

PREPARATION OF A VIRTUAL TOOLKIT FOR COMMUNICATION SYSTEMS

**Murat Tanyel, Kathrine Nguru
Dordt College**

Abstract

Communication Systems is offered at senior level as a three-credit, one semester course at Dordt College. Due to the number of topics covered, it turns out to be a very compact course with little time for experimentation. However, the course content is rich with processes that are best described by block diagrams which should come alive as students implement these processes and have a chance to test signals at each block. In order to facilitate such an experimental environment with limited institutional resources and student time for hardware implementations, a virtual toolkit of computer simulations was developed using LabVIEW. LabVIEW's graphical interface with live help windows and other visual feedback rendered this project achievable and enjoyable even for a freshman student programmer. This paper will recount the project both from a professor's point of view and from that of a freshman student, who did not know about the programming language nor about the processes she was to simulate at the onset of the project. The paper will also describe the toolkit prepared to demonstrate a wide variety of topics covered in a typical communication systems course and present some examples of the virtual instruments (VIs) developed.

I. Introduction

The undergraduate engineering education environment is very different from the environment that the senior author has experienced. The personal computer has now become ubiquitous and more powerful than the mainframe computers of the late 70's and early 80's while the educational utilization of the computer has been transformed from writing specific programs in FORTRAN for carrying out calculations for individual problems to mastering specialized applications for simulations or for data acquisition/processing. With the proliferation of special purpose applications, undergraduate electrical engineering courses have been enriched with computer simulations with many textbooks incorporating some form of computer-aided tool¹. Simulations have made 'what if' type investigations possible even in situations where accessing the actual hardware to experiment with is, for one reason or another, impractical. Whereas simulations fail to achieve some of the goals of hardware hands-on exercises, they still present much more powerful alternatives to static textbook examples, allowing playful interaction with the simulated system.

One of the summer 2001 projects on Dordt College campus was the development of a software toolkit that may be used with a typical communication systems course. This project was undertaken upon the enthusiastic suggestion of the senior *Communications Systems* students in spring '01 who had taken *Digital Signal Processing (DSP)* the previous semester. LabVIEW was employed as the tool for data processing in the DSP class and the experience is described in an article presented last year¹ with a follow up this year². This paper will describe the development of the toolkit specific for *Communications Systems*. We will report on the progress of the project as well as the outcome. The fact that the student programmer in this project had only completed one semester of her freshman year presented certain challenges to the programmer while making the completion of the project all the more rewarding. Section II will present an overview of the environment: the target course and the available resources. Section III will report on the work completed, section IV will describe some examples on how the software may be utilized and section V will have concluding words and a discussion on future development.

II. The Overall Environment: The Course and the Resources

The engineering department at Dordt College and its offerings are described in ¹ and ³. EGR 363, *Introduction to Communication Systems*, is a class elective in the spring semester of the senior year. For the last two years, EGR 363 has been chosen over its alternative, *Introduction to Power Systems*. EGR 363 is a three-credit class that meets for three 50-minute periods a week. The mode of instruction employs active learning in which students are required to read the topic of the day prior to coming to class and the class period is utilized to clear concepts, emphasize important points and to study practical applications. The textbook for this offering was Couch's *Digital and Analog Communication Systems* ⁴. This textbook was chosen because of its good pace through different communication techniques as well as its many examples involving MATLAB and MATHCAD. The topics covered in this offering were: properties of signals and noise, Fourier transform and spectra, power spectral density and autocorrelation, Fourier series, the discrete Fourier transform, pulse amplitude modulation, pulse code modulation, line codes and spectra, intersymbol interference, differential pulse code modulation, delta modulation, AM, FM and digital modulated systems.

Students who take EGR 363 will usually have chosen *DSP* the previous semester and will be familiar with LabVIEW by the time they take *Communication Systems*. In order to capitalize on their recently acquired programming skills, LabVIEW version 5.1, then available on the departmental lab computers, was chosen as the programming language for this toolkit. Although the author of the textbook provides certain MATLAB and MATHCAD files, developing the toolkit in a language which the students understand well and seem to like would offset the disadvantage of ignoring these utilities provided.

A position for a programmer was advertised in various engineering classes for which there was only one applicant. The applicant was in her freshman year, having started in the spring semester.

However, the student already had a degree in business and engineering would be her second degree. Given the student's maturity and her willingness to learn LabVIEW, she was hired as a summer workstudy student to be the programmer. Although she was happy to be given the opportunity, she had some concerns about the project: "I had never heard of, let alone worked with, LabVIEW before then. All I knew was ... that it was a programming environment which uses a graphical programming language to create programs in block diagram form. My major concern was that the programming language I had learnt was somewhat outdated and I had no knowledge of the programming languages that are being used in its place. Furthermore, I had only had one semester of engineering and certainly did not know enough to write a program for a senior level course. Besides being a freshman, my emphasis was mechanical engineering but the programs I would be writing would be mostly for electrical engineers. But I was assured that it would not be hard to learn." In order to address some of these concerns, we started out with a one-week training course in LabVIEW with her and some other students, basically covering all the essential programming skills from a freshman – level textbook on LabVIEW⁵.

III. Virtual Instruments Completed

In the textbook used to train the students, one exercise on loops is writing a VI that "generates" a sine wave. In other words, the students are asked to generate a number of data points using the sine block of LabVIEW and plot these data points. That simple exercise gave rise to our first VI in the toolkit, namely a signal generator. Figure 1 depicts the front panel of this signal generator. This particular version is for the students' visualization of the various periodical signals that may be selected, and is a top-level, or stand-alone VI. A similar one, named `Signal Generatorsub.vi` is available as a subVI, which is LabVIEW's term for subroutines. The subVI has been modified such that all the front panel controls of Fig. 1 as well as the frequency of the periodic waveform may be input into an icon representing the subVI. The data generated is available in the form of an array as well as the sampling frequency imposed by the parameter selections may be obtained from the same icon. Thus `Signal Generatorsub.vi` has become our test signal generator in writing other simulations.

Other VIs written are listed in alphabetical order:

AM.vi modulates the baseband signal (an array input) using the amplitude modulation technique.

Its other inputs are the amplitude of the carrier, the sampling frequency of the baseband signal, and the carrier frequency. The outputs are a graph of the AM signal, a graph of the input signal and array of the values of the modulated signal.

AM DSB-SC.vi is similar to `AM.vi` but uses the double sideband suppressed carrier AM technique.

BPSK.vi simulates the transmission of binary information over analog lines using binary phase-shift keying (BPSK). Its main input is the 'input bit' and the outputs are a graph of the BPSK

waveform and synchronization 'LED' that marks the beginning of each bit. This VI is for visualization only.

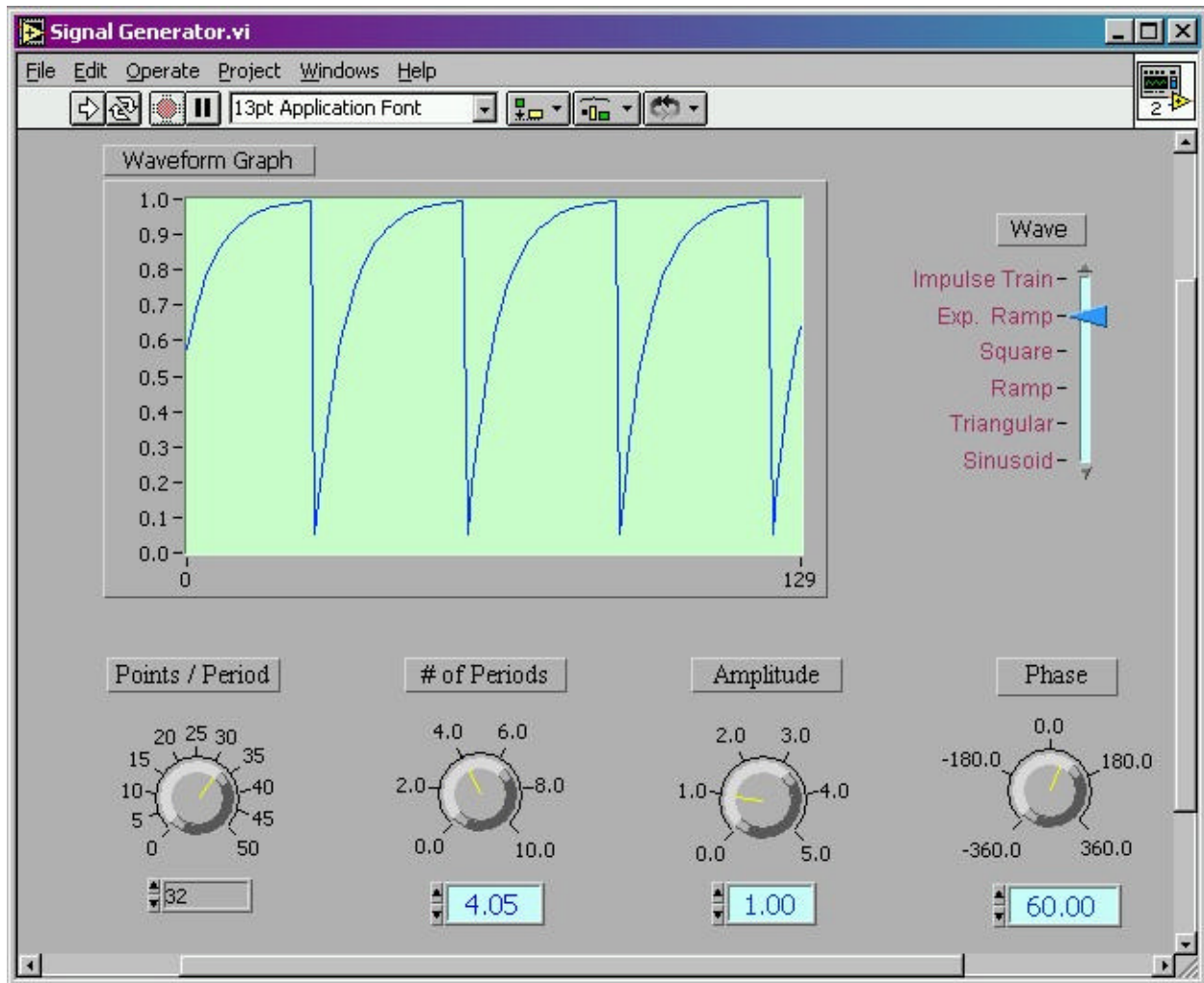


Figure 1: The Signal Generator VI, the first VI developed for this project.

BPSKsub.vi is the version of **BPSK.vi** that has been adapted such that it can be incorporated in other simulations' block diagrams with an input array of binary data and an output array of BPSK waveform data.

ENV_DET.vi simulates an envelope detector converting the input bandpass waveform into an output baseband waveform.

FlattopPAMr1.vi simulates Pulse Amplitude Modulation using instantaneous sampling.

FM.vi modulates an input signal using the frequency modulation technique.

FM_DET.vi simulates frequency demodulation using slope detection (details may be obtained from the textbook⁴, pp. 268-269). It makes use of LabVIEW's differentiator VI and our ENV_DET.vi.

Freq1.vi estimates the frequency of a periodic waveform by zero-crossing detection.

FSKv3.vi simulates the transmission of binary information over analog lines using frequency-shift-keying. It is similar to BPSK.vi in operation and is for visualization only.

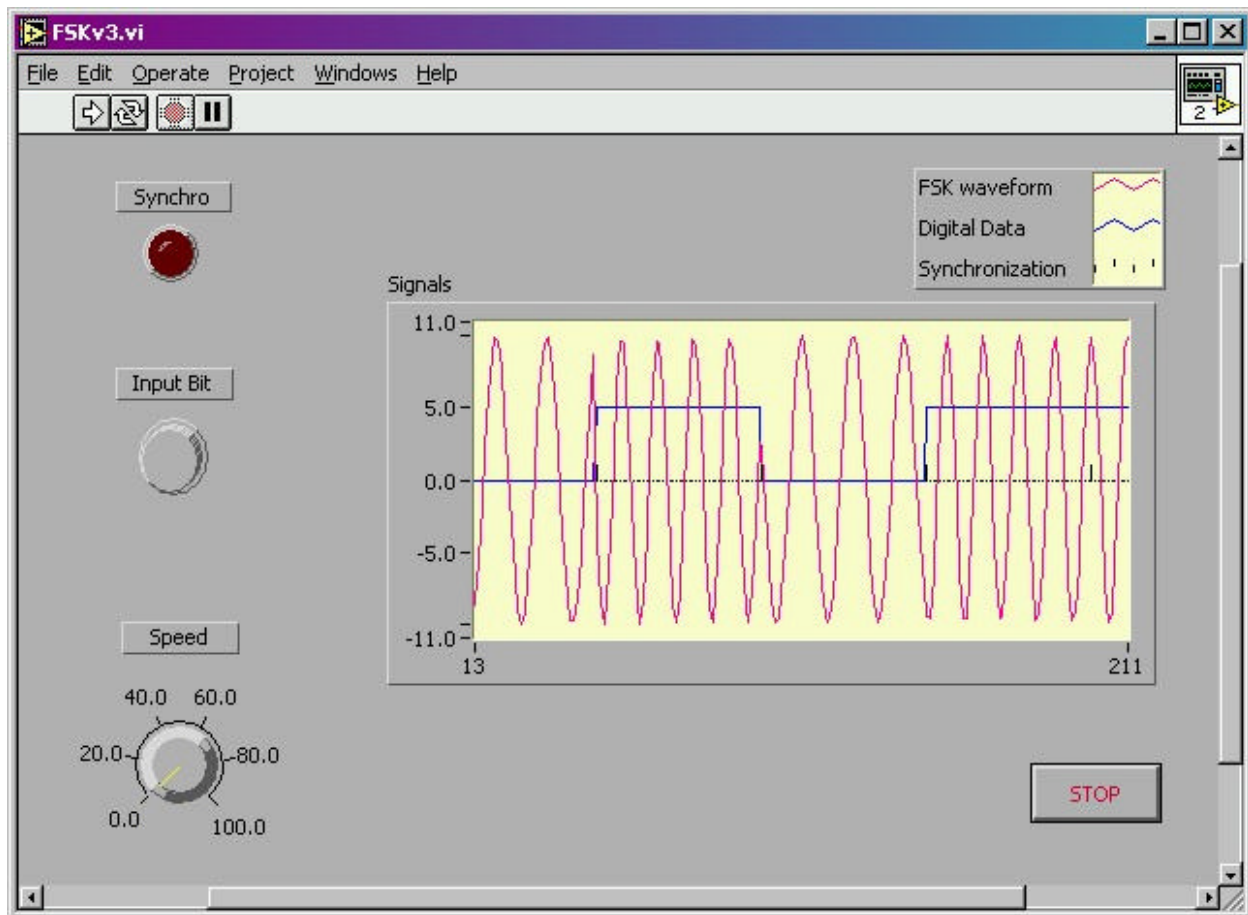


Figure 2: Operation of FSKv3.vi, demonstrating the frequency-shift-keying technique.

FSKv3subv3.vi is the version of FSKv3.vi that has been adapted to be incorporated into other diagrams.

Integralsub.vi calculates the indefinite integral of its input waveform using either the right endpoint or the left endpoint or the trapezoidal approximation.

NaturalPAM.vi simulates Pulse Amplitude Modulation using natural sampling.

OOK.vi simulates the transmission of binary information on analog lines using on-off keying.

PM.vi modulates an input signal using the phase modulation technique.

PROD_DET.vi simulates the function of a product detector.

IV. Some Examples

Figure 2 depicts the operation of the `FSKv3.vi`. This VI simulates the operation of a modem using the frequency-shift keying method. Its purpose is to display the analog output of the modem dynamically as the student inputs marks (binary 1's) and spaces (binary 0's) using the toggle button "Input Bit". The value of "Input Bit" is read at each synchronization instant, indicated by the blinking of the LED named "Synchro". The synchronization is based on a data rate of 300 bits/s. The graph displays a running window on the binary input (blue line), the corresponding sinusoid generated (red line) and the synchronization, which marks the beginning of a new bit. The "Speed" knob adjusts the speed with which the graph progresses by introducing a delay of variable milliseconds in the while loop that generates and graphs the data points, hence this simulation is not real time. The VI runs continuously until the stop button (lower right) is depressed. This simulation is based on the Bell-Type 103 FSK Modem, for which the transmit frequencies of the originate modem are 1,270 Hz for the mark and 1,070 Hz for the space⁴. However, with the window and graph size chosen, the difference between these two frequencies is not visually detectable and we have decided to change the mark and space frequencies to 1,370 and 970 Hz respectively.

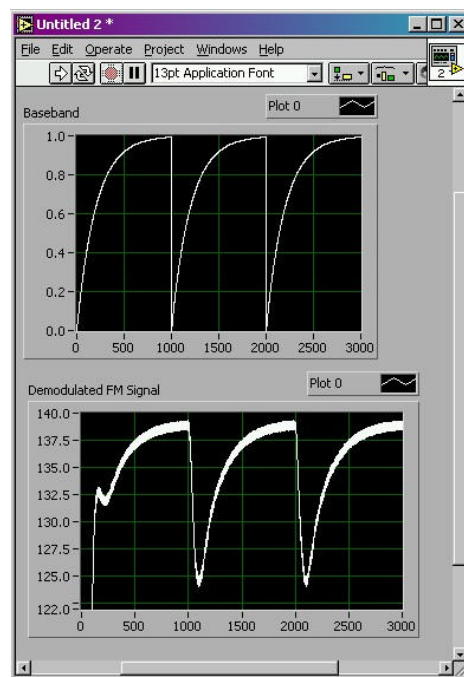


Figure 3: The front panel of a quick VI that demonstrates FM.

Figure 3 depicts the front panel of a simulation that utilizes a number of the subVIs from the toolkit. The Untitled 2 VI generates a baseband signal, modulates it utilizing the FM technique and then demodulates the result. The signals shown are the baseband message signal and the demodulated signal. A quick observation of the corresponding block diagram (figure 4) reveals that `Signal Generatorsub.vi`, `FM.vi` and `FM Det.vi` are utilized in this simulation.

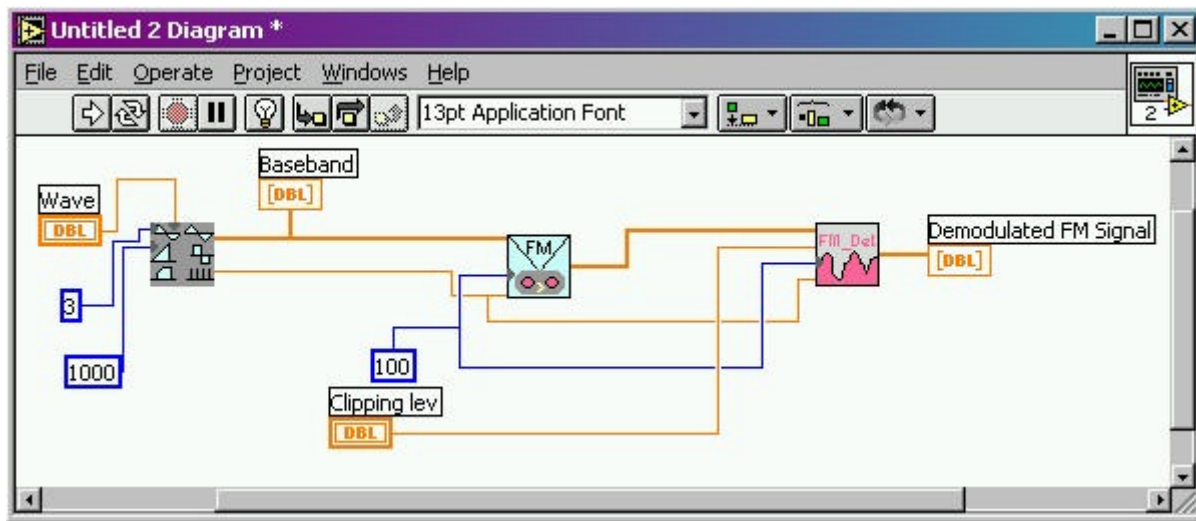


Figure 4: The block diagram of a quick VI that demonstrates FM.

`Signal Generatorsub.vi` (the leftmost block in figure 4) generates the baseband signal. The type of wave is input from the front panel of the calling VI (Untitled 2). Other inputs are a constant (3) for the number of periods to be generated and another constant (1000) for the number of points per period. The frequency of the periodic waveform is not specified, which implies that the default value of 1 is used. The outputs from this block are the data points (a numeric array of size 3000) and the sampling frequency calculated from the number of points per period and the frequency (which turns out to be 1000 for this example). The output array is graphed (the block labeled "Baseband" in figure 4) and then input into the block that performs FM (`FM.vi`). Other inputs of `FM.vi` are specified by a constant for the carrier frequency (100) and the sampling frequency, output by `Signal Generatorsub.vi`. The unspecified inputs of `FM.vi` (such as the amplitude and the modulation index) carry the default values of 1 and 0.1, respectively. The output of `FM.vi` is an array of data points representing the frequency modulated signal. This output is connected as the input array of `FM Det.vi`. Other inputs to this block are the carrier frequency and the sampling frequency. The clipping level can be adjusted from the front panel of the calling VI to study this parameter's effects. Finally, the output of the demodulator is graphed (labeled 'Demodulated FM Signal').

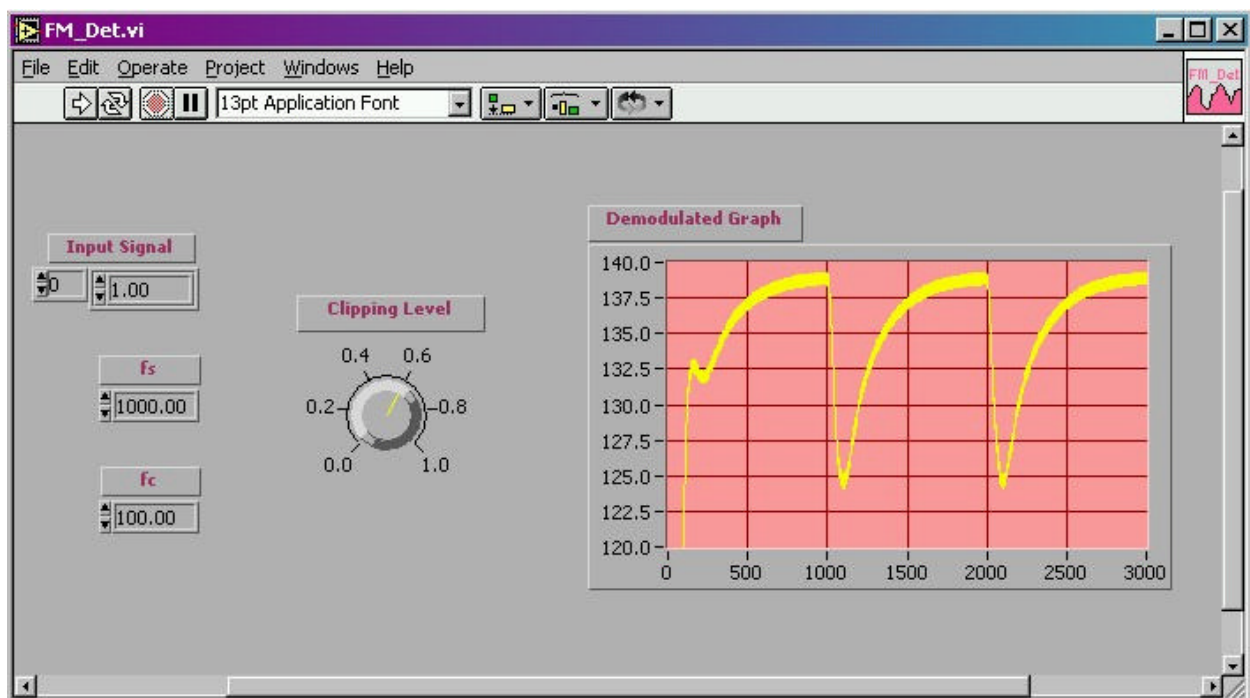
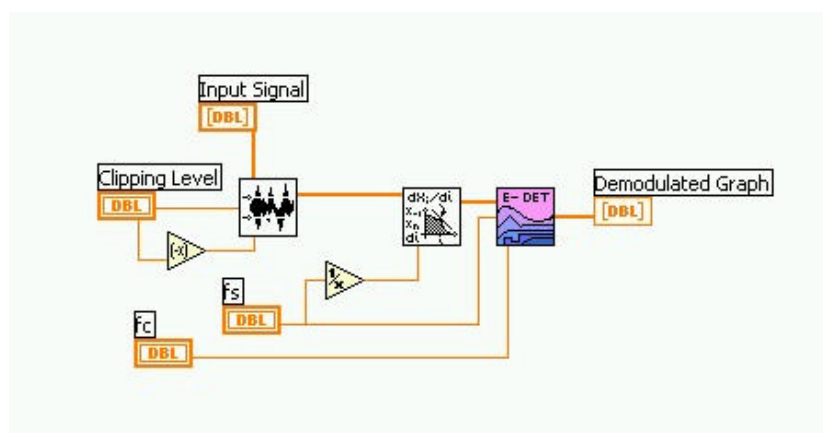
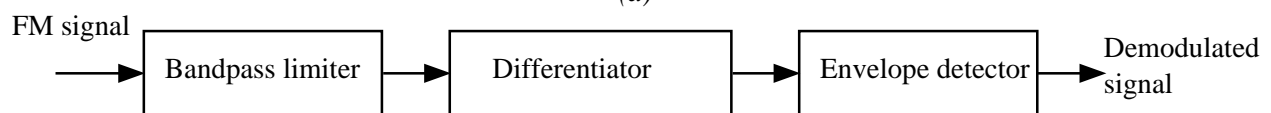


Figure 5: The front panel of *FM_Det.vi* after it has been called in our exercise.



(a)



(b)

Figure 6: (a) The block diagram of *FM_Det.vi*. (b) The block diagram that describes FM detection in a textbook.

The walk through the block diagram for this exercise demonstrated how the various VIs in this toolkit may be used to implement a simulation. We will now concentrate on `FM Det.vi` to demonstrate our utilization of modular programming in the development of the toolkit. The front panel of this VI, after it has been called in our exercise, is shown in figure 5 and has the input and output values from the example in figure 3. We note that the 0th element of the “Input Signal” array has a value of 1.00, the sampling frequency (“fs”) is read to be 1000, the carrier frequency is input as 100 and a clipping level of 0.6 has been selected.

We note that the block diagram for `FM Det.vi` (see Fig. 6-a) is very similar to the diagram on page 269 of the textbook⁴ describing the process of frequency modulation using slope detection (see Fig. 6-b). The LabVIEW blocks of Fig. 6-a, corresponding to the block diagram in Fig. 6-b are: `Y[i]=Clip{X[i]}`.vi that performs the function of the limiter, `Derivative x(t)`.vi which performs differentiation and `ENV_DET.vi` which performs envelope detection. The first two blocks are part of the full development version of LabVIEW 5.1 and the last block, the envelope detector, is part of our toolkit.

After describing some of the virtual instruments developed for this project, we would like to conclude our examples with one that demonstrates how the different tools in our toolkit may be used in a *Communication Systems* class in a way that “brings life” to these block diagrams and allows for playful interaction with the simulated system. Our example will simulate a typical textbook example that demonstrates what happens to a baseband signal spectrum after modulation. In most textbooks, baseband spectra are represented by an ideal lowpass spectrum. We shall apply double sideband, suppressed carrier amplitude modulation (DSB-SC AM) to such a baseband signal and observe the spectra of the baseband signal and the modulated signal as parameters are changed.

From the duality property of Fourier Transforms, students should be aware that an ideal lowpass spectrum originates from a *sinc* in time domain, which is not a physically realizable waveform according to criterion # 1 on page 34 of Couch⁴, which states that a physical waveform should last over a finite time interval. However, an approximation for the ideal lowpass spectrum may be obtained by a truncated *sinc*:

$$w(t) = \begin{cases} \text{sinc}(2\pi ft), & |t| < T \\ 0, & \text{otherwise} \end{cases} \quad (1).$$

As T increases, the spectrum approximates the ideal lowpass spectrum better and the parameter f determines the bandwidth of the lowpass spectrum. Figure 7 depicts the front panel of a virtual instrument that generates $w(t)$, displays it and its magnitude spectrum as well as the magnitude spectrum of the DSB-SC AM signal. The inputs T and f_{sinc} control the parameters T and f of eq. 1, while f_c determines the carrier frequency of DSB-SC AM. The front panel shows the result of a run where $T = 1.94$ sec, $f = 6.14$ Hz and $f_c = 33.39$ Hz. Although this front panel is

also static and hence there seems to be no improvement over a textbook example, it should be noted that this is a capture of the front panel at the end of a *continuous run*. In LabVIEW, a virtual instrument may be run continuously, which means that the input parameters will be monitored continuously and the output updated as the input parameters are changed. When this VI is run continuously, the student can observe that increasing the parameter T approximates the lowpass spectrum better, increasing or decreasing f_{sinc} makes the bandwidth wider or narrower, and increasing or decreasing the carrier frequency will move the sidebands higher or lower in the frequency band, all achieved by adjusting the “knobs” (f_{sinc} and f_c) and playing with the sliding switch (T). In fact, the reason why the parameters T , f and f_c seem to have arbitrary values (1.94 s rather than 2 s, 6.14 Hz rather than 6 Hz and 33.39 Hz rather than 33 Hz or 34 Hz) is the fact that the front panel was captured after “playing” with the parameters while the VI ran continuously.

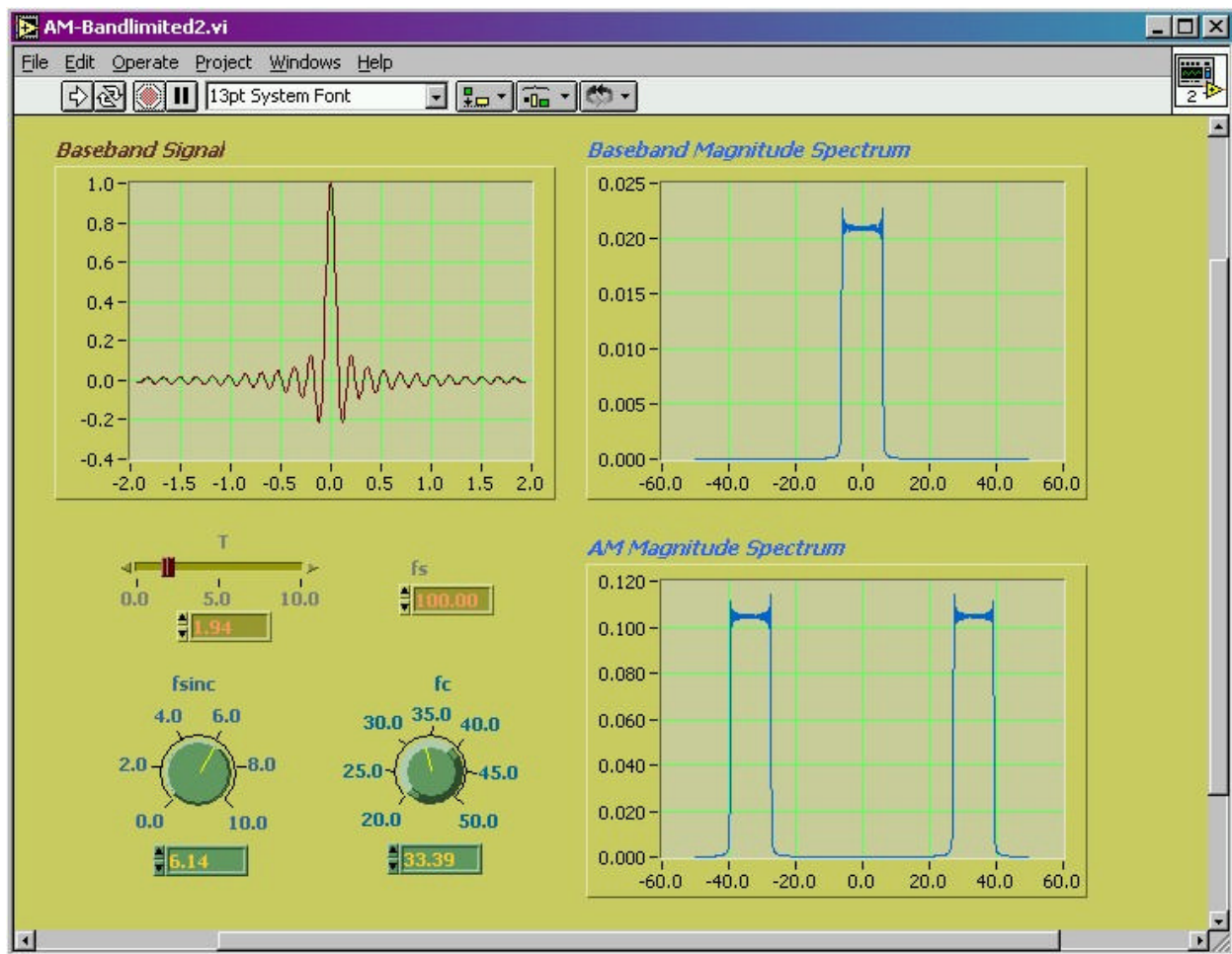


Figure 7: The front panel of a VI that demonstrates the effect of DSB-SC AM applied to a low-pass baseband signal

Before we end the discussion on this example, we should briefly explain how it was achieved. Figure 8 displays the block diagram that gives rise to this example. The blue block whose inputs are T , f_s and f_{sinc} , generates an array for $w(t)$ utilizing eq. 1, its outputs (the array, the beginning time and Δt for proper graphing) determine the graph labeled “Baseband Signal”. The magnitude spectrum of the baseband signal is determined by $\text{MagSpect} .vi$ (whose icon spells $|X(f)|$), borrowed from our DSP toolkit²). Finally the output of the *sinc* generator is applied to the $\text{AM DSB-SC} .vi$ of our toolkit. We should note that since we are interested in magnitude spectra only in this example, we have not made any effort to align the ‘0’ time instance of the *sinc* pattern and the carrier cosine.

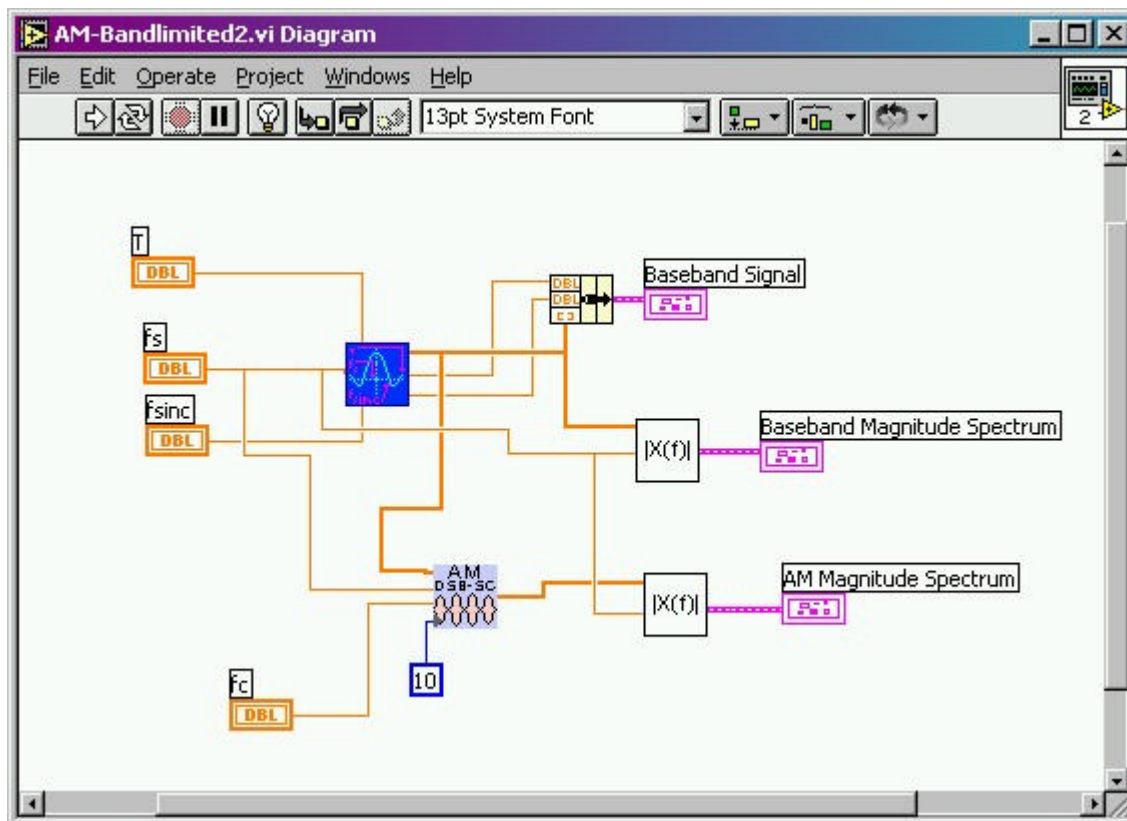


Figure 8: The block diagram that generates the output of Fig. 7.

V. Conclusions and Further Development

In a previous paper, we wrote: "However, the instrument-like user interface of LabVIEW, its abundance of analysis VIs make it very attractive and inviting to implement most DSP routines"¹. That paper also argued that for processes best explained by block diagrams, a graphical language such as LabVIEW is very intuitive to use. This project confirmed the same conclusions for communication systems. Although the student who worked on these VIs had completed only one semester of the freshman year in engineering, she was able to handle all the algorithms with close

and focused supervision. For the project supervisor, it was closer to teaching a course than to supervising a senior design project team. The programmer was appreciative of the experience: “I started learning the basics of LabVIEW and writing simple programs to get a complete grasp of the programming environment. We began working on one project at a time and each time we would build on it until the desired program was written. From this opportunity I not only learnt how to use LabVIEW but I also learnt some valid information that I know I will need for my future classes. I learnt about frequency spectra, modulation of signals, transmission of binary information and different kinds of waveforms and so much more! I was able to get hands-on knowledge on some of the things that electrical engineers do.” The project supervisor regarded this experience as investment for other LabVIEW programming projects for the department. In fact, the second phase of the project involved writing IEEE 488 drivers for the Tektronix TDS 210 oscilloscopes where the supervisor could rely more on the student’s expertise acquired on this project.

The utilization of the tools in this toolkit is understood better with a rudimentary understanding of discrete signals. Since these are computer simulations, hence, discrete-time (sampled) realizations of analog processes, we can take two approaches: either make the sampling frequency transparent to the user and limit the range of frequencies that may be used in the simulations or make the sampling frequency as one of the inputs and expect the students to pay attention to the Nyquist rate as they implement these simulations. We have chosen the latter option since our students usually take *Communication Systems* after they have taken *DSP*. Furthermore, many *Communication Systems* textbooks deal with both digital and analog systems and we can expect that students will be familiar with concepts of sampling and aliasing.

Clearly, the list of VIs given in section III is not exhaustive of all routines that may be used in a *Communication Systems* course. Spring 2002 offering of EGR 363 will help expand the number of routines and help develop a manual similar to the one available for MATLAB⁶.

ACKNOWLEDGEMENTS: The LabVIEW software used in this course, as well as the NI-488.2 cards and Tektronix TDS 210 Digital Oscilloscopes in the electronics lab were purchased through a grant from Johnson Controls in Holland, MI. The authors would like to thank Johnson Controls for the upgrade of the electronics lab.

Bibliography

1. Viss, M. and Tanyel, M. “From Block Diagrams to Graphical Programs in DSP,” in Proceedings of the 2001 American Society for Engineering Education Annual Conference & Exposition, Albuquerque, NM, June 24 -27 2001.

2. Tanyel, M. "Enhancing the DSP Toolkit of LabVIEW", *accepted for publication in* Proceedings of the 2002 American Society for Engineering Education Annual Conference & Exposition, Montreal, Quebec, June 2002.
3. Mulder, J., Tukker, K., Verhoef, P., Fikkert, J. and Tanyel, M., "The Hug Machine: An Innovative Approach to Helping Our Autistic Neighbor", Proceedings of 60th Annual ASEE North Midwest Section Meeting, Winona, MN, Oct. 1998
4. Couch II, L. W., *Digital and Analog Communication Systems, Sixth Edition*, Upper Saddle River, NJ: Prentice Hall (2001).
5. Tanyel, M., *Engineering Explorations with LabVIEW*, Philadelphia, PA: Harcourt Brace Custom Publishers (1994).
6. Proakis, J. G., Salehi, M., *Contemporary Communication Systems using MATLAB*, Pacific Grove, CA: Brooks/Cole (2000).

MURAT TANYEL

Murat Tanyel is a professor of engineering at Dordt College. He teaches upper level electrical engineering courses. Prior to teaching at Dordt College, Dr. Tanyel taught at Drexel University where he worked for the *Enhanced Educational Experience for Engineering Students (E⁴)* project, setting up and teaching laboratory and hands-on computer experiments for engineering freshmen and sophomores. For one semester, he was also a visiting professor at the United Arab Emirates University in Al-Ain, UAE where he helped set up an innovative introductory engineering curriculum. Dr. Tanyel received his B. S. degree in electrical engineering from Bogaziçi University, Istanbul, Turkey in 1981, his M. S. degree in electrical engineering from Bucknell University, Lewisburg, PA in 1985 and his Ph. D. in biomedical engineering from Drexel University, Philadelphia, PA in 1990.

KATHRINE NGURU

Kathrine Nguru is a graphic design student at Dordt College. She holds a bachelor's degree in business administration from Daystar University, Nairobi, Kenya and switched to her new major after two semesters of introductory engineering curriculum.