

RADVIEW - ROTOR ACOUSTIC DATA ACQUISITION AND PROCESSING USING THE LABVIEW LANGUAGE

Andrew Johnson
Sundial Engineering
3073 Bateman Street
Berkeley, CA 94705

Cahit Kitaplioglu and Megan McCluer
NASA Ames Research Center
Moffett Field, CA 94035

Abstract

A multi-channel high-speed acoustic data acquisition and processing system has been implemented using a Macintosh PC plus digitizer cards and the National Instruments LabVIEW graphical programming language. The advantages of the system are low cost, fast prototyping, and highly maintainable code. The system has been used to replace FFT analyzers, with increased channel count at reduced cost. A preliminary version of the system has been used to acquire and process data during several full-scale and model-scale rotorcraft tests in the National Full-Scale Aerodynamics Complex (NFAC) at NASA Ames Research Center. The paper describes the hardware for implementing the system, presents an overview of the data acquisition and data processing algorithm, and details some of the advanced features of the LabVIEW language that were utilized in implementing the program. Plans for improvement and further development of the program are outlined.

Introduction

An important portion of current aeronautical research involves rotorcraft. This arises from the fact that rotorcraft that can operate close to population and business centers are a means to improve the overall efficiency of the national transportation system.

One of the most difficult problems faced by the aerospace industry today is the noise created by rotorcraft, particularly during terminal area activi-

ties. NASA has an active program of research to improve the public acceptability of rotorcraft near populated centers. One important part of this research activity is the rotor experiments that are performed in the large-scale wind tunnels at Ames Research Center. To measure the acoustic characteristics around the rotor, a number of microphones, either fixed or on a traverse mechanism, are placed in the test section (Fig. 1).

Figure 2 shows a typical rotor acoustic signature. Both the time history and the corresponding frequency power spectrum are shown. Such measurements are made at a number of locations in space in order to characterize the directions of most intense sound emission, and at a number of operating conditions in order to characterize the dependence of the sound level on basic parameters such as Mach number, thrust coefficient, and attitude. The main features to be noted are the harmonic content of the lower frequencies. In fact, the sound emitted by the rotor contains most of its energy at frequencies that

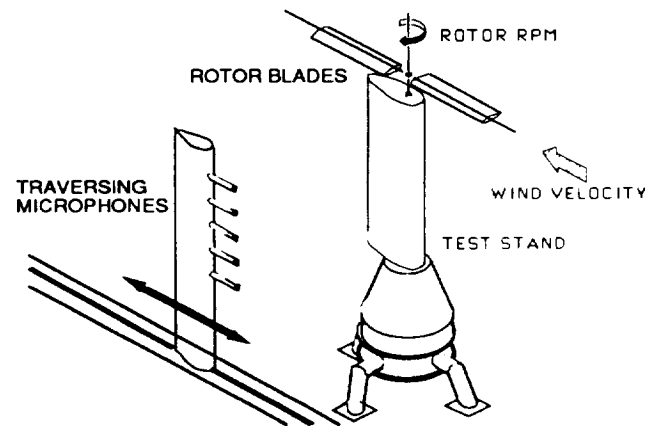


Figure 1. Typical Test Configuration

are multiples of the blade passage frequency. Also note that although the sound power decreases rapidly at higher frequencies, there is still substantial interest at these higher frequencies because of the sensitivity of the human ear as a detecting instrument.

Therefore, the characteristics of the acoustic signal from a rotor can be summarized as follows. It is highly RPM dependent and contains most of its energy at multiples of the blade passage frequency; it can have broadband content in addition to the harmonic content; it contains sufficient energy at higher frequencies to be detectable by the average human ear; its dynamic range (ratio of highest to lowest levels) can be as high as 80 dB and this dynamic range is detectable by the ear.

In the past, the output from the microphones were recorded on an FM tape recorder as the test proceeded. After completion of testing, the recorded

data were manually played back into a dual-channel spectrum analyzer and the result printed or saved to a file. Several attempts were made to computerize this procedure, but it remained basically a slow, manual process. It was impractical to process any significant amount of data as a test progressed to evaluate its quality or to note interesting trends that might necessitate changes to the test plan. As a result, test productivity suffered. Things that might have been noted and further investigated at minimal cost had to be left to later experiments.

To improve the process, it became obvious some years ago that a multi-channel digital data acquisition and processing system was needed. A survey of the market at the time revealed that existing systems would not meet our needs at reasonable cost; therefore, an effort was undertaken to implement a PC-based system.

Description of the Digital Data System

The characteristics of the desired digital data system can be summarized as follows:

- The system should at least double the number of channels that could be processed at one time. So, at least four channels, and preferably more were required.
- It should be low cost.
- It should provide quick, random access to the data (as opposed to the manual, sequential access of tapes).
- It should record data with an effective frequency span of up to 25-30 kHz, a dynamic range of 80 dB, and frequency resolution of 1 Hz.
- It should record sufficiently long data records to enable effective averaging of the data.
- It should be easy to program and maintain.

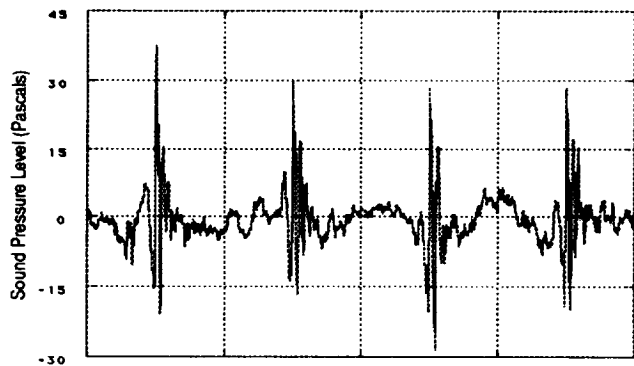


Figure 2a. Typical Time History

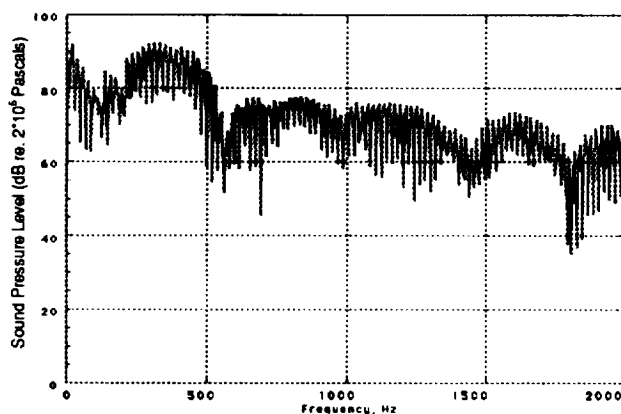


Figure 2b. Typical Frequency Power Spectrum

The availability of the Macintosh II computer, several types of digitizing boards, and, particularly, the availability of LabVIEW software from National Instruments led to the implementation of the present data acquisition and reduction system. Some compromises had to be made in the above outlined requirements in order to utilize off-the-shelf components and avoid costly customizing.

Hardware

The main components of the present data acquisition system (Fig. 3) consist of the microphones and power supplies, the amplifiers, the anti-aliasing filters, the computer, digitizing board, storage disks, and the RADVIEW computer program. A number of other monitoring instruments are also usually in use but are not part of the data acquisition system.

The present setup is a four-channel system; however, it could be expanded to eight channels with minimal effort. Through the use of electronic switching units we have been able to use this four-channel system to acquire data from eight microphones. However, simultaneity of measurements on all channels is lost with this technique.

Each microphone signal, after being amplified to the proper level, is low-pass filtered to eliminate aliasing effects. The filtered signal is then passed to one of the four digitizing channels of the board. The board also accepts an external trigger input. This is usually a 1/rev TTL pulse from an encoder on the rotor shaft that ensures record-to-record synchronization for proper time domain averaging. The filter cut-off frequency is matched to the sampling

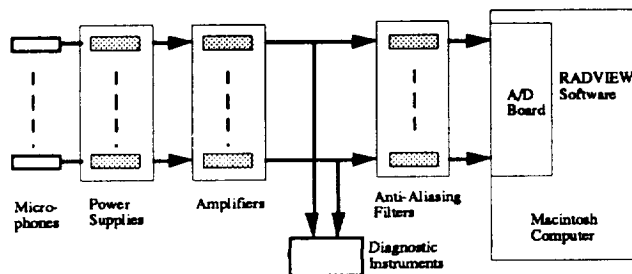


Figure 3. Typical Instrumentation Set-up

rate to ensure minimum distortion and contamination of the signal within the passband of interest. All four channels are sampled simultaneously with 16-bit resolution at rates up to 50,000 samples per second. The digitized data is stored on a large hard disk (or a removable cartridge) for later processing.

Operating Procedures

The data acquisition procedure consists of several steps. At least once per day (or more often at the discretion of the acoustics engineer) a calibration is performed on each microphone using a pistonphone that generates a known unsteady pressure. The calibration signal is recorded and used to convert the voltage output of each microphone to pascals. A data run begins by recording a “zero” data point with all systems powered up, but with the rotor and wind tunnel in a quiescent state. The rotor is then spun up to some predetermined operating RPM, following which the wind tunnel is started and brought up to speed. The first data point is a check point which is repeated during every run to check for consistency of data. A number of test points are then set during which various parameters of interest are systematically varied.

When the parameters for each test point are set, the microphone channels are monitored on an oscilloscope and/or an RMS-type voltmeter and the proper amplifier gains are set to ensure good signal level without overloading. When ready, a predetermined amount of data is digitized and stored. This may be repeated a number of times if more than four microphones are being used. If a microphone traverse is in use, the process is repeated at each of the traverse positions. The main operating parameters and amplifier gains are recorded in a log.

At the end of the run another “zero” point is recorded after the wind tunnel and rotor have come to a stop.

At opportune times during the run and after the completion of the run, some selected data can be reduced using the Quick Look function to evaluate

data quality and guide subsequent testing.

RADVIEW Program

RADVIEW is a computer program that implements the acoustic data acquisition and reduction system features described in the previous section. It is based on the LabVIEW programming language.

Description of LabVIEW

LabVIEW is a high-level programming environment developed by National Instruments Corporation (NI) of Austin, Texas. It utilizes the G language, also developed at NI (Ref. 1). It allows development of custom instrumentation software with graphical user interfaces on personal computers, and has extensive I/O capabilities when coupled with data acquisition hardware. LabVIEW has two distinct aspects; its front panel, which is presented to the user, and its block diagram, where the programmer develops the code that drives the program. The front panel is analogous to the input/output functions of a traditional programming language, and the block diagram is analogous to functions and subroutines.

The front panel consists of controls and indicators that can be configured to resemble an actual instrument, with meters, waveform displays, knobs, and numerics. These are adjusted using the mouse and keyboard. Typically, before a program is run, the user enters information on the panel and sets various controls to a desired state. This is analogous to setting or changing values in input statements or files in a traditional language. As the program runs, and after it has finished, results are displayed on the same panel. This is analogous to output tables or graphs in a traditional language.

The block diagram is similar to a flowchart. Specific operations are represented as blocks (icons), and the flow of information from block to block is accomplished with wires. The programmer can see where the process will start, which steps it will

encounter as it runs, and where it will finish.

While many development environments allow the construction of a custom user interface (front panel), what sets LabVIEW apart from all other systems is its block diagram. More than a flowchart, the block diagram is the executable code, not merely a graphic representation of it. Wires on the screen connect icons representing functions and actions. Variables are sent along the wires from one block to another.

LabVIEW includes many powerful programming concepts analogous to the traditional branching, case, and do loop structures. One of its primary features is transparent handling of array operations. Wiring an array to the LabVIEW equivalent of a "Do Loop" sets the number of times that loop will execute. There is no need for a separate control statement. Similarly, many array operations are executed without structures of any kind. Wiring an array of any dimension and a scalar to a multiply icon will multiply every element in the array by the scalar. The multiplied array is available at the output terminal of the multiply icon. These features were useful in development of RADVIEW as all analysis and manipulation of data are done on arrays of data. In general, any program written in a text- or line-based program such as FORTRAN can be written with LabVIEW, often with greater efficiency.

A LabVIEW program is driven by dataflow, which means that all parts of the code attempt to run at the same time, but execute individually only when all input data are present at an icon. Four parallel operations will execute simultaneously, while four sequential steps will execute in strict order, each sending a result to the next. The manner in which the icons are wired establishes sequential dependencies; in essence dataflow can be used to control the sequence of operation. The sample block diagram in Figure 4 demonstrates these ideas.

Since LabVIEW runs on a single processor, operations are not strictly parallel, but rather each action is allotted a small amount of CPU time in

rotation until all actions are complete. A future development we would highly recommend to NI is an extension of the language to multi-processor environments. The existing language lends itself naturally to such extension with little re-education required of the programmer.

The combination of a front panel and block diagram is called a Virtual Instrument, or VI. VIs can be run alone, or in combination with other VIs. They can be placed as elements on a block diagram, which may then be used as subroutines (subVIs) to perform some task which is part of the overall problem to be solved.

RADVIEW Interface

Users of RADVIEW are initially presented with a single VI. This VI's panel has text fields for entering test name and local hard drive (for data storage location), and buttons which activate different parts of the program (Fig. 5). The block diagram of this highest-level VI consists of a loop that waits for a button to be pressed, then runs the subVI corresponding to that button. When the subVI finishes, the loop resumes checking for pressed buttons, until

a "Quit" button is pressed. This structure is analogous to having menus at the top of a screen and allowing the user to select an item. Here, all options are laid out and visible at once. The user can select program functions in any order, though they are designed to be used in a sequential manner when acquiring data and preparing it for analysis.

Figures 6 and 7 illustrate the input/output interface ("Front Panel") and program code ("Block Diagram") for RADVIEW's QuickLook subroutine ("SubVI"). Each front panel control (input) and indicator (output) has a corresponding terminal on the diagram. The diagram in Figure 7 gets a binary file, which is an array containing all the sampled cycles, and tests each cycle using LabVIEW's MAX & MIN function. The appropriate cycles are sent to the front panel for display.

Signal conditioning and amplification are carried out through external hardware. Since there can be more than four microphones, a switching unit may be used to pass all microphone signals to the four-channel A/D board. The user can specify the microphone configurations before acquiring data, including any combination of microphones in a set.

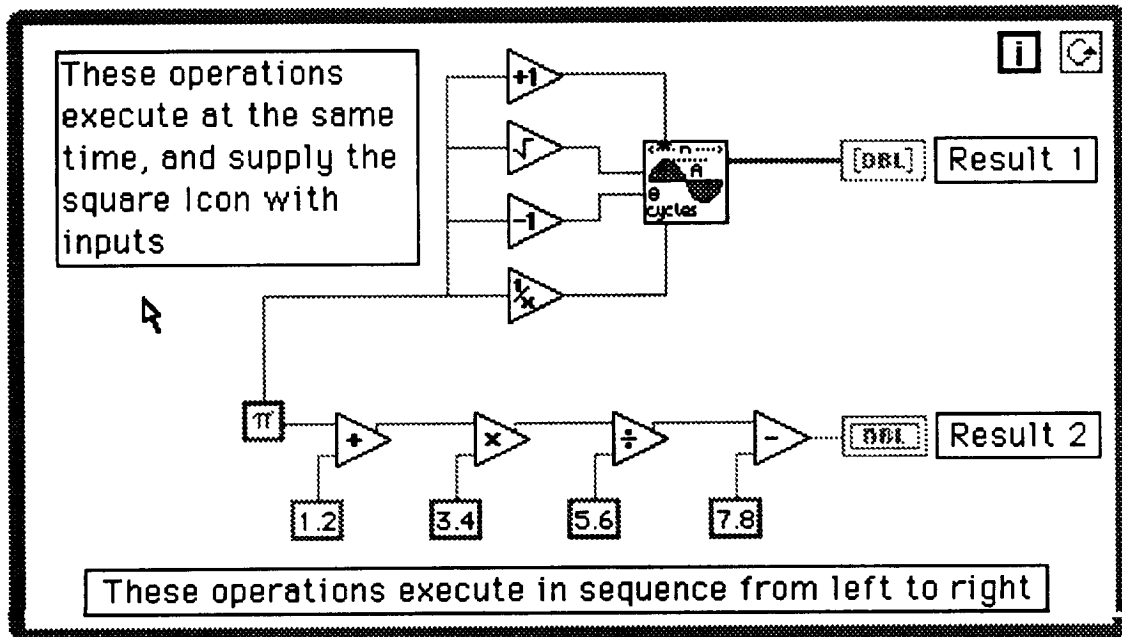


Figure 4. Block Diagram Example

Several passes over the same test point might be required to acquire data from all microphones. Although multiplexing the different microphones into RADVIEW is currently a manual process, this will be automated in future tests with a switching unit that RADVIEW will control.

RADVIEW creates data files as it acquires data, and automatically places the data in properly named folders. File creation follows a simple rule, where a typical file might be called R48 P6 M2 V1. This corresponds to Run 48, Point 6, Microphone 2, and Version 1. Run and Point refer to a specific test configuration and Version can refer to, for example, a different switch set or a different microphone traverse location. The program stores calibration files separately from data files, but with a similar naming convention. In the example shown in Figure 6 the folder "Cal 48" holds calibration signals acquired for use with data acquired during Run 48.

The program writes all data in binary format, but appends an ASCII header to each file. This allows most text processors to read the header information. An asterisk separates the header from the data that follows. Data are written in binary to reduce storage space and to greatly speed their placement and retrieval.

The user can add comments to the data header, to record any unusual or special conditions. Files can be converted entirely to ASCII for analysis by other software. The program can also import data from selected acquisition programs for analysis with RADVIEW.

A direct result of allowing RADVIEW to name files and folders as it acquires data is that the program can perform data operations on entire sets of data, as opposed to individual data files. For instance all data acquired during a test run can be averaged or calibrated in a single operation. There is no need to apply calibration constants to each file individually.

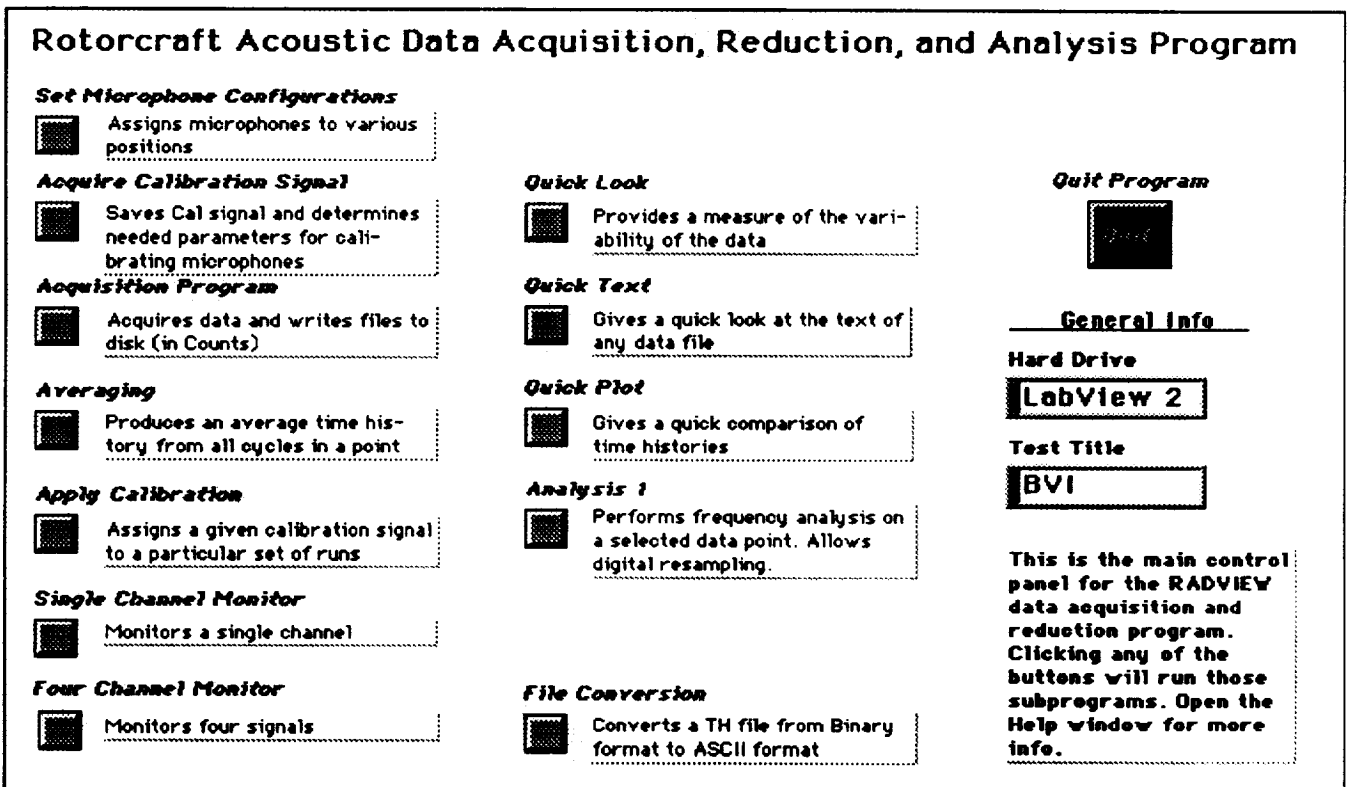


Figure 5. Front Panel of RADVIEW Program

This program looks at the variability of data within each cycle. The arrays with the greatest and least deviations are plotted with the overall average to determine (roughly) the consistency of a signal within a test point.

<input type="checkbox"/> Return to Main	<input type="checkbox"/> Run Point	<input type="checkbox"/> Mic 3	<input type="checkbox"/> Hard Drive	<input type="checkbox"/> Weaver	Min Diff: 0.83
<input type="checkbox"/> Go	<input type="checkbox"/> Version 1	Status: Ready to Start	<input type="checkbox"/> Test Title	<input type="checkbox"/> BVI	Max Diff: 1.14
			<input type="checkbox"/> Max	<input type="checkbox"/> 2	Deviation: 0.32
			<input type="checkbox"/> Min	<input type="checkbox"/> 7	
			<input type="checkbox"/> Avg	<input type="checkbox"/> —	

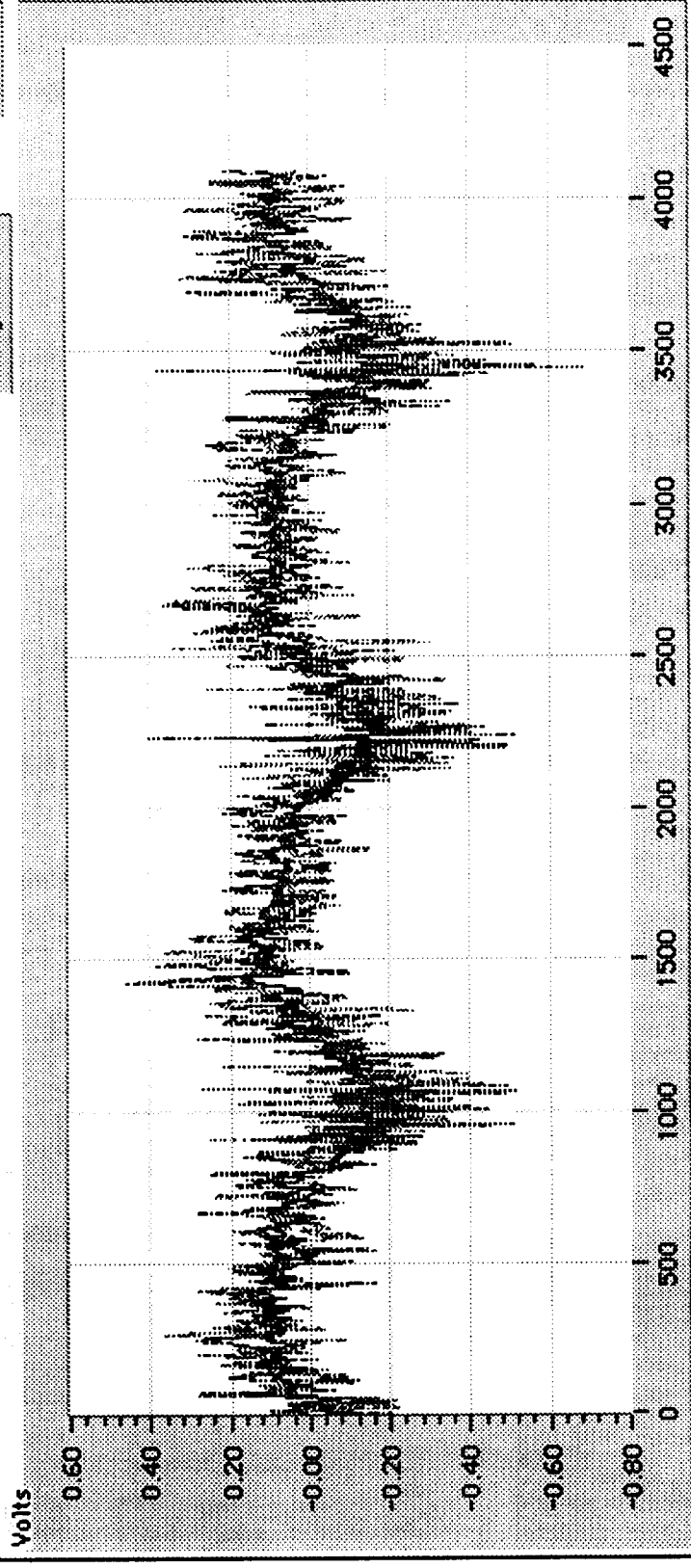


Figure 6. Front Panel of Quick Look SubVI

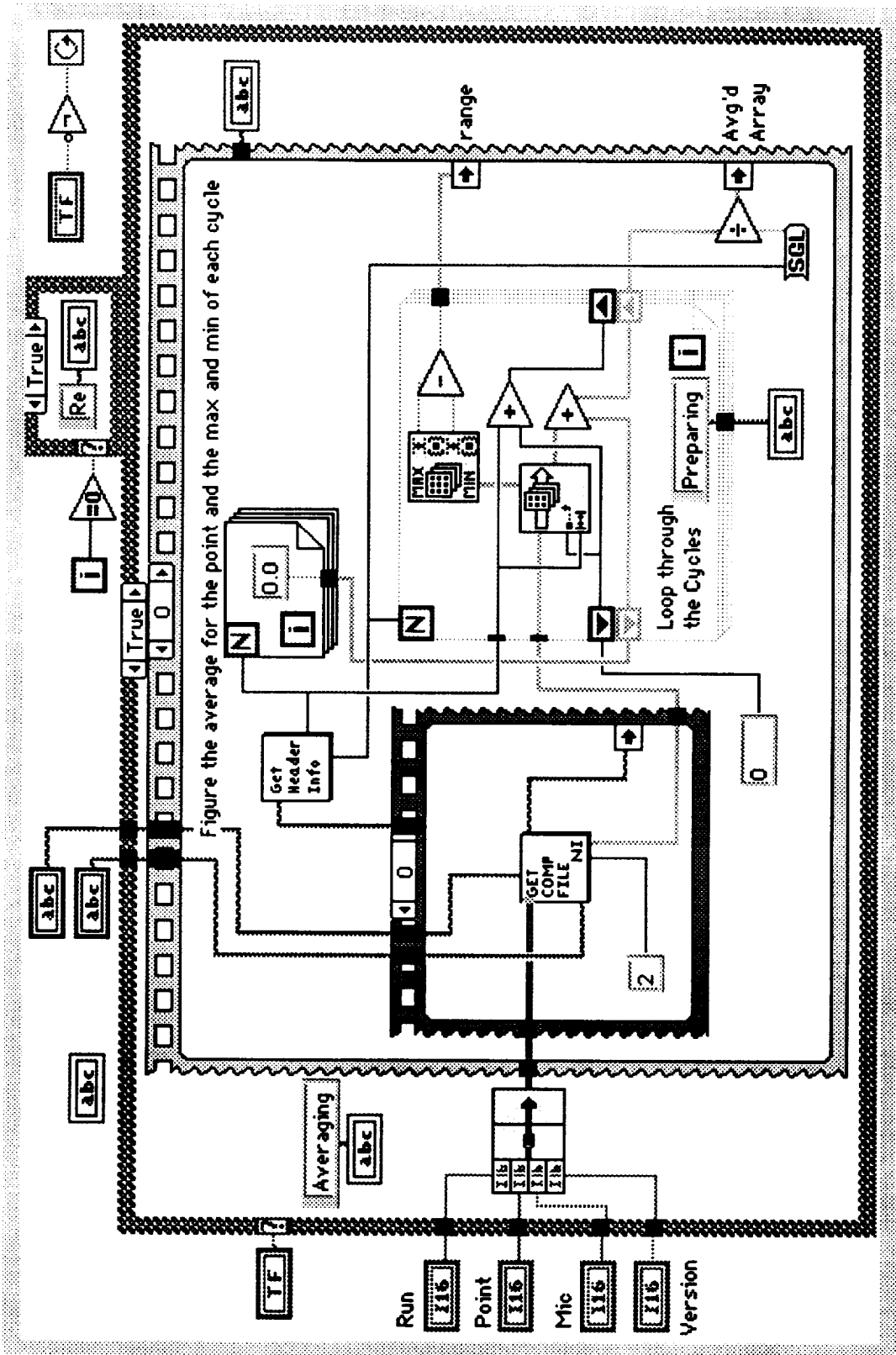


Figure 7. Block Diagram of Quick Look SubVI

This has been found to save considerable time after data have been acquired.

Finally, experience in development and use of the program has shown that correcting errors and making modifications is extremely fast and straightforward. The programming language is always at hand, as the block diagram exists just behind the front panel of any VI. Modifications to code can be made on the spot during a test and immediately put to use.

Operation Sequence

The following steps describe the general procedure for acquiring data with RADVIEW. The user specifies the name of the hard drive where data will be stored, and the name of the test (usually a short acronym). These names can be set to defaults for repeated use. Although data can be archived directly to media other than an internal hard drive, this is usually slower. Files can be moved to other media at the end of a test day.

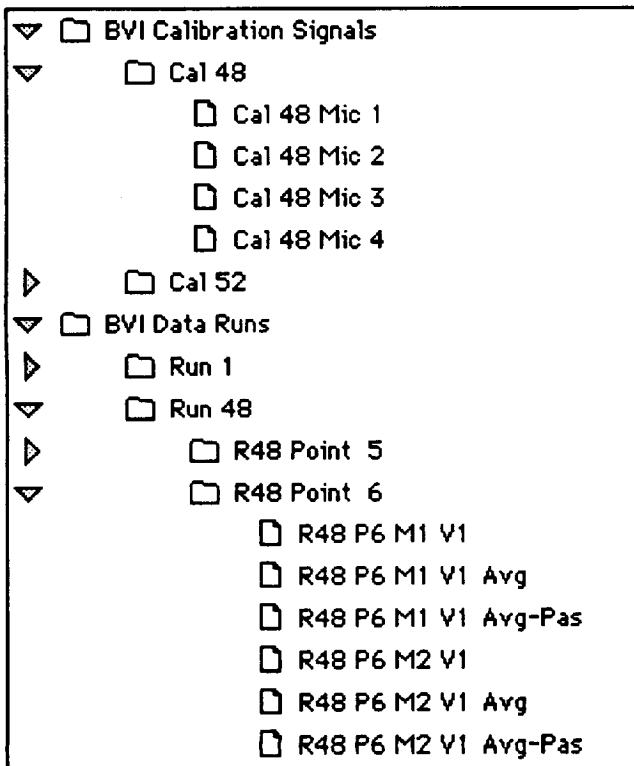


Figure 8. Data Storage Format on Hard Disk

Before acquiring any data, the user specifies one or several sets of microphone configurations. These records are stored on disk for future reference. Later, when acquiring data, the user will click on a button to select a microphone configuration.

Next, the user acquires calibration signals for each microphone. The user specifies the Calibration number (usually the same as the Run number), the Microphone number, the known sound level (usually 124 dB), and the number of samples to acquire. The user also selects a sampling rate from a choice of several dozen consistent with the acquisition hardware. When the user is ready, the program acquires the waveform, calculates a slope and intercept, and writes the calibration information to disk, along with the digitized calibration signal, for later use with raw data. The slope and intercept will be used to convert data from volts to pascals. The acquired signal and calculated values are presented on screen for verification.

With microphone configurations specified and a set of calibration signals stored on disk, the user is ready to acquire test data. Before acquisition the user specifies run, point, and the microphone set (version number) for this data set. The user also selects a sample size and sample rate from several available. Comments can be added to the data header. On command from the user the program acquires the data and writes them to disk.

Several rotor cycles (most useful is a power of 2) of data are acquired and stored in a single file at each test condition. Since the acquisition settings determine how much data will be written to disk, the user is shown an estimate of the hard disk space that will remain after the data are acquired. Typically the user will remain at this point in the program for a lengthy period, acquiring many data sets corresponding to different test conditions.

Data Processing

To properly analyze the data, a number of data processing techniques are necessary. The data must be calibrated (converted from raw counts to engineering units) and adjusted for any gain or attenuation introduced by the amplifiers. The data must be averaged to remove random, non-repeating features. Usually, the power spectrum of the data is of interest for acoustic research.

When all data have been acquired, or preliminary analysis is desired, the user selects the subVI that averages data. This VI allows the user to average data across several points, versions, and microphones in a single operation, though they are restricted to data from a single run at a time. The starting and ending data point are entered, as are the starting and ending microphones and versions. Averaged data are stored in the same folder as the raw data, with “-avg” appended to the file name. Note that raw data are written to disk as counts (-32,768 to 32,767 for 16 Bit ADC) but are scaled to volts when averaged. The additional “-avg” file does not require significantly more storage space because it contains only one cycle of data.

Averaged data can be calibrated, scaling their units from volts to pascals. This procedure is identical in form to the averaging routine described above. The calibration procedure can be applied to a single data set, or multiple data sets in a single step. Data in pascals are written to disk in floating point format with “-pas” appended to the file name. All of these files are in binary form.

Calibrated data can be analyzed for frequency content and spectral amplitudes. The user sets the run, point, microphone and version at the front panel, and can quickly step through any of these parameters. Data can be passed through several windowing functions prior to being transformed to the frequency domain. The program includes options for no window, Hanning, Hamming, Triangle, and Squaretop windows. Since windowing data dissi-

pates spectral energy, analytical correction factors are used to scale data after windowing. These factors are listed below in the Appendix. A single processed file is viewed at a time.

Several other functions can be performed from the main panel. Up to four data channels can be monitored in real-time. The user can plot raw, averaged, or calibrated data and compare several waveforms. The text content of files can be compared side by side. A “Quick Look” subVI shows three plots taken from a raw data file: the average of all cycles, the cycle with the least amplitude variation, and the cycle with the greatest variation. This gives an estimate of data repeatability. Finally, the user can convert RADVIEW files to ASCII for use by other programs.

Future Improvements

There are a number of improvements that can be made to the hardware and software. The present acquisition card has high resolution (16 bit), but does not allow external clocking and has a relatively low sampling rate (48 kHz maximum). 12 bit cards are available that sample four channels, each at 250 kHz, and allow external clocking. Using two of these cards in tandem would allow more microphones to be sampled simultaneously. Use of external clocking would let engineers match sampling rates to blade rotation.

More use can be made of GPIB control. Several operations that are currently performed by hand, such as configuring the switching unit, and setting amplifiers and filters, can be controlled by the program, thus leaving more time for analysis during testing.

Other potential improvements include the ability to produce presentation quality graphs directly from LabVIEW. Fuller use of the built-in functions that LabVIEW provides will increase analysis capabilities beyond the basic functions presented here. The development of the program is continuing and

many of these improvements will be implemented in the near future.

Conclusions

We have described an acoustic data acquisition system that utilizes a unique programming environment. The system allows engineers to acquire and analyze acoustic data using a Macintosh PC while using a single program. The program's analysis capabilities have allowed it to replace more expensive FFT analysis hardware.

References

1. Kodosky, J., MacCrisken, J., Rymar, G., "Visual Programming Using Structured Data Flow", Proceedings of the 1991 IEEE Workshop on Visual Languages, Kobe, Japan, October 8, 1991.
2. Harris, F. J., "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform", Proceedings of the IEEE, Vol. 66, No. 1, 1978.
3. Press, W. H., Flannery, B. P., Teukolsky, S. A., Vetterling, W. T., Numerical Recipes. The Art of Scientific Computing, Cambridge University Press, 1986.

Appendix

Description of Algorithms

Calibration Routine

When the program acquires a calibration signal, several steps are followed before data are written to disk. First the data are scaled from counts to volts by the simple conversion

$$\text{volts} = \frac{\text{Counts} * 10}{65536} \quad (1)$$

The user supplies the gain employed on the microphone being calibrated. The slope is calculated as

$$\text{Slope} = \frac{10^{\left(\frac{124 + \text{Gain}}{20}\right)} 2.0e-5}{\text{RMS}(x)} \quad (2)$$

where x is the acquired signal in volts and Gain is specified in dB. 2.0e-5 pascals is the reference value for obtaining acoustic power in dB.

The intercept is calculated as

$$\text{Intercept} = \text{Slope} * \text{Mean}(x) \quad (3)$$

The slope and intercept are stored in the data header of the Calibration file on disk, and are referenced when the program converts averaged data in volts to pascals. The raw calibration data is also stored for future reference.

Windowing Routines

The following scaling factors are used with each windowing function:

No window: 1.0
Hanning: 1.42
Hamming: 1.85
Triangle: 2.00
Flat Top: 3.92

These values are taken from Reference 2 and are applied to the results of the windowed FFT. They correct for the inherent dissipation of energy when signals are passed through a smoothing window.

Frequency domain transformation

Data are transformed to a frequency spectrum in the following manner. The array containing the calibrated and windowed time history is transformed with a Fast Fourier Transform. Data are typically acquired in sizes that are integer powers of two, so there is neither any truncation nor any zero padding done by the FFT. The program takes the square root of the sum of the squares of the real and imaginary parts of the FFT and divides the resulting array by the number of elements in the original time history. The spectral levels are multiplied by $\sqrt{2}$ to conform to the definition of the Fourier Transform (Ref. 3). They are then multiplied by the appropriate windowing factor, as described above.

The program retains the lower 78 percent of the frequency span because of known inaccuracies in the upper part of the span arising from anti-aliasing filter characteristics near the cut-off frequency. The results are then converted from pascals to decibels:

$$\text{dB} = 20 \log\left(\frac{\text{pascals}}{2.0e-5}\right) \quad (4)$$

Finally, the program displays the results on a dB scale, with the spectral level at each frequency given by

$$\text{Spectrum}(x) = 20 \log\left[\frac{\text{W.F.} \cdot \sqrt{2}}{N} \left[\sqrt{(\text{Re}(x))^2 + \text{Im}(x)^2} \right] \right] \quad (5)$$

where W.F. is the windowing function correction factor, x is the calibrated data, and N is the sample size.

