Field II to FOCUS Guide

Michigan State University

November 18, 2013

Contents

Introduction

This guide is intended to assist a user who has experience writing code with Field II and wants to begin writing code with FOCUS. If you are only looking at this guide to get a look at FOCUS code, skip to page 10.

The guide is split into four main sections. The first section walks through writing a typical script that calculates continuous-wave pressure using Field II. The second section breaks the script apart and shows how the Field II code would be written using FOCUS. The third section then puts this code back together and walks through a detailed script that calculates continuous-wave pressure using FOCUS. The last section presents basic functions that are used in FOCUS and will help you to begin writing your own code in FOCUS. More information on the functions that are used by FOCUS can be found in the FOCUS Function List.

Field II Continuous Wave Script

Let's begin by walking through a script that calculates the pressure field generated by a small planar array of rectangular transducers under continuous wave excitation in Field II.

```
% Field II CW Example
% Start Field II and set initial parameters

field_init(0);
f0=1e6;
Fs=100e6;
c = 1500;                   %  Speed of sound [m/s]
lambda = 1500/f0;
focus = [0 0 25*lambda];    %  Fixed focal point [m]
N_elements = 20;            %  Number of physical elements
width = 7e-4;               %  Width of element
height = 3e-3;              %  Height of element [m]
kerf = 5e-4;                %  Kerf [m]
elefocus = 1;               %  Whether to use elevation focus
Rfocus = 25e-3;             %  Elevation focus [m]

% Define the transducer

Th = xdc_linear_array (N_elements, width, height, kerf, 20, 1, focus);
figure(1);
show_xdc(Th);

% This sets up our trasducer array with 20 0.7 mm x 3 mm elements
% spaced 0.5 mm edge-to-edge. Now we need to define the simulation
% grid for field calculations. y = 0 for XZ plane.

spacing= width + kerf;
elements_x = 20;

xmin = -(width + spacing) * (elements_x/2+1);
xmax = (width + spacing) * (elements_x/2+1);
ymin = 0;
ymax = 0;
zmin = 1e-3;
zmax = 50 * lambda;

xpoints = 400;
ypoints = 1;
zpoints = 300;

dx = (xmax-xmin)/xpoints;
dy = (ymax-ymin)/ypoints;
dz = (zmax-zmin)/zpoints;

x = xmin:dx:xmax;
y = ymin:dy:ymax;
z = zmin:dz:zmax;

% This sets up our coordinate grid to cover the full width of the
```

```
% transducer array in the x direction and to measure the pressure field to
% 50 wavelengths in the z direction. Now we need to set the sampling
% frequency, the medium sound speed and the excitation signal of the
% apertures.

set_sampling(Fs);
set_field('c',c);
Ts = 1/Fs; % Sampling period
T = 50e-6;
te = 0:Ts:T; % Time vector
excitation = sin(2*pi*f0*te+pi); % Excitation signal
xdc_excitation(Th, excitation);

% The next step is to initialize the vectors, run the calc_hp function
% and display the resulting pressure field.

tic;
point = [0 0 0];
t1 = zeros(length(x),length(z)); % Start time for Thrture 1 pressure signal
P1n = t1; % Norm of pressure from Thrture 1
for n2 = 1:length(x)
clc;[ n2 length(x)]
    for n3 = 1:length(z)
        point(1)=x(n2);
        point(2)=0;
        point(3)=z(n3);
        [p1,t1(n2,n3)] = calc_hp(Th,point);
        P1n(n2,n3) = norm(p1);
    end
end

field_end;

figure(2);
h = pcolor(x*100,z*100,rot90(squeeze(abs(P1n)),3));
set(h,'edgecolor','none');
title('Pressure Field at y = 0 cm');
xlabel('x (cm)');
ylabel('z (cm)');
ylim([0 zmax*100])
toc;
```

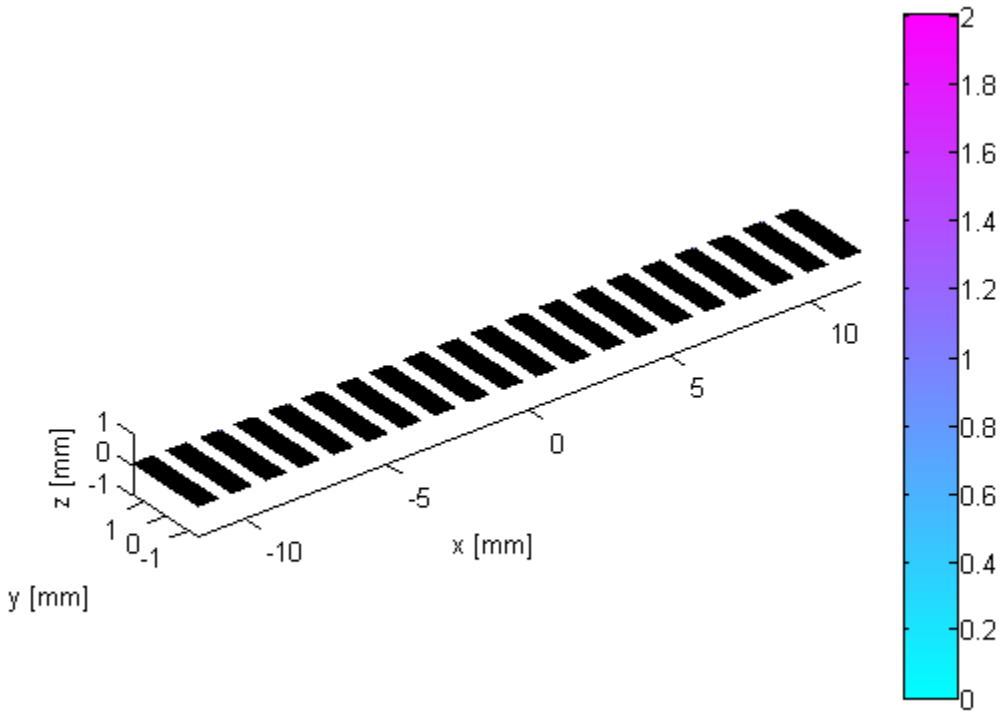

Figure 1: The transducer array used in this simulation.

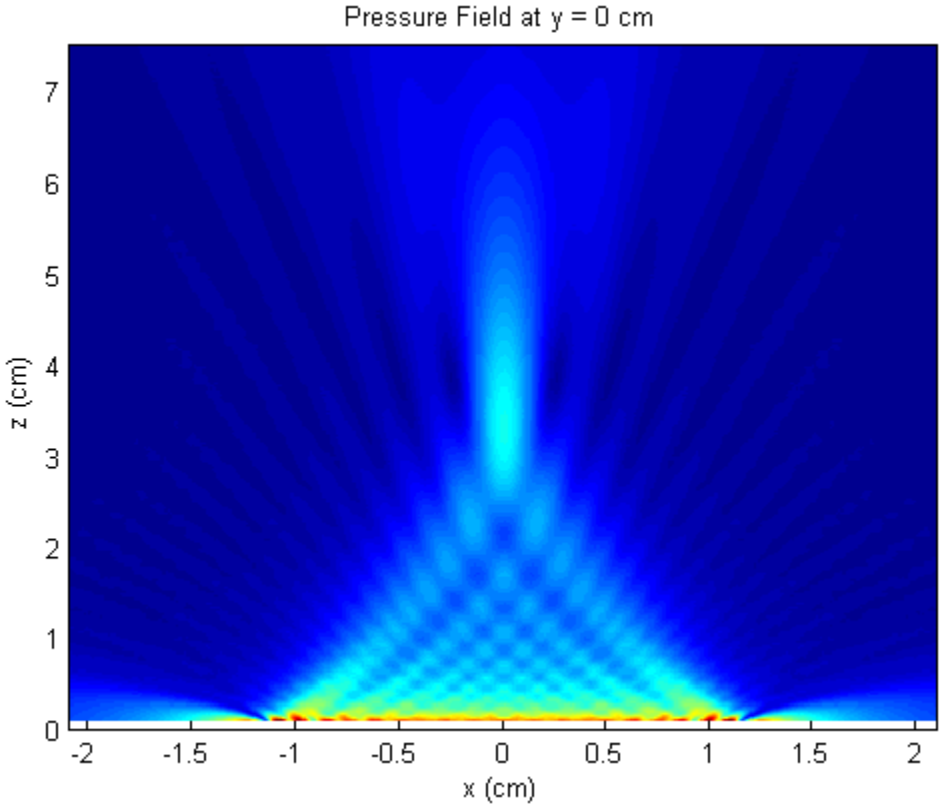

Figure 2: This example takes about 289 seconds to run on a 2.60GHz AMD Phenom 9950 Quad-Core Processor.

Translating Field II to FOCUS

Set up a Transducer

Field II

The basic procedure for setting up a transducer in Field II is to set the initial parameters and then define the transducer by calling a function to create transducers and arrays. We used the following code in the example above:

```
% Setting up a transducer in Field II
% Initial parameters

focus = [0 0 25*lambda];   %  Fixed focal point [m]
N_elements = 20;           %  Number of physical elements
width = 7e-4;              %  Width of element
height = 3e-3;             %  Height of element [m]
kerf = 5e-4;               %  Kerf [m]
elefocus = 1;              %  Whether to use elevation focus
Rfocus = 25e-3;            %  Elevation focus [m]

% Define the transducer

Th = xdc_linear_array (N_elements, width, height, kerf, 20, 1, focus);
show_xdc(Th);
```

FOCUS

The same basic procedure is used in FOCUS.

```
% Setting up a transducer in FOCUS
% Initial parameters

hw = 3.5e-4;              % Half width of the element
hh = 1.5e-3;              % Half height of the element
elements_x = 20;          % Number of elements in the x direction
elements_y = 1;           % Number of elements in the y direction
kerf = 5e-4;              % Kerf [m]
spacing = 2*hw + kerf;    % Center to center spacing

%Define the transducer

transducer_array = create_rect_planar(elements_x, elements_y, hw, hh, ...
spacing, spacing);
draw_array(transducer_array);  %Visualize in a 3-D space
```

Set General Parameters

Field II

In Field II, several parameters need to be set using the `set_field` command such as the speed of sound. Field II also requires that you set the excitation signal of the aperatures.

FOCUS

In FOCUS, to set the medium you simply need to call the `define_media` function. More information about `define_media` can be found in the FOCUS Function List. Unlike Field II, FOCUS does not require you to set the excitation of the transducer array when calculating the continuous-wave pressure.

Create a Coordinate Grid

Field II

Next we need to define a simulation grid for Field II calculation. This grid is then used in the computation commands.

FOCUS

In FOCUS a function called `set_coordinate_grid()` can be used to create a coordinate grid data structure used by FOCUS for various calculations. More information on this function can be found in the FOCUS Function List.

Computation Commands

Field II

The following code would be used in Field II to find the pressure of a continuous wave using the `calc_hp` function.

```
% Find the pressure field in Field II.

point = [0 0 0];
t1 = zeros(length(x),length(z)); % Start time for Thrture 1 pressure signal
P1n = t1; % Norm of pressure from Thrture 1
for n2 = 1:length(x)
clc;[ n2 length(x)]
    for n3 = 1:length(z)
        point(1)=x(n2);
        point(2)=0;
        point(3)=z(n3);
        [p1,t1(n2,n3)] = calc_hp(Th,point);
        P1n(n2,n3) = norm(p1);
    end
end
```

FOCUS

In FOCUS, the process is shortened by simply calling the `fnm_call` function. This function is the gateway between the Matlab and C++ binary. It does minimal error checking to ensure enough arguments are being passed.

```
% Find the pressure field in FOCUS

p_cw=fnm_call(transducer_array, coord_grid, lossless, ndiv, f0, 0);
```

FOCUS Continuous Wave Script

Now let's walk through writing a script that uses the Fast Nearfield Method to calculate the pressure field generated by a small planar array of rectangular transducers under continuous wave excitation.

```
% FNM CW Example
% General parameters in meters.

hw = 3.5e-4;              % Half width of the element [m]
hh = 1.5e-3;              % Half height of the element [m]
elements_x = 20;          % Number of elements in the x direction
elements_y = 1;           % Number of elements in the y direction
kerf = 5e-4;              % Kerf [m]
spacing = 2*hw + kerf;    % Center to center spacing [m]

% The following function creates a planar array of rectangular transducers.
transducer_array = create_rect_planar_array(elements_x, elements_y, hw, hh, ...
spacing, spacing);

% Now we can create a 3D representation of an array of FOCUS transducers.
draw_array(transducer_array);

% This sets up our transducer array with 20 0.7 mm x 3 mm elements spaced
% 0.5 mm edge-to-edge. Notice the spacing variable is used twice. This is
% because FOCUS lets you specify both x and y spacing for the transducer
% array. In this example, we only have a one-dimensional array, so the
% y-spacing doesn't matter. Next, we need to set up our coordinate grid.

define_media();      % Generates pre-set media usable for an calculation
f0 = 1e6;            % Center frequency of the array in Hz
lambda = (lossless.soundspeed / f0);

% Now we need to enter the arguments that will be used by set_coordinate_grid.
% The min and max arguments sets the boundaries of the calculation grid.
% The delta argument is the distance from each points to its nearest neighbor
% and most be in the form of a matrix.

% The lowest planes.
xmin = -(hw*2 + spacing) * (elements_x/2+1);
ymin = 0;
zmin = 0;

%The highest planes.
xmax = (hw*2 + spacing) * (elements_x/2+1);
ymax = 0;
zmax = 50*lambda;

% Coordinates of the focus.
focus_x = 0;
focus_y = 0;
focus_z = 25 * lambda;

% Number of points on specific plane.
xpoints = 400;
```

```
ypoints = 1;
zpoints = 300;

% Spacing between points on the specific plane.
dx = (xmax-xmin)/xpoints;
dy = (ymax-ymin)/ypoints;
dz = (zmax-zmin)/zpoints;

% Distance from each point to nearest neighbor on specific plane.
x = xmin:dx:xmax;
y = ymin:dy:ymax;
z = zmin:dz:zmax;

% Next is the delta argument which is the difference between two data points
% in a coordinate direction. Delta shoud always be a matrix of the form
% [ dx dy dz ].
delta = [dx dy dz];

% Now we can set our coordinate grid to cover the full width of the
% transducer array in the x direction and to measure the pressure
% field to 50 wavelengths in the z direction. After that we need to focus the
% transducer array.
coord_grid = set_coordinate_grid(delta, xmin, xmax, ymin, ymax, zmin, zmax);
disp(['Focusing array at (', num2str(focus_x), ', ', num2str(focus_y), ...
', ', num2str(focus_z), ')']);

% The next function will produce an array of complex_weight shifts to focus a transducer
% array at a given point. It uses the array we created and the focus
% coordinates we declared above. Lossless is the type of media we are
% using from the define_media function, f0 is the center frequency, and 200 is
% the number of abscissas used for calculations. The default value is 200. The
% final step is to run the FNM function and display the resulting pressure field.
transducer_array = find_single_focus_phase(transducer_array, focus_x, ...
focus_y, focus_z, lossless, f0, 200);

ndiv=6;  % The number of integral points to use.
tic();

% Finally we calculate the pressure field using the transducer and coordinate
% grid we created above. Again, lossless is the type of media we are using from the
% define_media function. Lastly, f0 is the center frequency and 0
% suppresses the dflag.
disp('Calculating pressure field...');
p_cw=fnm_call(transducer_array, coord_grid, lossless, ndiv, f0, 0);
disp(['Simulation complete in ', num2str(toc()), ' seconds.'])

% Now we make a figure to display the pressure field.
figure();
h = pcolor(x*100,z*100,rot90(squeeze(abs(p_cw)),3));
set(h,'edgecolor','none');
title('Pressure Field at y = 0 cm');
xlabel('x (cm)');
ylabel('z (cm)');
```

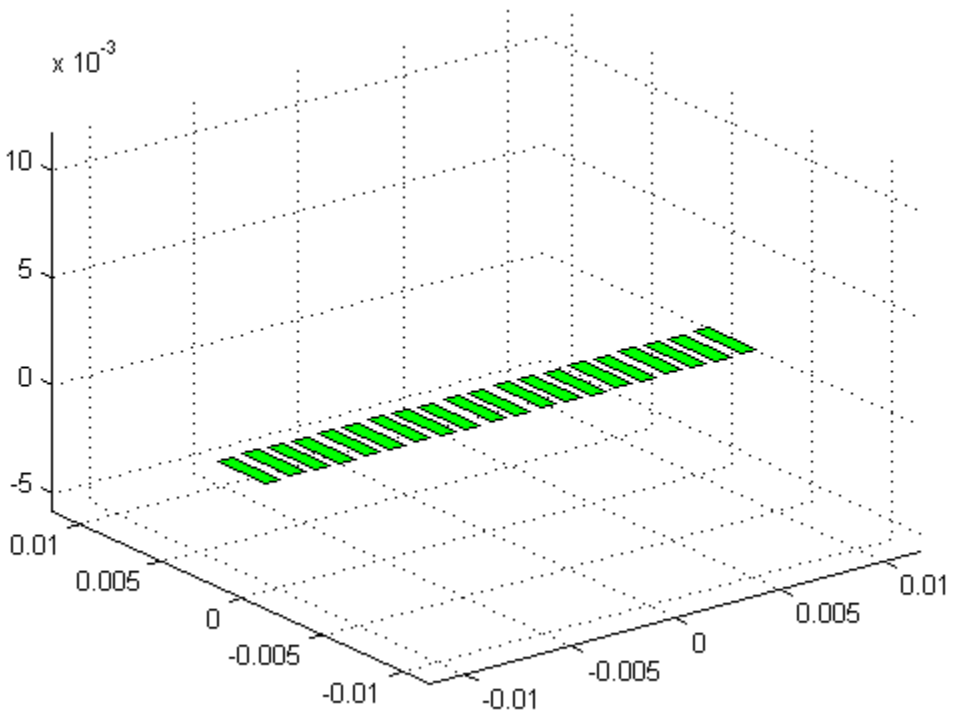

Figure 3: The transducer array used in this simulation.

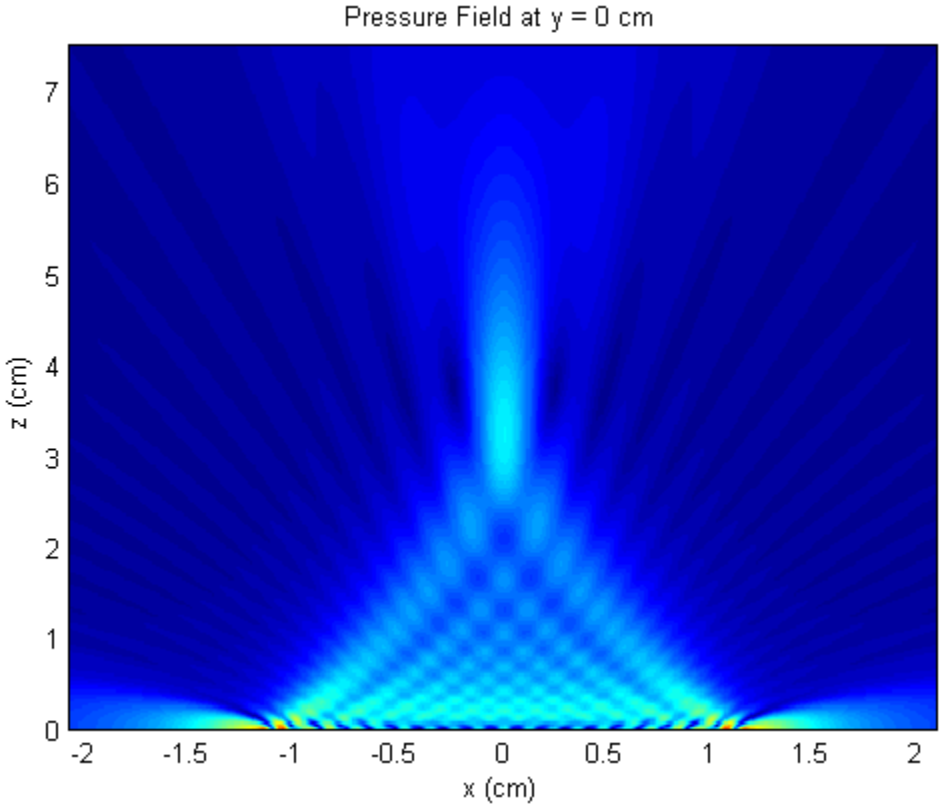

Figure 4: The above example takes about 29 seconds to run on a 2.60GHz AMD Phenom 9950 Quad-Core Processor.

Description of MATLAB Procedures

Array Commands

For this program, a few simplifying assumptions have been made about transducers. The first is the infinite planar baffle assumption. This assumption is simply that sound waves cannot reflect from sources behind transducers, and that all desired results are in front of all transducers. The second assumption is that the transducers have a uniform surface. This assumption holds true for smaller transducers but may be violated if larger transducers are used. The final assumption is an infinite coordinate grid.

The following functions are used to create transducers and arrays:

- get_circ
- get_rect
- create_rect_csa
- create_circ_csa
- create_rect_planar_array
- create_circ_planar_array
- create_rect_ssa
- create_circ_ssa

A function has also been added to visualize the transducer in a 3-d space.

- draw_array

Configure the Coordinate Grid

The coordinate grid structure contains all the information about the dimensions of the solution. The FNM equation operates on a point by point basis, so there are two ways to present the dimensions. The most common way for therapy is to provide the dimensions of a uniform grid. The program will automatically translate the dimensions into a list of points to be calculated, arranged in such a way that the final answer can be accessed as a three-dimensional MATLAB matrix, e.g. pressure(x,y,z).

The coordinate grid is structure with 9 fields. They are:

- xmin
- xmax
- ymin
- ymax
- zmin
- zmax
- delta
- vector_flag: -1 if using min/max notataion, 1 if using coordinate list.

- vector: ignored if using min/max notation, used if calculating based on a set of points.
- tmin: optional, used by the imaging portion of the code.
- tmax: optional, used by the imaging portion of the code.
- deltat: optional, used by the imaging portion of the code.

A coordinate grid structure can be created using the `set_coordinate_grid` function.

Computation Commands

FNM
There are several ways to call the program. There is an interactive mode for people unfamiliar with the program and there is also a scripting mode. There is also a direct call to the MEX file, but this usage is discouraged. The easiest way to run an FNM calculation is to use the fnm_run command. If you don't know what settings to use, run

```
Pressure=fnm_run('transducer',your_transducer);
```

This will ask you for all the necessary information and assumes the following: tolerance=1e-10 and the medium is water. If you need to put in more options, you can use::

```
Pressure=fnm_run('transducer',your_transducer,'arg_name',arg_val,...);
```

Just replace arg_name with the arguments that you want to specify. If you are looking for a script friendly function, use fnm_call. This function does minimal checking to prevent Matlab from crashing and does not prompt the user for input.

ASA
Once the 2d grid has been created, all you need to do is run the function. This can be done interactively using the asa_run function. For pressure fields generated by the FNM code, the proper call is:

```
Pressure = asa_run(source,z,medium,nfft,delta,type,f0)
```