APPENDIX D:
OVERVIEW OF SIMULATOR USED TO FIND BACKSCATTERED DATA

In Chapter 2, the expected backscattered voltage was directly related to the properties of the scatterers contained within some scattering region. In order to explore the scattering properties based on this model, a computer can be used to simulate the backscatter received by the transducer. The development of such a simulator is the focus of this Appendix. The simulator will first find the voltage resulting from a single scatterer at an arbitrary location in space. The voltage from multiple scatterers will then be found by randomly positioning scatterers throughout the medium and summing the signals from each scatterer.

Derivation of Simulation Equations:

The received voltage from Equation (2.39) for a single spherical shell at some location $\vec{r}_n$ with radius a_{eff} is given by

$$V_{refl}(\omega) \cong \frac{-2i\tilde{k}^3 V_{inc}(\omega)\Psi_o(\tilde{k})\Phi_o(\tilde{k})H^2(\omega)}{(4\pi)^2 S_T}, \tag{D.1}$$

where

$$\Phi_o(\tilde{k}) = e^{i2\tilde{k}z_T} e^{2id\tilde{k}_1 z_1} \iiint_{V'} d\vec{r}' \left(a_{eff}\delta\left(\left|\vec{r}' - \vec{r}_n\right| - a_{eff}\right) \left(G_o e^{-\left(\left(\frac{x'}{w_x}\right)^2 + \left(\frac{y'}{w_y}\right)^2 + \left(\frac{z'}{w_z}\right)^2\right)} \right)^2 e^{-i2\tilde{k}_1 z'} \right)$$

$$= e^{i2\tilde{k}z_T} e^{2id\tilde{k}_1 z_1} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} dx'dy'dz' \left(a_{eff}\delta\left(\left|\vec{r}'\right| - a_{eff}\right) \left(G_o e^{-\left(\left(\frac{x'+x_n}{w_x}\right)^2 + \left(\frac{y'+y_n}{w_y}\right)^2 + \left(\frac{z'+z_n}{w_z}\right)^2\right)} \right)^2 e^{-i2\tilde{k}_1(z'+z_n)} \right)$$

$$= e^{i2\tilde{k}z_T} e^{2id\tilde{k}_1 z_1} \int_0^{\infty}\int_0^{\pi}\int_0^{2\pi} r'^2 \sin(\theta')dr'd\theta'd\phi' \left(\begin{array}{l} a_{eff} e^{-i2\tilde{k}_1(r'\cos(\theta')+z_n)} \delta\left(r' - a_{eff}\right) \\ \left(G_o e^{-\left(\left(\frac{r'\sin(\theta')\cos(\phi')+x_n}{w_x}\right)^2 + \left(\frac{r'\sin(\theta')\sin(\phi')+y_n}{w_y}\right)^2 + \left(\frac{r'\cos(\theta')+z_n}{w_z}\right)^2\right)} \right)^2 \end{array} \right)$$

$$= a_{eff}^3 G_o^2 e^{i2\tilde{k}z_T} e^{2id\tilde{k}_1 z_1} \int_0^{\pi} \sin(\theta') e^{-i2\tilde{k}_1(r'\cos(\theta')+z_n)} d\theta'$$

$$\int_0^{2\pi} d\phi' \left(e^{-2\left(\left(\frac{a_{eff}\sin(\theta')\cos(\phi')+x_n}{w_x}\right)^2+\left(\frac{a_{eff}\sin(\theta')\sin(\phi')+y_n}{w_y}\right)^2+\left(\frac{a_{eff}\cos(\theta')+z_n}{w_z}\right)^2\right)} \right). \tag{D.2}$$

Assuming that the source is circularly symmetric so that $w_x = w_y$, then Equation (D.2) becomes

$$\Phi_o(\tilde{k}) = a_{eff}^3 G_o^2 e^{i2\tilde{k}z_T} e^{2id\tilde{k}_1 z_1} \int_0^{\pi} \sin(\theta') e^{-i2\tilde{k}_1(a_{eff}\cos(\theta')+z_n)} e^{-2\left(\frac{a_{eff}\cos(\theta')+z_n}{w_z}\right)^2} d\theta'$$

$$\int_0^{2\pi} d\phi' \left(e^{\frac{-2}{w_x^2}\left(\left(a_{eff}\sin(\theta')\cos(\phi')+x_n\right)^2+\left(a_{eff}\sin(\theta')\sin(\phi')+y_n\right)^2\right)} \right)$$

$$= a_{eff}^3 G_o^2 e^{\frac{-2}{w_x^2}\left(x_n^2+y_n^2\right)} e^{i2\tilde{k}z_T} e^{2id\tilde{k}_1 z_1} \int_0^{\pi} \sin(\theta') e^{-i2\tilde{k}_1(a_{eff}\cos(\theta')+z_n)} e^{\frac{-2}{w_x^2}\left(a_{eff}^2\sin^2(\theta')\right)} e^{-2\left(\frac{a_{eff}\cos(\theta')+z_n}{w_z}\right)^2} d\theta'$$

$$\int_0^{2\pi} d\phi' \left(e^{\frac{-4a_{eff}}{w_x^2}\sin(\theta')\left(x_n\cos(\phi')+y_n\sin(\phi')\right)} \right)$$

$$= a_{eff}^3 G_o^2 e^{\frac{-2}{w_x^2}\left(x_n^2+y_n^2\right)} e^{i2\tilde{k}z_T} e^{2id\tilde{k}_1 z_1} \int_0^{\pi} \sin(\theta') e^{-i2\tilde{k}_1(a_{eff}\cos(\theta')+z_n)} e^{\frac{-2}{w_x^2}\left(a_{eff}^2\sin^2(\theta')\right)} e^{-2\left(\frac{a_{eff}\cos(\theta')+z_n}{w_z}\right)^2} d\theta'$$

$$\int_0^{2\pi} d\phi' \left(e^{\frac{-4a_{eff}\sqrt{x_n^2+y_n^2}}{w_x^2}\sin(\theta')\left(\cos(\phi_n)\cos(\phi')+\sin(\phi_n)\sin(\phi')\right)} \right)$$

$$= a_{eff}^3 G_o^2 e^{\frac{-2}{w_x^2}\left(x_n^2+y_n^2\right)} e^{i2\tilde{k}z_T} e^{2id\tilde{k}_1 z_1} \int_0^{\pi} \sin(\theta') e^{-i2\tilde{k}_1(a_{eff}\cos(\theta')+z_n)} e^{\frac{-2}{w_x^2}\left(a_{eff}^2\sin^2(\theta')\right)} e^{-2\left(\frac{a_{eff}\cos(\theta')+z_n}{w_z}\right)^2} d\theta'$$

$$\int_0^{2\pi} d\phi' \left(e^{i\frac{i4a_{eff}\sqrt{x_n^2+y_n^2}}{w_x^2}\sin(\theta')\cos(\phi'-\phi_n)} \right)$$

$$= a_{eff}^3 G_o^2 e^{\frac{-2}{w_x^2}\left(x_n^2+y_n^2\right)} e^{i2\tilde{k}z_T} e^{2id\tilde{k}_1 z_1} \int_0^{\pi} \sin(\theta') e^{-i2\tilde{k}_1(a_{eff}\cos(\theta')+z_n)} e^{\frac{-2}{w_x^2}\left(a_{eff}^2\sin^2(\theta')\right)} e^{-2\left(\frac{a_{eff}\cos(\theta')+z_n}{w_z}\right)^2} d\theta'$$

$$J_o\left(\frac{i4a_{eff}\sqrt{x_n^2+y_n^2}}{w_x^2}\sin(\theta')\right)$$

$$= a_{eff}^3 G_o^2 e^{-\frac{2}{w_x^2}\left(x_n^2+y_n^2\right)-\frac{2z_n^2}{w_z^2}} e^{2id\tilde{k}_1 z_1} e^{i2\tilde{k}z_T - i2\tilde{k}_1 z_n} \int_0^{\pi} \begin{pmatrix} \sin(\theta') e^{-i2\tilde{k}_1 a_{eff}\cos(\theta')} e^{\frac{-2}{w_x^2}a_{eff}^2\sin^2(\theta')} e^{\frac{-2}{w_z^2}\left(a_{eff}^2\cos^2(\theta')+2a_{eff}z_n\cos(\theta')\right)} \\ J_o\left(\frac{i4a_{eff}\sqrt{x_n^2+y_n^2}}{w_x^2}\sin(\theta')\right) \end{pmatrix} d\theta'. \tag{D.3}$$

Now perform the substitution of letting $s_\theta = \cos(\theta')$, so that Equation (D.3) becomes

$$\begin{aligned} \Phi_o\left(\tilde{k}\right) &= a_{eff}^3 G_o^2 e^{-\frac{2}{w_x^2}\left(x_n^2+y_n^2\right)-\frac{2z_n^2}{w_z^2}} e^{2id\tilde{k}_1 z_1} e^{i2\tilde{k}z_T - i2\tilde{k}_1 z_n} \int_{-1}^{1} \begin{pmatrix} e^{-i2\tilde{k}_1 a_{eff} s_\theta} e^{\frac{-2}{w_x^2}a_{eff}^2\left(1-s_\theta^2\right)} e^{\frac{-2}{w_z^2}\left(a_{eff}^2 s_\theta^2+2a_{eff}z_n s_\theta\right)} \\ J_o\left(\frac{i4a_{eff}\sqrt{x_n^2+y_n^2}}{w_x^2}\sqrt{1-s_\theta^2}\right) \end{pmatrix} ds_\theta \\ &= a_{eff}^3 G_o^2 e^{-\frac{2}{w_x^2}\left(x_n^2+y_n^2\right)-\frac{2z_n^2}{w_z^2}} e^{2id\tilde{k}_1 z_1} e^{i2\tilde{k}z_T - i2\tilde{k}_1 z_n} \int_0^{1} \begin{pmatrix} e^{\frac{-2}{w_x^2}a_{eff}^2\left(1-s_\theta^2\right)} e^{\frac{-2a_{eff}^2 s_\theta^2}{w_z^2}} J_o\left(\frac{i4a_{eff}\sqrt{x_n^2+y_n^2}}{w_x^2}\sqrt{1-s_\theta^2}\right) \\ \left(e^{-i2\tilde{k}_1 a_{eff} s_\theta} e^{\frac{-4}{w_z^2}a_{eff}z_n s_\theta} + e^{i2\tilde{k}_1 a_{eff} s_\theta} e^{\frac{4}{w_z^2}a_{eff}z_n s_\theta}\right) \end{pmatrix} ds_\theta. \end{aligned} \tag{D.4}$$

In order to integrate Equation (D.4), assume that the scatterers are also small compared to the width of the beam in the *x, y,* and *z* directions. In effect, this is the same as assuming the field is constant across the surface of the scatterer. Then, to a first-order approximation, the integral becomes

$$\begin{aligned} \Phi_o\left(\tilde{k}\right) &\cong a_{eff}^3 G_o^2 e^{-\frac{2}{w_x^2}\left(x_n^2+y_n^2\right)-\frac{2z_n^2}{w_z^2}} e^{2id\tilde{k}_1 z_1} e^{i2\tilde{k}z_T - i2\tilde{k}_1 z_n} \int_0^1 \left(e^{-i2\tilde{k}_1 a_{eff} s_\theta} + e^{i2\tilde{k}_1 a_{eff} s_\theta}\right) ds_\theta \\ &\cong 2a_{eff}^3 G_o^2 e^{-\frac{2}{w_x^2}\left(x_n^2+y_n^2\right)-\frac{2z_n^2}{w_z^2}} e^{2id\tilde{k}_1 z_1} e^{i2\tilde{k}z_T - i2\tilde{k}_1 z_n} \int_0^1 \cos\left(2\tilde{k}_1 a_{eff} s_\theta\right) ds_\theta \\ &\cong 2a_{eff}^3 G_o^2 e^{-\frac{2}{w_x^2}\left(x_n^2+y_n^2\right)-\frac{2z_n^2}{w_z^2}} e^{2id\tilde{k}_1 z_1} e^{i2\tilde{k}z_T - i2\tilde{k}_1 z_n} \frac{\sin\left(2\tilde{k}_1 a_{eff}\right)}{2\tilde{k}_1 a_{eff}}, \end{aligned} \tag{D.5}$$

which is equivalent to the form factor for shells given by *Insana et al.* [1990].

Likewise, the scattering resulting from a single Gaussian scatterer can also be simulated. In this case, the expression for Φ_o would be given by

$$
\begin{aligned}
\Phi_o\left(\tilde{k}\right) &= e^{2id\tilde{k}_1 z_1} e^{i2\tilde{k}z_T} \iiint_{V'} d\vec{r}' \left(\gamma_{\max} e^{-\frac{\left|\vec{r}'-\vec{r}_n\right|^2}{d^2}} \left(G_o e^{-\left(\left(\frac{x'}{w_x}\right)^2+\left(\frac{y'}{w_y}\right)^2+\left(\frac{z'}{w_z}\right)^2\right)} \right)^2 e^{-i2\tilde{k}_1 z'} \right) \\
&= e^{2id\tilde{k}_1 z_1} e^{i2\tilde{k}z_T} \iiint_{V'} d\vec{r}' \left(\gamma_{\max} e^{-\frac{r'^2}{d^2}} \left(G_o e^{-\left(\left(\frac{x'+x_n}{w_x}\right)^2+\left(\frac{y'+y_n}{w_y}\right)^2+\left(\frac{z'+z_n}{w_z}\right)^2\right)} \right)^2 e^{-i2\tilde{k}_1 z'} \right) \\
&= \gamma_{\max} e^{2id\tilde{k}_1 z_1} e^{i2\tilde{k}z_T} \int_0^\infty \int_0^\pi \int_0^{2\pi} \left(\begin{array}{l} r'^2 \sin(\theta') dr' d\theta' d\phi' e^{-\frac{r'^2}{d^2}} e^{-i2\tilde{k}_1\left(r'\cos(\theta')+z_n\right)} \\ \left(G_o e^{-\left(\left(\frac{r'\sin(\theta')\cos(\phi')+x_n}{w_x}\right)^2+\left(\frac{r'\sin(\theta')\sin(\phi')+y_n}{w_x}\right)^2+\left(\frac{r'\cos(\theta')+z_n}{w_z}\right)^2\right)} \right)^2 \end{array} \right) \\
&= \gamma_{\max} G_o^2 e^{-\frac{2}{w_x^2}\left(x_n^2+y_n^2\right)} e^{2id\tilde{k}_1 z_1} e^{i2\tilde{k}z_T - i2\tilde{k}_1 z_n} \int_0^\infty dr' \left(\begin{array}{l} r'^2 e^{-\frac{r'^2}{d^2}} \\ \int_0^\pi d\theta' \left(\begin{array}{l} \sin(\theta') J_o\left(\frac{i4r'\sqrt{x_n^2+y_n^2}}{w_x^2} \sin(\theta') \right) \\ e^{-i2\tilde{k}_1 r'\cos(\theta') - 2\left(\frac{r'\cos(\theta')+z_n}{w_z}\right)^2 - \frac{2r'^2}{w_x^2}\sin^2(\theta')} \end{array} \right) \end{array} \right) \\
&= \gamma_{\max} G_o^2 e^{-\frac{2}{w_x^2}\left(x_n^2+y_n^2\right) - \frac{2z_n^2}{w_z^2}} e^{2id\tilde{k}_1 z_1} e^{i2\tilde{k}z_T - i2\tilde{k}_1 z_n} \int_0^\infty dr' \left(\begin{array}{l} r'^2 e^{-\frac{r'^2}{d^2}} \\ \int_0^\pi d\theta' \left(\begin{array}{l} \sin(\theta') J_o\left(\frac{i4r'\sqrt{x_n^2+y_n^2}}{w_x^2} \sin(\theta') \right) \\ e^{-i2\tilde{k}_1 r'\cos(\theta') - \frac{2}{w_z^2}\left(r'^2\cos^2(\theta')+2r'z_n\cos(\theta')\right) - \frac{2r'^2}{w_x^2}\sin^2(\theta')} \end{array} \right) \end{array} \right) \\
&\cong \gamma_{\max} G_o^2 e^{-\frac{2}{w_x^2}\left(x_n^2+y_n^2\right) - \frac{2z_n^2}{w_z^2}} e^{2id\tilde{k}_1 z_1} e^{i2\tilde{k}z_T - i2\tilde{k}_1 z_n} \int_0^\infty dr' \left(r'^2 e^{-\frac{r'^2}{d^2}} \frac{\sin\left(2\tilde{k}_1 r'\right)}{2\tilde{k}_1 r'} \right) \\
&= \left(\frac{\gamma_{\max} G_0^2}{2\tilde{k}_1} \right) e^{-\frac{2}{w_x^2}\left(x_n^2+y_n^2\right) - \frac{2z_n^2}{w_z^2}} e^{2id\tilde{k}_1 z_1} e^{i2\tilde{k}z_T - i2\tilde{k}_1 z_n} \int_0^\infty dr' \left(r' \sin\left(2\tilde{k}_1 r'\right) e^{-\frac{r'^2}{d^2}} \right),
\end{aligned}
\tag{D.6}
$$

where d is some characteristic length describing the Gaussian impedance distribution ($d \cong 0.643 a_{eff}$) [*Insana et al.*, 1990]. The integral in Equation (D.6) can then be found by integrating by parts to yield

$$\int_0^\infty dr'\left(r'\sin\left(2\tilde{k}_1 r'\right)e^{-\frac{r'^2}{d^2}}\right) = -\frac{d^2}{2}\int_0^\infty \sin\left(2\tilde{k}_1 r'\right)\left(-\frac{2r'}{d^2}e^{-\frac{r'^2}{d^2}}dr'\right)$$
$$= -\frac{d^2}{2}\left(\sin\left(2\tilde{k}_1 r'\right)e^{-\frac{r'^2}{d^2}}\Bigg|_0^\infty + 2\tilde{k}_1\int_0^\infty \cos\left(2\tilde{k}_1 r'\right)e^{-\frac{r'^2}{d^2}}dr'\right)$$
$$= -\frac{d^2\tilde{k}_1}{2}\left(2\int_0^\infty \cos\left(2\tilde{k}_1 r'\right)e^{-\frac{r'^2}{d^2}}dr'\right)$$
$$= -\frac{d^3\tilde{k}_1\sqrt{\pi}}{2}e^{-\tilde{k}_1^2 d^2}. \tag{D.7}$$

Hence, Equation (D.6) becomes

$$\Phi_o\left(\tilde{k}\right) = -\left(\frac{\gamma_{\max}G_0^2}{2\tilde{k}_1}\right)e^{-\frac{2}{w_x^2}\left(x_n^2+y_n^2\right)-\frac{2z_n^2}{w_z^2}}e^{2id\tilde{k}_1 z_1}e^{i2\tilde{k}z_T - i2\tilde{k}_1 z_n}\frac{d^3\tilde{k}_1\sqrt{\pi}}{2}e^{-\tilde{k}_1^2 d^2}$$
$$= -(0.1178)\gamma_{\max}a_{eff}^3 G_0^2 e^{-\frac{2}{w_x^2}\left(x_n^2+y_n^2\right)-\frac{2z_n^2}{w_z^2}}e^{2id\tilde{k}_1 z_1}e^{i2\tilde{k}z_T - i2\tilde{k}_1 z_n}e^{-0.4136\tilde{k}_1^2 a_{eff}^2} \tag{D.8}$$

which is also equivalent to the form factor for Gaussian spheres given by *Insana et al.* [1990].

Now that an expression for the scattering from a single scatterer has been obtained, either shell or Gaussian, scattering from multiple randomly oriented scatterers of various sizes can be found by simply adding up the contribution from each of the scatterers. Hence, for $\bar{n}$ scatterers per unit volume, the total scattered field at any given frequency would be given by

$$V_{refl}(\omega) \cong \frac{-2i\tilde{k}^3 V_{inc}(\omega)\Psi_o\left(\tilde{k}\right)H^2(\omega)}{(4\pi)^2 S_T}\sum_{n=1}^{N}\Phi_o\left(\tilde{k};a_{eff},x_n,y_n,z_n\right), \tag{D.9}$$

where

$$N = \bar{n}\cdot\left(10w_{xo}\right)\left(10w_{yo}\right)(16\,\text{mm})$$
$$x_n = -5w_{xo} + 10w_{xo}\chi_x$$
$$y_n = -5w_{yo} + 10w_{yo}\chi_y$$
$$z_n = -8\,\text{mm} + (16\,\text{mm})\chi_z \tag{D.10}$$

after limiting the scatterers to a reasonable number near the focus with $w_{xo,yo}$ being the Gaussian beamwidths at the center frequency for the transducer. $\chi_{x,y,z}$ are random variables satisfying a uniform distribution with possible values ranging from zero to one. Scatterers were only positioned to within 8 mm of the focus along the beam axis to improve computation time, and the simulation studies were always restricted to this region. However, for some of the simulations, Equation (D.10) was modified as

$$\begin{aligned} N &= \bar{n} \cdot (30w_{xo})(30w_{yo})(16\,\text{mm}) \\ x_n &= -15w_{xo} + 30w_{xo}\chi_x \\ y_n &= -15w_{yo} + 30w_{yo}\chi_y \\ z_n &= -8\,\text{mm} + (16\,\text{mm})\chi_z \end{aligned} \tag{D.11}$$

in order to include scatterers in a larger region about the focus.

Before concluding, the generation of the simulated waveforms is based on several approximations rather than a complete physics-capturing model in order to reduce the computational complexity (i.e., ~ 20 as compared to ~ $1\text{x}10^6$ operations per scatterer at each frequency assuming $\lambda/20$ gird spacing for a scatterer of radius $\lambda/4$ and order N^2 on the matrix inversion for the complete solution). These approximations should be justified. First, the simulator neglects multiple scattering, however, this approximation was shown to be valid by comparing the experimental phantom results to the simulation results in Chapter 3. Second, the simulator approximates the real velocity field from the focused source as a three-dimensional Gaussian distribution rather than solving for the complete field exactly. Once again, the validation of this approximation is demonstrated in Appendix E. Third, the generated waveforms do not consider the generation of shear waves (i.e., mode conversions) at the surface of the scatterer.

Shear waves are not significant in soft biological tissue at a macroscopic scale [*Haken et al.*, 1992]. However, shear waves may modify the scattering from individual scatterers at which point the simplified simulator presented in this Appendix would need to be modified. Namely, the calculation of the scattering from a single scatterer Φ_o would need to be modified to include shear waves. For simple scatterers, an expression for the scattering could still be solved in closed form, but the more complicated scatterers would require a numerical method (i.e., finite element or possibly method of moments) to find the scattering. Multiple scattering, however, would still not need to be included.

Matlab Code Used to Generate Received Voltage for Gaussian Scatterers:

```
clear all;
close all;
```

```
i=sqrt(-1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%BACKSCATTER SIMULATION                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%This is a program that computes the acoustic signal   %
%scattered from a tissue region buried beneath N-layers%
%of distinct tissue types.  The simulation neglects    %
%multiple scattering and assumes that the Born         %
%approximation is valid.  Region 1 is the layer with   %
%scatterers, and Region N is the layer with the        %
%transducer. The simulation assumes that the scatterers%
%are Gaussian distributions.                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Enter properties of the different layers.             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Enter the number of layers and density of scatterers
N=1;
nbar=5*(1000)^3; %#/m^3
nbar_mult=7;
mbar=nbar*nbar_mult;

%Enter the density and sound speed for each layer.

%*****************************************************%
%Typical Values for Different Tissue Types             %
%*****************************************************%
%Tissue        Density(Kg/m^3)       Speed(m/s)_2      %
% Fat          850_1,937_3           1476-1487         %
% Muscle       1050_1,1070_3         1500-1610         %
% Liver        1050_1,1064_3         1532-1570         %
% Brain        1021-1079_2           1501-1568         %
% Skull        1738-1914_2           2010-3360         %
% Bone (Cortical)   1700_1           3176-3406         %
%                                                      %
%1. Data from Table 4.1, pg. 53, NCRP Report No. 113.  %
%2. Human data Goss, Johnston, and Dunn "Ultrasonic    %
%   properties of mammalian tissues," JASA, 1978.      %
%3. Pig data Goss, Johnston, and Dunn "Ultrasonic      %
%   properties of mammalian tissues," JASA, 1978.      %
%*****************************************************%

rho_liver=1050;%+14*rand(1);
rho_muscle=1050;%+20*rand(1);
rho_fat=850;%+87*rand(1);
rho_water=1000;

rho=[rho_liver rho_muscle rho_fat rho_water];

c_liver=1532;%+38*rand(1);
c_muscle=1500;%+110*rand(1);
c_fat=1476;%+11*rand(1);
c_water=1540;
```

```
c=[c_liver c_muscle c_fat c_water];

%Find Transmission coefficients
T=1;
for ti=1:(N-1)
   Tij=2*rho(ti+1)*c(ti+1)/(rho(ti+1)*c(ti+1) + rho(ti)*c(ti));
   Tji=2*rho(ti)*c(ti)/(rho(ti+1)*c(ti+1) + rho(ti)*c(ti));

   T=T*Tij*Tji;
end

%Enter the attenuation properties for each layer.

%*************************************************%
%Typical Values for Different Tissue Types        %
%*************************************************%
%Tissue                   Attenuation a*f^b       %
%                  alpha_0(Np/cm/MHz^-b)   b      %
% Fat               0.053_5             1_5       %
% Muscle                                          %
%  Perpendicular    0.0145_3            1.90_3    %
%                   0.13-0.18_4         1_4       %
%  Parallel         0.3626_3            1.20_3    %
%                   0.33-0.47_4         1_4       %
% Liver             0.032-0.034_1       1.30-1.32_1 %
%                   0.08_2              1.13_2    %
% Brain             0.07_2              1.14_2    %
%  White            0.064_1             1.27_1    %
%  Grey             0.012_1             1.17_1    %
% Bone (Cortical)   0.46_4              1_4       %
%                                                 %
%1. Lyons and Parker "Absorption and attenuation in  %
%   soft tissues II - experimental results," IEEE, 1988%
%2. Goss, Frizzell, and Dunn "Ultrasonic absorption  %
%   and attenuation in mammalian tissues," UMB, 1979. %
%3. Kudo et al. "Basic study on the ultrasound       %
%   attenuation of fibrous biological tissue in the  %
%   frequency range of 10-40 MHz," IEEE-US, 1998.    %
%4. Goss, Johnston, and Dunn "Compilation of empirical %
%   properties of mammalian tissues. II," JASA, 1980. %
%5. Data from Table 5.2, pg. 64, NCRP Report No. 113. %
%*************************************************%

alpha_0=input('Enter attenuation in Np/cm/MHz:');
b=1;

%Give z location of each layer boundary.
z=[5e-2 2.6e-2 3e-2]; %m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Enter properties of the transducer.              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%*************************************************%
F=5e-2; %m focal length
f_num=4; %f# of transducer
fo=8e6; %Circuit resonance frequency for the transducer.
```

```
rel_zo=0; %Distance from focus to plate as fraction of zT.
%******************************************************%

a_source=0.5*F/f_num; %Transducer aperture radius

zT=sqrt(F^2 - a_source^2);  %Distance to aperture plane.

zo=zT*rel_zo;

St=pi*a_source^2;

%Focal gain of transducer.
Go=a_source^2/(2*F);

wx_m=0.87*f_num;

wz_m=6.01*(f_num^2);

lambda_o=c(1)/fo;

wx_o=wx_m*lambda_o;

wz_o=wz_m*lambda_o;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Determin Acoustic pulse in aperture plane             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Assume Kuv=1
%******************************************************%
BW=2e6 %Width of Gaussian describing transducer filtering.
vp=1000; %Amplitude of voltage spike excitation.
Kuv=1;%m/s V^-1
%******************************************************%

%Set incident voltage for each frequency.
dt=0.15/fo; %Sampling time separation.
Tp=2*(F+20e-3)/c(1); %Length of time window of interest.

v=zeros(1,ceil(Tp/dt)); %Excitation pulse.
v(35)=vp; %time1=35, time2=35

if mod(length(v),2)
   v=[v 0];
end

time=dt*[0:length(v)-1];
offset=time(35); %time1=35, time2=35

%Determine value for each frequency
Vinc=(ifft(v));
M=length(Vinc);

Vinc_p=Vinc(1:M/2);

%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 d_f2]*(1/dt)/(2*pi);
freq=[d_f1]*(1/dt)/(2*pi);

%Set Filtering Characteristics of Source.
H=abs(freq).*exp(-((abs(freq)-fo)/BW).^2); %Use Rayleigh Distribution

H=H/max(H);

Uz_o=Kuv*Vinc_p.*H;

uz_o=real(fft(Uz_o,M));

figure(1)
clf
plot(time*1e6,uz_o)
xlabel('Time (\mus)')
ylabel('Particle Velocity at Center of Aperture (m/s)')
grid

figure(2)
clf
plot(freq*1e-6,abs(Uz_o)/max(abs(Uz_o)))
xlabel('Frequency (MHz)')
ylabel('Normalized |U_z(0,0,z_T,\omega)|')
grid
axis([min(freq*1e-6) max(freq*1e-6) 0 1])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Enable to run many simulations in a row              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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

path='/u1_peur/bigelow/Backscatter_Simulation/simulation_data/';

if sv=='y'
   n=input('Enter filename to save scattered voltage and time: ','s');
   Num_Files=input('Enter number of independent scatterer ...
                    distributions: ');
else
   Num_Files = 1;
end

aeff_enter=input('Enter size of scatterers in micrometers: ');
aeff_enter=aeff_enter*1e-6;

clear uz_o

figure(77)
clf
hold
grid

time_start=clock;

for file_index=1:Num_Files
```

```
figure(77)
plot(etime(clock,time_start),100*(file_index-1)/Num_Files,'x')

V_scat=zeros(size(freq));

for mi=1:nbar_mult

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Find Scattered Field at each Frequency                %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Determine number of scatterers
    Numscat=ceil(nbar*100*wx_o*wx_o*(16e-3))
    aeff=aeff_enter
    g_max=1;
    Vs=zeros(1,length(freq));

    xn=-5*wx_o + 10*wx_o*rand(1,Numscat);
    yn=-5*wx_o + 10*wx_o*rand(1,Numscat);
    zn=-8e-3 + 16e-3*rand(1,Numscat);

    ko=2*pi*freq/c_water;

    k=2*pi*freq/c(1);

    alpha=100*alpha_0(1)*(freq/1e6).^b(1);

    k_comp=k+i*alpha;

    lambda=c(1)./freq;

    wx=wx_m*lambda;

    wz=wz_m*lambda;

    Psi=-2*i*((k_comp/(4*pi)).^2)*T;

    %Find V_scat
    term1=-0.1178 * g_max * Go^2 * aeff^3;

    term2a=exp(i*2*k_comp*zT); %Vector in Freq.

    term2b=exp(-i*2*k_comp'*zn); %Matrix in frequency, scatterer
                                 % number

    term3=exp(-2*(((wx.^-1)'*xn).^2 + ((wx.^-1)'*yn).^2 + ...
          ((wz.^-1)'*zn).^2)); %Matrix in frequency, scatterer number

    term4=exp(-0.4136*k_comp.^2*aeff^2); %Vector in frequency.

    term_freq_vector=(term1.*term2a.*term4 ...
                     .*Psi.*k_comp.*Vinc_p.*H.*H/St);

    term_freq_vector2=((term2b.*term3)*ones(Numscat,1))';

    Vs=(term_freq_vector.*term_freq_vector2).*(freq>1);
```

```
      V_scat=V_scat+Vs;
   end

   %Find V_plate

   termA=-(ko.^2.*wx.^2)*(Go^2) .* Vinc_p .* H .* H/(St*8*pi);
   termB=exp(i*ko*(2*zT - 2*zo));

   termC=exp(-(zo./wz).^2);

   V_plate=(termA.*termB.*termC).*(freq>1);

   figure(3)
   clf
   plot(freq*1e-6,abs(V_scat)/max(abs(V_scat)))
   grid
   hold
   plot(freq*1e-6,abs(V_plate)/max(abs(V_plate)),'r-.')
   xlabel('Frequency (MHz)')
   ylabel('Normalized |V|')


   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %Find Scattered Field in Time Domain                     %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   v_scat=real(fft(V_scat,M));

   figure(4)
   clf
   plot(time*1e6,v_scat)
   xlabel('Time (\mus)')
   ylabel('Scattered Voltage (V)')
   grid

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %PLATE SCATTER SIMULATION                                %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   V_plate(1) = 0;

   v_plate=real(fft(V_plate,M));

   figure(5)
   clf
   plot(time*1e6,v_plate)
   xlabel('Time (\mus)')
   ylabel('Plane Voltage (V)')
   grid

   TIME=time';
   VSCAT_TIME(:,file_index)=v_scat';

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Save Scattered Field to File                            %
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if sv=='y'

   txtfile=[path n '.txt'];

   fid=fopen(txtfile,'w');

   for ti=1:length(time)
      fprintf(fid,'%2.6e  ',time(ti));

      for fi=1:(Num_Files-1)
         fprintf(fid,' %2.6e  ',VSCAT_TIME(ti,fi));
      end

      fprintf(fid,' %2.6e \n',VSCAT_TIME(ti,Num_Files));

   end

   fclose(fid);
end

sv = input('Save Plate voltage and time: y/n ','s');

if sv=='y'

   n=input('Enter filename to save Plate voltage and time: ','s');
   name=[path n];
   txtfile=[path n '.txt'];

   fid=fopen(txtfile,'w');

   for ti=1:length(time)
      fprintf(fid,'%2.6e ',time(ti));
      fprintf(fid,'%2.6e \n',v_plate(ti));
   end

   fclose(fid);
end

sv = input('Save parameters: y/n ','s');

if sv=='y'
   n=input('Enter filename to save parameters: ','s');
   txtfile=[path n '.txt'];

   fid=fopen(txtfile,'w');

   fprintf(fid,'c (m/s) = %f \n',c(1));
   fprintf(fid,'aeff (um) = %f \n',aeff*1e6);
   fprintf(fid,'alpha (Np/cm MHz^-b) = %f \n',alpha_0);
   fprintf(fid,'b  = %f \n',b);
   fprintf(fid,'F (cm)  = %f \n',F*100);
   fprintf(fid,'f_num  = %f \n',f_num);
   fprintf(fid,'fo (MHz)  = %f \n',fo*1e-6);
   fprintf(fid,'BW (MHz)  = %f \n',BW*1e-6);
   fprintf(fid,'nbar (#/mm^3) = %f \n',mbar/1e9);
```

```
   fprintf(fid,'wx_m  = %f \n',wx_m);
   fprintf(fid,'wy_m  = %f \n',wx_m);
   fprintf(fid,'wz_m  = %f \n',wz_m);
   fprintf(fid,'zT (cm) = %f \n',zT*100);
   fprintf(fid,'dt*fo  = %f \n',dt*fo);
   fprintf(fid,'offset (us)  = %f \n',offset*1e6);

   fclose(fid);
end
```

Matlab Code Used to Generate Received Voltage for Shell Scatterers:

```
clear all;
close all;

i=sqrt(-1);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%BACKSCATTER SIMULATION                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%This is a program that computes the acoustic signal   %
%scattered from a phantom. The simulation neglects     %
%multiple scattering and assumes that the Born         %
%approximation is valid.  Region 1 is the phantom, and %
%Region 2 is the water containing the transducer.      %
%The simulation assumes that the scatterers are shells.%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Enter properties of the different layers.             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Enter the number of layers and density of scatterers
N=2;

%Enter the density and sound speed for each layer.

rho_phantom=1040;
rho_water=1000;

rho=[rho_phantom rho_water];

Temp=input('Enter temperature to calculate sound speed:');
Phan_Num=input('Enter Phantom Number (1=A, 2=B):');

c_water=CfromT(Temp);

if Phan_Num==1
   alpha0=6.3597/100; %Np/cm/MHz
   c_phantom=1534.4; %m/s
   nbar=5.1135*(1000)^3; %20.454#/m^3
   nbar_mult=4;
   mbar=nbar*nbar_mult;
```

```
else
   alpha0=9.5140/100; %Np/cm/MHz
   c_phantom=1539.6; %m/s
   nbar=5.9658*(1000)^3; %47.7266#/m^3
   nbar_mult=8;
   mbar=nbar*nbar_mult;
end

b=1;

c_phantom=c_phantom - 1.7*(22-Temp);

c=[c_phantom c_water];

%Find Transmission coefficients
T=1;
for ti=1:(N-1)
   Tij=2*rho(ti+1)*c(ti+1)/(rho(ti+1)*c(ti+1) + rho(ti)*c(ti));
   Tji=2*rho(ti)*c(ti)/(rho(ti+1)*c(ti+1) + rho(ti)*c(ti));

   T=T*Tij*Tji;
end

%Now consider window.
ff=[7 8 9];
TT=[.7910 .7491 .7093];

TTwin=polyfit(ff,TT,1);

%Give z location of each layer boundary.
Fdis=input('Enter distance to aperture plane of the transducer in m:');
zT=input('Enter distance from focal plane to surface of phantom in m:');

%z=[Fdis zT]; %m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Enter properties of the transducer.                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

f_num=input('Enter f-number for the transducer:');

a_source=input('Enter radius of transducer aperture in m:');

St=pi*a_source^2;

F=sqrt(Fdis^2 + a_source^2);  %Focal length of transducer.

%Focal gain of transducer.
Go=a_source^2/(2*F);

wx_m=0.87*f_num;

wz_m=input('Enter slope of wz (wz_m):');
wz_o=input('Enter intercept of wz (wz_o) in m:');

lambda_o=c(1)/9e6;
```

```
wx0=wx_m*lambda_o;

wz0=wz_m*lambda_o+wz_o;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Determine Acoustic pulse in aperture plane             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%*******************************************************%

fo=input('Enter value for fo in MHz:');

fo=fo*1e6; %Rayleigh resonance frequency for the transducer.

BW=input('Enter value for BW in MHz:');

BW=BW*1e6; %Width of Rayleigh describing transducer filtering.

vp=1000; %Amplitude of voltage spike excitation.
Kuv=1;%m/s V^-1
%*******************************************************%

%Set incident voltage for each frequency.
dt=1/1e8; %Sampling time separation.
Tp=2*(F+20e-3)/c(1);

v=zeros(1,ceil(Tp/dt)); %Excitation pulse.
v(50)=vp; %f/4=35, f/2,1=50
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if mod(length(v),2)
   v=[v 0];
end

time=dt*[0:length(v)-1];
offset=time(50); %f/4=35, f/2,1=50
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Determine value for each frequency
Vinc=(ifft(v));
M=length(Vinc);

Vinc_p=Vinc(1:M/2);

%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);

%Set Filtering Characteristics of Source.
H=abs(freq).*exp(-((abs(freq)-fo)/BW).^2); %Use Rayleigh Distribution

H=H/max(H);

Uz_o=Kuv*Vinc_p.*H;

uz_o=real(fft(Uz_o,M));
```

```
figure(1)
clf
plot(time*1e6,uz_o)
xlabel('Time (\mus)')
ylabel('Particle Velocity at Center of Aperture (m/s)')
grid

figure(2)
clf
plot(freq*1e-6,abs(Uz_o)/max(abs(Uz_o)))
xlabel('Frequency (MHz)')
ylabel('Normalized |U_z(0,0,z_T,\omega)|')
grid
axis([min(freq*1e-6) max(freq*1e-6) 0 1])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Enable to run many simulations in a row              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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

path='/u1_peur/bigelow/Backscatter_Simulation/simulation_data/';

if sv=='y'
   n=input('Enter filename to save scattered voltage and time: ','s');
   Num_Files=input('Enter number of independent scatterer distributions: ');
else
   Num_Files = 1;
end

clear uz_o

for file_index=1:Num_Files

   V_scat=zeros(size(freq));

   for mi=1:nbar_mult

      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      %Find Scattered Field at each Frequency              %
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      %Determine number of scatterers
      Numscat=ceil(nbar*100*wx0*wx0*(16e-3))
      aeff=24.5e-6;
      g_max=1;
      Vs=zeros(1,length(freq));

      xn=-5*wx0 + 10*wx0*rand(1,Numscat);
      yn=-5*wx0 + 10*wx0*rand(1,Numscat);
      zn=-8e-3 + 16e-3*rand(1,Numscat);

      ko=2*pi*freq/c_water;

      k=2*pi*freq/c(1);

      alpha=100*alpha0*(freq/1e6).^b;
```

```
    k_comp=k+i*alpha;

    lambda=c(1)./freq;

    wx=wx_m*lambda;

    wz=wz_m*lambda + wz_o;

    TTFix=polyval(TTwin,freq*1e-6);

    Psi=-2*i*((k_comp/(4*pi)).^2)*T.*TTFix;

    %Find V_scat
    term1=2 * Go^2 * aeff^3;

    term2a=exp(i*2*k_comp*zT+i*2*ko*(Fdis-zT)); %Vector in Freq.

    term2b=exp(-i*2*k_comp'*zn); %Matrix in frequency, scatterer
                                 %number

    term3=exp(-2*(((wx.^-1)'*xn).^2 + ((wx.^-1)'*yn).^2 + ...
          ((wz.^-1)'*zn).^2)); %Matrix in frequency, scatterer number

    term4=sinc(2*k_comp*aeff/pi); %Vector in frequency.

    term_freq_vector=(term1.*term2a.*term4 ...
                     .*Psi.*k_comp.*Vinc_p.*H.*H/St);

    term_freq_vector2=((term2b.*term3)*ones(Numscat,1))';

    Vs=(term_freq_vector.*term_freq_vector2).*(freq>1);

    V_scat=V_scat+Vs;
end

%Find V_plate

termA=-(ko.^2.*wx.^2)*(Go^2) .* Vinc_p .* H .* H/(St*8*pi);
termB=exp(i*ko*(2*Fdis));

V_plate=(termA.*termB).*(freq>1);


figure(3)
clf
plot(freq*1e-6,abs(V_scat)/max(abs(V_scat)))
grid
hold
plot(freq*1e-6,abs(V_plate)/max(abs(V_plate)),'r-.')
xlabel('Frequency (MHz)')
ylabel('Normalized |V|')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Find Scattered Field in Time Domain                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
   v_scat=real(fft(V_scat,M));

   figure(4)
   clf
   plot(time*1e6,v_scat)
   xlabel('Time (\mus)')
   ylabel('Scattered Voltage (V)')
   grid

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %PLATE SCATTER SIMULATION                              %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   V_plate(1) = 0;

   v_plate=real(fft(V_plate,M));

   figure(5)
   clf
   plot(time*1e6,v_plate)
   xlabel('Time (\mus)')
   ylabel('Plane Voltage (V)')
   grid

   TIME=time';
   VSCAT_TIME(:,file_index)=v_scat';

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Save Scattered Field to File                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if sv=='y'

   txtfile=[path n '.txt'];

   fid=fopen(txtfile,'w');

   for ti=1:length(time)
      fprintf(fid,'%2.6e  ',time(ti));

      for fi=1:(Num_Files-1)
         fprintf(fid,' %2.6e  ',VSCAT_TIME(ti,fi));
      end

      fprintf(fid,' %2.6e \n',VSCAT_TIME(ti,Num_Files));

   end

   fclose(fid);
end

sv = input('Save Plate voltage and time: y/n ','s');

if sv=='y'
```

```
   n=input('Enter filename to save Plate voltage and time: ','s');
   name=[path n];
   txtfile=[path n '.txt'];

   fid=fopen(txtfile,'w');

   for ti=1:length(time)
      fprintf(fid,'%2.6e ',time(ti));
      fprintf(fid,'%2.6e \n',v_plate(ti));
   end

   fclose(fid);
end

sv = input('Save parameters: y/n ','s');

if sv=='y'
   n=input('Enter filename to save parameters: ','s');
   txtfile=[path n '.txt'];

   fid=fopen(txtfile,'w');

   fprintf(fid,'c_phantom (m/s) = %f \n',c_phantom);
   fprintf(fid,'c_water (m/s) = %f \n',c_water);
   fprintf(fid,'alpha (Np/cm MHz^-b) = %f \n',alpha0);
   fprintf(fid,'b  = %f \n',b);
   fprintf(fid,'Phan_Num = %f \n', Phan_Num);
   fprintf(fid,'Temp = %f \n', Temp);
   fprintf(fid,'F (cm)  = %f \n',F*100);
   fprintf(fid,'Fdis (cm)  = %f \n',Fdis*100);
   fprintf(fid,'zT (cm) = %f \n',zT*100);
   fprintf(fid,'f_num  = %f \n',f_num);
   fprintf(fid,'a_source (cm) = %f \n',a_source*100);
   fprintf(fid,'wx_m  = %f \n',wx_m);
   fprintf(fid,'wy_m  = %f \n',wx_m);
   fprintf(fid,'wz_m  = %f \n',wz_m);
   fprintf(fid,'wz_o  = %f \n',wz_o);
   fprintf(fid,'fo (MHz)  = %f \n',fo*1e-6);
   fprintf(fid,'BW (MHz)  = %f \n',BW*1e-6);
   fprintf(fid,'nbar (#/mm^3) = %f \n',mbar/1e9);
   fprintf(fid,'dt  = %f \n',dt);
   fprintf(fid,'offset (us)  = %f \n',offset*1e6);

   fclose(fid);
end
```