Ricker-compliant and pseudo-unitary decon

Jon Claerbout

ABSTRACT

Ricker compliant deconvolution spikes the center lobe of the Ricker wavelet. It enables deconvolution to preserve and enhance seismogram polarities. It works by tapering at small lags the anti-symmetric part of the time-domain representation of the log spectrum. A byproduct of this decon is a pseudo-unitary (very clean) debubble filter where bubbles are lifted off the data while onset waveforms (usually Ricker) are untouched.

INTRODUCTION

Seismogram polarity becomes more apparent in seismic data when deconvolution removes the correct source wavelet. A problem is that predictive deconvolution does not do a good job defining that wavelet. Predictive decon assumes minimum phase while marine seismology typically exhibits the Ricker wavelet, a wavelet which is both theoretically and practically, marginally or doubtfully minimum phase. This problem is resolved in this paper by shifting the time origin to the middle main lobe of the Ricker wavelet and simultaneously estimating both this now noncausal shot waveform and its inverse. A byproduct of this approach is a debubble process giving results of outstanding clarity.

Textbook deconvolution produces a white output, but it cannot be true that we wish energy at the Nyquist frequency while the sampling rate is quite arbitrary. Recently I discovered that parameterizing the logarithm of the spectrum in the time domain lays the problem out in a manner where practical issues sort themselves out along the "quefrency" axis. I came to this approach because my inverse theory (with Guitton and with Fu) lacked a certain regularization this provides. Results here are excellent, and computed in $N \log_2 N$ time. I regard them as a final analytical stage before invoking iterative inverse theory.

BASICS OF LAG-LOG SPACE

Let $F(\omega)$ be a filter in frequency domain. Let $U(\omega) = \ln F(\omega)$ so the filter is $F(\omega) = e^{U(\omega)}$. With the definition $Z = e^{i\omega\Delta t}$ Fourier transforms become polynomials (Z-transforms). Thus $U(\omega)$ relates to the time function u_τ by a Fourier sum $U = \sum_{\tau=0}^{2048} u_\tau Z^\tau$. (The 2048 simply means "the whole trace".)/ The u_τ values will be our

parameterization of the filter. Historically, the τ axis is called the "quefrency" axis though we sometimes refer to it as the "lag-log" axis.

The property of exponentials that $e^{A+B+C} = e^A\ e^B\ e^C$ has an interesting meaning when we exponentiate a Z-transform $\exp(A+B+C) = \exp(\sum_{\tau=1}^{2048} u_\tau Z^\tau)$. The Z-transform sum may be split up into small lags, medium lags, and large lags. This decomposes a filter (or waveform) into a sequence of three filters, each with its own meaning, for example, in common marine seismology:

$$e^{A+B+C} = e^A\ e^B\ e^C \tag{1}$$

$$e^{\sum_{\tau=1}^{2048} u_\tau Z^\tau} = e^{\sum_1^2}\ e^{\sum_3^{15}}\ e^{\sum_{16}^{2048}} \tag{2}$$

$$\text{(wavelet)} = \text{(continuity)(Ricker)(bubble)} \tag{3}$$

Equation (2) defines the boundaries of the three regions abruptly although in practice we blend them smoothly. Changing the sign of $(A+B+C)$ changes a filter to its inverse. Both are parameterized by the same A, B, and C.

We may specify u_τ from prior knowledge, or from knowledge gained from various kinds of data averaging, or from some mixture of the two. Commonly, we begin using Kolmogoroff spectral factorization (next section) to give all the u_τ. We may design a filter e^{A+B+C} by overriding Kolmogoroff with $A = 0$ and $B = 0$. To see what happens, consider the filter $e^C = 1 + C + C^2/2! + \cdots \approx 1 + C$. Examine its leading coefficients. They are $(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, u_{16}, u_{17}, u_{18}, \cdots)$. Figure 1 shows the application of such a filter. This operation on the data is called "debubbling". The 14 interval gap on 4ms data is 60ms, a number halfway between the "end of the ghosts" and the "onset of the bubbles". This result may be described as "textbook quality" (meaning it is the best I have ever produced).

Setting all of A, B, and C to zero makes a unitary filter that is simply an impulse. Setting to zero any of the three, or any combination, or suppressing any combination tends to make a filter more unitary, hence the appellation "pseudo-unitary". Debubbling in this manner seems to leave first arrivals untouched, further justifying the term. Before we go on to attack the middle-lag terms we review the starting point, Kolmogoroff spectral factorization.

Kolmogoroff spectral factorization

When a time function such as c_t vanishes at all negative time lags it is said to be causal. Its Z transform is $C(Z) = c_0 + c_1 Z + c_2 Z^2 + c_3 Z^3 + \cdots$. Observe that $C(Z)^2$ is also causal because it has no negative powers of Z, alternately, because the convolution of a causal with a causal is causal. Likewise e^C is causal because it is a sum of causals.

$$e^C = 1 + C + C^2/2! + C^3/3! + \cdots \tag{4}$$

Happily, this infinite series always converges because of the strong influence of the denominator factorials. The time-domain coefficients for e^C could be computed the

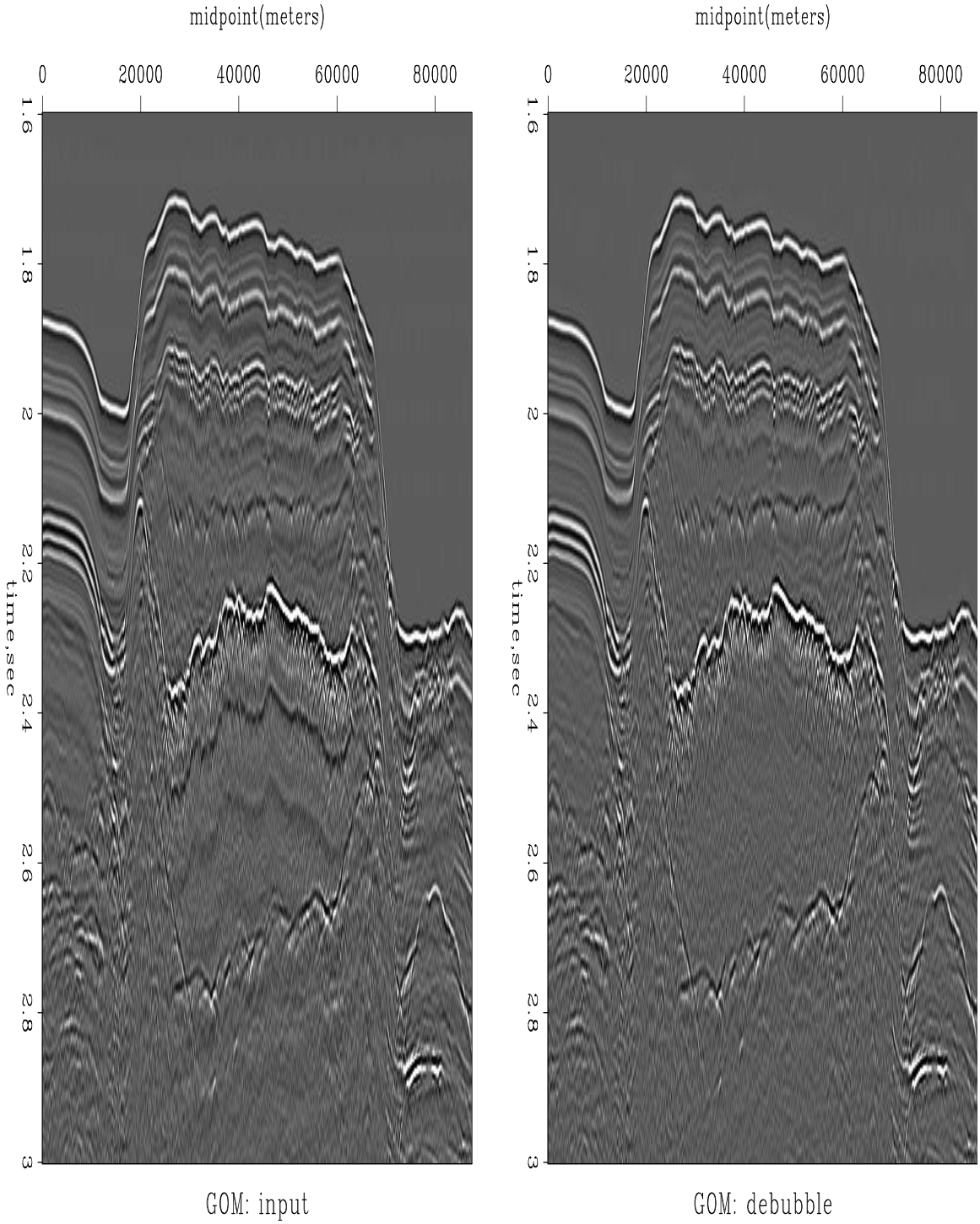

Figure 1: Gulf of Mexico data before and after pseudo-unitary debubble. This process preserves the wave onset while it lifts off the bubbles. Here the effect of the process is visible nearly everywhere after 2.4 sec, but also visible around 1.85 sec. On blinking displays it is easy to see bubble removed nearly everywhere. [**ER**]

hard way, putting polynomials into power series, or e^C may be computed by Fourier transforms. To do so, we would evaluate $C(Z = e^{i\omega})$ for many real ω, and then invoke an inverse Fourier transform program to uncover the time-domain coefficients.

Let $r = r(\omega)$, $\phi = \phi(\omega)$, and $Z^\tau = e^{i\omega\tau}$. Let us investigate the consequence of exponentiating a causal filter.

$$|r|e^{i\phi} \;=\; e^{\ln|r|+i\phi} \;=\; e^{\sum_\tau c_\tau Z^\tau} \;=\; \exp\left(\sum_\tau c_\tau Z^\tau\right) \tag{5}$$

Notice a pair of filters, both causal and inverse to each other.

$$|r|e^{i\phi} \;=\; e^{+\sum_\tau c_\tau Z^\tau} \tag{6}$$

$$|r|^{-1}e^{-i\phi} \;=\; e^{-\sum_\tau c_\tau Z^\tau} \tag{7}$$

A filter from any such pair is said to be "minimum phase". Many filters are not minimum phase because they have no causal inverse. For example the delay filter Z. It's inverse, Z^{-1} is not causal. Such filters do not relate to a causal complex logarithm. If they have a logarithm, it must be non causal.

Given a spectrum $r(\omega)$ we can construct a minimum-phase filter with that spectrum. Since $r(\omega)$ is a real even function of ω, the same may be said of its logarithm. Let the inverse Fourier transform of $\ln|r(\omega)|$ be u_τ, where u_τ is a real even function of time. Imagine a real odd function of time v_τ.

$$|r|e^{i\phi} \;=\; e^{\ln|r|+i\phi} \;=\; e^{\sum_\tau (u_\tau+v_\tau)Z^\tau} \tag{8}$$

The phase $\phi(\omega)$ transforms to v_τ. We can assert causality by choosing v_τ so that $u_\tau + v_\tau = 0$ for all negative τ. This defines v_τ at negative τ. Since v_τ is odd, we also know its values at positive lags. This creates a causal exponent which creates a causal minimum-phase filter with the specified spectrum. The code does this by multiplying u_τ by a step function of height 2, the doubling accounting for the zeroing of half the axis. This computation is called Kolmogoroff spectral factoring. The word "factoring" enters because in applications one begins with an energy spectrum $|r|^2$ and factors it into an $re^{i\phi}$ times its conjugate (time reverse).

It is a exercise for the student to show that a complex-valued time function has a positive spectrum that is non-symmetrical in frequency and may also be factored.

Ricker compliant decon

Start with the u_τ resulting from a Kolmogoroff factorization. (Optionally you might weight down portions making it more unitary.) Split it into even and odd parts, $u_\tau^{\text{odd}} = (u_\tau - u_{-\tau})/2$ and $u_\tau^{\text{even}} = (u_\tau + u_{-\tau})/2$ whose sum is u_τ. The even part Fourier transforms to the logarithm of the amplitude spectrum. The odd part Fourier transforms to the phase spectrum. Here we monkey with the phase while not changing

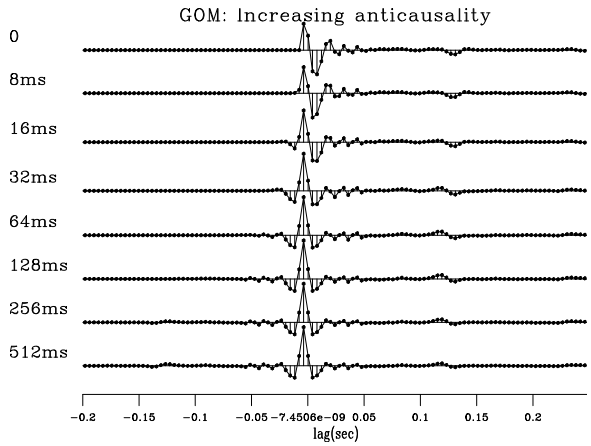

Figure 2: Gulf of Mexico $\Delta t =$ 4ms: Increasing the anticausality in Ricker compliant decon. Too much causes half the bubble to appear before the shot. Numerical values of anticausality refer to the lag τ at which the sine-squared weight upon u_τ^{anti} reaches unity where weighting ceases. [**ER**]

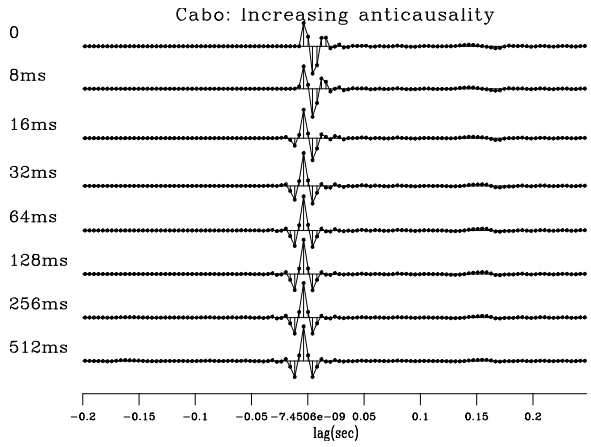

Figure 3: Gulf of California $\Delta t =$ 4ms: Results similar to Figure 2. Notice in both cases the strictly causal wavelet looks as much like a doublet as like a Ricker wavelet. [**ER**]

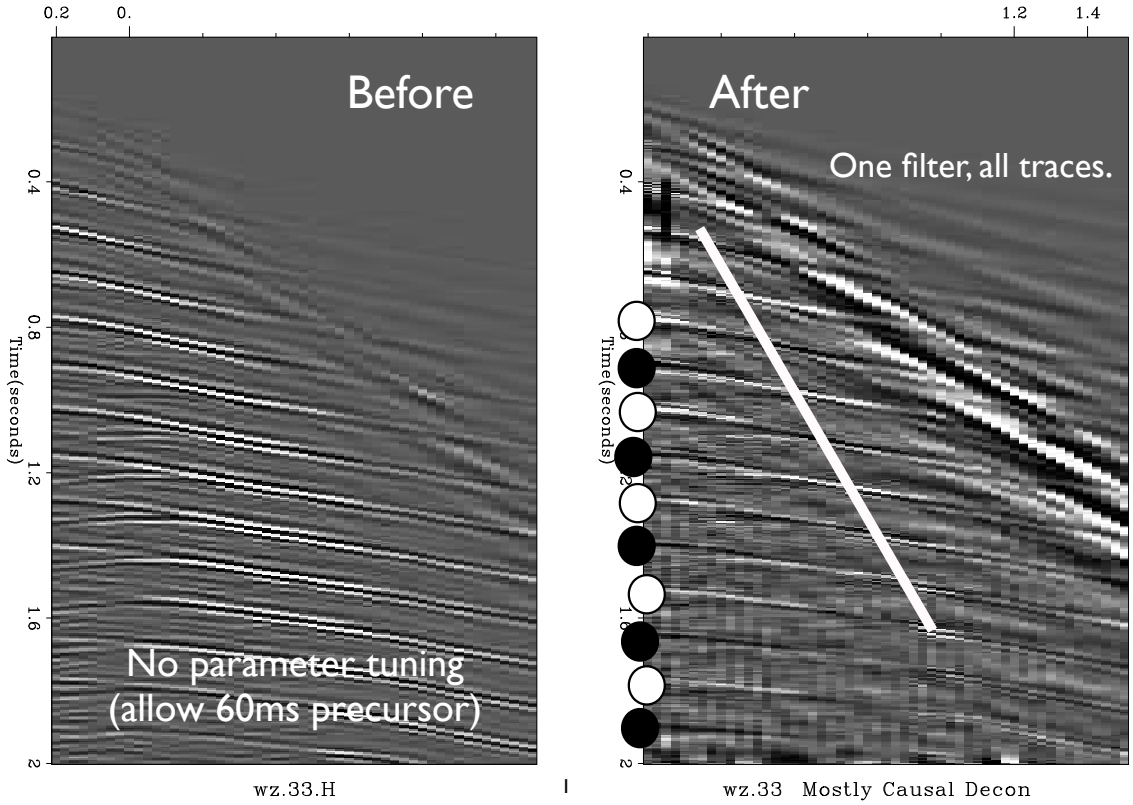

Figure 4: The spectrum of each trace in a shot gather was averaged and used to produce a single Ricker compliant deconvolution filter used on all traces. Because of changes with angle, consistently alternating polarity of multiples is best observed along a diagonal line. [**NR**]

the amplitude. We simply taper u_τ^{odd} towards zero for small lags. Figure 2 examines the consequences of various numerical choices of "small". As we increase the anti-causality, the time function of the wavelet e^U increases in symmetry near $t = 0$. The example in Figure 2 is based on the average spectra of 1001 traces from a line in the Gulf of Mexico.

Our favored choice of 60ms is larger than the Ricker width, about 20ms, but not as large as the bubble delay, about 150ms, an easy choice. The result of dividing such a wavelet from data is shown in Figures 4 and 5.

Band limiting

The very short lags in the lag-log space raise deep issues about the meaning of functions. Just the fact that a continuous function of time may be sampled densely on the time axis implies a large amplitude ratio between the central frequency band and the Nyquist frequency. Do we really want the deconvolution to produce a white output all the way to the Nyquist? Probably not. Consistent with our quefrency approach these issues show up at very small values of τ. They are there because the prediction error filter for a nearly continuous function should be something like $(1, -1)$ which vanishes at the Nyquist frequency, has a log spectrum will tend to blowing up (negatively).

An appealing aspect of deconvolution is that it defines in principle "the" wavelet which connects you to some canonical trace. So a second deconvolution of your data should take you nowhere because you are already "there". What other answer can there be but a white trace? This appears to invite some kind of an optimization formulation—beyond the scope of this study.

A pragmatic answer not meeting any philosophical goals is an adjustable parameter for the code. This parameter should repress the tendency of deconvolution to whiten near the Nyquist frequency. The parameter may be thought of as specifying our expected time resolution in milliseconds. The default `tresol=.01` says we are thinking of time resolution about 10ms. Specifying `tresol=0` would say we are envisioning great detail on the time axis, so we'd get a white output. This parameter is is implemented in the same manner as the pseudo-unitary decon. The only difference is the parameter τ_{tresol} defaults to 10ms instead of 60ms. It is installed in the code defaulting at $\tau_{\text{tresol}} = 10$ms. An example is shown in Figure 6.

WHAT IS LEFT TO DO?

This paper starts from a given spectrum, here an average of many traces. In reality, the spectrum might vary from trace to trace. The spectrum will vary from one offset to the next. These basic aspects are not addressed here.

Inverse theory incorporates other real-world complications beyond the scope of the present study. Our ongoing work with it shows it reveals polarity even more

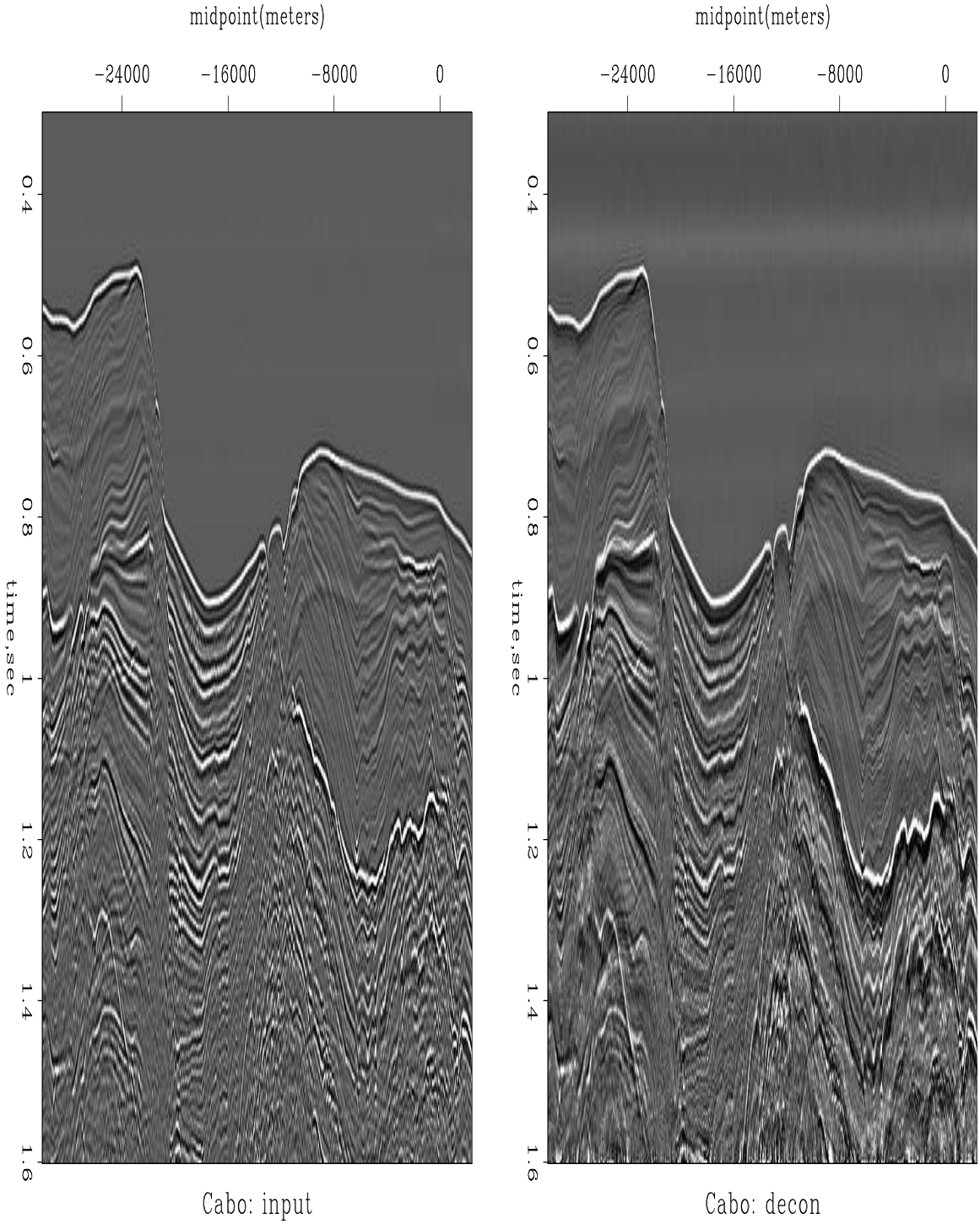

Figure 5: Gulf of California data and its decon. This deconvolution, called Ricker-compliant decon is converting the water bottom Ricker wavelet to a positive pulse. Imperfect debubbling may be explained by (1) guns changing along the line, and/or (2) the depth 550ms being comparable to the bubble delay 180ms (which calls for inverse theory.) [**ER**]

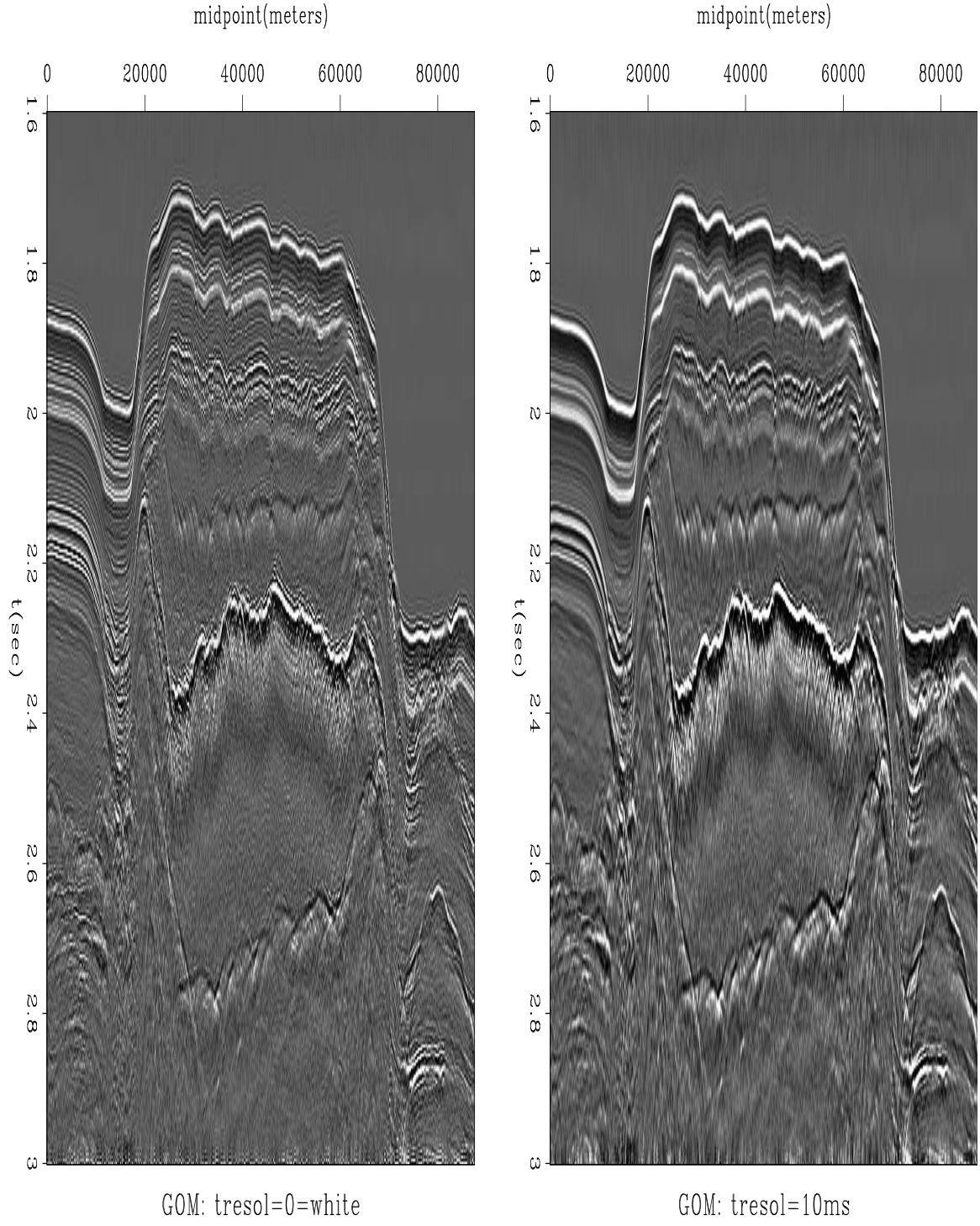

Figure 6: Getting a smoother output by suppressing lags less than about 10ms. [**ER**]

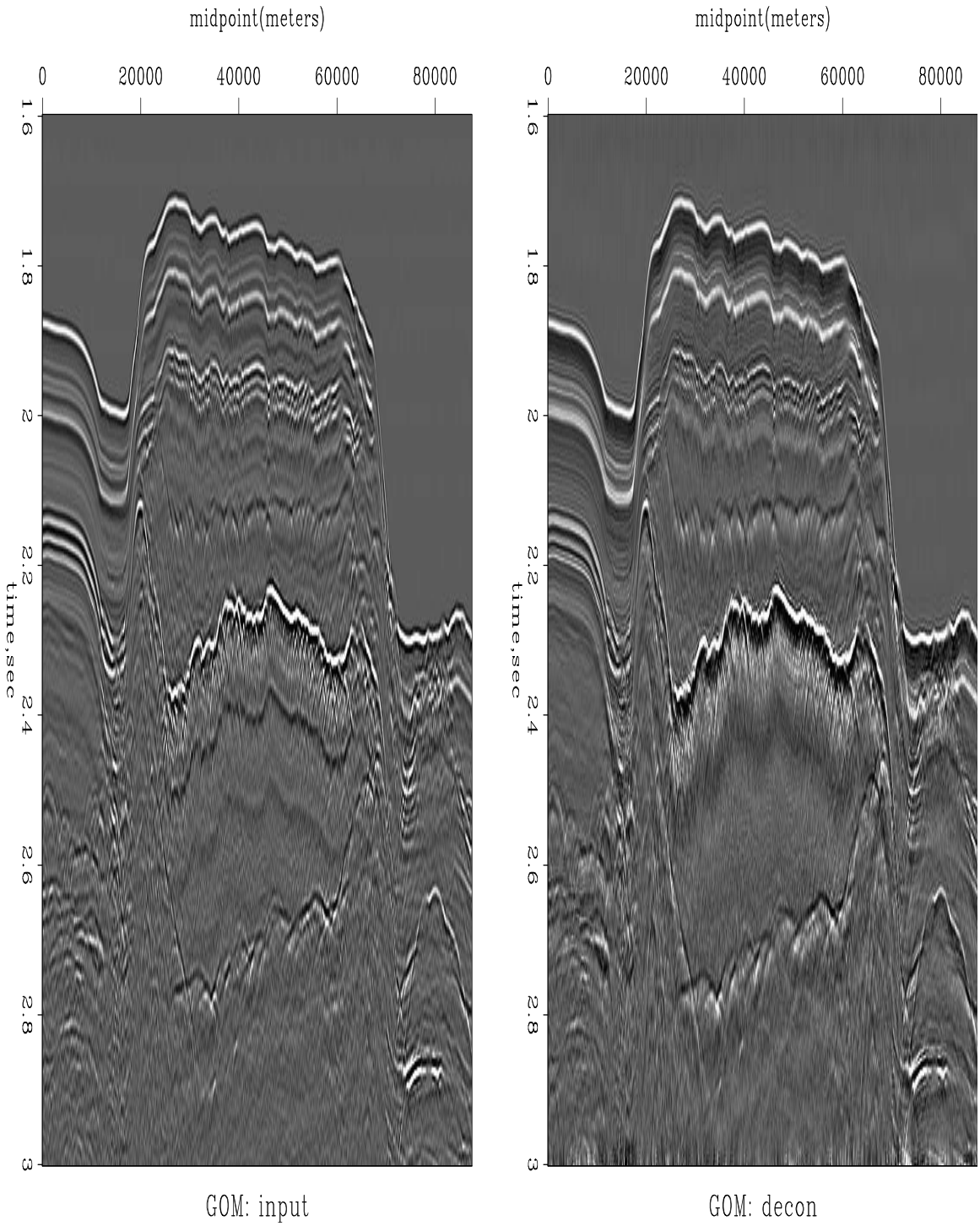

Figure 7: Gulf of Mexico data before and after decon. The Ricker wavelet should have been turned into a positive pulse. Unfortunately, very low frequencies have appeared making this less apparent. [**ER**]

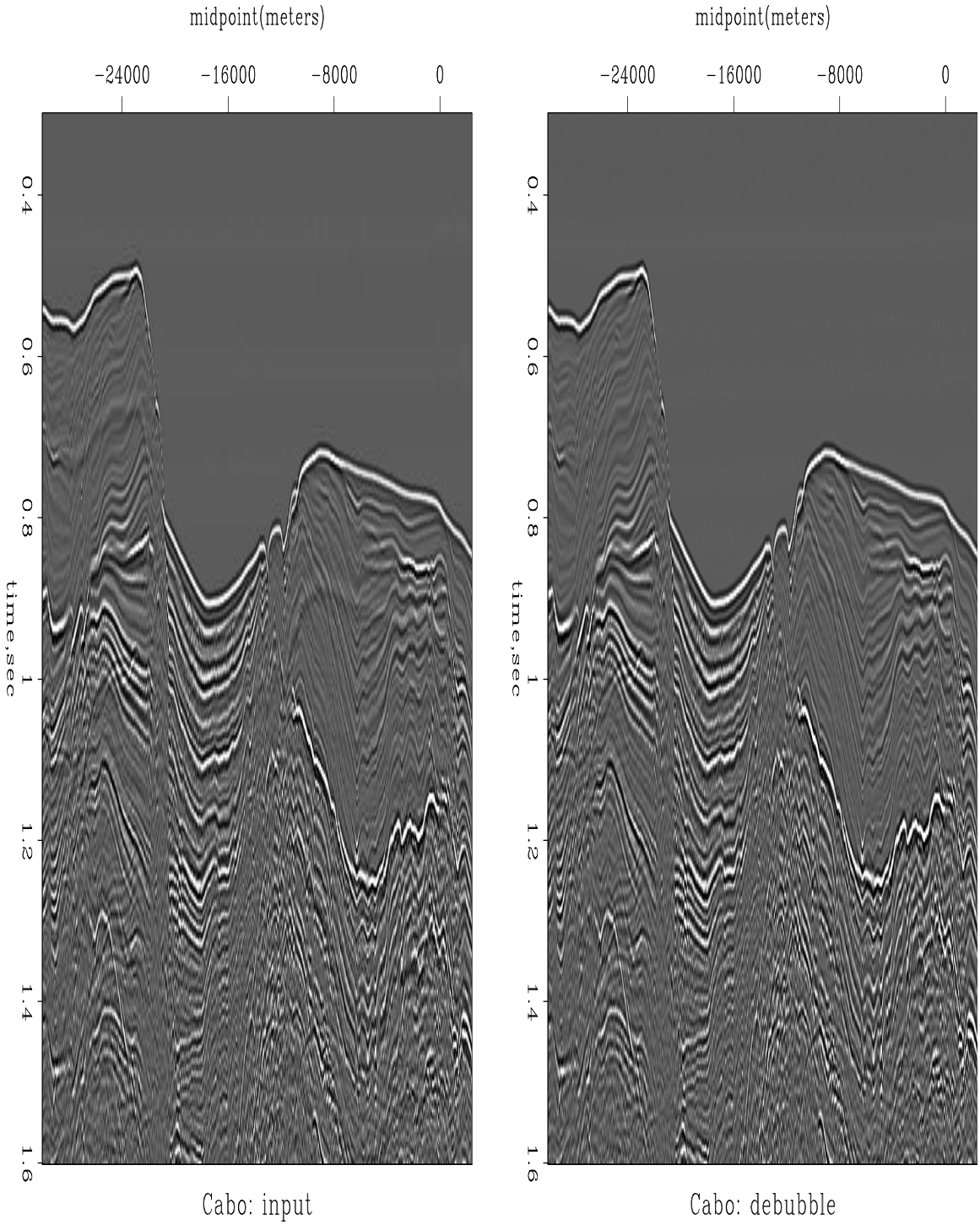

Figure 8: Bubble about 180ms after water bottom is obviously removed. Blink display shows bubble in many locations being lifted off the data. [**ER**]

clearly. Perhaps so because it uses ℓ_1-like statistics. Also it correctly handles gain and filtering as non commuting operators. The method of this paper, however, wins when it comes to clear and simple parameter choice.

SELF DOCUMENTATION

```
# <in.H Rickdecon debubl=.06 ricker=.06 tresol=.01 shot=shot.H >out.H
# Inputs: d1=dt,debubl,ricker,tresol  must have identical units of time (seconds)
#         debubl=.06      =0 means no debubble
#         ricker=.06      =0 means no Ricker compliance
#         tresol=.01      time resolution, =0 for whitening decon
# Outputs: out.H is deconvolved data
# Outputs: shot(n)  centered at (n+1)/2.   Note n=2^^N >n1
#
# subroutine rickdecon( shot,n, data,n1,n2, d1,debubl,ricker,tresol)
```

APPENDIX

Subroutine `ftu` below is an ancient FT program from my book FGDP with conventional scaling consistent with Z-transforms. Data length must be a power of two. Subroutine `kolmogoroff` below was taken from my book PVI, converted from energy spectra to amplitude spectra.

While looking at the code you might notice that you could easily taper large lags to shorten your filter response. This might be useful should you want to crop off downgoing multiples from your source waveform. It could also be helpful when you have insufficient data to be estimating long source waveforms.

Beginning from the spectrum `cx(n)`, below is the code that makes the filter, also `cx(n)`.

```
subroutine kolmogoroff( n, cx, dt, debubl,ricker,tresol)  # Spectral factorization.
real                           dt, debubl,ricker,tresol, weight, tau
                                            # Adapted from PVI, converted energy-->amplitude
integer i,              n                   # input:  cx = amplitude spectrum
complex cx(n)                               # output: cx = FT of min phase wavelet

do i= 1, n
        cx(i) = clog( cx(i) )

call ftu( -1., n, cx)
do i= 2, n/2 {                      # Make it causal changing only the odd part.
        cx(i)     = cx(i) * 2.
        cx(n-i+2) = 0.
        }

## BEGIN weighting u
tau  = dt;  i=2;   while ( tau < debubl) {
        weight = sin( .5 * 3.14159265 * tau/(debubl+1.e-20))**2
        cx(i)     = cx(i)     * weight
        cx(n-i+2) = cx(n-i+2) * weight
```

```
        i = i+1;    tau = tau + dt
        }
tau  = dt;  i=2;   while ( tau < tresol) {
        weight = sin( .5 * 3.14159265 * tau/(tresol+1.e-20))**2
        cx(i)     = cx(i)     * weight
        cx(n-i+2) = cx(n-i+2) * weight
        i = i+1;    tau = tau + dt
        }
tau  = dt;  i=2;   while ( tau < ricker) {
        weight = sin( .5 * 3.14159265 * tau/(ricker+1.e-20))**2
        eve = (cx(i) + cx(n-i+2))/2.
        odd = (cx(i) - cx(n-i+2))/2.
        odd = odd  * weight
        cx(i)     = eve + odd
        cx(n-i+2) = eve - odd
        i = i+1;    tau = tau + dt
        }
# END weighting u

call ftu( +1., n, cx)
do i= 1, n
        cx(i) = cexp( cx(i))
return; end


subroutine ftu( signi, nx, cx )
#   complex Fourier transform with traditional scaling (FGDP)
#
#               1         nx          signi*2*pi*i*(j-1)*(k-1)/nx
#   cx(k)  =  -------- * sum cx(j) * e
#              scale     j=1             for k=1,2,...,nx=2**integer
#
#  scale=1 for forward transform signi=1, otherwise scale=1/nx
integer nx, i, j, k, m, istep, pad2
real    signi, arg
complex cx(nx), cmplx, cw, cdel, ct
if( nx != pad2(nx) )    call erexit('ftu: nx not a power of 2')
do i= 1, nx
        if( signi<0.)
                cx(i) = cx(i) / nx
j = 1;  k = 1
do i= 1, nx {
        if (i<=j) { ct = cx(j); cx(j) = cx(i); cx(i) = ct }
        m = nx/2
        while (j>m && m>1) { j = j-m; m = m/2 }         # "&&" means .AND.
        j = j+m
        }
repeat {
        istep = 2*k;   cw = 1.;   arg = signi*3.14159265/k
        cdel = cmplx( cos(arg), sin(arg))
        do m= 1, k {
                do i= m, nx, istep
                        { ct=cw*cx(i+k);  cx(i+k)=cx(i)-ct;  cx(i)=cx(i)+ct }
                cw = cw * cdel
```

```
                }
        k = istep
        if(k>=nx) break
        }
return; end
```