A Practical Introduction to Digital Signal Processing through Microsoft Visual C++ and LabVIEW Programming*

Richard R. Schultz
University of North Dakota
rschultz@nyquist.ee.und.nodak.edu

Abstract

Conventional wisdom has relegated the introduction of digital signal processing (DSP) fundamentals to the senior year of the electrical and computer engineering curriculum, since a background in complex variables is required for the frequency domain analysis and design of digital filters. However, practical aspects of DSP can be taught to students at the sophomore level, since only a minimal proficiency in mathematics is required to understand concepts such as analog-to-digital (A/D) and digital-to-analog (D/A) conversion, sampling, and data analysis. "Computer Aided Measurement and Control" has been taught to electrical engineering sophomores at the University of North Dakota during the past several years. This course has evolved as a means of not only introducing students to the C++ and LabVIEW programming languages, but also as a practical introduction to DSP fundamentals. Students begin the course by learning programming skills, writing simple Microsoft Visual C++ programs which perform linear circuit simulation, random number generation and the calculation of statistics, digital function generation, and digital filtering. These experiments pave the way for discussions on A/D and D/A conversion and sampling. Next, the National Instruments NI-DAQ software library is linked with the students' C++ object code to perform relevant instrumentation experiments using the Lab-PC-1200 data acquisition card. Finally, LabVIEW, a graphical programming language developed by National Instruments specifically for instrumentation purposes, is introduced as an alternative to writing text-based source code. Since LabVIEW incorporates the same programming constructs as C++, namely selection structures and loop control statements, students are able to quickly grasp the design of LabVIEW virtual instruments. In addition, the visualization capabilities provided by LabVIEW help the students achieve an intuitive understanding of sampling, time-frequency duality, and filtering. The natural progression of exposure to (1) Microsoft Visual C++ programming; (2) practical DSP fundamentals; (3) the NI-DAQ software library; and (4) LabVIEW provides the students with a number of real-world engineering skills that can be applied in any instrumentation laboratory.

1. Introduction

Traditionally, digital signal processing (DSP) has been taught to electrical and computer engineering (ECE) seniors and graduate students from the classic text by Oppenheim and Schafer[1] or a similar text which relies on the Z-transform to aid in the analysis and design of digital filters in the frequency domain. More contemporary DSP textbooks, such as the text by Mitra[2], use MATLAB and other computer tools to aid in the visualization of sampled signals in both time and frequency. For these texts, a high degree of mathematical proficiency--particularly in the area of complex variables--is required to

* This work was supported in part by the National Science Foundation Faculty Early Career Development (CAREER) Program, grant number MIP-9624849. In addition, this material is based upon work supported in part by the U.S. Army Research Office under contract number DAAH04-96-1-0449.

understand the concepts and to participate fully in the digital filter design process. However, understanding the basic concepts behind discrete signals, including sampling, signal processing, and data analysis, are skills required by all engineering students at an earlier stage of their academic careers, since computers have become fixtures in the instrumentation labs of all engineering disciplines. There is a growing trend to write more intuitive DSP texts for this audience, beginning with the excellent overview of digital audio and computer music by Steigletz[3], and, more recently, the text by Lyons[4] which uses a plethora of examples to promote an understanding of basic concepts. One of the most innovative papers presented at the 1997 ASEE Conference by M. A. Yoder[5] boldly claimed that students should be exposed to DSP in an introductory electrical engineering course *before* they are exposed to analog circuit theory, since "computers are prevalent and easy to work with." Inspired by this highly logical, yet seemingly revolutionary idea, the course Computer Aided Measurement and Control at the University of North Dakota has been revised to incorporate a modified version of "DSP First[5]."

Computer Aided Measurement and Control is a required sophomore-level electrical engineering (EE) course, introduced originally as an avenue for teaching undergraduate students the C programming language and instrumentation applications using a data acquisition card. The decision to place the course within the sophomore year of the curriculum was two-fold: first, learning a programming language early helps students think logically and develop their creativity, skills which must be applied later in more design-oriented courses; and secondly, a large number of students participating in the cooperative education program after their sophomore year are involved in writing software, often using the C and C++ programming languages. The author started teaching this course in the fall semester of 1995, and has since been directly involved in three evolutions of the course. Computer Aided Measurement and Control still, in essence, concentrates on fundamental programming skills; however, the languages that are taught are now C++ and LabVIEW. Laboratory assignments have evolved to the point where each experiment exposes students to some practical aspect of digital signal processing, including proper sampling rate selection during data acquisition, digital filtering, and data analysis. Unlike courses based on practical DSP texts which cover sampled signals in both time and frequency, students enrolled in Computer Aided Measurement and Control are introduced to discrete mathematics in the time domain only. Improving the students' programming skills is still the number one priority of the class, but they also obtain a practical knowledge of DSP through the laboratory assignments.

This paper is organized as follows. Section 2 explains the philosophy behind teaching practical DSP concepts to sophomores rather than waiting for students to reach their senior year so that they have the mathematical proficiency necessary to understand the theory and design methodologies associated with digital signal processing. Section 3 specifies the hardware and software configuration used in the Computer Aided Measurement and Control laboratory, while the C++ and LabVIEW programming experiments are described in Section 4. A brief summary of the paper along with the planned course evolution is provided in Section 5.

2. Teaching Practical Digital Signal Processing Concepts

As alluded to previously, Computer Aided Measurement and Control is not a conventional DSP course, in the sense that students are not introduced to the Nyquist sampling theorem, the Z-transform, and methodologies of digital filter design in both the time and frequency domains. The original intent was to provide electrical engineering students a course in C programming as an alternative to the FORTRAN course formerly required of all engineering students at the University of North Dakota. As the instructor of Computer Aided Measurement and Control, the author feels that all engineering students, particularly electrical engineering undergraduates, need to learn a high-level programming language and obtain

practice in writing software for a variety of applications. Thus, the main priority remains that of providing software engineering experience, a highly marketable skill in today's engineering companies. No room in the EE curriculum existed to teach an additional required course in DSP fundamentals, although this is also a necessary component of a modern electrical engineering undergraduate education. Under these constraints, the solution was to modify the programming assignments in Computer Aided Measurement and Control, such that the students would be exposed to practical DSP concepts while simultaneously gaining experience in the use of selection structures and loop constructs. An emphasis is placed on helping the students achieve an *intuition* into the use of various signal processing tools, rather than teaching them theoretical concepts. With the proliferation of computer instrumentation in all technical fields, engineers will almost certainly be involved in the design of hardware/software systems to perform automated data acquisition and digital control at some point in their careers. Laboratory exercises were designed so that the students are exposed to a number of real-world DSP applications that they will face in the future, namely:

- Data Analysis and the Computation of Signal Statistics;
- Time Domain Mathematical Representation of Discrete Deterministic Test Signals;
- Selection of Data Acquisition (DAQ) Cards;
- Microsoft Visual C++ and National Instruments LabVIEW;
- Software Libraries for Use in Accessing DAQ Card Functionality;
- Sampling Rate Selection and the Visualization of Aliasing Artifacts;
- Digital Signal Processing Applied to Digital Speech and Music;
- Finite Impulse Response (FIR) Digital Filters; and
- Digital Image Processing Applied to Multidimensional Signals.

Certainly, this is not an exhaustive list of DSP concepts, but it is representative subset of the most important practical applications that will be encountered in an instrumentation laboratory.

3. Laboratory Hardware and Software Configuration

In this section, the computing hardware, test equipment, and software required for the Computer Aided Measurement and Control laboratory is described. Initial expenditures are somewhat high, so an alternative, lower-cost strategy is also discussed.

3.1. Computing and Instrumentation Equipment

The hardware required in the instrumentation laboratory represents the largest expenditure. Eight instrumentation benches are currently in use, each containing the following devices:

1. A personal computer (PC) containing at least 64 MB of RAM, a 2 GB hard disk, and the Microsoft Windows NT operating system is the workstation of choice. A Gateway 2000 Pentium-based PC with a clock speed of 200-MHz retails for under $2,000 (January 1998 price list), and prices are dropping on a monthly basis. Workstations running at the highest clock speeds available are not required for this educational laboratory; a substantial savings can be obtained by buying a cluster of PCs which are approximately a year behind the computational speed of state-of-the-art machines.

2. The National Instruments Lab-PC-1200 data acquisition card (formerly the Lab-PC+ card[6]) was purchased with a terminal block and interface cable for approximately $700 with an educational discount (June 1997 price list) This DAQ card is Plug and Play ISA-compatible, and it has the following input/output (I/O) specifications: sampling rates up to 100,000 samples/sec; eight analog input channels with 12-bit resolution; two analog output channels with 12-bit resolution; and 24 digital I/O lines, with the TTL lines grouped as three 8-bit ports. The Lab-PC-1200 card was chosen

for several reasons: (1) National Instruments manufactures DAQ cards which are controllable by linking Microsoft Visual C++ source code with the NI-DAQ software library; (2) National Instruments provides device drivers for its DAQ cards which interface with the LabVIEW graphical programming language; and (3) the Lab-PC-1200 is the lowest cost DAQ card manufactured by National Instruments with a sufficiently high sampling rate and the diverse functionality required for use in an educational instrumentation laboratory.

3. Conventional electrical engineering test equipment is required at each station, including a combination multimeter, DC power supply, function generator, and frequency counter test instrument (approximately $500) and an analog oscilloscope (approximately $1,000, reconditioned). With this set-up, the DAQ card can be used to sample analog input signals with variable frequency, amplitude, and DC offset, as well as output digital signals that have been generated or filtered by a computer program for visualization on the oscilloscope screen.

The total cost of all hardware for a single instrumentation test bench is approximately $4,200. If cost is a significantly prohibitive factor in setting up the laboratory, all signal input and output can be simulated. In this case, the only hardware cost is the PC workstation for under $2,000. However, this does severely limit the real-world exposure students receive in digital signal processing applications, since they cannot access an actual DAQ card in the laboratory assignments.

3.2. Required Software Packages

Software costs are extremely low in comparison to the instrumentation equipment expenditures. The following five software packages are used in most of the experiments, with alternative choices suggested as appropriate:

1. Microsoft Visual C++ has been chosen as the compiler used in Computer Aided Measurement and Control. Through educational pricing, this compiler can be purchased for approximately $50 per license (August 1997 price list). For the C++ laboratory assignments, students write Win32 Console Applications that execute in a DOS shell. In this course, the students do not become involved in writing more complex Windows applications. With Microsoft Visual C++ installed in the laboratory, more advanced software utilities can be developed by the students for subsequent courses and design projects. Furthermore, this compiler was chosen since Microsoft Visual C++ object code can be linked with the NI-DAQ software library[7] to access the Lab-PC-1200 DAQ card functionality. In this manner, data acquisition and digital control applications can be written in C++, with low-level sampling and data transfer handled by the NI-DAQ utility functions. Although Microsoft Visual C++ was selected for use in the laboratory, Microsoft Visual BASIC and Borland C++ can also be used to control the Lab-PC-1200 card via the NI-DAQ library.

2. The NI-DAQ software library is a set of compiled functions which handles low-level DAQ card operations such as A/D and D/A conversion, memory management, and timing. This library and its associated documentation[7] are enclosed with the Lab-PC-1200 card at no additional cost. For simplicity, essentially two functions are used by the students: `AO_VWrite()` performs a single D/A conversion to handle digital signal generation and control applications, and `Lab_ISCAN_Op()` performs multiple A/D conversions and places the samples into a buffer array for data acquisition applications.

3. The Student Edition of National Instruments LabVIEW[8, 9] is installed on each computer to implement instrumentation algorithms using a graphical programming language. The student edition was chosen since a single copy can be purchased for under $80 (August 1997 price list), while the professional version is significantly more expensive. Obviously, National Instruments provides a device driver

for the Lab-PC-1200 card to interface with LabVIEW. Currently, the Student Edition of LabVIEW will not operate properly within the Windows NT environment, so each computer also requires a Windows 95 boot partition to accommodate LabVIEW. With an educational discount, Windows 95 can be obtained for less than $50 per license (August 1997 price list).

4. Microsoft Excel is a spreadsheet utility integrated with the Microsoft Office software package. This program is used for plotting digital signals that have been generated or processed by C++ programs and written to output text files. Each Gateway 2000 computer is delivered with Microsoft Office installed on the hard drive, so Excel does not contribute any additional expense. Any spreadsheet software package can be used in the place of Excel, since all spreadsheet programs are capable of importing text files and plotting the sample values.

5. Microsoft Word is the word processing utility integrated with Microsoft Office, used by the students to generate laboratory reports. Again, no additional expense is incurred, since Microsoft Office is included in the purchase of the computer. Alternatively, any word processing package can be used instead of Word.

Total software costs are approximately $180 for each workstation configured in this manner. In summary, each instrumentation laboratory set-up, including all hardware and software, can be obtained for approximately $4,380.

4. Laboratory Assignments

The first 12 weeks of Computer Aided Measurement and Control are dedicated to teaching C++ text-based programming using *Essential C++ for Engineers and Scientists* by J. R. Hanly[10], and the four remaining weeks of the semester are spent teaching LabVIEW graphical programming from *The LabVIEW Student Edition User's Guide* by L. K. Wells[8]. LabVIEW is a graphical programming language which allows for the development of an integrated hardware and software instrumentation environment. Data acquisition, digital control, data analysis, and data logging can all be performed quite easily by building *virtual instruments* (VIs) and connecting them together. LabVIEW utilizes *dataflow program execution*, in which an individual system module executes only after all data and parameters have arrived at the input to the module. This mode of execution is quite similar to that of a real-world signal processing or control system. Since the students understand C++ programming constructs by the time they are introduced to LabVIEW, they find graphical programming to be rather intuitive. Students are taught to think in terms of implementing algorithms using both C++ and LabVIEW. Figure 1 shows a C++ function which computes the sample values for a digital sine wave and places the values into a 1-D array. The Front Panel and the Block Diagram of a LabVIEW VI which performs the same operation are shown in Figure 2. This shows how a `for` loop can be implemented in either language. Likewise, the C++ function and the LabVIEW VI shown in Figures 3 and 4, respectively, describe how selection structures (`if-else` in C++, Case Structure in LabVIEW) are used to implement a square pulse.

Students enrolled in Computer Aided Measurement and Control are exposed to a number of programming and instrumentation laboratory assignments which provide an intuitive understanding of digital signal processing and data analysis. In this section, the Microsoft Visual C++ and LabVIEW experiments are described, along with the primary C++ and DSP-related concepts learned through each assignment.

```
void Sine(double array[1024], double amplitude)
{
   const double PI = 3.1415927;

   for (int i=0; i<1024; i++)
      array[i] = amplitude*sin(2.0 * PI * 0.01 * i);
}
```

Figure 1: C++ function to generate a 1-D array containing digital sine wave sample values.

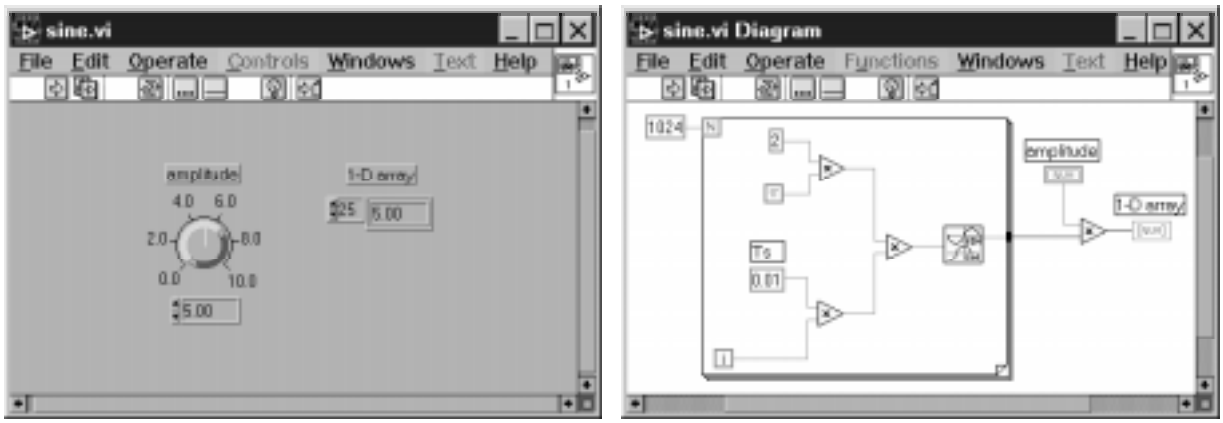

Figure 2: LabVIEW sine wave generator VI, showing the Front Panel (left) and Block Diagram (right).

```
void Pulse(double &y, double x)
{
   if (x>=0 && x<=5.0)
      y = 5.0;
   else
      y = 0.0;
}
```

Figure 3: C++ function to calculate a square pulse sample value.

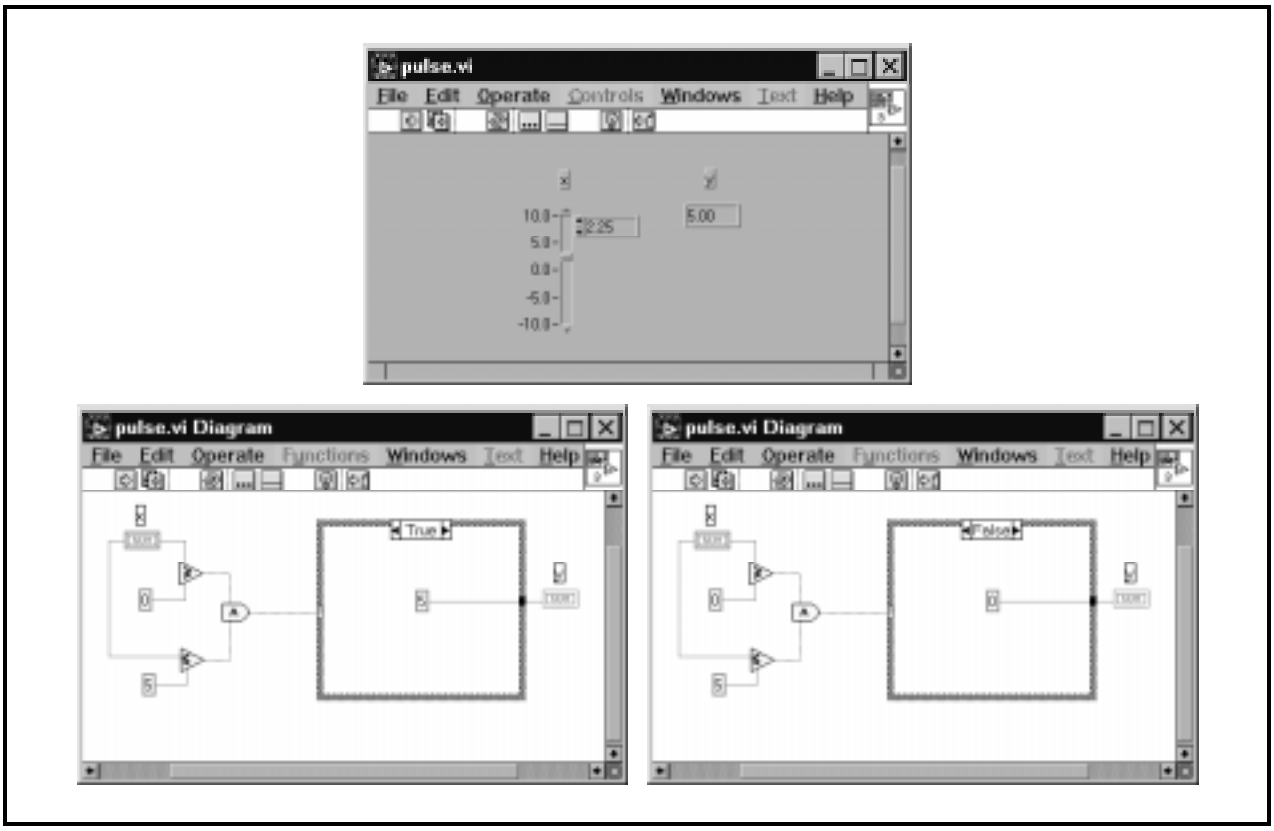

Figure 4: LabVIEW digital pulse generator VI, showing both possible cases within the Block Diagram.

4.1. Microsoft Visual C++ Programming Assignments

All Microsoft Visual C++ programming assignments are implemented as Win32 Console Applications which execute in a DOS shell within Windows NT. Windows programming is not introduced in Computer Aided Measurement and Control due to time considerations, although this feature is available for students to use in later course work and design projects. Although most experiments are DSP-related, they have been designed to provide students with practice in each of the important C++ programming topics.

Lab #1: Editing, Compilation, And Execution of Microsoft Visual C++ Source Code

This laboratory acquaints students with the use of the Microsoft Visual C++ editor and compiler. An example program is provided, which the students type into the editor verbatim, compile, and execute in a DOS shell. Primary programming topics: (1) exposure to the Microsoft Visual C++ integrated development environment (IDE); and (2) introduction to the `main()` function and elementary C++ syntax.

Lab #2: Circuit Simulation of the Maximum Power Transfer Theorem

Students verify the maximum power transfer theorem by examining a resistive Thevenin equivalent circuit and calculating the resistive load which results in maximum power delivery. A loop is used to compute the power dissipated for a range of load resistances, and an output text file is written with a list of these values. Primary C++ programming topics: (1) `while` loop construct; (2) C++ mathematical function syntax; (3) keyboard input; and (4) text file output.

Lab #3: Computer Generation of Uniform and Gaussian Random Variables

Uniform- and Gaussian-distributed random variables are generated through computer simulation. Students calculate the mean, variance, and standard deviation of each random variable, and write an output text file containing a histogram of the samples which is plotted using Microsoft Excel. Primary C++ programming topics: (1) `for` loop construct; (2) file I/O functions; and (3) mathematical operators. Primary DSP topics: (1) data analysis; and (2) the computation of signal statistics.

Lab #4: Implementation of a Digital Function Generator

Digital sine, square, and triangular wave sample values are generated using the discrete time domain mathematical representations of the periodic signals. The digital function generator allows the user to select the DC voltage offset, amplitude, period, and sampling interval for each discrete signal. Primary C++ programming topics: (1) modular programming using functions; and (2) loop constructs. Primary DSP topics: (1) time domain mathematical representation of discrete deterministic test signals; (2) introduction to signal sampling and sampling rate selection; and (3) preparation for understanding D/A and A/D conversion.

Lab #5: Introduction to Digital Signal Processing

Simple finite impulse response (FIR) digital filters are implemented and applied to the sampled sine, square, and triangular signals generated in Lab #4. In addition, PCM-encoded (pulse code modulated) sample values contained within a WAV digital audio file are filtered; students may listen to the original input signal and the digitally filtered output signal to hear the effects. Three-tap lowpass and two-tap highpass filter implementations are applied to the sampled data. As the programs become more involved in the course, a C++ source code framework is provided by the instructor in which the students are required to fill in missing functions and essential algorithms. The source code framework handed out to the students for Lab #5 is included as an appendix at the end of this paper. Primary C++ programming topics: (1) object-oriented programming; (2) classes and objects; and (3) 1-D arrays. Primary DSP topics: (1) 1-D FIR digital filters; (2) exposure to the WAV digital audio format; and (3) discussion of digital audio sampling rates.

Lab #6: Data Acquisition and Digital Control Using the National Instruments Lab-PC-1200 Card

Students are introduced to the National Instruments Lab-PC-1200 card for use in data acquisition, computer aided control, signal generation, and digital filtering. Most of the C++ source code is provided, so that the students can concentrate on learning how to operate the instrumentation equipment. Exposure to D/A conversion is provided, in which functions written in Lab #4 are used to generate digital sine, square, and triangular waves; the appropriate NI-DAQ library functions are called to convert the digital samples to analog signals using the Lab-PC-1200 card; and the periodic test signals are viewed on the oscilloscope screen. Exposure to A/D conversion and data acquisition is also provided, in which the students use the computer to sample a 100 Hz sine wave produced by an analog function generator. Different sampling rates are applied to the analog signal by the students, and the corresponding digital samples are written to an output text file for visualization in Microsoft Excel. Aliasing artifacts are identified, and practical methods of selecting a proper sampling rate for a signal of a particular fundamental frequency are discussed. Primary C++ programming topic: 1-D arrays. Primary DSP topics: (1) selection of data acquisition cards; (2) introduction to NI-DAQ software library functions for A/D and D/A conversions; and (3) sampling rate selection and the visualization of aliasing artifacts.

Lab #7: Introduction to Digital Image Processing

In each undergraduate course taught by the author, at least one experiment is provided which deals with image processing to give students an introduction to his research interests. Students receive a

source code framework which implements several image processing algorithms that may be applied to grayscale (8-bit) images. Students write functions to compute a digital negative image; calculate the mean, standard deviation, minimum, and maximum of the image samples; and perform lowpass and highpass FIR filtering using 3 x 3 element convolution masks. The portable gray map (PGM) image format is used, due to its simplicity. Primary C++ programming topics: (1) multidimensional (2-D) arrays; and (2) nested `for` loops. Primary DSP topics: (1) digital image processing; (2) exposure to the PGM digital image format; (3) 2-D FIR digital filters; and (4) data analysis and the computation of signal statistics.

4.2. National Instruments LabVIEW Programming Assignments

The final two assignments in the course utilize LabVIEW. A comparison between text-based C++ programming and graphical programming using LabVIEW is emphasized. Students are taught that the LabVIEW Front Panel is analogous to the input and output parameters of a C++ function, while the LabVIEW Block Diagram is similar to the actual C++ source code which implements the function.

Lab #8: Introduction to National Instruments LabVIEW

Students write a simple LabVIEW VI to become familiar with the Front Panel and Block Diagram interfaces, and they learn basic LabVIEW programming by experimenting with example VIs included within the Student Edition of LabVIEW. Primary LabVIEW programming topics: (1) exposure to the LabVIEW Front Panel and Block Diagram interfaces; (2) writing an icon-based graphical program; (3) LabVIEW For Loop construct; and (4) LabVIEW Case Structure construct. Primary DSP topic: exposure to an alternative software package used in signal processing system development.

Final Project: Implementation of a Digital Filter Using Microsoft Visual C++ and LabVIEW

A five-tap FIR digital lowpass filter is implemented and applied to actual voltage waveforms. The digital filter is implemented twice, first using Microsoft Visual C++ and a second time using National Instruments LabVIEW. In this manner, a direct comparison between these two instrumentation tools with respect to ease of algorithm implementation and user-friendliness can be made. Primary LabVIEW programming topics: (1) For Loop construct; and (2) Shift Register construct; Primary DSP topics: learning how to access the I/O channels of the Lab-PC-1200 DAQ board through LabVIEW virtual instruments.

5. Conclusion and Course Evolution

Computer Aided Measurement and Control is a required course for electrical engineering sophomores at the University of North Dakota. It is a rather unique course, in that students are taught many practical DSP concepts by completing the programming assignments. Microsoft Visual C++ and National Instruments LabVIEW are used to implement a digital signal generator, examine the output of 1-D and 2-D FIR digital filters, calculate statistics for data analysis purposes, and perform data acquisition and digital control applications by interfacing a DAQ card. This course is a modification of the "DSP First" concept, in that students are introduced to time domain sampling and real-world applications of digital signal processing at an early stage in the curriculum; however, they are not exposed to the Z-transform and frequency domain analysis techniques associated with the design of digital filters.

Computer Aided Measurement and Control has evolved from its origination as a course in C programming with simple data acquisition experiments to a more comprehensive course in C++ and LabVIEW programming with instrumentation assignments that provide students an intuitive understanding of DSP concepts. In future iterations of the course, 12 weeks will be devoted to C++ and

four weeks will be devoted to LabVIEW, as this allowed enough time to teach the important concepts in each language. Once the students are proficient in using C++ selection structures and loop constructs, they quickly learn how to use the analogous constructs in LabVIEW. The DSP-related programming assignments will be refined, with the possible addition of experiments in signal construction using the truncated Fourier series, utilizing the fast Fourier transform (FFT) to compute the frequency spectrum of a sampled signal, infinite impulse response (IIR) digital filtering, and the real-time processing of digital speech and music signals.

References

[1] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Prentice-Hall, Inc., 1989.

[2] S. K. Mitra, *Digital Signal Processing: A Computer-Based Approach*. The McGraw-Hill Companies, Inc., 1998.

[3] K. Steigletz, *A DSP Primer: With Applications to Digital Audio and Computer Music*. Addison-Wesley Publishing Company, 1996.

[4] R. G. Lyons, *Understanding Digital Signal Processing*. Addison Wesley Longman, Inc., 1997.

[5] J. H. McClellan, R. W. Schafer, and M. A. Yoder, "Experiences in Teaching DSP First in the ECE Curriculum." Presented at the 1997 American Society for Engineering Education Annual Conference, Session 1220 Digital Signal Processing, (Milwaukee, WI), June 15-18, 1997.

[6] National Instruments Corporation, *Lab-PC+ User Manual: Low-Cost Multifunction I/O Board for ISA.* Part Number 320502B-01, June 1996 Edition.

[7] National Instruments Corporation, *NI-DAQ Function Reference Manual for PC Compatibles*, Version 4.8. Part Number 320499C-01, May 1995 Edition.

[8] L. K. Wells, *The LabVIEW Student Edition User's Guide*. Prentice-Hall, Inc., 1995.

[9] L. K. Wells and J. Travis, *LabVIEW for Everyone: Graphical Programming Made Even Easier*. Prentice-Hall, Inc., 1996.

[10] J. R. Hanly, *Essential C++ for Engineers and Scientists.* Addison Wesley Longman, Inc., 1997.

RICHARD R. SCHULTZ received the B.S.E.E. degree (summa cum laude) from the University of North Dakota in 1990, and the M.S.E.E. and Ph.D. degrees from the University of Notre Dame in 1992 and 1995, respectively. He joined the faculty of the Department of Electrical Engineering at the University of North Dakota in 1995, where he is currently an Assistant Professor. In 1996, Dr. Schultz received a National Science Foundation Faculty Early Career Development (CAREER) award to integrate his image processing research and educational activities. Dr. Schultz is a member of the ASEE, IEEE, SPIE, Eta Kappa Nu, and Tau Beta Pi. His current research interests include digital signal, image, and video processing.

Appendix

Lab #5: Introduction to Digital Signal Processing
C++ Source Code Framework

```
/*********************************************************************
   EE 304 Computer Aided Measurement and Control
   University of North Dakota
   Fall Semester 1997

   Author:         R. R. Schultz (rschultz@nyquist.ee.und.nodak.edu)
   File Name:      lab05.cpp
   Lab Number:     5
   Lab Title:      Introduction to Digital Signal Processing
                   C++ Source Code Framework
*********************************************************************/
#include <iostream.h>
#include <iomanip.h>
#include <fstream.h>
#include <math.h>


// Macros
#define SQ(x) ((x)*(x))


// Constants
const int LP=1;
const int HP=2;
const int STATS=3;
const int NUM=8192;        // maximum number of samples allowed
const double PI=3.1415927;


// Function prototypes
void GetFileName(char []);
int DspOperation(void);


// Class definition
class Wave {
   private:
      int N;
      double x[NUM];

   public:
      Wave(void);
      Wave(int, double []);
      int AccessNumberSamples(void);
      void LowpassFilter(void);
      void HighpassFilter(void);
      void CalculateStatistics(double &, double &, double &, double &);

   friend void WaveRead(char [], Wave &);
   friend void WaveWrite(char [], Wave &);
};


// Default constructor function
Wave::Wave(void)
{}


// Constructor function
Wave::Wave(int num, double array[])
```

```
{
   // **** SOURCE CODE MISSING !!!! ****
}


// Accessor function
int Wave::AccessNumberSamples(void)
{
   // **** SOURCE CODE MISSING !!!! ****
}


// Lowpass filter
void Wave::LowpassFilter(void)
{
   double y[NUM];

   // Lowpass digital filter coefficients:
   //    y[n] = a1*x[n-1] + a2*x[n] + a3*x[n+1]
   //    a1 = 0.3333
   //    a2 = 0.3333
   //    a3 = 0.3333

   // **** SOURCE CODE MISSING !!!! ****

   // Output values
   for (n=0; n<N; n++)
      x[n] = y[n];
}


// Highpass filter
void Wave::HighpassFilter(void)
{
   double y[NUM];

   // Highpass digital filter coefficients:
   //    y[n] = a1*x[n-1] + a2*x[n]
   //    a1 = -1.0
   //    a2 = 1.0

   // **** SOURCE CODE MISSING !!!! ****

   // Output values
   for (n=0; n<N; n++)
      x[n] = y[n];
}




// Compute signal statistics
void Wave::CalculateStatistics(double &mean, double &sd, double &min, double &max)
{
   // **** SOURCE CODE MISSING !!!! ****
}


// Signal input
void WaveRead(char filename[], Wave &waveobj)
{
   double sample;

   ifstream infile(filename, ios::in);
```

```
   int num=0;
   for (infile >> sample; !infile.fail(); infile >> sample) {
      waveobj.x[num] = sample;
      num++;
   }

   waveobj.N = num;
}


// Signal output
void WaveWrite(char filename[], Wave &waveobj)
{
   ofstream outfile(filename, ios::out);

   outfile << setiosflags( ios::fixed | ios::showpoint ) << setprecision(6);

   for (int n=0; n<waveobj.N; n++)
      outfile << setw(12) << waveobj.x[n] << endl;
}


int main(int argc, char **argv)
{
   Wave waveobj;

   char infilename[81];
   char outfilename[81];

   double mean, sd, min, max;

   // Get name of input file
   cout << "Name of input file:" << endl;
   GetFileName(infilename);

   // Read input signal
   WaveRead(infilename, waveobj);

   // Determine signal operation
   int dsp = DspOperation();

   // Process the data
   switch (dsp) {
      case LP:
         waveobj.LowpassFilter();
        break;
      case HP:
         waveobj.HighpassFilter();
         break;

      case STATS:
         waveobj.CalculateStatistics(mean, sd, min, max);
         cout << "Signal Statistics:" << endl;
         cout << "\t Number of Samples:   " << waveobj.AccessNumberSamples()
              << endl;
         cout << "\t Mean:                " << mean << endl;
         cout << "\t Standard Deviation:  " << sd << endl;
         cout << "\t Minimum:             " << min << endl;
         cout << "\t Maximum:             " << max << endl;
         cout << endl << endl;
         break;

      default:
         cout << "Should not be possible." << endl;
   }

   // Get name of output file
```

```
   cout << "Name of output file:" << endl;
   GetFileName(outfilename);

   // Write processed signal to a file
   WaveWrite(outfilename, waveobj);

   // Return successfully
   return 0;
}


// Get name of signal file
void GetFileName(char filename[])
{
   cout << "\tFile name (less than 80 characters) >>> ";
   cin >> filename;
}


// Get DSP operation
int DspOperation(void)
{
   int op;

   // **** SOURCE CODE MISSING !!!! ****

   return op;
}
```