12 Ultrasonic Measurement Modeling with MATLAB

In this Chapter we will implement complete ultrasonic measurement models in a series of MATLAB functions and scripts for the pulse-echo setup of Fig. 12.1. These measurement models will be used to simulate a number of measurement setups where a reference reflector such as a spherical pore, a flat-bottom hole, or a side drilled hole is present. Reference reflectors are commonly used in NDE tests to serve as calibration standards and they are also used to measure system performance. Here we will demonstrate the ability of the measurement models to simulate experimentally determined signals from these types of reference reflectors [12.1]. Similar demonstrations have been carried out worldwide by a number of researchers in a recent series of benchmark studies (see [12.2] for an overview of these activities from 2001- 2005). In those studies a variety of beam models and flaw scattering models were employed. Here, we will use the multi-Gaussian beam model of Chapter 9 and two of the flaw scattering models discussed Chapter 10 (the Kirchhoff approximation and the method of separation of variables) in conjunction with the various measurement models described in Chapter 11.

The MATLAB models of this Chapter can be used by the reader as the basis for implementing and studying many of the concepts and results discussed in this book in a more hands-on fashion, where the parameters can be readily changed and the results easily illustrated. Although the models are implemented for a simple pulse-echo configuration (Fig. 12.1) they can be used for a number of advanced purposes, such as examining ultrasonic beam behavior at curved interfaces, for example, and they can serve as the starting point for developing more complex simulation models.

12.1 A Summary of the Measurement Models

In the previous Chapter we developed measurement models suitable for several different testing situations. These included a general model that

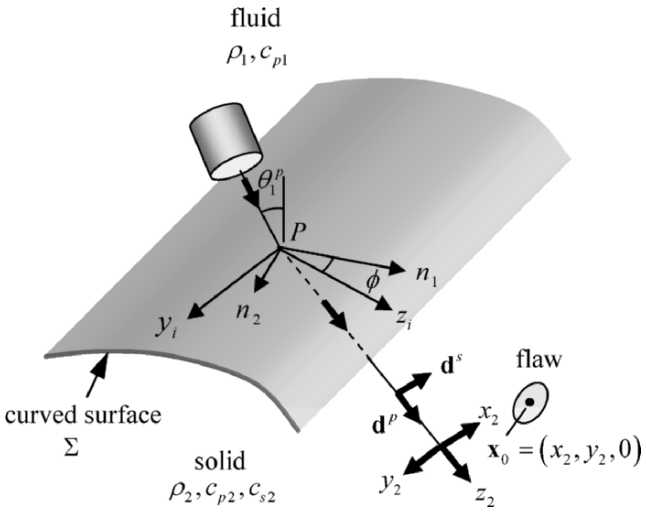

Fig. 12.1. Parameters for defining the problem of pulse-echo inspection of a flaw in a solid through a fluid-solid interface.

only relied on linearity and reciprocity and assumed the incident beam could be written in quasi-plane wave form. For that model the frequency components of the measured voltage were given by

$$V_R(\omega) = s(\omega)\left[\frac{4\pi\rho_2 c_{\alpha 2}}{-ik_{\alpha 2} Z_r^{T;a}}\right] \cdot \int_{S_f} \hat{V}^{(1)}(\mathbf{x},\omega)\hat{V}^{(2)}(\mathbf{x},\omega)\mathcal{A}(\mathbf{x},\omega)\exp\left[ik_{\alpha 2} e_n^{(2)} x_n\right] dS, \tag{12.1}$$

where, recall,

$$\mathcal{A}(\mathbf{x},\omega) = \frac{1}{4\pi\rho_2 c_{\alpha 2}^2}\left[\tilde{\tau}_{ji}^{(1)} d_i^{(2)} + C_{ijkl} d_k^{(2)}\left(e_l^{(2)}/c_{\alpha 2}\right)\tilde{v}_i^{(1)}\right] n_j \tag{12.2}$$

involves the stresses and velocity on the surface of the flaw normalized by the incident wave displacement amplitude at the flaw, i.e.

$$\tilde{\tau}_{ij}^{(1)} = \frac{-i\omega\tau_{ij}^{(1)}}{v_T^{(1)}\hat{V}^{(1)}}$$
$$\tilde{v}_j^{(1)} = \frac{-i\omega v_j^{(1)}}{v_T^{(1)}\hat{V}^{(1)}}. \tag{12.3}$$

The terms $\hat{V}^{(\alpha)}(\mathbf{x},\omega)$ $(\alpha = 1,2)$ are the incident velocity field amplitudes on the flaw surface for states (1) and (2), where in state (1) the transmitting transducer is firing with a unit velocity on its face and for state (2) the receiving transducer is firing with a unit velocity on its face. Both of these amplitude terms, therefore, can be calculated with appropriate ultrasonic beam and attenuation models. The remaining fields in the $\mathcal{A}(\mathbf{x},\omega)$ term are the total fields on the surface of the flaw normalized by the displacement of the incident wave. Those fields can also be modeled with an appropriate flaw scattering model. This measurement model is quite general and should apply to most testing situations. Note that in this form the flaw far-field scattering amplitude does not appear directly but, as shown in the last Chapter, $\mathcal{A}(\mathbf{x},\omega)$ is closely related to the component of the scattering amplitude that appears in other measurement models (see Eq. (11.34)).

The second model developed assumed that the flaw was small enough so that the incident fields did not vary significantly over the surface of the flaw. In that case we found

$$V_R(\omega) = s(\omega)\hat{V}_0^{(1)}(\omega)\hat{V}_0^{(2)}(\omega)A(\omega)\left[\frac{4\pi\rho_2 c_{\alpha 2}}{-ik_{\alpha 2}Z_r^{T;a}}\right], \tag{12.4}$$

where

$$\hat{V}_0^{(1)} = \hat{V}^{(1)}(\mathbf{x}_0,\omega)$$
$$\hat{V}_0^{(1)} = \hat{V}^{(1)}(\mathbf{x}_0,\omega)$$

are the now the velocity amplitude terms evaluated at the "center" of the flaw and a flaw far-field scattering amplitude term, $A(\omega)$, is directly a part of the measurement model.

For a cylindrical scatterer where beam variations are not negligible we can again apply the measurement model of Eq. (12.1). For a small cylindrical scatterer, however, where beam variations over the scatterer cross section are negligible we found

$$V_R(\omega) = s(\omega)\left[\int_L \hat{V}_0^{(1)}(z,\omega)\hat{V}_0^{(2)}(z,\omega)\,dz\right]\left[\frac{A(\omega)}{L}\right]\left[\frac{4\pi\rho_2 c_{\alpha 2}}{-ik_{\alpha 2}Z_r^{T;a}}\right], \qquad (12.5)$$

where, recall,

$$\hat{V}_0^{(1)}(z,\omega) \equiv \hat{V}^{(1)}(\mathbf{y}_0, z, \omega)$$
$$\hat{V}_0^{(2)}(z,\omega) \equiv \hat{V}^{(2)}(\mathbf{y}_0, z, \omega)$$

are now the incident velocity amplitude terms calculated at the "center" of the scatterer and at any axial position along its length. The far-field scattering amplitude of the flaw appearing in Eq. (12.5) is the same 3-D scattering amplitude in Eq. (12.4), but as mentioned in the last Chapter we also can use a 2-D scattering amplitude calculation in Eq. (12.5) if we use the relationship of Eq. (11.48).

Each of the measurement models described above has three components: 1) the system function, $s(\omega)$, describing all the electrical and electromechanical elements of the measurement system, 2) the velocity fields $\hat{V}^{(1)}, \hat{V}^{(2)}$ that characterize the incident fields on the flaw from the transmitting transducer or receiving transducer, respectively, when there is a unit driving velocity on those transducer faces, and 3) the scattering properties of the flaw itself, described in terms of $\mathcal{A}(\mathbf{x},\omega)$ or $A(\omega)$. In this Chapter we will develop a series of MATLAB functions that model each of these three components and implement the measurement models described above.

The problem we will consider is shown in Fig. 12.1 where a transducer is performing a pulse-echo inspection of a flaw in an immersion setup. First, assume that the flaw is small enough so that the beam variations over its surface can be neglected and the measurement model of Eq. (12.4) can be used. The distances along a ray (a path satisfying Snell's law) extending normally from the center of the transducer are z_1, z_2 for the fluid and solid, respectively, and the center of the flaw is located at a point (x_2, y_2) relative to that central ray as shown in Fig. 12.1, where the y_2-axis is normal to the plane of incidence. The acute angle of the central ray in the fluid and the normal to the interface (at point *P* where that ray intersects the interface) is the angle θ_1^p. The (y_i, z_i) coordinates are in the tangent plane of the interface and y_i is normal to the plane of incidence. The angle of the z_i-axis from one of the principal directions, n_1, of the surface is the angle ϕ. [Important: note that these definitions are different

from some of those used in Chapter 9 and Chapter 11 so that in the MATLAB measurement models of this Chapter one should relate the quantities in those models back to Fig. 12.1].

We can express the measurement model of Eq. (12.4) more explicitly by examining the various pieces that contribute to the velocity terms. Since we are considering a pulse-echo setup here, our measurement model can be written as

$$V_R(\omega) = s(\omega)\left[\hat{V}_0^{(1)}(\omega)\right]^2 A(\omega)\left[\frac{4\pi\rho_2 c_{\alpha 2}}{-ik_{\alpha 2} Z_r^{T;a}}\right] \tag{12.6}$$

and the incident velocity field, $\hat{V}_0^{(1)}$, can be written as

$$\hat{V}_0^{(1)} = \exp\left[-\alpha_{p1}(\omega) z_1 - \alpha_{\gamma 2}(\omega) z_2\right]\left[V_i^{\gamma} / v_0\right] \tag{12.7}$$

where z_1, z_2 are the distances the sound beam from the transducer has propagated in the fluid and the solid, respectively, and $\alpha_{p1}(\omega), \alpha_{\gamma 2}(\omega)$ are the frequency dependent attenuation coefficients for the compressional wave in the fluid and the wave of type γ in the solid, respectively. The term V_i^{γ} / v_0 is the ideal velocity field (for a material with no losses) at the flaw normalized by the normal velocity, v_0, on the face of the transducer. This ideal field will be described by a multi-Gaussian beam model of the type discussed in Chapter 9. The types of transducer we will consider in the setup of Fig. 12.1 with a multi-Gaussian beam model are circular planar and spherically focused piston transducers. In the following section we will use the general formulation of Chapter 9 to derive a multi-Gaussian beam model that is directly applicable to a setup of the type given in Fig. 12.1.

12.2 The Multi-Gaussian Beam Model

In developing the multi-Gaussian beam model the interface will assumed to be either planar or curved, with the plane of incidence of the transducer aligned with one of the principal curvatures of the interface (i.e. $\phi = 0$ in Fig. 12.1). For a single fluid-solid interface on transmission through the interface it is not necessary to rotate axes and the angle $\lambda = 0$ in Eqs. (9.89)-(9.91). Also, we do not need to put the transmission coefficient in matrix form, but can use the simpler scalar relation of Eq. (9.79). The ideal normalized velocity for a wave of type γ in the solid as computed by

the multi-Gaussian beam model (with 15 coefficients) for this case is then given by (see Fig. 12.1)

$$V_i^{\gamma}\mathbf{d}^{\gamma} = \sum_{r=1}^{15} T_{12}^{\gamma;p}\mathbf{d}^{\gamma}\left[V_1^{p}(0)\right]_r \frac{\sqrt{\det\left[\mathbf{M}_2^{\gamma}(z_2)\right]_r}}{\sqrt{\det\left[\mathbf{M}_2^{\gamma}(0)\right]_r}} \frac{\sqrt{\det\left[\mathbf{M}_1^{p}(z_1)\right]_r}}{\sqrt{\det\left[\mathbf{M}_1^{p}(0)\right]_r}} \cdot \exp\left[ik_{p1}z_1 + ik_{\gamma 2}z_2 + i\frac{k_{p1}}{2}\mathbf{y}^T\left[c_{p1}\mathbf{M}_2^{\gamma}(z_2)\right]_r\mathbf{y}\right], \quad (\gamma = p,s) \tag{12.8}$$

where, $\mathbf{y}^T = (x_2, y_2)$ and at the face of the transducer

$$\left[V_1^{p}(0)\right]_r = A_r v_0(\omega)$$
$$\left[\mathbf{M}_1^{p}(0)\right]_r = \begin{bmatrix} \dfrac{iB_r}{c_{p1}D_R} & 0 \\ 0 & \dfrac{iB_r}{c_{p1}D_R} \end{bmatrix} \tag{12.9}$$

in terms of the Wen and Breazeale coefficients A_r, B_r. The polarization vector, $\mathbf{d}^{\gamma}$, is shown in Fig. 12.1 for both P-waves and SV-waves. The plane wave transmission coefficient, $T_{12}^{\gamma;p}$ is based on a velocity ratio. The parameter $D_R = k_{p1}a^2/2$ is the Rayleigh distance, where the radius of the transducer is *a* and k_{p1} is the wave number for P-waves in medium one. Similarly $k_{\gamma 2}\ (\alpha = p,s)$ are wave numbers for P- or S-waves in medium two. From the propagation law for medium one, from Eq. (9.28) we have

$$\left[M_1^{p}(z_1)\right]_r = \begin{bmatrix} \dfrac{1}{c_{p1}(z_1 - iD_R/B_r)} & 0 \\ 0 & \dfrac{1}{c_{p1}(z_1 - iD_R/B_r)} \end{bmatrix}. \tag{12.10}$$

From Eq. (12.9) and Eq. (12.10) then it follows that

$$\frac{\sqrt{\det\left[\mathbf{M}_1^{p}(z_1)\right]_r}}{\sqrt{\det\left[\mathbf{M}_1^{p}(0)\right]_r}}\left[V_1^{p}(0)\right]_r = \frac{A_r v_0}{1 + i z_1 B_r / D_R}. \tag{12.11}$$

The transmission law across the interface also gives (Eq. (9.94))

$$\left[M_2^{\gamma}(0)\right]_r = \begin{bmatrix} \dfrac{M_1}{c_{p1}\left(z_1 - iD_R/B_r\right)} & 0 \\ 0 & \dfrac{M_2}{c_{p1}\left(z_1 - iD_R/B_r\right)} \end{bmatrix}, \tag{12.12}$$

where

$$M_1 = \left(\cos^2\theta_1^p + Kh_{11}\right)/\cos^2\theta_2^{\gamma} \tag{12.13}$$

and

$$M_2 = 1 + Kh_{22} \tag{12.14}$$

are given in terms of the principal interface curvatures $\left(h_{11}, h_{22}\right)$ and

$$K = \left(z_1 - iD_R/B_r\right)\left(\cos\theta_1^p - \frac{c_{p1}}{c_{\gamma 2}}\cos\theta_2^{\gamma}\right). \tag{12.15}$$

Finally, from the propagation law (Eq. (9.28)) for the propagation in medium two we have

$$\left[M_2^{\gamma}(z_2)\right]_r = \begin{bmatrix} \dfrac{\left[\left\{M_2^{\gamma}(0)\right\}_{11}\right]_r}{1 + z_2 c_{\gamma 2}\left[\left\{M_2^{\gamma}(0)\right\}_{11}\right]_r} & 0 \\ 0 & \dfrac{\left[\left\{M_2^{\gamma}(0)\right\}_{22}\right]_r}{1 + z_2 c_{\gamma 2}\left[\left\{M_2^{\gamma}(0)\right\}_{22}\right]_r} \end{bmatrix}. \tag{12.16}$$

Thus, we have

$$\frac{\sqrt{\det\left[\mathbf{M}_2^{\gamma}(z_2)\right]_r}}{\sqrt{\det\left[\mathbf{M}_2^{\gamma}(0)\right]_r}} = \frac{1}{\sqrt{1 + z_2\dfrac{c_{\gamma 2}}{c_{p1}}\dfrac{M_1}{\left(z_1 - iD_R/B_r\right)}}} \cdot \frac{1}{\sqrt{1 + z_2\dfrac{c_{\gamma 2}}{c_{p1}}\dfrac{M_2}{\left(z_1 - iD_R/B_r\right)}}} \tag{12.17}$$

and

$$c_{p1}\left[\mathbf{M}_2^{\gamma}\left(z_2\right)\right]_r = \begin{bmatrix} \dfrac{1}{\dfrac{\left(z_1 - iD_R / B_r\right)}{M_1} + z_2 \dfrac{c_{\gamma 2}}{c_{p1}}} & 0 \\ 0 & \dfrac{1}{\dfrac{\left(z_1 - iD_R / B_r\right)}{M_2} + z_2 \dfrac{c_{\gamma 2}}{c_{p1}}} \end{bmatrix}. \tag{12.18}$$

To put the final expressions in a more compact form, let

$$\begin{aligned} Z_1^r &= \frac{\left(z_1 - iD_R / B_r\right)}{M_1} \\ Z_2^r &= \frac{\left(z_1 - iD_R / B_r\right)}{M_2}. \end{aligned} \tag{12.19}$$

[Note: Z_1^r, Z_2^r are distances, not impedances here]. Then the multi-Gaussian beam model becomes, finally

$$\begin{aligned} V_i^{\gamma}\mathbf{d}^{\gamma} = T_{12}^{\gamma;p}\mathbf{d}^{\gamma} v_0 \sum_{r=1}^{15} & \frac{A_r}{1 + i\, z_1 B_r / D_R} \\ & \cdot \frac{\sqrt{Z_1^r}}{\sqrt{Z_1^r + z_2\left(c_{\gamma 2} / c_{p1}\right)}} \frac{\sqrt{Z_2^r}}{\sqrt{Z_2^r + z_2\left(c_{\gamma 2} / c_{p1}\right)}} \\ & \cdot \exp\left[ik_{p1} z_1 + ik_{\gamma 2} z_2 + i\frac{k_{p1}}{2} \mathbf{y}^T \left[c_{p1} \mathbf{M}_2^{\gamma}\left(z_2\right)\right]_r \mathbf{y} \right] \end{aligned} \tag{12.20}$$

with

$$c_{p1}\left[\mathbf{M}_2^{\gamma}\left(z_2\right)\right]_r = \begin{bmatrix} \dfrac{1}{Z_1^r + z_2\left(c_{\gamma 2} / c_{p1}\right)} & 0 \\ 0 & \dfrac{1}{Z_2^r + z_2\left(c_{\gamma 2} / c_{p1}\right)} \end{bmatrix}. \tag{12.21}$$

The square roots appearing in Eq. (12.21) are unambiguous so that they can be calculated directly.

12.3 Measurement Model Input Parameters

In order to model the single interface problem shown in Fig.12.1, there are a significant number of input parameters that need to be defined. Here we will outline those parameters and the manner in which they will be represented in MATLAB. First, there are several general parameters that we will call setup parameters:

Setup Parameters
f….the frequencies at which the response will be calculated (MHz)
type1….the type of wave ('p' or 's') in medium one (a string)
type2….the type of wave ('p' or 's') in medium two (a string)

Although we will initially only consider problems where medium one is a fluid where type1 = 'p', we will leave type1 arbitrary to show the structure of input parameters in a more general setting.

Next, we need to define parameters that will allow us to determine the system function:

System Parameters
sysf….the name of a function that will either model the system function or calculate it experimentally (a string).
amp….the amplitude of a modeled system function (volts/MHz)
fc….the center frequency of a modeled system function (MHz)
bw….the bandwidth of a modeled system function
z1r….the distance in the fluid used in a reference scattering configuration to calculate the system function experimentally
en….the noise constant used in a Wiener filter when obtaining the system function experimentally
ref_file….the name of a MAT-file (a string). This file must contain the time axis and measured waveform obtained from the reference scattering configuration. These measured values are used in the function whose name is contained in sysf

In an ultrasonic system the system function determines the effects of all the electrical and electromechanical components. The sysf parameter allows us to use either an experimentally determined system function in the measurement model or a model-based function. If this value is the string 'systf' then the model-based function systf (which is defined later) will be used. The function systf obtains the amplitude, center frequency, and bandwidth to be used in calculating the system function from the amp,

fc, and bw parameters, respectively. Otherwise the user must supply the name of a compatible function that calculates the system function experimentally. Examples of the use of both types of these functions will be given. The function that calculates the system function experimentally needs to have as one of its inputs a measured waveform from a reference scattering configuration. This waveform and its time axis is contained in a MATLAB MAT-file whose filename is given by the contents of ref_file. In this MAT-file the time axis is a MATLAB vector named t_ref and the reference waveform is a MATLAB vector named ref. The function that calculates the system function experimentally also must use the same transducer parameters, pulser/receiver settings, etc. as in a flaw measurement so that a system function can be determined that is also appropriate to the flaw measurement. However, in a reference experiment where the waves received from the front surface of an immersed block can be used to calculate the system function, as described in Chapter 6, the water path length might be different from that of a flaw measurement setup. Thus, this water path length is given by the parameter z1r. If there are other parameters that are different in the reference experiment from those used in the flaw measurement (such as the material properties of the block, etc.) then they must also be included as additional setup system parameters.

There are also parameters associated with the transducer. For circular piston probes we need to specify:

Transducer parameters
d….the transducer diameter (mm)
fl….the transducer geometrical focal length (mm)

There are also a number of geometry parameters:

Geometry Parameters
z1….the distance traveled by the sound in medium one along a central ray path (mm)
z2….the distance traveled by the sound in medium two along a central ray path (mm)
x2….the perpendicular distance from the central ray axis to the center of the flaw (see Fig. 12.1) in the plane of incidence (mm)
y2….the perpendicular distance from the central ray axis to the center of the flaw (see Fig. 12.1) in a plane perpendicular to the plane of incidence (mm)
i_ang….the acute angle between the normal to the transducer and the normal to the interface at the point where the central ray from the

transducer strikes the interface (deg) [This is the angle θ_1^p shown in Fig. 12.1].
R1....the principal radius of curvature (Fig. 12.1) in the n_1 direction (mm)
R2....the principal radius of curvature (Fig. 12.1) in the n_2 direction (mm)
p_ang....the angle between the plane of incidence and the n_1 direction (deg) [This is the angle ϕ shown in Fig. 12.1].

The present study will assume that the plane of incidence and the n_1 direction are aligned so that p_ang = 0, but this parameter has been included for generality.

Not surprisingly, there are also quite a number of material parameters to specify:

Material Parameters
d1….the density of medium one (gm/cm^3)
d2….the density of medium two (gm/cm^3)
cp1….the P-wave speed of medium one (m/sec)
cs1….the S-wave speed of medium one (m/sec)
cp2….the P-wave speed of medium two (m/sec)
cs2….the S-wave speed of medium two (m/sec)
p1….P-wave attenuation fitting coefficients for medium one
s1….S-wave attenuation fitting coefficients for medium one
p2….P-wave attenuation fitting coefficients for medium two
s2….S-wave attenuation fitting coefficients for medium two

Again, for generality, we will leave the possibility of medium one having shear properties. The attenuation fitting coefficients will be used to define the attenuation coefficients in terms of powers of frequency. These will be discussed when we develop the attenuation model term.

The "flaw" cases we will consider in these examples will be of simple shapes (e.g. spherical voids, cylindrical holes, circular cracks) so that only several parameters will be needed in addition to the name of the function that will calculate the scattering amplitude:

Flaw Parameters
b…. radius of the flaw (mm)
f_ang….acute angle of the flaw with respect to the central ray (deg) (see Fig. 12.1)
Afunc….the name of the function that will calculate the far-field scattering amplitude of the flaw (a string)

Finally, we have a number of parameters associated with the particular types of waves we are considering in medium one and two. They are the wave speeds in medium one and two for the specified wave types in those media and the corresponding plane wave transmission coefficient. We have labeled these parameters wave parameters:

Wave Parameters
c1….the wave speed for the wave of type1 in medium one (m/sec)
c2….the wave speed for the wave of type2 in medium two (m/sec)
T12….the plane wave transmission coefficient (based on velocity or displacement ratios) appropriate to waves of type1 and type2

There is one difference between the wave parameters and the other parameters in that the wave parameters are *derived* parameters so that if the wave types and/or wave speeds are changed these wave parameter values will not be consistent with those choices unless they also are appropriately changed. *Thus, it is necessary to update these wave parameters with the current values present in the setup before using them*.

Because there are a considerable number of parameters, it is essential to have a flexible method to examine, retrieve, and change them and to pass them to other functions. Thus we have placed all of these parameters in a MATLAB structure named setup. This setup structure has a number of fields called system (for the system function), trans (for transducer), geom (for geometry), matl (for material), flaw (for flaw), and wave (for wave parameters). These fields in turn have fieldnames that are associated with the parameters just listed. A MATLAB function called setup_maker defines a complete set of the default parameters needed for a measurement model suitable for problems of the type shown in Fig. 12.1 and generates the setup structure (Code Listing 12.1). In setup_maker all the setup parameters are first defined and then placed into the setup structure. Both of these operations could have been performed in one step but they have been separated here strictly to make them more explicit for the reader.

Code Listing 12.1. The MATLAB function for generating a default structure, setup, that contains all the parameters needed for a measurement model of the case shown in Fig 12.1

```
function setup =setup_maker( )

%setup parameters
f = 5;
type1 = 'p';
type2 ='p';
% system function parameters
sysf ='systf';
amp = 5.0E-02;
fc = 5;
bw = 3;
z1r = 0.0;
en =0.01;
ref_file ='empty';
% transducer parameters
d = 12.7;
fl= inf;
%geometry parameters
z1 = 0;
z2 = linspace(0,200,512);
x2 = 0.0;
y2 =0.0;
i_ang = 0;
R1 = inf;
R2 = inf;
p_ang = 0;
% material parameters
d1 = 1.0;
d2 = 1.0;
cp1 =1480;
cs1 = 0;
cp2 =1480;
cs2 = 0;
p1 = zeros(1,5);
s1 = zeros(1,5);
p2 = zeros(1,5);
s2 = zeros(1,5);
%flaw parameters
b =0.0;
f_ang = 0.0;
```

```
Afunc = 'empty';
%wave parameters
c1 =1480;
c2 = 1480;
T12 =1.0;

% put setup in a structure
setup.f = f;
setup.type1 = type1;
setup.type2 = type2;
setup.system.sysf = sysf;
setup.system.amp =amp;
setup.system.fc = fc;
setup.system.bw = bw;
setup.system.z1r =z1r;
setup.system.en =en;
setup.system.ref_file = ref_file;
setup.trans.d = d;
setup.trans.fl =fl;
setup.geom.z1 = z1;
setup.geom.z2 = z2;
setup.geom.x2 = x2;
setup.geom.y2 = y2;
setup.geom.i_ang = i_ang;
setup.geom.R1 =R1;
setup.geom.R2 = R2;
setup.geom.p_ang = p_ang;
setup.matl.d1 =d1;
setup.matl.d2 = d2;
setup.matl.cp1 =cp1;
setup.matl.cs1 = cs1;
setup.matl.cp2 = cp2;
setup.matl.cs2 =cs2;
setup.matl.p1 = p1;
setup.matl.s1 =s1;
setup.matl.p2 = p2;
setup.matl.s2 = s2;
setup.flaw.b = b;
setup.flaw.f_ang = f_ang;
setup.flaw.Afunc = Afunc;
setup.wave.c1 = c1;
setup.wave.c2 = c2;
setup.wave.T12 = T12;
```

It can be seen from Code Listing 12.1 that the default parameters are for a 5 MHz center frequency, 3 MHz bandwidth system function and a 12.7 mm diameter planar transducer radiating a P-wave directly into a single medium (water), since the material properties for water are used for both materials. The P-wave response is to be calculated at a single frequency of 5 MHz at 512 points along the transducer central axis from zero to 200 mm, with no attenuation and with the flaw parameters initially set to zero. It can be seen that the wave parameters are also made consistent with the other setup parameters in this default case. However, to remain consistent these wave parameters must be recomputed whenever the wave types or materials are changed, as mentioned previously.

This default set of parameters would be suitable for generating, for example, a central axis transducer beam response similar to those shown in Chapter 8 (see, for example, Fig. 8.9). We will demonstrate the use of this default set of parameters (and others) after we have developed the necessary MATLAB multi-Gaussian beam model.

The setup structure makes it easy to manipulate all the problem parameters and to set up various cases. Examples of using this structure will be given when we begin to discuss specific case studies later in this Chapter. A MATLAB function display_setup has also been defined that allows one to examine all these setup parameters.

12.4 A Multi-Gaussian Beam Model in MATLAB

To generate a complete multi-Gaussian beam model that can simulate the ideal normalized velocity field, V_i^{γ}/v_0 of Eq. (12.8), in addition to a subset of the setup parameters (attenuation parameters and flaw parameters, for example, are not needed for this beam model) we need the Gaussian coefficients and we must calculate the appropriate plane wave transmission coefficient. The Wen and Breazeale fifteen complex coefficients, $\left(A_r, B_r\right)$, have been placed in a MATLAB function gauss_c15 that returns their values. This function is given in the following listing:

Code Listing 12.2. A MATLAB function that returns the fifteen Wen and Breazeale coefficients. These coefficients are used to generate a multi-Gaussian beam model of a circular piston transducer.

```
function [a, b] = gauss_c15

a = zeros(15,1);
b = zeros(15,1);
a(1) = -2.9716 + 8.6187*i;
a(2) = -3.4811 + 0.9687*i;
a(3) = -1.3982 - 0.8128*i;
a(4) = 0.0773 - 0.3303*i;
a(5) = 2.8798 + 1.6109*i;
a(6) = 0.1259 - 0.0957*i;
a(7) = -0.2641 - 0.6723*i;
a(8) = 18.019 + 7.8291*i;
a(9) = 0.0518 + 0.0182*i;
a(10) = -16.9438 - 9.9384*i;
a(11) = 0.3708 + 5.4522*i;
a(12) = -6.6929 + 4.0722*i;
a(13) = -9.3638 - 4.9998*i;
a(14) = 1.5872 - 15.4212*i;
a(15) = 19.0024 + 3.6850*i;
b(1) = 4.1869 - 5.1560*i;
b(2) = 3.8398 - 10.8004*i;
b(3) = 3.4355 - 16.3582*i;
b(4) = 2.4618 - 27.7134*i;
b(5) = 5.4699 + 28.6319*i;
b(6) = 1.9833 - 33.2885*i;
b(7) = 2.9335 - 22.0151*i;
b(8) = 6.3036 + 36.7772*i;
b(9) = 1.3046 - 38.4650*i;
b(10) = 6.5889 + 37.0680*i;
b(11) = 5.5518 + 22.4255*i;
b(12) = 5.4013 + 16.7326*i;
b(13) = 5.1498 + 11.1249*i;
b(14) = 4.9665 + 5.6855*i;
b(15) = 4.6296 + 0.3055*i;
```

The plane wave transmission coefficient must be calculated consistent with the material properties and wave types specified in the setup structure parameters. We will use a MATLAB function that is passed the setup

structure and returns the appropriate transmission coefficient. The MATLAB function fluid_solid, (see Code Listing 12.3) for example, calculates the plane wave transmission coefficient for a fluid-solid interface using the explicit expressions given in Appendix D (Eq. (D.59)). For a refracted S-wave, this transmission coefficient will be complex if the first critical angle is exceeded. The function fluid_solid calculates this complex transmission coefficient *for positive frequencies only*. Thus, if one wants to synthesize a pulse with these calculations, one will need to follow the steps discussed in Appendix A in performing the necessary FFT.

Code Listing 12.3. A MATLAB function for calculating the plane wave transmission coefficient for a fluid-solid interface.

```
function  T12 = fluid_solid(setup)
% fluid_solid(setup) computes the P-P (tpp)
% and P-S (tps) transmission coefficients based on velocity ratios
% for a plane fluid-solid interface. It obtains the necessary input
% parameters from the setup structure and then returns the
% appropriate transmission coefficient


% get setup parameters
type1 =setup.type1;
type2 =setup.type2;
inc= setup.geom.i_ang;
d1 = setup.matl.d1;
d2 =setup.matl.d2;
cp1 = setup.matl.cp1;
cs1 =setup.matl.cs1;
cp2 =setup.matl.cp2;
cs2 =setup.matl.cs2;

% consistency check (if material one is not a fluid
% then can't use this fluid-solid trans. coefficient)

if strcmp(type1, 's') | cs1 ~=0
   error('wrong wave type or wave speed for medium 1')
end


% calculate transmission coefficients
```

```
iang = (inc.*pi)./180;
sinp = (cp2/cp1)*sin(iang);
sins =(cs2/cp1)*sin(iang);
len = length(sinp);
for j=1:len
if sinp(j) >= 1
        cosp(j) = i*sqrt(sinp(j)^2 - 1);
        else
        cosp(j) = sqrt(1 - sinp(j)^2);
        end
end
for j=1:len
if sins(j) >= 1
        coss(j) = i*sqrt(sins(j)^2 - 1);
        else
        coss(j) =sqrt(1 - sins(j)^2);
        end
end
denom = cosp + (d2/d1)*(cp2/cp1)*sqrt(1-sin(iang).^2).*(4.*((cs2/cp2)^2)…
.*(sins.*coss.*sinp.*cosp) + 1 - 4.*(sins.^2).*(coss.^2));
tpp = (2*sqrt(1 - sin(iang).^2).*(1 - 2*(sins.^2))).*denom;
tps = -(4*cosp.*sins.*sqrt(1 - sin(iang).^2)).*denom;

%select appropriate coefficient
if strcmp(type2, 'p')
   T12 = tpp;
elseif strcmp(type2, 's')
   T12 = tps;
else
   error('wrong wave type specification')
end
```

Having the setup structure, the multi-Gaussian beam coefficients, and the plane wave transmission coefficient, we now are in a position to develop the complete multi-Gaussian beam model. The MATLAB function MGbeam extracts the setup parameters it needs from the setup structure (which is the only input to MGbeam); calls the function c_gauss15 to obtain the Gaussian beam coefficients; updates the setup.wave parameters c1 and c2 to be consistent with the wave types; calls the fluid_solid function to compute the plane wave transmission coefficient (and then updates the setup structure with that coefficient); computes some of the

additional parameters appearing explicitly in the beam model, and then computes the ideal velocity field in Eq. (12.20). A function init_z is called to generate an empty array of velocity values before the beam model calculations are performed. That function is given in Code Listing 12.4. This function decides what the largest size of matrix is present for the parameters f, z1, z2, x2, and y2, and pre-allocates an array of zeros of the same size for the velocity field, v, to be calculated. This pre-allocation is for efficiency only. One could have instead simply initialized v with v = 0. MGbeam is coded to allow f, z1, z2, x2, and y2 to be either scalars, vectors, or 2 by 2 arrays so that one can perform a number of different studies and plot various combinations of parameters, as will be shown shortly. MGbeam is not coded to allow the incident angle with the interface to be other than a single scalar value. However, multiple calls to MGbeam with different values of setup.geom.i_ang could be used to perform those types of studies.

Code Listing 12.4. A MATLAB function for pre-allocating memory for the velocity calculations of the same size as the largest array present in the input parameters f, z1, z2, x2, y2.

```
function v =init_z(setup)
% get parameters that may not be scalars
f =setup.f;
z1 = setup.geom.z1;
z2=setup.geom.z2;
x2 =setup.geom.x2;
y2 = setup.geom.y2;
%get dimensions, put in rows
A = [size(f); size(z1);size(z2);size(x2); size(y2)];
%get product of dimensions for each varaible
prod =A(:,1).*A(:,2); % this is a column vector
%find which row (or rows) have largest dimension
ind = find( prod = = max(prod));
%pick first row with largest dimension
val = ind(1);
% initialize v with  zeros of same size
% as the parameter(s) with largest dimensions
v = zeros(A(val,:));
```

For a spherically focused probe the Gaussian beam coefficients B_r are simply changed by letting $B_r \to B_r + iD_R / F$, where D_R is the Rayleigh length and F is the focal length, as discussed in Chapter 9. The propagation term $\exp\left(ik_{p1}z_1 + ik_{\gamma 2}z_2\right)$ is not included in the calculations since this term only generates a time delay $t_0 = z_1 / c_{p1} + z_2 / c_{\gamma 2}$ in going from the transducer to the point in the solid and this delay can easily be added in separately, if needed, by simply shifting the time axis appropriately. Thus, for pulses calculated using MGbeam the time $t = 0$ corresponds to the time when the incident quasi-plane wave is at the "center" of the flaw. MGbeam returns the ideal velocity field, V_i^{γ} / v_0, and the updated setup structure. As can be seen from Code Listing 12.5, the multi-Gaussian beam model is calculated in only the last fourteen lines of that Code. All the other parts of MGBeam simply prepare the necessary input parameters. Thus, except in very special cases there are no alternative beam models as simple and fast as a multi-Gaussian beam model.

Code Listing 12.5. A MATLAB function MGbeam for calculating the wave field of circular piston transducer (planar or focused) radiating through a fluid-solid interface into a solid. The function uses a multi-Gaussian beam model.

```
function [v,setup ]=MGbeam(setup)

% get setup parameters
f = setup.f;                    %frequency or frequencies (MHz)
type1 = setup.type1;            % wave type in medium one
type2 = setup.type2;            % wave type in medium two

a = setup.trans.d/2;            % transducer radius (mm)
Fl = setup.trans.fl;            % transducer focal length (mm)

z1 = setup.geom.z1;             % water path length (mm)
z2 = setup.geom.z2;             % path length in solid (mm)
x2 =setup.geom.x2;              % distance (mm) from ray axis in POI
y2 = setup.geom.y2;             % distance (mm) perpendicular to the POI
Rx = setup.geom.R1;             % interface radius of curvature (mm) in POI
Ry =setup.geom.R2;          % interface radius of curvature (mm) out of POI
iang = setup.geom.i_ang;        % incident angle (deg)

d1 = setup.matl.d1;             % density (fluid)
d2 =setup.matl.d2;              % density (solid)
cp1 = setup.matl.cp1;           % compressional wave speed -fluid  (m/sec)
```

```
cp2 = setup.matl.cp2;                % compressional wave speed -solid (m/sec)
cs2 = setup.matl.cs2;                % shear wave speed -solid (m/sec)

[A, B] = gauss_c15;                  % Wen and Breazeale coefficients (15)

% update setup.wave wave speeds
if strcmp(type1, 'p')
   setup.wave.c1 =cp1;
elseif strcmp(type1, 's')
   setup.wave.c1 = cs1;
else
   error('wrong wave type (must be p or s) ')
end

if strcmp(type2, 'p')
   setup.wave.c2 =cp2;
elseif strcmp(type2, 's')
   setup.wave.c2 = cs2;
else
   error('wrong wave type (must be p or s)')
end
% calculate transmission coefficient, update setup
setup.wave.T12 = fluid_solid(setup);

% wave speeds and transmission coefficient for the beam model
c1 =setup.wave.c1;
c2 =setup.wave.c2;                   % wave speed for wave type2
T = setup.wave.T12;                  % transmission coefficient

% parameters appearing in beam model

cosi = cos(pi*iang/180);             % cosine of incident angle
sinr = (c2/c1)*sin(pi*iang/180);        % sine of refracted angle from Snell's law
if sinr >= 1
  error('Beyond the Critical angle')     % no transmitted wave of given wave type
else
  cosr = sqrt( 1 - sinr^2);
end

  h11 = 1/Rx;  %curvature
  h22 = 1/Ry;  %curvature
zr = eps*(f == 0) + 1000*pi*(a^2)*f./c1; % "Rayleigh" distance
k1 = 2*pi*1000*f./c1;                  % wave number in fluid

%initialize predicted velocity with zeros of a size
% compatible with largest array in f, z1, z2, x2, y2 parameters
```

```
v = init_z(setup);

%multi-Gaussian beam model

for j = 1:15                        % form up multi-Gaussian beam model

 b =B(j) + i*zr./Fl;                % modify coefficients for focused probe
                                    % Fl = inf for planar probe

q = z1 - i*zr./b;
K = q.*(cosi -(c1/c2)*cosr);
M1 = (cosi^2 +K.*h11)./cosr^2;
M2 =1 + K.*h22;
ZR1 = q./M1;
ZR2 =q./M2;
m11 = 1./(ZR1 +(c2/c1).*z2);
m22 = 1./(ZR2 +(c2/c1).*z2);
  t1 = A(j)./(1 + (i.*b./zr).*z1);
  t2 = t1.*T.*sqrt(ZR1).*sqrt(ZR2).*sqrt(m11).*sqrt(m22);
  v = v + t2.*exp(i.*(k1./2).*(m11.*(x2.^2) + m22.*(y2.^2)));

end
```

As a simple test of this multi-Gaussian beam model we can use the default setup structure to simulate the on-axis wave field of a 5MHz, 12.7 mm diameter circular piston transducer radiating into water. The following MATLAB commands will generate the plot shown in Fig. 12.2:

```
>> setup = setup_maker;
>> [v, setup] = MGbeam(setup);
>> z2 =setup.geom.z2;
>> plot(z2, abs(v))
>> xlabel('z-distance (mm)')
>> ylabel('|v/v_0|')
```

As seen in Fig. 12.2 the beam model accurately predicts the near-field of the transducer down to a distance of approximately a transducer diameter, as discussed in Chapter 9.

Other plots also easy to simulate. From Fig. 12.2 we see that there is an on-axis null near z2 = 70 mm, so we can examine the cross-axis behavior at that distance through the commands:

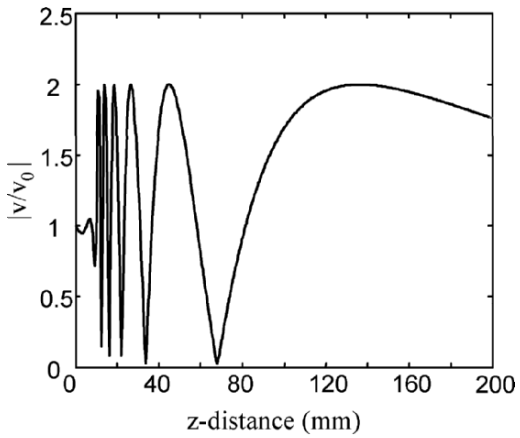

Fig. 12.2. The on-axis field of a 5 MHz, 12.7 mm diameter circular piston transducer radiating into water as calculated with a multi-Gaussian beam model.

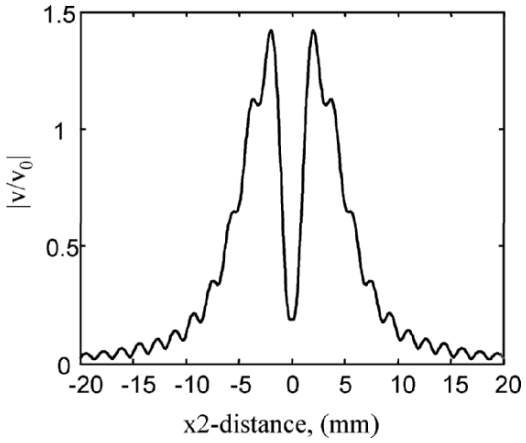

Fig. 12.3. The wave field in a plane perpendicular to the axis of a 5 MHz, 12.7 mm diameter planar piston transducer radiating into water at a distance approximately equal to one-half a near field distance along the axis.

```
>> setup.geom.z2 =70;
>> x2 = linspace(-20,20, 512);
>> setup.geom.x2 = x2;
>> [v, setup] = MGbeam(setup);
>> plot(x2, abs(v))
>> xlabel('x2-distance, (mm)')
>> ylabel(' | v/v_0 |')
```

The results are shown in Fig. 12.3. In a similar fashion we can see a 2-D cross-section of the entire wave field with the commands:

```
>> %  recall, we already had set x2 = linspace(-20,20, 512);
>> z2 = linspace(0, 200, 512);
>> [zz, xx] =meshgrid(z2, x2);
>> setup.geom.z2 = zz;
>> setup.geom.x2 = xx;
>> [v, setup] = MGbeam(setup);
>> image(z2, x2,abs(v)*50)          % scale the result to get a good
                                    % color map

>> xlabel('z2-distance (mm)')
>> ylabel('x2-distance (mm)')
```

The results are shown in Fig. 12.4.

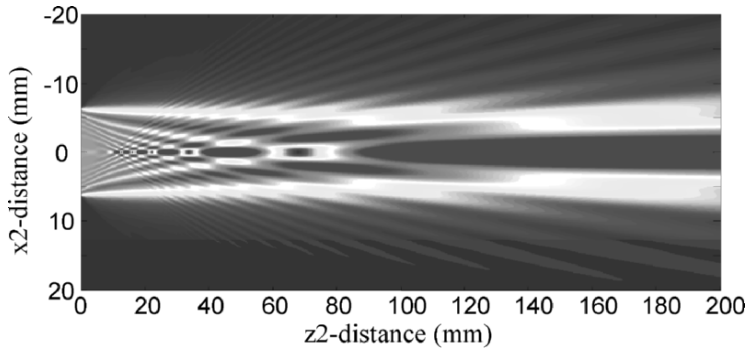

Fig. 12.4. A 2-D image of the near-field beam profile for a 5 MHz, 12.7 mm diameter planar piston transducer radiating into water. Note the scales on the two axes are very different.

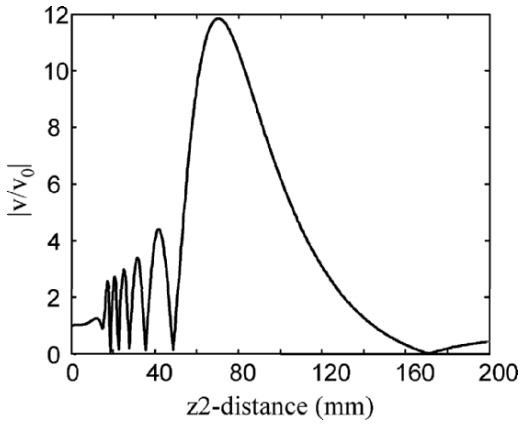

Fig. 12.5. The on-axis wave field of a 10 MHz, 12.7mm diameter, 76.2mm focal length focused transducer radiating into water as calculated with a multi-Gaussian beam model.

To simulate a spherically focused probe and examine the on-axis response, consider a 10 MHz, 12.7 mm diameter, 76.2 mm focal length transducer radiating into water. This can be simulated via the commands:

```
>> setup.f =10;
>> setup.geom.x2 =0.;
>> setup.geom.z2 =z2;  % put a vector set of values back into setup
>> setup.trans.fl = 76.2;
>> [v, setup] = MGbeam(setup);
>> plot(z2, abs(v))
>> xlabel('z2-distance (mm)')
```

The results are shown in Fig. 12.5. Note that we changed the frequency of the calculation by changing the setup.f parameter, not the setup.system.fc (center frequency) parameter. The center frequency parameter refers to a parameter of the frequency profile of the system function which is needed to synthesize a time domain waveform. This center frequency parameter will not affect beam calculations performed at a single frequency. To synthesize a transducer pulse, however, we would have to let setup.f be an array of frequencies and multiply the output of MGbeam function by a system function to simulate the spectral behavior of the system. We will show simulation examples of this type later.

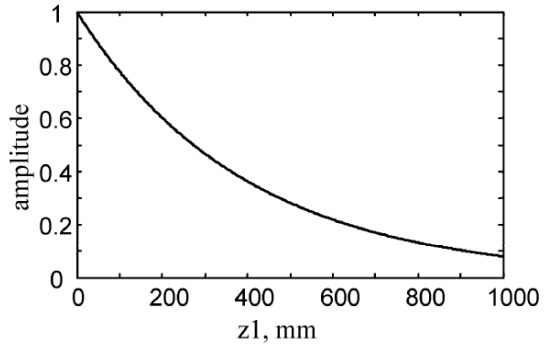

Fig. 12.6. The attenuated amplitude versus distance for propagation in water at room temperature and at a frequency of 10 MHz.

12.5 Ultrasonic Attenuation in the Measurement Model

Ultrasonic material attenuation is a part of the measurement model which must be determined experimentally. The linear attenuation terms appearing in the attenuation expression $\exp\left[-\alpha_{p1}(\omega)z_1 - \alpha_{\gamma 2}(\omega)z_2\right]$ are frequency dependent so that normally one fits the measured values of these linear attenuation terms to functions with a simple frequency dependency (linear, quadratic, etc.) that best match the experimental results over the bandwidth of the measurement system. The MATLAB function attenuate in Code Listing 12.6 defines each of the linear attenuation coefficients for the appropriate wave types traveling in medium one and two in terms of five fitting coefficients for a polynomial of up to fourth order in frequency, i.e. we use a fitting expression for an attenuation coefficient α in the form $\alpha = a_1 + a_2 f + a_3 f^2 + a_4 f^3 + a_5 f^4$. Those fitting coefficients must be placed in setup.matlp1, setup.matls1, setup.matlp2, and setup.matls2 .

Code Listing 12.6. A MATLAB function for calculating attenuation terms for propagation in two adjacent media.

```
function y = attenuate(setup)
% atten(setup) generates a frequency dependent attenuation factor
% as a function of the frequency, f, and the distances z1, z2 in (mm)
% traveled in two media
% For water at room temp for the first medium , take p1(1) = p1(2) = p1(4)
% =p1(5)=0,
% and p1(3) = 25.3E-06 if f is in MHz, distances are in mm

f=setup.f;
type1=setup.type1;
type2=setup.type2;
z1 =setup.geom.z1;
z2 =setup.geom.z2;
p1 =setup.matl.p1;
s1 =setup.matl.s1;
p2=setup.matl.p2;
s2=setup.matl.s2;
if strcmp(type1, 'p')
        a1 =p1;
elseif  strcmp(type1, 's')
        a1 =s1;
else
error('wrong wave type')
end

if strcmp(type2, 'p')
        a2 =p2;
elseif  strcmp(type1, 's')
        a2 =s2;
else
error('wrong wave type')
end

d1 = a1(1) + a1(2)*f + a1(3)*f.^2  + a1(4)*f.^3  + a1(5)*f.^4;
d2 = a2(1) + a2(2)*f + a2(3)*f.^2  + a2(4)*f.^3  + a2(5)*f.^4;

y = exp(-d1.*z1).*exp(-d2.*z2);
```

To illustrate this function, consider the attenuated amplitude versus distance in water at room temperature for a frequency of 10 MHz where the attenuation coefficient is $\alpha = 25.3 \times 10^{-6} f^2$ with f the frequency in MHz. Using the default setup structure and the MATLAB commands:

```
>> setup.f =10.;
>> z1 =linspace(0,1000,512);
>> setup.geom.z1  = z1;
>> setup.geom.z2 =0.0;
>> setup.matl.p1 = [ 0  0  25.3E-06  0  0];
>> y=attenuate(setup);
>> plot(z1, y)
>> xlabel('z1, mm')
>> ylabel('amplitude')
```

we obtain the plot show in Fig. 12.6 (the default type1 ='p' here and the other attenuation fitting coefficients are all zero).

12.6 The System Function

The system function, $s(\omega)$, is found in practice by either performing a measurement of the received voltage in a calibration setup or by measuring all the ultrasonic system components in the sound generation and reception processes and combining them to form up the $s(\omega)$, as described in previous Chapters. However, we can also simulate this function directly to model its effects on the measurement process.

To model the system function we will use a simple Gaussian function of the type discussed in Appendix A given by

$$F(f) = A\exp\left[-4\pi^2 a^2 (f - f_c)^2\right] = A\exp\left[-a^2(\omega - \omega_c)^2\right], \qquad (12.22)$$

where A is the amplitude, $f = 2\pi\omega$ is the frequency and f_c is the center frequency, both measured in MHz. The inverse Fourier transform of this function can be obtained analytically as

$$f(t) = \frac{A}{2a\sqrt{\pi}}\exp(-2\pi i f_c t)\exp(-t^2/4a^2), \qquad (12.23)$$

which is complex since we have not included any negative frequency components in $F(f)$. As shown in Appendix A we can recover a real time domain signal, $v(t)$, from only the positive frequency components if we take twice the real part of Eq. (12.23) which gives

$$v(t) = \frac{A}{a\sqrt{\pi}} \cos(2\pi f_c t) \exp\left(-t^2 / 4a^2\right). \tag{12.24}$$

In all the model terms in our measurement models, we will likewise only model those terms for positive frequencies and then take twice the real part of the result to recover real time domain functions.

It is convenient to rewrite $F(f)$ in a form which is parameterized not in terms of *a* but instead in terms of the bandwidth, *bw*, where *bw* is the width of the Gaussian, in MHz, where its amplitude is one-half of its maximum value (see Fig. A.5). This gives

$$\exp\left[-4\pi^2 a^2 \left(f_0 - f_c\right)^2\right] = \exp\left[-\pi^2 a^2 \left(bw\right)^2\right] = \frac{1}{2} \tag{12.25}$$

so solving for *a* in terms of *bw* we find

$$a = \frac{\sqrt{\ln 2}}{\pi\, bw}. \tag{12.26}$$

For small center frequencies and large bandwidths, the simple Gaussian function in Eq. (12.22) will have a non-zero D.C. (zero frequency) component. Most transducers band limit the measured ultrasonic response so that the response should be very small at low frequencies. To model this behavior we therefore modify the Gaussian slightly through a sine function that tapers the response to zero at zero frequency. Thus, the simulated system transfer factor, $s(f)$, we will model is given by

$$s(f) = \begin{cases} F(f) \sin\left[\dfrac{\pi f}{2 f_c}\right] & f < f_c \\ F(f) & f \geq f_c \end{cases}. \tag{12.27}$$

This modification means that the corresponding time domain waveform will not be given exactly by Eq. (12.24) but in many cases the difference is small. The MATLAB function in Code Listing 12.7 returns the system function given in Eq. (12.27):

Code Listing 12.7. A MATLAB function for simulating the system function.

```
function y = systf (setup)
% systf(setup) models the system function by a Gaussian window function
% of amplitude amp centered at frequency fc and with a bandwidth bw defined to
% be the spread in frequency at the half amplitude point in the Gaussian.
% The Gaussian is tapered to zero at frequencies below fc with a sine function to
% guarantee the dc value is always zero.
% For small fc and large bw, this tapering will distort the Gaussian
%
f =setup.f;
amp = setup.trans.amp;
fc = setup.trans.fc;
bw = setup.trans.bw;
a = sqrt(log(2))/(pi*bw);
s1 = exp(-(2*a*pi*(f - fc)).^2).*(f > fc);
s2 = exp(-(2*a*pi*(f - fc)).^2).*sin(pi*f/(2*fc)).*(f <= fc);
y = amp*(s1 + s2);
```

To illustrate this function we can use the default setup structure where amp = .05 volts/MHz, fc = 5 MHz, and bw = 3 MHz with the commands:

```
>> f = linspace(0, 20, 512);
>> setup.f = f;
>> y=systf(setup);
>> plot(f, y)
>> xlabel( ' f, MHz')
>> ylabel('volts/MHz')
```

to obtain the system function shown in Fig. 12.7. Note that the system function modeled here is a purely real function. A measured system function, however, will generally be a complex-valued function.

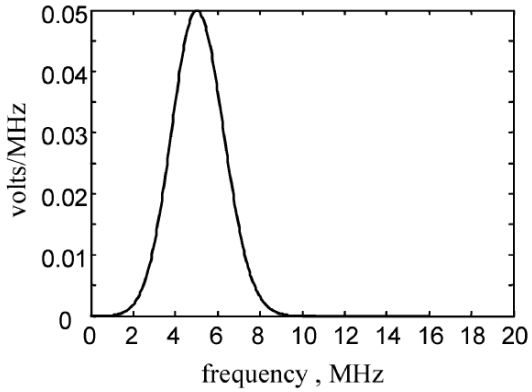

Fig. 12.7. A simulated system function.

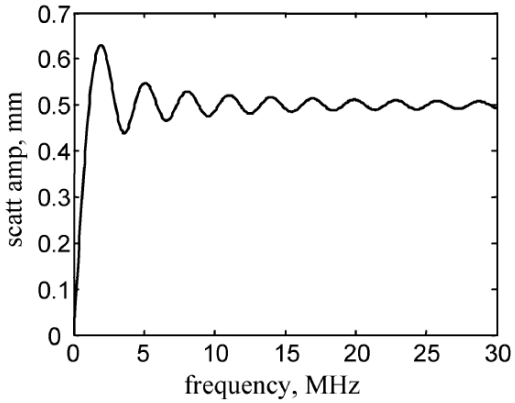

Fig. 12.8. The pulse-echo far-field scattering amplitude versus frequency for a 1 mm radius spherical void in steel, calculated using the Kirchhoff approximation.

12.7 Flaw Scattering Models

As shown in Chapter 10, the Kirchhoff approximation is a very useful approximation for obtaining the flaw scattering properties of a number of flaws. We will develop MATLAB functions that will use the Kirchhoff

approximation for modeling the pulse-echo far-field scattering amplitude of a spherical void and a circular crack. The explicit expressions for these scattering amplitudes were given in Chapter 10. For the spherical void of radius b we found (Eq. (10.14)):

$$A\left(\mathbf{e}_i^{\beta};-\mathbf{e}_i^{\beta}\right)=\frac{-b}{2}\exp\left(-ik_{\beta}b\right)\left[\exp\left(-ik_{\beta}b\right)-\frac{\sin\left(k_{\beta}b\right)}{k_{\beta}b}\right], \tag{12.28}$$

while for wave incident on a circular crack of radius b at an angle, θ, with respect to the crack normal we found (Eq. (10.36)):

$$A\left(\mathbf{e}_i^{\beta};-\mathbf{e}_i^{\beta}\right)=\frac{ib\cos\theta}{2\sin\theta}J_1\left(2k_{\beta}b\sin\theta\right). \tag{12.29}$$

Code Listing 12.8 describes the function A_void that uses Eq. (12.28) and returns the pulse-echo scattering amplitude of the spherical void.

Code Listing 12.8. A MATLAB function for modeling the pulse-echo far-field scattering amplitude of a spherical void.

```
function A = A_void(setup)
% A_VOID calculates the pulse-echo far-field scattering amplitude
% of a spherical void in the Kirchhoff approximation, using
% the frequency f in setup.f, the radius b in  setup.flaw.b,
% and the wave speed for the wave type in setup.wave.c2.
% The calling sequence is A = A_void(setup). The scattering
% amplitude, A, (in mm) is returned.

%get the parameters
f =setup.f;
c = setup.wave.c2;
b = setup.flaw.b;

%calculate the wave number kb (f in MHz, b in mm, c in m/sec)
kb = (2000*pi*b*f)./c;

%calculate the pulse-echo scattering amplitude
kb = kb + eps*(kb == 0);   % prevent division by zero
A =(-b/2)*exp(-i*kb).*(exp(-i*kb)-sin(kb)./(kb));
```

Similarly, Code Listing 12.9 gives the MATLAB function A_crack that uses Eq. (12.29) and returns the pulse-echo scattering amplitude of the circular crack.

Code Listing 12.9. A MATLAB function for modeling the pulse-echo far-field scattering amplitude of a circular crack.

```
function A = A_crack(setup)
% A_CRACK calculates the pulse-echo far-field scattering amplitude
% of a circular crack in the Kirchhoff approximation, using the
% frequency f in setup.f, the radius b in setup.flaw.b, the acute
% angle between the incident wave direction and the crack normal in
% setup.flaw.f_ang, and the wave speed for the wave type in
% setup.wave.c2.
% The calling sequence is A = A_crack(setup). The
% scattering amplitude,A, (in mm) is returned.

%get the parameters
f = setup.f;
c = setup.wave.c2;
b = setup.flaw.b;
ang = setup.flaw.f_ang;

% put the angle in radians, calculate the wave number
iang = ang.*pi./180;
kb = (2000*pi*b*f)./c;

% calculate the pulse-echo scattering amplitude
arg = 2*sin(iang).*kb;       % argument of bessel function
arg = arg + eps*(arg == 0);  % prevent division by zero
A = i*kb.*b.*cos(iang).*(besselj(1, arg)./arg);
```

We can use these functions to verify some of the results presented in Chapter 10. First, consider the pulse-echo frequency domain response of a 1 mm radius spherical void in steel (c_{p2} = 5900 m/sec). Using the commands:

```
>> clear
>> setup=setup_maker;
>> setup.f =linspace(0,30,512);
```

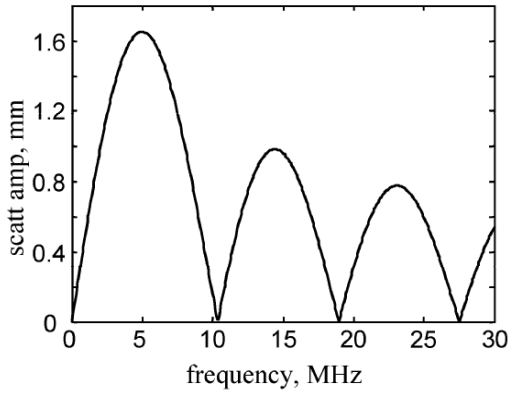

Fig. 12.9. The pulse-echo far-field scattering amplitude versus frequency for a 1 mm radius circular crack in steel, calculated using the Kirchhoff approximation. The incident angle $\theta = 10^{\circ}$ with respect to the crack normal.

```
>> setup.wave.c2 =5900;
>> setup.flaw.b =1.;
>> setup.flaw.Afunc ='A_void';
>> f = setup.f;
>> A = feval(setup.flaw.Afunc, setup);
>> plot(f, abs(A))
>> xlabel('frequency, MHz')
>> ylabel('scatt amp, mm')
```

generates the plot shown in Fig. 12.8 which is identical to Fig. 10.6. Notice that we put the frequencies and wave speed into the appropriate parameters in setup and we have placed the name of the flaw function in the setup structure and then retrieved it to evaluate it with the function feval. This process was done simply to illustrate how in a measurement model the setup structure will be used to obtain the flaw response. In this case we could have just called the function A_void directly with setup as its argument.

The same type of pulse-echo response for a 1 mm radius crack in steel where the incident direction is at 10^{o} from the crack normal can be found using the same setup parameters just defined plus the commands

```
>> setup.flaw.f_ang = 10;
>> setup.flaw.Afunc ='A_crack';
>> Ac =feval(setup.flaw.Afunc, setup);
```

```
>> plot(f, abs(Ac))
>> xlabel('frequency, MHz')
>> ylabel('scatt amp, mm')
```

These commands generate the plot shown in Fig. 12.9 which is identical to the same plot shown in Fig. 10.17.

12.8 The Thompson-Gray Measurement Model

We now have all the MATLAB functions defined that will allow us to construct a complete ultrasonic measurement model of the type given in Eq.(12.6) where the flaw is assumed to be small enough so that we can neglect the beam variations over the flaw surface. Thompson and Gray first developed this type of measurement model in 1983 [11.2]. The MATLAB function TG_PE_MM (Code Listing 12.10), like all our other functions uses only the setup structure as its input. TG_PE_MM returns an updated setup structure and the measured voltage, V_R, in the frequency domain obtained from a flaw in the solid using the Thompson-Gray measurement model for a pulse-echo immersion setup of the type shown in Fig 12.1. The multi-Gaussian beam model function MGbeam is used to predict the transducer velocity field at the flaw and the far-field scattering amplitude is obtained by the MATLAB function whose name is specified in the setup parameter setup.flaw.Afunc. The system function is modeled by the MATLAB function systf if the setup.sysf contains the string 'systf' (the default) or this function is obtained experimentally by use of the function whose name is contained in setup.sysf. The attenuation of the materials in the measurement model is accounted for by the MATLAB function attenuate.

Code Listing 12.10. The MATLAB function TG_PE_MM for modeling the response of a flaw using the Thompson-Gray ultrasonic measurement model.

```
function [Vf, setup] =TG_PE_MM(setup)
% TG_PE_MM generates the frequency components of the
% output voltage, Vf, of an ultrasonic pulse-echo immersion
% measurement system generated by a flaw.
% The function returns Vf  as well as an updated setup structure
% The calling sequence is [Vf, setup] =TG_mm(setup);

% First, compute the incident beam velocity and update
```

```
% the setup structure
[v, setup] = MGbeam(setup);

%get the setup parameters  needed for the constant term
%in the measurement model
f = setup.f;
r= setup.trans.d/2;    % transducer radius
d1 =setup.matl.d1;
d2 =setup.matl.d2;
c1 = setup.wave.c1;
c2 = setup.wave.c2;

%compute wave number in medium two and
%the constant term in the measurement model

k2 = (2000.*pi.*f)./c2;
k2 =k2 + eps*( k2 == 0);  % prevent division by zero
K= (4.*d2.*c2)./(-i.*k2.*r^2.*d1.*c1);

% check to see if a model-based or experimentally determined system
% function is to be used
if strcmp(setup.sysf, 'systf')
   sys = systf(setup);
else
   sys =feval(setup.sysf, setup);
end

% find flaw type to be used
if strcmp( setup.flaw.Afunc, 'empty')
   error('flaw function not specified in setup')
else
   A = feval(setup.flaw.Afunc, setup);
end

%compute output voltage, Vf, (volts/MHz)
Vf = sys.*(v.*attenuate(setup)).^2.*A.*K;
```

To illustrate an application of the MATLAB function TG_PE_MM we will describe a MATLAB calculation that uses the setup shown in Fig. 12.10 (b), where a planar, 5 MHz transducer is being used in pulse-echo to examine a spherical 0.6921 mm diameter void in a glass block at normal incidence through a water-solid interface. These parameters are

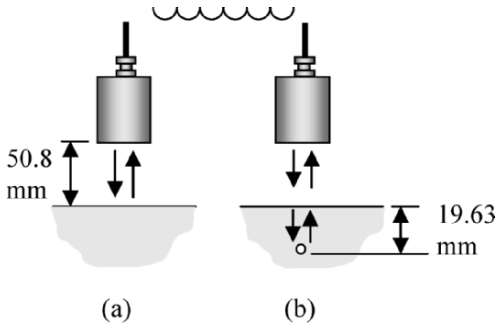

Fig. 12.10. (a) A reference scattering configuration where a planar 12.7 mm diameter, 5 MHz transducer receives the P-waves reflected from a water-glass interface. **(b)** A pulse-echo flaw measurement setup where the transducer in (a) receives the P-waves scattered from a 0.6921 mm diameter spherical void in glass located on the central axis of the transducer. The water path length is the same (50.8 mm) in both measurements.

similar to those of a experimental setup that we will discuss next. We will simulate the received voltage time-domain waveform from the void. If we call the function setup_maker then we need to change only those parameters that are different from the default setup structure that is generated by this function. In this case we will set up a range of frequencies from 0 to 20 MHz to do our calculations and define the measured wave speed of the water (the water density was taken as the default value of 1.0) and also the density and wave speed of the glass:

```
>> setup = setup_maker;
>> f = s_space(0, 20, 200);
>> cp1 = 1484;
>> d2 = 2.2;
>> cp2 = 5969.4;
>> cs2 = 3774.1;
```

The MATLAB function s_space (xmin, xmax, num) used here (the MATLAB code listing is given in Appendix G) is similar to the MATLAB function linspace. The s-space function gives a set of num evenly spaced sampled values from xmin to xmax - dx, where dx = (xmax - xmin)/num is the sample spacing, whereas the MATLAB function linspace(xmin, xmax, num) gives set of num evenly sampled values from xmin to xmax with sample spacing dx = (xmax – xmin)/(num-1). As discussed in Appendix A

the function s_space generates precisely the sampled values needed in both the time and frequency domains to perform Fourier analysis with FFTs, but the built-in MATLAB function linspace does not.

We will also specify the water path length from the transducer to the interface and distance from the interface to center of the spherical void in the solid (see Fig. 12.10 (b)):

```
>> z1 = 50.8;
>> z2 = 19.62725;
```

The default system function center frequency of 5 MHz can be left unchanged but the system function amplitude and bandwidth will be chosen to be similar to the experimental example we will discuss shortly:

```
>> amp = 0.08;
>> bw = 4;
```

Although in this example the parameters amp and bw are the only values needed to predict the system function, when we determine this function experimentally we will also need to specify the water path length to be used in a reference experiment so that anticipating the need for that variable, we will also set it appropriately here:

```
>> z1r = 50.8;
```

The transducer diameter (12.7 mm) and focal length (infinity) are compatible with the default values generated by setup_maker. The attenuation of the glass block is very small so that it will be neglected. The P-wave attenuation of the water is included as a quadratic function of frequency:

```
>> p1 = [ 0 0 .02479E-03 0 0];
```

Finally, the flaw radius is specified and the name of the function that calculates the pulse-echo far-field scattering amplitude of a spherical void in the Kirchhoff approximation is given:

```
>> b = .34605;
>> flaw_name = 'A_void';
```

All of the other default setup parameters can be used unchanged so it is only necessary to update these parameters:

```
>> setup.f = f;
>> setup.trans.amp = amp;
>> setup.trans.bw = bw;
>> setup.z1r = z1r;
>> setup.geom.z1 =z1;
>> setup.geom.z2 =z2;
>> setup.matl.cp1 = cp1;
>> setup.matl.d2 = d2;
>> setup.matl.cp2 = cp2;
>> setup.matl.cs2 =cs2;
>> setup.matl.p1 = p1;
>> setup.flaw.b =b;
>>setup.flaw.Afunc =flaw_name;
```

With these changes then the output voltage in the frequency domain, Vf, and an updated setup structure can be calculated:

```
>> [Vf, setup] = TG_PE_MM(setup);
```

If we want to examine the time-domain waveform from the void, we must extend the maximum frequency beyond the 20 MHz value used in the calculations and zero pad the Vf values. Here we have extended the maximum frequency to 100 MHz, using the same frequency spacing, df, used in calculating Vf. The sampling time interval, dt, is then the reciprocal of this max frequency, and we can use this time interval to generate a time window, t. Since we are only going to use the positive frequency components of the response to calculate the wave form, we have also divided the zero frequency value of Vf by two:

```
>> df = f(2) - f (1);
>> dt = 1/(1000*df);
>> t= s_space(0,1000*dt , 1000);
>> Vfe = [ Vf  zeros(1, 800)];
>> Vfe(1) = Vfe(1)/2;
```

We are now able to calculate the time domain void response with an inverse FFT of these positive frequency components:

```
>> vt = 2*real(IFourierT(Vfe, dt));
```

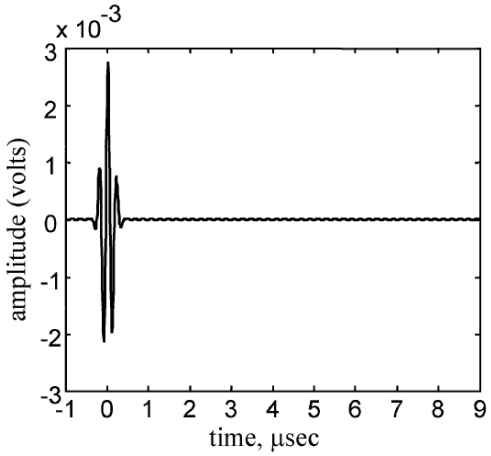

Fig. 12.11. The simulated response pulse-echo P-wave response of a spherical void for the setup shown in Fig. 12.10 (b).

and we can plot the result. Since we have omitted all the time delay terms in these calculations, t = 0 corresponds to when the waves reach the center of the flaw so that we need to use the t_shift and c-shift functions to obtain a result where the responses before t = 0 are not in the upper part of the window:

```
>> plot(t_shift (t, 100), c_shift(vt,100))
```

The simulated wave form (in volts) is shown in Fig. 12.11. All of the above MATLAB commands are contained in the MATLAB script TG_sphere_example1(Code Listing 12.11). This simple example shows how one can use the MATLAB functions to model a flaw response where the system function was taken to be the simple Gaussian function described previously.

Code Listing 12.11. A MATLAB script for calculating the pulse-echo response of an on-axis pore at normal incidence through a fluid-solid interface.

```
% TG_sphere_example1 script
% This script calculates the pulse-echo P-wave response of an on-axis
% spherical pore interrogated by a 5 MHz planar probe through a
% fluid-solid interface at normal incidence
clear
setup = setup_maker;
```

```
% setup parameters that need to be specified for this example
f =s_space(0, 20, 200);
cp1 = 1484.;
d2 = 2.2;
cp2 = 5969.4;
cs2 = 3774.1;
z1 = 50.8;
z2 = 19.62725;
amp =0.08;
bw = 4.;
z1r =50.8;
p1 = [ 0 0 0.02479E-03  0 0];
b =0.34605;
flaw_name = 'A_void';
setup.f =f;
setup.system.amp = amp;
setup.system.bw = bw;
setup.system.z1r =z1r;
setup.geom.z1 = z1;
setup.geom.z2 = z2;
setup.matl.cp1 = cp1;
setup.matl.d2 = d2;
setup.matl.cp2 = cp2;
setup.matl.cs2 = cs2;
setup.matl.p1 = p1;
setup.flaw.b = b;
setup.flaw.Afunc = flaw_name;
% calculate received voltage
[Vf, setup] = TG_PE_MM(setup);
% extend frequency components to permit
% taking FFT
df = f(2)-f(1);
dt = 1/(1000*df);
t = s_space(0, 1000*dt, 1000);
Vfe = [Vf zeros(1,800)];
Vfe(1) = Vfe(1)/2;
vt =2*real(IFourierT(Vfe, dt));
plot(t_shift(t,100), c_shift(vt,100))
```

As shown in Chapter 7, it is relatively easy to calculate the system function experimentally in a reference experiment, and this function then truly represents the effects of all the electrical and electromechanical compo nents of the system (pulser/receiver, cabling, transducers) at a specific

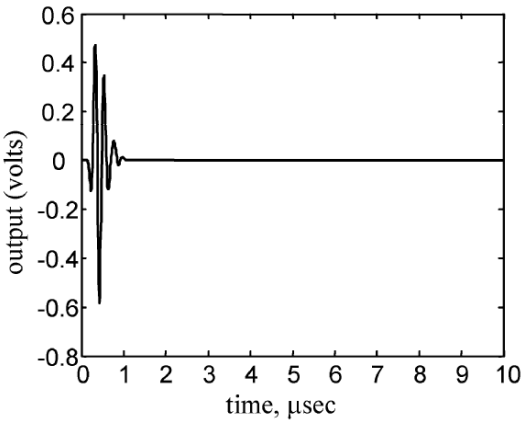

Fig. 12.12. The voltage received from the fluid-solid interface for the reference scattering configuration shown in Fig. 12.10 (a).

set of instrument settings. It is also easy to incorporate such a measured system function into our measurement model. All that is needed is to replace the output of the systf function in the previous example with a compatible set of measured values of the system function. This can be done for the example just discussed by measuring the waves received from the front surface of the glass block, as shown in Fig. 12.10(a). Since the acoustic/elastic transfer function is known for this configuration, deconvolution (with the aid of a Wiener filter) of the frequency components of the measured response by the transfer function, as shown in Chapter 7, will give us the measured system function. Figure 12.12 shows the experimental wave form received by a 5 MHz, 12.7 mm diameter planar transducer from the interface as shown in Fig. 12.10 (a). The 1000 point wave form and its corresponding time axis are stored as MATLAB variables ref and t_ref, respectively in the MATLAB MAT-file sphere_ref.mat. The MATLAB function exp_systf is used in place of the model-based systf function to calculate the system function experimentally. The function exp_systf loads the ref and t_ref variables into MATLAB (assuming that the sphere_ref file is contained in the current MATLAB directory), computes the frequency components of this measured response and then deconvolves those components with the acoustic/elastic transfer function for this configuration, using the MATLAB function Wiener_filter defined in Appendix C with a noise constant defined by the parameter setup.system.en. The function then

returns the measured system function. The listing of exp_systf is given in Code Listing 12.12.

Code Listing 12.12. A function for calculating the system function from an experimentally measured wave form in the reference scattering configuration of Fig. 12.10 (a).

```
function s = exp_systf(setup)
% EXP_SYSTF generates the system function from the
% measured voltage received by a circular, planar or focused
% transducer from the planar front surface of a
% solid. It is assumed that the solid is the same as the one
% in the flaw measurement where this system function is to be used
% as is the rest of the measurement setup except that the fluid
% path length can be different from the one used in a flaw measurement.
% This function assumes that there are 1000 sampled
% values in the reference wave form and time axis
% and the sampling frequency is 100MHz
filename =setup.system.ref_file;
load(filename) % load reference wave form (in the variable ref)
% and the time axis values (in the variable  t_ref) from a MAT-file

dt = t_ref(2)-t_ref(1);
% calculate Fourier Transform
V =FourierT(ref, dt);
%generate frequency axis
fs = s_space(0, 1/dt, 1000);

% get setup frequency values and check for consistency
f = setup.f;
df= f(2) - f(1);
dfs =fs(2) - fs(1);
fsize=size(f);
numf = fsize(2);
if df > (dfs + .001) | df < (dfs - .001)
   error('frequency spacing mismatch of setup and exp values')
end
if f(end) > (fs(end) +dfs)/2
   error('max frequency in setup exceeds Nyquist')
end
% keep number of measured voltage frequency components
% compatible with that in setup
Vc=V(1:numf);
% get remaining setup parameters
```

```
z1r =setup.system.z1r;
en =setup.system.en;
d1 =setup.matl.d1;
cp1 = setup.matl.cp1;
d2 = setup.matl.d2;
cp2 = setup.matl.cp2;
cs2 = setup.matl.cs2;
a = setup.trans.d/2;
p1 =setup.matl.p1;
alphac =p1(3); % frequency squared attenuation coefficient
fl = setup.trans.fl;

% if transducer is focused, z1r must be the same as the focal length
if fl ~= inf
   if z1r > fl +.01  | z1r < fl - .01
      warning(' reference water path is not the focal length, using focal length')
      z1r = fl;
   end
end

% calculate wave number , reflection coefficient of fluid-solid interface
% and argument for acoustic/elastic transfer function
ka =2000.*pi.*f.*a./cp1;
R12 = (cp2*d2 - cp1*d1)/(cp2*d2 + cp1*d1);
arg = (a/z1r)*ka;
alpha = alphac*f.^2;
% calculate acoustic-elastic transfer function, leave out propagation phase

ta = 2*R12*exp(-2*alpha.*z1r).*(1 -exp(i*arg/2).*(BesselJ(0, arg/2)...
     -i*BesselJ(1, arg/2)));
if fl ~= inf
   ta = -conj(ta);
end

% deconvolve measured voltage frequency components with transfer function
% to get system function
s = Wiener_filter(Vc, ta, en);
```

To use exp_systf for our spherical void example in place of the function systf which generates a model-based system function, we need only have the appropriate setup parameters, which can be obtained by first running the script TG_sphere_example1, and then updating setup.sysf to indicate we now are going to use an experimentally determined system function.

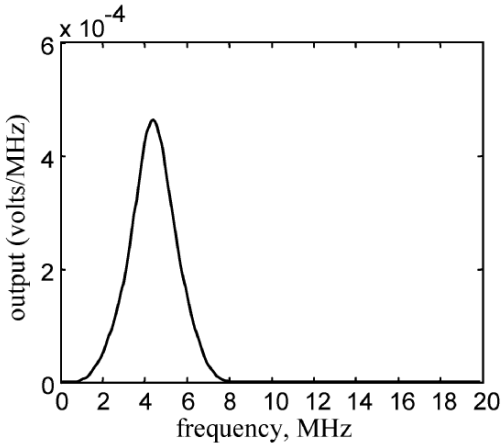

Fig. 12.13. The magnitude of the frequency components of the voltage received from an on-axis spherical void for the configuration shown in Fig. 12.10 (b) as predicted by the Thompson-Gray measurement model using an experimentally determined system function and the Kirchhoff approximation for the far-field scattering of the void.

The Wiener filter constant, en, is set at a default value of 0.01 in the setup parameters but it can be changed, if necessary. We also need to specify the MAT-file that contains the reference wave form obtained from the configuration in Fig. 12.10 (a). Note that the distance z1r has already been defined appropriately.

```
>> clear
>> TG_sphere_example1
>> setup.system.sysf = 'exp_systf';
>> setup.system.ref_file = 'sphere.ref';
```

Then we can run the measurement model and plot the output:

```
>> [Vout, setup] = TG_PE_MM(setup);
>> plot(f, abs(Vout))
```

The results are shown in Fig. 12.13. If we now pad these frequency domain values with zeros to extend the frequency range to 100 MHz and do an inverse FFT, the time domain received wave form can be plotted:

```
>> Ve =[Vout, zeros(1, 800)];
```

```
>> Ve(1) = Ve(1)/2 ;
>> vt = 2*real(IFourierT(Ve, dt));
>> plot(t, vt)
```

The results are shown in Fig. 12.14. All of the MATLAB commands needed to generate this waveform are in the MATLAB script TG_sphere_example2 (see Code Listing 12.13). The intermediate frequency plot of Fig. 12.13, however, is omitted in that script.

Code Listing 12.13. A MATLAB script for calculating the A-scan wave form for a spherical void using an experimentally determined system function.

```
% script TG_sphere_example2
% calculates the waveform for a spherical void
% using an experimentally determined system function
clear
% run TG_sphere_example1 script to get system parameters
TG_sphere_example1
%specify use of experimentally determined system function
%and reference waveform
setup.system.sysf='exp_systf';
setup.system.ref_file ='sphere_ref';
%run measurement model
[Vout, setup] = TG_PE_MM(setup);
% plot(f, abs(Vout))  intermediate plot omitted
% pad frequency domain amplitude with zeros
Ve= [ Vout, zeros(1,800)];
Ve(1) = Ve(1)/2;   % Now, compute wave form and plot
vt =2*real(IFourierT(Ve, dt));
plot(t, vt)
```

For comparison, the actual measured wave form from the flaw can also be plotted. This wave form, vexp, and its corresponding time axis, t_exp, are contained in the file sphere_flaw.mat. We can load that file and display that flaw signal on the same plot as the one just obtained:

```
>> hold on
>> load 'sphere_flaw'
>> plot(t, vexp, '--')
>> hold off
```

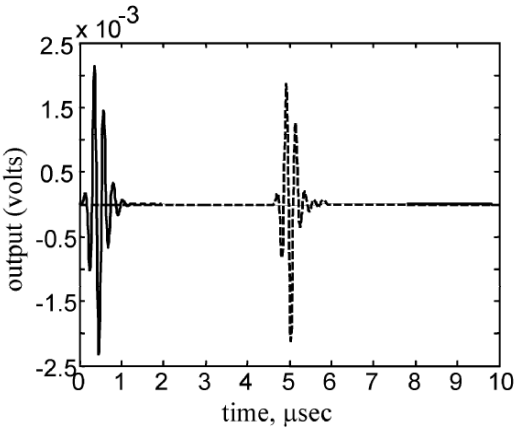

Fig. 12.14. The voltage received from an on-axis spherical void for the configuration shown in Fig. 12.10 (b) as predicted by the Thompson-Gray measurement model using an experimentally determined system function and the Kirchhoff approximation for the far-field scattering of the void (solid line) and the experimentally measured flaw signal (dashed line).

We can see in Fig. 12.14 that the two waveforms are close in amplitude and general shape. No attempt was made to match the time of arrivals of the two signals. In fact, in the calculation of these signals the phase terms that represent the time delays present due to propagation in the fluid and solid media were omitted. The measurement model predicts a slightly larger response than the measured response and there are some very small late time differences between the two signals. Fig. 12.14 shows that the Thompson-Gray measurement model coupled with the Kirchhoff approximation does a remarkably good job of predicting the flaw signal in this example even though the non-dimensional wave number, $k_{p2}b$, of the flaw for P-waves based on the transducer center frequency of 5 MHz is only $k_{p2}b = 1.8$. Formally the Kirchhoff approximation is a high frequency approximation where we must have $k_{p2}b >> 1$ but we see this approximation still works well at much lower frequencies (or smaller flaw sizes) where $k_{p2}b$ is not large. This is consistent with our discussion of that approximation in Chapter 10. However, as shown in Chapter 10, if $k_{p2}b < 1$ then the Kirchhoff approximation generally will not be accurate. Also note that even the completely modeled signal of Fig. 12.11, has

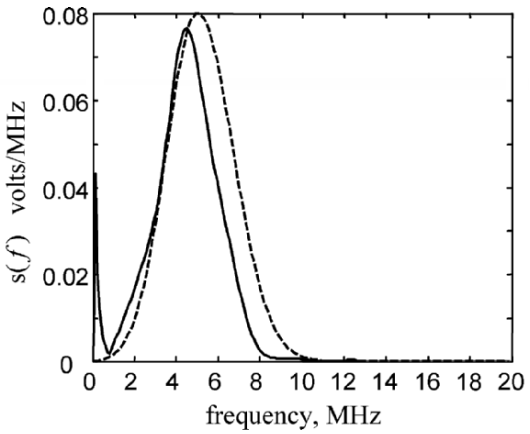

Fig. 12.15. The magnitude of the measured system function for the configuration of Fig 12.10 (a) (solid line) and the magnitude of the system function synthesized using the function systf (dashed line).

approximately the same amplitude as the experimental signal although the waveform details are different. Those differences in waveform shape come primarily from the fact that a purely real model-based system function was used in calculating the response in Fig. 12.11 while the complex-valued measured system function was used in Fig. 12.14. There are also some differences in the amplitudes and widths of the two different system functions used in Figs. 12.11 and 12.14. Figure 12.15 compares the magnitudes of these two system functions versus frequency. It can be seen that although the transducer being used is listed as a 5 MHz transducer, the system function determined experimentally peaks at a slightly lower value. For the modeled system function, we centered the Gaussian function at the 5 MHz value. Likely we could improve our predictions of the wave form obtained using a model-based system function by making the amplitude and bandwidth of that function agree more closely with the experimentally determined system function.

In Chapter 10 we gave the separations of variables solution for the pulse echo P-wave response of a spherical void. Those expressions have been encoded in the MATLAB function A_void_Psep (see Appendix G for a code listing). We can simply replace the Kirchhoff-based function A_void in the setup structure by this function:

```
>> setup.flaw.Afunc ='A_void_Psep';
```

and then rerun the measurement model and compare with the experimentally measured sphere response:

```
>> [Vout, setup] = TG_PE_MM(setup);
>> Ve = [Vout, zeros(1,800)];
>> Ve(1) = Ve(1)/2 ;
>> vt = 2*real(IFourierT(Ve, dt));
>> plot(t, vt)
>> hold on
>> load 'sphere_flaw'
>> plot(t, vexp, '--')
>> hold off
```

The results are shown in Fig. 12.16. From that figure we see that the amplitude of the modeled flaw signal is now very close to that of the experimental signal.

We can also examine the sphere with a spherically focused probe. The script TG_sphere_example3 given in Code Listing 12.14 again uses the TG_sphere_example1 script to set up most of the parameters. The transducer used is a 12.46 mm diameter, 172.9 mm focal length probe, so those parameters in setup are changed. These transducer parameters are both measured effective values, found by the methods discussed in Chapter 7. In this case the water path length for the flaw measurement is again 50.8 mm so that value need not be changed but the reference experiment to determine the system function must be carried out with the spherically focused transducer at a water path equal to the focal length to use the transfer function found in Chapter 8. Thus, the setup.system.z1r must also be changed. The function exp_systf again can calculate the system function for this focused probe. In this case the reference waveform is contained in the MAT-file 'sphere_ref_foc'. For this example we will also use the Kirchhoff approximation to determine the scattering amplitude of the void, so that we set setup.flaw.Afunc = ‘A_void’. With these updates made to setup, the measurement model can be run and the waveform synthesized as before. The experimentally measured response of the void to this focused probe is contained in the .mat file ‘sphere_flaw_foc’ in the variable vexp so if we load this file and then plot it alongside our modeled response we obtain the results shown in Fig. 12.17. It can be seen from that figure that the Kirchhoff approximation does a very good job of reproducing the measured flaw signal.

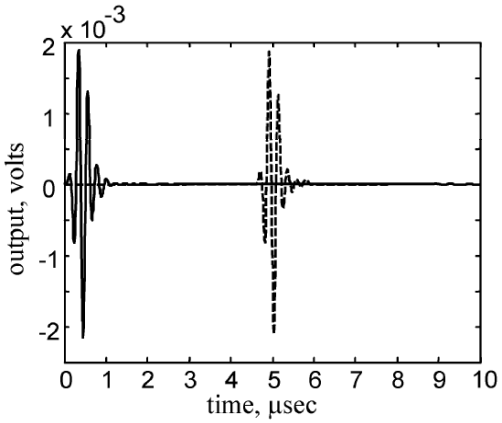

Fig. 12.16. The voltage received from an on-axis spherical void for the configu ration shown in Fig. 12.10 (b) as predicted by the Thompson-Gray measurement model using an experimentally determined system function and the method of separation of variables for the far-field scattering of the void (solid line). The experimentally measured flaw signal is shown for comparison (dashed line).

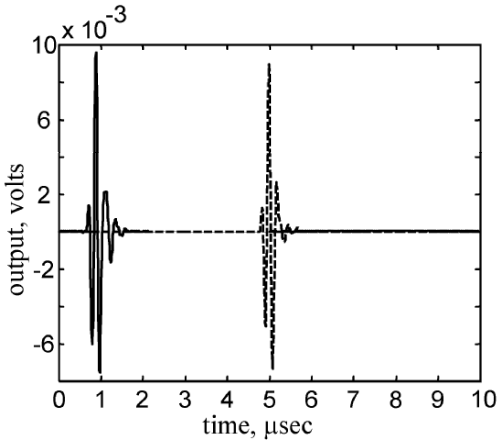

Fig. 12.17. The voltage received from an on-axis spherical void for the configuration shown in Fig. 12.10 (b) using a spherically focused probe. The wave form was predicted by the Thompson-Gray measurement model using an experimentally determined system function and the Kirchhoff approximation for the far-field scattering of the void (solid line). The experimentally measured flaw signal is shown for comparison (dashed line).

Code Listing 12.14. A script for calculating the response of a spherical void in the configuration shown in Fig. 12.10 (b) where a spherically focused probe is used. The predicted response uses an experimentally determined system function and a flaw response given by the Kirchhoff approximation which is then plotted and compared to an experimentally measured signal.

```
% script TG_sphere_example3
% calculates the waveform for a spherical void
% using an experimentally determined system function; focused probe case
clear
% run TG_sphere_example1 script to get most system parameters
TG_sphere_example1
%update setup
setup.trans.d = 12.46;
setup.trans.fl =172.9;
setup.system.z1r =172.9;
setup.flaw.Afunc = 'A_void';
%specify use of experimentally determined system function
%and reference waveform
setup.system.sysf='exp_systf';
setup.system.ref_file ='sphere_ref_foc';
%run measurement model
[Vout, setup] = TG_PE_MM(setup);
% plot(f, abs(Vout))  intermediate plot omitted
% pad frequency domain amplitude with zeros
Ve= [ Vout, zeros(1,800)];
Ve(1) = Ve(1)/2 ;   %Now, compute wave form and plot
vt =2*real(IFourierT(Ve, dt));
plot(t, c_shift(vt, 600))
load 'sphere_flaw_foc'
hold on
plot(t, vexp,'--')
hold off
```

12.9 A Large Flaw Measurement Model

We could also use the Thompson-Gray measurement model to predict the response of other scatterers in the configuration of Fig. 12.10 (b) such as

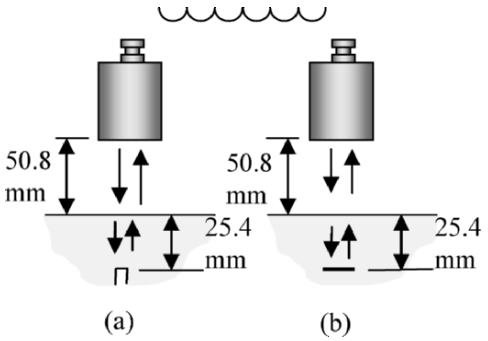

Fig. 12.18. A scattering configuration where **(a)** a flat-bottom hole or **(b)** a flat circular crack is interrogated by a planar transducer at normal incidence through a fluid-solid interface. In both cases the center of the scatterer is located on the central axis of the transducer.

the flat-bottom hole shown in Fig. 12.18 (a) or the flat circular crack shown in Fig. 12.18 (b). However, both of these scatterers are very "specular", i.e. they reflect much of the incident waves directly back to the transducer from their flat surfaces. As a consequence, the assumption of the Thompson-Gray measurement model that the wave field of the transducer beam is nearly constant over the flaw surface leads to significant errors if the sizes of the flat-bottom hole or crack being considered are not very small. In contrast, it has been found that the spherical void is much more tolerant to the small flaw assumption and the Thompson-Gray measurement model works well even for large spherical flaws. To account for beam variations we will use the more general measurement model of Eq. (12.1) coupled with a Kirchhoff approximation model for the scattering of a crack. In the Kirchhoff approximation this same flaw scattering model is appropriate also for the flat-bottom hole since the sides of the hole do not contribute anything in that approximation when the incident waves are at normal incidence to the circular, flat end of the hole. Since we are considering a pulse-echo setup for P-waves we have $\hat{V}^{(1)} = \hat{V}^{(2)} = \hat{V}\left(\mathbf{x}, \omega\right)$ in Eq. (12.1) and from the Kirchhoff approximation and the fact that we have a stress-free surface, we find (see Eq. (10.12))

$$\begin{aligned} \mathcal{A}\left(\mathbf{x}, \omega\right) &= -\frac{ik_{p2}}{2\pi}\left(\mathbf{d}_i^p \cdot \mathbf{n}\right)\exp\left[ik_{p2}\left(\mathbf{d}_i^p \cdot \mathbf{x}\right)\right] \\ &= \frac{ik_{p2}}{2\pi}, \end{aligned} \tag{12.30}$$

where we have used the fact that on the flat surface S $\mathbf{d}_i^p \cdot \mathbf{n} = -1$ and $\mathbf{d}_i^p \cdot \mathbf{x} = 0$ .Then Eq. (12.1) becomes

$$V_R(\omega) = s(\omega)\left[\frac{4\pi\rho_2 c_{p2}}{-ik_{p2}Z_r^{T;a}}\right]\frac{ik_{p2}}{2\pi}\int_{S_f}\left[\hat{V}(\mathbf{x},\omega)\right]^2 dS. \tag{12.31}$$

Note that because of the symmetry of the incident field in the configuration of Fig. 12.18 we have $\hat{V}(\mathbf{x},\omega) = \hat{V}(r,\omega)$, where r is the radial distance from the center of the scatterer and the transducer axis. Thus, in this case we have

$$V_R(\omega) = s(\omega)\left[\frac{4\pi\rho_2 c_{p2}}{-ik_{p2}Z_r^{T;a}}\right]ik_{p2}\int_{r=0}^{r=b}\left[\hat{V}(r,\omega)\right]^2 r dr. \tag{12.32}$$

If we break the total integration into a series segments from $r = r_m$ to $r = r_{m+1}$, with $r_m = (m-1)b/(M-1)$ $(m = 1,2,...M-1)$ then we can approximate the velocity field as constants over the centroids of those segments given by $\hat{V}(\overline{r}_m,\omega)$, where $\overline{r}_m = (r_{m+1} + r_m)/2$ is an average radius. Each of these segments represent a circular ring except the first one which is a complete circular area of radius $b/(M-1)$ since $r_1 = 0$. For that circular segment we let $\overline{r}_1 = 0$ so that fields over that segment are calculated on the transducer axis, which is consistent with what we would do normally for a very small on-axis crack or flat-bottom hole. Equation (12.32) becomes

$$V_R(\omega) = \sum_{m=1}^{M-1} s(\omega)\left[\frac{4\pi\rho_2 c_{p2}}{-ik_{p2}Z_r^{T;a}}\right]\left[\hat{V}(\overline{r}_m,\omega)\right]^2\frac{ik_{p2}\left(r_{m+1}^2 - r_m^2\right)}{2}. \tag{12.33}$$

In the Kirchhoff approximation the normal incidence pulse-echo P-wave far-field scattering amplitude of a flat crack of radius r_m is just (see Eq. (10.38)):

$$A_m\left(\mathbf{e}_i^p; -\mathbf{e}_i^p\right) = \frac{ik_{p2}r_m^2}{2} \tag{12.34}$$

so that we can write Eq. (12.33) as:

$$V_R(\omega) = \sum_{m=1}^{M-1} s(\omega) \left[\frac{4\pi\rho_2 c_{p2}}{-ik_{p2} Z_r^{T;a}} \right] \left[\hat{V}(\overline{r}_m, \omega) \right]^2 \cdot \left[A_{m+1}(\mathbf{e}_i^p; -\mathbf{e}_i^p) - A_m(\mathbf{e}_i^p; -\mathbf{e}_i^p) \right]. \tag{12.35}$$

Comparing Eq. (12.35) and Eq. (12.6), we see that we can obtain the voltage by merely combining appropriately a number of Thompson-Gray measurement model terms for the scattering of a circular crack. Thus, we can use the TG_PE_MM function in conjunction with A_crack to model this case.

The MATLAB script FBH_example1 (Code Listing 12.15) implements Eq. (12.35) for a #8 flat-bottom hole in a steel block. The reference wave form for calculating the system function resides in the file FBH_ref.mat and the experimental flaw response is in the file FBH_flaw_n8.mat. The script calculates the FBH response and then plots both it and the experimental signal. The results are shown in Fig. 12.19.

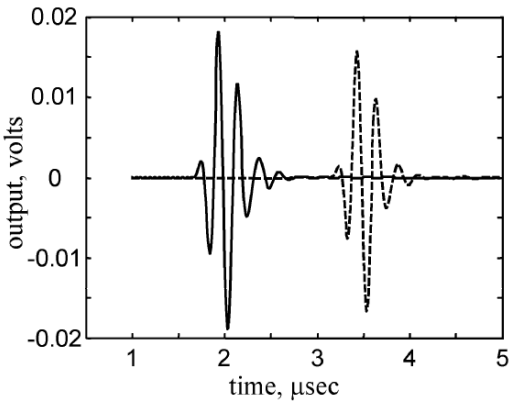

Fig. 12.19. The voltage received from an on-axis #8 flat-bottom hole for the configuration shown in Fig. 12.18 (a) as predicted by a measurement model that accounts for field variations over the end of the flat-bottom hole and uses an experimentally determined system function and the Kirchhoff approximation for the far-field scattering of the hole (solid line). The experimentally measured flat-bottom hole signal is shown for comparison (dashed line).

Code Listing 12.15. A script for calculating the response of a #8 flat-bottom hole, taking into account the variations of the incident transducer beam over the bottom of the hole.

```
% FBH_example1 script
% This script calculates the pulse-echo P-wave response of an on-axis
% #8 flat-bottom hole interrogated by a 5 MHz planar probe through a
% fluid-solid interface at normal incidence

clear
setup = setup_maker;
% setup parameters that need to be specified
% for this example
f =s_space(0, 20, 200);
cp1 = 1484.;
d2 = 7.86;
cp2 = 5940.;
cs2 = 3230.;
z1 = 50.8;
z2 = 25.4;
amp =0.12;
bw = 3.;
z1r =50.8;
p1 = [ 0 0 0.02479E-03  0 0];
b =1.5875;   % number eight FBH
flaw_name = 'A_crack';
sysfunc ='exp_systf';
reffile='FBH_ref';
setup.f =f;
setup.system.amp = amp;
setup.system.bw = bw;
setup.system.z1r =z1r;
setup.system.sysf = sysfunc;
setup.system.ref_file = reffile;
setup.geom.z1 = z1;
setup.geom.z2 = z2;
setup.matl.cp1 = cp1;
setup.matl.d2 = d2;
setup.matl.cp2 = cp2;
setup.matl.cs2 = cs2;
setup.matl.p1 = p1;
setup.flaw.b = b;
setup.flaw.Afunc = flaw_name;

% break up hole end into rings
```

```
nR= 10;  % use 9 rings (10 points)
rm = linspace(0, b, nR);    %ring edges
rmu = rm(2:nR); %upper edges
rml =rm(1:nR-1); %lower edges
rc =(rmu-rml)/2 + rml; %ring centroids
rc(1) = 0; %make first centroid at origin

Vf = zeros(size(f));

% calculate received voltage

for nd = 1:nR-1
   setup.geom.x2 = rc(nd);
   setup.flaw.b =rm(nd);
   [Vf1, setup] = TG_PE_MM(setup);
   setup.flaw.b = rm(nd+1);
   [Vf2, setup] = TG_PE_MM(setup);
   Vf = (Vf2-Vf1) +Vf;
end

% extend frequency components to permit
% taking FFT

df = f(2)-f(1);
dt = 1/(1000*df);
t = s_space(0, 1000*dt, 1000);
Vfe = [Vf zeros(1,800)];
Vfe(1) = Vfe(1)/2;
vt =2*real(IFourierT(Vfe, dt));
vs =c_shift(vt, 700);
plot(t(100:500), vs(100:500))
%plot(t_shift(t,700), c_shift(vt,700))
hold on
load 'FBH_flaw';
plot(t(100:500), vexp(250:650), '--')
hold off
```

12.10 A Measurement Model for Cylindrical Reflectors

The third measurement model discussed previously was for treating the pulse-echo response of cylindrical reflectors such as a side-drilled hole (SDH) where the beam variations can be neglected over the cross-section

of the scatterer. In terms of the geometry parameters defined in Fig. 12.1, this measurement model (see Eq. (12.5)) is:

$$V_R(\omega) = s(\omega)\left[\int_L \left(\hat{V}_0^{(1)}(y_2,\omega)\right)^2 dy_2\right]\left[\frac{A(\omega)}{L}\right]\left[\frac{4\pi\rho_2 c_{\alpha 2}}{-ik_{\alpha 2}Z_r^{T;a}}\right]. \qquad (12.36)$$

This measurement model is similar to the Thompson-Gray measurement model (Eq. (12.6)) but now we must replace the square of the incident velocity field in that model (for pulse-echo) by the integrated velocity squared term in Eq. (12.36) and the 3-D scattering amplitude in the

Code Listing 12.16. A MATLAB function for calculating the normalized far-field scattering amplitude of a side-drilled hole in pulse-echo using the Kirchhoff approximation.

```
function A =A_SDH(setup)
% A_SDH calculates the pulse-echo 3-D normalized far-field scattering
% amplitude,A/L, of a side-drilled hole in the Kirchhoff approximation
% using the frequency f in setup.f, the radius b in  setup.flaw.b,
% and the wave speed for the wave type in setup.wave.c2.
% The calling sequence is A = A_SDH(setup). The scattering
% amplitude, A, (in mm) is returned. In the calculation of the
% Struve function, an integration routine is used. Thus, the
% frequency, f, must be at most a vector to use this function
% effectively. It is not vectorized for f being a matrix.

f =setup.f;
b =setup.flaw.b;
c=setup.wave.c2;
kb =2000*pi*b.*f./c;
A =(kb./2).*(besselj(1, 2*kb)-i*struve(2*kb)) +i*kb./pi;

function y = struve(z)
num = length(z);
y=zeros(1,num);
for k = 1:num
y(k) = quadl(@struve_arg, 0, 1, [ ],[ ], z(k));
end

function y = struve_arg(x, z)
y = (4./pi).*z.*x.^2.*sin(z.*(1-x.^2)).*sqrt(2-x.^2);
```

Thompson-Gray model is now replaced by the normalized 3-D scattering amplitude, A/L, of the cylindrical scatterer, where L is the scatterer length. In the Kirchhoff approximation this normalized scattering amplitude was previously given by Eq. (10.53) for a SDH and has been coded in the MATLAB function A_SDH (Code Listing 12.16).

The multi-Gaussian beam model defined by the MATLAB function MGbeam has been modified so that it returns the integral of the square of the velocity field at the center of the SDH as well as an updated setup structure. The new MATLAB function is called I_MGbeam (Code-Listing 12.17). It is assumed that the y_2-coordinate of the flaw is now given in setup.geom.y2 as a vector of values and the integral in Eq. (12.36) is calculated approximately in I_MGbeam as a simple sum:

$$\int_L \left(\hat{V}\left(y_2,\omega\right)\right)^2 dy_2 = \sum_{i=1}^{N} \left(\hat{V}\left(y_i,\omega\right)\right)^2 \Delta y, \tag{12.37}$$

where $\hat{V}$ is the ideal velocity field (no attenuation) calculated by the multi-Gaussian beam model. In most cases the length of the hole extends the full width of a test block so that the hole length may be larger than the width of the incident beam. In that case, we can treat the SDH as infinitely long and simply sum over y_2-values where the fields are significant.

Code Listing 12.17. A MATLAB function for returning the integrated square of the velocity field for use in a measurement model for cylindrical reflectors where beam variations along the length of the reflector must be considered.

```
function [vi,setup ]=I_MGbeam(setup)

% get setup parameters
fin = setup.f;                  %frequency or frequencies (MHz)
type1 = setup.type1;            % wave type in medium one
type2 = setup.type2;            % wave type in medium two

a = setup.trans.d/2;            % transducer radius (mm)
Fl = setup.trans.fl;            % transducer focal length (mm)

z1 = setup.geom.z1;             % water path length (mm)
z2 = setup.geom.z2;             % path length in solid (mm)
x2 =setup.geom.x2;              % distance (mm) from ray axis in POI
yin = setup.geom.y2;            % distance (mm) perpendicular to the POI
Rx = setup.geom.R1;             % interface radius of curvature (mm) in POI
Ry =setup.geom.R2;            % interface radius of curvature (mm) out of POI
```

```
iang = setup.geom.i_ang;            % incident angle (deg)

d1 = setup.matl.d1;                 % density (fluid)
d2 =setup.matl.d2;                  % density (solid)
cp1 = setup.matl.cp1;               % compressional wave speed -fluid  (m/sec)
cp2 = setup.matl.cp2;               % compressional wave speed -solid (m/sec)
cs2 = setup.matl.cs2;               % shear wave speed -solid (m/sec)

% form frequency, y2-values needed for integration into arrays
[f,y2]=meshgrid(fin, yin);
% update setup with these values temporarily (need for init_z)
% setup values will be returned to fin, yin values later
setup.f =f;
setup.geom.y2 = y2;

% define y -increment
dy = yin(2) - yin(1);


[A, B] = gauss_c15;          % Wen and Breazeale coefficients (15)

% update setup.wave wave speeds
if strcmp(type1, 'p')
   setup.wave.c1 =cp1;
elseif strcmp(type1, 's')
   setup.wave.c1 = cs1;
else
   error('wrong wave type (must be p or s) ')
end

if strcmp(type2, 'p')
   setup.wave.c2 =cp2;
elseif strcmp(type2, 's')
   setup.wave.c2 = cs2;
else
   error('wrong wave type (must be p or s)')
end
% calculate transmission coefficient, update setup
setup.wave.T12 = fluid_solid(setup);

% wave speeds and transmission coefficient for the beam model
c1 =setup.wave.c1;
c2 =setup.wave.c2;          % wave speed for wave type2
T = setup.wave.T12;         % transmission coefficient

% parameters appearing in beam model
```

```
cosi = cos(pi*iang/180);              % cosine of incident angle
sinr = (c2/c1)*sin(pi*iang/180);      % sine of refracted angle from Snell's law
if sinr >= 1
  error('Beyond the Critical angle') % no transmitted wave of given wave type
else
  cosr = sqrt( 1 - sinr^2);
end

  h11 = 1/Rx;  %curvature
  h22 = 1/Ry;  %curvature
zr = eps*(f == 0) + 1000*pi*(a^2)*f./c1;              % “Rayleigh” distance
k1 = 2*pi*1000*f./c1;                                 % wave number in fluid

%initialize predicted velocity with zeros of a size
% compatible with largest array in f, z1, z2, x2, y2 setup parameters
v = init_z(setup);
% return to original frequency, fin, and distance, yin, values in setup
setup.f = fin;
setup.geom.y2 =yin;

%multi-Gaussian beam model

for j = 1:15                          % form up multi-Gaussian beam model

 b =B(j) + i*zr./Fl;                  % modify coefficients for focused probe
                                      % Fl = inf for planar probe

q = z1 - i*zr./b;
K = q.*(cosi -(c1/c2)*cosr);
M1 = (cosi^2 +K.*h11)./cosr^2;
M2 =1 + K.*h22;
ZR1 = q./M1;
ZR2 =q./M2;
m11 = 1./(ZR1 +(c2/c1).*z2);
m22 = 1./(ZR2 +(c2/c1).*z2);
  t1 = A(j)./(1 + (i.*b./zr).*z1);
  t2 = t1.*T.*sqrt(ZR1).*sqrt(ZR2).*sqrt(m11).*sqrt(m22);
  v = v + t2.*exp(i.*(k1./2).*(m11.*(x2.^2) + m22.*(y2.^2)));

end
% sum over y-values squared times dy to integrate
vs =v.^2;
vi=sum(vs.*dy, 1);
```

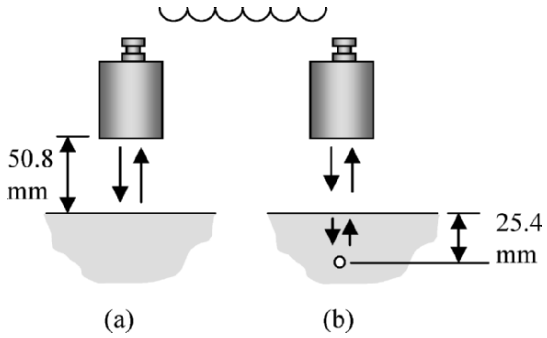

Fig. 12.20. (a) The reference scattering configuration for determining the system function and **(b)** the setup for measuring the pulse-echo response of a side-drilled hole.

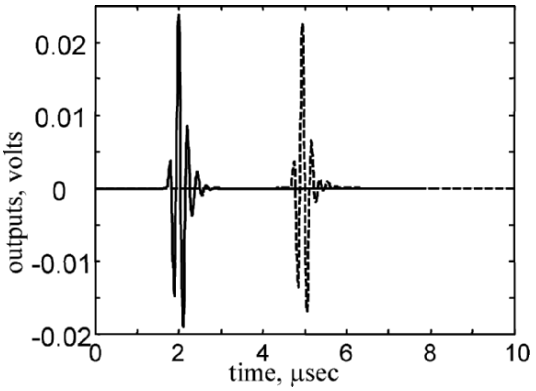

Fig. 12.21. The output voltage simulated for the pulse-echo P-wave response of a 1 mm side-drilled hole in the configuration shown in Fig. 12.20 (b) (solid line) and the corresponding experimentally measured response (dashed line).

The MATLAB function SDH_PE_MM (Code Lisitng 12.18) uses I_MGbeam and A_SDH to generate the system output voltage. The MATLAB script SDH_example1(Code Listing 12.19) uses SDH_PE_MM to simulate the response of a one mm diameter SDH in an aluminum sample in a configuration shown in Fig. 12.20 (b). Again, the system function is determined experimentally from a measured front-surface reflection as shown in Fig. 12.20 (a). The integration over the length of the

hole here is taken from −50 mm to +50 mm based on an evaluation of the incident fields on the SDH for this problem (that evaluation is not shown here explicitly but can be easily done with the MGbeam function). For other SDH problems the limits of integration will have to be determined in this same way on a case by case basis. The predicted voltage using the MATLAB script SDH_example1 is shown in Fig. 12.21 along with the corresponding experimentally observed signal. Again, the Kirchhoff approximation does a very good job of representing the measured signal.

Code Listing 12.18. A MATLAB function that computes the output voltage for a cylindrical reflector using the measurement model of Eq. (12.21).

```
function [Vf, setup] =SDH_PE_MM(setup)
% SDH_PE_MM generates the frequency components of the
% output voltage, Vf, of an ultrasonic pulse-echo immersion
% measurement system generated by a side-drilled hole.
% The function returns Vf  as well as an updated setup structure
% The calling sequence is [Vf, setup] =SDH_PE_MM(setup);

% First, compute the integrated beam velocity squared term
% and update the setup structure. This does not include
% attenuation
[vs, setup] = I_MGbeam(setup);

%get the setup parameters  needed for the constant term
%in the measurement model
f = setup.f;
r= setup.trans.d/2;    % transducer radius
d1 =setup.matl.d1;
d2 =setup.matl.d2;
c1 = setup.wave.c1;
c2 = setup.wave.c2;

%compute wave number in medium two and
%the constant term in the measurement model

k2 = (2000.*pi.*f)./c2;
k2 =k2 + eps*( k2 == 0);  % prevent division by zero
K= (4.*d2.*c2)./(-i.*k2.*r^2.*d1.*c1);

% check to see if a model-based or experimentally determined system
% function is to be used
if strcmp(setup.system.sysf, 'systf')
   sys = systf(setup);
```

```
else
   sys =feval(setup.system.sysf, setup);
end

% find flaw type to be used
if strcmp( setup.flaw.Afunc, 'empty')
   error('flaw function not specified in setup')
else
   A = feval(setup.flaw.Afunc, setup);
end

%compute output voltage, Vf, (volts/MHz)
Vf = sys.*(vs).*(attenuate(setup)).^2.*A.*K;
```

Code Listing 12.19. A MATLAB script for calculating the pulse-echo P-wave response of a 1 mm diameter side-drilled hole in the configuration of Fig. 12.20 (b) using the Kirchhoff approximation to calculate the scattering of the side-drilled hole and an experimentally determined system function found from the reference configuration of Fig. 12.20 (a). The predicted response is compared to the experimentally observed signal.

```
%SDH_example1 script
% This script calculates the pulse-echo P-wave response of an on-axis
% 1 mm diam side-drilled hole interrogated by a 5 MHz planar probe through a
% fluid-solid interface at normal incidence
clear
setup = setup_maker;
% setup parameters that need to be specified
% for this example
f =s_space(0, 20, 200);
y2 =linspace(-50, 50, 500);
cp1 = 1484.;
d2 = 2.75;
cp2 = 6416.;
cs2 = 3163.;
z1 = 50.8;
z2 = 25.4;
amp =0.12;
bw = 3.;
z1r =50.8;
p1 = [ 0 0 0.02479E-03  0 0];
```

```
b =0.5;   % 0.5 mm radius
flaw_name = 'A_SDH';
sysfunc ='exp_systf';
reffile='SDH_ref';

% put parameters in setup

setup.f =f;
setup.system.amp = amp;
setup.system.bw = bw;
setup.system.z1r =z1r;
setup.system.sysf = sysfunc;
setup.system.ref_file = reffile;
setup.geom.z1 = z1;
setup.geom.z2 = z2;
setup.geom.y2 = y2;
setup.matl.cp1 = cp1;
setup.matl.d2 = d2;
setup.matl.cp2 = cp2;
setup.matl.cs2 = cs2;
setup.matl.p1 = p1;
setup.flaw.b = b;
setup.flaw.Afunc = flaw_name;

[Vf, setup] = SDH_PE_MM(setup);

% extend frequency components to permit
% taking FFT
df = f(2)-f(1);
dt = 1/(1000*df);
t = s_space(0, 1000*dt, 1000);
Vfe = [Vf zeros(1,800)];
Vfe(1) = Vfe(1)/2;
vt =2*real(IFourierT(Vfe, dt));
vs =c_shift(vt, 700);
plot(t, vs)
hold on
load 'SDH_flaw_1';
plot(t, vexp, '--')
hold off
```

12.11 References

12.1 Lopez-Sanchez A, Kim HJ, Schmerr LW, Gray TA (2006) Modeling the response of ultrasonic reference reflectors. Research in NDE 17: 49-70

12.2 Song SJ, Schmerr LW, Thompson RB (2006) Ultrasonic benchmarking study: overview up to year 2005. In: Thompson DO, Chimenti DE (eds) Review of progress in quantitative nondestructive evaluation. American Institute of Physics, Melville, NY, pp 1844-1853