Code for Micro-Reflection Limit

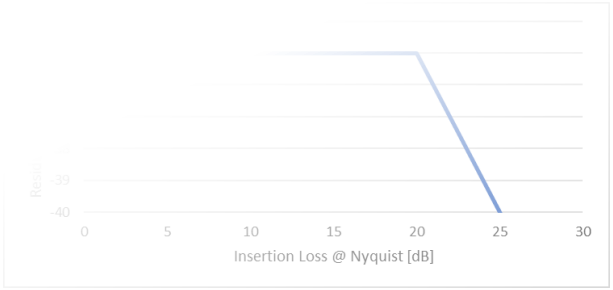
Contribution to IEEE 802.3cy

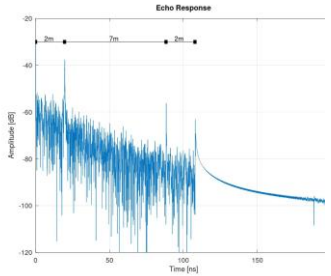
Ragnar Jonsson
Marvell

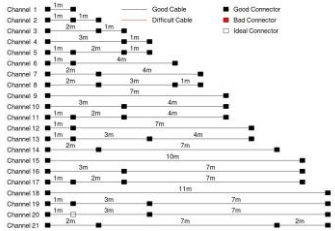
September 28, 2021

Introduction

- In the Telephonic Interim Meeting on March 30, 2021, the micro-reflection limit text from jonsson_3cy_01a_03_30_21 was adopted
- Contribution jonsson_3cy_01_08_10_21 suggests specific limits to use in the text
- This contribution provides the accompanying MATLAB/Octave code to make it easier to evaluate the micro-reflection limits







Values Proposed in jonsson_3cy_01_08_10_21

Parameter	Parameter Value	Parameter Description
Δf	**2.5MHz**	The sample frequency spacing for the frequency domain transfer function measurements
N	**4096**	Number of sampling points to use for the time domain representation of the echo impulse response
N_{seg}	**4**	Number of samples in each segment
$N_{discard}$	**12**	Number of largest segments to discard

- f_c is **4GHz**,
- *REMmax* is **-30dB** and
- *REMoffset* is **20dB**

Using the MATLAB code

Run the code on the S-parameter data in the Touchstone file `cable.s4p` using:

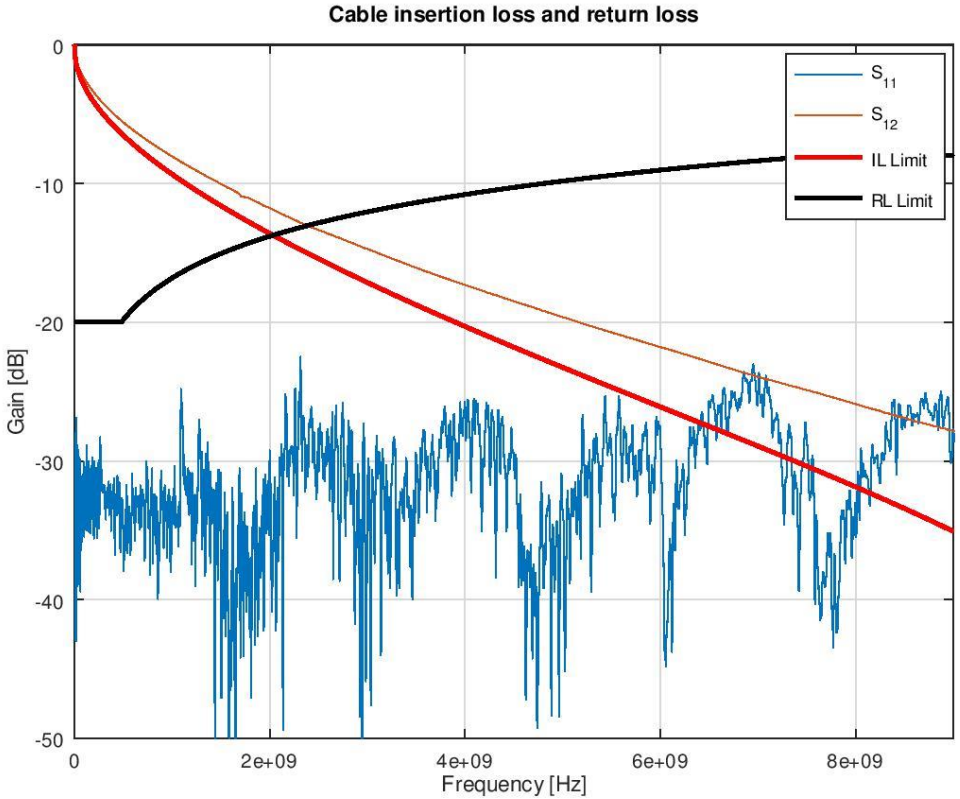
```
[REM,ETM,REMlimit,h_echo,sort_ix,P_k,N_bins,N_discard] = jonsson_3cy_01_09_28_21('cable.s4p');
```

- The function will generate several plots and output the micro-reflection metrics along with other values from the calculations
- If the frequency spacing in the Touchstone file does not match the expected frequency spacing, the function will issue a warning and interpolate the S-parameter data to the expected frequency spacing
- The function takes several optional parameters for experimenting with values other than those proposed in jonsson_3cy_01_08_10_21
- The function to read the .s4p files has been made to be as simple as possible, so it has limited error checks and may have problems with reading files with line endings that do not match the standard line ending of operating system (i.e., Windows, Linux, Mac)
- The function also calls updated version of previously shared MATLAB function:

https://www.ieee802.org/3/cy/public/sep21/jonsson_3cy_01a_03_23_21.m

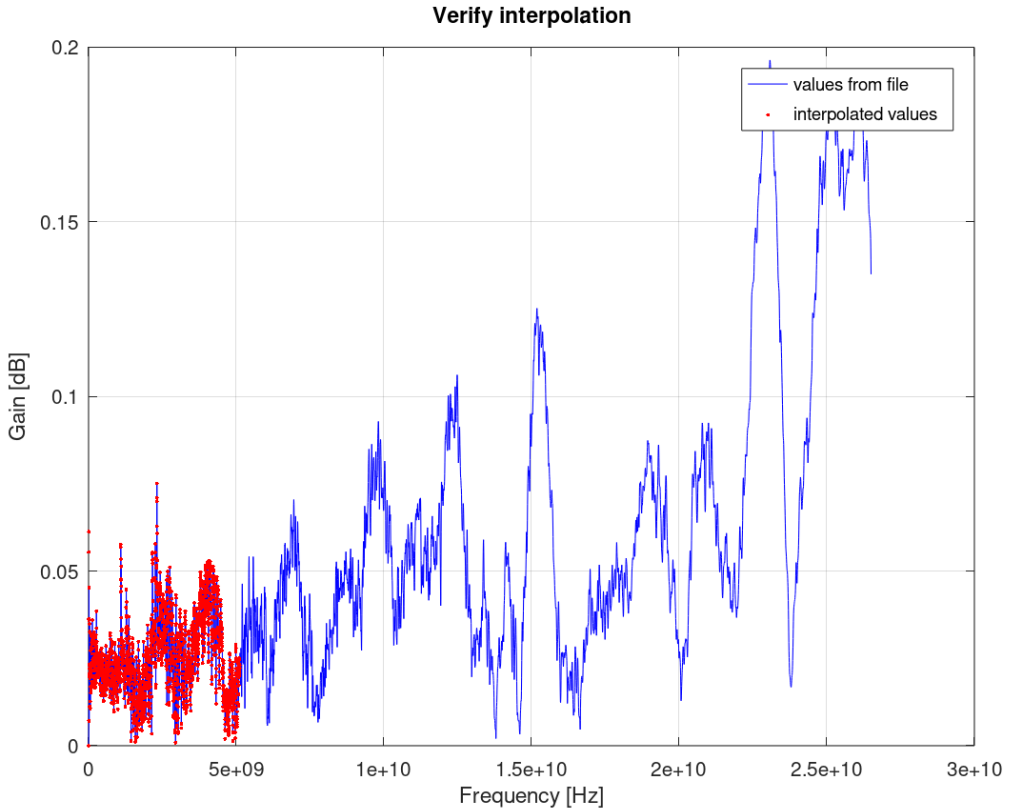
Cable IL and RL Plot

- The first plot generated shows the S-parameters corresponding to IL and RL
- The purpose of this plot is to provide a sanity check of how the function interoperates the data in the file
- This can also be used as a reference to see how the IL and RL compare to the existing limits



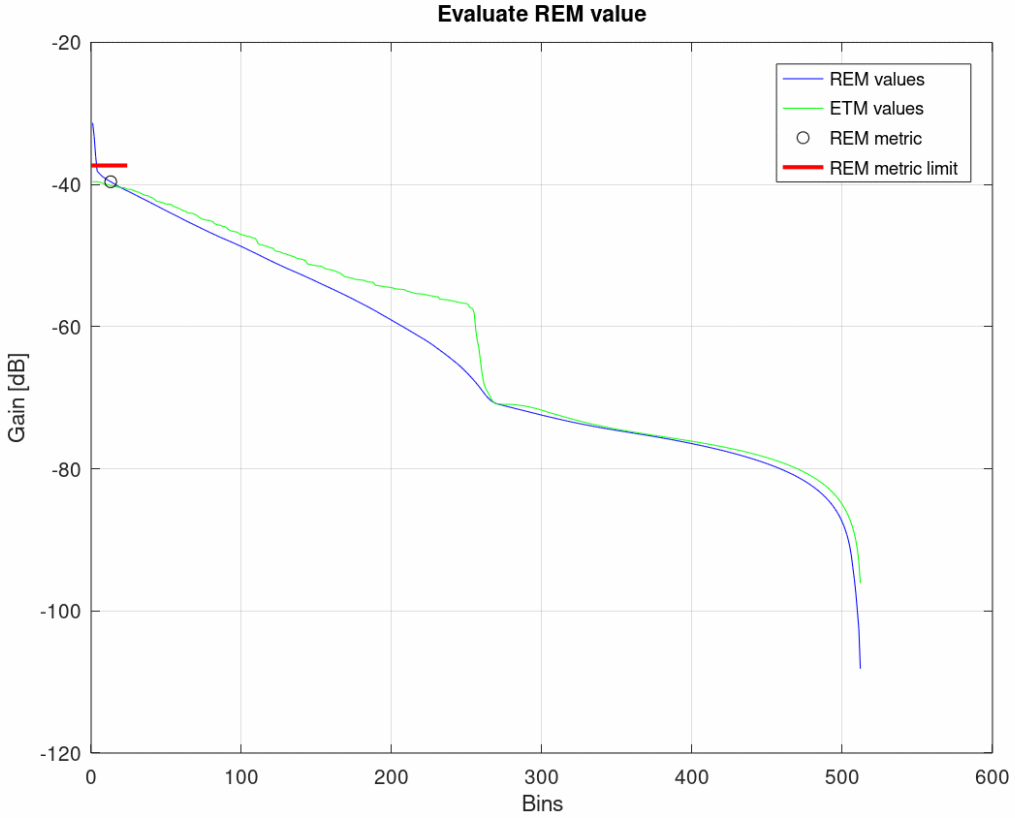
Verify Interpolation Plot

- The second plot is only generated if the frequency spacing in the input file does not match the required frequency spacing
- The purpose of this plot is to help evaluate if the interpolation to the required frequencies was done correctly



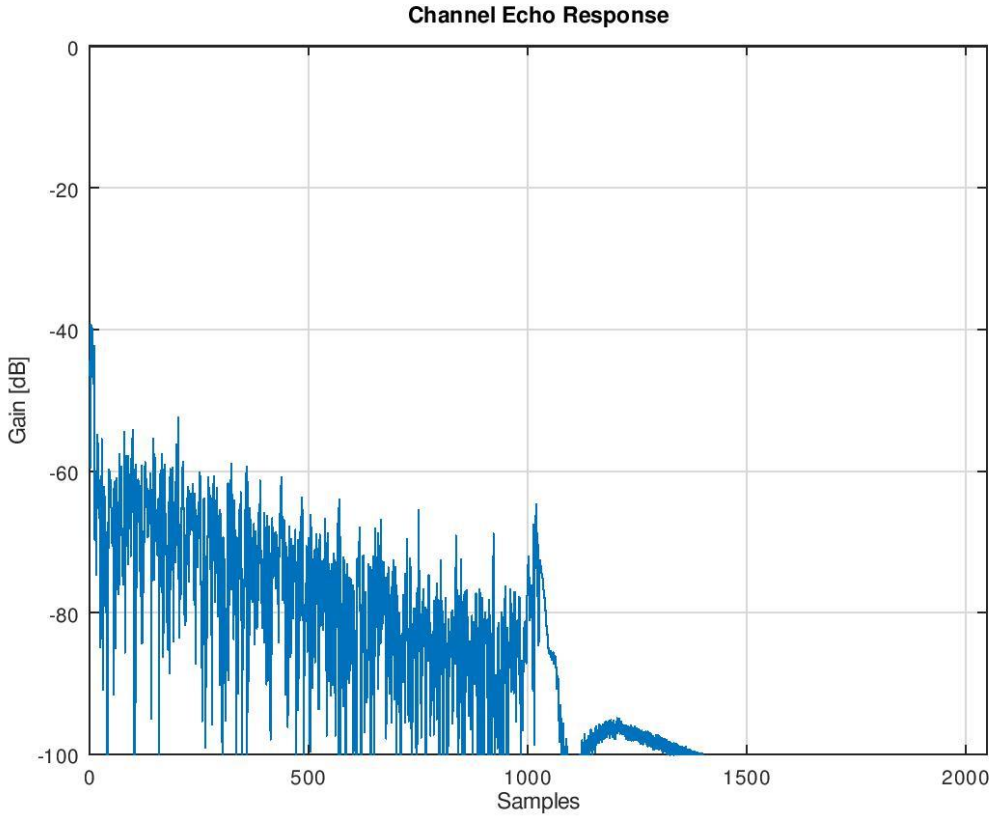
Evaluate REM Value Plot

- The third plot generated is the key information from the function
- The plot shows the REM and ETM values for different number of bins
- The plot also shows the REM metric (a single value) and the corresponding limit on this REM metric



Channel Echo Response Plot

- The fourth plot shows the time domain echo response calculated from the S11 parameters in the file
- This plot is informative and can also help evaluate if the code has correctly interpreted the S-parameter data in the file



Conclusion

The micro-reflection text, adopted on March 30, 2021, has several TBD values

We have suggested specific values for some of these TBD values (see jonsson_3cy_01_08_10_21)

This presentation provides MATLAB code to help others evaluate the proposed limits

MATLAB Code

```
function [REM,ETM,REMlimit,h_echo,sort_ix,P_k,N_bins,N_discard] = jonsson_3cy_
01_09_28_21(filename,f_c,REMmax,REMoffset,N_discard,N_seg,N,df,show_plot)
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
% FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
% THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
% LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
% FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
% DEALINGS IN THE SOFTWARE.

%%% check arguments %%%
if((exist('f_c')~=1) || isempty(f_c))
  f_c = 4e9;
end
if((exist('REMmax')~=1) || isempty(REMmax))
  REMmax = -30;
end
if((exist('REMoffset')~=1) || isempty(REMoffset))
  REMoffset = 20;
end
if((exist('N_discard')~=1) || isempty(N_discard))
  N_discard = 12;
end
if((exist('N_seg')~=1) || isempty(N_seg))
  N_seg = 4;
end
if((exist('N')~=1) || isempty(N))
  N = 4096/2;
end
if((exist('df')~=1) || isempty(df))
  df = 2.5e6;
end
if((exist('show_plot')~=1) || isempty(show_plot))
  show_plot = 1;
end

%%% initialize variables
N_bins = N/N_seg;

%%% read Touchstone file with S-parameters %%%
[s,f,comments,par,sdd]=rtouchstone( filename );
s11 = sdd{1,1};
s12 = sdd{1,2};
if(f(1) ~= 0)
  f = [0 f];
  s11 = [0 sdd{1,1}];
  s12 = [s12(1) s12];
end

%%% plot S11 and S12 against limits %%%
f_ix = find(f < 9e9);
f_MHz = f(f_ix)/1e6;
IL_limit = 0.00135*f_MHz + 0.3564*(f_MHz.^0.45) + 0.495*(f_MHz/7500).^6;
RL_limit = min(20,max(8,20-10*log10(f_MHz/480)));
if(show_plot)
  figure
  plot(f,20*log10(abs(s11)))
  hold on
  plot(f,20*log10(abs(s12)));
  plot(f_MHz*1e6,-IL_limit,'r','LineWidth',2);
  plot(f_MHz*1e6,-RL_limit,'k','LineWidth',2);
  title('Cable insertion loss and return loss');
  axis([0 9e9 -50 0])
  grid on
  xlabel('Frequency [Hz]')
  ylabel('Gain [dB]')
  legend('S_{11}','S_{12}','IL Limit','RL Limit');
end

%%% find REM limit value %%%
IL_fc = -20*log10(abs(spline(f,s12,f_c)));
REMlimit = min(REMmax,-IL_fc-REMoffset);

%%% verify frequency spacing and adusting if needed %%%
if(any(diff(f)-df))
  warning('The frequency spacing is not correct in .s4p file')
  warning('Converting to correct frequency spacing -
 this will affect the results')

  %%% test date from .S4P file %%%
  if(f(end) < N*df)
    warning('The frequency range is not sufficient in the .s4p file -
 compensating');
    N = floor(f(end)/df/2)*2
    N_bins = N/N_seg;
  end

  %%% convert to "correct" frequency spacing %%%
  fq = df*[0:N];
  sq = spline(f,s11,fq);

  %%% show original and interpolated data for comparison %%%
  if(show_plot)
    figure
    plot(f,abs(s11),'b');
    hold on;
    plot(fq,abs(sq),'r.');
    title('Verify interpolation');
    grid on
    xlabel('Frequency [Hz]')
    ylabel('Gain [dB]')
    legend('values from file','interpolated values')
  end

  %%% work with interpolated data from now on
  f = fq;
  s11 = sq;
end

%%% calculate REM and EMT %%%
[REM,ETM,h_echo,sort_ix,P_k] = jonsson_3cy_01a_03_23_21(f,s11,N_bins,N_seg,N_d
iscard,0);

%%% show REM value and limit %%%
if(show_plot)
  figure
  plot([1:N_bins],REM,'b',[1:N_bins],ETM,'g',N_discard+1,REM(N_discard+1),'ok'
,[0 2*N_discard],REMlimit*[1 1],'r','LineWidth',2)
  title('Evaluate REM value')
  grid on
  legend('REM values','ETM values','REM metric','REM metric limit')
  xlabel('Bins')
  ylabel('Gain [dB]')

  figure
  plot(20*log10(abs(h_echo)))
  title('Channel Echo Response')
  grid on
  axis([0 N -100 0])
  xlabel('Samples')
  ylabel('Gain [dB]')
end

end
```

```
function [s,f,comments,par,sdd]=rtouchstone( filename )
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
% FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
% THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
% LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
% FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
% DEALINGS IN THE SOFTWARE.

% NOTE: This function is made to be as simple as possible, so it has limited
% error checks and may struggle with reading files with incorrect line ending


  %%% parse extention for S-matrix size %%%
  [filepath,name,ext] = fileparts(filename);
  M = str2num(ext(3:end-
1)); %%% this is a primitive way of finding size of S matrix

  %%% read file %%%
  [fp, emessage] = fopen( filename, 'rt' );
  if (fp < 0)
    error( emessage );
  end

  %%% read data from file %%%
  comments = {};
  st = 0;
  v = [];
  while ((isempty(v)==1) & (st ~= -1))
    v=fscanf(fp,'%g');
    if (isempty(v))
      st = fgets(fp);
      comments{end+1} = st;
      if (st(1) == '#')
        %par = strsplit(st);
        par = regexp(st, '\s+', 'split');
      end
    end
  end
  fclose(fp);

  %%% reshape the S parameter data %%%
  N=length(v);
  M2 = M*M;
  ML = 2*M2 + 1;
  v = reshape(v,ML,N/ML);

  %%% convert file data into linear complex number data %%%
  dformat = par{4};
  switch(lower(dformat))
    case 'ri'
      A = zeros(M2+1,ML);
      A(1,1) = 1;
      for n = 1:M2
        A(n+1,2*n+[0 1]) = [1 i];
      end
      v = A*v;
    case 'db'
      tmp = 10.^(v(2:2:end,:)/20).*exp(i*pi/180*v(3:2:end,:));
      v = [v(1,:); tmp];
    otherwise
      error(['unknown format: ' par{4}]);
  end

  %%% convert frequency to Hz %%%
  switch(lower(par{2}))
    case 'hz'
      f_scale = 1;
    case 'khz'
      f_scale = 1e3;
    case 'mhz'
      f_scale = 1e6;
    case 'ghz'
      f_scale = 1e9;
    case 'thz'
      f_scale = 1e12;
    otherwise
      error(['unknown frequency unit: ' par{2}])
  end
  f = f_scale * v(1,:);

  %%% convert to square S-matrix %%%
  for m = 1:M,
    for n = 1:M,
      ix = (m-1)*M + (n-1) + 2;
      s{m,n} = v(ix,:);
    end
  end

  %%% create diffrential-differential matrix (if requested) %%%
  seq = [1:4];
  if(nargout > 4)
    sdd{2,2} = 0;
    Mdd = M/2;
    for m = 1:Mdd,
      for n = 1:Mdd,
        m1 = seq(m);
        n1 = seq(n);
        m2 = seq(m+Mdd);
        n2 = seq(n+Mdd);
        sdd{m,n} = 0.5*(s{m1,n1}-s{m1,n2}-s{m2,n1}+s{m2,n2});
      end
    end
  end
end
```

```
function [REM,ETM,h_echo,sort_ix,P_k] = jonsson_3cy_01a_03_23_21(f,s11,N_bins,
N_seg,N_discard,df_nonstandard)
%Evaluate micro-reflections
%Usage:
%  jonsson_3cy_01a_03_23_21(f,s11,N_bins,N_seg,N_discard,df_nonstandard)
%where <f> is frequency, <s11> is S11 parameters, <N_bins> is the number
%of time domain bins used in the calculation, <N_seg> is the number
%of samples in each time domain bin, and <N_discard> is the number of
%segments to discard when computing REM and ETM.
%The parameter <df_nonstandard> can be used if the
%frequency sampling of s11 is not according to specification.
%The function returns the metric values and intermediate values.
%Version 1.1 -- March 23, 2021
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
% FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
% THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
% LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
% FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
% DEALINGS IN THE SOFTWARE.

%%% check input arguments %%%
if(nargin < 6)
  df_nonstandard = 0; %% non-standard frequency support off by default
end

%%% number of samples to use %%%
K_N = N_bins * N_seg;
N_samples = K_N*2;  %even number of samples

%%% experimental code to convert non-
standard frequency sampling to standard %%%
if(df_nonstandard)
  fq = df_nonstandard*[0:K_N];
  sq = cr_spline(f,s11,fq);
else
  fq = f;
  sq = s11;
end

%%% calculate echo impulse response and power %%%
h_echo = jonsson_3cy_01_03_23_21_f2t(sq,fq,N_samples);

%%% find power in each time bin %%%
h2 = h_echo.^2;
P_k = zeros(1,N_bins);
m1 = 0;
for n=1:N_bins,
  m0 = m1 + 1;
  m1 = round(n*N_seg);
  P_k(n) = mean(h2(m0:m1))*(N_seg);
end

%%% find residual echo metric %%%
[p_sort,sort_ix] = sort(P_k);
sort_ix = sort_ix(end:-1:1);
p_sum = cumsum(p_sort);
REM = 10*log10(p_sum(end:-1:1));

%%% find residual echo RE_k %%%
RE_k = P_k;
RE_k(sort_ix(1:N_discard)) = 0;

%%% calculate echo tail metric %%%
tmp = cumsum(RE_k(end:-1:1));
ETM = 10*log10(tmp(end:-1:1));

end %%% End of jonsson_3cy_01a_03_23_21
```

```
function h_n = jonsson_3cy_01_03_23_21_f2t(H,f,N)

%RJf2t - Impulse (time) response for a given frequency response.
%Usage:
%  h = jonsson_3cy_01_03_23_21_f2t(H,f,N)
% where <H> is the frequency response given at frequencies <f>,
% <T> is the sampling interval, and <N> is the number of output
% samples (must be even).

%%% test arguments %%%
N_H = prod(size(H));
N_good = prod(size(find(H == H)));
if(N_H ~= N_good)
  error('Signal has invalid samples')
end

nonuniform_spacing = sum(abs(diff(abs(diff(f)))));
if(nonuniform_spacing)
  error('spacing of frequency points is not uniform')
end

if(f(1) ~= 0)
  error('frequency does not start at DC')
end

%%% re-shape arguments %%%
E_k = H(:);

%%% adjust phase of frequency response %%%
K_N = ceil(N/2);
ang_N = angle(E_k(K_N+1));
x0 = ang_N/(pi);
E_k = E_k.*exp(-j*2*pi*x0*[0:K_N]'/K_N/2);
E_k(1) = real(E_k(1));
H_k = [E_k(1:K_N); real(E_k(K_N+1)); conj(E_k(K_N:-1:2))];

%%% find impulse response from IDFT %%%
h_n = real(ifft(H_k));

end %%% End of jonsson_3cy_01_03_23_21_f2t
```

```
function s = cr_spline(x,y,xq)
% Catmull-Rom spline
%   s = cr_ spline(x,y,xq);
% See more at https://en.wikipedia.org/wiki/Cubic_Hermite_spline

%%% initialize %%%
N_x = length(x);
[M_xq,N_xq] = size(xq);

%%% reshape arguments %%%
x = x(:);
y = y(:);
xq = xq(:);

%%% Catmull-Rom spline coefficients %%%
cr0 = [0 0 1 0];
cr1 = [0 1 0 -1]/2;
cr2 = [-1 4 -5 2]/2;
cr3 = [1 -3 3 -1]/2;

%%% find closest points for x-values %%%
xm = (x(1:end-1) + x(2:end) - 1e-10)/2; %% midpoint in interval
[d,m]=min(abs(reshape(xm,N_x-1,1) - reshape(xq,1,N_xq*M_xq)));
u = (xq - x(m))./(x(m+1) - x(m));

%%% Use Farrow Structure to implement spline %%%
y0 = conv(y, cr0);
y1 = conv(y, cr1);
y2 = conv(y, cr2);
y3 = conv(y, cr3);

s = y3(m+2).*(u.^3)+y2(m+2).*(u.^2) + y1(m+2).*u + y0(m+2);
s = reshape(s,M_xq,N_xq);

end %%% End of cr_spline
```

Essential technology, done right™