APPENDIX A
MATLAB CODE TO COMPUTE STRUCTURAL RESPONSE

```
clear

global l R to h Rac lac

%This program computes the spatial displacement response of a cylinder
%actuated by 2 PZT actuators.
%This program is based on Fred Lalande's work to compute the response of a
%cylinder to two actuators actuated in phase

%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++
% Program for impedance modeling of in-phase,
% actuation of induced strain actuators.
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++

% Frequency domain definition ================================================
fre=981*2*pi; %2*pi*(500:2.5:900)';
L=ones(length(fre),1);
fHz  = fre/2/pi;
nbit = 16;          % Number of iterations used for the modal expansion
nbmodes = 3;
mperin=1/39.37;

% Resolution of the elements
xelem=12;      %number of elements in x
telem=36;      %number of elements in theta

% Parameters for the shell ===================================================
h   = .25*mperin;             % thickness
R   = 5*mperin-h/2;           % radius of curvature
Rac = R-h/2;                  % Inside Radius
l   = 16*mperin;              % Length of the cylinder
lac = 16*mperin;              % Acoustic length of the cylinder
E   = 64e9*(1+i*0.005);       % Complex Young modulus (Al)
nu  = 0.3;                    % Poisson ratio
rho = 2700;                   % density of aluminium
D   = E*(h^3)/(12*(1-nu^2));    % Bending stiffness
K   = E*h/(1-nu^2);           % Extensional stiffness

% Parameters for the PZT actuators ===========================================
tp = 14*pi/180;               % Actuator tangential size
Rp=R*tp;
```

```
lp = 1.5*mperin;                % Actuator axial size
to = 0*(pi/180);                % Actuator center
xo = (16/4)*mperin;             % Actuator center
hp    = .0095*mperin;           % actuator thickness (top and bottom PZT)
voltage=20/.7071;       % Voltage Amplitude
Eins  = voltage/hp;     % applied electric field on the inside actuator
d_32 = -166e-12;      % PZT coefficient
Ep   = 63e9;          % Young's modulus
nup  = 0.3;           % Poisson ratio
rhop = 7600;          % Density
x1 = xo-lp/2;         % Actuator Coordinates
x2 = xo+lp/2;
t1 = to-tp/2;
t2 = to+tp/2;

%Preliminary calculations for type of actuation ============================
lambdains = Eins*d_32;
lambdamax = lambdains;

% Natural Frequencies and Magnitude Ratios calculations ====================
 for m=1:nbit/2
  for n=1:nbit
   alp = m*pi/l;
   bet = n/R;
   k11 = K*(alp^2 + (1-nu)/2*bet^2);
   k12 = K*(1+nu)/2*alp*bet;
   k13 = K*nu/R*alp;
   k22 = (K + D/R^2)*((1-nu)/2*alp^2 + bet^2);
   k23 = -K/R*bet - D/R*bet*(alp^2 + bet^2);
   k33 = D*(alp^2 + bet^2)^2 + K/R^2;
   a1 = -1/(rho*h)*(k11 + k22 + k33);
   a2 = 1/((rho*h)^2)*(k11*k33 + k22*k33 + k11*k22 - k23^2 - k12^2 - k13^2);
   a3 = 1/((rho*h)^3)*...
        (k11*k23^2 + k22*k13^2 + k33*k12^2 + 2*k12*k23*k13 - k11*k22*k33);
   cte = acos((27*a3 + 2*a1^3 - 9*a1*a2)/(2*sqrt((a1^2 - 3*a2)^3)));
% 1) Bending modes =============
   omega1(n,m) = sqrt(-2/3*sqrt(a1^2 - 3*a2)*cos((cte+0*pi)/3) - a1/3);
   AC1(n,m) = (-k13*(rho*h*omega1(n,m)^2 - k22) + k12*k23)/...
     ((rho*h*omega1(n,m)^2 - k11)*(rho*h*omega1(n,m)^2 - k22) - k12^2);
   BC1(n,m) = (-k23*(rho*h*omega1(n,m)^2 - k11) + k12*k13)/...
     ((rho*h*omega1(n,m)^2 - k11)*(rho*h*omega1(n,m)^2 - k22) - k12^2);
% 2) Torsional modes ===========
   omega2(n,m) = sqrt(-2/3*sqrt(a1^2 - 3*a2)*cos((cte+2*pi)/3) - a1/3);
   AC2(n,m) = (-k13*(rho*h*omega2(n,m)^2 - k22) + k12*k23)/...
     ((rho*h*omega2(n,m)^2 - k11)*(rho*h*omega2(n,m)^2 - k22) - k12^2);
   BC2(n,m) = (-k23*(rho*h*omega2(n,m)^2 - k11) + k12*k13)/...
```

```
    ((rho*h*omega2(n,m)^2 - k11)*(rho*h*omega2(n,m)^2 - k22) - k12^2);
% 3) Axial modes ===============
  omega3(n,m) = sqrt(-2/3*sqrt(a1^2 - 3*a2)*cos((cte+4*pi)/3) - a1/3);
  AC3(n,m) = (-k13*(rho*h*omega3(n,m)^2 - k22) + k12*k23)/...
    ((rho*h*omega3(n,m)^2 - k11)*(rho*h*omega3(n,m)^2 - k22) - k12^2);
  BC3(n,m) = (-k23*(rho*h*omega3(n,m)^2 - k11) + k12*k13)/...
    ((rho*h*omega3(n,m)^2 - k11)*(rho*h*omega3(n,m)^2 - k22) - k12^2);
 end %n
end %m

% n=0 modes
 for m=1:nbit/2
  alp = m*pi/l;
  k11 = K*alp^2;
  k13 = K*nu/R*alp;
  k22 = (K + D/R^2)*(1-nu)/2*alp^2;
  k33 = D*alp^4 + K/R^2;
  a1 = -1/(rho*h)*(k11 + k22 + k33);
  a2 = 1/((rho*h)^2)*(k11*k33 + k22*k33 + k11*k22 - k13^2);
  a3 = 1/((rho*h)^3)*(k22*k13^2 - k11*k22*k33);
  cte = acos((27*a3 + 2*a1^3 - 9*a1*a2)/(2*sqrt((a1^2 - 3*a2)^3)));
% 1) Bending modes =============
  omega10(m) = sqrt(-2/3*sqrt(a1^2 - 3*a2)*cos((cte+0*pi)/3) - a1/3);
  AC10(m) = (-k13*(rho*h*omega10(m)^2 - k22))/...
    ((rho*h*omega10(m)^2 - k11)*(rho*h*omega10(m)^2 - k22));
% 2) Torsional modes ===========
  omega20(m) = sqrt(-2/3*sqrt(a1^2 - 3*a2)*cos((cte+2*pi)/3) - a1/3);
  AC20(m) = (-k13*(rho*h*omega20(m)^2 - k22))/...
    ((rho*h*omega20(m)^2 - k11)*(rho*h*omega20(m)^2 - k22));
% 3) Axial modes ===============
  omega30(m) = sqrt(-2/3*sqrt(a1^2 - 3*a2)*cos((cte+4*pi)/3) - a1/3);
  AC30(m) = (-k13*(rho*h*omega30(m)^2 - k22))/...
    ((rho*h*omega30(m)^2 - k11)*(rho*h*omega30(m)^2 - k22));
 end %m
 clear k11 k12 k13 k22 k23 k33 a1 a2 a3 cte alp bet c K D

%Admittance                                                    calculations
==================================================
  Hxx = 0;  Hxt = 0;  Hxr = 0;
  Htx = 0;  Htt = 0;  Htr = 0;
  Hrx = 0;  Hrt = 0;  Hrr = 0;
  for qq=1:nbmodes
   if qq ==1
    omega = omega1;
    omega0 = omega10;
    AC = AC1;
```

```
    AC0 = AC10;
    BC = BC1;
   elseif qq == 2
    omega = omega2;
    omega0 = omega20;
    AC = AC2;
    AC0 = AC20;
    BC = BC2;
   else
    omega = omega3;
    omega0 = omega30;
    AC = AC3;
    AC0 = AC30;
    BC = BC3;
   end %if

% n=0 modes ====================
   for m = 1:nbit/2
     alp = m*pi/l;
     Cx = cos(alp*x1) - cos(alp*x2);
     Sx = sin(alp*xo);
     N0in = R*l*pi*(AC0(m)^2+1);
     factor0 = -i*fre*R/(rho*h)./(omega0(m)^2*L-fre.^2);
     Hxx = Hxx + (2*AC0(m)^2/N0in)*(t2-t1)*Cx^2*factor0/Rp;
     Hrx = Hrx + (-2/(R*alp)*AC0(m)/N0in)*(t2-t1)*Cx^2*factor0/lp;
     Hxr = Hxr + (2*AC0(m)/N0in)*(t2-t1)^2*Cx*Sx*factor0/Rp;
     Hrr = Hrr + (-2/(R*alp)/N0in)*(t2-t1)^2*Cx*Sx*factor0/lp;
   end %m
% n,m>0 modes ====================
   for m = 1:nbit/2
    for n = 1:nbit
     alp = m*pi/l;
     Cx = cos(alp*x1) - cos(alp*x2);
     Sx = sin(alp*xo);
     St = sin(n*t1) - sin(n*t2);
     Ct = cos(n*t1) - cos(n*t2);
     SCt = St*cos(n*to) - Ct*sin(n*to);
     Nin  = R*pi*l/2*(AC(n,m)^2 + BC(n,m)^2 + 1);
     factor = -i*fre*R/(rho*h)./(omega(n,m)^2*L-fre.^2);
     Hxx = Hxx + (2/n*AC(n,m)^2/Nin)*SCt*Cx^2*factor/Rp;
     Htx = Htx + (-2/(R*alp)*AC(n,m)*BC(n,m)/Nin)*SCt*Cx^2*factor/lp;
     Hrx = Hrx + (-2/(R*alp)/n*AC(n,m)/Nin)*SCt*Cx^2*factor/lp;
     Hxt = Hxt + (2/n*AC(n,m)*BC(n,m)/Nin)*(St^2+Ct^2)*Cx*Sx*factor/Rp;
     Htt = Htt + (-2/(R*alp)*BC(n,m)^2/Nin)*(St^2+Ct^2)*Cx*Sx*factor/lp;
     Hrt = Hrt + (-2/(R*alp)*BC(n,m)/n/Nin)*(St^2+Ct^2)*Cx*Sx*factor/lp;
     Hxr = Hxr + (2/n^2*AC(n,m)/Nin)*(St^2+Ct^2)*Cx*Sx*factor/Rp;
```

```
    Htr = Htr + (-2/(R*alp)*BC(n,m)/n/Nin)*(St^2+Ct^2)*Cx*Sx*factor/lp;
    Hrr = Hrr + (-2/(R*alp)/n^2/Nin)*(St^2+Ct^2)*Cx*Sx*factor/lp;
   end %n
  end %m
 end %qq

%                                   Impedance                                   Calculation
==================================================
Det=Hxx.*(Htt+Hrt)-Hxt.*(Htx+Hxr);
Ztt=Hxx./Det;
Zxt=-(Htx+Hxr)./Det;
Ztx=-Hxt./Det;
Zxx=(Htt+Htr)./Det;
clear Det

% Calculate impedance of PZT patch ====================================
k=sqrt(rhop/Ep)*fre;
Zaxx=-i*Ep*Rp*hp*k./(fre.*tan(k*lp));
Zatt=-i*Ep*lp*hp*k./(fre.*tan(k*Rp));



%Admittance Plots============================
%semilogy(fHz,abs(Hxx),'-',fHz,abs(Htt),':',...
%       fHz,abs(Htx),'--',fHz,abs(Hrx),'-.',fHz,abs(Hrt),'-.')
%xlabel('Frequency  (Hz)')
%ylabel('Admittance Magnitude  (m/N/s)')
%pause
%Impedance Plots============================
%semilogy(fHz,abs(Zxx),'-',fHz,abs(Ztt),':',...
%fHz,abs(Ztx),'--',fHz,abs(Zxt),'-.',fHz,abs(Zaxx),'-',fHz,abs(Zatt),':')
%xlabel('Frequency  (Hz)')
%ylabel('Impedance Magnitude  (N.s/m)')
%pause



% Actuator force calculations ============================================
M11=k.*cos(k*lp).*(L+Zxx./Zaxx-nup*(Rp/lp)*Ztx./Zaxx);
M12=k.*cos(k*Rp).*((lp/Rp)*Zxt./Zatt-nup*Ztt./Zatt);
M21=k.*cos(k*lp).*((Rp/lp)*Ztx./Zaxx-nup*Zxx./Zaxx);
M22=k.*cos(k*Rp).*(L+Ztt./Zatt-nup*(lp/Rp)*Zxt./Zatt);
Det=M11.*M22-M12.*M21;
A=(M22-M12)*lambdamax./Det;
C=(M11-M21)*lambdamax./Det;
Fxx=-i*fre.*(A.*Zxx.*sin(k*lp)+C.*Zxt.*sin(k*Rp));
```

```
Ftt=-i*fre.*(A.*Ztx.*sin(k*lp)+C.*Ztt.*sin(k*Rp));
Neq = -2*E*h*lambdamax/(1-nu)/(2+E*h/(Ep*hp))*L;
Nla = -2*(Ep*hp/(1-nup))*lambdamax*L;

Nx = 2*Fxx/Rp;
Nt = 2*Ftt/lp;
clear M11 M12 M21 M22 A C Det k factor factor0 Nin SCt Sx Cx St Ct N0in

%Force Plots==================================
%semilogy(fHz,abs(Nx),'-',fHz,abs(Nt),':',fHz,abs(Nla),':')
%xlabel('Frequency  (Hz)')
%ylabel('Actuator Output Force Magnitude (N/M)')
%pause

% Varying the Location of computations ==============================
for ii=1:1:(xelem+1)
 xc=l*(ii-1)/xelem;
 x(ii)=xc;
 ii
 for jj=1:1:(telem+1)
  tc=(jj-1)*(360/telem)*pi/180;
   if ii==1
    t(jj)=tc*180/pi;
   end %stores the t vector only once!
%Displacements                                                    calculation
================================================
 u = 0; v = 0; w = 0;
 usta=0; vsta=0; wsta=0;
  for qq=1:nbmodes
   if qq ==1
    omega = omega1;
    omega0 = omega10;
    AC = AC1;
    AC0 = AC10;
    BC = BC1;
   elseif qq == 2
    omega = omega2;
    omega0 = omega20;
    AC = AC2;
    AC0 = AC20;
    BC = BC2;
   else
    omega = omega3;
    omega0 = omega30;
    AC = AC3;
    AC0 = AC30;
```

```
    BC = BC3;
   end %if
% n=0 modes ======================
  for m = 1:nbit/2
     alp = m*pi/l;
     Cx = cos(alp*x1) - cos(alp*x2);
     Nin = R*l*pi*(1+AC0(m)^2);
     d1 =  AC0(m)*Nx - Nt/(R*alp);
     factor = R/(rho*h)/Nin./(omega0(m)^2*L-fre.^2).*d1;
     u = u + AC0(m)*factor*(t2-t1)*Cx*cos(alp*xc);
     w = w + factor*(t2-t1)*Cx*sin(alp*xc);
     %d1sta =  (AC0(m) - 1/(R*alp))*Nla;
     %factorsta = R/(rho*h)/Nin./(omega0(m)^2*L-fre.^2).*d1sta;
     %usta = usta + AC0(m)*factorsta*(t2-t1)*Cx*cos(alp*xc);
     %wsta = wsta + factorsta*(t2-t1)*Cx*sin(alp*xc);
  end %m

% n,m>0 modes ====================
 for m = 1:nbit/2
  for n = 1:nbit
    alp = m*pi/l;
    Cx = cos(alp*x1) - cos(alp*x2);
    St = sin(n*t1) - sin(n*t2);
    Ct = cos(n*t1) - cos(n*t2);
    c1 = St*cos(n*tc) - Ct*sin(n*tc);
    c2 = St*sin(n*tc) + Ct*cos(n*tc);
    Nin = R*l*pi/2*(AC(n,m)^2 + BC(n,m)^2 + 1);
    d1 = -AC(n,m)*Nx/n +BC(n,m)*Nt/(R*alp) +Nt/(R*alp*n);
    factor = R/(rho*h)/Nin./(omega(n,m)^2*L-fre.^2).*d1;
    u = u + AC(n,m)*factor*c1*Cx*cos(alp*xc);
    v = v + BC(n,m)*factor*c2*Cx*sin(alp*xc);
    w = w + factor*c1*Cx*sin(alp*xc);
    %d1sta = (-AC(n,m)/n +BC(n,m)/(R*alp) +1/(R*alp*n))*Nla;
    %factorsta = R/(rho*h)/Nin./(omega(n,m)^2*L-fre.^2).*d1sta;
    %usta = usta + AC(n,m)*factorsta*c1*Cx*cos(alp*xc);
    %vsta = vsta + BC(n,m)*factorsta*c2*Cx*sin(alp*xc);
    %wsta = wsta + factorsta*c1*Cx*sin(alp*xc);
  end %n
 end %m
end %qq
W(ii,jj)=w;
end %jj, theta
end %ii, x

%Compute the acceleration vector
A=(-fre^2).*W;
```

```
clear factorsta factor qq Nin c1 c2 d1 d1sta Cx Ct St AC AC0 BC omega omega0 m n

%Displacements Plots==========================

%Need to Correct the sign of the displacement

signcorr=sign(real(W));

%corrimag=sign(imag(W));
%corrreal=sign(real(W));

%for ii=1:1:(xelem+1)
% for jj=1:1:(telem+1)

%  if corrimag(ii,jj)==1
%   signcorr(ii,jj)=1;
%  elseif corrreal(ii,jj)==1
%   signcorr(ii,jj)=1;
%  else
%   signcorr(ii,jj)=-1;
%  end %if

% end %jj
%end %ii


%_____________________________________________________________________
surf(t,(1/mperin)*x,signcorr.*abs(W)),grid
title('Phase Adjusted Cylinder Response')
xlabel('Angular Position, Theta [deg.]')
ylabel('Axial Position, x [m]')
zlabel('Displacement Magnitude [m]')
view([15,45])
%_____________________________________________________________________

%semilogy(fHz,abs(u),'-',fHz,abs(v),':',fHz,abs(w),'--',fHz,abs(usta),'-
',fHz,abs(vsta),':',fHz,abs(wsta),'--')
%xlabel('Frequency  (Hz)')
%ylabel('Displacement Magnitude  (m)')
```