```
% TDVNAv21.m
% Version 2.1  7 September 2009, J.R.Andrews, KH6HTV
% PICOSECOND PULSE LABS, Boulder, Colorado USA
% J.R.Andrews (JRA) &  K.J.Schoen (KJS)
% V1.0, JRA 24 nov 2002 - original programs TDNAS11.m & TDNAS21.m
% V1.1, JRA rev. 12/3/02 added removal of baselines & keyboard plot scaling
% V1.2, JRA 4/12/04 added data output to disc A:
% V1.3, JRA 4/20/04 added data zero padding & minor refinements
% V1.4, KJS 5/18/04 merged TDNAS11.m & TDNAS21.m, V1.3, expanded zero
padding
%
% V2.0, JRA 8/27/04 -- Major rewrite of complete program.  Steamlined data
% handling.  Added selection of data in/out via either A: or C: drives.
% Major modification in treatment of step-like pulse waveforms.
% Incorporated Riad's [3] combo, complete FFT of steps using Nicholson's
% ramp subtraction [4] and Gan's square wave [5].
% Changed plots from connected lines to points to emphasize calculations
% are at discrete frequencies and not continuous functions.  If want to use
% line plots, then eliminate the '.b' term in the plot commands.
% 9/6/04, jra, error checking & minor de-buging
% V2.1 9/7/09, jra -- minor mod. added semilog plots
%
% 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.  Data files
% from other oscilloscopes have not been tested on this program.
%
% 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
% [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, "Time Domain Spectrum Analysis & S-Parameter Vector
```

```
% Network Analysis", PSPL App. Note AN-16, May, 2004
%
disp(' ')
disp('TDVNAv21.m - MatLab Program')
disp('version 2.1, J.R.Andrews, KH6HTV, 7 Sept 2009')
disp('PICOSECOND PULSE LABS')
disp('TIME DOMAIN VECTOR NETWORK ANALYSER -- TDVNA')
disp('Data comes from an oscilloscope as TDR or TDT .txt input files')
disp('Incident test signal, vinc, is either an impulse or step')
disp('Time window should be large enough to include entire transient')
disp('Program uses FFTs to compute S11 or S21 vs. frequency')
disp('S11 = fft(vtdr)/fft(vinc)   S21 = fft(vout)/fft(vinc)')
disp('Program plots insertion or return loss in dB and group delay vs.
frequency')
disp('Program gives option of saving data 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(' ')
meas = input('Type of measurement? 1 = S11, 2 = S21:  ');
type = input('Type of test signal? 1 = step, 2 = impulse:  ');
Tw = input('Enter Time window in nanoseconds (ns):  ');
disp('NOTE: Baseline Correction is strongly encouraged for enhanced
accuracy')
baseline = input('Do you want to do baseline correction?  (1=yes  0=no)
');
disp('Please input oscilloscope data files')
if meas == 1  % i.e. S11
    disp('For TDR -- S11 measurement')
    fname1 = input('Enter short circuit ref. data file name:  ','s');
    dname1 = [fname1, '.txt'];
    if drive == 1
        dname1 = ['A:', fname1, '.txt'];
    end
    fname2 = input('Enter TDR data file name:  ','s');
    dname2 = [fname2, '.txt'];
    if drive == 1
        dname2 = ['A:', fname2, '.txt'];
```

```
    end
    vin = load(dname1);
    vout = load(dname2);
    if baseline == 1
        fname3 = input('Enter 50 ohm ref. baseline data file name:  ','s');
        dname3 = [fname3, '.txt'];
        if drive == 1
            dname3 = ['A:', fname3, '.txt'];
        end
        fname4 = input('Enter TDR data baseline file name:  ','s');
        dname4 = [fname4, '.txt'];
        if drive == 1
            dname4 = ['A:', fname4, '.txt'];
        end
        vblin = load(dname3);
        vblout = load(dname4);
    end
end
if meas == 2  % i.e. S12
    disp('For TDT  --- S21 measurement')
    fname1 = input('Enter Input Pulse data file name:  ','s');
    dname1 = [fname1, '.txt'];
    if drive == 1
        dname1 = ['A:', fname1, '.txt'];
    end
    fname2 = input('Enter Output Pulse data file name:  ','s');
    dname2 = [fname2, '.txt'];
    if drive == 1
        dname2 = ['A:', fname2, '.txt'];
    end
    vin = load(dname1);
    vout = load(dname2);
    if baseline == 1
        fname3 = input('Enter Input baseline data file name:  ','s');
        dname3 = [fname3, '.txt'];
        if drive == 1
            dname3 = ['A:', fname3, '.txt'];
        end
        fname4 = input('Enter Output baseline file name:  ','s');
        dname4 = [fname4, '.txt'];
        if drive == 1
        dname4 = ['A:', fname4, '.txt'];
        end
        vblin = load(dname3);
        vblout = load(dname4);
    end
```

```
end
N = length(vin);  % = # of data points
f0=1/Tw; %  fundamental freq. & freq. spacing in GHz
dt=Tw/N; %  sample spacing, delta t, in ns
for i=1:N
   t(i)=i*dt;
end
% plot input data
if meas == 1 % i.e. S11
   if baseline == 1
      plot(t,vin,'b',t,vblin,'g',t,vout,'r')
      grid
      xlabel('time in ns')
      ylabel('Volts')
      title('Short (blue), 50 ohm (green) & TDR (red) Waveforms from
Oscilloscope Data')
   end
   if baseline == 0
      plot(t,vin,'b',t,vout,'r')
      grid
      xlabel('time in ns')
      ylabel('Volts')
      title('Short (blue) & TDR (red) Waveforms from Oscilloscope Data')
   end
end
if meas == 2 % i.e. S21
   if baseline == 1
      plot(t,vin,'b',t,vout,'g',t,vblin,'c',t,vblout,'m')
      grid
      xlabel('time in ns')
      ylabel('Volts')
      title('Input (blue) & Output (green) plus Baseline Waveforms from
Oscilloscope Data')
   end
   if baseline == 0
      plot(t,vin,'b',t,vout,'r')
      grid
      xlabel('time in ns')
      ylabel('Volts')
      title('Input (blue) & Output (red) Waveforms from Oscilloscope Data')
   end
end
   disp(' ')
   disp('Input data plot displayed - press any key to continue')
   pause
% optional baseline correction -- subtact baseline from test data
```

```
if baseline == 1
    vin = vin -vblin;
    vout = vout -vblout;
end
%
if meas == 1 % i.e. S11, invert vin to correct for -1 rho from short
circuit
    vin = -vin;
end
% optional zero padding
disp(' ')
zeropad = input('Do you want to do "zero padding" to increase time window?
(1=yes  0=no) ');
if zeropad == 1
    disp('Original input array size is')
    disp(N)
    multiple = input('Enter desired multiple of original array size, (power
of 2 reqd.):  ');
    new_N = N*multiple;
    disp('The new array size is')
    disp(new_N)
    disp('...processing...')
    if type == 1   % i.e. step
    % determine mean value of last 10 pts. of in & tdr arrays
        for i=1:10
            zpadin(i) = vin(N+1-i);
            zpadout(i) = vout(N+1-i);
        end
    end
    if type == 2   % i.e. impulse
    % determine mean value of first 10 pts. of in & tdr arrays
        for i=1:10
            zpadin(i) = vin(i);
            zpadout(i) = vout(i);
        end
    end
    zpadVin = mean(zpadin);
    zpadVout = mean(zpadout);
    for i=N+1:new_N
        vin(i) = zpadVin;
        vout(i) = zpadVout;
        t(i) = i*dt;
    end
    N = new_N;
    Tw = N*dt;
    f0 = 1/Tw;
```

```
    plot(t,vin,t,vout)
    grid
    xlabel('time in ns')
    ylabel('Volts')
    title('Incident (blue) & TDR or Output (green) Padded Waveforms')
    disp(' ')
    disp('Zero padded plot displayed - press any key to continue')
    pause
end
if type == 2  % i.e. impulse waveforms -- use normal processing for FFTs
    disp('...processing...')
    Vin = fft(vin);
    Vin = Vin/N;
    Vout = fft(vout);
    Vout = Vout/N;
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.
disp('...processing...')
    for i=1:N
        vrampin(i) = vin(1) + (i-1)*(vin(N)-vin(1))/(N-1);
        vsrin(i) = vin(i) - vrampin(i);
        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.
    vsqin = vin;
    vsqout = vout;
    for i= 1:N
        vsqin(i+N) = (vin(1)+vin(N)) - vin(i);
        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
    Vsrin = fft(vsrin);
    Vsrin = Vsrin/N;
    Vsqin = fft(vsqin);
    Vsqin = Vsqin/Nsq;
```

```
    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]
    Vin = Vsqin;
    Vout = Vsqout;
    for i=3:2:Nsq-1
        Vin(i) = Vsrin((i+1)/2);  % i.e. insert ramp spec. into nulls of
sq.spec.
        Vout(i) = Vsrout((i+1)/2);
    end
    N = Nsq;
    f0 = f0sq;
end   % end of special processing for step-like pulses
%
% use FFTs to calc S11 or S21(complex) = FFT(vout(t)) / FFT(vin(t))
disp('...processing...')
Sx1 = Vout ./ Vin;
%note:  ./ necessary to perform simple division, not matrix division
% now throw away dc, nyquest freq & neg. freqs (i.e. >N/2)
for i=1:(N/2 - 1)
    Hmag(i) = abs(Sx1(i+1));
    Hph(i) = angle(Sx1(i+1));  % phase in radians
    f(i) = i*f0;
end
Sx1db = 20*log10(Hmag);
Sx1ph = unwrap(Hph);
GD = -diff(Sx1ph) / (2*pi*f0);  % GD = dPhase/dFreq
Np = length(Sx1ph);
GD(Np) = GD(Np-1);
% end of calculation of S11 or S21
% plot frequency domain results
disp(' ')
fny = 1/(2*dt);
disp('Fundamental frequency and frequency spacing for plots (in GHz) is')
disp(f0)
disp('Nyquist (highest) frequency (in GHz) is')
disp(fny)
```

```
if meas == 1  % i.e. S11
    DC = 20*log10(abs(Sx1(1)));
    disp('S11(dc) in dB =')
    disp(DC)
    disp(' ')
    S11_xmax = input('Enter the max frequency to be plotted for S11
magnitude (in GHz): ');
    plot(f,Sx1db,'.b')
    grid
    xlabel('Frequency in GHz')
    ylabel('S11(f) in dB')
    title('TDVNA:  S11(f) Return Loss vs. Frequency')
    XLim([0 S11_xmax])
    disp('S11 plot displayed - press any key to continue')
    pause
    semilogx(f,Sx1db,'.b')
    grid
    xlabel('Frequency in GHz')
    ylabel('S11(f) in dB')
    title('TDVNA:  S11(f) Return Loss vs. Frequency')
    XLim([0 S11_xmax])
    pause
    disp(' ')
    GDLF = GD(1);
    disp('Low freq. group delay in ns =')
    disp(GDLF)
    TD = GD(1)/2;
    disp('One way time delay from test port to reflection in ns =')
    disp(TD)
    disp(' ')
    GD_xmax = input('Enter the max frequency to be plotted for S11 group
delay (GHz): ');
    plot(f,GD,'.b')
    grid
    xlabel('Frequency in GHz')
    ylabel('Group Delay in ns')
    title('TDVNA:  S11(f) Group Delay vs. Frequency')
    XLim([0 GD_xmax])
    disp('Group delay plot displayed - press any key to continue')
    pause
    semilogx(f,GD,'.b')
    grid
    xlabel('Frequency in GHz')
    ylabel('Group Delay in ns')
    title('TDVNA:  S11(f) Group Delay vs. Frequency')
    XLim([0 GD_xmax])
```

```
    pause
end
if meas == 2  % i.e. S21
    DC = 20*log10(Sx1(1));
    disp('S21(dc) in dB =')
    disp(DC)
    disp(' ')
    S21_xmax = input('Enter the max frequency to be plotted for S21
magnitude (in GHz): ');
    plot(f,Sx1db,'.b')
    grid
    xlabel('Frequency in GHz')
    ylabel('S21(f) in dB')
    title('TDVNA:  S21(f) Return Loss vs. Frequency')
    XLim([0 S21_xmax])
    disp('S21 plot displayed - press any key to continue')
    pause
    semilogx(f,Sx1db,'.b')
    grid
    xlabel('Frequency in GHz')
    ylabel('S21(f) in dB')
    title('TDVNA:  S21(f) Return Loss vs. Frequency')
    XLim([0 S21_xmax])
    pause
    disp(' ')
    GDLF = GD(1);
    disp('Low freq. group delay in ns =')
    disp(GDLF)
    disp(' ')
    GD_xmax = input('Enter the max frequency to be plotted for S21 group
delay (GHz): ');
    plot(f,GD,'.b')
    grid
    xlabel('Frequency in GHz')
    ylabel('Group Delay in ns')
    title('TDVNA:  S21(f) Group Delay vs. Frequency')
    XLim([0 GD_xmax])
    disp('Group delay plot displayed - press any key to continue')
    pause
    semilogx(f,GD,'.b')
    grid
    xlabel('Frequency in GHz')
    ylabel('Group Delay in ns')
    title('TDVNA:  S21(f) Group Delay vs. Frequency')
    XLim([0 GD_xmax])
    pause
```

```
end
% output results to working directory
disp(' ')
dataout = input('Do you want to save results to the working directory?
(1=yes  0=no)  ');
if dataout == 1
    f = (f)';
    Sx1db = (Sx1db)';
    GD = (GD)';
    name_out = input('Please enter the desired pre-fix for the output file
names:  ','s');
    freq1_name = strcat(name_out,'_S11Freq.txt');
    if drive == 1
        freq1_name = strcat('A:',name_out,'_S11Freq.txt');
    end
    S11_name = strcat(name_out,'_S11dB.txt');
    if drive == 1
        S11_name = strcat('A:',name_out,'_S11dB.txt');
    end
    GD11_name = strcat(name_out,'_S11GD.txt');
    if drive == 1
        GD11_name = strcat('A:',name_out,'_S11GD.txt');
    end
    freq2_name = strcat(name_out,'_S21Freq.txt');
    if drive == 1
        freq2_name = strcat('A:',name_out,'_S21Freq.txt');
    end
    S21_name = strcat(name_out,'_S21dB.txt');
    if drive == 1
        S21_name = strcat('A:',name_out,'_S21dB.txt');
    end
    GD21_name = strcat(name_out,'_S21GD.txt');
    if drive == 1
        GD21_name = strcat('A:',name_out,'_S21GD.txt');
    end
    if meas == 1  % i.e. S11
        dlmwrite(freq1_name,f,'\r');
        disp('S11Freq saved')
        dlmwrite(S11_name,Sx1db,'\r');
        disp('S11dB saved')
        dlmwrite(GD11_name,GD,'\r');
        disp('S11GD saved')
    end
    if meas == 2  % i.e. S21
        dlmwrite(freq2_name,f,'\r');
        disp('S21Freq saved')
```

```
        dlmwrite(S21_name,Sx1db,'\r');
        disp('S21dB saved')
        dlmwrite(GD21_name,GD,'\r');
        disp('S21GD saved')
    end
end
disp('end of TDVNAv21.m program')
warning on;
end
```