A MATLAB Tool for Experimental and Analytical Shock and Vibration Data

Morris Berman, U.S. Army Research Laboratory, Adelphi, Maryland

A new MATLAB® tool provides the shock and vibration community with the ability to display and analyze data while minimizing the probability of bookkeeping errors.

Test and analysis operations often result in the generation of large quantities of experimental and analytical data. Often this data must be processed in a variety of ways and then plotted and compared with data from the same test series as well as historical data. Many times, there is a significant amount of metadata that accompanies the data of interest that is critical to the understanding of the analysis and comparisons. A MATLAB tool was developed to ease these processes and minimize the probability of analysis and comparison errors.

One of the key features of this tool is that it couples all of the available metadata with the measured/computed data set and provides easy access to that metadata. This tool supports a variety of specific data analysis methods as well as many standard data analysis methods. Its prime focus has been on enabling the user to easily display both the data of interest as well as the descriptive attributes that give the data meaning.

This tool takes advantage of MATLAB's object oriented constructs and thus is very simple to use for an individual who is already familiar with it. Most of the relevant operators have been defined to operate on these datasets in a meaningful fashion. In addition, MATLAB's plot routine has also been extended to work directly with these data objects.

Often times the datasets utilize extremely long record lengths. This fact often complicates analysis simply due to memory and speed constraints. The tool was implemented in an optimized manner in recognition of many of the complications that arise from this fact.

Introduction

Many years of testing and simulation have resulted in the use of a wide variety of data collection and storage systems. Each of these systems is well suited to using the data that it generates. However, the data storage formats used are generally proprietary and not well suited to interoperability. Due to the rapidly changing nature of current data acquisition hardware and software, older systems rapidly become obsolete and often render data collected by them difficult or impossible to access. Although many of these systems are capable of writing their data in commonly used formats, often times some of the metadata as well as the precision associated with the original data are lost during the conversion process.

These factors have lead to the development of a toolbox for addressing experimental and analytical datasets, with time or frequency as the independent variable. XYData is a toolbox that was developed in MATLAB to address these needs. This new toolbox was motivated by the need to bring collected data from disparate data acquisition systems and analyses together into a common framework. It is important to be able to analyze data from legacy, current and new acquisition systems together as well as including data generated by finite-element (FE) or other simulations. It is also often necessary to include data generated by external organizations.

Another significant motivating factor in the development of XYData is the management of the metadata associated with a measurement set. It is trivial to store numeric data in a common format that is easily accessible. However, data without context is often meaningless. The inclusion of complete metadata becomes critical when large-channel-count tests and large numbers of datasets with varying conditions are considered. XYData was designed from its roots to retain all of the metadata available for any dataset as well as tightly coupling the metadata to the numerical data. Modern data acquisition systems are capable of storing a large quantity of data, both in terms of number of channels as well as sample size. XYData was designed with this in mind, easily allowing the analyst to consider partial datasets without the need to waste time loading irrelevant data. XYData is ideally suited to batch processing large quantities of data through a common set of analysis routines.

A number of considerations lead to MATLAB[1] as the choice for a development platform. MATLAB is a very common platform in the academic environment; almost all graduating engineers have some minimal familiarity with it. MATLAB includes object-oriented programming constructs and is designed to be very extensible, including the ability to develop graphical-user-interface (GUI) elements. MATLAB's implementation of these features is platform independent so that the same routines can run on the simplest laptops as well as the highest performing supercomputers without modification. One significant drawback posed by MATLAB is its significant upfront and maintenance costs. There is an open-source package called Octave that implements many of the same programmatic elements as MATLAB. It should be a trivial matter to port XYData to Octave, eliminating the costs associated with MATLAB with, perhaps, a performance penalty.

Design Philosophy

The design goals of XYData were developed to ease and encourage its use as a widespread engineering tool for the analysis of a wide variety of experimental and analytical data. They can be summarized by extensibility, ease of use and management of large datasets.

XYData was specifically developed using object-oriented programming (OOP) techniques. The use of OOP allows XYData to realize an extremely tight coupling of data and metadata. This coupling is critical to minimizing the probability of data descriptor errors. Appropriate use of OOP principles dictates that access to a record's data occurs only through specific interface methods designed for that purpose. All data is accessed through *get* and *set* functions that are aware of the specific data storage algorithm. As a result, the underlying structure of the data can be changed and optimized as the need arises without any detrimental effect on currently existing algorithms.

Using OOP allows XYData-type variables to be used as any other MATLAB datatype. All of the common arithmetic and array reference operations are implemented with XYData. As a result, two XYData objects can be added together as easily as two scalars using the same operator and syntax. Many of the common functions have also been implemented in XYData as well, including *min, max* and *sqrt.* Most importantly though, the *plot* function has been implemented in XYData in a manner that eliminates the need for the user to track data and its descriptors separately.

The ability to handle large datasets implies both an efficiency of storage and I/O operations as well as the ability to select and manage large numbers of records and their descriptive attributes. XYData permits loading of partial datasets either through the loading of a subset of records from the complete dataset or loading partial records. To this end, the loading of only the metadata is supported so that the choice of which records to load can be made on the basis of the metadata. Most of the implemented data analysis methods use highly efficient memory management and vectorized algorithms, improving analysis speed for large datasets.

A large portion of managing large datasets is ensuring that the data are properly documented. XYData permits inclusion of an

Based on a paper presented at the 82nd Shock and Vibration Symposium, October 30-November 4, Baltimore, MD, 2011 arbitrary number of textual descriptive lines as well as the inclusion of arbitrarily named attributes that can contain scalars, arrays, matrices and cell arrays. This flexibility ensures that the data can be fully documented and that documentation remains attached to the relevant record. In addition, each analysis routine adds a tag to the resulting records that describes the type of analysis and relevant parameters. Every process acting on a data record leaves a trail in the resulting record. This way, the user can track how a particular data record has been modified from the original measured data. In large experimental efforts, mistakes inevitably occur in data storage and labeling. XYData implements a means to fix and document these errors automatically at the time data are loaded.

XYData was designed to leverage a user's existing knowledge of MATLAB to the fullest extent possible. As previously noted, OOP has enabled implementation of most MATLAB operators and a large number of common functions. In addition, significant efforts have been made to automatically account for data inconsistency resulting from disparate or unevenly spaced time vectors. For instance, when adding two records, the abscissas are automatically interpolated to common values before the addition operation occurs. XYData greatly simplifies plotting data and its descriptors. The user need only call out the records to be plotted and the associated metadata fields.

Basic Usage

An XYData object is created through the use of the *xydata* command. This command can take five different forms that create an empty record, load existing data directly, create a record given X and Y vectors, create a sine wave record or launch a GUI for viewing and operating on data interactively. The detailed arguments for each function can be found in the embedded XYData help. A novel mechanism is used for calling the appropriate XYData loader routine. All of the XYData functions for loading data begin with the word "load" followed by the type of data to be loaded. For instance, the loader for reading SD data is named *loadsd*. When *xydata* is called with a string as its first argument, the path is searched for a function with the name of that string preceded by the word load. So, to load SD data for event 1, channel 2, the user executes *xydata*('sd',1,2). This implementation permits trivial implementations of additional data loaders without modifying the *xydata* function.

A user simply needs to create a new data loader and ensure that the name starts with "load." XYData will automatically recognize the new loader as long as it is in the path. Each XYData record has a type associated with it. The types currently available are time, spectral, waterfall or srs. These types are used to ensure that certain processes only occur on certain types of data as well as setting various defaults for plotting.

Once data are loaded, manipulation with the basic arithmetic operators is fully supported. Addition and subtraction of a scalar and a XYData object adds or subtracts that scalar to the ordinate of the XYData. If both arguments are XYData objects, then the ordinates are added together. If the abscissas differ, a common abscissa is computed and the ordinates are interpolated on the new abscissa prior to the addition. Array multiplication (.*) uses the same logic as addition, and matrix multiplication (*) is only defined between XYData and a scalar or vector. The exponential operator is also defined for XYData.

A large number of basic mathematical functions have been implemented for XYData. MATLAB's *abs, angle, log, log10*, *sqrt, cumtrapz,* and *detrend* functions are implemented to act on XYData objects just as they would act on an array of real values. Although not a native MATLAB function, *centraldiff* provides a differentiation operator. The results are returned with the same abscissa as the input, with the ordinate being the result of the function. Other functions such as *min* and *max* return a scalar for each XYData record input. The *mean* function returns either the scalar mean of each XYData record or the mean of all input XYData records at each abscissa point. XYData implements several MATLAB functions for sampled data as well including *decimate, resample* and *downsample.*

Another class of functions implemented for XYData enables simple manipulations of XYData objects. One group of functions act on the ordinate, and these include *clipy, scaley, truncy* and *yshift*. A complementary group of functions act on the abscissa: *scalex, truncx*, *truncpt* and *tshift*. The *clipy* function limits the maximum ordinate value of its argument. The scale functions multiply the ordinate or abscissa by a scalar. The shift functions add a scalar to all values of the ordinate or abscissa, and the trunc functions limit the maximum and minimum values of the abscissa or ordinate. *Truncpt* selects a segment of the abscissa based on the number of points, not the values of those points. The *sum* function adds the ordinates of two or more XYData objects ensuring a common abscissa. The *concat* function combines two or more XYData object sequentially. The abscissa is not checked for consistency.

Several signal processing functions are provided. *Movavg* and *removedc* smooth data and provide a simple method for removing the initial DC offset often encountered in Wheatstone bridge devices. The *sdof* function provides the response of a single-degree-of-freedom system to the provided input. The *envelope* function outputs the peak ordinate at each abscissa point among the input XYData records, resulting in a "highest" value at each abscissa among the input records. Two simplified filtering algorithms are currently implemented. *Buttfilt* implements MATLAB's built-in Butterworth filter given cut-off frequencies and filter order. *Cfcfilt* is used in the same way as *buttfilt*; however, its arguments are consistent with the SAE J211 channel frequency class filters.

Unique Functionality

The utility of XYData lies in its unique functionality that closely couples data and metadata. The user never directly accesses the underlying data structures in a record; *get* and *set* are used to access all of the data and descriptor elements of a XYData record. This permits the underlying structure to change over time, resulting in improved efficiency and speed while still maintaining compatibility with all the code that has already been written. *Get* retrieves a field from one or multiple XYData records and *set* places a value into a field of XYData records.

There are certain fields that are available in all XYData records. Some of these fields are X, Y, desc, event, channel and id. The user also has the option of creating fields with arbitrary names. A core field used in XYData is the "descriptive lines." These are an arbitrary number of lines of text of arbitrary length. They are generally used to provide information specific to a channel or test condition in each record. It is helpful if a given line in a test series contains the same type of information. For instance descriptive line 1 might always contain the forcing frequency and descriptive line 2 might always contain the amplitude for a particular series of sine wave tests.

Since arbitrary field names can be created, it is important to be able to determine what fields exist in any given record as well as being able to list the contents of those fields. There are two methods to determine the fieldnames. The first method is to *get* the field named "fieldlist." This returns a cell array of all of the field names available for that record. The second method, the *showfields* function, returns a listing of all fields in the record that are non-blank.

Another important function for managing datasets is the *listinfo* function. This function provides both a cell array and a tabular listing of the requested fields for all referenced records. For example, an array of XYData records may exist from a particular test series; however, the user does not know which record corresponds to which tested condition. The *listinfo* function is designed to easily provide that information.

XYData provides facilities for extracting records based on a complex combination of conditions. It is important to be able to select data based on conditions and not just on a specific index value into an array. XYData provides two functions, *findrec* and *findrecnum,* for that purpose. One returns the actual XYData records, and the other returns the indices of the selected records.

These functions take a series of field-value pairs as their arguments. The "field" argument identifies which field the condition should be tested against. The "criteria" argument is the test condition. If multiple field-value pairs are specified, all of them must be satisfied for a record to be selected. In its simplest form, the "criteria" argument is just a text string that must match the

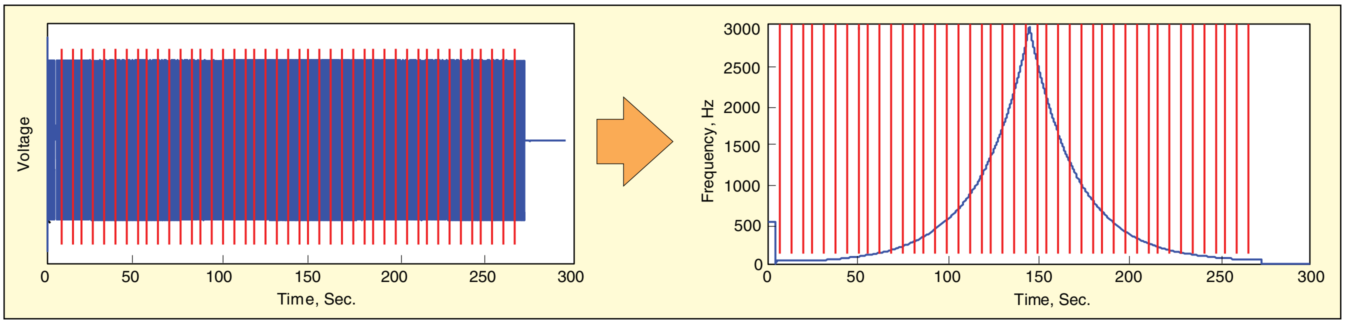

Figure 1. Swept sine divided into time slices.

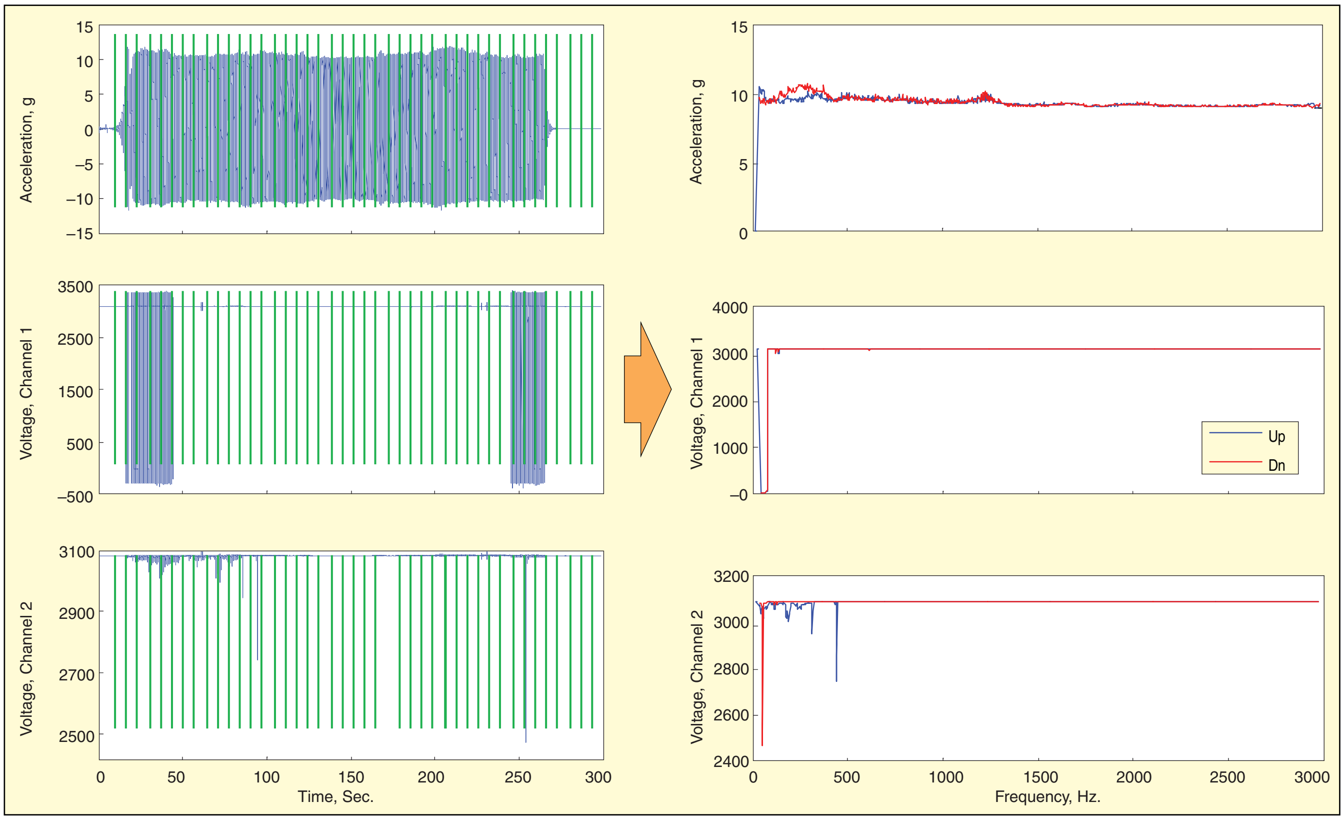

Figure 2. Swept sine analysis results.

selected record. A cell array of values can be provided for a given field. In that case, the series of values are connected by an "or" operator. For more complex search conditions, the full range of MATLAB's regular expressions are supported in the contents of the "criteria" argument.

The *showprocess* and *addprocess* functions are provided to enable documentation of the operations performed on any particular record. Each time a record is operated on, the resulting record contains information about the operation performed. This documentation is added to the record with the *addprocess* function. If a user wishes to understand what processes have been applied to a particular record, the *showprocess* function details the chain of processes for that record.

Plotting

A key function of the XYData toolbox is to facilitate plotting of experimental and simulation data while minimizing the probability of labeling errors. The *plot* command provides this core functionality. In its minimal form, it takes a single argument, which is an array of XYData records. However, its real utility is derived from its second argument. This argument is a scalar or cell array that identifies which descriptors should be included in the legend for each curve. The cell array can contain a scalar number to identify a particular descriptive line or any field name. If an XYData set contains three records [XYD1 XYD2 XYD3] and the second descriptive line of each record contains the projectile's weight, the command *plot ([XYD1 XYD2 XYD3], 2)* would plot all three records on a single axis, and the legend would contain each projectile's weight. The user does not need to track the data separate from its descriptors nor issue a separate MATLAB *legend* command as would normally be the case. As long as the descriptors are correct when the data is read into the XYData records (or are corrected by the included XYData mechanism), there is little possibility of misapplying legend labels, resulting in more certainty if the data is inconsistent with expectations.

For comparing large numbers of XYData records, the *splot* command can be used to plot multiple axes on a single page, with each axis containing a single XYData record. As the user wishes to increase the complexity of the plots, the third argument can contain the full range of property-value pairs acceptable by MATLAB's native plot command.

In a similar fashion, XYData also implements the *plotyy* and *loglog* MATLAB plotting commands. The *plotyy* command simplifies plotting two different XYData records on the same axis with disparate ordinate values. For instance, where the ordinate of one record is on the order of 0.001 and the ordinate of the second record is on the order of 10,000. The *loglog* function simply plots the records on logarithmic X and Y axes. XYData also implements *bode* to create a bode plot of complex valued XYData records.

Injury Analysis Methods

XYData has been used extensively in the assessment of human

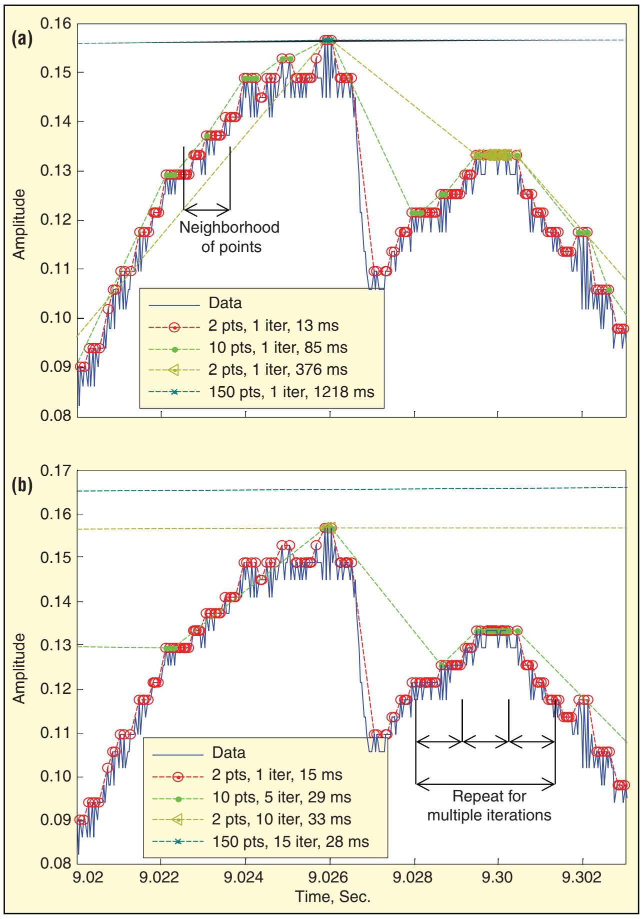

Figure 3. Peak finding; (a) varying neighborhood and (b) varying iterations.

injury probability from anthropomorphic test device (ATD) data. As a result several functions are included within XYData that address injury criteria. The *dri* function implements the dynamic response index injury criteria. The DRI represents the compression of the human body's core, given a vertical input at the pelvis.[2] The response is modeled as a single-degree-of-freedom system and the result used in computing the probability of injury.

The *iband* function provides a measure of the load's amplitude versus its duration at that level. This measurement is used in the computation of other injury criteria. The output of the *iband* function is maximum duration or cumulative duration that the XYData record remains at or above a given amplitude.

Swept-Sine Analysis Methods

The XYData toolbox incorporates significant functionality for analyzing swept-sine data. Traditionally swept sine tests provide the response filtered at the excitation frequency. In most cases, this is sufficient. However, situations arise where the response is a DC or other time-varying signal, and there is a need to characterize how that response varies depending on the input frequency. In this situation, the traditional analysis methods fail.

XYData's *sinswp* function is able to provide an analysis of a signal of interest where the response does not occur at the excitation frequency. The algorithm utilizes a record that comprises a clean measurement of the input sine wave. This could be the signal input to the shaker or an unattenuated drive signal provided by the swept-sine controller. This signal is divided into a series of time slices determined by the user. Each time slice (red lines on the left of Figure 1) is then analyzed to determine its average frequency. This yields a XYData record of time vs. excitation frequency (right side of Figure 1).

The resulting XYData record of time vs. frequency is then used to divide the record of interest into the same time slices (green lines on the left of Figure 2). Each of those time slices is then analyzed, and a resulting scalar is determined. Generally the analysis comprises the maximum or minimum value in the time slice. The resulting scalars are then inserted into another XYData record with the frequency as the abscissa, resulting in a frequency vs. value of interest plot (plots on the right of Figure 2).

Frequency Analysis Methods

A variety of frequency analysis techniques are implemented in the XYData toolbox. The transfer function estimate as well as cross and power spectral density functions are implemented using MATLAB's underlying signal analysis routines. However, the *cpsd*, *psd*, *tfestimate* functions of XYData use the information available in the abscissa to generate the correct frequency axis associated with those functions. These functions provide arguments to divide the XYData record into frames of a fixed number of points or a fixed duration for an ensemble analysis. A shock response spectrum as well as a pseudo-velocity shock response spectrum is implemented in XYData. Both the *srs* and *psrs* use Smallwood's algorithms.[3]

XYData also includes both plotting and analysis routines to simplify generating waterfall plots. The *psdwfall* function generates a series of PSDs using the *psd* command. XYData data's *plot* command is aware of these PSDs and will generate an appropriate waterfall plot when they are provided as an argument to *plot*.

Peak Analysis Methods

The need often arises to identify peaks within a set of data. When the data are noisy and contain a variety of local and global peaks, this becomes a complicated problem. XYData implements *getpeaks* to address this type of analysis. The essence of the algorithm is to compare each point to its neighbors and determine if its value is higher than those neighbors. This algorithm can be executed with varying the number of points in the neighborhood as well as executing multiple iterations of the algorithm. The simplest means to implement this algorithm is through a loop over all the points in the data record. Unfortunately, for large records, this method is extremely slow. XYData implements a very efficient vectorized method that is able to handle very large data records with high speed.

Figure 3 illustrates the peak-finding algorithm implemented in XYData with various parameters. A sample set of data is shown with the solid dark blue curve. Although only 12 ms of the data is shown, the entire data set consists of 20 seconds of data. It is the same in Plots 3a and 3b. In Plot 3a, the number of points in the neighborhood is varied, and a single iteration of *getpeaks* is executed. The legend indicates the number of points considered and the computational time needed. A larger neighborhood results in the identification of fewer localized peaks. In Plot 3b, the number of points in the neighborhood is fixed at 2, but the number of iterations is varied. Depending on the nature of a particular dataset, it may be more beneficial to vary the size of the neighborhood or the number of iterations. These two controls allow the user to tailor the algorithm to provide the most efficient analysis for the data under consideration.

Temporal Moment Analysis Methods

XYData includes an implementation of D.O. Smallwood's temporal moment algorithms.[4] The first five moments are computed. In addition, the band-limited temporal moments can also be computed by XYData. Temporal moments provide another means for extracting signal features from data. These algorithms can be used to provide additional information to augment the weaknesses in other signal analysis methods

Input/Output Capabilities

One of XYData's strong suits is its ability to interface to a variety of data sources. Loading algorithms currently exist to load data generated by finite-element programs, oscilloscopes, ASCII data, several data acquisition systems and photographic motion analysis programs. To facilitate the two-way flow of information, data can also be output as finite-element analysis program load curves, ASCII data and reference spectra for control systems.

A graphical user interface (GUI) was also developed to provide the user a quick method of reading in data and displaying the metadata sorted in a variety of ways. The data is then easily plotted with some analysis capabilities available. This facilitates the user comparing a large number of data records in a meaningful manner from a variety of different data sources. From this type of "bird's-eye" view of the data, the analyst can then determine which more detailed analyses should be executed on which records.

Future Expansion

Although several years of development have gone into XYData, there is still a significant amount of functionality that should be added to it. The framework of the toolbox is well established, and there is plenty of opportunity to implement various algorithms. Additionally, the toolbox will always benefit from the addition of new data loaders.

The addition of the GUI to the XYData toolbox is relatively recent. The primary focus on the GUI had been the presentation of the metadata in useful and convenient format that allowed arbitrary sorting of the data. With that implemented, additional analysis capabilities should be incorporated into the GUI so that the casual user need not develop M-files to execute various analyses.

The third major area of expansion for XYData is to port it to open-source software. At this point, Octave seems to be the most likely candidate, since it is highly compatible with MATLAB and incorporates OOP capabilities. Although executing XYData in another application may result in a performance decrement, there is a definite benefit to eliminating the reliance on proprietary commercial software.

References

1. *Using MATLAB Version 6*, The Mathworks, Natick, MA, 2002.
2. MIL-DTL-9479E, Detail Specification Seat System, Upward Ejection, Aircraft, General Specification for, September 17, 1999.
3. Smallwood, D. O.,"An Improved Recursive Formula for Calculating Shock Response Spectra," *Shock and Vibration Bulletin: Proceedings of the 51st Symposium on Shock and Vibration*, San Diego, CA, 1980.
4. Smallwood, D. O., S*hort Course Notes*, Sandia National Laboratories, Jan 1990.

The author can be reached at: morris.berman@us.army.mil.