Applications of DSP to Explicit Dynamic FEA simulations of elastically-dominated impact problems

Ted Diehl[a], Doug Carroll[b] and Ben Nagaraj[c]
[a]*Mechanical Technology Center, PCS, Motorola, 8000 W. Sunrise Blvd., Fort Lauderdale, FL 33322, USA*
Tel.: +1 954 723 8024; Fax: +1 954 723 5584;
E-mail: Etd012@email.mot.com
[b]*Advanced Technology, Smart and Connected Products, Motorola, 1500 Gateway Blvd., Boynton Beach, FL 33426, USA*
Tel.: +1 561 739 3818; Fax: +1 561 739 3486;
E-mail: Edc002@email.mot.com
[c]*Mechanical Technology Center, PCS, Motorola, 8000 W. Sunrise Blvd., Fort Lauderdale, FL 33322, USA*
Tel.: +1 954 723 3098; Fax: +1 954 723 5584;
E-mail: R10887@email.mot.com

Received 16 August 1999
Revised 6 April 2000

Explicit Dynamic Finite Element techniques are increasingly used for simulating impact events of personal electronic devices such as portable phones and laptop computers. Unfortunately, the elastically-dominated impact behavior of these devices greatly increases the tendency of Explicit Dynamic methods to calculate noisy solutions containing high-frequency ringing, especially for acceleration and contact-force data. For numerous reasons, transient FEA results are often improperly recorded by the analyst, causing corruption by aliasing. If aliasing is avoided, other sources of distortion can still occur. For example, filtering or decimating Explicit Dynamic data typically requires extremely small normalized cutoff frequencies that can cause significant numerical problems for common DSP programs such as MATLAB. This paper presents techniques to combat the unique DSP-related challenges of Explicit Dynamic data and then demonstrates them on a very challenging transient problem of a steel ball impacting a plastic LCD display in a portable phone, correlating simulation and experimental results.

1. Introduction

Impact and drop analysis of personal electronic devices, such as portable phones and laptop computers, differs dramatically from the more established analysis of car crashes. Whereas a car crash is dominated by plasticity, the impact behavior of these electronic devices is dominated by elasticity. When a portable phone is dropped to the floor, its housing usually does not dent or crush like when a car smashes into a rigid barrier. Without sufficient inelastic deformation to dissipate energy, the resulting solution variables for an elastically-dominated impact problem can contain significant amounts of high-frequency energy, often leading to extremely noisy results when solved using Explicit Dynamic FEA codes. Acceleration and contact force have been the most susceptible to noise, and to a lesser extent, velocity, strain, and stress.

Attempts to validate predictions from Explicit Dynamic models of elastically-dominated structures using physically measured accelerometer data usually result in very poor correlation. There are several reasons for this, both on the experimental and simulation side. Concentrating on the simulation error sources, one of the most likely and frequent problems is corruption of the FEA data by aliasing. While antialiasing filters are common in most (but not all) modern digital data acquisition systems, it is very important for everyone to understand that there is no such safety feature in commercial Explicit Dynamic FEA programs! Whereas experimental systems deal with 10 or maybe hundreds of transient data channels, the FEA codes typically deal with 10^4 to 10^7 (or more) "data channels". The nature of these codes currently makes it infeasible to have antialias filtering internal to the program. Many of the FEA vendors that supply these Explicit codes do not provide proper tools and training to correctly deal with highly transient data. Hence, it is up to the FEA user to properly store and process their transient data –

Shock and Vibration 7 (2000) 167–177
ISSN 1070-9622 / $8.00

something which is not a trivial task. It is important to note that it has been the observation of these authors that many engineers doing such impact analyses with Explicit Dynamic codes have had little, if any, training in the processing of highly transient signals. Digital Signal Processing (DSP) is not commonly a part of a Mechanical Engineer's core classes.

To provide a base of understanding for the DSP concepts used here, the reader is directed to [3,4] for an applied overview. Additional details on DSP theory are found in [1,6,7,9,11–14]. Lastly, all the DSP related calculations presented here are performed using Diehl's DSP Extensions [2], an easy-to-use extension pack that works with Mathcad, a mathematical and engineering program for the MS Windows operating system. The extension pack is specially designed for transient impact analysis and can be downloaded for free at http://mathcad.adeptscience.co.uk/dsp/.

2. Unique challenges in storing and processing Explicit Dynamic FEA data

A typical portable-phone drop analysis might simulate 5 milliseconds of physical time using an average time increment of 0.1 microseconds (due to solution stability requirements). Each variable in the simulation is a digital signal containing approximately 50,000 data points (approximately 0.2 Megabytes, single precision). Considering that a typical model may easily contain over 10^6 variables (acceleration, velocity, displacement, stress, strain, etc.), that implies a total of 5×10^{10} data points are computed for the model (approximately 186 Gigabytes, single precision). In general, this amount of data is infeasible to store in its entirety. Out of necessity, a common approach used in the FEA community is to simply request results output at some time interval that is much greater than the actual solution's time increment, reducing the data size down to say a couple hundred points per variable. For noisy variables such as acceleration or contact forces, this approach frequently results in corruption by aliasing.

Besides the huge amount of data that is generated, there are several additional challenges to working with Explicit Dynamic FEA data. The stability requirement of the central difference integration scheme typically requires the time increment for these models to be two to four orders of magnitude greater than the desired frequency content of interest for the structure. Applying DSP filters to this type of heavily oversampled data can cause significant numerical distortions. Worst yet, the time increment throughout the solution is not constant; it is always changing to achieve as efficient a solution as possible. Thus, the data must be regularized to a time vector that has a constant time increment (constant sampling rate) before any DSP can be applied.

2.1. *Potential problems with digital filtering*

To properly deal with this transient data, DSP programs must be used. The most significant DSP method that is employed is lowpass filtering. This section demonstrates several undesirable filter distortions that various commercial programs create when filtering Explicit Dynamic data. To demonstrate the problems, simple test signals are passed through lowpass filters in various programs. While the signals tested here seem trivial, they will clearly point out many issues. More complicated, real-world signals are evaluated in the ball impact problem at the end of this paper.

For the filtering analyses, the following definitions are used. Normalized frequency Ω and normalized cutoff frequency Ω_c are defined as

$$\Omega = \frac{\omega}{\omega_s} \qquad \Omega_c = \frac{\omega_c}{\omega_s} \tag{1}$$

where ω is frequency, ω_s is the sampling rate, and ω_c is the cutoff frequency. (Note that these normalized frequency definitions are different than those used by MATLAB.) The generic z-domain transfer function H(z) for a Infinite Impulse Response (IIR) filter is defined as

$$H(z) = \frac{B_0 + B_1 z^{-1} + B_2 z^{-2} + \ldots + B_L z^{-L}}{1 + A_1 z^{-1} + A_2 z^{-2} + \ldots + A_L z^{-L}} \tag{2}$$

where L is the filter order, vectors A and B are filter coefficients, and z is a complex variable related to frequency. If the A coefficients are all zero, a Finite Impulse Response (FIR) filter is realized. A direct method to compute the filter's gain over a range of frequencies Ω is simply

$$\text{Gain} = H\left(z = e^{2\pi j \Omega}\right) \tag{3}$$

where $j = \sqrt{-1}$ and an $e^{j\omega t}$ time dependence is assumed. Similar to the use of Laplace transforms with continuous systems, we use z-transforms with digital signals to map between the frequency and time-domains. Through the z-transform, we can implement digital filters entirely in the time-domain. Given a discrete time domain data sequence x_i (digital signal) and filter coefficient vectors A and B, the filtered time do-

main response y_i is simply computed as (Ifeachor [7, p. 143])

$$y_i = \sum_{j=0}^{L} B_j x_{i-j} - \sum_{j=1}^{L} A_j y_{i-j} \tag{4}$$

To improve numerical accuracy, especially for cases of very small normalized cutoff frequencies (say $\Omega_c < 0.01$), the actual numerical implementation of an Lth-order filter should be done as a cascade of 2nd-order filters (see [14, p. 100], for more details). While this distinction may seem trivial, it is important to realize that many DSP programs are written with the expectation that the data is not "enormously oversampled". The idea that you may sample at 10 MHz and be interested in data at or below 5 kHz will seem foolish to many DSP programmers. Hence, it is likely that this condition will not be tested for in their codes. However, this is the exact condition that the FEA analysts finds themselves in when modeling these elastically-dominated structures.

For time domain analyses, undesirable filter-induced distortions can be created by filter start-up and ending effects. These are caused by the common assumption that the signal to be filtered has zero amplitude for all time before and after the signal. While this common assumption is correct for most DSP applications, it is not correct for many FEA analyses. For example, stress and strain components for a preloaded structure have nonzero values at the beginning of the analysis. In a drop analysis that begins just prior to impact, the initial velocity will be nonzero. In almost all FEA simulations, the computation will be stopped prior to most variables settling to a steady state or zero value. Filtering any of these signals that violate the "zero value" assumption can cause transient distortions because of the inherent non-smooth transition between the assumed zero amplitude region before the signal's beginning and the actual signal content itself. Similar distortions can occur at the end of the signal. This type of error can be minimized by artificially projecting the original signal back in time by a finite amount. MATLAB's filtfilt function uses a reflected mirror algorithm to minimize start-up distortions. The filter algorithms used in Diehl's DSP Extensions offer several methods to minimize filter-induced distortions, including the following assumptions: zero, constant, reflected mirror, and a prediction algorithm based on Mathcad's "Predict" function [10]. In all cases for this section of the paper, the prediction algorithm is used by Diehl.

Figure 1 evaluates MATLAB's Signal Processing Toolbox, V5.1 [12] relative to results computed by Diehl's DSP Extensions [2]. The evaluation consists of defining an 8th-order Butterworth lowpass IIR filter, computing the magnitude of it's gain, and then filtering a sloped line and a 1.0 kHz sine wave (with a 70 degree phase shift). The sample rate for all the signals is 6.0 MHz. Two cutoff frequencies are studied, 30 kHz ($\Omega_c = 0.005$) and 18 kHz ($\Omega_c = 0.003$). All parameters are chosen to be representative of values seen in a typical Explicit Dynamic simulation.

Figure 1(a) shows the magnitude of the filter gains computed by Diehl's DSP Extensions (denoted as Diehl) and by MATLAB (using two different MATLAB algorithms). The results from Diehl are quite accurate and not sensitive to the normalized cutoff frequency. Diehl simply evaluated the filter gain using a modified version of Equation (3), coded for a cascaded filtering approach as defined in [14]. The plots show that MATLAB can yield very poor results and that their algorithms' numerical stability are sensitive to normalized cutoff frequency. Neither of the MATLAB algorithms utilize a cascade filtering approach. Additionally important to note is that MATLAB provided no warning to the user that the gain computation would potentially have numerical problems. While a savvy DSP expert might have suspected potential problems with such small cutoff frequencies, a FEA analyst would likely not.

Figures 1(b) and (c) compare results from passing the test signals through the lowpass filters. Both the filter from Diehl and the MATLAB filter (using the filtfilt function) used a double-pass, zero-phase filtering approach (see [2,12] for details). Also, both filters use pre and post data projection algorithms to minimize end-effects (more to be said shortly). Since both signals have no frequency content above either cutoff frequency, the Ideal results after filtering should be the original signals. The results show that Diehl matches the benchmark while MATLAB has significant problems as the normalized cutoff frequency is made smaller. As before, Diehl used a cascaded filtering approach and MATLAB did not. This problem may seem trivial, but it is not because when filtering Explicit Dynamic data, values of normalized frequencies similar to these will be encountered. Furthermore, it is reasonable to expect low frequency content (structural deformation modes) to be contained in the signal and we would expect this frequency content to be undisturbed by a lowpass filter.

Figure 2 evaluates the filtering capabilities of ABAQUS/Post V5.8 [5]. The figure shows the time-domain results of some simple signals before and after lowpass filtering. The test signals depicted in (a)

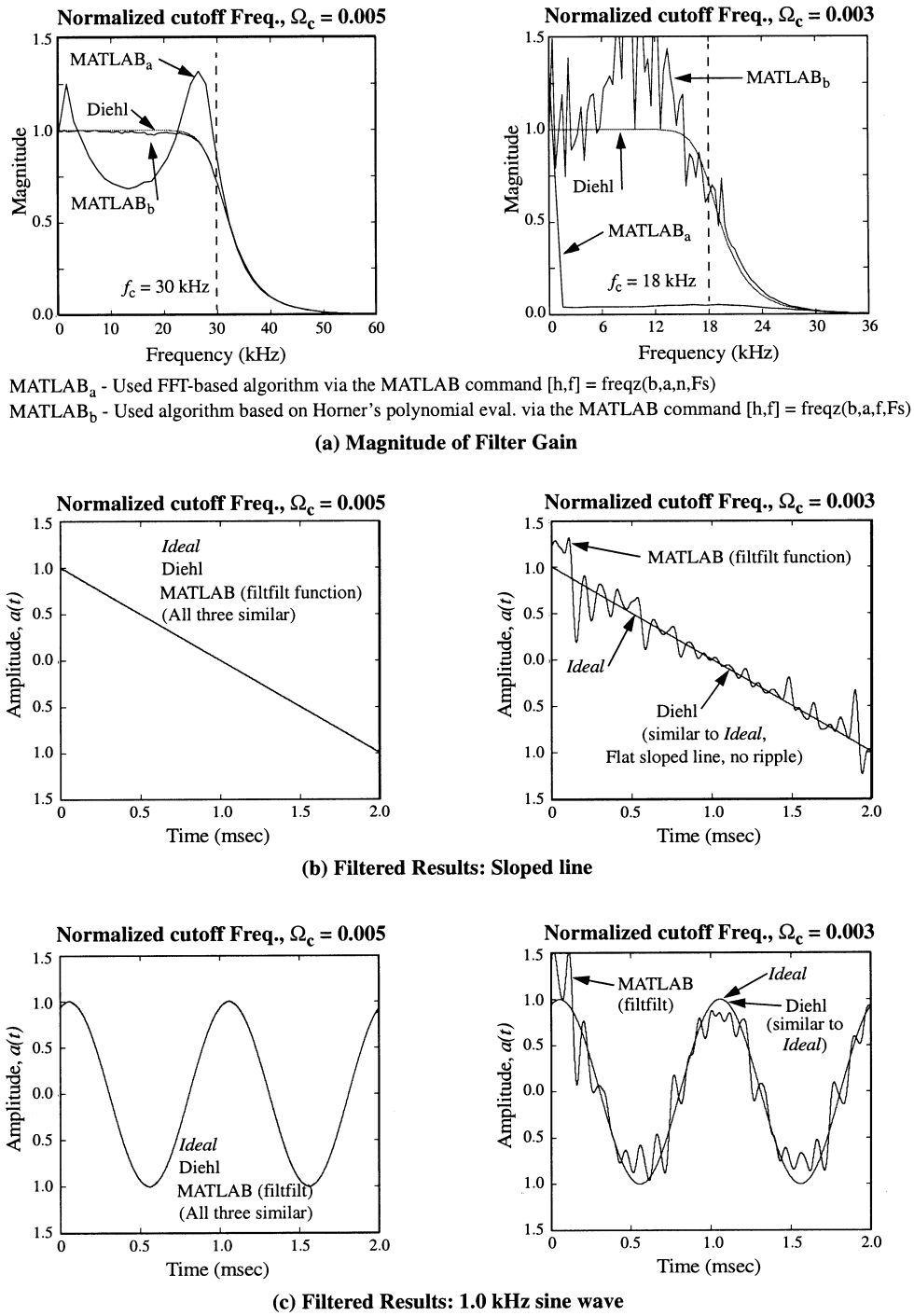

Fig. 1. Comparing MATLAB lowpass filtering and Diehl's DSP Extensions using some digital test signals. Sample rate of all signals is 6 MHz.

are a sloped line and three sine waves with frequencies of 1.3 kHz, 6.0 kHz, and 29.0 kHz (note that the sine waves are defined such that they begin with non-zero amplitude). The curves depicted in (b)–(e) show the ideal solution, results from ABAQUS/Post, and results from Diehl's DSP Extensions. ABAQUS/Post uses a single-pass sine-Butterworth IIR filter (a slight variation on the more common Butterworth filter) with no

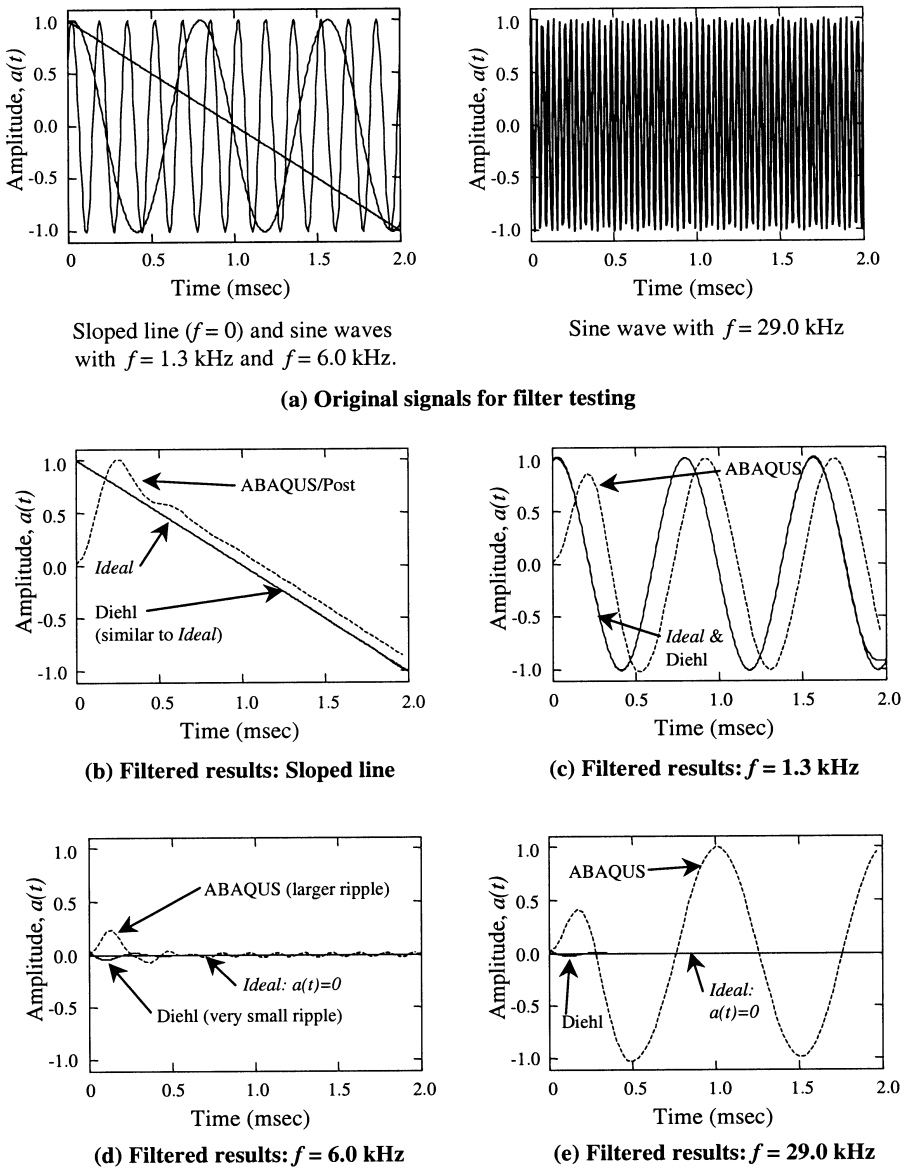

Fig. 2. Comparing ABAQUS/Post lowpass filtering and Diehl's DSP Extensions using some digital test signals. Sample rate of all signals is 240 kHz and cut-off frequency is 3.0 kHz.

filter distortion compensation. The filter from Diehl is a Butterworth double-pass zero-phase filter with filter start-up minimization. (Similar results are easily achieved with other common filters such as Cheby I or a sinc-based FIR.) Both ABAQUS and Diehl's filters were 6th-order. Figures 2(a–d) demonstrate the deficiencies of the ABAQUS filter implementation; end-distortions and time delay are clearly evident. The same figures show the relatively distortion-free results from Diehl.

Figure 2(e) shows a serious deficiency with the ABAQUS lowpass filter implementation, it produced an aliased result! ABAQUS/Post has a default resampling algorithm that will resample (decimate or interpolate) the original signal to a sample rate that is 10 times the specified cut-off frequency of the filter (The ABAQUS/Post manual [5] incorrectly states the resampling to be 5 times the cut-off freq). This resampling is done prior to applying the lowpass filter. In our test case, the 29.0 kHz sine wave, which originally had a sample rate of 240 kHz, was decimated to a sample rate of 30 kHz (10 times 3 kHz). The 30 kHz sample rate

caused the 29.0 kHz signal to be aliased to a 1.0 kHz signal. This resampled signal then passed through the 3 kHz filter "untouched", except for the filter distortions already discussed. These simple cases show that the ABAQUS filter implementation not only creates undesirable distortions but that it is very susceptible to inducing aliasing errors.

Other commercial programs have also been tested by the authors and many show deficiencies similar to those demonstrated in Figs 1 and 2. For example, LS-Taurus [8] (the post processor for LS-Dyna) aliases output when the user requests the program to output data to an ascii file.

While these issues may seem trivial, or are the result of special "trick" conditions, they are not. They are the result of the Explicit Dynamic FEA world and the DSP world coming together in an imperfect fashion. Until greater awareness of the these DSP issues are achieved in both communities, errors like the ones shown will continue to happen. The lesson learned from these studies is that one should check out any DSP functionality of a potential program with simple functions. Then you can fully understand what the program is (and is not) doing to your data.

2.2. Summary of proper sampling and decimation of Explicit Dynamic FEA data

The fundamental problem when dealing with transient FEA data is that we generally do not know the maximum frequency content of the various solution variables ahead of time. If we did, then we would know a safe sampling rate. Lacking knowledge of the maximum frequency content, the following template will help to insure aliased-free data. See [3] for more specifics relative to the use of ABAQUS.

1. For every variable of interest, output the result at *every* time increment. Since the resulting output file might get quite large, you will need to be selective on how many variables you output, say 10 or so. Additionally, the actual value of the change in time at each of the time increments should be requested directly form the solver and stored. Simply attempting to calculate the time increments from the stored ASCII output data of the total time vector might have significant errors if the output is only stored with six or less digits of accuracy, which is common for many programs.
2. Because the time increment in an Explicit Dynamic analysis changes throughout the solution, regularize the data to a constant time increment (DSP algorithms require this). The original data must be interpolated onto a new constant-increment time vector. Two reasonable choices for the time increment of the new time vector are the solution's average time increment or it's minimum time increment. The user should evaluate a plot of the time increment as a function of time to determine which one to select. When selecting this choice, two error sources must be considered: A) aliasing caused by selecting the average time increment if this value is much greater than the minimum time increment and B) interpolation error caused by selecting the minimum time increment if it is much smaller than the average increment and the minimum time increment is localized in time (occurs only a few times throughout the solution).
3. Reduce the regularized data sets down to 10 times the highest frequency of interest. To decimate the data safely, it must be first lowpass filtered to sufficiently attenuate all the frequency content that is above the Nyquist frequency of the desired (reduced) sample rate. Note, the cutoff frequency of the antialias filter must be sufficiently less than the ideal cutoff frequency (Nyquist frequency of desired sample rate) to account for the transition band of the filter. Once this frequency content is "removed", the data sets can be safely decimated to the new sample rate. To avoid filter distortions, special precautions discussed previously must be utilized.

Any program with appropriate DSP capabilities may be used, but they must be able to properly handle very small normalized cutoff frequencies (values on the order of $\Omega_c = 0.01$ or less). The functions in Diehl's DSP Extensions are capable of handling the filtering plus it offers functions to easily regularize and decimate the data.

Some additional practical things to remember. Responses such as acceleration and contact force will be very noisy, especially in solid elements. This type of data often has large amplitude, high frequency noise components. All data of this type that is to be evaluated should be post-processed in the manner described above. On the other end of the spectrum is displacement data. By its nature, high frequency displacement components have very low amplitude and therefore pose a

relatively low risk of alias corruption. Stress, strain, and velocity are in between these two cases.

Unfortunately, application of the process outlined above is not generally feasible for animated contours, as too much initial data must be stored. A reasonable approach around this limitation is to output data at a few representative nodes (or elements) within the contour by the method outlined above. Using this selected dataset, a safe sampling rate can be determined which will avoid aliasing. Then the entire data set can be stored (output) from the solution using this safe sampling rate. Remember, you must be sure that there is no significant frequency content in the solution variables of interest greater than one half of your output frequency. If there is, aliasing will occur which you will not be able to detect nor correct.

3. Ball bearing impact on a portable phone lens

Figure 3 depicts a very challenging ball bearing impact problem. A plastic housing with plastic display lens is supported at 4 bosses and subjected to the impact of a 130 g, 31.75 mm diameter steel ball which is dropped from 500 mm. The quantities of interest are the len's transient responses of acceleration and displacement under the point of impact (back side of the lens). The objective of this example is to correlate both experimental and FEA results.

The Explicit Dynamic FEA model is composed of shell elements for the housing and solid elements for the variable thickness lens (three elements through the thickness). The plastic material is modeled using only Hooke's law (no plasticity or viscous effects). The model is solved using ABAQUS/Explicit. The experimental measurement of the acceleration utilized an Endevco 2255B-01 Isotron accelerometer connected to an Endevco model 133 Signal Conditioner with all the Conditioner's HP & LP filters turned off. The lightweight accelerometer has a resonance of 300 kHz and an internal 2-pole Butterworth lowpass analog filter with an approximate cutoff frequency of 28 kHz. The experimental acceleration was captured, alias free, at a sample rate of 250 kHz and is displayed in Fig. 3(c). The ABAQUS/Explicit prediction of acceleration from a single node on the bottom side of the lens, directly under the point of impact, is displayed in Fig. 3(d). This data has already been regularized per the methods outlined in Section 2.2. (The regularized and raw FEA data are very similar and would be indistinguishable on this plot.) Figures 3(e–f) show these results plotted in the frequency domain (The frequency spectrum of the time-domain signal was computed with Diehl's DSP Extensions). The question we need to answer is "Does the simulation and experiment correlate"? Based on the data presented in Fig. 3, we would have to say "no". Let's try to further analyze the data to see if things improve.

A common approach to improve correlation is to apply a lowpass filter to remove noise that is likely present in both the experiment and the simulation. A common cutoff frequency used for this type of structure might be between 1 kHz and 10 kHz. For our case, we will choose 5 kHz. Using the rule of thumb of 10x, we decide that our desired sampling frequency should be 50 kHz. Figure 4 presents what happens if we just sample the raw FEA data at 50 kHz without protecting against aliasing. The experimental acceleration (sampled at 250 kHz) and the FEA acceleration (sampled at 50 kHz) displayed in Fig. 4(a) definitely do not correlate. Figure 4(b) presents the results after lowpass filtering with a 5 kHz Butterworth filter. To match filter responses for the two data sets, the experimental data utilized an 8th-order Butterworth and the simulation data used a 7th-order Butterworth. To match filter responses as close as possible, different filter orders are used because the two data sets had different sampling rates. (Remember, filter responses are a function of the normalized cutoff frequency.) Even after filtering, the two acceleration data sets look completely unrelated. Lastly, we compare displacement data (Fig. 4(c)). Displacements for the experiment are computed by double integrating the experimental acceleration signal. The FEA code computes displacements directly. Interestingly enough, the raw FEA displacement curve and the integrated experimental displacement curve correlate quite well. However, acceleration and displacement are directly related and we observed that the accelerations were completely different! Also plotted in Fig. 4(c) are "integrated FEA displacements" computed by double integration of the FEA acceleration curves from Fig. 4(a–b). Both of these integrated results are very different from the raw displacements computed directly by ABAQUS. How could that be? The answer is that all the FEA acceleration data in Figure 4 is aliased! It is aliased because we sampled the acceleration data without first removing the high frequency content above 25 kHz (half of 50 kHz). Figure 3(f) (regularized based on sampling every solution increment) shows that the original acceleration signal contains significant frequency content up to approximately 1 MHz. This extremely high frequency content is caused by, among other things, the individual element vibrational modes.

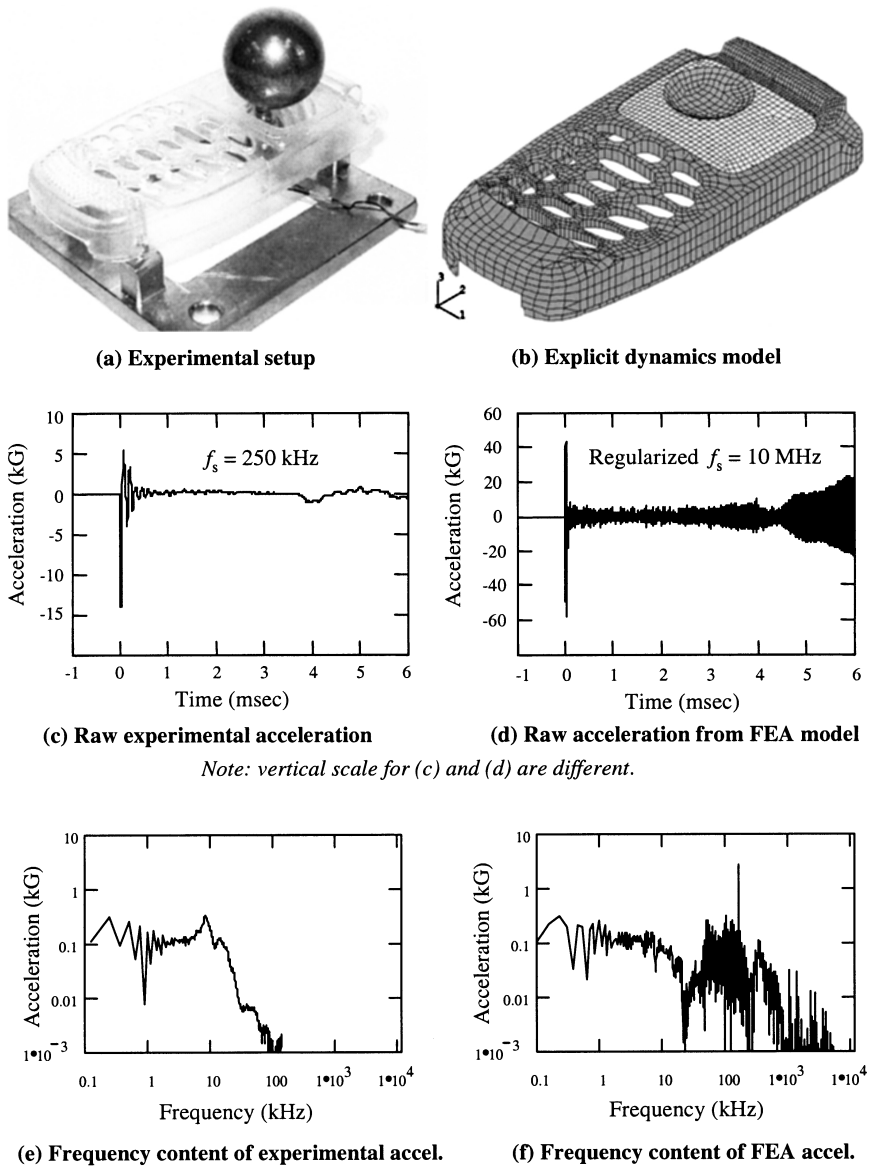

Fig. 3. Ball bearing impact example, experiment and /Explicit FEA model.

Figure 5 presents data that is properly decimated to 50 kHz by the process described in Section 2.2. In addition, the experimental data has also been decimated down to a sample rate of 50 kHz. The results in Fig. 5(a), before the 5 kHz LP filter, look very promising. After filtering, the correlation between the simulation and experimental accelerations is excellent (Fig. 5(b)). Now, the integrated displacements for the simulation match the raw FEA displacement data (Fig. 5(c)). An important thing to note is that the success of obtaining correlation was dependent on proper sampling technique, not filter form (IIR or FIR). Equally good results are obtained with a Cheby I IIR or sinc-based FIR filter provided that the "proposed filtering method" described previously is used and that the filter parameters are defined such that the magnitude of the filter responses are similar.

4. Conclusions

It is imperative that highly transient Explicit Dynamic FEA solution data is properly handled, especially for elastically-dominated impact problems. Proper DSP

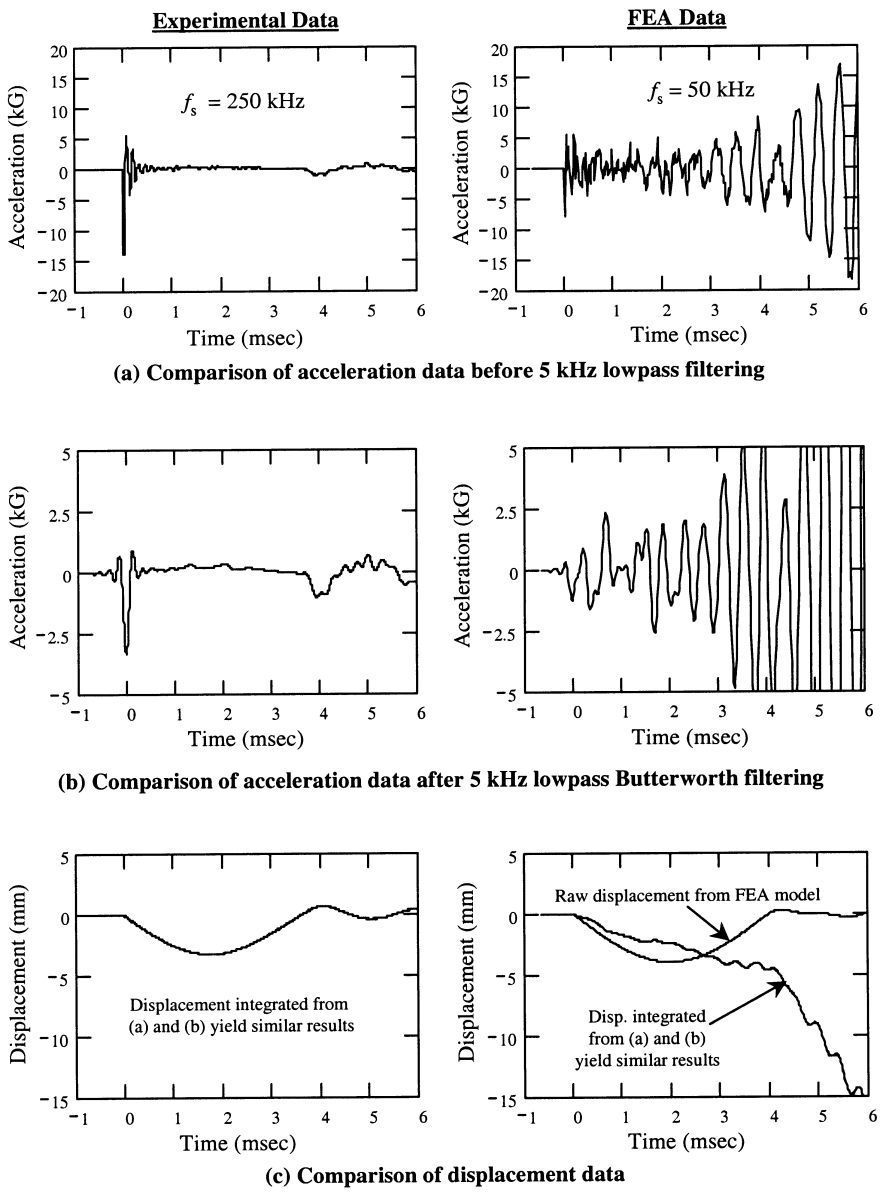

Fig. 4. Processing data without protecting against aliasing. FEA data is simply requested from solution at a 50 kHz sample rate.

processing can make the difference between realistic and nonsensical conclusions. The key DSP issues related to this application are:

1. The proper method to process transient data and avoid aliasing errors is to output the desired FEA solution variables at every solution increment, regularize this data, and then decimate the data (incorporating an antialias lowpass filter) to a manageable sample rate. Applying this approach to large data sets, such as animations of contours, is presently not feasible. For large data sets, this approach should be applied to a small selected set to determine a sufficient sample rate that would avoid aliasing for the entire set.
2. Quantities such as acceleration and contact force are the most susceptible to alias errors, and to a lesser extent, velocity, strain, and stress. Displacements are the least susceptible to alias errors. In general, it is impossible to determine if a given sampled signal has aliased frequency content without detailed knowledge of the original signal prior to sampling. Even smooth, low-frequency

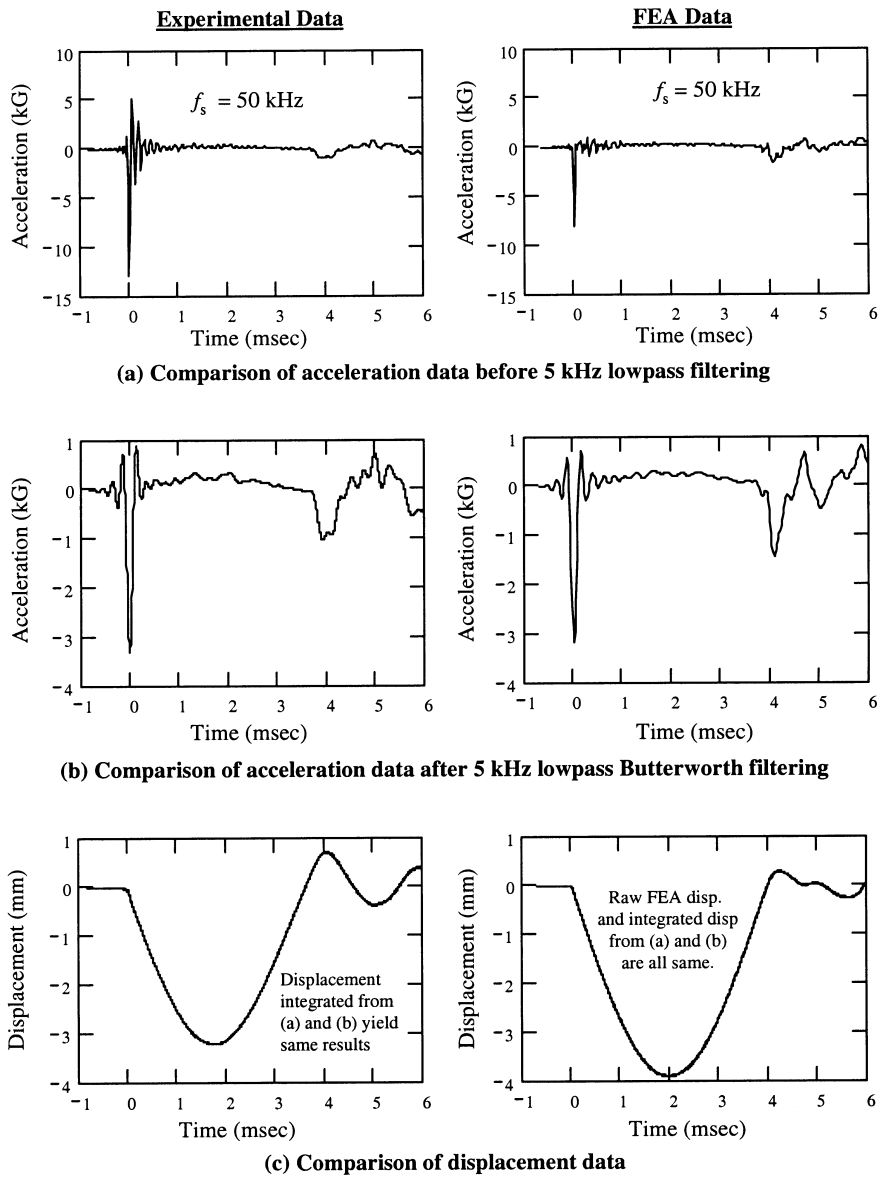

Fig. 5. Processing data with antialias filtering. FEA data is properly decimated down to a 50 kHz sample rate.

signals can be corrupted by aliasing.

3. Users need to be aware of the many problems that can occur when processing and filtering Explicit Dynamic FEA data. The normalized cut-off frequencies can be quite extreme. As a result, many commercial programs that offer DSP features may not properly process Explicit Dynamic FEA data. Vendors of Explicit Dynamic FEA programs need to significantly improve their ability to process highly transient data.
4. When proper DSP methodology is utilized, excellent correlation between experimental and explicit dynamic FEA data can be achieved, even at very challenging locations such as the point of impact.

Acknowledgments

The authors wish to sincerely thank colleagues Dr. Dave Yeager and Dr. Jason McIntosh for their numer-

ous discussions of and insight into the world of DSP. Additional appreciation is owed to Dr. McIntosh for his efficient C^{++} coding of many of the algorithms used in this work.

References

[1] Brillouin, L., *Wave Propagation and Group Velocity,* Academic Press, 1960.

[2] Diehl, T., *Diehl's DSP Extensions,* Available for download at http://mathcad.adeptscience.co.uk/dsp/, 1999.

[3] Diehl, T., Nagaraj, B. and Carroll, D., Using Digital Signal Processing (DSP) to Significantly Improve the Interpretation of Drop/Impact Simulation Results for Personal Electronic Devices: Part I – Theory, *70th Shock & Vibration Symposium, November 15–19,* Albuquerque, NM, SAVIAC.

[4] Diehl, T., Nagaraj, B. and Carroll, D., Using Digital Signal Processing (DSP) to Significantly Improve the Interpretation of Drop/Impact Simulation Results for Personal Electronic Devices: Part II – Applications, *70th Shock & Vibration Symposium, November 15–19,* Albuquerque, NM, SAVIAC.

[5] Hibbitt, Karlsson and Sorensen, *ABAQUS /Post,* V5.8.

[6] IES, *Handbook for Dynamic Data Acquisition and Analysis,* Institute of Environmental Sciences, ISBN 1-877862-47-9, no copyright date listed in document.

[7] Ifeachor, E. and Jervis, B., *Digital Signal Processing: A Practical Approach,* Addison-Wesley, 1993.

[8] Livermore Software Technology Corp., *LS-Taurus User's Manual,* 1997.

[9] Madisetti, V. and Williams, D., *The Digital Signal Processing Handbook,* CRC Press, 1998.

[10] Mathsoft, *Mathcad 7 User's Guide,* Mathsoft Inc., 1997.

[11] Mathsoft, *Mathcad Signal Processing Function Pack,* Mathsoft Inc., 1998.

[12] Math Works, *Signal Processing Toolbox For Use with MATLAB: UserŠs Guide,* The Math Works Inc., 1998.

[13] Oppenheim, A. and Schafer, R., *Digital Signal Processing,* Prentice-Hall, 1975.

[14] Stearns, S. and David, R., *Signal Processing Algorithms,* Prentice-Hall, 1988.

[15] Ziemer, R., Tranter, W. and Fannin, D., *Signals and Systems: Continuous and Discrete,* Macmillan Publishing, 1983.