Measurement Notes

Note 56

January 2001

The Complete Fast Fourier Transform and Cascaded Transition-Band Filters to Reduce the Noise of Deconvolution

Paull E. Patterson, Ktech Corp.

High Power Electromagnetics Department
Sandia National Laboratories
Albuquerque, NM 87185

Abstract

A measurement system's components: cabling, delay line, waveform recorder, etc., degrade acquired signals and their respective bandlimited frequency responses. Compensation software corrects for this frequency-dependent spectral degradation by deconvolving the transfer function of the entire measurement system out of the measured signal spectra.

This report describes methods to transfer the characteristics of a wide bandwidth repetitive sampling oscilloscope to a single-shot transient digitizer, characterize the measurement system, develop a cascaded transition-band filter, and compensate data acquired with the filtered, characterized measurement system. These procedures are easily implemented, execute quickly, and successfully compensate waveforms possessing endpoint discontinuities.

Waveforms possessing endpoint discontinuities are made to appear duration-limited and continuous. The spectra for these modified waveforms are correct, including at dc. The deconvolution process introduces unavoidable noise. Filtering is applied to reduce the deconvolution noise while minimally affecting compensated waveform risetime and amplitude. Resultant compensated data retains its initial dc baseline offset with improved waveform fidelity and low noise of deconvolution.

Contents

Figures

The Complete Fast Fourier Transform and Cascaded Transition-Band Filters to Reduce the Noise of Deconvolution

Overview

This report discusses the 'Complete FFT' and describes software procedures for a cascaded transition-band filter. Their use in measurement system characterization and data compensation is illustrated. FORTRAN codes for the described procedures are included as appendices to this report.

The Signal Compensation Problem

Although each element of a measurement system should be, within reason, the finest available; ultimately, the system bandwidth can be no better than that of the poorest element or measurement.

While the bandwidth of high-quality signal cable far exceeds that of typical measurement instruments, the chief culprits causing signal degradation in cables are phase dispersion and the fact that attenuation per unit length of cable increases with frequency. Minimization of signal cable length and careful attention paid to measurement techniques are of paramount importance. An unavoidable exception is encountered when instrument throughput times require a minimal jitter trigger signal derived from the signal to be measured. This requires a lengthy signal delay line incorporating an oscilloscope 'trigger pickoff'.

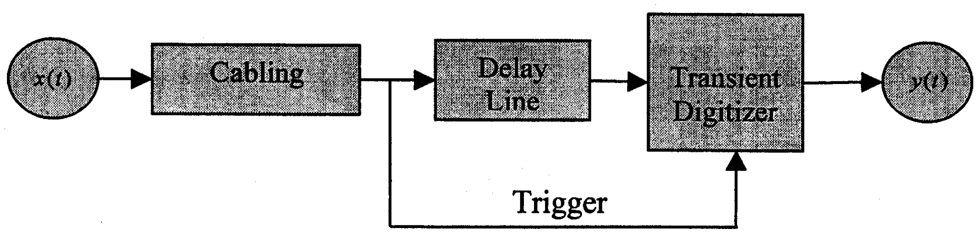

Figure 1. Diagram of a typical wide-bandwidth measurement system.

Traditionally, signal compensation has been accomplished by constructing and applying hardware compensation devices, such as self-integrating monitors or filters, to the measurement system. Note that hardware compensation corrects only for signal corruption by the measurement system up to, but not including, the measurement instrument (transient digitizer or oscilloscope). Furthermore, hardware compensation can be difficult to implement and the data acquired and saved after applying hardware compensation has embedded in it, forever, the characteristics of the compensation network.

The benefits of signal compensation in software become more apparent with each compromise made to accommodate an assortment of measurement system hardware deficiencies [1].

Description of Signal Compensation Software

For the purposes of the software compensation algorithms, the measurement system is defined to include the entire signal path, consisting of cabling, delay lines, and a single-shot transient digitizer.

There are three distinct procedures involved in this software compensation method: system characterization, filter development, and subsequent signal compensation of data.

1. System characterization develops an estimate of the measurement system's impulse response (or transfer function in the frequency domain).

2. Filter development strives to reduce noise of deconvolution in the measurement system's impulse response and data acquired with this system.

3. Signal compensation deconvolves the filtered measurement system's impulse response from acquired waveforms to estimate their true input signals. (Implemented in the frequency domain.)

Before performing software compensation of signals, the measurement system must be characterized. Generally, compensated signals require filtering due to 'noise of deconvolution'. Rather than applying the filter to the data, it is decided to filter the system through which the data flows. In either case, before performing data compensation, the additional step of developing a suitable filter must be undertaken.

This report describes methods to transfer the characteristics of a wide bandwidth repetitive sampling oscilloscope to a single-shot transient digitizer, characterize the measurement system, develop a cascaded transition-band filter, and compensate data acquired with the filtered, characterized measurement system.

The 'Complete' Fast Fourier Transform

Discussion of the validity of "duration-limiting" discrete data is beyond the scope of this report and the reader is directed to references [2, 3, 4].

Generally, all signals used in the characterization and compensation procedures will possess endpoint discontinuities. Some method must be used to avoid the introduction of errors in FFT processing due to these discontinuities.

The complete FFT, as described elsewhere [5], is a method to compute the frequency-domain transform of a waveform (with step-like features) by first making it 'duration-limited' (no endpoint discontinuities). It is a hybrid method combining the results of the Nicolson de-ramping [2] and Gans-Nahman [3] methods. Basically, the even and odd harmonics from the two methods are interleaved to form the complete FFT. The result is a spectrum with a correct dc value, possessing twice the spectral resolution of an FFT processed in the usual manner. For the procedures described in this report, the complete FFT is used for measurement system characterization and cascaded transition-band filter development.

A FORTRAN routine for this method is in included as an appendix to this report and has been verified by time-domain deconvolution and traditional sine-wave testing.

Measurement System Characterization

The system transfer function, $H(j\omega)$, is developed in the characterization of the measurement system. While $H(j\omega)$ includes the effects of cabling, delay-line, and transient digitizer, it is neither necessary nor desirable to measure these effects individually. Note that $H(j\omega)$, the exact transfer function of the measurement system, must only be considered an estimate because of unavoidable noise of deconvolution.

Riad [6], Miller [7], and Brigham [8] discuss estimation and the problems encountered in deconvolving the measurement system's excitation from its response ($x(t)$ and $y(t)$, respectively) to yield $h(t)$, the impulse response for the measurement system. Nicolson [9], Cronson and Mitchell [10], and Andrews [11] describe procedures for measuring $h(t)$ on a time-domain basis.

All spectra developed in this procedure make use of the complete FFT.

Fast, stable, high-output step generation devices are readily available [12] and particularly well suited for measurement system characterization since the system insertion losses for frequencies within the passband (lowpass) can be confidently and easily determined. Additionally, measurement system throughput and waveform degradation is readily discernible [1, 13].

A calibrated excitation waveform, $x_c(t)$, is passed through a measurement system and is measured as $y_c(t)$. The symbol '$*$' denotes convolution in the time domain.

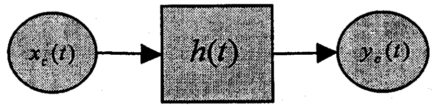

Figure 2a. Diagram of $y_c(t) = h(t) * x_c(t)$.

$y_c(t)$ is the transient digitizer time-domain response (averaged over a nominal 128 single shots) taken at the output of the measurement system during characterization.

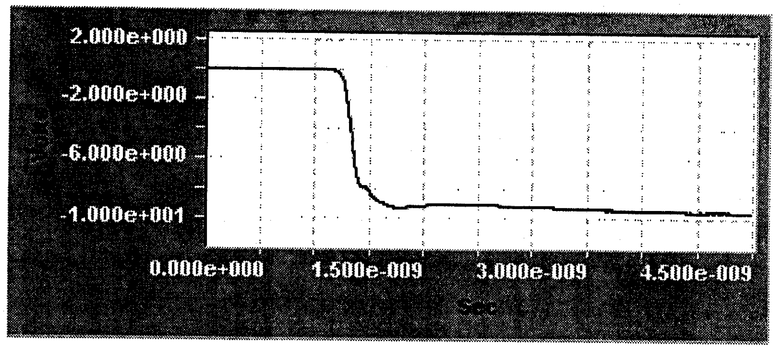

Figure 2b. $y_c(t)$ acquired at output of measurement system.

Now, measure $x_c(t)$ with a sampling oscilloscope possessing a much wider bandwidth:

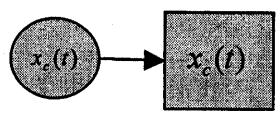

Figure 3a. Diagram of 'true' $x_c(t)$ measured with sampling scope.

The measured signal, $x_c(t)$, is the sampling oscilloscope *composite* measurement which represents the 'true' $x_c(t)$ characterization signal as input to the measurement system. By *composite* is meant the averaging of multiply-acquired sampled waveforms.

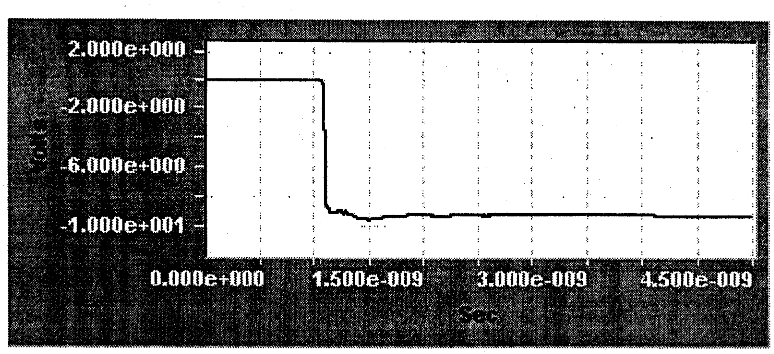

Figure 3b. 'True' $x_c(t)$ as measured with sampling scope.

It must be assumed that the responses, $x_c(t)$ and $y_c(t)$, are obtained from the same step waveform (a single shot) and are simultaneously attained from the measurement system's input and output ports, respectively. In practice, this is neither achievable nor necessary provided that a stable characterization source is available.

Convolution in the time domain corresponds to multiplication in the frequency domain.

$$Y_c(j\omega) = H(j\omega) \cdot X_c(j\omega)$$

The system transfer function, defined to be $H(j\omega) = Y(j\omega)/X(j\omega)$, is computed as the ratio of the spectra of the acquired (measured) signal, $y_c(t)$, and the true characterization signal, $x_c(t)$. Here $Y_c(j\omega) = fft\{y_c(t)\}$ and $X_c(j\omega) = fft\{x_c(t)\}$. It is assumed that the very wide bandwidth measurement of $x_c(t)$ is sufficiently accurate in comparison to other errors and noise introduced by this processing so that the measured signal, $x_c(t)$, can be used to represent the true characterization signal. Thus, the characterization procedure yields the transfer function, $H(j\omega) = Y_c(j\omega)/X_c(j\omega)$, which incorporates the wide bandwidth measurement of the characterization signal, $x_c(t)$, with its estimated spectrum, $X_c(j\omega) = fft\{x_c(t)\}$. Lawton, et al. [14], and Andrews [12, 15 and 16] describe correct time-domain measurement procedures.

To characterize the measurement system, find its transfer function, $H(j\omega)$.

$$H(j\omega) = \frac{Y_c(j\omega)}{X_c(j\omega)}$$

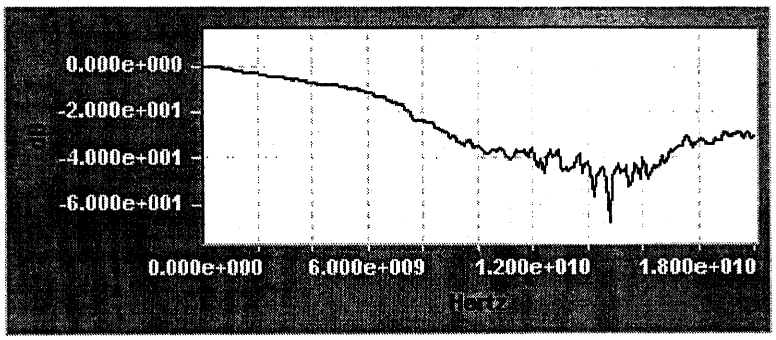

Figure 4. $H(j\omega)$, the measurement system's transfer function. (detail)

Noise of deconvolution arises from the previous operation as the poles and zeroes of $Y_c(j\omega)$ are, generally, those of $X_c(j\omega)$, leading to indeterminacies at these points in the calculation of $H(j\omega)$ [17].

Needed is $C_\lambda(j\omega)$, a Wiener transition-band filter [18], to reduce this noise.

$$C_\lambda(j\omega) = \frac{|X_c(j\omega)|^2}{|X_c(j\omega)|^2 + \lambda}$$

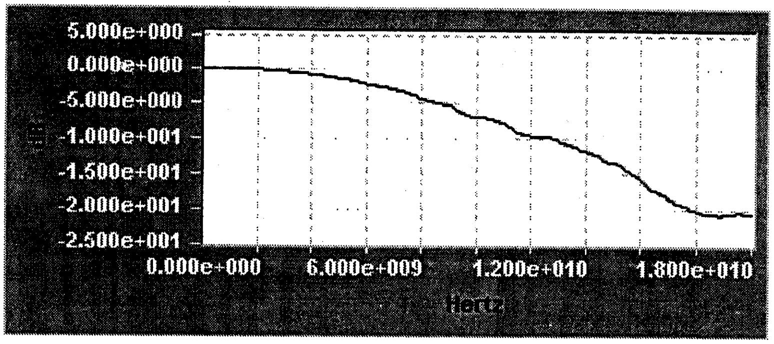

Figure 5. dB magnitude plot of $C_\lambda(j\omega)$. (detail)

An estimate of the measurement system's transfer function with reduced noise of deconvolution is formed by the product of $H(j\omega)$ and $C_\lambda(j\omega)$.

$$\hat{H}(j\omega) = H(j\omega) \cdot C_\lambda(j\omega)$$

Shown is the measurement system's estimated impulse response, $\hat{h}(t) = ifft\{\hat{H}(j\omega)\}$.

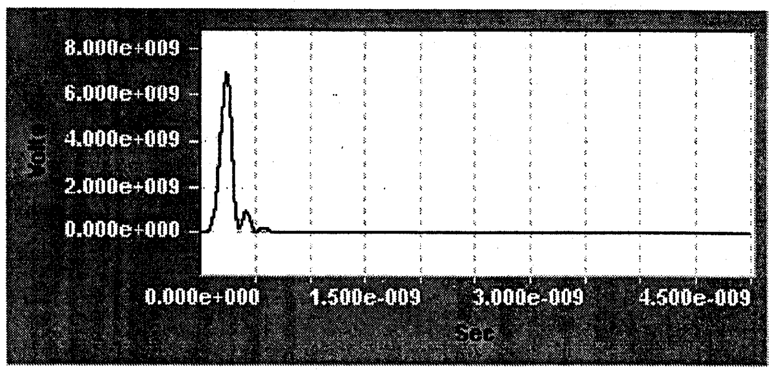

Figure 6. $\hat{h}(t)$, the estimated impulse response for the measurement system.

It remains to find a value for λ that gives a satisfactory estimate of $\hat{h}(t)$. Bennia and Riad [18] describe a method to obtain a value for λ that when used with $C_\lambda(j\omega)$ will give an estimate, with reduced noise of deconvolution and good fidelity, for the measurement system's impulse response. An alternate method for determining λ is described in this report.

A Cascaded Transition-Band Filter

The transition-band filter, described in the previous section, reduces noise of deconvolution in the measurement system's impulse response. A second transition-band filter will be used when deconvolving the filtered, characterized measurement system from the data to be compensated. Used together, these filters comprise the cascaded transition-band filter.

All spectra used in this procedure are obtained using the complete FFT.

A single-shot waveform, $x_m(t)$, is acquired with the measurement system as $y_m(t)$.

$$y_m(t) = h(t) * x_m(t)$$

The measurement system is deconvolved from the acquired data; this is performed by division in the frequency domain after first finding $Y_m(j\omega) = fft\{y_m(t)\}$. $H(j\omega)$ was determined during system characterization. No filter has been applied and noise of deconvolution is clearly evident.

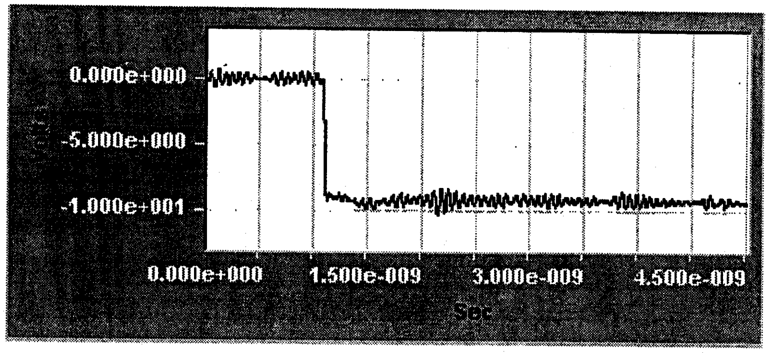

Figure 7a. $x_m(t) = ifft\{Y_m(j\omega)/H(j\omega)\}$ exhibits noise of deconvolution.

Next, filter $H(j\omega)$ by $C_\lambda(j\omega)$ and compute: $\hat{x}_m(t) = ifft\{Y_m(j\omega)/\hat{H}(j\omega)\}$

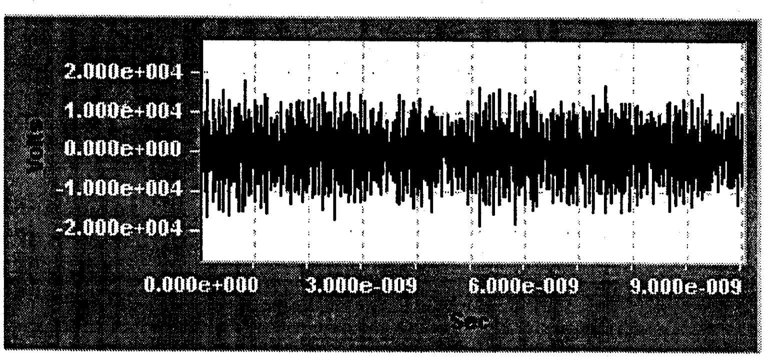

Figure 7b. $\hat{x}_m(t)$, ($H(j\omega)$ filtered by $C_\lambda(j\omega)$).

As was noted previously, poles and zeroes of $\hat{H}(j\omega)$ are, generally, the same as those of $Y_m(j\omega)$, and noise of deconvolution will arise from this operation. Needed is a transition-band filter, $R_{\lambda,\beta}(j\omega)$, developed in a manner similar to $C_\lambda(j\omega)$, to reduce this noise:

$$R_{\lambda,\beta}(j\omega) = \frac{\left|\hat{H}(j\omega)\right|^2}{\left|\hat{H}(j\omega)\right|^2 + \beta}$$

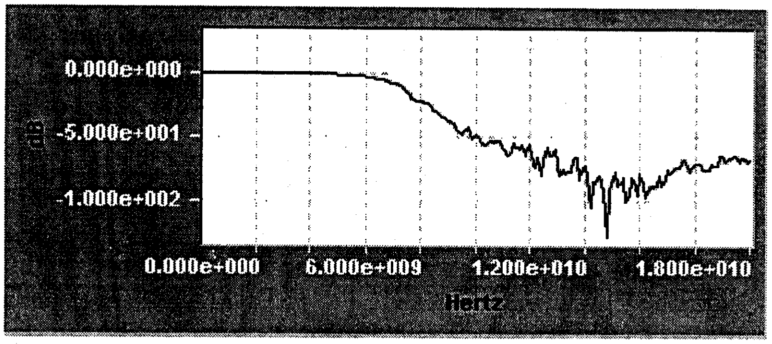

Figure 8. dB magnitude plot of $R_{\lambda,\beta}(j\omega)$. (detail)

Now, an estimate of $\hat{X}_m(j\omega)$ that possesses reduced noise of deconvolution can be expressed as:

$$\hat{\hat{X}}_m(j\omega) = \frac{Y_m(j\omega)}{\hat{H}(j\omega)} \cdot R_{\lambda,\beta}(j\omega)$$

Substituting $H(j\omega) \cdot C(j\omega)$ for $\hat{H}(j\omega)$ in the expression above yields:

$$\hat{\hat{X}}_m(j\omega) = \frac{Y_m(j\omega)}{H(j\omega) \cdot C_\lambda(j\omega)} \cdot R_{\lambda,\beta}(j\omega)$$

Re-writing: $\hat{\hat{X}}_m(j\omega) = \dfrac{Y_m(j\omega)}{H(j\omega)} \cdot \dfrac{R_{\lambda,\beta}(j\omega)}{C_\lambda(j\omega)}$ <u>or</u> $\hat{\hat{H}}(j\omega) = H(j\omega) \cdot \dfrac{R_{\lambda,\beta}(j\omega)}{C_\lambda(j\omega)}$

The quantity, $R_{\lambda,\beta}(j\omega)/C_\lambda(j\omega)$, is the cascaded transition-band filter. It resembles a low-pass filter exhibiting improved fidelity in the pass-band, gentle roll-off above the corner frequency, and greatly reduced noise in the stop-band.

NB: Use of the cascaded transition band filter is preferred over an FIR (or similar) low-pass filter since spectral content above the passband is not entirely eliminated thereby allowing transient features of the compensated waveform to be better estimated. Spectral components within the transition band of an FIR low-pass filtered transfer function have no correspondence with those of the physical measurement system. Hence, compensating data with an FIR low-pass filtered transfer function, while eliminating contributions from noise of deconvolution within the stop band, is likely to introduce emphasized spectral content from noise present within the transition band.

$\hat{\hat{h}}(t)$, the fully compensated estimated measurement system impulse response, is shown overplotted with the partially compensated estimate, $\hat{h}(t)$.

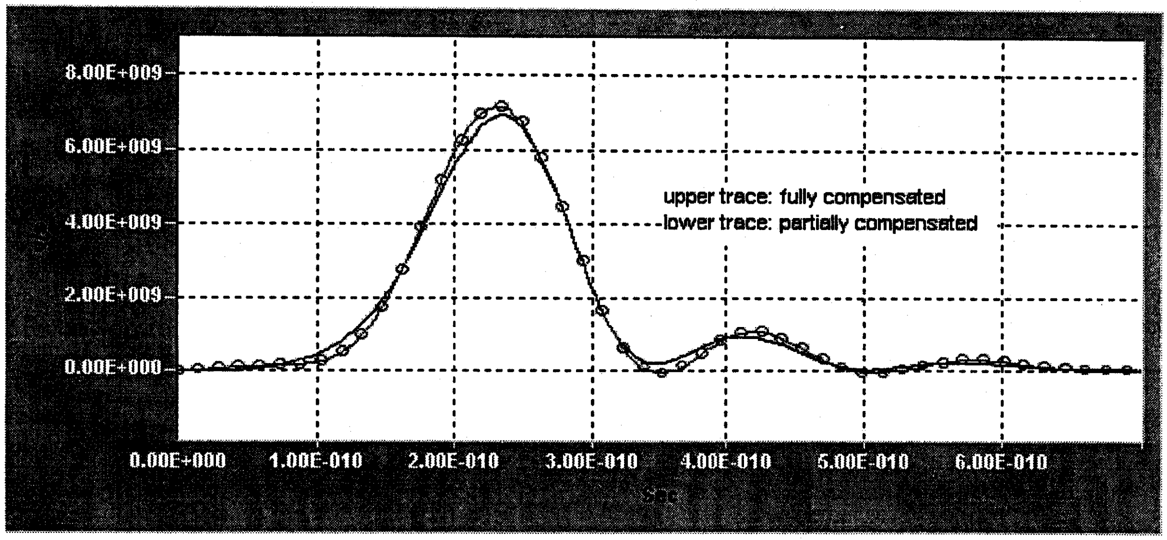

Figure 9. $\hat{\hat{h}}(t) = ifft\{H(j\omega)\cdot R_{\lambda,\beta}(j\omega)/C_{\lambda}(j\omega)\}$. (detail)

Optimization of Filter Parameters

Values of λ and β must now be found that will give a satisfactory estimates for $\hat{\hat{h}}(t) = ifft\{\hat{\hat{H}}(j\omega)\}$ or $\hat{\hat{x}}_m(t) = ifft\{\hat{\hat{X}}_m(j\omega)\}$. The system transfer function, $H(j\omega)$, filtered with $C_{\lambda}(j\omega)$ and $R_{\lambda,\beta}(j\omega)$, will provide a good compromise between noise and fidelity [17, 18] when estimating the measurement system's impulse response or compensating waveform data. These characteristics are easily determined by the overplotting and inspection of $C_{\lambda}(j\omega)$ and $R_{\lambda,\beta}(j\omega)$.

In the steps that follow, it is necessary to normalize the components of the transition-band filter to the dc terms of $R_{\lambda,\beta}(j\omega)$ and $C_{\lambda}(j\omega)$. This is done to prevent the addition of any dc offset when using the filter.

Set $\lambda = 0$.

1. Compute, normalize, and plot $20 * \log_{10}|R_{\lambda,\beta}(j\omega)|$. Vary β so that major perturbations in the transition band are better than -30 dB.

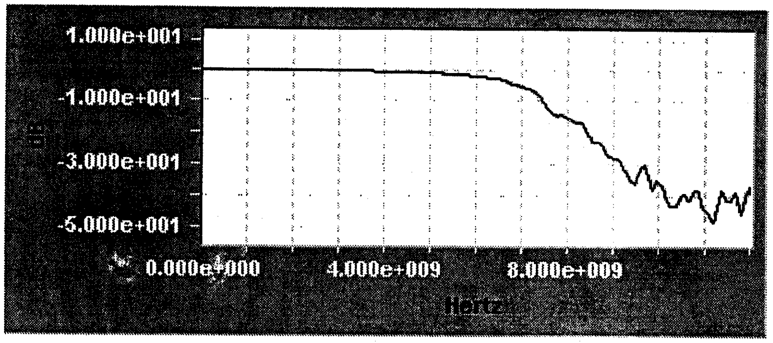

Figure 10. dB magnitude plot of $R_{\lambda,\beta}(j\omega)$, $\lambda = 0$. (detail)

2. Increase the value of λ and, using the value of β determined in the previous step, compute, normalize, and plot $C_\lambda(j\omega)$ with $R_{\lambda,\beta}(j\omega)$.

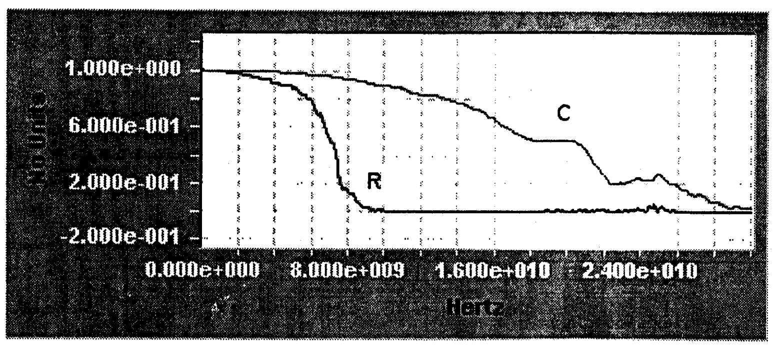

Figure 11. $C_\lambda(j\omega)$ and $R_{\lambda,\beta}(j\omega)$, using trial value of λ. (detail, overplotted)

3. Repeat step 2, as required, increasing the value of λ until there is good agreement throughout the passband when $C_\lambda(j\omega)$ and $R_{\lambda,\beta}(j\omega)$ are overplotted.

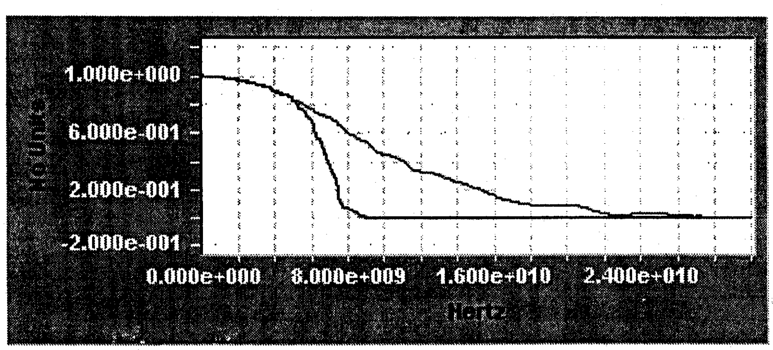

Figure 12. $C_\lambda(j\omega)$ and $R_{\lambda,\beta}(j\omega)$. (detail, overplotted)

On completion of the previous steps, values for λ and β are sufficient for a satisfactory cascaded transition-band filter. These values are not especially critical. Figure 12 illustrates good agreement of $C_\lambda(j\omega)$ and $R_{\lambda,\beta}(j\omega)$ over the passband (lowpass).

Compute and plot $20 * \log_{10}\left|R_{\lambda,\beta}(j\omega)/C_\lambda(j\omega)\right|$ using the values of β and λ determined in steps 1 and 3, respectively. Note the flatness within the passband and the extension of the corner frequency.

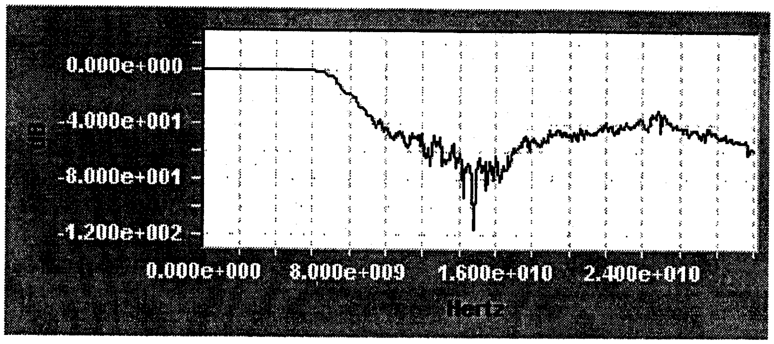

Figure 13. dB magnitude plot of $R_{\lambda,\beta}(j\omega)/C_\lambda(j\omega)$. (detail)

Finally, compute

$$F_{\lambda,\beta}(j\omega) = H(j\omega) \cdot \frac{C_\lambda(j\omega)}{R_{\lambda,\beta}(j\omega)}$$

$F_{\lambda,\beta}(j\omega)$ should be used instead of $H(j\omega)$ when compensating data acquired with the characterized measurement system.

Here, the transfer function, filtered by the cascaded transition-band filter, is overplotted with the unfiltered transfer function.

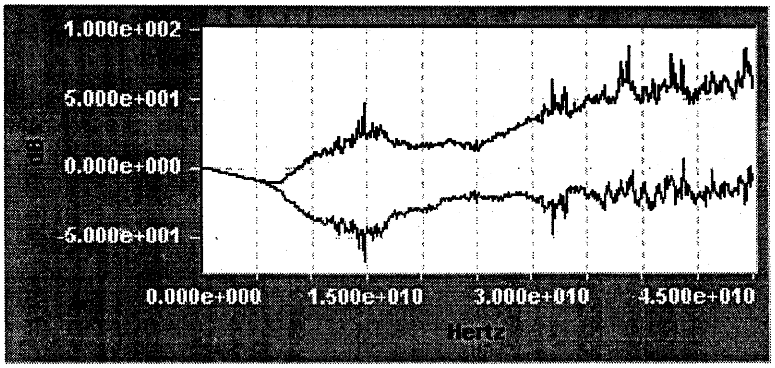

Figure 14. dB magnitude plots of $F_{\lambda,\beta}(j\omega)$ and $H(j\omega)$. (detail, overplotted)

Preparation of Data for Waveform Compensation

A single-shot waveform, $y_m(t)$, with a 'discontinuous' end-point, is to be compensated. The waveform is first made duration-limited by the Gans-Nahman method. This satisfies a criterion for FFT processing in that the signal appears to be periodic. Performing this step correctly preserves the dc offset for this waveform.

$y_a(t)$, prepared for conventional FFT processing, is overplotted with the original waveform, $y_m(t)$.

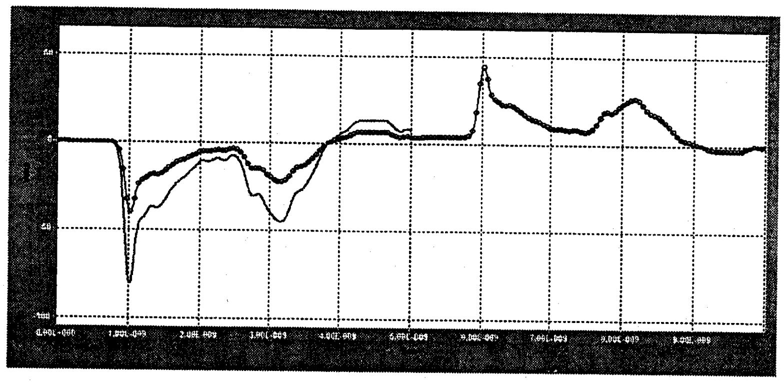

Figure 15. $y_a(t)$, duration-limited, prior to FFT processing. (overplotted with $y_m(t)$)

The correct spectrum of $y_m(t)$ is computed using the complete FFT.

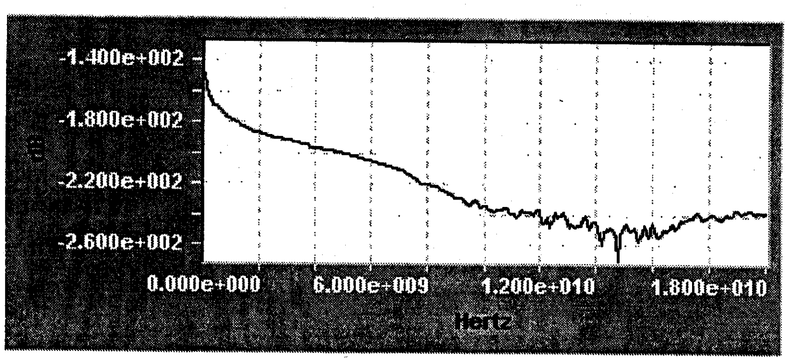

Figure 16. $Y_m(j\omega) = fft\{y_m(t)\}$. (detail)

Measurement System Deconvolution and Signal Compensation

To compensate data from a single shot, the true signal, $x_m(t)$, is acquired as $y_m(t)$. Prepare the waveform, $y_m(t)$, for compensation. Its spectrum, $Y_a(j\omega) = fft\{y_a(t)\}$, is computed using the conventional FFT. This is not the spectrum of $y_m(t)$ [5].

Next, the estimated system transfer function, $F_{\lambda,\beta}(j\omega)$, is deconvolved from the previously obtained spectrum:

$$\hat{\hat{X}}_a(j\omega) = \frac{Y_a(j\omega)}{F_{\lambda,\beta}(j\omega)}$$

$\hat{\hat{X}}_a(j\omega)$ is not the spectrum of $\hat{\hat{x}}_m(t)$. The correct estimated spectrum, $\hat{\hat{X}}_m(j\omega)$, is obtained using the complete FFT once the waveform data compensation process is completed.

The nearly correct compensated input signal is obtained with the inverse FFT:

$$\hat{\hat{x}}_a(t) = ifft\{\hat{\hat{X}}_a(j\omega)\}$$

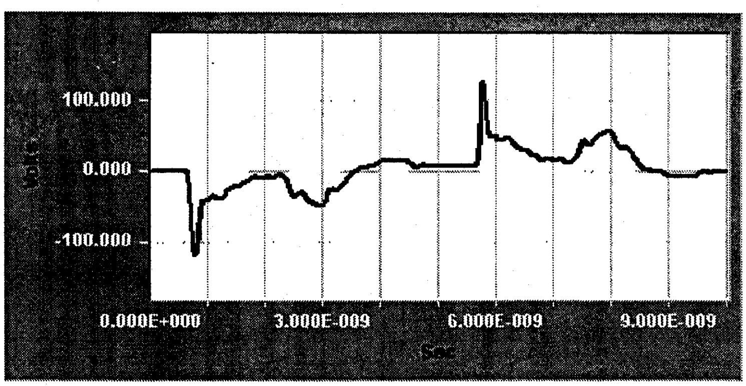

Figure 17. $\hat{\hat{x}}_a(t)$ prior to final processing.

Values of $\hat{\hat{x}}_a(t)$ beyond the final time of $y_m(t)$ are invalid and must be discarded.

The result, $\hat{\hat{x}}_m(t)$, is a good estimate of the true signal being measured. Upon deconvolving $F_{\lambda,\beta}(j\omega)$ from $Y_a(j\omega)$ and performing the described post-processing, the compensated waveform, $\hat{\hat{x}}_m(t)$, is as if it had been acquired using a measurement system with much improved bandwidth.

Figure 18 shows the improvement in risetime and amplitude of the compensated waveform versus the uncompensated waveform. Noise of deconvolution is dramatically reduced. For this example, $\lambda = 1.4 \cdot 10^{-20}$ and $\beta = 2.1 \cdot 10^{-2}$, the –3dB bandwidth is improved to 6.65GHz.

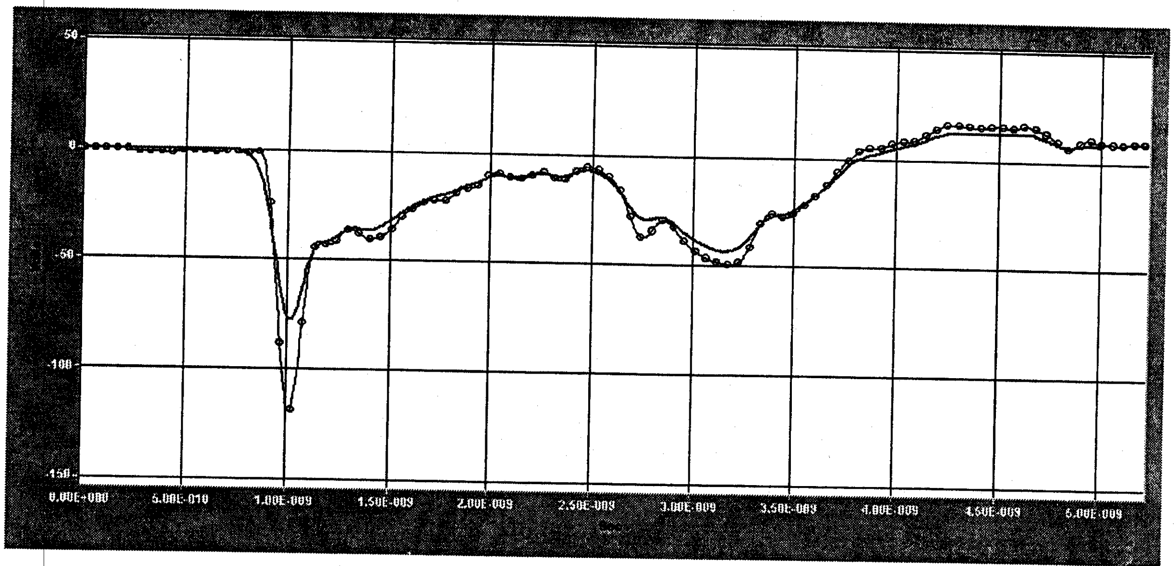

Figure 18. $y_m(t)$ and $\hat{\hat{x}}_m(t)$. (overplotted)

Software Usage

'CFFT', computes the complete FFT components of the system transfer function or the correct spectrum of any waveform.

'DECONVOLVE', computes the measurement system's transfer function. Ensure that the output filename is 'Hf.dat'.

'CONVOLVE' is a companion program for 'DECONVOLVE'. It is not used in the procedures described in this report, but is included for completeness.

'MAKEFF', computes the individual transition-band filters, 'Cf.dat' and 'Rf.dat', and the filtered system transfer function, 'Ff.dat'. The user supplies values of Lambda and Beta.

'NUCOMP', compensates raw data waveforms using the system transfer function filtered with the cascaded transition-band filter. The user supplies a filename for the waveform to be compensated. The same filename is pre-pended with 'comp_' and is assigned to the compensated waveform data.

Conclusion

The prudent implementation of the characterization procedure and subsequent employment of a cascaded transition-band filter for waveform compensation has reduced errors due to noise of deconvolution to a very acceptable level.

References

1. Patterson, P.E., et al., *15-GHz Bandwidth Enhancement of Transient Digitizers in a Portable Data Acquisition System*, SAND93-2280, Sandia National Laboratories, Feb., 1994.

2. Nicolson, A.M., *Forming the fast Fourier Transform of a Step Response in Time-Domain Metrology*, Electronics Letters, v. 9(14):317-318, July 12, 1973.

3. Gans, W.L., and Nahman, N.S., *Continuous and Discrete Fourier Transforms of Steplike Waveforms*, IEEE Trans. on Instrumentation & Measurement, v. IM-31(2):97-101, June, 1982.

4. Waldmeyer, J., *Fast Fourier Transform for Step-Like Functions: The Synthesis of Three Apparently Different Methods*, IEEE Trans. on Instrumentation & Measurement, v. IM-29(1):36-39, March, 1980.

5. Shaarawi, A.M, and Riad, S.M., *Computing the Complete FFT of a Step-Like Waveform*, IEEE Trans. on Instrumentation & Measurement, v. IM-35(1):91-92, Mar., 1986.

6. Riad, S.M., *The Deconvolution Problem: An Overview*, Proc. of IEEE, v. 74(1):82-85, Jan., 1986.

7. Miller, E.K. (Ed.), **Time-Domain Measurements in Electromagnetics**, Van Nostrand Reinhold, 1986; Ch. 3, written by S.M. Riad.

8. Brigham, E.O., **The Fast Fourier Transform and its Applications**, Prentice-Hall, 1988; Sec. 14.5 on FFT deconvolution.

9. Nicolson, A.M., *Broad-Band Microwave Transmission Characteristics from a Single Measurement of the Transient Response*, IEEE Trans. on Instrumentation & Measurement, v. IM-17(4):395-402, Dec., 1968.

10. Cronson, H.M., and Mitchell, P.G., *Time-Domain Measurements of Microwave Components*, IEEE Trans. on Instrumentation & Measurement, v. IM-22(4):320-325, Dec., 1973.

11. Andrews, J.R., *Automatic Network Measurements in the Time Domain*, Proc. of IEEE, v. 66(4):414-423, Apr., 1978.

12. Andrews, J.R., *10 ps Risetime Transmission & TDR Measurements Using the PSPL 4015B Generator & HP 54121A Oscilloscope*, Picosecond Pulse Labs Application Note AN-5a (July, 1992).

13. Bennia, A., and Nahman, N.S., *Deconvolution of Causal Pulse and Transient Data*, IEEE Trans. on Instrumentation & Measurement, v. IM-39(6):933-939, Dec., 1990.

14. Lawton, R.A., et al., *Pulse and Time-Domain Measurements*, Proc. of IEEE, v. 74(1):77-81, Jan., 1986.

15. Andrews, J.R., *Pulse Measurements in the Picosecond Domain*, Picosecond Pulse Labs Application Note AN-3a (Oct., 1989).

16. Andrews, J.R., TDR, *Step Response & S-Parameter Measurements in the Time Domain*, Picosecond Pulse Labs Application Note AN-4 (May, 1989).

17. Parruck, B., and Riad, S.M., *An Optimization Criterion for Iterative Deconvolution*, IEEE Trans. on Instrumentation & Measurement, v. IM-32(1):137-140, Mar., 1983.

18. Bennia, A., and Riad, S.M., *An Optimization Technique for Iterative Frequency-Domain Deconvolution*, IEEE Trans. on Instrumentation & Measurement, v. 39(2):358-362, Apr., 1990.

Appendix A

```
      program  cfft

! Compute 'complete' FFT of a waveform

! Paull E. Patterson
! High-Power Electromagnetics Department
! Sandia National Laboratories
! Albuquerque, New Mexico  87185-1153
! 505.284.4197
! pepatte@sandia.gov

! May 8, 2000
!------
      real tDUM,yDUM

      real(8), allocatable :: ya(:),yb(:)
      real(8), allocatable :: d(:),e(:),f(:), g(:)
      real(8), allocatable :: tim(:),c(:)

      real(8) dt
      real(8) RDeltf
      real(8) Deltf
      real(8) freq

      real(8) m,b,dy,dx

      integer i,j,k,len,dlen

      character YT*64,YF*64

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      write(*,'(A\)') ' time-domain Filename: '
      read(*,'(A)') YT

      write(*,'(A\)') ' frequency-domain Filename: '
      read(*,'(A)') YF

      open(unit=8, file=YT, status='unknown')
      open(unit=7, file=YF, status='unknown')

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! pre-read y(t) to initialize counter 'len'

      len = 0

  111 read(8,*,end=99) tDUM,yDUM
      len = len+1
      goto 111

   99 rewind(8)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! read y(t) from file

      dlen = 2*len

      allocate(tim(dlen),c(len),d(len),e(dlen))

      do i = 1, len
        read (8,*) tim(i),c(i)
      end do

      close(8)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      tim = tim - tim(1)

      dt = tim(2) - tim(1)

      dy = c(len) - c(1)
      dx = tim(len) + dt

      m  = dy/dx

      RDeltf = real(dlen,8) * dt
      Deltf = 1.0d0/Rdeltf

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! perform Nicolson de-ramping on y(t)

      do i = 1, len
        d(i) = c(i) - m*tim(i)
      end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! prepare for FFT

      allocate(f(dlen))

      f = 0.0d0

      j = 1
      do i = 1, len
        f(j) = d(i)/real(dlen,8)
        j=j+2
      end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
perform FFT on deramped y(t)

      call fftsub(f,len,-1)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! perform Gans on y(t)

    do i = 1,len
      e(i) = c(i)
    end do

! invert, offset, and append
    do i = 1,len
      e(len+i) = e(len) + (e(1) - e(i))
    end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! prepare for FFT

    allocate(g(2*dlen))

    g = 0.0d0

    j = 1
    do i = 1, dlen
      g(j) = e(i)/real(dlen,8)
      j=j+2
    end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! perform FFT on inverted and appended y(t)

    call fftsub(g,dlen,-1)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! scale transform

    g = g/2.0d0

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! interleave odd and even harmonics

    allocate(ya(dlen),yb(dlen))

    j = 1
    k = 3

    do i = 1,dlen,2

! Nicolson: 3   5   7...
      ya(j) = f(i)
      yb(j) = f(i+1)

! Gans:   2   4   6   8...
      ya(j+1) = g(k)
      yb(j+1) = g(k+1)

      j = j+2
      k = k+4

    end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Gans: ensure correct DC value

    ya(1) = g(1)
    yb(1) = g(2)
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! write Y(f) to file

      freq = 0.0d0

      do i = 1, dlen
        write(7,100) freq,ya(i)*RDeltf,yb(i)*RDeltf
        freq = freq + Deltf
      end do

      close(7)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      deallocate(f)
      deallocate(g)
      deallocate(ya,yb)
      deallocate(c,d,e,tim)

  100 format(2x, ES23.15E3, 2x, ES23.15E3, 2x, ES23.15E3)

      stop
      end
```

Appendix B

```
      program  convolve

!  Convolve two spectra

!------
!  Paull E. Patterson
!  High-Power Electromagnetics Department
!  Sandia National Laboratories
!  Albuquerque, New Mexico  87185-1153
!  505.284.4197
!  pepatte@sandia.gov

!  June 22, 2000
!------

      real(8), allocatable :: fH(:),Ha(:),Hb(:)
      real(8), allocatable :: fX(:),Xa(:),Xb(:)
      real(8), allocatable :: Ya(:),Yb(:)

      complex(8) Ytmp

      real fDUM,aDUM,bDUM

      integer  i,len

      character HF*64,XF*64,YF*64

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      write(*,'(A\)') ' spectrum1 Filename: '
      read(*,'(A)') HF

      write(*,'(A\)') ' spectrum2 Filename: '
      read(*,'(A)') XF

      write(*,'(A\)') ' resultant spectrum Filename: '
      read(*,'(A)') YF

      open(6, file=HF, status='unknown')
      open(7, file=XF, status='unknown')
      open(8, file=YF, status='unknown')

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!  pre-read H(f) to initialize counter 'len'

      len = 0

  111 read(6,*,end=99) fDUM,aDUM,bDUM
      len = len+1
      goto 111

   99 rewind(6)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  read H(f) and X(f)

      allocate (fH(len),Ha(len),Hb(len))
      allocate (fX(len),Xa(len),Xb(len))
```

```
      do i = 1, len
        read(6,*) fH(i),Ha(i),Hb(i)
        read(7,*) fX(i),Xa(i),Xb(i)
      end do

      close(6)
      close(7)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! compute Y(f)=H(f)*X(f)

      allocate (Ya(len),Yb(len))

      do i = 1, len

        Ytmp = dcmplx(Ha(i),Hb(i))*dcmplx(Xa(i),Xb(i))

        Ya(i) = real(Ytmp,8)
        Yb(i) = dimag(Ytmp)

      end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! write Y(f) to file

      do i = 1, len
        write(8,100) fH(i),Ya(i),Yb(i)
      end do

      close(8)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      deallocate(fH,Ha,Hb)
      deallocate(fX,Xa,Xb)
      deallocate(Ya,Yb)

100   format(2x, ES22.14E3, 2x, ES22.14E3, 2x, ES22.14E3)

      stop
      end
```

Appendix C

```
      program  deconvolve

! Deconvolve one spectrum from another

!------
! Paull E. Patterson
! High-Power Electromagnetics Department
! Sandia National Laboratories
! Albuquerque, New Mexico  87185-1153
! 505.284.4197
! pepatte@sandia.gov

! June 22, 2000
!------

      real(8), allocatable :: fY(:),Ya(:),Yb(:)
      real(8), allocatable :: fX(:),Xa(:),Xb(:)
      real(8), allocatable :: Ha(:),Hb(:)

      complex(8) Htmp

      real fDUM,aDUM,bDUM

      integer  i,len

      character XF*64,YF*64,HF*64

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      write(*,'(A\)') ' spectrum1 Filename: '
      read(*,'(A)') YF

      write(*,'(A\)') ' spectrum2 Filename: '
      read(*,'(A)') XF

      write(*,'(A\)') ' resultant spectrum Filename: '
      read(*,'(A)') HF

      open(6, file=YF, status='unknown')
      open(7, file=XF, status='unknown')
      open(8, file=HF, status='unknown')

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! pre-read Y(f) to initialize counter 'len'

      len = 0

  111 read(6,*,end=99) fDUM,aDUM,bDUM
      len = len+1
      goto 111

   99 rewind(6)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! read Y(f) and X(f)

      allocate (fY(len),Ya(len),Yb(len))
      allocate (fX(len),Xa(len),Xb(len))
```

```
      do i = 1, len
        read(6,*) fY(i),Ya(i),Yb(i)
        read(7,*) fX(i),Xa(i),Xb(i)
      end do

      close(6)
      close(7)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! compute H(f)=Y(f)/X(f)

      allocate (Ha(len),Hb(len))

      do i = 1, len

        if ((Xa(i).EQ.0.0D0).AND.(Xb(i).EQ.0.0D0)) Xa(i) = 1.0d0

        Htmp = dcmplx(Ya(i),Yb(i))/dcmplx(Xa(i),Xb(i))

        Ha(i) = real(Htmp,8)
        Hb(i) = dimag(Htmp)

      end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! write H(f) to file

      do i = 1, len
        write(8,100) fX(i),Ha(i),Hb(i)
      end do

      close(8)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      deallocate(fY,Ya,Yb)
      deallocate(fX,Xa,Xb)
      deallocate(Ha,Hb)

100   format(2x, ES23.15E3, 2x, ES23.15E3, 2x, ES23.15E3)

      stop
      end
```

Appendix D

```
      program makeff
!
! Compute transition-band filter C(f) provided parameter  Lambda
! Compute transition-band filter R(f) provided parameters Lambda and Beta
!
! Compute filtered transfer function; F(f)=H(f)*C(f)/R(f)

! Paull E. Patterson
! High-Power Electromagnetics Department
! Sandia National Laboratories
! Albuquerque, New Mexico  87185-1153
! 505.284.4197
! pepatte@sandia.gov

! August 16, 2000
!-------

      real(8), allocatable :: fX(:),Xa(:),Xb(:)
      real(8), allocatable :: fH(:),Ha(:),Hb(:)
      real(8), allocatable :: Cf(:),Rf(:)

      complex(8), allocatable :: Xf(:)
      complex(8), allocatable :: Hf(:)

      complex(8) Ftmp

      real(8) mX2
      real(8) mH2

      real(8) Lambda
      real(8) Beta

      integer i,len

      real fDUM, aDUM, bDUM

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      open( 6, file='XF.dat', status='unknown')
      open( 7, file='HF.dat', status='unknown')

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! pre-read X(f) to initialize counter 'len'

      len = 0

  111 read(6,*,end=99) fDUM,aDUM,bDUM
      len = len+1
      goto 111

   99 rewind(6)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! read H(f) and X(f)

      allocate (fX(len),Xa(len),Xb(len))
      allocate (fH(len),Ha(len),Hb(len))
```

```
      do i=1,len
        read(6,*) fX(i),Xa(i),Xb(i)
        read(7,*) fH(i),Ha(i),Hb(i)
      end do

      close(6)
      close(7)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! initialize arrays

      allocate (Xf(len),Hf(len))

      do i=1,len
        Xf(i)=dcmplx(Xa(i),Xb(i))
        Hf(i)=dcmplx(Ha(i),Hb(i))
      end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      write(*,'(A\)') ' Enter value of Lambda: '
      read *, Lambda

      write(*,'(A\)') ' Enter value of Beta: '
      read *, Beta

      allocate (Cf(len),Rf(len))

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! compute C(f)

      do i=1,len
        mX2 = (cdabs(Xf(i)))**2.0d0
        Cf(i) = mX2/(Lambda + mX2)
      end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! compute R(f)

      do i=1,len
        mH2 = (cdabs(Hf(i)*Cf(i)))**2.0d0
        Rf(i) = mH2/(Beta + mH2)
      end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      open(8, file='Cf.dat', status='unknown')
      open(9, file='Rf.dat', status='unknown')

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! write normalized C(f) and R(f)

      do i=1,len
        write(8,100) fH(i),Cf(i)/Cf(1)
        write(9,100) fH(i),Rf(i)/Rf(1)
      end do

      close(8)
      close(9)
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
! compute C(f)/R(f)

      do i = 1,len
        Cf(i) = Cf(i) / Rf(i)
      end do

! normalize

      do i = 1,len
        Cf(i) = Cf(i) / Cf(1)
      end do

! compute F(f) = H(f)*C(f)/R(f)

      do i = 1,len

        Ftmp = dcmplx(Ha(i),Hb(i)) * Cf(i)

        Ha(i) = real(Ftmp,8)
        Hb(i) = dimag(Ftmp)

      end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! write F(f) to file

      open(10, file='Ff.dat', status='unknown')

      do i = 1, len
        write(10,200) fH(i),Ha(i),Hb(i)
      end do

      close(10)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

  100 format(2x, ES23.15E3, 2x, ES23.15E3)
  200 format(2x, ES23.15E3, 2x, ES23.15E3, 2x, ES23.15E3)

      deallocate(fX,Xa,Xb)
      deallocate(fH,Ha,Hb)
      deallocate(Xf,Hf)
      deallocate(Cf,Rf)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      stop
      end
```

Appendix E

```
      program  nucomp

! Deconvolution of filtered system transfer function from Measured Response
!    Uses the Gans-Nahman method to preserve dc baseline offset
!    (described in SAND2000-1310)

!------
! Paull E. Patterson
! High-Power Electromagnetics Department
! Sandia National Laboratories
! Albuquerque, New Mexico  87185-1153
! 505.284.4197
! pepatte@sandia.gov

! June 22, 2000
!------

      real fDUM,aDUM,bDUM

      real(8), allocatable :: f(:),ya(:),yb(:)
      real(8), allocatable :: fH(:),Ha(:),Hb(:)
      real(8), allocatable :: xa(:),xb(:)
      real(8), allocatable :: time1(:),c1(:)

      real(8) dt,dy,dx,m
      real(8) ytmp

      integer i,j,len,dlen

      complex(8) xtmp

      character YT*64,XT*64,CCC*5

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      write(*,'(A\)') ' raw data Filename: '
      read(*,'(A)') YT

      open(7, file='Ff.dat', status='unknown')
      open(8, file=YT, status='unknown')

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Pre-read F(f) to determine value of index 'dlen'

      dlen = 0

  111 read(7,*,end=99) fDUM,aDUM,bDUM
        dlen = dlen+1
      go to 111

   99 rewind(7)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      allocate (fH(dlen),Ha(dlen),Hb(dlen))
      allocate (time1(dlen),c1(dlen))
      allocate (f(2*dlen),ya(dlen),yb(dlen))
      allocate (xa(dlen),xb(dlen))

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Read F(f) and y(t)

      do i = 1, dlen
        read(7,*) fH(i),Ha(i),Hb(i)
      end do

      len = dlen/2

      do i = 1, len
        read (8,*) time1(i),c1(i)
      end do

      close(7)
      close(8)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Prepare waveform for compensation
!   by making waveform duration-limited
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! perform Gans-Nahman method on y(t)

! invert, offset, and append

      do i = 1,len
        c1(len+i) = c1(len) + (c1(1) - c1(i))
      end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Perform FFT on duration-limited y(t)

      f = 0.0d0

      j = 1
      do i = 1, dlen
        f(j) = c1(i)/real(dlen,8)
        j=j+2
      end do

      call fftsub(f,dlen,-1)

      j = 1
      do i = 1,2*dlen,2
        ya(j) = f(i)
        yb(j) = f(i+1)
        j = j+1
      end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Perform deconvolution: X(f) = Y(f)/F(f)

      i = dlen/2 + 1

      if ((Ha(i).EQ.0.0D0).AND.(Hb(i).EQ.0.0d0)) Ha(i) = 1.0d0

      j = 1
      do i = 1, dlen
        xtmp = dcmplx(ya(i),yb(i))/dcmplx(Ha(i),Hb(i))

        f(j)   = real(xtmp,8)
        f(j+1) = dimag(xtmp)
```

```
        j=j+2
      end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Perform IFFT on X(f)

      call fftsub(f,dlen,1)

      j = 1
      do i = 1, 2*dlen, 2
        xa(j) = f(i)
        j = j+1
      end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Write estimated x(t) to file

      CCC = 'comp_'

      open(9, file=CCC//YT, status='unknown')

      do i = 1, len
        write(9,100) time1(i),xa(i)
      end do

      close(9)

  100 format(2x, ES23.15E3, 2x, ES23.15E3)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      deallocate (fH,Ha,Hb)
      deallocate (time1,c1)
      deallocate (f,ya,yb)
      deallocate (xa,xb)

      stop
      end
```

Appendix F

```
      subroutine fftsub(f,nn,isig)

! Subroutine adapted from FFT in "Numerical Recipes"
!
! provided by:
!
! Douglas J. Riley
! Electromagnetics and Plasma Physics Analysis Department
! Sandia National Laboratories
! Albuquerque, New Mexico  87185-1152
! 505.845.3469
! djriley@sandia.gov
!
! February 17, 1999

!-----
!
!     isig = -1    forward transform
!     isig =  1    reverse transform
!
!     nn  number of complex elements
!
!     f   array to be transformed
!             contains nn complex elements
!             format:  f(1) real part
!                      f(2) complex part

      integer i,j,n,nn,m,istep,mmax,iqt,isig
      real(8) tempr,tempi,theta,wr,wi,xpi,f,x1,x2

      dimension f(1)

      n=2*nn
      j=1

      DO 5 i=1,n,2

        IF (i-j) 1,2,2

 1      tempr=f(j)
        tempi=f(j+1)

        f(j)=f(i)
        f(j+1)=f(i+1)

        f(i)=tempr
        f(i+1)=tempi

 2      m=n/2

 3      IF (j-m) 5,5,4

 4      j=j-m
        m=m/2

        IF (m-2) 5,3,3
 5      j=j+m

      mmax=2
```

```
6 IF (mmax-n) 7,9,9

7 istep=2*mmax

  xpi=4.0d0*atan(1.0d0)

  DO 8 m=1,mmax,2

    iqt=isig*(m-1)

    x1=real(iqt,8)
    x2=real(mmax,8)

    theta=xpi*x1/x2

    wr=cos(theta)
    wi=sin(theta)

    DO 8 i=m,n,istep

      j=i+mmax

      tempr = wr*f(j) - wi*f(j+1)
      tempi = wi*f(j) + wr*f(j+1)

      f(j)   = f(i)   - tempr
      f(j+1) = f(i+1) - tempi

      f(i)   = f(i)   + tempr
8     f(i+1) = f(i+1) + tempi

  mmax=istep
  goto 6

9 return
  end
```