APPENDIX G:
MATLAB CODE USED TO EVALUATE NONLINEAR INDICATORS

This appendix provides the MATLAB code that was used to calculate and evaluate the different nonlinear indicators. The code corresponds to the discussion of the experimental results in Chapter 7.

Main Program

```
clear
close all

%This is the program that will evaluate each transducer data set
%in terms of the nonlinear index.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Determine the voltage data files and the pressure data files    %
%that will be used.                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

info='00068, f/#=1, f ~ 6 MHz';

%Transducer ckt data
file='/dunn/bigelow/Transducer_impedance/00066.mat';%file RLC info
load(file); %Load RLC values and resonance frequency

%Location of data files.
path='/dunn/bigelow/nonlinear_data/00066/11-10-01b/';

%The voltage will be the voltage waveform that has been applied to
%the real transducer

v_file='v00066p3_';

%The pressure will be the pressure waveform measured at the focus
%of the transducer

p_file='00066p3_';

%Water temperature
Tc=20.288; %in C
c=CfromT(Tc);
```

```
%Transducer parameters
a=0.9525e-2; %Transducer radius in m

%Error parameters
dc=5.0125; %Variation in p_c extrapolation (total width +-dc/2)
dr=1.3139; %Variation in p_r extrapolation (total width +-dr/2)
da=3.2124; %Variation in p_avg extrapolation (total width +-da/2)

%Flag for including pa in plots
pa_in=1; %1 to include
%*************************************************************

%Settings for RA-30/Attenuators
RA30={'30_' '24_' '18_' '12_' '06_' '00_'};

ritec={'02' '06' '10' '14' '18' '22' '30' '38' '46' '54' '58'…
       '62' '66' '70' '78' '86'};

%Calibration factor
cal_factor=0.0425; %V/MPa for the Hydrophone.

%Code paramters
Npol=8; %Degree of polynomial to fit extrapolation error data.
extrap_err=10; %Extrapolation error in %
ref_low=1; %The low voltae reference for the relative NI

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Load in the data                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

di=0;
P=[];
for ii=1:length(RA30)
   for ji=1:length(ritec)
      dat_p=[path p_file char(RA30(ii)) char(ritec(ji)) '.txt'];
      dat_v=[path v_file char(RA30(ii)) char(ritec(ji)) '.txt'];

      fid_p=fopen(dat_p,'r');

      if fid_p~=-1
         di=di+1;

         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
         %Read in the data
         %Get close pressure data
         A=fscanf(fid_p,'%f',[2,inf]);
```

```
p=1e6*(A(2,:)-mean(A(2,:)))/cal_factor;
t=A(1,:);

%Determine some important pressure values
pc(di)=max(p);
pr(di)=abs(min(p));
pa(di)=(pc(di)+pr(di))/2;

P=[P p'];

%Determine frequency of maximum pressure value.
f_peak(di)=find_fpeak(p,t);

%Get voltage data
fid_v=fopen(dat_v,'r');
B=fscanf(fid_v,'%f',[2,inf]);

v=100*(B(2,:)-mean(B(2,:)));
tv=B(1,:);

%Determine some important voltage values
v_pp(di)=(max(v)-min(v));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Determine extrapolation factor                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Find the signal values assuming an RLC circuit
%Determine value for each frequency
Vs=fft(v);

M=length(Vs);

%Find corresponding freq. values
d_f1=[0:(M/2-1)]*2*pi/M;
d_f2=[(M/2):(M-1)]*2*pi/M - 2*pi;

d_f=[d_f1 d_f2];

dt=tv(2)-tv(1);

freq=d_f*(1/dt)/(2*pi);

%Now for each frequency, find the voltage accross R
for fi=1:length(freq)
```

```
            w=freq(fi)*2*pi;

            if abs(w)<1e-12
               Vout(fi)=0;
            else
               Zc1=R1 + j*w*L1 + 1/(j*w*C1);
               Vout(fi)=(R1/Zc1)*Vs(fi);
            end
        end
        %The voltage value at the maximum frequency
        %(Scaled for better fitting)
        volt(di)=max(abs(Vout))/1000;

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        %Calculate F, alpha, and G
        [m1,n1]=max(abs(p));
        [m2,n2]=max(abs(v));

        time_prop=t(n1)-tv(n2); %Determine propagation time
        F=c*time_prop; %Determin F (focal length)

        %Determine the transducer G factor used to find sigma_m
        G(di)=sqrt((F/(pi*(c/f_peak(di))*…
                             (0.8224*(F/(2*a)))^2))^2  + 1);

        %Determine alpha
        alpha(di)=asin(a/F);

        %Determine Ro value
        Ro=(c/f_peak(di))/(1-cos(alpha(di)));

        %Determine F/Ro value
        F_Ro(di)=F/Ro;

        fclose(fid_p);
        fclose(fid_v);
     end
  end
end

N=di; %Save number of scans

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Calculate the possible NI                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for ri=1:N
```

```
   %1 Asymmetric Ratio%%%%%
   p_asym(ri)=pc(ri)/pr(ri);
   %%%%%%%%%%%%%%%%%%%%%%%%%

   %2 Sigma_s%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   [sigma_s(ri),sigma_sa(ri)]=find_sigma_s(pc(ri),pr(ri),F_Ro(ri)…
                       ,alpha(ri),c);
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %3 Sigma_m%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   sigma_m(ri)=find_sigma_m(pa(ri),f_peak(ri),2*G(ri),F,c);
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %4 Field Sigma%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   sigma_z(ri)=find_sigma_z(pa(ri),f_peak(ri),F,c);
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %5 Second Harmonic Ratio%%%%%%%%%%
   h2(ri)=find_h2(P(:,ri),t(2)-t(1));
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %6 Absolute SI%%%%%%%%%%%%%%%%%%%%%%%%%%%
   [SI_fa15(ri),SI_fa2(ri)]=fa_si(P(:,ri));
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %Determine the extrapolation gain, Gv
   Gv=volt(ri)/volt(ref_low);

   %7 Relative SI%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   SI_me(ri)=my_si(P(:,ri),P(:,ref_low),Gv);
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %8 Focal Pressues%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   Pce(ri)=100*abs(pc(ri)-Gv*pc(ref_low))/pc(ri);
   Pre(ri)=100*abs(pr(ri)-Gv*pr(ref_low))/pr(ri);
   Pae(ri)=100*abs(pa(ri)-Gv*pa(ref_low))/pa(ri);
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %Show that above is sensitive to value of ref_low

   %Determine linearly extrapolated pressures for plotting
   pc_lin(ri)=Gv*pc(ref_low);
   pr_lin(ri)=Gv*pr(ref_low);
   pa_lin(ri)=Gv*pa(ref_low);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Determine value of extrapolation factor when error is no longer %
%acceptable                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Find spline curve representation of pc,pr, and pa data that can
%be used to extrapolate pressure values at intermediate voltages.
%(Polynomial Extrapolation will increase error at low voltages
%since curve will not pass through data points).
SP_pc=spline(volt,pc);
SP_pr=spline(volt,pr);
SP_pa=spline(volt,pa);

%Find the largest factor for which the error for all smaller
%factors is less than the acceptable error level.
Vce=fzero('find_extrap_factor',volt(ceil(length(volt)/2)),[],…
           SP_pc,pc,volt,Npol,extrap_err);
Vre=fzero('find_extrap_factor',volt(ceil(length(volt)/2)),[],…
           SP_pr,pr,volt,Npol,extrap_err);
Vae=fzero('find_extrap_factor',2*max([Vce Vre]),[],…
           SP_pa,pa,volt,Npol,extrap_err);

%Determine the error voltages about this point
Vce1=fzero('find_extrap_factor',Vce,[],SP_pc,pc,volt,Npol,…
            extrap_err-dc/2);
Vce2=fzero('find_extrap_factor',Vce,[],SP_pc,pc,volt,Npol,…
            extrap_err+dc/2);

Vre1=fzero('find_extrap_factor',Vre,[],SP_pr,pr,volt,Npol,…
            extrap_err-dr/2);
Vre2=fzero('find_extrap_factor',Vre,[],SP_pr,pr,volt,Npol,…
            extrap_err+dr/2);

Vae1=fzero('find_extrap_factor',Vae,[],SP_pa,pa,volt,Npol,…
            extrap_err-da/2);
Vae2=fzero('find_extrap_factor',Vae,[],SP_pa,pa,volt,Npol,…
            extrap_err+da/2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Determine Values of NI when error is no longer acceptable       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Determine Polynomials to fit the NI
SPasym=spline(volt,p_asym);

SPsigs=spline(volt,sigma_s);
SPsigsa=spline(volt,sigma_sa);
```

```
SPsigm=spline(volt,sigma_m);

SPsigz=spline(volt,sigma_z);

SPh2=spline(volt,h2);

SPsi15=spline(volt,SI_fa15);
SPsi2=spline(volt,SI_fa2);

SPsir=spline(volt,SI_me);

SPce=spline(volt,Pce);
SPre=spline(volt,Pre);
SPae=spline(volt,Pae);

%Determine NI at values of acceptable error voltage

VAL_pasym=ppval(SPasym,[Vce Vre Vae]');

VAL_sigs(1,:)=ppval(SPsigs,[Vce Vre Vae]')';
VAL_sigs(2,:)=ppval(SPsigsa,[Vce Vre Vae]')';


VAL_sigm=ppval(SPsigm,[Vce Vre Vae]');

VAL_sigz=ppval(SPsigz,[Vce Vre Vae]');

VAL_h2=ppval(SPh2,[Vce Vre Vae]');

VAL_si(1,:)=ppval(SPsi15,[Vce Vre Vae]')';
VAL_si(2,:)=ppval(SPsi2,[Vce Vre Vae]')';


VAL_sir=ppval(SPsir,[Vce Vre Vae]');

[VAL_pfocal(1)]=ppval(SPce,Vce);
[VAL_pfocal(2)]=ppval(SPre,Vre);
[VAL_pfocal(3)]=ppval(SPae,Vae);

%Determin NI values corresponding to suspected error bars (Error
%in extrapolation factor).
ERR_pasym=[abs(ppval(SPasym,Vce2)-ppval(SPasym,Vce1)) ...
      abs(ppval(SPasym,Vre2)-ppval(SPasym,Vre1)) ...
      abs(ppval(SPasym,Vae2)-ppval(SPasym,Vae1))];

ERR_sigs=[abs(ppval(SPsigs,Vce2)-ppval(SPsigs,Vce1)) ...
      abs(ppval(SPsigs,Vre2)-ppval(SPsigs,Vre1)) ...
      abs(ppval(SPsigs,Vae2)-ppval(SPsigs,Vae1)); ...
      abs(ppval(SPsigsa,Vce2)-ppval(SPsigsa,Vce1)) ...
      abs(ppval(SPsigsa,Vre2)-ppval(SPsigsa,Vre1)) ...
      abs(ppval(SPsigsa,Vae2)-ppval(SPsigsa,Vae1))];

ERR_sigm=[abs(ppval(SPsigm,Vce2)-ppval(SPsigm,Vce1)) ...
      abs(ppval(SPsigm,Vre2)-ppval(SPsigm,Vre1)) ...
      abs(ppval(SPsigm,Vae2)-ppval(SPsigm,Vae1))];

ERR_sigz=[abs(ppval(SPsigz,Vce2)-ppval(SPsigz,Vce1)) ...
      abs(ppval(SPsigz,Vre2)-ppval(SPsigz,Vre1)) ...
      abs(ppval(SPsigz,Vae2)-ppval(SPsigz,Vae1))];

ERR_h2=[abs(ppval(SPh2,Vce2)-ppval(SPh2,Vce1)) ...
      abs(ppval(SPh2,Vre2)-ppval(SPh2,Vre1)) ...
      abs(ppval(SPh2,Vae2)-ppval(SPh2,Vae1))];

ERR_si=[abs(ppval(SPsi15,Vce2)-ppval(SPsi15,Vce1)) ...
      abs(ppval(SPsi15,Vre2)-ppval(SPsi15,Vre1)) ...
      abs(ppval(SPsi15,Vae2)-ppval(SPsi15,Vae1)); ...
      abs(ppval(SPsi2,Vce2)-ppval(SPsi2,Vce1)) ...
      abs(ppval(SPsi2,Vre2)-ppval(SPsi2,Vre1)) ...
      abs(ppval(SPsi2,Vae2)-ppval(SPsi2,Vae1))];

ERR_sir=[abs(ppval(SPsir,Vce2)-ppval(SPsir,Vce1)) ...
      abs(ppval(SPsir,Vre2)-ppval(SPsir,Vre1)) ...
      abs(ppval(SPsir,Vae2)-ppval(SPsir,Vae1))];

ERR_pfocal=[abs(ppval(SPce,Vce2)-ppval(SPce,Vce1)) ...
      abs(ppval(SPre,Vre2)-ppval(SPre,Vre1)) ...
      abs(ppval(SPae,Vae2)-ppval(SPae,Vae1))];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Plot the results                                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Plot the focal pressures
figure(1)
clf
plot(v_pp,pc/1e6,'o-')
hold
grid
plot(v_pp,pa/1e6,'rx-.')
plot(v_pp,pr/1e6,'gd--')

plot(v_pp,pc_lin/1e6)
plot(v_pp,pa_lin/1e6,'r-.')
plot(v_pp,pr_lin/1e6,'g--')
```

```
xlabel('V_p_p (V)')
ylabel('Pressure at Focus (MPa)')
legend(' = Peak Compressional Pressure (MPa)',…
       ' = Average Peak Pressure (MPa)',…
       ' = Peak Rarefractional Pressure (MPa)');
title(info);



%%%%%%%%%%%%%
%THESIS PLOT%
%%%%%%%%%%%%%
figure(2)
clf
subplot(321)
semilogx(v_pp,p_asym,'o-','LineWidth',2,'MarkerSize',3)
grid
hold
semilogx(v_pp,VAL_pasym(1)*ones(1,length(v_pp)),…
         'r-','LineWidth',2);
semilogx(v_pp,VAL_pasym(2)*ones(1,length(v_pp)),…
         'r-.','LineWidth',2);
if pa_in==1;
   semilogx(v_pp,VAL_pasym(3)*ones(1,length(v_pp)),…
            'r--','LineWidth',2);
end
xlabel('v_p_p (V)')
ylabel('Pulse Asymmetry (p_c/p_r)')
if pa_in==1
   legend('= Indicator','= p_c Threshold','= p_r Threshold',…
          '= p_a_v_g Threshold');
else
   legend('= Indicator','= p_c Threshold','= p_r Threshold');
end
title('(a)');

subplot(322)
semilogx(v_pp,sigma_sa,'o-','LineWidth',2,'MarkerSize',3)
grid
hold
semilogx(v_pp,VAL_sigs(2,1)*ones(1,length(v_pp)),…
         'r-','LineWidth',2);
semilogx(v_pp,VAL_sigs(2,2)*ones(1,length(v_pp)),…
         'r-.','LineWidth',2);
if pa_in==1;
   semilogx(v_pp,VAL_sigs(2,3)*ones(1,length(v_pp)),…
            'r--','LineWidth',2);
```

```
end
xlabel('v_p_p (V)')
ylabel('\sigma_s (From P_a_s_y_m)')
title('(b)');

subplot(323)
semilogx(v_pp,sigma_s,'o-','LineWidth',2,'MarkerSize',3)
grid
hold
semilogx(v_pp,VAL_sigs(1,1)*ones(1,length(v_pp)),…
         'r-','LineWidth',2);
semilogx(v_pp,VAL_sigs(1,2)*ones(1,length(v_pp)),…
         'r-.','LineWidth',2);
if pa_in==1;
   semilogx(v_pp,VAL_sigs(1,3)*ones(1,length(v_pp)),…
            'r--','LineWidth',2);
end
xlabel('v_p_p (V)')
ylabel('\sigma_s (From p_r)')
title('(c)');

subplot(324)
semilogx(v_pp,sigma_m,'o-','LineWidth',2,'MarkerSize',3)
grid
hold
semilogx(v_pp,VAL_sigm(1)*ones(1,length(v_pp)),…
         'r-','LineWidth',2);
semilogx(v_pp,VAL_sigm(2)*ones(1,length(v_pp)),…
         'r-.','LineWidth',2);
if pa_in==1;
   semilogx(v_pp,VAL_sigm(3)*ones(1,length(v_pp)),…
            'r--','LineWidth',2);
end
xlabel('v_p_p (V)')
ylabel('\sigma_m')
title('(d)');

subplot(325)
semilogx(v_pp,sigma_z,'o-','LineWidth',2,'MarkerSize',3)
grid
hold
semilogx(v_pp,VAL_sigz(1)*ones(1,length(v_pp)),…
         'r-','LineWidth',2);
semilogx(v_pp,VAL_sigz(2)*ones(1,length(v_pp)),…
         'r-.','LineWidth',2);
if pa_in==1;
   semilogx(v_pp,VAL_sigz(3)*ones(1,length(v_pp)),…
            'r--','LineWidth',2);
end
xlabel('v_p_p (V)')
ylabel('\sigma_z')
title('(e)');

subplot(326)
semilogx(v_pp,h2,'o-','LineWidth',2,'MarkerSize',3)
grid
hold
semilogx(v_pp,VAL_h2(1)*ones(1,length(v_pp)),'r-','LineWidth',2);
semilogx(v_pp,VAL_h2(2)*ones(1,length(v_pp)),'r-.','LineWidth',2);
if pa_in==1;
   semilogx(v_pp,VAL_h2(3)*ones(1,length(v_pp)),…
            'r--','LineWidth',2);
end
xlabel('v_p_p (V)')
ylabel('H_I_I')
title('(f)');

figure(3)
clf
subplot(321)
semilogx(v_pp,SI_fa15,'o-','LineWidth',2,'MarkerSize',3)
grid
hold
xlabel('v_p_p (V)')
ylabel('SI (From 1.5\omega_1)')
semilogx(v_pp,VAL_si(1,1)*ones(1,length(v_pp)),…
         'r-','LineWidth',2);
semilogx(v_pp,VAL_si(1,2)*ones(1,length(v_pp)),…
         'r-.','LineWidth',2);
if pa_in==1
   semilogx(v_pp,VAL_si(1,3)*ones(1,length(v_pp)),…
            'r--','LineWidth',2);
   legend('= Indicator','= p_c Threshold','= p_r Threshold',…
          '= p_a_v_g Threshold');
else
   legend('= Indicator','= p_c Threshold','= p_r Threshold');
end
title('(a)');

subplot(322)
semilogx(v_pp,SI_fa2,'o-','LineWidth',2,'MarkerSize',3)
grid
hold
xlabel('v_p_p (V)')
```

```
ylabel('SI (From 2\omega_1)')
semilogx(v_pp,VAL_si(2,1)*ones(1,length(v_pp)),…
         'r-','LineWidth',2);
semilogx(v_pp,VAL_si(2,2)*ones(1,length(v_pp)),…
         'r-.','LineWidth',2);
if pa_in==1;
   semilogx(v_pp,VAL_si(2,3)*ones(1,length(v_pp)),…
            'r--','LineWidth',2);
end
title('(b)');

subplot(323)
semilogx(v_pp,SI_me,'o-','LineWidth',2,'MarkerSize',3)
grid
hold
xlabel('v_p_p (V)')
ylabel('Relative SI')
semilogx(v_pp,VAL_sir(1)*ones(1,length(v_pp)),'r-','LineWidth',2);
semilogx(v_pp,VAL_sir(2)*ones(1,length(v_pp)),…
         'r-.','LineWidth',2);
if pa_in==1;
   semilogx(v_pp,VAL_sir(3)*ones(1,length(v_pp)),…
            'r--','LineWidth',2);
end
title('(c)');

subplot(324)
semilogx(v_pp,Pce,'o-','LineWidth',2,'MarkerSize',3)
hold
grid
semilogx(v_pp,VAL_pfocal(1)*ones(1,length(v_pp)),…
         'r-','LineWidth',2)
xlabel('v_p_p (V)')
ylabel('P_c_e')
title('(d)');

subplot(325)
semilogx(v_pp,Pre,'o-','LineWidth',2,'MarkerSize',3)
hold
grid
semilogx(v_pp,VAL_pfocal(2)*ones(1,length(v_pp)),…
         'r-.','LineWidth',2)
xlabel('v_p_p (V)')
ylabel('P_r_e')
title('(e)');

subplot(326)
```

```
semilogx(v_pp,Pae,'o-','LineWidth',2,'MarkerSize',3)
hold
grid
if pa_in==1
   semilogx(v_pp,VAL_pfocal(3)*ones(1,length(v_pp)),…
            'r--','LineWidth',2)
end
xlabel('v_p_p (V)')
ylabel('P_a_e')
title('(f)');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Plot the threshold values of NI's
figure(4)
clf
bar(VAL_pasym,'c')
hold
grid
errorbar(VAL_pasym,ERR_pasym,'.')
ylabel('Pulse Asymmetry (p_c/p_r)')
text([.8 1.8 2.8],[-.2 -.2 -.2],{'p_c Threshold' 'p_r Threshold'…
     'p_a_v_g Threshold'})

figure(5)
clf
subplot(311)
bar(VAL_sigm,'c')
hold
grid
errorbar(VAL_sigm,ERR_sigm,'.')
ylabel('\sigma_m')

subplot(312)
bar(VAL_sigz(1,:),'c')
hold
grid
errorbar(VAL_sigz,ERR_sigz,'.')
ylabel('\sigma_z')

subplot(313)
subplot(121)
bar(VAL_sigs(1,:),'c')
hold
grid
errorbar(VAL_sigs(1,:),ERR_sigs(1,:),'.')
ylabel('\sigma_s')
title('p_r Based')
```

```
text([.7 1.7 2.7],[-.2*max(VAL_sigs(1,:)) -.2*max(VAL_sigs(1,:))…
      -.2*max(VAL_sigs(1,:))],{'p_c Threshold' 'p_r Threshold'…
       'p_a_v_g Threshold'})

subplot(122)
bar(VAL_sigs(2,:),'c')
hold
grid
errorbar(VAL_sigs(2,:),ERR_sigs(2,:),'.')
ylabel('\sigma_s')
title('p_c/p_r Based')
text([.7 1.7 2.7],[-.2*max(VAL_sigs(2,:)) -.2*max(VAL_sigs(2,:))…
      -.2*max(VAL_sigs(2,:))],{'p_c Threshold' 'p_r Threshold'…
      'p_a_v_g Threshold'})

figure(6)
clf
bar(VAL_h2,'c')
hold
grid
errorbar(VAL_h2,ERR_h2,'.')
ylabel('Second Harmonic Ratio')
text([.8 1.8 2.8],[-.1*max(VAL_h2) -.1*max(VAL_h2) …
      -.1*max(VAL_h2)],{'p_c Threshold' 'p_r Threshold'…
      'p_a_v_g Threshold'})

figure(7)
clf
subplot(211)
bar(VAL_si(1,:),'c')
hold
grid
errorbar(VAL_si(1,:),ERR_si(1,:),'.')
ylabel('SI based on f_a')
title('f_a = 1.5f_o')

subplot(212)
bar(VAL_si(2,:),'c')
hold
grid
errorbar(VAL_si(2,:),ERR_si(2,:),'.')
ylabel('SI based on f_a')
title('f_a = 2f_o')
text([.8 1.8 2.8],[-.2*max(VAL_si(2,:)) -.2*max(VAL_si(2,:)) …
      -.2*max(VAL_si(2,:))],{'p_c Threshold' 'p_r Threshold'…
      'p_a_v_g Threshold'})
```

```
figure(8)
clf
bar(VAL_sir,'c')
hold
grid
errorbar(VAL_sir,ERR_sir,'.')
ylabel('Relative SI')
text([.8 1.8 2.8],[-.1*max(VAL_sir) -.1*max(VAL_sir) …
      -.1*max(VAL_sir)],{'p_c Threshold' 'p_r Threshold'…
      'p_a_v_g Threshold'})

figure(9)
clf
bar(VAL_pfocal,'c')
hold
grid
errorbar(VAL_pfocal,ERR_pfocal,'.')
ylabel('Relative Focal Pressure')
text([.8 1.8 2.8],[-.1*max(VAL_sir) -.1*max(VAL_sir) …
      -.1*max(VAL_sir)],{'p_c Threshold' 'p_r Threshold'…
      'p_a_v_g Threshold'})

figure(10)
clf
plot(volt,v_pp)
grid
xlabel('volt')
ylabel('v_p_p')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Save the results                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sv   = input('Save data: (y/n) ','s');

if sv=='y'
   n=input('Enter filename to save NI data and errors: ','s');
   txtfile=['NI_data/',n,'.txt'];
   fid=fopen(txtfile,'a');

   fprintf(fid,'%2.4f\t%2.4f\t%2.4f\n',VAL_pasym);
   fprintf(fid,'%2.4f\t%2.4f\t%2.4f\n',ERR_pasym);

   fprintf(fid,'%2.4f\t%2.4f\t%2.4f\t%2.4f\t%2.4f\t%2.4f\n',…
           VAL_sigs');
   fprintf(fid,'%2.4f\t%2.4f\t%2.4f\t%2.4f\t%2.4f\t%2.4f\n',…
           ERR_sigs');
   fprintf(fid,'%2.4f\t%2.4f\t%2.4f\n',VAL_sigm');
   fprintf(fid,'%2.4f\t%2.4f\t%2.4f\n',ERR_sigm');

   fprintf(fid,'%2.4f\t%2.4f\t%2.4f\n',VAL_sigz');
   fprintf(fid,'%2.4f\t%2.4f\t%2.4f\n',ERR_sigz');

   fprintf(fid,'%2.4f\t%2.4f\t%2.4f\n',VAL_h2);
   fprintf(fid,'%2.4f\t%2.4f\t%2.4f\n',ERR_h2);

   fprintf(fid,'%2.4f\t%2.4f\t%2.4f\t%2.4f\t%2.4f\t%2.4f\n',…
           VAL_si');
   fprintf(fid,'%2.4f\t%2.4f\t%2.4f\t%2.4f\t%2.4f\t%2.4f\n',…
           ERR_si');

   fprintf(fid,'%2.4f\t%2.4f\t%2.4f\n',VAL_sir);
   fprintf(fid,'%2.4f\t%2.4f\t%2.4f\n',ERR_sir);

   fprintf(fid,'%2.4f\t%2.4f\t%2.4f\n',VAL_pfocal);
   fprintf(fid,'%2.4f\t%2.4f\t%2.4f\n',ERR_pfocal);

   fclose(fid);

end
```

Subfunctions to Find Values of Some Indices

```
function [sigma_s,sigma_sa]=find_sigma_s(pc,pr,F_Ro,alpha,co)

%This is a MATLAB function that solves for sigma_s in order to
%evaluate the nonlinearity of a signal at a given voltage setting.
%Sigma_s is the value of sigma defined by Ostrovskii and Sutin,
%but derived using my evaluation.

%Inputs
%    pc = peak compressional pressure at the focus.
%    pr = peak rarefractional pressure at the focus.
%    F_Ro = F/Ro amplification factor
%    alpha = half-angle of transducer
%    co = small-signal sound speed (Determine from temperature).
%Medium properties (Saved in this file)
%    beta = nonlinear parameter beta=(1+B/2A)
%    rho_o = static density
%Output
%    sigma_s = the calculated non-linear parameter based on the
%              peak rarefractional pressure.
%    sigma_sa = the calculated non-linear parameter based on the
%               asymmetric ratio.
```

```
%***************************************************************

beta=3.5;
rho_o=1000;

%Find term to multiply the pressure

s=pi*beta*log(F_Ro)/(2*(alpha*sin(alpha))*rho_o*co^2);

%Find sigma_s based on pr and G
sigma_s=s*pr/(1-s*pr);

%Find sigma_s based on pulse asymmetry
a=pc/pr;

sigma_sa=(a-1)/(a+1);
```

```
function [sigma_m]=find_sigma_m(pa,f,G,z,co)

%This is a MATLAB function that solves for sigma_m in order to
%evaluate the nonlinearity of a signal at a given voltage setting.

%Inputs
%    pa = peak average pressure at the focus.
%    f = center frequency of transducer.
%    G = transducer gain.
%    z = focal length of the transducer.
%    co = small-signal sound speed (Determined from temperature).
%Medium properties (Saved in this file)
%    beta = nonlinear parameter beta=(1+B/2A)
%    rho_o = static density
%Output
%    sigma_m = the calculated non-linear parameter
%***************************************************************

beta=3.5;
rho_o=1000;

sigma_m=z*pa*2*pi*f*beta*(log(G+sqrt(G^2 - 1))/sqrt(G^2 -...
        1))/(rho_o*co^3);
```

```
function [sigma_z]=find_sigma_z(pa,f,z,co)

%This is a MATLAB function that solves for sigma_z in order to
```

```
%evaluate the nonlinearity of a signal at a given voltage setting.

%Inputs
%    pa = peak average pressure at the focus.
%    f = center frequency of transducer.
%    z = focal length of the transducer.
%    co = small-signal sound speed (Determined from temperature).
%Medium properties (Saved in this file)
%    beta = nonlinear parameter beta=(1+B/2A)
%    rho_o = static density
%Output
%    sigma_z = the calculated non-linear parameter
%**************************************************************

beta=3.5;
rho_o=1000;

sigma_z=z*pa*2*pi*f*beta/(rho_o*co^3);
```

```
function [h2]=find_h2(p,dt);

%This is a MATLAB function that solves for the second harmonic
%ratio in order to evaluate the nonlinearity of a signal at a
%given voltage setting.

%Inputs
%    p = current pressure waveform in time.
%    dt = time increment of waveform
%Output
%    h2 = the second harmonic ratio.

%**************************************************************
%Remove DC
p=p-mean(p);

%Find the measured spectrum
P_true=abs(fftshift(fft(p,2048)));

M=length(P_true);
Pt=P_true((M/2+1):M);

%Find the frequency values
freq=[0:((M/2)-1)]*(1/dt)/(M-1);

%Find maximum frequency in pulse
[P_max1,n]=max(Pt);
```

```
f_max1=freq(n);

%Find second harmonic value
n2=2*n-1;

old_max=Pt(n2);
N=1;
pk=0;

while pk==0
   P=Pt((n2-N):(n2+N));

   f=freq((n2-N):(n2+N))';

   [P_max2,n2p]=max(P);

   if P_max2==old_max
      pk=1;
   end

   old_max=P_max2;
   N=N+1;
end

h2=P_max2/P_max1;
```

```
function [si_15,si_2]=fa_si(p);

%This is a MATLAB function that solves for the SI based on f_a in
%order to evaluate the nonlinearity of a signal at a given voltage
%setting.

%Inputs
%    p = current pressure waveform in time.
%Output
%    si_15 = the spectral index with f_a of 1.5*fundamental.
%    si_2 = the spectral index with f_a of 2*fundamental.

%*****************************************************************
%Remove DC
p=p-mean(p);

%Find the measured spectrum
P_true=abs(fftshift(fft(p,2048)));
```

```
M=length(P_true);
Pt=P_true((M/2+1):M);

%Find maximum frequency in pulse
[P_max,n]=max(Pt);

%Determine the different f_a values to be used.
N_fa15=floor(1.5*n - 1);
N_fa2=2*n;

%Integrate for each fa
P_fa15=Pt(N_fa15:length(Pt));
P_fa2=Pt(N_fa2:length(Pt));

num15=sum(P_fa15);
num2=sum(P_fa2);

den=sum(Pt);

%Find each si
si_15=num15/den;
si_2=num2/den;
```

```
function [si]=my_si(p,p_low,Gv);

%This is a MATLAB function that solves for my version of the SI in
%order to evaluate the nonlinearity of a signal at a given voltage
%setting.

%Inputs
%    p = current pressure waveform in time.
%    p_low = low voltage pressure waveform in time.
%    Gv = current value of the voltage gain
%Output
%    si = my version of the spectral index.

%*****************************************************************
%Find P_low the low voltage spectrum
P_low=abs(fft(p_low));

%Find the measured spectrum
P_true=abs(fft(p));

%Find the estimated spectrum
P_est=P_low*Gv;
```

```
%Find the si
num=sum((P_true-P_est).^2);

den=sum(P_true.^2);

si=num/den;
```

Function Used to Find Threshold Voltages

```
function [diff_err]=find_extrap_factor(volt_extrap,SP,p,...
                                       volt,Npol,lin_err)

%This is a MATLAB function that is used to find the maximum
%allowed value of extrapolation voltage given the acceptable value
%for extrapolation error.

%Inputs
%    volt_extrap = current value of extrapolation voltage
%    SP = Spline curve coefficients to determine pressure values
%         at current voltage.
%    p = Meausred pressure values
%    volt = Values of extrapolation factor at measurement points
%    Npol = Degree of polynomial used to fit error curves.
%    lin_err = acceptable value for extrapolation error.
%Output
%    diff_err = difference between the peak error and allowed
%               error.
%*****************************************************************

p_ref=ppval(SP,volt_extrap);

for ri=1:length(volt)
   %Find Current Extrapolation Factor
   Gv=volt(ri)/volt_extrap;

   %Find Current Error Values
   Pe(ri)=100*abs(p(ri)-Gv*p_ref)/p(ri);
end

%Determine Coefficients of Error Polynomial
Pole=polyfit(volt,Pe,Npol);

%Set voltage values overwhich error will be computed
dv=volt(2)-volt(1); %Set step in voltage.
vlt=0:dv:volt_extrap;
```

```
%Determine the peak error in this range
peak_error=max(polyval(Pole,vlt));

%Determine the difference in the error
diff_err = peak_error-lin_err;
```

Miscellaneous other Subfunctions

```
function [c]=CfromT(Tc)
%SOUNDSPEED
%   This is a function that calculates the speed of sound based on
%   temperature.
%   INPUT:
%      Tc - Temperature of the water in degrees C.
%      plt - Determines if data is to be plotted 'y'.
%   OUTPUT:
%      c - Speed of sound found from water temperature.

%Now calculate c2 based on Tc
T=Tc/100;
c=1402.7 + 488*T - 482*(T^2) + 135*(T^3);
%Equation 5.6.8 from Kinsler with Pg = 0.
```

```
function [fpeak]=find_fpeak(p,time)

%This is a MATLAB function that finds the location of the
%frequency peak.

%Inputs
%    p = pressure waveform.
%    time = corresponding time values
%Output
%    fpeak = location of peak in Hz
%*****************************************************************

%Determine time step.
dt=time(2)-time(1);

%Determine value for each frequency
Ps=abs(fft(p));
M=length(Ps);

%Find corresponding freq. values
d_f1=[0:(M/2-1)]*2*pi/M;
d_f2=[(M/2):(M-1)]*2*pi/M - 2*pi;
```

```
freq=d_f1*(1/dt)/(2*pi);

[m,n]=max(Ps(1:M/2));

fpeak=freq(n);
```