BME 8730
Fall 2019

Jensen's FIELD II program

FIELDII is the de facto standard in linear diffraction modeling in medical ultrasound research. It isn't especially efficient but it is very widely used and accepted. Thus, it makes publishing new work easier if it is based on a well-accepted standard than some one-off Matlab model that isn't well known (e.g. our own homemade code we used previously)

Go through the FIELDII download. Take care to set up your directories.

Main page:
http://field-ii.dk/

Download (match to your OS and to your version of Matlab – might not be this version)
http://field-ii.dk/?./downloading_8_20.html

Papers:
http://field-ii.dk/?papers.html

Key paper – review this paper very closely. Cite this one at a minimum ALWAYS when using FIELDII in publishable work
http://field-ii.dk/documents/jaj_nbs_uffc_1992.pdf

User Guide – review this document very closely. It comprehensively describes the full capabilities of the code – extends to modeling fields, speckle modeling, effects of finite sampling rate (i.e. quantization), modeling blood flow situations
http://field-ii.dk/?users_guide.html

Using calc_h, xdc_piston, xdc_rectangle (or xdc_linear_array – it doesn't matter too much but the former is more appropriate)
(see the relevant examples in the user guide), extend as per the previous homework and calculate: velocity potential impulse response (i.e. h from calc_h), pressure impulse response and response when convolved with a sinusoid (I suggest a 10 cycle sinusoid and that you take the amplitude of the envelope mid pulse duration to get the required CW amplitude to go into the field values)

Replicate the following figures:

Lockwood
Figure 3, 6, 7, 8 and 13 (use 10 to 16 sub-elements in each direction by default) (10,10,10,10,10)

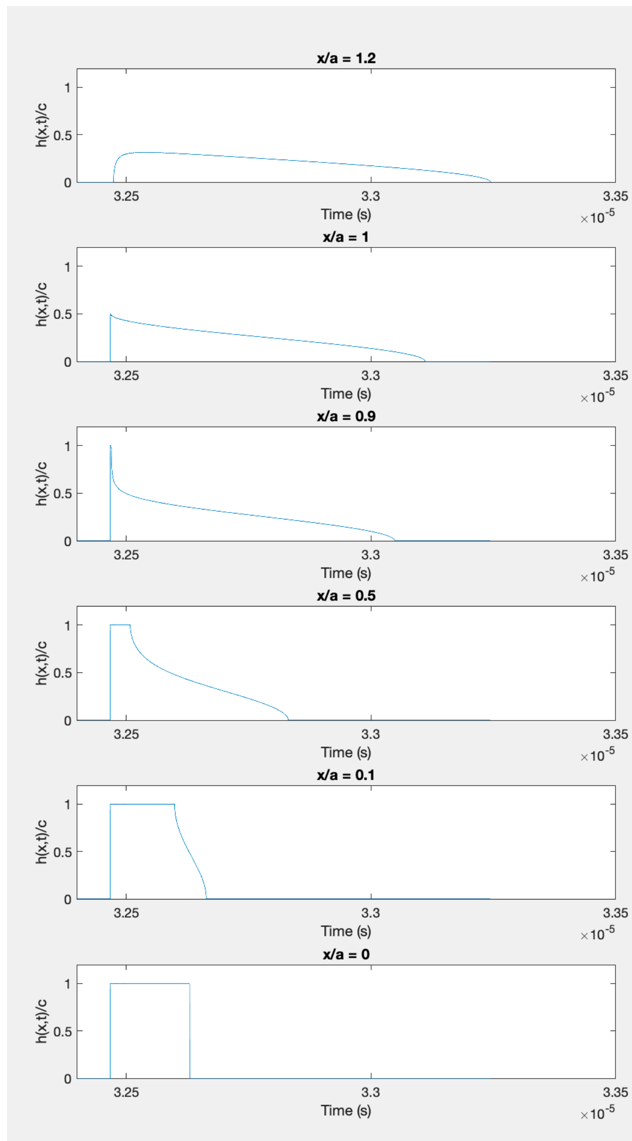
For Figures 3 d and f and 6 d show a result using a single sub-element and one where the overall width dimension is divided into 16 sub-elements (the code allows you to select the number of sub-elements in each of width and height in rectangles and the dimension of sub-elements for circular apertures)

For the CW beamplots, you should consider convolving with a long sinusoidal pulse and detecting the peak to peak amplitude near the center of the pulse (to avoid transient effects near the beginning and end) or use an FFT of the impulse response and extract the correct Fourier component. (If one doesn't work, try the other.) Notice that you need to define a and λ to make your code work. It isn't critical what actual values you use so long as they are consistent and produce the right results.

Replicate the previous array diffraction homework (using our "homemade" array code) using FIELDII for questions 2-6 of that homework. Use xdc_linear_array – or other approach of your choice – note the examples near the end of the user guide. You can change the sub-element size to suit your needs – i.e. save time. I suggest you vary the sub-element and verify the presence or absence of impact on final result.

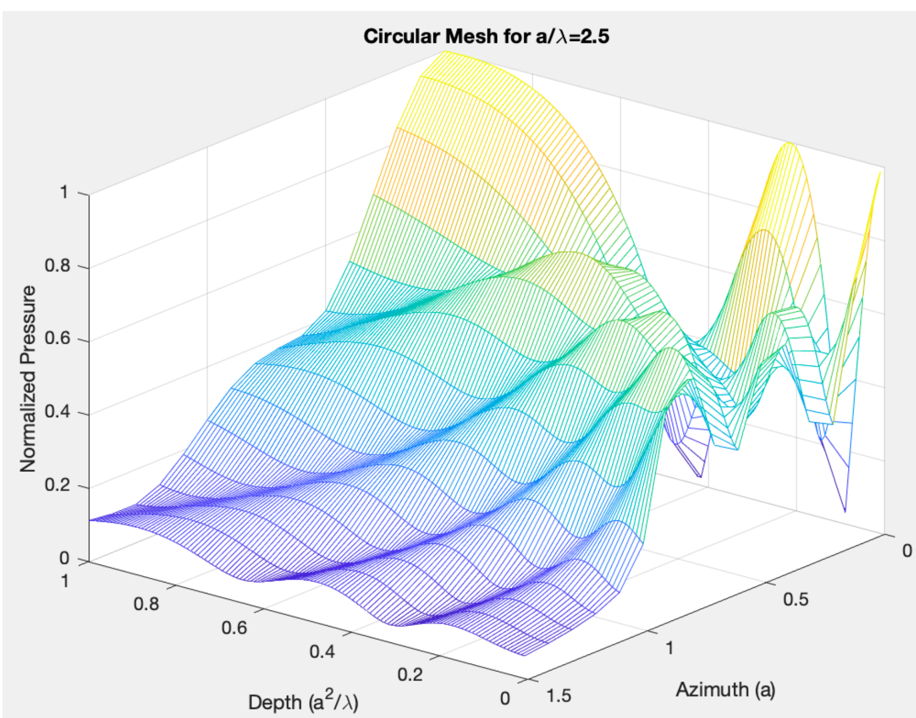
Please try to do as much of the coding yourself but work together to debug (and acknowledge) Submit, paper or PDF/word doc, figures and code – at least the core code representative for your solutions to each of one Lockwood figure replication and array diffraction replication.

Recreation of figure 3, code on next page

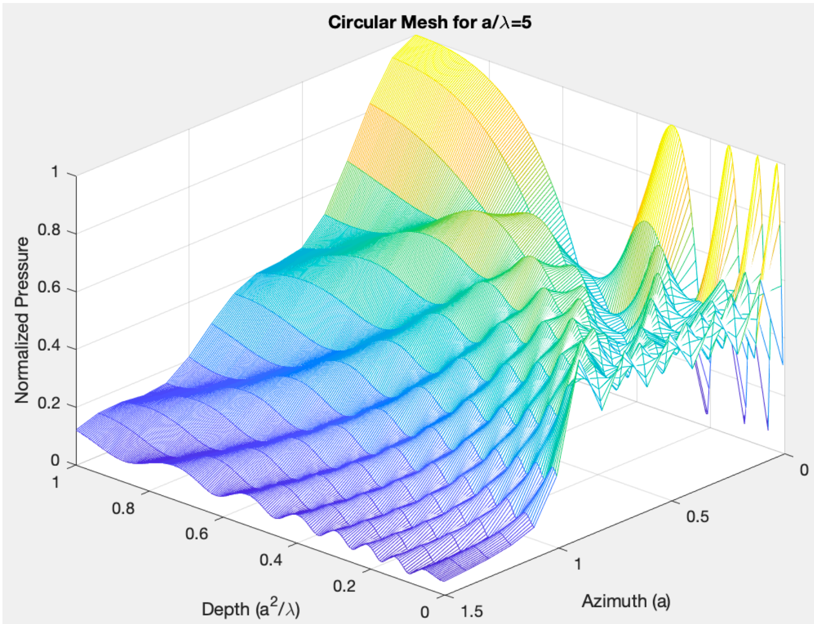


```
%% Field call
path(path,'/Users/samuelsklar/Documents/Classwork/Ultrasound/Field_II_ver_3_24_mac');
field_init(-1);
%% Variables
a=5e-3;                    % Half x width
x=[1.2,1,0.9,0.5,0.1,0].*5e-3;    % X points
y=zeros(1,length(x));     % Y points
z=(50).*1e-3;              % Z point
z=z*ones(1,length(x));
%% Field Parameters
frq=1e11;
set_sampling(frq);
center=[0,0,0];
focus=[0,0,50e-3];
tmax=4e-5;
c=1540;
%% Field call
Th = xdc_piston (5e-3, 5e-6);  % Transducer definition
[h,t]=calc_h (Th,[x',y',z']); % Transfer function calc
h=h./max(max(h));
%% Add padding on front end
hfront=zeros(ceil(((t-(3e-5))*frq)),6);
tf=(3e-5:1/frq:t+length(h)/frq);
h=[hfront;h];
%% plot
figure
for i=1:length(x)
    subplot(length(x),1,i)
    plot(tf,h(:,i));
    xlabel('Time (s)')
    ylabel('h(x,t)/c')
    axis([3.24e-5 3.35e-5 0 1.2])
    title(['x/a = ',num2str(x(i)/5e-3)])
end
```

Recreation of figure 7, code same as for 8 but with double lam variable

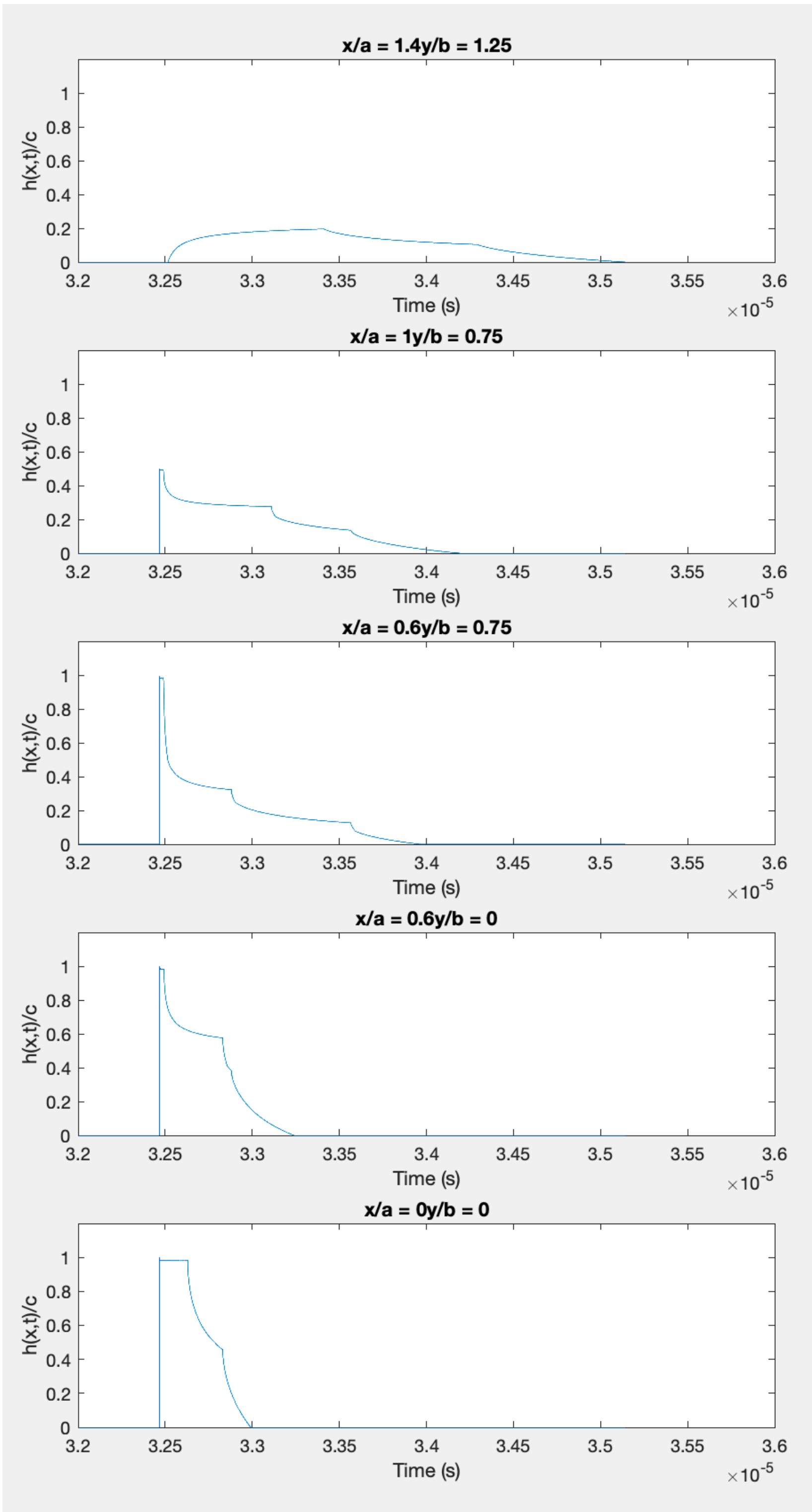


Recreation of figure 8



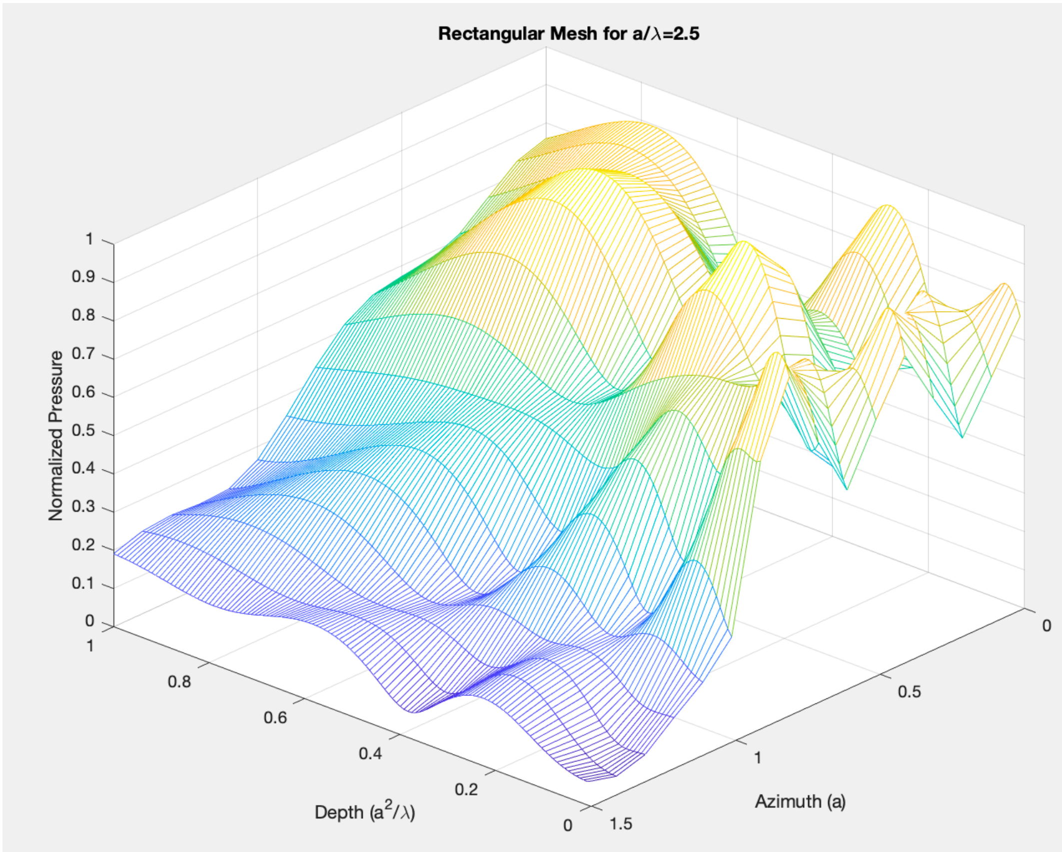

```matlab
%% Field call
path(path,'/Users/samuelsklar/Documents/Classwork/Ultrasound/Field_II_ver_3_24_mac');
field_init(-1);
%% Variables
a=5e-3;                     % Half x width
lam=1e-3;                   % Wavelength
zend=((a^2)./lam)*1e3;            % End Z point
z=(.1:.1:zend).*1e-3;       % Z field
y=0;                        % Y field
x=(0:.1:1.5).*5e-3;         % X field
%% Field Parameters
frq=1e10;
set_sampling(frq);
center=[0,0,0];
focus=[0,0,12.5e-3];
tmax=4e-5;
c=1540;
%% Field call and pressure calc
Th = xdc_piston (5e-3, 5e-6);  % Transducer definition

p=zeros(length(x),length(z));             % Pressure array initialization
for i=1:length(z)
    i
    for n= 1:length(x)
        [h,t]=calc_h (Th,[x(n),y,z(i)]); % Transfer function calc
        t=[t:1/frq:tmax];                % Define times for sin wave
        wav=sin(2*pi*t*c./lam);          % creation of sin wave
        hwc=conv(h,wav);                 % convolution
        hwc=diff(hwc);                   % differentiation
        prS=floor((length(hwc)./2)-(length(hwc)./5));    % Lower bounding for amplitude check
        prE=ceil((length(hwc)./2)+(length(hwc)./5));     % Higher bounding for amplitude check
        p(n,i)=max(hwc(prS:prE))-min(hwc(prS:prE));          % Calculate pressure
    end
end
p=p./max(max(p)); % Pressure Normalization
%% plot
figure
    mesh(z/((a^2)/lam),x/(5e-3),p);
    xlabel('Depth (a^2/\lambda)')
    ylabel('Azimuth (a)')
    zlabel('Normalized Pressure')
    title('Circular Mesh for a/\lambda=5')
```

Recreation of figure 6
Code on next page



```matlab
%% Field call
path(path,'/Users/samuelsklar/Documents/Classwork/Ultrasound/Field_II_ver_3_24_mac');
field_init(-1);
%% Variables
a=5e-3;                 % Half x width
b=7.5e-3;               % Half Y width
y=[1.25,0.75,0.75,0,0].*7.5e-3;          % Y points
x=[1.4,1,0.6,0.6,0].*5e-3;               % X points
z=(50).*1e-3;           % Z point
z=z*ones(1,length(x));
pitch=.02e-3;           % Element sizes
xpts=-a:pitch:a;        % X coordinates of element corners
ypts=-b:pitch:b;        % Y coordinates of element corners
noels=(length(xpts)-1)*(length(ypts)-1); % Number of elements
%% Rectangles input definition
rect=zeros(noels,19);
rect(:,1)=1;
rect(:,14)=1;
rect(:,15)=pitch;
rect(:,16)=pitch;
count=0;
for j=1:length(ypts)-1
    for k=1:length(xpts)-1
        count=count+1;
        rect(count,2:3)=[xpts(k),ypts(j)];
        rect(count,5:6)=[xpts(k+1),ypts(j)];
        rect(count,8:9)=[xpts(k+1),ypts(j+1)];
        rect(count,11:12)=[xpts(k),ypts(j+1)];
        rect(count,17:18)=[mean(xpts(k:k+1)),mean(ypts(j:j+1))];
    end
end
%% Field Parameters
frq=1e11;
set_sampling(frq);
center=[0,0,0];
focus=[0,0,50e-3];
tmax=4e-5;
c=1540;
%% Field call
Th = xdc_rectangles (rect, center, focus);  % Transducer definition
[h,t]=calc_h (Th,[x',y',z']); % Transfer function calc
h=h./max(max(h));
%% Add padding on front end
hfront=zeros(ceil(((t-(3e-5))*frq)),5);
tf=(3e-5:1/frq:t+length(h)/frq);
h=[hfront;h];
%% plot
figure
for i=1:length(x)
    subplot(length(x),1,i)
    plot(tf,h(:,i));
    xlabel('Time (s)')
    ylabel('h(x,t)/c')
    axis([3.2e-5 3.6e-5 0 1.2])
    title(['x/a = ',num2str(x(i)/5e-3),'y/b = ',num2str(y(i)/7.5e-3)])
end
```

Recreation of figure 13
Code on next page



Overall these meshes for both rectangular and circular transducer are a bit choppier those from the home made solution but the code is also much simpler and all of the important morphologies can be made out.

```matlab
%% Field call
path(path,'/Users/samuelsklar/Documents/Classwork/Ultrasound/Field_II_ver_3_24_mac');
field_init(-1);
%% Variables
a=5e-3;                    % Half x width
b=7.5e-3;                  % Half Y width
lam=2e-3;                  % Wavelength
z=(.1:.1:12.5).*1e-3;      % Z field
y=(0).*7.5e-3;             % Y field
x=(0:.1:1.5).*5e-3;        % X field
pitch=.04e-3;              % Element sizes
xpts=-a:pitch:a;           % X coordinates of element corners
ypts=-b:pitch:b;           % Y coordinates of element corners
noels=(length(xpts)-1)*(length(ypts)-1); % Number of elements
%% Rectangles input definition
rect=zeros(noels,19);
rect(:,1)=1;
rect(:,14)=1;
rect(:,15)=pitch;
rect(:,16)=pitch;
count=0;
for j=1:length(ypts)-1
    for k=1:length(xpts)-1
        count=count+1;
        rect(count,2:3)=[xpts(k),ypts(j)];
        rect(count,5:6)=[xpts(k+1),ypts(j)];
        rect(count,8:9)=[xpts(k+1),ypts(j+1)];
        rect(count,11:12)=[xpts(k),ypts(j+1)];
        rect(count,17:18)=[mean(xpts(k:k+1)),mean(ypts(j:j+1))];
    end
end
%% Field Parameters
frq=1e10;
set_sampling(frq);
center=[0,0,0];
focus=[0,0,12.5e-3];
tmax=4e-5;
c=1540;
%% Field call and pressure calc
Th = xdc_rectangles (rect, center, focus);  % Transducer definition

p=zeros(length(x),length(z));             % Pressure array initialization
for i=1:length(z)
    i
    for n= 1:length(x)
        [h,t]=calc_h (Th,[x(n),y,z(i)]); % Transfer function calc
        t=[t:1/frq:tmax];                % Define times for sin wave
        wav=sin(2*pi*t*c./lam);          % creation of sin wave
        hwc=conv(h,wav);                 % convolution
        hwc=diff(hwc);                   % differentiation
        prS=floor((length(hwc)./2)-(length(hwc)./5));   % Lower bounding for amplitude check
        prE=ceil((length(hwc)./2)+(length(hwc)./5));    % Higher bounding for amplitude check
        p(n,i)=max(hwc(prS:prE))-min(hwc(prS:prE));          % Calculate pressure
    end
end
p=p./max(max(p)); % Pressure Normalization
%% plot
figure
    mesh(z/((a^2)/lam),x/(5e-3),p);
    xlabel('Depth (a^2/\lambda)')
    ylabel('Azimuth (a)')
    zlabel('Normalized Pressure')
    title('Rectangular Mesh for a/\lambda=2.5')
```