NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

COMPUTER PROGRAMS SUPPORTING INSTRUCTION IN ACOUSTICS

by

Kevin Andrew Melody

March, 1998

Thesis Advisor: James V. Sanders
Second Reader: Kevin B. Smith

Approved for public release, distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1998	Master's Thesis

4. TITLE AND SUBTITLE

COMPUTER PROGRAMS SUPPORTING INSTRUCTION IN ACOUSTICS

5. FUNDING NUMBERS

6. AUTHOR(S)

Kevin A. Melody

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release, distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Acoustics is a field of study not easily understood and laboratory experiments which might shed light on problems in acoustics are complex and expensive to accomplish. Computers have become a valuable tool in many fields of study in order to examine problems which would be difficult and expensive, if not impossible to study using traditional methods. This thesis is an extension of work previously completed by Thomas Green to support instruction utilizing the text, Fundamentals of Acoustics, Third Edition, John Wiley & Sons, Inc., by Coppens, Frey, Kinsler, and Sanders. The fourth edition of Fundamentals of Acoustics is currently in revision and the computer programs explained in this thesis will be used to support it. All programs utilize MATLAB™, a widely accepted programming language for accomplishing numerical analysis of engineering problems. The benefit of these programs will be very dependant on students using them in conjunction with the text.

14. SUBJECT TERMS	15. NUMBER OF PAGES
acoustics, MATLAB	120
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Approved for public release: distribution is unlimited.

COMPUTER PROGRAMS SUPPORTING INSTRUCTION IN ACOUSTICS

Kevin A. Melody

Lieutenant, United States Navy

B.S., U.S. Naval Academy, 1992

Submitted in partial fulfillment

of the requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING ACOUSTICS

from the

NAVAL POSTGRADUATE SCHOOL

March 1998

Author: Kevin A. Melody

Kevin A. Melody

Approved by: James V. Sanders

James V. Sanders, Thesis Advisor

Kevin B. Smith

Kevin B. Smith, Second Reader

Kevin B. Smith

Kevin B. Smith, Chairman

Engineering Acoustics Academic Committee

ABSTRACT

Acoustics is a field of study not easily understood and laboratory experiments which might shed light on problems in acoustics are complex and expensive to accomplish. Computers have become a valuable tool in many fields of study in order to examine complex problems which would be difficult and expensive, if not impossible to study using traditional methods. This thesis is an extension of work previously completed by Thomas Green to support instruction utilizing the text, Fundamentals of Acoustics, Third Edition, John Wiley & Sons, Inc., by Coppens, Frey, Kinsler, and Sanders. The fourth edition of Fundamentals of Acoustics is currently in revision and the computer programs explained in this thesis will be used to support it. All programs utilize MATLAB™, a widely accepted programing language for accomplishing numerical analysis of engineering problems. The benefit of these programs will be very dependant on students using them in conjunction with the text.

TABLE OF CONTENTS

I. INTRODUCTION

A. BACKGROUND

Modeling has become a widely accepted way to analyze complex engineering problems. One of the most powerful tools for this is MATLAB™, a computer programing language specifically geared towards numerical modeling of engineering problems. It has a wide variety of uses and can help explain engineering principles which would be difficult or too expensive to actually conduct in a laboratory.

B. PURPOSE

This thesis is an extension of work done by LCDR Thomas A. Green of providing computer algorithms for topics in the text, Fundamentals of Acoustics, Third Edition, John Wiley & Sons, Inc., by Kinsler, Frey, Coppens, and Sanders. The programs are to provide an interactive tool for the student to use in broadening their understanding of material in the text.

C. REQUIREMENTS

Although these programs can be used without reference to text material, they will be most useful when used in conjunction with the sections of KFCS which apply. The student will be able to more quickly grasp the relationships between the input and output of the program when the text is used in concert with the computer as the programs are designed to further explain the text material.

The programs in this thesis will run on any machine which supports MATLAB™, Professional Version 4.2 or later. Although the student version of MATLAB™ can be used in most cases, some of the algorithms use large vectors and matrices which cannot

be handled by the student version of MATLAB™.

To obtain more information about MATLAB™, contact The MathWorks, Inc., University Sales Department, 24 Prime Part Way, Natick, Massachusetts 01760-1500, or call them at (508) 653-1415, or go to their World Wide Web site at http://www.mathworks.com.

A copy of the programs in this thesis can be obtained by sending a formatted disk to :

Professor James V. Sanders
Physics Department (Code PH)
Naval Postgraduate School
833 Dyer Road, Room 203
Monterey, CA 93943-5117

II. DAMAGED ARRAYS

The program BKNARRAY.M displays plots of beam pattern for arrays of any size. The most useful feature of this program is it's ability to illustrate how the performance of an array declines as elements within it fail. This program supports instruction in concepts discussed in Chapter 8 of KFCS.

A. CONCEPTS

The first assumption is that the distance to the receiver is much greater than the length of the array. It is generally accepted that a difference of one order of magnitude satisfies this requirement . Additionally, the program assumes that the elements radiate spherical waves with the same amplitude and phase. Although this program was developed as an array of sources, by the principle of reciprocity, the beam patterns can also be viewed as patterns of reception.

Section 8.13 of KFCS discusses the simple line array. Because of our assumption that the elements emit waves of the same amplitude and phase, the form of the pressure field of each source is

$$\frac{A}{r_i} e^{j(\omega t - kr_i)} \tag{2.1}$$

Where r_i is the distance from the ith element to the position in the far field. Because the

beam pattern is normalized to the pressure on the axis of the main lobe, the amplitude term can be ignored and the term which contributes to the beam pattern of an unsteered array is

$$e^{-jkr_i} \tag{2.2}$$

which is basically the phase contribution of each element. The total pressure in the far field is then simply the sum of the pressure contribution of all elements in the array or

$$\sum_{i=1}^{N} e^{-jkr_i} \tag{2.3}$$

Arrays are normally designed to have a single major lobe as narrow as possible. The criterion to achieve this is

$$d=\frac{\lambda(N-1)}{N} \tag{2.4}$$

where d is the array spacing. This program uses the optimal array spacing to obtain a single major lobe so frequency is not a user input.

Finally, the plots are referenced to the acoustic pressure emitted by a single element and the same dB scale is maintained for each set of plots.

B. PROGRAM DESCRIPTION

The program has two user-controllable parameters. The first is the number of elements in the array; this must be a positive number. If the user is only interested in the

beam pattern for a specific number of elements then this is the only input. To see how the array behaves as elements fail they can input the total number of elements they would like to see fail. The program plots both the beam pattern for a fully functioning array and the beam pattern as each additional element fails. The failed elements are chosen at random by the computer. Functioning elements in the array are denoted by a green "o" and failed elements are denoted by a red "x". These are displayed above the array's beam pattern. Figure 2.1 is a sample of the plots generated by BKNARRAY.M. On the lefthand

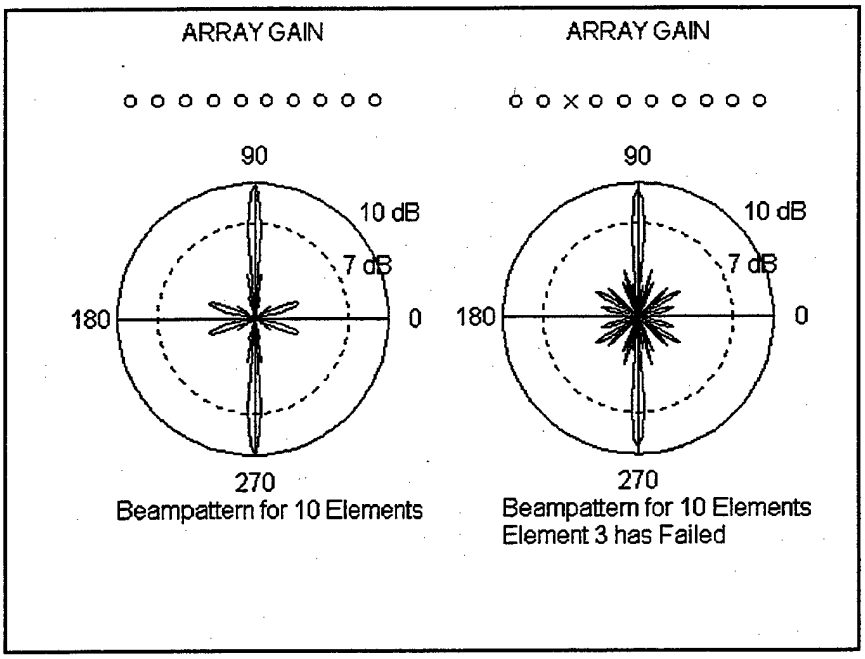

Figure 2.1 Beam Pattern for a 10 Element Array

side of the figure is a plot of the beam pattern of a ten element array and on the right is the plot of the same array with the third element not functioning. BKNARRAY.M uses a modified version of the MATLAB™ program POLAR.M which is included in the appendix.

C. ALGORITHM bknarray.m

```
function bknarray(elements,failures)
% BKNARRAY is a function used to display the  beampattern of an n-element array. The
% number of elements is chosen by the user. In addition the user can disable any number
% of elements in the array to see how this affects the beampattern. If it is desired to see
% just the fully functioning array pattern then the user must input "0" as the number of
% failed elements. The user is not able to select which elements they want to disable,
% only the total number. The function randomly selects which elements to disable and
% displays the array format below the array pattern. The dB reference is the gain above
% what a single element would see.
global radius          % radius is declared as a global variable in order to allow the
                       % function POLAR3 to plot all beampatterns of an array on the
                       %same scale.
global r3db
%-----------------------
% This is the user protection portion of the program.
```

```
if nargin == 1
        failures=0;
        if (elements < 1)
                error('Number of elements must be more than 0.')
        end
elseif nargin == 2
        if (elements < 1)
                error('Number of elements must be more than zero.')
        end
        if (failures < 0)
                error('Number of failures must be zero or greater.')
        end
        if (failures > elements)
                error('BKNARRAY requires failures < or = elements.')
        end
else
        error('BKNARRAY requires at least one input argument.')
end
%-------------------------
frequency=250;                          % in hertz
```

```
soundspd=1500;                                  % in meters/sec
k=(2*pi*frequency)/soundspd;                    % wave number
lambda=soundspd/frequency;                      % wavelength
arrayspace=(lambda*(elements-1))/(elements);    % spacing of array elements
range=(arrayspace*(elements-1)*500);            % distance in meters
theta=0:(2*pi/180):(2*pi);                      % determines number of analysis points
failed_element=[];
element=ones(1,elements);
plot_position=1;
radius=10*log10(elements);
if elements~=1
        r3db=radius-3;
else r3db=0;
end
element_position=([0:(elements-1)].*arrayspace)-...
            ((elements/2)*arrayspace)+(arrayspace/2);
element_position=element_position';
[THETA,POS]=meshgrid(theta,element_position);
% element_range computes the range to each analysis position from
% each array element
```

```
element_range=(((range*sin(THETA)).^2)+...
                (((range*cos(THETA))+POS).^2)).^(.5);
pressure=(exp(-j.*element_range.*k));          % computes pressure contribution
                                               %  of each element
if failures~=0
    ele_mat=1:elements;
    for ind=1:failures;
        choice=ceil(rand*length(ele_mat));
        failed_element=[failed_element ele_mat(choice)];
        ele_mat(choice)=[];
    end
end
for count=0:failures,
    if count~=0
        pressure(failed_element(count),:)=zeros(size(pressure(1,:)));
        element(failed_element(count))=0;
    end
    if (elements==1)
        beampattern=10*log10(abs(pressure));
    else
```

```
        beampattern=10*log10(abs(sum(pressure)));
        one_gain=((beampattern>0)|(beampattern==0));
        beampattern=beampattern.*one_gain;
        % sums pressure contribution of each array element
        % to obtain beampattern
end
figure(ceil((count+1)/2))
if plot_position==3
        plot_position=1;
end
subplot(1,2,plot_position)
polar3(theta,beampattern)
title('ARRAY GAIN')
y=1.6*radius;
hold on
for ind=1:elements
        x=2*(ind-(elements/2)-.5);
        if element(ind)==1
                plot(x,y,'go')
        elseif element(ind)==0
```

```
                plot(x,y,'rx')
        end
end
hold off
y=1.1*radius*sin(45*pi/180);
x=y;
text(x,y,[num2str(round(radius)),' dB'])
if elements>2
        y=1.1*r3db*sin(30*pi/180);
        x=1.1*r3db*cos(30*pi/180);
        text(x,y,[num2str(round(r3db)),' dB'])
end
text((-radius),(-1.5*radius),...
        ['Beampattern for ',num2str(elements),' Elements'],...
        'verticalalignment','bottom')
if count~=0
text((-radius),(-1.7*radius),...
        ['Element ',num2str(failed_element(count)),...
        ' has Failed'],'verticalalignment','bottom')
end
```

```
        plot_position=plot_position+1;
end
```

III. RESONANCE IN PIPES

The program FPIPE.M computes the acoustic resistance and reactance of a flanged, open ended pipe. Plots are generated which can be used to determine resonant and antiresonant frequencies for the pipe. This program supports instruction in concepts developed in Chapter 9, Section 2 of KFCS.

A. CONCEPTS

To simplify the analysis of resonance within the flanged pipe the analysis is restricted to frequencies such that ka<<1, where k is the wave number and a is the radius of the pipe so only plane waves will propagate in the pipe. The frequencies of resonance for any mechanical system occur at the points where the imaginary component of input mechanical impedance, the input mechanical reactance, becomes zero. If the input mechanical resistance is non-zero, these zero points can also define antiresonance. Antiresonance and resonance can be separated by calculating the real component of input resistance at these zero points. Resonant points are those with small values of resistance and antiresonant points have large values of resistance. The equation for input mechanical impedance of a pipe with characteristic impedance $\rho_0 c$ and cross section S is

$$\frac{Z_{m0}}{\rho_0 cS} = \frac{\frac{Z_{mL}}{\rho_0 cS} + j\tan kL}{1 + j\frac{Z_{mL}}{\rho_0 cS}\tan kL} \tag{3.1}$$

where $\mathbf{Z}_{mL}$ for a flanged pipe is given by

$$\frac{Z_{mL}}{\rho_0 cS} = \frac{1}{2}(ka)^2 + j\frac{8}{3\pi}ka \tag{3.2}$$

By substituting (3.2) into (3.1) and solving for $\mathbf{Z}_{m0}/(\rho_0 cS)$ the resonant and antiresonant frequencies of the pipe can be found.

B. PROGRAM DESCRIPTION

The program has three arguments for user input. These are pipe length, pipe diameter, and sound speed. All three of these arguments are required to be entered for the program to function. Pipe length should be entered in meters while pipe diameter should be in centimeters and sound speed should be in meters/second.

The program determines a maximum frequency by using the requirement that wavelength be much larger than the radius of the pipe. This requirement generally means that the wavelength be an order of magnitude larger than the pipe radius and FPIPE.M uses this criterion to set the frequency limit.

The values for reactance and impedance have been normalized by the characteristic impedance of the fluid. This does not affect the frequencies of resonance or antiresonance so they are accurate for any fluid medium with the stated sound speed.

Figure 3.1 is a plot of the input reactance for a 1.5 meter long pipe which is 6 centimeters in diameter in water with a sound speed of 1500 m/s. The plot has been

magnified using the "zoom" command to show the zero crossings. Figure 3.2 shows a plot of the input resistance of the same pipe. The points where the reactance and the resistance become zero are the resonant points. Antiresonance occurs where the resistance becomes a maximum.

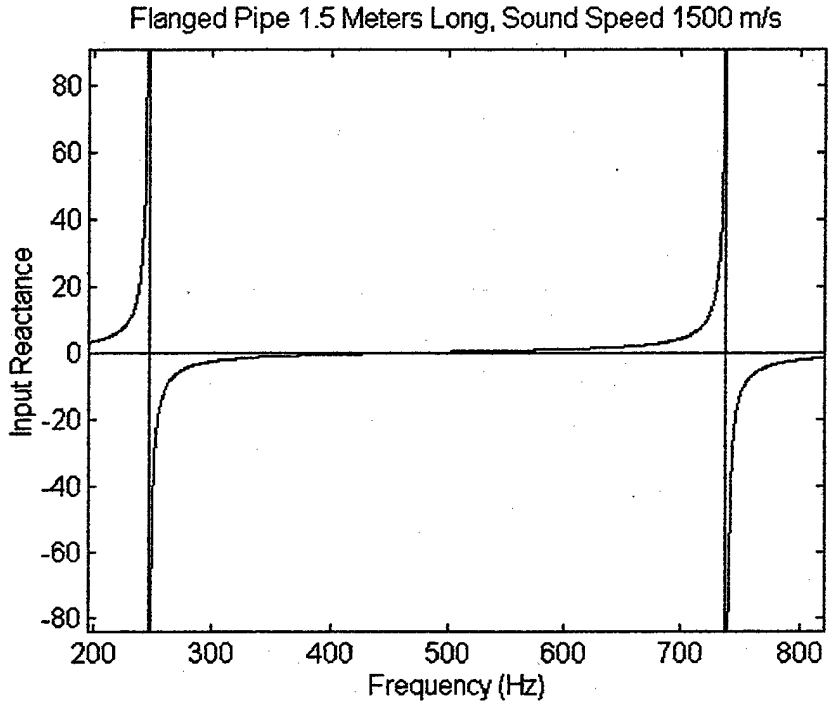

Figure 3.1 Input Reactance for a Flanged Pipe.

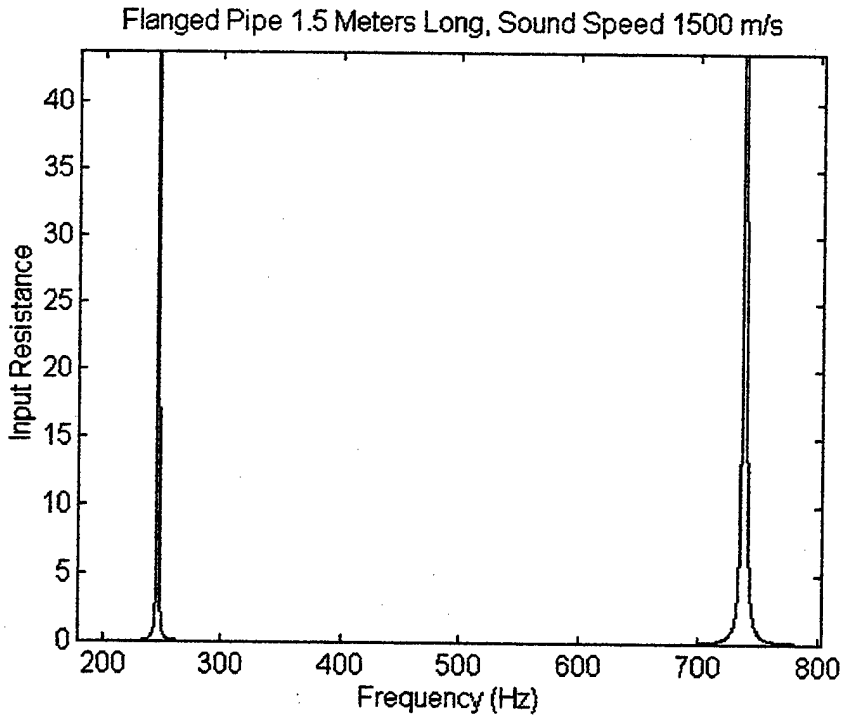

Figure 3.2 Input Resistance for a Flanged Pipe.

C. ALGORITHM fpipe.m

```
function fpipe(plength,pdiam,sndspd)

% FPIPE(pipe length,pipe diameter,sound speed) This program determines the load
% resistance and reactance for  a flanged pipe. Because it is assumed that the wavelength
% is much larger than the transverse dimension of the pipe, the  program only plots
% frequencies whose wavelength is an order of  magnitude greater than the pipe width
% chosen. The resistance and reactance information is plotted against frequency in two
% separate plots. This information can then be used to find values of resonance and
% anti-resonance for the pipe. "plength" should be in meters, "pdiam" should be in
```

```
% centimeters, and "sndspd" should be meters/sec.
%This is the user protection portion of the program.
if nargin~=3
        error('FPIPE requires three input arguments.')
end
chk_input=min([plength,pdiam,sndspd]);
if (chk_input<=0)
        error('All arguments of FPIPE must be positive.');
end
% This is the beginning of the computations.
pipe_radius=pdiam/(2*100);
max_freq=round(sndspd/(pipe_radius*10));  % long wavelength requirement
freqs=1:.5:max_freq;
k=(2.*pi.*freqs)./sndspd;
% This section finds the input mechanical impedance assuming that the pipe is flanged.
Zml=((((k.*pipe_radius).^2)./2)+(j.*(8/(3*pi)).*(k.*pipe_radius)));
Zm0=(Zml+(j.*tan(k.*plength))).(1+(j.*(Zml.*(tan(k.*plength)))));
Rm0=real(Zm0);
Xm0=imag(Zm0);
zero_ref=zeros(size(freqs));
```

```
figure(1)
plot(freqs,(Rm0));
ylabel('Input Resistance')
xlabel('Frequency (Hz)')
title(['Flanged Pipe ',num2str(plength),' Meters Long, Sound Speed ',...
        num2str(sndspd),' m/s'])
figure(2)
plot(freqs,zero_ref)
hold on
plot(freqs,(Xm0));
hold off
ylabel('Input Reactance')
xlabel('Frequency (Hz)')
title(['Flanged Pipe ',num2str(plength),' Meters Long, Sound Speed ',...
        num2str(sndspd),' m/s'])
```

IV. POWER RADIATION

The program PIPEPWR.M computes the power radiated from open ended pipes of different sizes for both flanged and unflanged terminations. This program supports concepts developed in Chapter 9, Section 3 of KFCS.

A. CONCEPTS

For a pipe of radius a terminated in a flange the transmission coefficient is

$$T_\pi = \frac{2(ka)^2}{[1+\frac{1}{2}(ka)^2]^2+(\frac{8}{3\pi})^2(ka)^2} \tag{4.1}$$

Similarly the transmission coefficient for an unflanged pipe is

$$T_\pi = \frac{(ka)^2}{[1+\frac{1}{4}(ka)^2]^2+(0.6ka)^2} \tag{4.2}$$

These equations are valid when ka<<1 so that the denominator is nearly one and the two equations become

$$T_\pi \doteq 2(ka)^2 \tag{4.3}$$

for the flanged pipe and

$$T_\pi \doteq (ka)^2 \tag{4.4}$$

for the unflanged pipe. PIPEPWR.M computes all of these transmission coefficients and plots them to allow the user to see the difference that this assumption makes.

B. PROGRAM DESCRIPTION

The program uses only a single argument, pipe diameter, which should be in centimeters. Figure 4.1 is a plot of the transmission coefficient using both equations for a flanged pipe. Figure 4.2 is the same type of plot for an unflanged pipe.

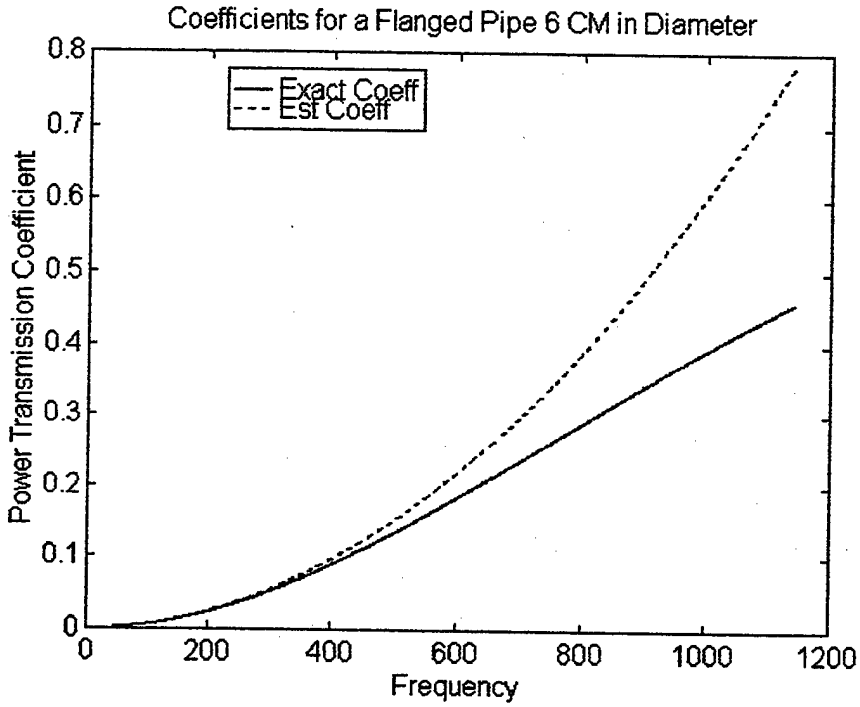

Figure 4.1 6cm Diameter Flanged Pipe Transmission Coefficient

Notice that in both the flanged and unflanged case the estimated transmission coefficient increases more rapidly than the exact coefficient. This is because we have

ignored the contribution of ka in the denominator and as frequency increases this term becomes more important.

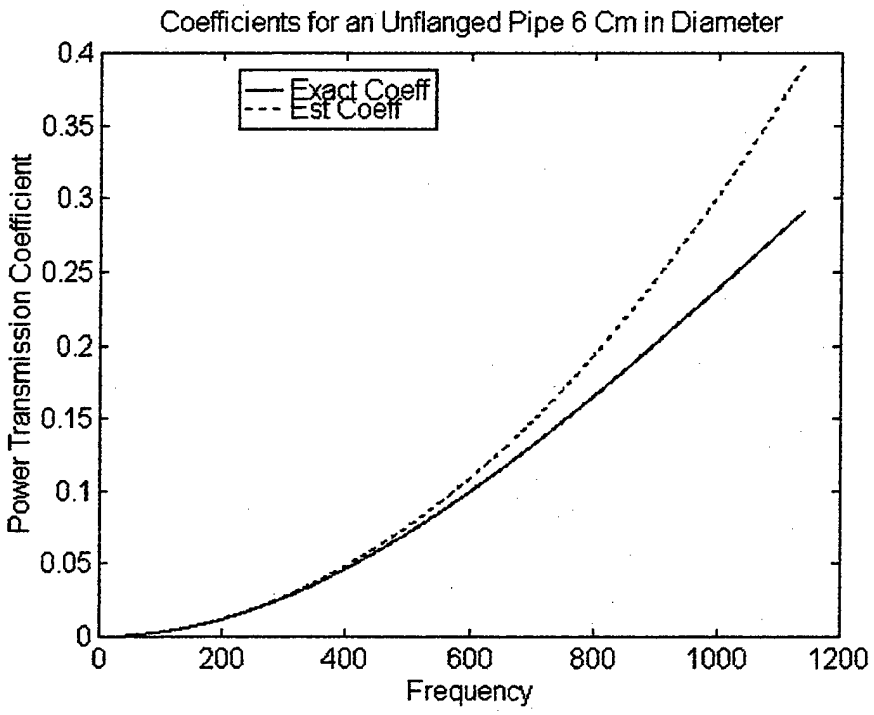

Figure 4.2 6cm Diameter Unflanged Pipe Transmission Coefficient

Figure 4.3 shows the power transmission coefficient for a flanged pipe 1 centimeter in diameter. Notice that the shape of the plot is very similar to that of Figure 4.1 which is a pipe 6 centimeters in diameter. It is evident that the transmission coefficient of the 1 cm pipe is much lower than that of the 6 cm pipe at the same frequency. This illustrates the idea that small acoustic devices are poor transmitters of relatively long wavelength signals.

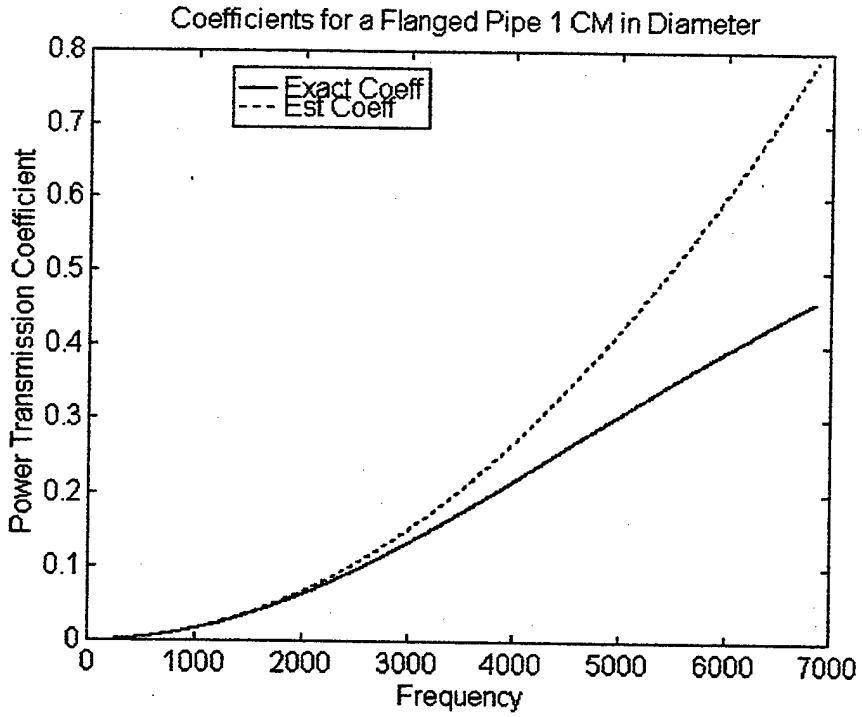

Figure 4.3 1cm Diameter Flanged Pipe Transmission Coefficient

Finally by comparing Figures 4.1 and 4.2 the flanged pipe's estimated coefficient is double that of the unflanged pipe's coefficient as is expected. Also as expected the actual transmission coefficient of the flanged pipe is much less than twice the actual coefficient of the unflanged pipe.

C. ALGORITHM pipepwr.m

```
function pipepwr(diam)
% PIPEPWR computes the power transmission coefficient for small diameter pipes
% which are flanged and unflanged. The upper curve of each plot is the exact solution
% when the assumption is only that the wavelength is much larger than the diameter of
```

```
% the pipe. The lower curve is the solution when it is assumed that the wavenumber
% times the radius of the pipe is much less than one. The argument "diam" is the
% diameter of the pipe. Values for diam should be in units of centimeters.
% This is the user protection portion of the program.
if nargin~=1
        error('PIPEPWR requires only one input argument.')
end
chk_input=min(diam);
if chk_input<=0
        error('All values for radius must be positive.')
end
% This begins the body of the program.
radius=diam/(2*100);
max_freq=343/(radius*10);   % Low frequency limit
freqs=1:(round(max_freq));
k=2*pi.*freqs./343;
ka2=(k.*radius).^2;
T_flange=(2.*(ka2))./(((1+(.5*ka2)).^2)+(((8/(3*pi))^2).*(ka2)));
T_fl_est=2*ka2;
```

```
T_unflange=(ka2)./(((1+(.25*ka2)).^2)+(.36*ka2));
T_unfl_est=ka2;
figure(1)
plot(freqs,T_flange,'y-')
hold on
plot(freqs,T_fl_est,'r:')
hold off
xlabel('Frequency')
ylabel('Power Transmission Coefficient')
title(['Coefficients for a Flanged Pipe ',...
        num2str(diam),' CM in Diameter'])
legend('Exact Coeff ','Est Coeff ')
figure(2)
plot(freqs,T_unflange,'y-')
hold on
plot(freqs,T_unfl_est,'r:')
hold off
xlabel('Frequency')
ylabel('Power Transmission Coefficient')
```

```
title(['Coefficients for an Unflanged Pipe ',...
        num2str(diam),' Cm in Diameter'])
legend('Exact Coeff ','Est Coeff ')
```

V. STANDING WAVES

The program STNDWAVE.M uses the single argument of standing wave ratio (SWR) to compute the load impedance over a range of nodal position values. The data is then plotted and the load impedance for a given SWR and node position can be read from the graph. This program supports instruction in concepts discussed in KFCS Chapter 9, Section 4.

A. CONCEPTS

A load impedance will cause a reflection of an acoustic wave from the load. The ratio of the reflected pressure wave to the incident pressure wave in a pipe of length L and cross sectional area S can be determined from the SWR by the following equation

$$\frac{B}{A}=\frac{SWR-1}{SWR+1} \tag{5.1}$$

By substituting (5.1) into

$$\frac{Z_{mL}}{\rho_0 cS}=\frac{1+\frac{B}{A}e^{j\theta}}{1-\frac{B}{A}e^{j\theta}} \tag{5.2}$$

the complex load impedance normalized by the characteristic impedance $\rho_0 c$ can be determined. If θ, the phase angle between the reflected and incident waves, is known the equation determining the phase angle is

$$\theta = 2k(L - x_1) - \pi \tag{5.3}$$

where x_1 is the position of the node closest to the load. The argument $(L-x_1)$ can be scaled so that it is measured as a fraction of a wavelength. This will allow cancellation with the k term and make the determination of θ independent of frequency.

B. PROGRAM DESCRIPTION

The program uses only the single user input of SWR. The results obtained from this program apply to any fluid medium, any sound speed, and any length pipe. It also applies to any frequency or pipe diameter as long as the long wavelength criterion ($\lambda >> a$) is met. Figure 5.1 shows the load resistance of a pipe with a SWR of 4. The resistive

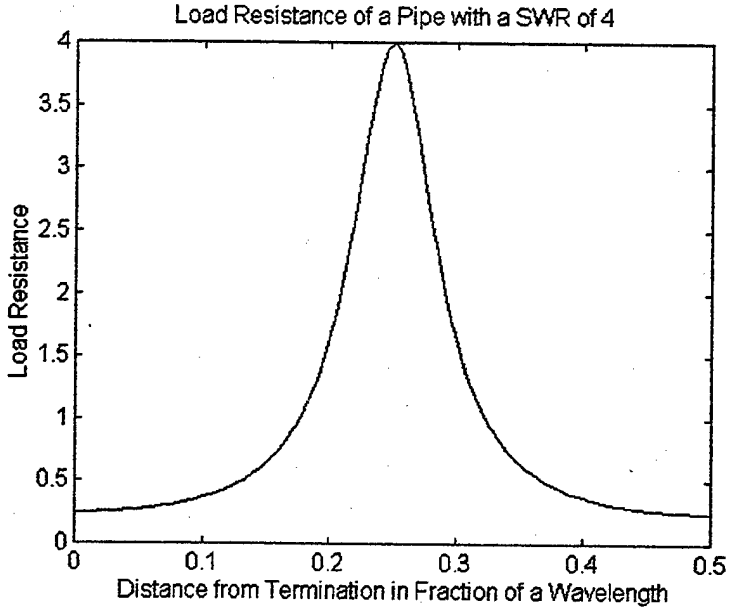

Figure 5.1 Load Resistance with SWR=4

load reaches a peak when the node is one quarter wavelength from the end. This peak resistive load corresponds to the point where the maximum energy is being transmitted to the medium. Figure 5.2 is a plot of the load reactance with a standing wave ratio of 4.

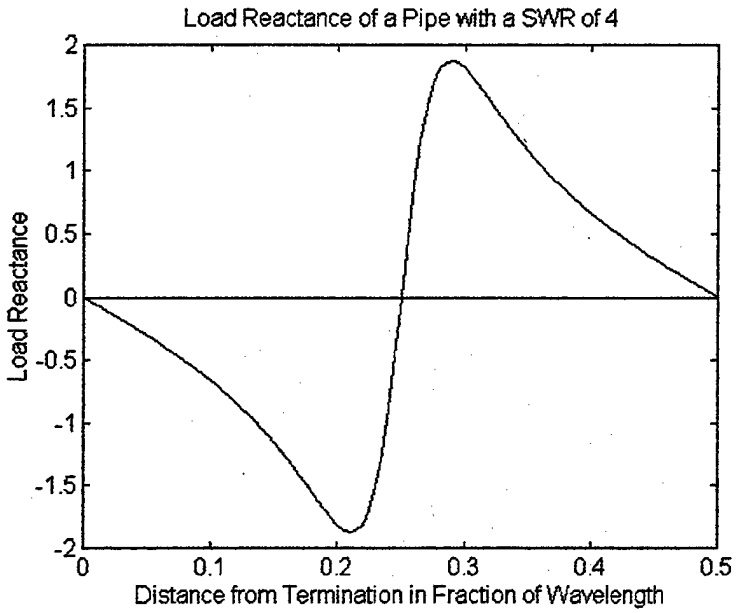

Figure 5.2 Load Reactance with SWR=4

Notice that the reactance passes through zero at the same quarter wave point where the resistance reached its peak. Figures 5.3 and 5.4 are plots of load resistance and impedance for a SWR of 2. Notice that as the SWR goes down the load resistance and impedance also go down.

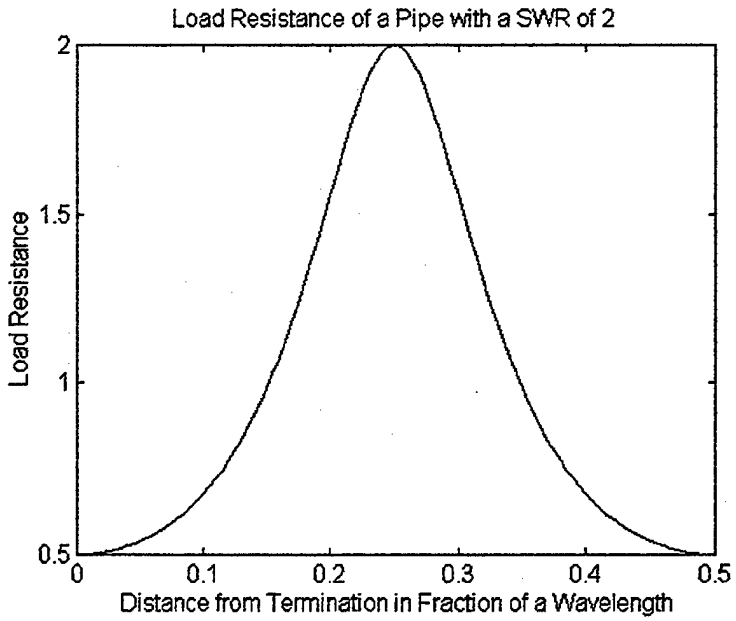

Figure 5.3 Load Resistance for a SWR=2

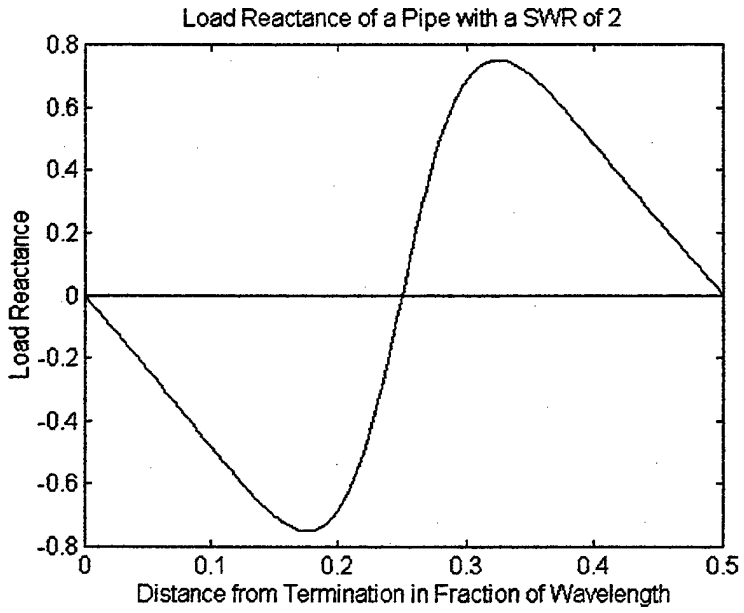

Figure 5.4 Load Reactance for a SWR=2

C. ALGORITHM stndwave.m

```
function stndwave(SWR)
% STNDWAVE computes and plots the load impedance for a pipe with a given
% measured standing wave ratio over a range of L-x distances. L-x is the distance from
% the load end of the pipe to the first acoustic node. This distance is measured as a
% fraction of a wave length. Since there are two nodes per wavelength, it is only
% necessary to plot up to one half of a wavelength. SWR is a unitless ratio of the
% acoustic pressure at a node to that at an antinode.
% This is the user protection portion of the program.
if nargin~=1
        error('STNDWAVE requires only one input argument.')
end
if SWR<1
        error('Values for SWR must be equal to or greater than 1.')
end
% This begins the body of the program.
off_dist=0:.005:.5;
press_ratio=(SWR-1)/(SWR+1);
theta=((4*pi).*off_dist)-pi;
Z=(1+(press_ratio.*exp(j.*theta))).(1-(press_ratio.*exp(j.*theta)));
```

```
x=[0,.5];

y=[0,0];

figure(1)

plot(off_dist,real(Z))

xlabel('Distance from Termination in Fraction of a Wavelength')

ylabel('Load Resistance')

title(['Load Resistance of a Pipe with a SWR of ',num2str(SWR)])

figure(2)

plot(off_dist,imag(Z))

hold on

plot(x,y)

hold off

xlabel('Distance from Termination in Fraction of Wavelength')

ylabel('Load Reactance')

title(['Load Reactance of a Pipe with a SWR of ',num2str(SWR)])
```

VI. COMBINED DRIVER-PIPE SYSTEMS

The program SYSRES.M determines resonance frequencies of combined driver and pipe assemblies. It plots system reactance against a scale normalized by kL which allows the user to determine resonance conditions. This program supports concepts discussed in Chapter 9, Section 6 of KFCS.

A. CONCEPTS

When combining two or more elements into a system it is important to understand how the components interact. Earlier chapters of KFCS addressed the resonance of drivers and sections of Chapter 9 address the resonance of pipes. When these are combined the system will resonate when the reactance of the driver is equal to the reactance of the pipe. Equation (6.1) shows the relationship of driver impedance to pipe impedance.

$$\frac{(\cos kL)(\sin kL)}{\sin^2 kL+(\alpha L)^2\cos^2 kL}=akL-\frac{b}{kL} \tag{6.1}$$

Where a is the ratio of the moving mass m of the driver to the mass of the fluid in the pipe.

$$a=\frac{m}{S\rho_0 L} \tag{6.2}$$

b is the ratio of the stiffness s of the suspension of the driver to the stiffness of the fluid in the pipe.

$$b=\frac{sL}{S\rho_0 c^2} \tag{6.3}$$

α is the absorption coefficient of the pipe.

B. PROGRAM DESCRIPTION

The program requires two inputs from the user, the values of a and b. Figure 6.1 is a plot of the reactances for a light, flexible driver and Figure 6.2 is a plot for a heavy, stiff driver.

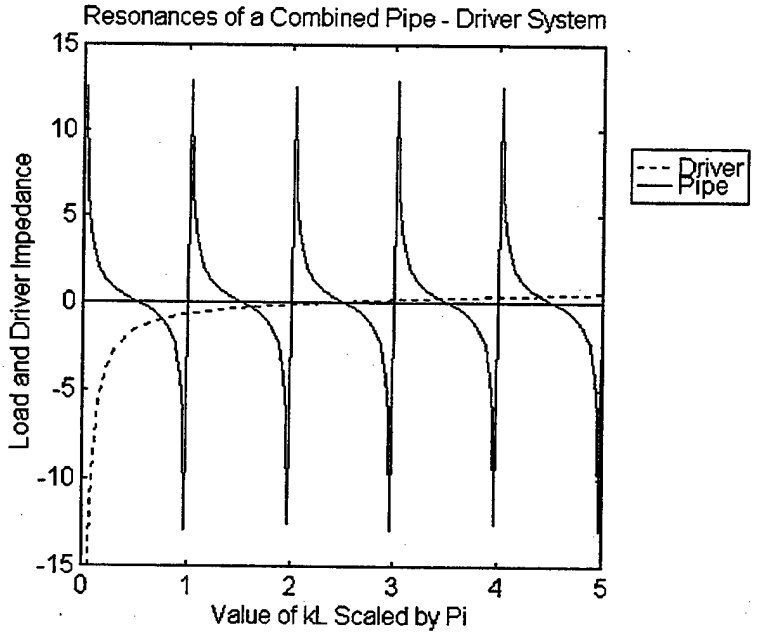

Figure 6.1 Light Flexible Driver.

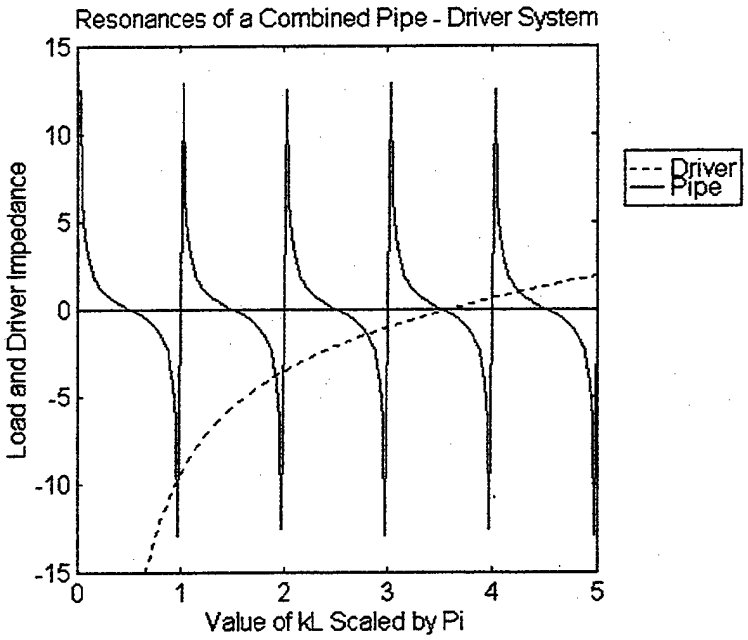

Figure 6.2 Heavy Stiff Driver

The following values were used for Figure 6.1; a=.04, and b=2.67. For Figure 6.2 the values were; a=.25, and b=32. Where two lines cross corresponds to a system resonance. It must be remembered that the values you pick for k need to satisfy the long wavelength criteria for the analysis to be valid.

C. ALGORITHM sysres.m

```
function sysres(a,b)
% SYSRES computes and plots the resonant points of a combined driver and pipe
% assembly. "a" is the ratio of the mass of the driver to the mass of the fluid in the tube.
% "b" is the ratio of the stiffness of the driver to the stiffness of the fluid in the tube. The
% tube is closed at one end.
```

```
% This is the user protection portion of the program.
if nargin~=2
        error('SYSRES requires two input arguments.')
end
chk_input=min([a,b]);
if chk_input<0
        error('Arguments for SYSRES must zero or greater.')
end
% This begins the body of the program.
domain=5*pi;
dom_int=domain/200;
kL=dom_int:dom_int:domain;
alpha_L=.01;
plot_dom=5;
plot_dom_int=5/200;
KL=plot_dom_int:plot_dom_int:plot_dom;
react_driver=(a.*kL)-(b./kL);
react_pipe=(cos(kL).*sin(kL))./(((sin(kL)).^2)+((alpha_L).^2).*...
                (cos(kL)));
```

```
max_react=1.2*max(react_pipe);
if max_react>15 max_react=15;
end
zero_pty=[0,0];
zero_ptx=[0,plot_dom];
figure(1)
plot(KL,react_driver,':')
hold on
plot(KL,react_pipe,'-')
plot(zero_ptx,zero_pty,'w')
hold off
axis([0,plot_dom,-max_react,max_react])
title('Resonances of a Combined Pipe - Driver System')
xlabel('Value of kL Scaled by Pi')
ylabel('Load and Driver Impedance')
legend('Driver ','Pipe',-1)
```

VII. RECTANGULAR CAVITY

The program RECTCAV.M displays the nodal pattern of a rectangular cavity. The user inputs the dimensions of the cavity and specifies the desired mode. The program computes the pattern along with the frequency to excite that mode. This program supports instruction in concepts discussed in Chapter 9, Section 7 of KFCS.

A. CONCEPTS

The boundary conditions of a rigid-walled rectangular cavity require that the pressure of a normal mode be a combination of cosines so the acoustic pressure is a maximum at the walls and solutions for the wave equation will be of the form

$$p_{lmn}=A_{lmn}(\cos k_{xl}x)(\cos k_{ym}y)(\cos k_{zn}z)e^{j\omega_{lmn}t} \tag{7.1}$$

with frequencies given by

$$\omega_{lmn}=\sqrt{\cos^2(l\pi/L_x)+\cos^2(m\pi/L_y)+\cos^2(n\pi/L_z)}$$

B. PROGRAM DESCRIPTION

The program RECTCAV.M requires six input arguments from the user. These are the lengths of the three sides of the cavity and the number of nodes in each of the three directions. The program then determines the resonant frequency of the given mode and plots the nodal pattern in three dimensions. The user can use the "view" command to

change which side of the cavity is being viewed. By changing the axis limits the user can look inside the box to see the amplitude at a specific location. The program plots in color and as such will not reproduce in black and white. In the plot, red is maximum amplitude or antinodes. A red area surrounded by blue and green is 180 degrees out of phase with a red area surrounded by yellow and orange. Light blue corresponds to nodes.

C. ALGORITHM rectcav.m

```
function rectcav(length,width,height,l,m,n)
% RECTCAV computes the resonant frequency of a rectangular cavity with a given
% mode. It then plots the nodal pattern and displays the resonant frequency.
% This is the user protection section.
if nargin~=6
        error('RECTCAV requires six arguments.')
end
chk_input=min([length,width,height]);
if (chk_input<0)|(chk_input==0)
        error('Dimensions of cavity must be greater than 0.');
end
chk_input=min([l,m,n]);
if (chk_input<0)
        error('Values for l,m, and n must be 0 or greater.');
```

```
end
if ((l==0)&(m==0)&(n==0))
        error('At least one value of l,m, or n must be greater than zero.');
end
% Start of computations.
no_steps=30;
sndspd=343;
freq=((((l*pi/length)^2)+((m*pi/width)^2)+((n*pi/height)^2))^.5);
freq=freq*(sndspd/(2*pi));
colormap(hsv);
max_length=max([length,width,height]);
x=linspace(0,(l*pi),no_steps);                % Computes arguments of cosine
y=linspace(0,(m*pi),no_steps);
z=linspace(0,(n*pi),no_steps);
if (l==0)
        x=zeros(1,no_steps);
end
if (m==0)
        y=zeros(1,no_steps);
end
```

```
if (n==0)
        z=zeros(1,no_steps);
end
Lx=linspace(0,length,no_steps); Ly=linspace(0,width,no_steps);
Lz=linspace(0,height,no_steps);
[X,Y]=meshgrid(x,y);  xysurf=(cos(X)).*(cos(Y));
coeff=ones(size(xysurf));
figure(1)
hold on
for ind=1:(size((Lz),2))
        Z=coeff.*(Lz(ind));
        xyzsurf=xysurf.*(cos(z(ind)));
        mesh(Lx,Ly,Z,xyzsurf)
end
axis([0,max_length,0,max_length,0,max_length])
view(3)
title(['Rectangular Cavity with Resonant Frequency of ',num2str(round(freq)),' Hz'])
xlabel('Length of Cavity')
ylabel('Width of Cavity')
zlabel('Height of Cavity')
```

VIII. RECTANGULAR WAVEGUIDES

The program GRPSPD.M computes the group speed and phase speed in a rectangular, rigid-walled waveguide. This program supports instruction in concepts developed in KFCS Chapter 9, Section 8.

A. CONCEPTS

The modes found in a waveguide for a specific drive frequency are given by

$$(\omega/c)^2=k_{lm}^2+k_z^2 \tag{8.1}$$

where ω is the driving frequency, k_z is the wave number of the propagating wave, and k_{lm} is the wave number of the lm mode. If $\omega/c < k_{lm}$ then the lm mode is evanescent. The wave number of a propagating mode is

$$k_z=\sqrt{(\omega/c)^2+k_{lm}^2} \tag{8.2}$$

The equation for the group speed is given by equation (8.3).

$$c_g=c\sqrt{1-(\omega_{lm}/\omega)^2} \tag{8.3}$$

So energy travels down the waveguide at a speed less than the speed of sound.

The phase speed, which is how fast a surface of constant phase will travel in the waveguide, is

$$c_p = \frac{c}{\sqrt{1-(\omega_{lm}/\omega)^2}} \tag{8.4}$$

Phase information travels down the waveguide at a speed greater than the speed of sound. In fact the phase speed becomes infinite when the driving frequency equals the cutoff frequency.

B. PROGRAM DESCRIPTION

The program requires the user to provide values of waveguide height and width and mode number. The height and width should be in meters and at least one of the mode numbers should be greater than zero. The program plots the straight line at a ratio of one which corresponds to plane wave propagation when both phase speed and group speed are equal to the speed of sound. The program also plots the mode input by the user and a mode which is two more in height and one more in width. Figure 8.1 is a plot of the (1,1) mode and the (3,2) mode for a 6 cm by 6 cm waveguide. As the driven frequency approaches infinity the phase and group speeds for all modes are equal to the sound speed. The (0,0) mode is a plane wave for all drive frequencies so its phase speed and group speed are both equal to the speed of sound for all frequencies. This graph can be used to determine group speed and phase speed for any mode in any fluid medium.

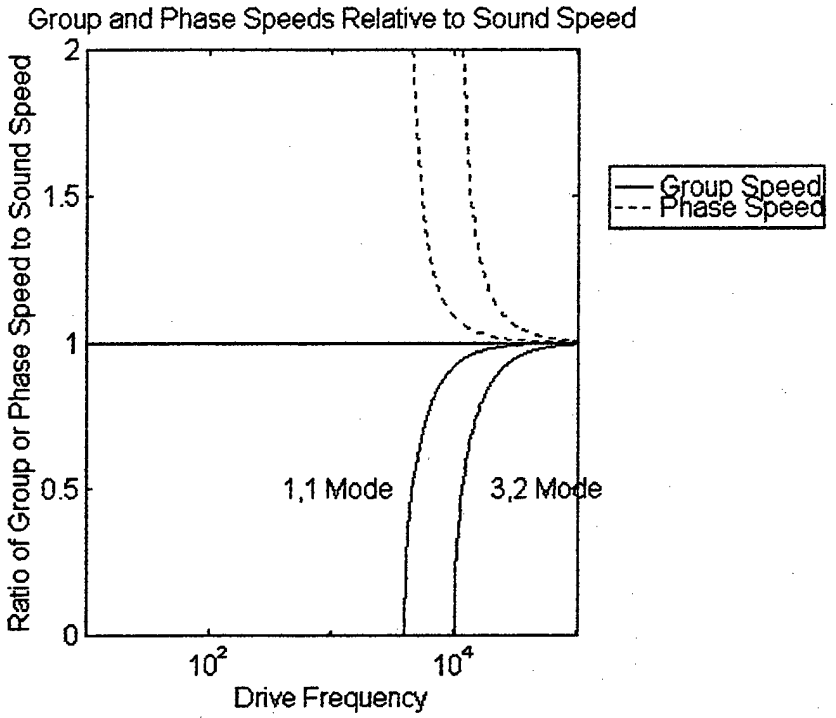

Figure 8.1 Group and Phase Speeds of a Rectangular Cavity 6cm by 6 cm.

C. ALGORITHM grpspd.m

```
function grpspd(height,width,l,m)

% GRPSPD plots the relation of group and phase speed to drive frequency. The user is

% required to provide values for waveguide height, waveguide width, and mode

% numbers. Height and width should be in meters and at least one mode number should

% be greater than 0.

% This is the user protection portion of the program.

if nargin~=4

        error('GRPSPD requires four input arguments.')
```

```
end
if min([height,width])<=0
        error('Height and width of waveguide must be positive.')
end
if min([l,m])<0
        error('Mode numbers must be zero or greater.')
end
if (l==0)&(m==0)
        error('At least one of the mode numbers must be greater than 0.')
end
% This begins the body of the program.
l2=l+2;
m2=m+1;
f_co=round((343/(2*pi))*((l*pi/height)^2 + (m*pi/width)^2)^.5);
f_co2=ceil((343/(2*pi))*((l2*pi/height)^2 + (m2*pi/width)^2)^.5);
f_end=10*f_co2;
freqs1=logspace(log10(f_co),log10(f_end),100);
freqs2=logspace(log10(f_co2),log10(f_end),100);
freqs0=logspace(1,log10(f_end),100);
w1=2*pi.*freqs1;
```

```
w2=2*pi.*freqs2;
wco1=2*pi*f_co;
wco2=2*pi*f_co2;
group0=ones(size(freqs0));
group1=real((1-((wco1./w1).^2)).^.5);
group2=real((1-((wco2./w2).^2)).^.5);
phase1=1./group1;
phase2=1./group2;
figure(1)
semilogx(freqs1,group1,'-')
hold on
semilogx(freqs1,phase1,':')
semilogx(freqs2,group2,'-')
semilogx(freqs2,phase2,':')
semilogx(freqs0,group0,'w-')
hold off
axis([10,f_end,0,2])
title('Group and Phase Speeds Relative to Sound Speed')
xlabel('Drive Frequency')
ylabel('Ratio of Group or Phase Speed to Sound Speed')
```

```
text(f_co/10,0.5,[num2str(l),',',num2str(m),' Mode'])

text(f_co2*2,0.5,[num2str(l2),',',num2str(m2),' Mode'])

legend('Group Speed','Phase Speed',-1)
```

IX. HELMHOLTZ RESONATOR

The Helmholtz resonator is an example of a lumped acoustic element. Where all dimensions are much smaller than a wavelength. The two programs HELMRES1.M and HELMRES2.M determine resonant frequencies of Helmholtz resonators over a range of neck lengths and a range of neck opening sizes respectively. These programs support instruction in concepts developed in KFCS Chapter 10, Section 2.

A. CONCEPTS

A Helmholtz resonator is a cavity with an opening. The frequency of its resonance can be determined by establishing the impedance characteristics of the cavity. The reactance of the resonator is controlled by a mass-type element and a stiffness-like element. For the Helmholtz resonator the mass is provided by the fluid in the neck. Because the neck is open end effects must be accounted for so the equation of effective mass is given by: $m=\rho_0 SL'$ where L' for a flanged neck as: $L'=L+1.7a$, a being the diameter of the neck. For an unflanged neck the equation is: $L'=L+1.5a$. The stiffness is determined by the volume V of the cavity, as the fluid in the cavity acts like a spring as it is compressed. The stiffness is given by (9.1)

$$s=\rho_0 c^2 \frac{S^2}{V} \tag{9.1}$$

where S is the area of the neck. So now the reactance can be determined by substituting

into the equation

$$X_m=(\omega m-s/\omega) \tag{9.2}$$

and solving this when reactance equals zero gives

$$\omega_0=c\sqrt{\frac{S}{L'V}} \tag{9.3}$$

This relationship holds for any cavity as long as the long wavelength criterion is met.

B. PROGRAM DESCRIPTION

The programs HELMRES1.M and HELMRES2.M both compute resonant frequencies for cavities and require three inputs from the user. For HELMRES1.M, the user is required to input values of sound speed, neck diameter, and cavity volume while HELMRES2.M requires sound speed, neck length, and cavity volume. Both programs plot resonances for flanged and unflanged necks but only the flanged neck results are shown here. Figure 9.1 shows the resonant frequencies over a range of neck lengths and Figure 9.2 shows the frequencies over a range of neck diameters.

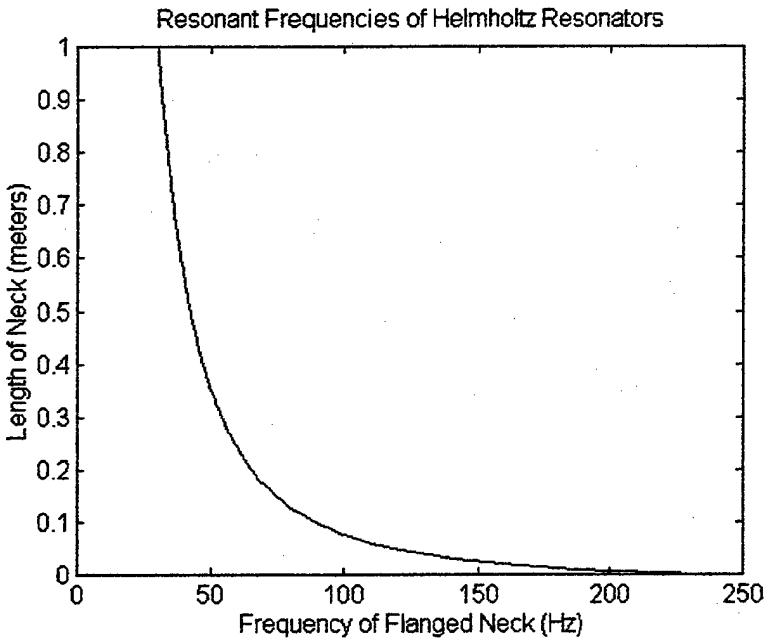

Figure 9.1 Resonances of Different Neck Lengths

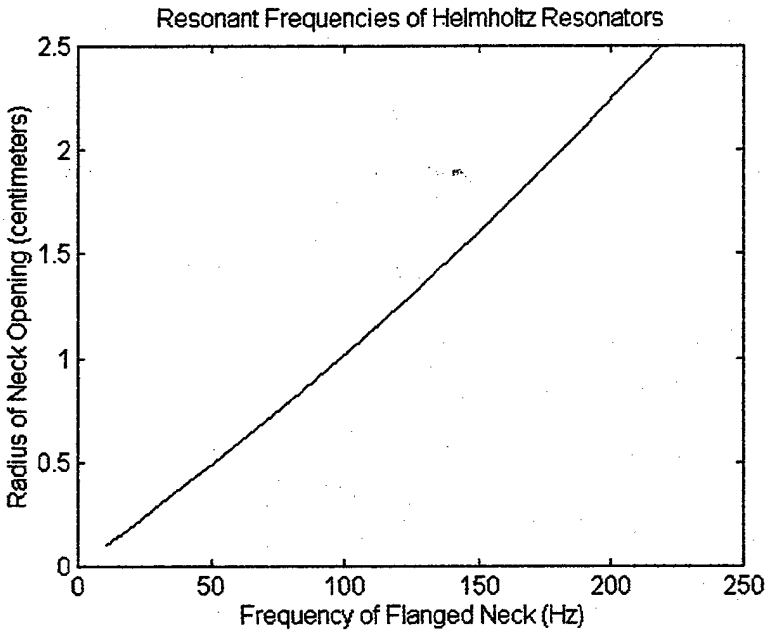

Figure 9.2 Resonances for Different Neck Openings

C. ALGORITHM helmres1.m

```
function helmres1(sndspd,hole_diam,volume)
% HELMRES1 computes resonant frequencies of a Helmholtz Resonator as a function of
% neck length. "sndspd" should be in units of meters/sec, "hole_diam" should be in
% centimeters, and "volume" should be in cubic centimeters.
% This is the user protection portion of the program.
if nargin~=3
        error('HELMRES1 requires three input arguments.')
end
chk_input=min([sndspd,hole_diam,volume]);
if chk_input<=0
        error('All arguments of HELMRES1 must be positive numbers.')
end
% This starts the body of the program.
hole_diam=hole_diam/100;
volume=volume/1e6;
radius=hole_diam/2;
L=0:.01:1;
flangeL=L+(1.7*radius);
unflangeL=L+(1.5*radius);
```

```
S=pi*(radius^2);
flange_arg=((S./(flangeL.*volume)).^.5);
unflange_arg=((S./(unflangeL.*volume)).^.5);
flange_freq=flange_arg.*(sndspd/(2*pi));
unflange_freq=unflange_arg.*(sndspd/(2*pi));
figure(1)
plot(flange_freq,L,'w')
xlabel('Frequency of Flanged Neck (Hz)')
ylabel('Length of Neck (meters)')
title('Resonant Frequencies of Helmholtz Resonators')
figure(2)
plot(unflange_freq,L,'w')
xlabel('Frequency of Unflanged Neck (Hz)')
ylabel('Length of Neck (meters)')
title('Resonant Frequencies of Helmholtz Resonators')
```

D. ALGORITHM helmres2.m

```
function helmres2(sndspd,neck_length,volume)
% HELMRES2 computes resonant frequencies of a Helmholtz Resonator as a function of
% neck opening size. "sndspd" should be in units of meters/sec, "neck_length" should be
% in centimeters, and  "volume" should be in cubic centimeters.
```

```
% This is the user protection portion of the program.
if nargin~=3
        error('HELMRES2 requires three input arguments.')
end
chk_input=min([sndspd,neck_length,volume]);
if chk_input<=0
        error('All arguments of HELMRES2 must be positive numbers.')
end


% This starts the body of the program.
volume=volume/1e6;          % converts volume to cubic meters
radius=.001:.001:.025;
Radius=.1:.1:2.5;
L=neck_length/100;          % converts neck length to meters
flangeL=L+(1.7.*radius);    % These compute the equivalent
unflangeL=L+(1.5.*radius);  % neck lengths.
S=pi.*(radius.^2);
flange_arg=((S./(flangeL.*volume)).^.5);
unflange_arg=((S./(unflangeL.*volume)).^.5);
flange_freq=flange_arg.*(sndspd/(2*pi));
```

```
unflange_freq=unflange_arg.*(sndspd/(2*pi));

figure(1)

plot(flange_freq,Radius,'w')

xlabel('Frequency of Flanged Neck (Hz)')

ylabel('Radius of Neck Opening (centimeters)')

title('Resonant Frequencies of Helmholtz Resonators')

figure(2)

plot(unflange_freq,Radius,'w')

xlabel('Frequency of Unflanged Neck (Hz)')

ylabel('Radius of Neck Opening (centimeters)')

title('Resonant Frequencies of Helmholtz Resonators')
```

X. RESONANT BUBBLES

A small bubble in water can be viewed as a lumped acoustic element. The two programs explore the resonant frequencies and the acoustic losses of small air bubbles in water. The program only analyzes to a depth of 100 meters as there are not many air bubbles of any consequence below this depth. These programs support instruction in concepts explored in Chapter 10, Section 3 of KFCS.

A. CONCEPTS

As in the case of the Helmholtz resonator, the bubble can be treated as a lumped acoustic element. Therefore its resonant frequency is controlled by the same mechanical properties which lets us determine that the resonant frequency is

$$\omega_0 = \frac{1}{a}\sqrt{\frac{3\gamma \mathcal{P}_0}{\rho_0}} \qquad (10.1)$$

where a is the radius of the bubble, γ is the ratio of specific heat of gas in the bubble, $\mathcal{P}_0$ is the ambient pressure, and ρ_0 is the density of the water.

These small bubbles can be important in determining the losses of a transmitted acoustic wave because they absorb, reflect, and reradiate the sound energy. The total effect of energy losses due to the bubble is described by its extinction cross section. It is defined as the ratio of energy absorbed and scattered to the intensity of the incident energy. This can be broken down into a scattering cross section and an absorption cross

section. The ratio of the scattering cross section to the extinction cross section gives us an idea of whether scattering or absorption is the more dominant loss mechanism and can be found by using the equation

$$\frac{\sigma_s}{\sigma}=k_0 aQ \tag{10.2}$$

where k_0 is the wave number of the resonant frequency in water and the quality factor Q is given by

$$Q=\frac{1}{k_0 a+1.6\times10^{-4}\sqrt{\omega_0}} \tag{10.3}$$

B. PROGRAM DESCRIPTION

1. bubble.m

This program is used to determine the frequency at which a bubble of a specified diameter will resonate. The diameter of the bubble should be entered by the user in millimeters. The depths analyzed range from 0 to 100 meters depth. It is important to realize that the program determines the resonance for a bubble with a constant diameter whatever the depth. However, a bubble which starts at a certain size at 100 meters depth will grow larger as it rises and so this will also contribute to a change in the resonant frequency. Figure 10.1 is a plot of the resonant frequencies of two bubbles, one with a

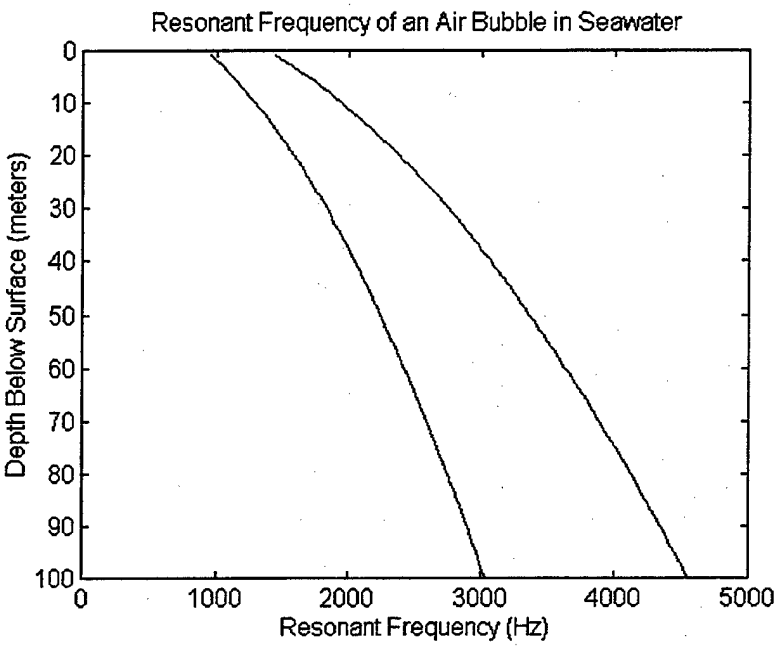

Figure 10.1 Resonant frequencies of an air bubble 2 mm in diameter and one of 3mm in diameter.

diameter of 2 millimeters and one with a diameter of 3 millimeters. From equation (10.1) it can be seen that as the bubble diameter decreases the resonant frequency will increase, so the 2mm bubble is the righthand curve and the 3mm bubble is the lefthand curve. The user can input as many different sizes of bubbles as desired in a single chart by using vector notation to define the argument of bubble diameter.

2. bubble2.m

This program determines the ratio of scattering cross section to extinction cross section at resonant frequency which allows us to determine which loss mechanism is dominant. Figure 10.2 shows a plot of this ratio for the same two bubbles of 2 mm and 3 mm in diameter. Again the 2 mm bubble is on the left and the plot of the 3 mm bubble is

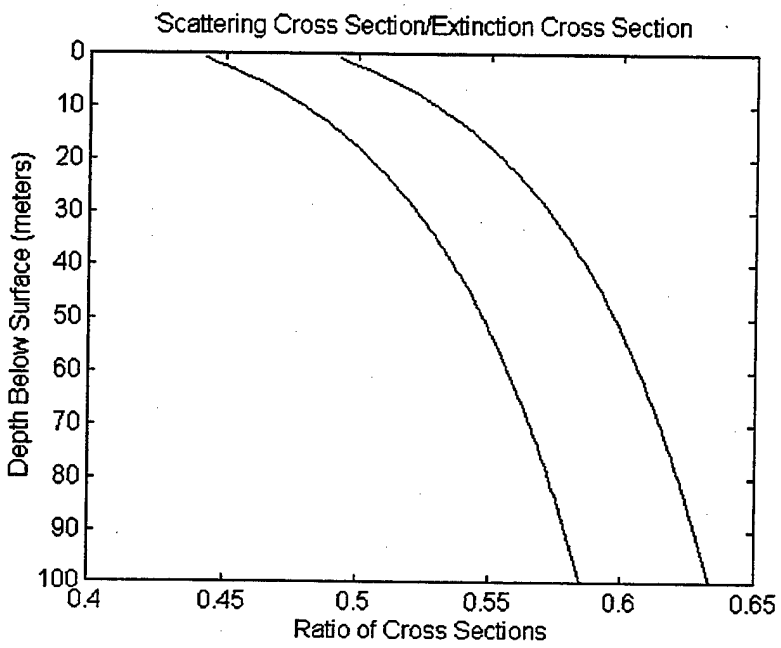

Figure 10.2 Ratio of scattering cross section to extinction cross section for a 2mm diameter bubble and a 3mm diameter bubble.

on the right. It can be seen from the graph that as depth increases the scattering losses become more important than the absorption losses. Again the user can plot as many bubble sizes on the same plot as desired by using vector notation.

C. ALGORITHM bubble.m

```
function bubble(radius)
% BUBBLE computes the resonant frequency of air bubbles within the water column
% down to a depth of 100 meters. Any number of values for radius may be entered by
```

```
% using vector notation.  i.e. BUBBLE([val1,val2,...,valN]). This allows the user to
% compare air bubbles of different size on the same graph. Values for radius should be in
% units of millimeters.
% This is the user protection portion of the program.
if nargin~=1
        error('To enter more than one value for radius use vector notation.')
end
chk_input=min(radius);
if chk_input<=0
        error('All values for radius must be positive.')
end
% This begins the body of the program.
depth=1:100;
radius=radius./1000;
spec_heat=1.01;
rho=1026;
pressure=1.013e5+((1026*9.81).*depth);
[P,R]=meshgrid(pressure,radius);
omega=((3*spec_heat.*P./rho).^.5)./R;
freq=omega/(2*pi);
```

```
figure(1)
plot(freq,depth,'w')
axis('ij')
xlabel('Resonant Frequency (Hz)')
ylabel('Depth Below Surface (meters)')
title('Resonant Frequency of an Air Bubble in Seawater')
```

D. ALGORITHM bubble2.m

```
function bubble2(radius)
% BUBBLE2 computes the ratio of the scattering cross section to the extinction cross
% section for different size air bubbles in the water column down to a depth of 100
%  meters. Any number of values for radius may be entered by using vector notation.
% i.e. BUBBLE2([val1,val2,...,valN]). This allows the user to compare air bubbles of
% different size on the same graph. Values for radius should be in units of millimeters.
% This is the user protection portion of the program.
if nargin~=1
        error('To enter more than one value for radius use vector notation.')
end
chk_input=min(radius);
if chk_input<=0
        error('All values for radius must be positive.')
```

```
end
% This begins the body of the program.
depth=1:100;
radius=radius./1000;
spec_heat=1.01;
rho=1026;
sndspd=1500;
pressure=1.013e5+((1026*9.81).*depth);
coeff=ones(size(pressure));
[C,R]=meshgrid(coeff,radius);
Radius=C.*R;
[P,R]=meshgrid(pressure,radius);
omega=((3*spec_heat.*P./rho).^.5)./R;
k0=omega/sndspd;
Q=1./((k0.*Radius)+((1.6e-4).*(omega.^.5)));
ratio=k0.*Radius.*Q;
figure(1)
plot(ratio,depth,'w')
axis('ij')
xlabel('Ratio of Cross Sections')
```

```
ylabel('Depth Below Surface (meters)')

title('Scattering Cross Section/Extinction Cross Section')
```

XI. REFLECTED WAVES IN PIPES

The program PIPEWAVE.M displays the reflection and transmission properties of a pipe at an abrupt change in diameter. This program supports instruction in concepts developed in Chapter 10, Section 5 of KFCS.

A. CONCEPTS

The general equation for sound power reflection coefficient is given by

$$R_\pi = \frac{(R_0 - \rho_0 c/S)^2 + X_0^2}{(R_0 + \rho_0 c/S)^2 + X_0^2} \tag{11.1}$$

where R_0 and X_0 are resistance and reactance of the termination.

In the case of an infinite pipe the terminating acoustic impedance is purely resistive, $\rho_0 c S_2$, simplifying the equation

$$R_\pi = \frac{(S_1 - S_2)^2}{(S_1 + S_2)^2} \tag{11.2}$$

The power transmission coefficient can be found in a similar way and is

$$T_\pi = \frac{4 S_1 S_2}{(S_1 + S_2)^2} \tag{11.3}$$

The power transmitted depends only on the cross sectional areas of the two parts of the pipe. Therefore the power transmission for any pipe in any medium at any frequency can be found if the ratio of areas is known.

B. PROGRAM DESCRIPTION

The program PIPEWAVE.M has no user inputs and plots the reflection and transmission coefficients for an infinite pipe with a range of cross sectional area ratios. Figure 11.1 is a plot of the reflection and transmission coefficients. Several things can be

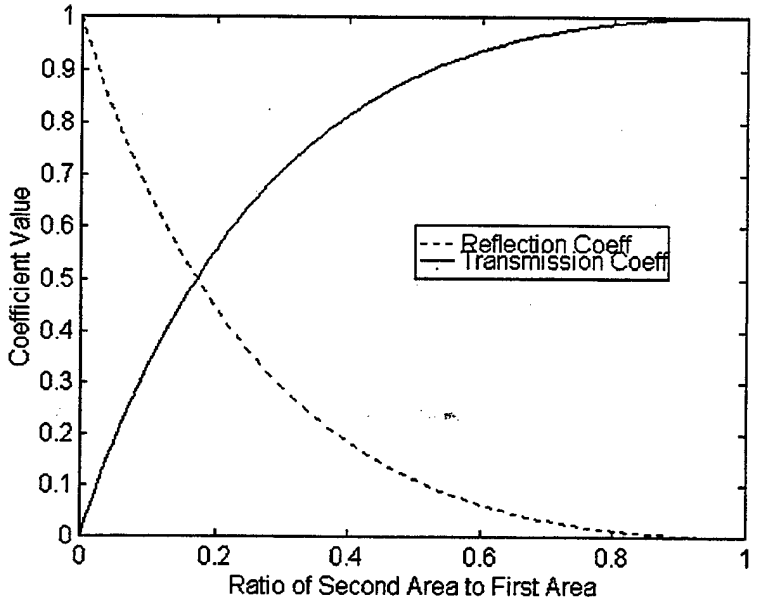

Figure 11.1 Reflection and Transmission Coefficients in a Pipe

seen from this figure. As the two sections of pipe get closer to the same size the transmission coefficient becomes one. This is expected as it is one long pipe of constant cross section and the acoustic impedance remains unchanged. As the ratio of areas tends

towards zero, the reflection coefficient becomes one, which is the same as a pipe being terminated in a rigid cap. Notice the sum of reflection and transmission coefficients is always one, this is expected as no loss mechanisms are incorporated in the model.

C. ALGORITHM pipewave.m

```
function pipewave
% PIPEWAVE computes and displays reflection and transmission coefficients for waves
% in a long pipe which encounter a reduction in the pipe diameter. The program requires
% no user input and plots for a range of pipe diameters.
A=0:.01:1;
R=((1-A).^2)./((1+A).^2);
T=(4.*A)./((1+A).^2);
figure(1)
plot(A,R,'w:')
hold on
plot(A,T,'w-')
hold off
ylabel('Coefficient Value')
xlabel('Ratio of Second Area to First Area')
legend('Reflection Coeff', 'Transmission Coeff')
```

XII. ACOUSTIC FILTERS

A. LOW-PASS FILTERS

1. Concepts

A short section of large diameter pipe of length L and area $S_1=\pi a^2$ inserted in a smaller pipe of area S acts as a low pass filter. When kL<<1 and ka<<1, the reactance is

$$X_b=-\frac{\rho_0 c^2}{\omega S_1 L} \tag{12.1}$$

Substituting into the equation for transmission coefficient of a branch

$$T_\pi=\frac{R_b^2+X_b^2}{(\frac{\rho_0 c}{2S}+X_b^2)^2+X_b^2} \tag{12.2}$$

where R_b and X_b are the input resistance and reactance of the branch gives

$$T_\pi=\frac{1}{1+(\frac{1}{2}\frac{S_1}{S}kL)^2} \tag{12.3}$$

for values of kL<<1.

A more complicated analysis results in an equation good for all values of kL

$$T_{\pi}=\frac{4}{4\cos^2 kL+(\frac{S_1}{S}+\frac{S}{S_1})^2\sin^2 kL} \qquad (12.4)$$

Both of these equations are used in the program LOPASSAC.M.

2. Program Description

LOPASSAC.M requires arguments for diameter of the pipe and the filter section. Both of these values should be in centimeters.

The program computes the estimated values of T_{π} and the more exact solution and plots them on the same graph. Figure 12.1 is a plot of the transmission coefficient for a pipe 2 cm in diameter and a filter section 4 cm in diameter. The coefficients are plotted against values of kL from 0 to 1. Notice the estimated values of transmission drop off continuously as kL increase. However the more exact solution shows the transmission coefficient reaches a minimum when kL is $\pi/2$, or one-quarter wavelength. It then increases to a maximum of 1 when kL is equal to a half wavelength. This pattern continues for values of kL greater than 1. One other interesting aspect of this program is that it can be used to determine the transmission coefficient of a pipe with a restriction. The estimated transmission coefficient will only be good for very small values of kL but the coefficient determined by the more exact solution is correct over a larger range of frequencies as long as $ka \ll 1$.

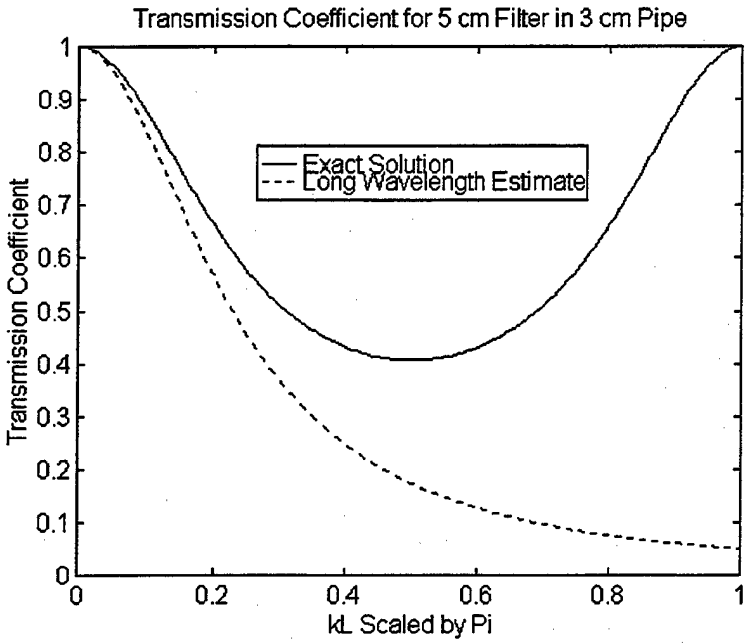

Figure 12.1 Pipe with diameter of 2cm and filter section 4cm in diameter.

3. Algorithm lopassac.m

```
function lopassac(diam_pipe,diam_filt)

% LOPASSAC computes and plots the transmission characteristics of a low pass acoustic

% filter in a pipe. "diam_pipe" and "diam_filt" can be any units the user wishes as long

% as they are consistent.

% This is user protection portion of program.


if nargin~=2

        error('LOPASSAC requires two input arguments.')

end
```

```
if (min([diam_pipe,diam_filt])<=0)
        error('Arguments for LOPASSAC must be positive.')
end
% This begins the body of the program.
steps=200;
domain=(1*pi);
dom_step=domain/steps;
plot_dom=domain/pi;
plot_dom_step=plot_dom/steps;
kL=0:dom_step:domain;
KL=0:plot_dom_step:plot_dom;
S1=((diam_filt)^2)*pi;
S=((diam_pipe)^2)*pi;
T_exac=4./((4.*(cos(kL)).^2)+((((S1/S)+(S/S1))^2).*(sin(kL)).^2));
T_est=1./(1+((.5*(S1/S).*kL).^2));
figure(1)
plot(KL,T_exac,'-')
hold on
plot(KL,T_est,':')
hold off
```

```
title(['Transmission Coefficient for ',num2str(diam_filt),...
    ' cm Filter in ',num2str(diam_pipe),' cm Pipe'])
xlabel('kL Scaled by Pi')
ylabel('Transmission Coefficient')
legend('Exact Solution', 'Long Wavelength Estimate')
```

B. HIGH-PASS FILTERS

1. Concepts

A small branch of length L and radius a, in the wall of a pipe, will act as a high-pass filter. If kL<<1 and ka<<1, the branch impedance is

$$Z_b = \frac{\rho_0 c k^2}{4\pi} + j\frac{\rho_0 L' \omega}{\pi a^2} \tag{12.5}$$

Substituting this back into equation (12.2) will give the transmission coefficient. The ratio of branch resistance to reactance is

$$\frac{R_b}{X_b} = \frac{ka^2}{4L'} \tag{12.6}$$

where L′ is the effective length and is equal to L + 1.5a. If $ka^2 << 1$, we can assume the resistance is negligible and the transmission coefficient is

$$T_{\pi}=\frac{1}{1+(\frac{\pi a^2}{2SL'k})^2} \tag{12.7}$$

The cutoff frequency, defined as the frequency where $T_{\pi} = 0.5$, is given by

$$f_c=\frac{a^2c}{4SL'} \tag{12.8}$$

2. Program Description

HIPASSAC.M computes the estimated transmission coefficient for frequencies up to 2500 Hz. The user is required to provide four input arguments. Branch diameter and length and pipe diameter should be in centimeters and sound speed should be in meters/second. The program plots both the more exact transmission coefficient and the estimated coefficient.

Figure 12.2 is a plot for a 6 cm diameter pipe in air with an opening 3.1 cm in diameter and a length of 6 mm. Looking at equation (12.7) attenuation can be increased by increasing the diameter of the opening or by decreasing the length of the opening.

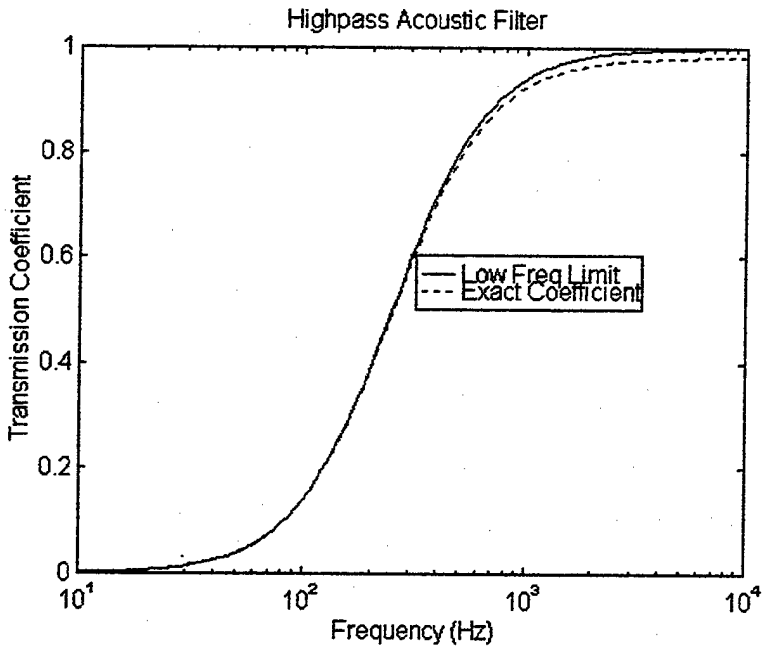

Figure 12.2 High-pass filter.

3. Algorithm hipassac.m

```
function hipassac(hole_diam,hole_length,pipe_diam,sndspd)
% HIPASSAC computes and plots the transmission characteristics of a high pass acoustic
%filter in a pipe. Hole diameter, hole length and pipe diameter should all be in units of
% centimeters. Sound speed should be in meters/second.
% This is user protection portion of program.
if nargin~=4
        error('HIPASSAC requires four input arguments.')
end
if (min([hole_diam,pipe_diam])<=0)
```

```
        error('Hole diameter and pipe diameter must be positive.')
end
if (hole_length<0)
        error('Hole length must be zero or greater.')
end
if (sndspd<=0)
        error ('Sound speed must be positive.')
end
% This begins the body of the program.
freqs=logspace(1,4,100);
w=2*pi.*freqs;
k=2*pi.*freqs./sndspd;
S=((pipe_diam/(2*100))^2)*pi;
hole_length=hole_length/100;
a=hole_diam/(2*100);
L_eff=hole_length+(1.5*a);
rho=1.2;
R=rho*sndspd.*(k.^2)./(4*pi);
X=rho*L_eff.*w./(pi*(a^2));
```

```
T_est=1./(1+(((pi*(a^2))./(2*S*L_eff.*k)).^2));

T=((R.^2)+(X.^2))./((((rho*sndspd/(2*S)) + R).^2)+(X.^2));

figure(1)

semilogx(freqs,T_est,'-')

hold on

semilogx(freqs,T,':')

hold off

xlabel('Frequency (Hz)')

ylabel('Transmission Coefficient')

title('Highpass Acoustic Filter')

legend('Low Freq Limit', 'Exact Coefficient')
```

C. BAND-STOP FILTERS

1. Concepts

A Helmholtz resonator attached to a pipe acts as a band-stop filter. The resonator does not absorb sound so resistance is zero and the reactance is

$$X_r=\rho_0\left(\frac{\omega L'}{S_r}-\frac{c^2}{\omega V}\right) \qquad (12.9)$$

where V is the volume of the resonator and S_r is the area of the resonator opening. Substituting into equation 12.2 the transmission coefficient is

$$T_{\pi}=\frac{1}{1+\frac{c^2}{4S^2(\frac{\omega L'}{S_r}-\frac{c^2}{\omega V})^2}} \tag{12.10}$$

The center frequency of the band-stop filter is given by the resonant frequency of the resonator

$$f_s=\frac{c}{2\pi}\sqrt{\frac{S_r}{L'V}} \tag{12.11}$$

2. Program Description

The program BDSTOPAC.M requires four input arguments. Neck diameter and length, and pipe diameter should be in centimeters and volume should be cubic centimeters. Figure 12.3 is a plot of a band-stop filter with an opening 3.1 cm in diameter, a neck length of 6 mm, a volume of 1120 cc, and a pipe diameter of 6 cm.

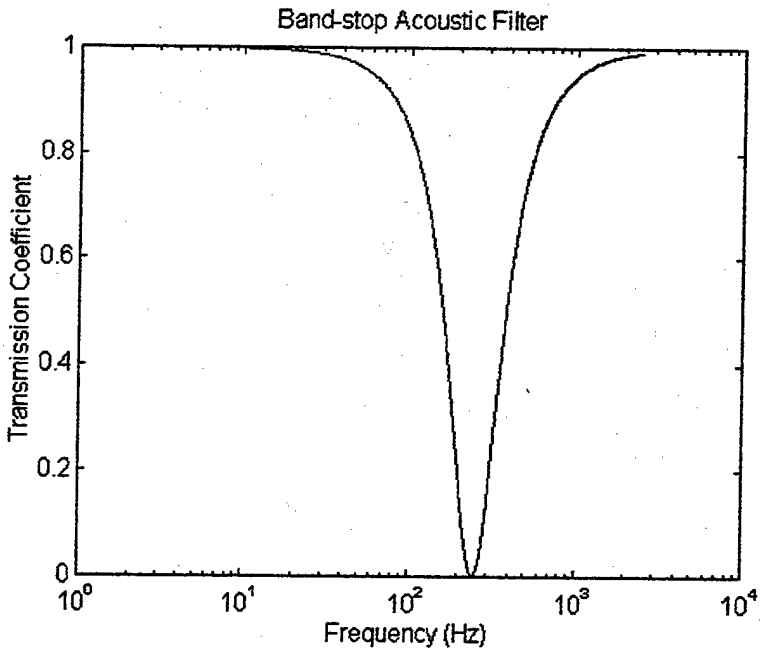

Figure 12.3 Band-stop filter.

3. Algorithm bdstopac.m

```
function bdpassac(hole_diam,hole_length,res_vol,pipe_diam)
% BDSTOPAC.M computes and plots the transmission characteristics of a band pass
% acoustic filter in a pipe. Hole diameter, hole length and pipe diameter should all be in
% units of centimeters. Resonator volume should be in units of cubic centimeters.
% This is user protection portion of program.
if nargin~=4
        error('BDSTOPAC requires three input arguments.')
end
if (min([hole_diam,pipe_diam,res_vol])<=0)
```

```
        error('Hole diameter, pipe diameter and resonator volume must be positive.')
end
if (hole_length<0)
        error('Hole length must be zero or greater.')
end
% This begins the body of the program.
freqs=1:2500;
omega=2*pi.*freqs;
sndspd=343;
S=((pipe_diam/(2*100))^2)*pi;
hole_length=hole_length/100;
a=hole_diam/(2*100);
L_eff=hole_length+(1.7*a);
Sb=pi*(a^2);
vol=res_vol/(1e6);
T=1./(1+((sndspd^2)./(4*(S^2).*((((omega.*L_eff)./Sb)-...
                ((sndspd^2)./(omega.*vol))).^2))));
figure(1)
semilogx(freqs,T,'w')
xlabel('Frequency (Hz)')
```

```
ylabel('Transmission Coefficient')

title('Band-stop Acoustic Filter')
```

XIII. RECEIVER OPERATOR CHARACTERISTICS

A. CONCEPTS

The program ROC.M and its several supporting programs are designed to allow the user to determine Receiver Operator Characteristics curves. This program supports instruction in concepts discussed in Chapter 11 of KFCS.

B. PROGRAM DESCRIPTION

This program actually consists of two main programs and 5 smaller support programs. Figure 1 is the user interface that is generated by the program ROC.M.

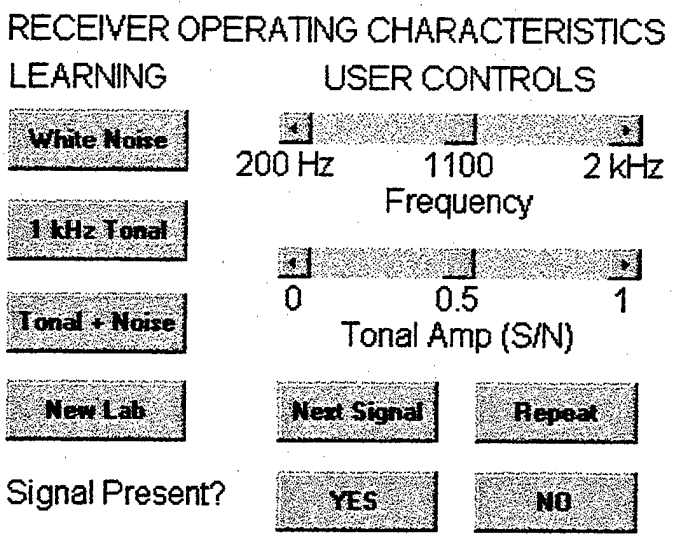

Figure 13.1 ROC User Interface.

When the program is started the pushbuttons "YES", "NO", "NEXT SIGNAL",

and "REPEAT" are not yet needed and cannot be seen by the user. Throughout the run when a pushbutton is not needed it is not visible to the user.

The slide bars "Frequency" and "Tonal Amp (S/N)" allow the user to adjust the frequency and amplitude of the tonal. The value can be changed by moving the slide bar or by depressing the button at either end of the slide bar. The frequency and amplitude are displayed below the slide bar. These sliders can be moved at anytime however, once a run is started the frequency and amplitude of the tonal cannot change until the run is complete.

For training purposes, "White Noise" and the "1 kHz Tonal" pushbuttons allow the user to listen to the noise or tonal separately. The "Tonal + Noise" pushbutton allows the user to listen to the two signals combined at equal amplitudes. These three buttons can be pressed at anytime during a run but the frequency of the tone will always be 1 kHz.

"New Lab" starts a run and makes the "Next Signal" button visible. Currently the program creates ten signals per run, to make the data more statistically significant the program variable "trials" in NEWLAB.M can be changed to any number the user wishes. All signals contain random noise and the program makes a random yes/no decision for each signal to determine whether it will contain a tonal. If the decision is yes the program adds a tonal at the frequency and amplitude specified by the user to the signal.

When the "Next Signal" is depressed the first signal will be played for four

seconds. The "YES", "NO", and "Repeat" buttons then show up. The user may elect to hear it again by depressing "Repeat". After the user depresses the "YES" or "NO" button the next signal can be heard by again depressing the "Next Signal" button. After the run is complete the buttons will again become invisible and the "New Lab" will be visible.

After completion the results of the run will be displayed. The number of signals, number with a tone present, number correctly identified, number with no tone that were incorrectly identified, a probability of detection, and a probability of false alarm will all be displayed.

ROC curves can now be generated using information from this lab. However, it is important to realize several things. First, it takes several points to accurately define any single curve. To do this it is best to keep the S/N ratio constant during several trials but change the criterion for answering "YES". The first time the criterion might be to answer yes if there is even the slightest possibility there is a signal. The next criterion might be to say yes only if you're positive there is a signal. This will give several points along the same curve. Other curves can be determined by changing the S/N ratio and using the same yes/no criteria.

C. ALGORITHMS FOR ROC LAB.

1. Algorithm roc.m

```
%function roc_lab
% This program is the first to be called for the ROC lab. It generates the user interface .
```

```
% It requires no user protection because the interface itself only allows certain inputs to
% be made.
clear
global trial
global signals
global trials
global resp
noise1=randn(1,((2^13)*4)-4);
freq1=500;
arg_step=(2*pi*freq1*4)/(((2^13)*4)-1);
sin_arg=0:arg_step:((2*pi*freq1*4)-arg_step);
sig1=sin(sin_arg);
sig2=sig1(1,1:(length(sig1)-3));
mix_sig=noise1+sig2;
roc_win=figure('units','normal','position',[.25 .25 .5 .5],...
        'name','Receiver Operator Characteristics Lab',...
        'color','white','numbertitle','off');
axis off
text('units','normalized','position',[.5 .95],...
        'horizontalalignment','center','string',...
```

```
        'RECEIVER OPERATING CHARACTERISTICS','color','black');


text('units','normalized','position',[.1 .85],...
        'horizontalalignment','center','string',...
        'LEARNING','color','black');
text('units','normalized','position',[.7 .85],...
        'horizontalalignment','center','string',...
        'USER CONTROLS','color','black');
only_noise=uicontrol('style','pushbutton','units',...
        'normalized','position',[.11 .65 .225 .1],...
        'backgroundcolor','magenta','foregroundcolor','black',...
        'horizontalalignment','center','string','White Noise',...
        'callback','sound(noise1);');
only_tone=uicontrol('style','pushbutton','units',...
        'normalized','position',[.11 .5 .225 .1],...
        'backgroundcolor','magenta','foregroundcolor','black',...
        'horizontalalignment','center','string','1 kHz Tonal',...
        'callback','sound(sig2);');
tone_and_noise=uicontrol('style','pushbutton','units',...
        'normalized','position',[.11 .35 .225 .1],...
```

```
        'backgroundcolor','magenta','foregroundcolor','black',...
        'horizontalalignment','center','string','Tonal + Noise',...
        'callback','sound(mix_sig);');
new_lab=uicontrol('style','pushbutton','units',...
        'normalized','position',[.11 0.2 .225 .1],...
        'backgroundcolor','magenta','foregroundcolor','black',...
        'horizontalalignment','center','string',...
        'New Lab','callback','newlab');
user_freq=uicontrol('style','slider','units','normalized',...
        'position',[.45 .7 .45 .05],'backgroundcolor','magenta',...
        'foregroundcolor','black','min',200,'max',2000,...
        'value',1100,'callback','freqdisp');
u_f=num2str(get(user_freq,'value'));
text('units','normalized','position',[.7 .675],...
        'horizontalalignment','center','string',u_f,'color','black');
text('units','normalized','position',[.5 .675],...
        'horizontalalignment','right','string',...
        '200 Hz','color','black');
text('units','normalized','position',[.9 .675],...
        'horizontalalignment','left','string',...
```

```
        '2 kHz','color','black');
text('units','normalized','position',[.7 .6],...
        'horizontalalignment','center','string',...
        'Frequency','color','black');
tonal_amp=uicontrol('style','slider','units','normalized',...
        'position',[.45 .475 .45 .05],'backgroundcolor','magenta',...
        'foregroundcolor','black','min',0,'max',1,...
        'value',0.5,'callback','ampdisp');
t_a=num2str(get(tonal_amp,'value'));
text('units','normalized','position',[.7 .4],...
        'horizontalalignment','center','string',t_a,'color','black');
text('units','normalized','position',[.45 .4],...
        'horizontalalignment','right','string',...
        '0','color','black');
text('units','normalized','position',[.95 .4],...
        'horizontalalignment','left','string',...
        '1','color','black');
text('units','normalized','position',[.7 .325],...
        'horizontalalignment','center','string',...
        'Tonal Amp (S/N)','color','black');
```

```
text('units','normalized','position',[0.15 0],...
        'horizontalalignment','center','string',...
        'Signal Present?','color','black');
sig_yes=uicontrol('style','pushbutton','units',...
        'normalized','position',[.45 0.05 .2 .1],...
        'backgroundcolor','magenta','foregroundcolor','black',...
        'horizontalalignment','center','string',...
        'YES','callback','resp=1;results;','visible','off');

sig_no=uicontrol('style','pushbutton','units',...
        'normalized','position',[.7 0.05 .2 .1],...
        'backgroundcolor','magenta','foregroundcolor','black',...
        'horizontalalignment','center','string',...
        'NO','callback','resp=2;results;','visible','off');
next_sig=uicontrol('style','pushbutton','units',...
        'normalized','position',[.45 0.2 .2 .1],...
        'backgroundcolor','magenta','foregroundcolor','black',...
        'horizontalalignment','center','string',...
        'Next Signal','callback','nextsig;',...
        'visible','off');
```

```
repeat_sig=uicontrol('style','pushbutton','units',...
        'normalized','position',[.7 0.2 .2 .1],...
        'backgroundcolor','magenta','foregroundcolor','black',...
        'horizontalalignment','center','string',...
        'Repeat','callback','sound(signals(trial,:));',...
        'visible','off');
```

2. Algorithm newlab.m

```
%function newlab
% This program generates the set of signals to be used during a run for the roc lab.
global trials
global signals
global trial
global tonal_coeff
% The following section sets defaults for variables used in program
trial=1;
frequency=get(user_freq,'value');                    % frequency of tonal
in Hz
noise_duration=4;                    % duration of sound in seconds
trials=10;
sig_mag=get(tonal_amp,'value');                    % relative magnitude of tonal
```

```
noise_length=(2^13)*noise_duration;          % this is total number of samples
                                             % that will be used, 2^13 per second
                                             % determined by sampling rate of
                                             % sound function which is 8192 Hz
noise=randn(trials,noise_length);            % Generates white noise signal
% The following section creates the tonal and determines which signals will have
% the tonal present.
a=(2*pi*user_freq*noise_duration)/(size(noise,2)-1);    % finds the step size for tonal
                                                        % generation
freq_count=(2*pi*user_freq*noise_duration);
sig=0:a:freq_count;          % generates all pts for argument of sin
tonal=2*sig_mag*sin(sig);
rand('seed',sum(100*clock));
tonal_coeff=round(rand(trials,1));
tone_matrix=(tonal_coeff*tonal);
signals=noise+tone_matrix;
set(next_sig,'visible','on');
set(new_lab,'visible','off');
refresh(roc_win);
```

3. Algorithm nextsig.m

```
%function nextsig
% This program produces the tone to begin each signal for the roc lab.
sound(signals(trial,:));
set(next_sig,'visible','off');
set(repeat_sig,'visible','on');
set(sig_yes,'visible','on');
set(sig_no,'visible','on');
refresh(roc_win);
```

4. Algorithm freqdisp.m

```
% This is a script file used to display the current frequency.
prev_f=u_f;
u_f=num2str(get(user_freq,'value'));
text('units','normalized','position',[.7 .675],...
        'horizontalalignment','center','string',prev_f,...
        'color','white');
text('units','normalized','position',[.7 .675],...
        'horizontalalignment','center','string',u_f,'color','black');
```

5. Algorithm ampdisp.m

```
% This is a script file used by thr roc lab to display the S/N ratio.
prev_ta=t_a;
```

```
t_a=num2str(get(tonal_amp,'value'));
text('units','normalized','position',[.7 .4],...
        'horizontalalignment','center','string',prev_ta,...
        'color','white');
text('units','normalized','position',[.7 .4],...
        'horizontalalignment','center','string',t_a,'color','black');
```

6. Algorithm results.m

```
%function results
% This function is used by the roc lab to compute results at the end of a run of
% frequencies.
global resp
global tonal_coeff
global trial
global trials
if trial==1
        detect=0;
        sig_pres=0;
        false_alarm=0;
        sig_notpres=0;
end
```

```
if resp==1                    % Answer is yes
        sig_there=1;
elseif resp==2                % Answer is no
        sig_there=0;
end
if tonal_coeff(trial)==1
        if sig_there==1
                detect=detect+1;
                sig_pres=sig_pres+1;
        elseif sig_there==0
                sig_pres=sig_pres+1;
        end
elseif tonal_coeff(trial)==0
        if sig_there==0
                sig_notpres=sig_notpres+1;
        elseif sig_there==1
                sig_notpres=sig_notpres+1;
                false_alarm=false_alarm+1;
        end
end
```

```
if trial==trials
        fprintf('%3.0f trials were run.\n',trials)
        fprintf('%3.0f trials had a signal present.\n',sig_pres)
        fprintf('You correctly identified %3.0f of these signals.\n',detect)
        fprintf('In addition %3.0f of the trials were incorrectly\n',false_alarm)
        fprintf(' identified as having a signal present.\n')
        fprintf('This results in a Pd of %1.2f \n',(detect/sig_pres))
        fprintf('And a Pfa of %1.2f \n',(false_alarm/sig_notpres))
        set(new_lab,'visible','on');
        set(sig_yes,'visible','off');
        set(sig_no,'visible','off');
        set(repeat_sig,'visible','off');
else
        set(next_sig,'visible','on');
        set(repeat_sig,'visible','off');
        trial=trial+1;
end
refresh(roc_win);
```

APPENDIX. MODIFIED POLAR PLOT PROGRAM LISTING

```
function hpol = polar(theta,rho,line_style)
%POLAR      Polar coordinate plot.
%      POLAR(THETA, RHO) makes a plot using polar coordinates of
%      the angle THETA, in radians, versus the radius RHO.
%      POLAR(THETA,RHO,S) uses the linestyle specified in string S.
%      See PLOT for a description of legal linestyles.
%
%      See also PLOT, LOGLOG, SEMILOGX, SEMILOGY.

%      Copyright (c) 1984-94 by The MathWorks, Inc.
% This program has been modified to be used with the program BKNARRAY.M
% by Kevin Melody.
global radius
global r3db

if nargin < 1
        error('Requires 2 or 3 input arguments.')
elseif nargin == 2
```

```
    if isstr(rho)
        line_style = rho;
        rho = theta;
        [mr,nr] = size(rho);
        if mr == 1
            theta = 1:nr;
        else
            th = (1:mr)';
            theta = th(:,ones(1,nr));
        end
    else
        line_style = 'auto';
    end
elseif nargin == 1
    line_style = 'auto';
    rho = theta;
    [mr,nr] = size(rho);
    if mr == 1
        theta = 1:nr;
    else
```

```
            th = (1:mr)';
            theta = th(:,ones(1,nr));
        end
end
if isstr(theta) | isstr(rho)
        error('Input arguments must be numeric.');
end
if any(size(theta) ~= size(rho))
        error('THETA and RHO must be the same size.');
end

% get hold state
cax = newplot;
next = lower(get(cax,'NextPlot'));
hold_state = ishold;

% get x-axis text color so grid is in same color
tc = get(cax,'xcolor');

% Hold on to current Text defaults, reset them to the
```

```
% Axes' font attributes so tick marks use them.
fAngle  = get(cax, 'DefaultTextFontAngle');
fName   = get(cax, 'DefaultTextFontName');
fSize   = get(cax, 'DefaultTextFontSize');
fWeight = get(cax, 'DefaultTextFontWeight');
set(cax, 'DefaultTextFontAngle',  get(cax, 'FontAngle'), ...
        'DefaultTextFontName',   get(cax, 'FontName'), ...
        'DefaultTextFontSize',   get(cax, 'FontSize'), ...
        'DefaultTextFontWeight', get(cax, 'FontWeight') )

% only do grids if hold is off
if ~hold_state

% make a radial grid
        hold on;
        hhh=plot([0 max(theta(:))],[0 max(abs(rho(:)))]);
        v = [get(cax,'xlim') get(cax,'ylim')];
        ticks = length(get(cax,'ytick'));
        delete(hhh);
        rmin=0;
```

```
        rmax=radius;


% define a circle
        th = 0:pi/50:2*pi;
        xunit = cos(th);
        yunit = sin(th);
% now really force points on x/y axes to lie on them exactly
   inds = [1:(length(th)-1)/4:length(th)];
   xunits(inds(2:2:4)) = zeros(2,1);
   yunits(inds(1:2:5)) = zeros(3,1);


        %rinc = (rmax-rmin)/rticks;
        for i=rmax
                plot(xunit*i,yunit*i,'-','color',tc,'linewidth',1);
                %text(1,i+rinc/20,[' ' num2str(i)],'verticalalignment','bottom' );
        end
        for i=r3db
                plot(xunit*i,yunit*i,':','color',tc,'linewidth',1);
                %text(1,i+rinc/20,[' ' num2str(i)],'verticalalignment','bottom' );
        end
```

```
% plot spokes
    th = (1:2)*2*pi/4;
    cst = cos(th); snt = sin(th);
    cs = [-cst; cst];
    sn = [-snt; snt];
    plot(rmax*cs,rmax*sn,'-','color',tc,'linewidth',1);

% annotate spokes in degrees
    rt = 1.2*rmax;
    for i = 1:max(size(th))
        text(rt*cst(i),rt*snt(i),int2str(i*90),'horizontalalignment','center' );
        if i == max(size(th))
            loc = int2str(0);
        else
            loc = int2str(180+i*90);
        end
        text(-rt*cst(i),-rt*snt(i),loc,'horizontalalignment','center' );
    end

% set viewto 2-D
```

```
        view(0,90);
% set axis limits
        axis(rmax*[-1 1 -1.1 1.1]);
end


% Reset defaults.
set(cax, 'DefaultTextFontAngle', fAngle , ...
        'DefaultTextFontName',   fName , ...
        'DefaultTextFontSize',   fSize, ...
        'DefaultTextFontWeight', fWeight );


% transform data to Cartesian coordinates.
xx = rho.*cos(theta);
yy = rho.*sin(theta);


% plot data on top of grid
if strcmp(line_style,'auto')
        q = plot(xx,yy);
else
        q = plot(xx,yy,line_style);
```

```
end
if nargout > 0
        hpol = q;
end
if ~hold_state
        axis('equal');axis('off');
end

% reset hold state
if ~hold_state, set(cax,'NextPlot',next); end
```

LIST OF REFERENCES

KFCS Kinsler L. E., Frey A. R., Coppens A. B., Sanders J. V.., "Fundamentals of Acoustics", Third Edition, John Wiley & Sons, Inc., Monterey, CA, 1980

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Road, Ste 0944
   Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, California 93943-5101

3. Undersea Warfare, Code 37 1
   Naval Postgraduate School
   Monterey, California 93943-5002

4. Dr. James V. Sanders, Code PH/Sd 2
   Department of Physics
   Naval Postgraduate School
   Monterey, California 93943-5002

5. Dr. Al Coppens 1
   P.O. Box 335
   Black Mountain, North Carolina 28711-0335

6. LT Kevin A. Melody 2
   Department Head Class 152
   SWOSCOLCOM
   446 Cushing Road
   Newport, Rhode Island 02841-1209