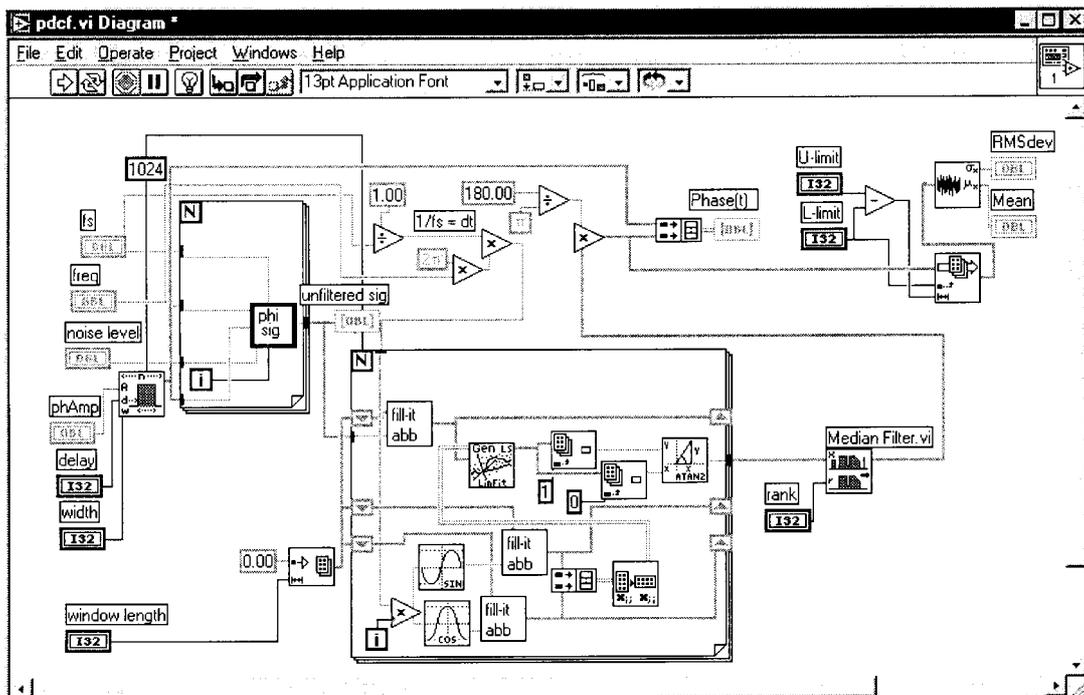


Fig. 6. Test-bed wired with least squares-based algorithm.

