```
% JitterDeconV1.m
% Jim Andrews, Version V1 --  8/28/09
% Basic principle:  model averaged jitter as Gaussian (n=2) Low Pass Filter
% Determine noise floor of Vout(f).  Only apply pre-emphasis H(f) Gaussian
% filter up to freq where Vout(f) first reaches noise floor.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% JitterDecon.m program is modification to basic decon program VinDeconV2.m
% for jitter program -- deleted all other selectable decon filters
% VinDeconV2.m Version 2.0, PICOSECOND PULSE LABS, written by J.R.Andrews,
11/16/04
% modified from HdeconV2.m --- original ver, 1.0, 10/29/04, jra
%   ver.1.1, 11/16/04, corrected 2X scale factor error in impulse decon.
%   ver.2.0, 11/16/04, eliminated impulse option. I found that the step
decon
%   method works equally well for impulse type signals also. jra
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Deconvolution & Jitter --- TECHNICAL REFERENCES:
% [1] J.R.Andrews & M.G.Arthur, "SPECTRUM AMPLITUDE --- Definition,
% Generation and Measurement", NBS Tech.Note 699, NBS, Boulder, Colo.USA,
% Oct. 1977, 92 pages.
% [2] W.L.Gans & J.R.Andrews, "Time Domain Automatic Network Analyzer for
% Measurement of RF & Microwave Components", NBS Tech.Note 672, NBS,
% Boulder, Colo.USA, Sept.1975, 165 pages.  See chp. 3 for square wave
% treatment of step-like pulses.  See chp. 9 for jitter
% [3] A.Shaarawi & S.Riad, "Computing the Complete FFT of a Step-Like
% Waveform", IEEE Trans. Inst.& Meas., vol IM-35, no.1, pp.91-92,March 1986
% [4] A.M.Nicolson, "Forming the fast Fourier transform of a step response
% in time-domain metrology", Electron.Lett., vol.9,pp.317-318, July 1973
% [5] W.L.Gans & N.S.Nahman, "Continuous and discrete Fourier tranform of
% step-like waveforms", IEEE Trans. Inst. & Meas., vol.IM-31, pp.97-101,
% June 1982
% [6] J.Waldmeyer, "Fast Fourier transform for step-like functions: The
% synthesis of three apparently different methods", IEEE Trans. Inst. &
% Meas., vol, IM-29, pp.36-39, March 1980
% [7] J.R.Andrews, "Deconvolution of System Impulse Responses & Time Domain
% Waveforms", PSPL AN-18, Oct. 2004
% [8] W.L.Gans, "Calibration and Error Analysis of a Picosecond Pulse
% Waveform Measurement System at NBS", Proc. of IEEE, vol. 74, no. 1,
% pp. 86-90, Jan. 1986
% [9] Reference Data for Engineers, 7th edition, 1985, H.W. Sams & Co.,
% chp. 44 "Probability & Statistics", pp. 44-4 & 44-5
% [10] J.R.Andrews, "Removing Jitter From Picosecond Pulse Measurements"
% PSPL AN-23, Aug. 2009
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp('  ')
```

```
disp('JitterDeconV1.m --- MatLab Program')
disp('Picosecond Pulse Labs, Boulder, Colo, USA, J.R.Andrews, 28 Aug 2009')
disp('Deconvolution of vin(t) from vout(t) & jitter Gaussian impulse
response hj(t)')
disp('Data comes from an oscilloscope as .txt input files')
disp('Data should be voltage values only. One data point per line.')
disp('All headers and time data should be stripped from data file.')
disp('Time window, Tw, should be large enough to include entire transient')
disp('dt should be small enough for Nyquist freq. 1/2dt to be in noise
floor')
disp('Program works well with both steps & impulses')
disp('Program calculates vin(t) = ifft(fft(vout(t))/ Hj(f))')
disp('where Hj(f) = exp[-0.5*(f/sigmaf)**2)')
disp('Program gives option of saving results in .txt format')
disp('  ')
clear
warning off
disp('Please use the keyboard to respond to the following questions.')
drive = input('Data in/out via drive A: or C:?  (1=A: 3=C:)   ');
if drive == 1
    disp('Data in/out will be as .txt files via floppy disc in drive A:')
end
if drive == 3
    disp('Data in/out will be as .txt files in current C: drive directory')
end
disp(' ')
% type = input('Type of signal? 1 = step, 2 = impulse:  ');
type = 1; % eliminated impulse option, v2.0, 11/16/04 no longer necessary
Tw = input('Enter Time Window in NanoSeconds (ns)?  ');
% Note: This works with floppy disc data files written by the HP-54750
% oscilloscope.  Verbose data headers need to be first stripped off. The HP
% o'scope data is organized as one voltage data point per line.
% Tested to also work with data from LeCroy, Tek & Agilent o'scopes 8/09
fname1 = input('Enter Output Waveform Data File Name:   ','s');
dname1 = [fname1, '.txt'];
if drive == 1
    dname1 = ['A:', fname1, '.txt'];
end
disp('NOTE: Baseline Correction is strongly encouraged for enhanced
accuracy')
baseline = input('Do you want to do baseline correction?  (1=yes  0=no)
');
if baseline == 1
    fname2 = input('Enter Output Baseline Data File Name:   ','s');
    dname2 = [fname2, '.txt'];
    if drive == 1
```

```
        dname2 = ['A:', fname2, '.txt'];
    end
end
% if type == 1 % i.e. step  % deleted with v2.0, 11/16/04
%    disp('Note: For steps, h(t) must be for a time window of 2*Tw')
% end
jsd = 0.001; % default jitter std.dev. (sigma) is 0.001ns or 1ps
jsd = input('Enter Jitter Std.Deviation (sigma) in ns  ');
disp('loading data')
vout = load(dname1);
if baseline == 1
    vblout = load(dname2);
    vout = vout - vblout;
end
N = length(vout);  % = # of data points
dt=Tw/N;  %(in ns)
f0=1/Tw;  %(in GHz)
fny=0.5/dt;  %(in GHz)
for i=1:N
    t(i)=i*dt;
    f(i) = (i-1)*f0;
end
tout = t;
disp('Number of data points is')
disp(N)
disp('Delta T time spacing in ns is')
disp(dt)
disp('Nyquest folding frequeny, fny, in GHz is')
disp(fny)
disp('Jitter Std.Dev (sigma) is')
disp(jsd)
disp('  ')
disp('After each graph, press any key to continue')
plot(tout,vout,'b')
grid
xlabel('time in ns')
ylabel('Volts')
title('Output Waveform from Oscilloscope')
pause % pause command allows program to stop running until key is pressed
% use FFTs to calc freq. domain Fourier transforms from time domain signals
% NOTE:  FFT arrays range from i = 1 to N   FFT(1) is DC value, FFT(2) is
% first harmonic at f0, FFT(3) is second harmonic at 2*f0, etc. up to
% FFT(N/2+1) which is the Nyquist Folding Frequency, fny. Beyond N/2+1 are
% the negative frequency components running from -fny back to -f0 at FFT(N)
%
if baseline == 1
```

```
% Determine spectrum of baseline
    Vblout = fft(vblout);
    Vblout = Vblout/N;
% for plotting purposes, discard DC, Nyquist & neg. freqs (i.e. >N/2)
    for i=1:(N/2-1)
        fpbl(i) = i*f0;
        SAdbVblout(i) = 20*log10((2*Tw*1000)*abs(Vblout(i+1)));
    end
end
%  note: program no longer uses type=2 (impulse) 8/27/09, jra
if type == 2  % i.e. impulse waveforms -- use normal processing for FFTs
    % note -- need to divide fft by N to get proper scaling in volts
    Vout = fft(vout);
    Vout = Vout/N;
    disp('Lowest frequency, fo, and freq. spacing, df, in GHz is')
    disp(f0)
end
%%%%%%%%%%%%%%%%%%%
if type == 1  % i.e. step-like waveform -- special processing required
%   Create new waveform, vsr, by subtracting Nicolson [4] linear ramp from
%   step-like pulse waveform.
%
    for i=1:N
        vrampout(i) = vout(1) + (i-1)*(vout(N)-vout(1))/(N-1);
        vsrout(i) = vout(i) - vrampout(i);
    end
%  Convert step pulse of N pts. to a Gans' [5] square wave, vsq, of 2N pts.
    vsqout = vout;
    for i= 1:N
        vsqout(i+N) = (vout(1)+vout(N)) - vout(i);
    end
    Nsq=2*N;
    Twsq=2*Tw;
    f0sq=1/Twsq; %(in GHz)
    for i=1:Nsq
        tsq(i)=i*dt;
    end
%   calculate the FFTs
% note: for proper scaling of periodic waveforms, need to divide FFT by N
% to get correct V(f) in units of Volts
    Vsrout = fft(vsrout);
    Vsrout = Vsrout/N;
    Vsqout = fft(vsqout);
    Vsqout = Vsqout/Nsq;
% note: f0 of ramped waveform is 1/Tw.  All harmonics of f0 are present up
% to Nyquist freq, fny = 1/2*dt.  The dc value is not valid.
```

```
% For sq.wave waveform the fundamental is 1/2Tw and only the odd harmonics
% are present up to the Nyquist freq, fny.  The even harmonics of 1/2Tw are
% zero due to symmetry.  The dc value is valid.  The frequency lines of the
% ramped waveform and the square wave are interleaved and are all valid.
%
% Mesh Vsr & Vsq arrays into single array V. See Shaarawi & Riad [3]
    Vout = Vsqout;
    for i=3:2:Nsq-1  % i.e. insert ramp spec. into nulls of sq.spec.
        Vout(i) = Vsrout((i+1)/2);
    end
    N = Nsq;
    t = tsq;
    f0 = f0sq;
    disp('Lowest frequency, fo, and freq. spacing, df, in GHz is')
    disp(f0)
end   % end of special processing for step-like pulses
%
% for plotting purposes, discard DC, Nyquist & neg. freqs (i.e. >N/2)
for i=1:(round(N/2)-1)
    fp(i) = i*f0;
    SAdbVout(i) = 20*log10((2*Tw*1000)*abs(Vout(i+1)));
end
if baseline == 1
    semilogx(fp,SAdbVout,'b',fpbl,SAdbVblout,'g')
    grid
    xlabel('Frequency in GHz')
    ylabel('Spectrum Amplitude in dBuV/MHz')
    title('ffts Vout(blue), BLout(green) vs. Freq.')
    pause
end
if baseline == 0
    semilogx(fp,SAdbVout,'b')
    grid
    xlabel('Frequency in GHz')
    ylabel('Spectrum Amplitude in dBuV/MHz')
    title('fft of Vout vs. Freq.')
    pause
end
% analyze noise floor of vout(t) & Vout(f) for >200GHz
% valid using N/5 for Tw=1ns, N(vin data)=1000
% note: necessary for the upper part of the spectrum to be in the noise
% floor
fnfa = (round(N/5) + 1)*f0;
disp('Noise Floor analyzed from f(nyquist) down to GHz ')
disp(fnfa)
for i= round(N/5) : round(N/2)-1
```

```
    k = i -round(N/5) +1;
SAnf(k) = SAdbVout(i);
end
nfdb = mean(SAnf);
% now find freq where Vout(f) first drops to noise floor mean value
fpn = 1;
while SAdbVout(1:fpn) > nfdb , fpn = fpn+1; end
fpnf = fpn*f0;
disp('Mean Noise Floor in dB is')
disp(nfdb)
disp('Vout(f) reaches Noise Floor at frequency in GHz')
disp(fpnf)
%
% create H(f) for Gaussian Low Pass Jitter Filter
% use FFTs to calc freq. domain Fourier transforms from time domain signals
% NOTE:  FFT arrays range from i = 1 to N   FFT(1) is DC value, FFT(2) is
% first harmonic at f0, FFT(3) is second harmonic at 2*f0, etc. up to
% FFT(N/2+1) which is the Nyquist Folding Frequency, fny. Beyond N/2+1 are
% the negative frequency components running from -fny back to -f0 at FFT(N)
sdjf = 1/(2*pi*jsd);
for i = 1: round(N/2)+1
    f = (i-1)*f0;
    Hf(i) = exp(-0.5*(f/sdjf)^2);
end
% set Jitter Filter neg. freqs. to be conjugate of pos.freqs.
for i=2: round(N/2)
    Hf(N+2-i) = conj(Hf(i));
end
for i=1:(round(N/2)-1)
    Hfmag(i) = abs(Hf(i+1));
end
semilogx(fp,Hfmag)
grid
xlabel('Frequency in GHz')
ylabel('H(f)')
title('Jitter Gaussian Low-Pass Filter, H(f)')
pause
% first approximation for vin(t) is vout(t)
Vin = Vout;
% now apply jitter pre-emphasis, Hf, only up to freq. where Hf drops to
% noise floor
% Vin = Vout ./ Hf;  % note ./ necessary to force single element division
for i=1:(fpnf/f0)+1
    Vin(i) = Vout(i) ./ Hf(i);
end
% now do it also for conjugate freqs
```

```
for i=(N-fpn+1):N
    Vin(i) = Vout(i) ./ Hf(i);
end
for i=1:(N/2-1)
VinSAdB(i) = 20*log10((2*Tw*1000)*abs(Vin(i+1)));
NFline(i) = nfdb;
end
semilogx(fp,VinSAdB,'r',fp,SAdbVout,'b',fp,NFline,'g')
grid
xlabel('Frequency in GHz')
ylabel('Spectrum Amplitude in dBuV/MHz')
title('ffts Vout(blue), deconVin(red) & noise floor(green) vs. Freq.')
pause
dbdiff = VinSAdB - SAdbVout;
semilogx(fp,dbdiff)
grid
xlabel('Frequency in GHz')
ylabel('dB')
title('Vin(f)- Vout(f) Spectrum Amplitude in dBuV/MHz vs. Freq.')
pause
semilogx(fp,VinSAdB)
grid
xlabel('Frequency in GHz')
ylabel('dB')
title('Vin(f) Spectrum Amplitude in dBuV/MHz vs. Freq.')
pause
% calc vin(t)  by invFFT of Vin(f)
vin = ifft(Vin);
vin = N*vin;
vinre = real(vin);
vinim = imag(vin);
% %%%%%%%%%%
if type == 1
    % need to correct vinre.  At this point it is off by a scale factor
    % of 2X and it also has a ramp in it which needs to be removed.
    vinc = vinre/2;
    gi = input('Gain Correction Factor (1=H(dc) or 2=H(f0))?   ');
    Hdc = Hf(gi);
    vinint = vout(1)/Hdc;
    vinfin = vout(N/2);
    voff = vinc(1) - vinint;
    vinc = vinc - voff;
    for i=1:N
        vrampin(i) = vinint + (i-1)*(vinfin-vinint)/(N-1);
    end
    for i=1:N
```

```
        vinc(i) = vinc(i) + vrampin(i);
    end
    clear vin
    for i=1:round(N/2)
        vin(i) = vinc(i);
    end
end
if type == 2
    clear vin
    vin = vinre;
end
plot(tout,vin)
grid
xlabel('time in ns')
ylabel('Volts')
title('Deconvolved Input Signal vin(t)')
pause
end
plot(tout,vin,'r',tout,vout,'b')
grid
xlabel('time in ns')
ylabel('Volts')
title('Deconvolved vin(t) (red) & signal averaged vout(t) (blue)')
pause
reply = input('do you want to save results to disc? (1=yes, 0=no)  ');
if reply == 1
    fname = input('Enter Output Data File Prefix Name:   ','s');
% '\r' or '\n' is delimiter for carriage return (newline), i.e. 'enter' key
% this writes output file with one data point per line
% note: this doesn't appear correct in NotePad,
% but is correct in WordPad or Word
    disp('writing Input Signal, vin(t).txt')
    dname = [fname, 'vin.txt'];
    if drive == 1
        dname = ['A:', fname, 'vin(t).txt'];
    end
    dlmwrite(dname,vin,'\r');
    disp('Data written to disc')
end
disp('end of JitterDeconV1.m program')
warning on;
```