FOCUS Quick Start Guide

Michigan State University

November 18, 2013

Introduction

FOCUS is a continuous and transient wave ultrasound simulator designed to simulate ultrasound waves in a variety of media. FOCUS takes advantage of two primary simulation methods: the Fast Nearfield Method (FNM) and the Angular Spectrum Approach (ASA). The FNM is used for calculating pressure fields near the transducer face, while the Angular Spectrum Approach is used to quickly calculate the farfield pressure.

FOCUS is distributed as a set of MATLAB M-files and MEX files. The M-files are written in standard MATLAB; you can take a look at their contents to get a better idea of how FOCUS works. The MEX files, on the other hand, are compiled from C++ code and are much faster than the equivalent MATLAB code. This is how all of the core FOCUS routines (e.g. fnm_cw) are distributed.

This document will explain how to install FOCUS on your computer and how to configure MATLAB to communicate with FOCUS. The document also contains three example MATLAB M-files that you can copy and paste into MATLAB to make sure that FOCUS is working correctly and to get started writing your own simulations using FOCUS.

For a more comprehensive overview of the functions provided by FOCUS and their use, take a look at our online documentation at http://www.egr.msu.edu/~fultras-web/documentation/functions/ or view the document "FOCUS Function List," distributed with FOCUS or available from our downloads page at http://www.egr.msu.edu/~fultras-web/download.php.

Installation

System Requirements

Any computer powerful enough to run MATLAB is powerful enough to run FOCUS. However, some of the more advanced features FOCUS provides (e.g. SSE instructions) have certain hardware requirements. These requirements are documented in the FOCUS Function List PDF file.
There are a number of issues with MATLAB R2008b on Unix systems that can cause problems with FOCUS so we must recommend that unix users avoid this version of MATLAB.

Windows Users

Make sure that you have the Microsoft Visual C++ 2008 Redistributable Package installed as the FOCUS MEX files have been compiled using Visual Studio 2008. There are download links for both the 32- and 64-bit versions of this software on the FOCUS downloads page.

Installing FOCUS

1. Decompress the .zip file you downloaded from the FOCUS website. If you're reading this, you've already gotten this far.
2. Move the files to the folder of your choice.
3. Add the folder containing the FOCUS files to your MATLAB path.
4. Make sure to remove any old versions of FOCUS from your MATLAB path before continuing.

Adding files to the MATLAB path

To add files to the matlab path, go to File -> Set Path or use the `addpath()` command from the MATLAB console.
That's it! Now you're ready to start using FOCUS. Before writing your first FOCUS script, we suggest that you take a look at some of the prepared examples in the Examples directory.

Using FOCUS

This section is to help you get started writing ultrasound simulations with FOCUS. The following pages contain several MATLAB scripts that can be pasted into MATLAB and run to generate the output shown at the end of each example. We start with a simulation of continuous wave excitation of a planar array and work our way towards using the Angular Spectrum Approach to simulate the farfield pressure generated by a cylindrical section array of rectangular transducers.

You must add the FOCUS folder to your MATLAB path before running any of these examples. Additional example MATLAB scripts can be found in the FOCUS Examples directory.

Contents

Description of FOCUS Parameters

FOCUS takes advantage of a number of unique data structures to expedite computation and simplify the process of writing simulations. Understanding these structures and what values to use is critical for successful use of the software. FOCUS functions also take several arguments that may be unfamiliar to new users. What follows is a description of these arguments and data structures and the most common methods for calculating them or their canonical values.

The Transducer Object

FOCUS uses the following MATLAB structure to represent ultrasound transducers:

Transducer
- shape: 'circ','rect',or 'shel'
- radius: value in meters
- width: value in meters
- height: value in meters
- complex_weight: complex value between 0 and 1; default is 1
- time_delay: time delay in seconds
- center: 1x3 vector containing x, y, and z coordinates of the center in meters
- euler: 1x3 vector containing Euler angles describing the rotation in radians

The Medium Object

FOCUS uses the following MATLAB structure to represent media:

Medium
- specificheatofblood: specific heat of blood in J/kg/K
- bloodperfusion: blood perfusion of tissue in $kg/m^3/s$
- density: density of the medium in kg/m^3
- soundspeed: speed of sound in m/s
- powerlawexponent: unitless
- attenuationdBcmMHz: attenuation in dB/cm/MHz
- specificheat: specific heat of medium in J/kg/K
- thermalconductivity: thermal conductivity of medium in W/m/K
- nonlinearityparameter: unitless

define_media

FOCUS is distributed with a MATLAB script called define_media.m. This script defines a frequency variable (f0) if the variable is not present in the workspace and sets it to 1×10^6 Hertz. Then, the script creates the following medium structs

- lossless
- attenuated
- water
- skin
- fat
- muscle

- liver

For details on these objects and their properties, see the documentation for define_media in the FOCUS Function List document.

Coordinate Grids in FOCUS

All FOCUS calculations are performed on a Cartesian cordinate grid in three dimensions. Typically, the grid is defined using maximum and minimum values for each dimension and a spatial step (dx, dy, or dz) with set_coordinate_grid(). For documentation of the other ways you can define a coordinate grid in FOCUS, see the documentaion for set_coordinate_grid in the FOCUS Function List.

ndiv

The value of ndiv describes the number of abscissas used to calculate the pressure integral; a higher ndiv value results in higher accuracy. The required ndiv value depends entirely on the desired accuracy. In general, the standard procedure for calculating the necessary value of ndiv requires the user to calculate a reference pressure field using fnm_cw with a very large ndiv value, for example 200. Then, calculate the same pressure field starting with an ndiv of 1 and using progressively larger values until the desired accuracy is attained.

A good first approximation for ndiv is to use the formula

$$\text{ndiv} = 2 \times \frac{d}{\lambda}$$

where d is the largest dimension of the transducer (or of the largest transducer in an array) and λ is the wavelength of the excitation signal.

Sampling Frequency (f_s) or Delta t

In FOCUS, transient calculations will generate accurate results regardless of the sampling frequency chosen, however, we have found that sampling at the Nyquist frequency is insufficient near the transducer face due to the presence of higher-order harmonics. To avoid aliasing these waves, we recommend a sampling frequency of about eight times the center frequency of the transducer.

Examples

Example 1: Writing Your First Continuous Wave Script

To illustrate how easy it is to get started writing scripts for FOCUS, 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

width = 1e-3;
height = 5e-3;
elements_x = 32;
elements_y = 1;
kerf = 4e-4;
transducer_array = create_rect_planar_array(elements_x, elements_y, width, height, ...
kerf, kerf);


% This sets up our transducer array with 20 0.7 mm x 3 mm elements spaced
% 0.5 mm edge-to-edge. Notice the kerf variable is used twice. This is
% because FOCUS lets you specify different x and y spacings 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();
f0 = 1e6;
lambda = (lossless.soundspeed / f0);
xmin = -(2*width + kerf) * (elements_x/2+1);
xmax = (2*width + kerf) * (elements_x/2+1);
ymin = 0;
ymax = 0;
zmin = 0;
zmax = 50 * lambda;

focus_x = 0;
focus_y = 0;
focus_z = 25 * 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;

delta = [dx dy dz];
coord_grid = set_coordinate_grid(delta, xmin, xmax, ymin, ymax, zmin, 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 focus the
% transducer array.
disp(['Focusing array at (', num2str(focus_x), ', ', num2str(focus_y), ...
', ', num2str(focus_z), ')']);

transducer_array = find_single_focus_phase(transducer_array, focus_x, ...
focus_y, focus_z, lossless, f0, 200);

% The next step is to run the FNM function and display the resulting
% pressure field.

ndiv=6;
tic();

disp('Calculating pressure field...');
p_cw=cw_pressure(transducer_array, coord_grid, lossless, ndiv, f0);
disp(['Simulation complete in ', num2str(toc()), ' seconds.'])

draw_array(transducer_array);

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)');
```

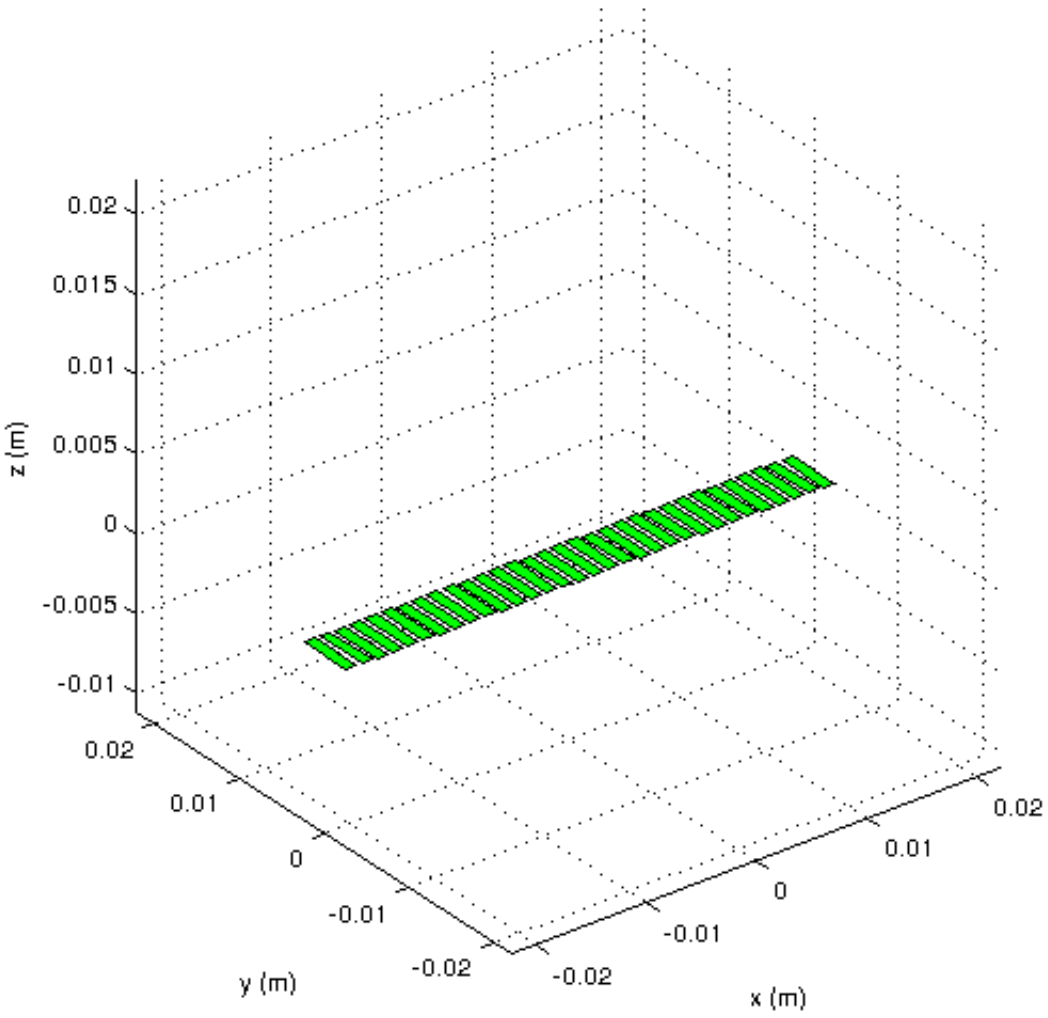

Figure 1: The transducer array used in this simulation.

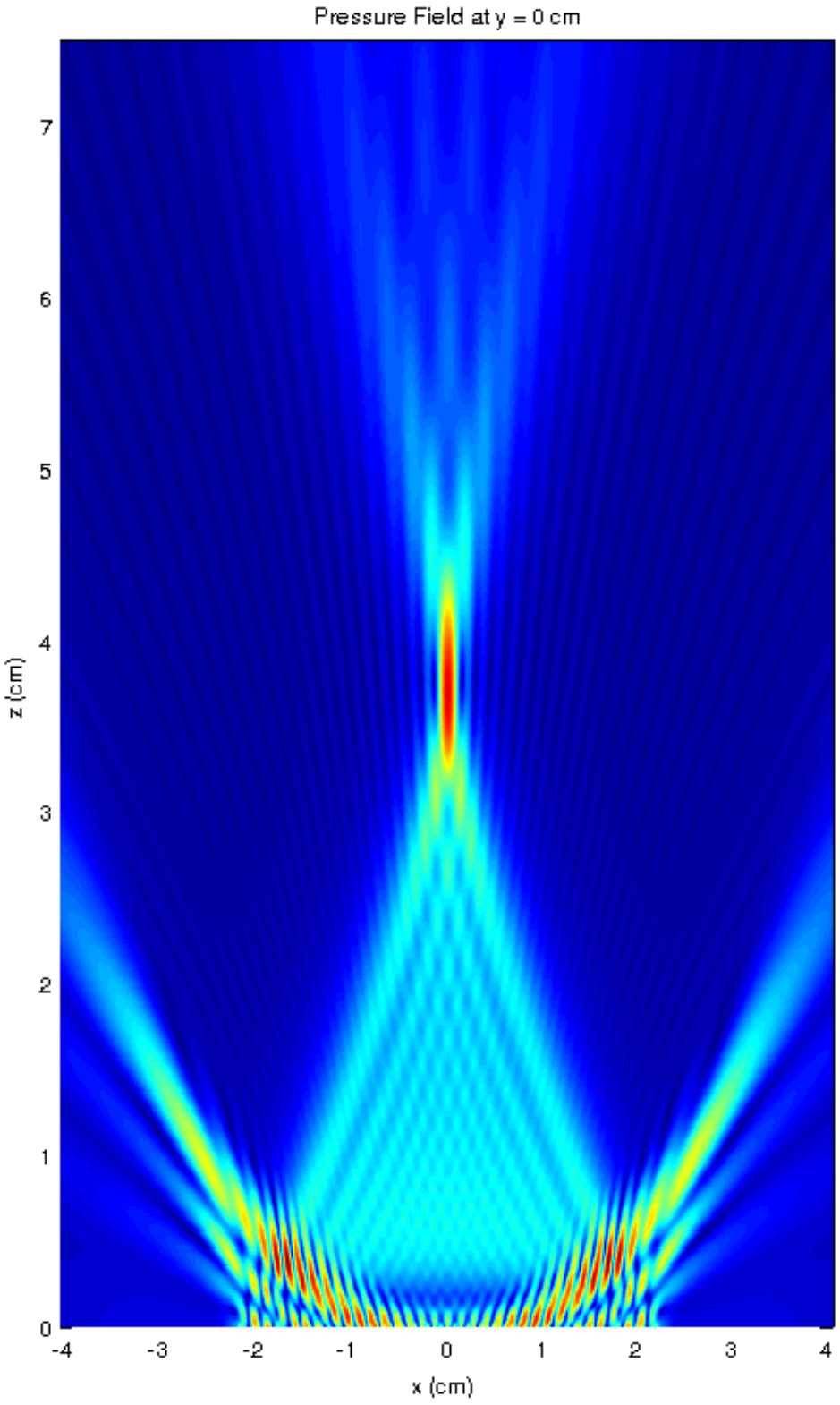

Figure 2: This example takes about 3 seconds to run on a 2.8GHz Intel Core i7 860. The above plot was generated at a higher resolution (800x600) than the plot that will be generated by this script (400x300) for illustration purposes.

Example 2: Writing Your First Transient Script

To demonstrate FOCUS's ability to calculate transient pressure fields, below is an example script that calculates the transient pressure field generated by the same rectagular transducer array using Time-Space Decomposition.

```
% FNM TSD Example

width = 1e-3;
height = 5e-3;
elements_x = 32;
elements_y = 1;
kerf = 4e-4;
transducer_array = create_rect_planar_array(elements_x, elements_y, width, height, ...
kerf, kerf);

draw_array(transducer_array);

% This sets up our transducer array with 20 1 mm x 3 mm elements spaced
% 0.5mm edge-to-edge.

define_media();
fs = 5e6;
f0 = 1e6;
ncycles = 3;
lambda = (lossless.soundspeed / f0);
deltat = 1/fs;

xmin = -(2*width + kerf) * (elements_x/2+1);
xmax = (2*width + kerf) * (elements_x/2+1);
ymin = 0;
ymax = 0;
zmin = 0;
zmax = 15 * lambda;

focus_x = 0;
focus_y = 0;
focus_z = 6 * lambda;

xpoints = 200;
ypoints = 1;
zpoints = 200;

delta = [(xmax-xmin)/xpoints (ymax-ymin)/ypoints (zmax-zmin)/zpoints];
coord_grid=set_coordinate_grid(delta, xmin, xmax, ymin, ymax, zmin, zmax);

[tmin, tmax] = impulse_begin_and_end_times(transducer_array, coord_grid, lossless);
t = tmin:deltat:tmax;
time_struct = set_time_samples(deltat, tmin, tmax);

% This sets up our coordinate grid to cover the width of the transducer
% array on the x axis and from 0 to 15 wavelengths on the z axis. This code
% also sets up a time sampling structure that tells fnm_tsd to sample from
% t=0 to t=2 periods at intervals of the period of the sampling frequency
% (5MHz). Now for the excitation function.
```

```
input_func = set_excitation_function(2, f0, ncycles/f0, 0);

% This sets our excitation function to be a Hanning weighted tone burst
% with an amplitude of 1 and center frequency of f0 (1MHz). See the
% documentation for set_excitation_function for details on how FOCUS
% handles excitation functions. The next step is to focus the array.
disp(['Focusing array at (', num2str(focus_x), ', ', num2str(focus_y), ...
', ', num2str(focus_z), ')']);

transducer_array = set_time_delays(transducer_array, focus_x, ...
focus_y, focus_z, lossless, fs);

% Now we run the FNM TSD function and display the output.

ndiv=4;
tic();

disp('Calculating pressure field...');
p_tsd=transient_pressure(transducer_array, coord_grid, lossless, time_struct, ndiv, input_func);

disp(['Simulation complete in ', num2str(toc()), ' seconds.'])
maxpressure = max(max(max(max(p_tsd))));
nt = size(p_tsd, 4);
x = xmin:delta(1):xmax;
y = ymin:delta(2):ymax;
z = zmin:delta(3):zmax;

figure();
for it = 1:nt,
    mesh(z*100, x*100, squeeze(p_tsd(:, :, :, it)))
    title(['FNM TSD Example, t = ', sprintf('%0.3f',(it/fs) * 1e6), '\mu','s'])
    zlabel('pressure (Pa)')
    xlabel('z (cm)')
    ylabel('x (cm)')
    temp=axis();
    temp(5)=-maxpressure;
    temp(6)=maxpressure;
    axis(temp);
    drawnow
end
```

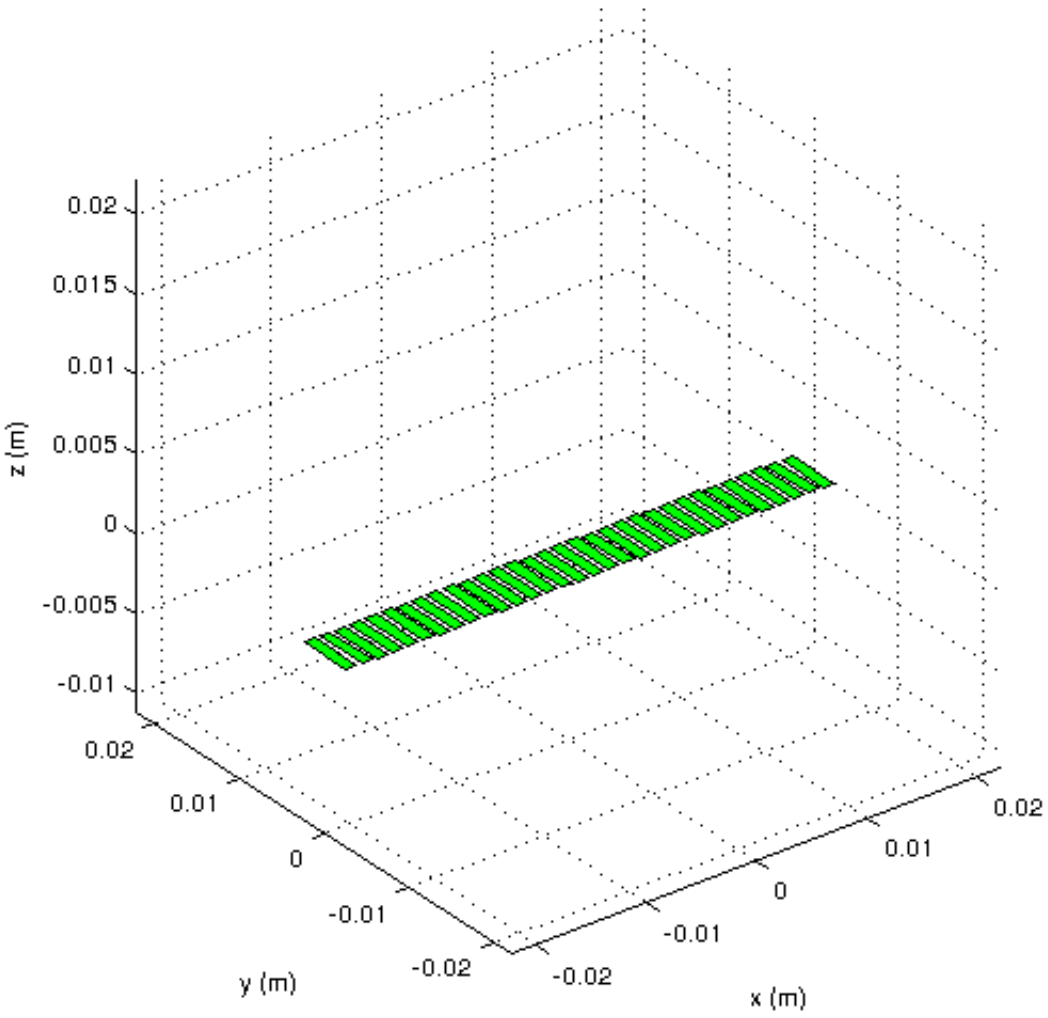

Figure 3: The transducer array used in this simulation.

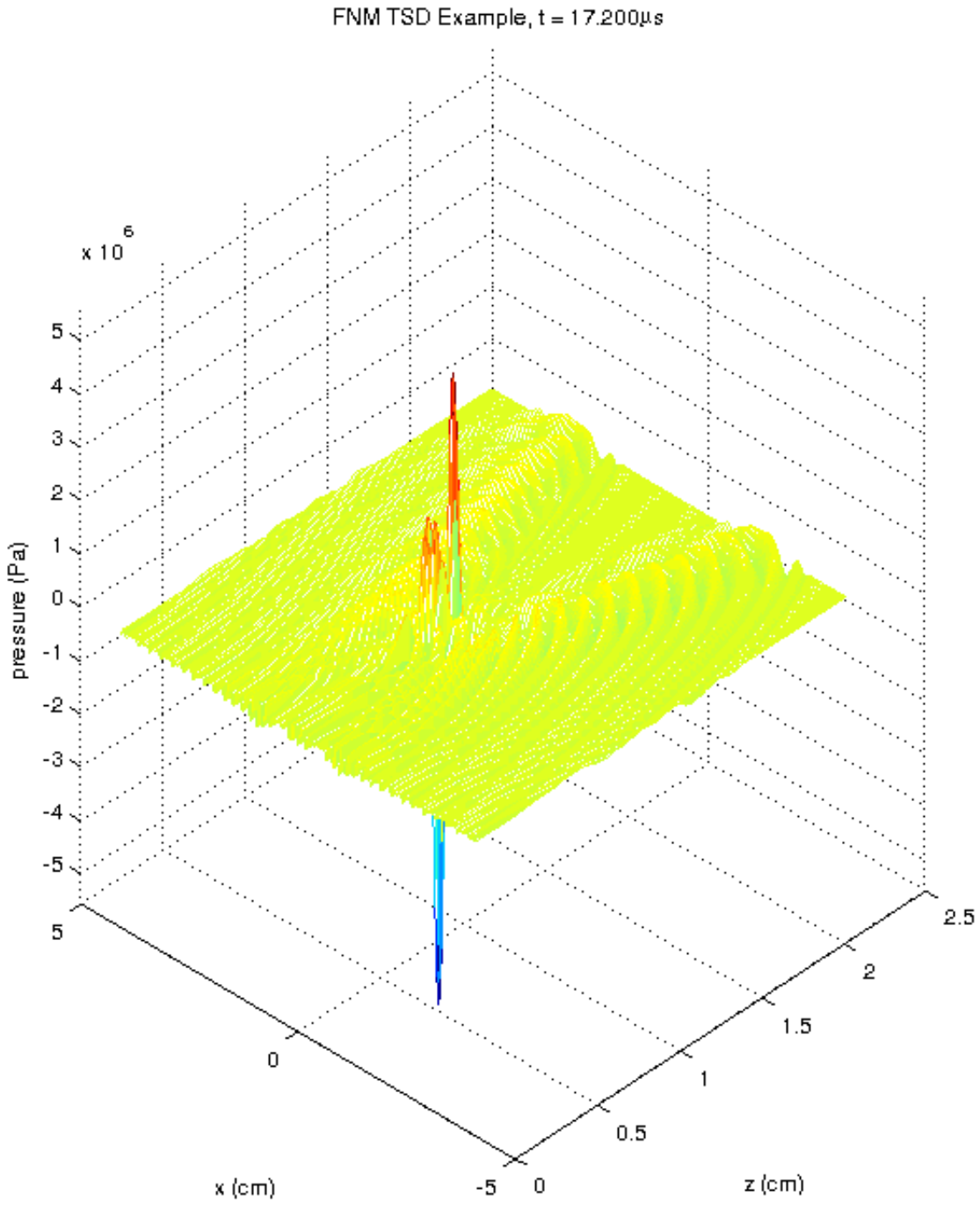

Figure 4: This simulation takes about 3 seconds to run on a 2.8GHz Intel Core i7 860. The resulting plot will be animated.

Example 3: Continuous Wave Excitation of a Cylindrical Section Array

FOCUS also allows users to simulate transducer arrays of various geometries. In this example, we will simulate the pressure field generated by continuous wave excitation of a cylindrical section array of rectangular transducers.

```
% FNM CW CSA Example

width = 1e-3;
height = 5e-3;
elements_x = 64;
elements_y = 1;
kerf = 4e-4;
spacing = width + kerf;
r_curv = (elements_x*spacing)/pi;
transducer_array = create_rect_csa(elements_x, elements_y, width, height,...
kerf, kerf, r_curv);
draw_array(transducer_array);

% This sets up a cylindrical section transducer array with 20 0.7 mm x 3 mm
% elements spaced 0.5 mm edge-to-edge.

define_media();
f0 = 1e6;
lambda = (lossless.soundspeed / f0);
xmin = -1.2 * r_curv;
xmax = 1.2 * r_curv;
ymin = 0;
ymax = 0;
zmin = r_curv; % Don't capture the pressure field inside the transducer array
zmax = 40 * lambda;

focus_x = 0;
focus_y = 0;
focus_z = 30 * 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;

delta = [dx dy dz];
coord_grid = set_coordinate_grid(delta, xmin, xmax, ymin, ymax, zmin, 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 80 wavelengths in the z direction. Now we need to focus the
```

```
% transducer array.
disp(['Focusing array at (', num2str(focus_x), ', ', num2str(focus_y), ...
', ', num2str(focus_z), ')']);

transducer_array = find_single_focus_phase(transducer_array, focus_x, ...
focus_y, focus_z, lossless, f0, 200);

% The next step is to run the FNM function and display the resulting
% pressure field.

ndiv=3;
tic();

disp('Calculating pressure field...');
p_cw=cw_pressure(transducer_array, coord_grid, lossless, ndiv, f0);
disp(['Simulation complete in ', num2str(toc()), ' seconds.'])

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)');
```

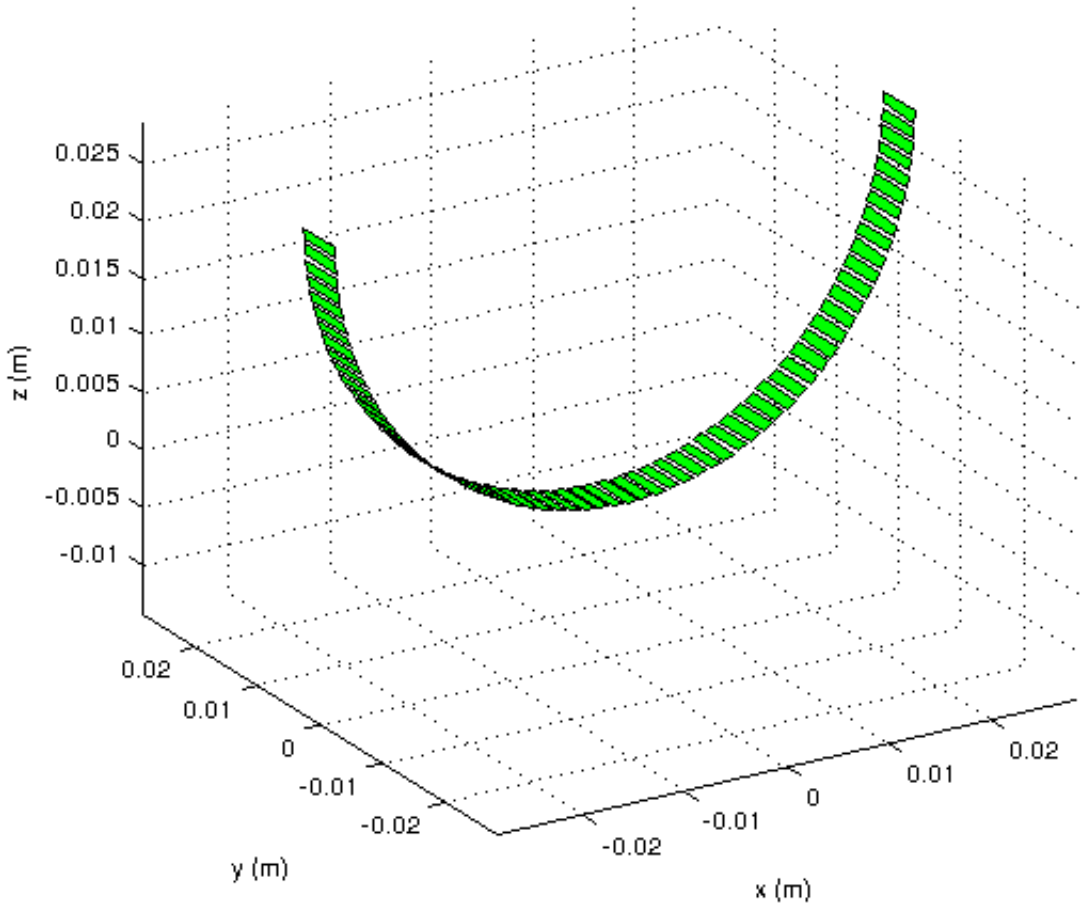

Figure 5: The transducer array used in this simulation.

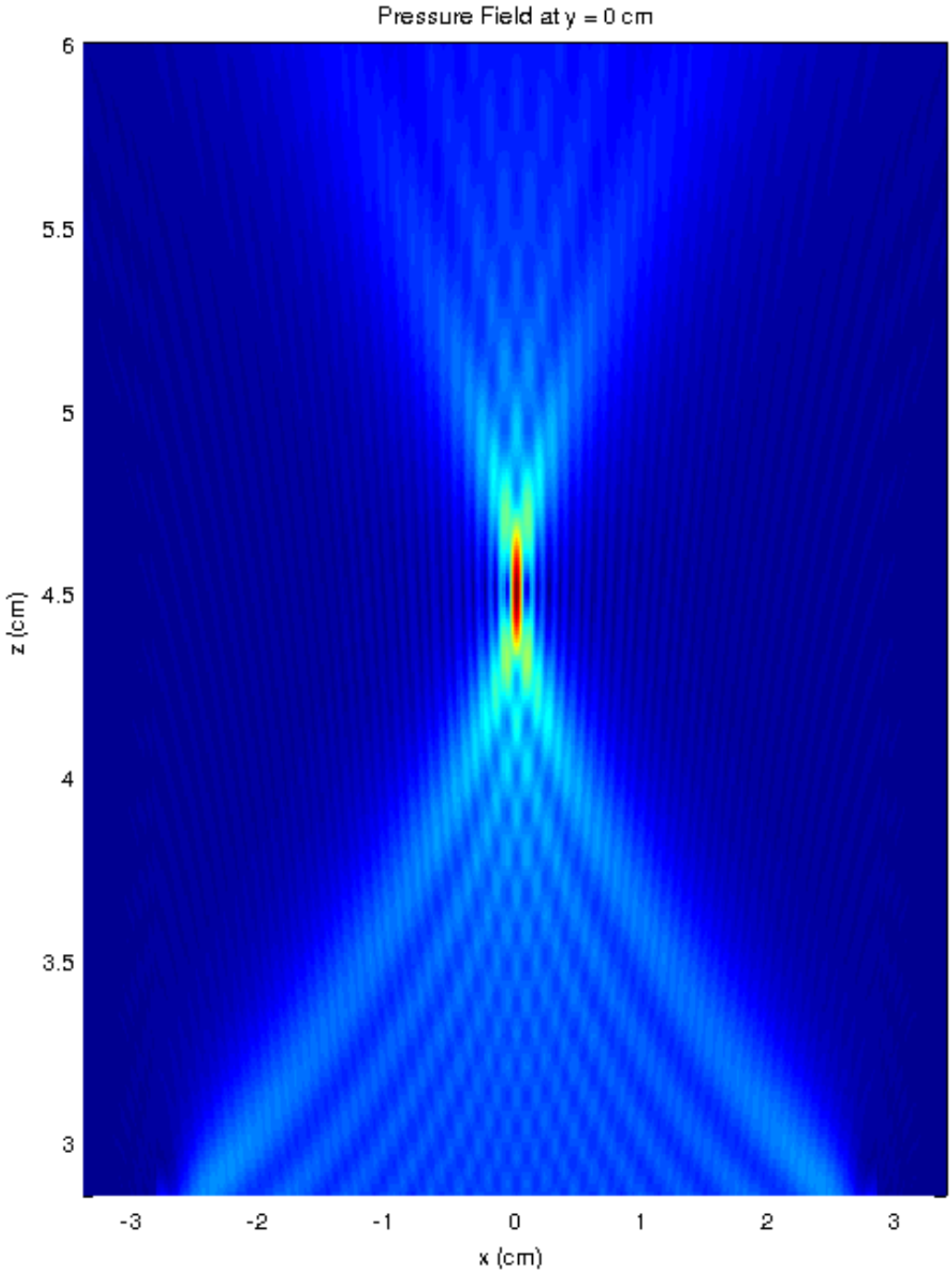

Figure 6: This simulation takes about 3 seconds to run on a 2.8GHz Intel Core i7 860.

Example 4: Using the Angular Spectrum Approach

Calculating 3D pressure fields often takes a very long time. The Angular Spectrum Approach models the diffraction of acoustic waves by superposing a number of plane waves travelling in different directions and propagating them in the spatial frequency domain. This is much faster than integral approaches that calculate the field at each point (like Rayleigh-Sommerfeld and the Fast Nearfield Method) because it takes advantage of the speed of the Fast Fourier Transform.

In order to use the ASA code distributed with FOCUS, you must first calculate a source pressure plane for the ASA to propagate. This field is usually calculated using the Fast Nearfield Method at a position on the z-axis that is very close to the face of the transducer array, usually around $z = \frac{\lambda}{2}$ or $z = \lambda$. When calculating this source plane, it is very important to make sure that as much spectral information as possible is captured. We therefore reccommend that users use a coordinate grid that extends 10-20% beyond the x and y dimensions of their array. For example, with an array that extends from -10cm to 10cm in x and -1cm to 1cm in y, a good coordinate grid should span at least x = -12cm to 12cm and y = -1.2cm to 1.2cm. It is also important to make sure that the coordinate grid is uniformly-spaced, i.e. $dx = dy = dz$.

The FOCUS function that implements the Angular Spectrum Approach is called `asa_call`. It requires a source pressure plane, its location, all of the z-coordinates where the pressure should be calculated, the medium being used, the number of FFT terms to use in the calculation, the grid spacing, the type of ASA calculation to use, and the excitation frequency of the array. Below is an example that uses the Fast Nearfield Method to calculate the source pressure plane at $z = \frac{\lambda}{4}$ and propagate it out to $z = 20\lambda$. The following example can also be found in the Examples directory of FOCUS.

```
% Set up the array
ele_x = 128;
ele_y = 1;
width = 0.245e-3;
height = 7e-3;
kerf_x = 0.03e-3;
kerf_y = 0;

xdc = create_rect_planar_array(ele_x, ele_y, width, height, kerf_x, kerf_y);

f0 = 1e6;
medium = set_medium('lossless');
lambda = medium.soundspeed/f0;

% Set up the coordinate grid
xmin = -((ele_x/2) * (width+kerf_x))*1.2;
xmax = -xmin;
ymin = -((ele_y/2) * (height+kerf_y))*1.2;
ymax = -ymin;
zmin = 0;
zmax = 40*lambda;

focus_x = 0;
focus_y = 0;
focus_z = 20 * lambda;

dx = lambda/8;
dy = lambda/8;
dz = lambda/8;
```

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

% Determine where the source pressure will be calculated
z0 = lambda/4;
y_index = floor((ymax-ymin)/2/dy);

cg_p0 = set_coordinate_grid([dx dy 1], xmin,xmax,ymin,ymax,z0,z0);
cg_z = set_coordinate_grid([dx 1 dz],xmin,xmax,ymin,ymax,zmin,zmax);

% Focus the array
xdc = find_single_focus_phase(xdc,focus_x,focus_y,focus_z,medium,f0,200);

% Calculate the pressure
ndiv = 10;
disp('Calculating p0 with FNM... ');
tic();
p0 = cw_pressure(xdc, cg_p0, medium, ndiv, f0);
disp(['Done in ', num2str(toc()), ' seconds.']);

disp(['Calculating 3D pressure (', length(x) * length(y) * length(z), ' points) with ASA... ']);
tic();
p_asa = cw_angular_spectrum(p0, cg_z, medium, f0, 1024);
disp(['Done in ', num2str(toc()), ' seconds.']);

figure(1);
pcolor(x*1000, y*1000, rot90(abs(squeeze(p0(:,:,1)))));
xlabel('x (mm)');
ylabel('y (mm)');
shading flat;
title(['p0 (Calculated with FNM at z = ', num2str(z0*1000), ' mm)']);

figure(2);
pcolor(z*1000, x*1000, abs(squeeze(p_asa(:,y_index,:))));
xlabel('z (mm)');
ylabel('x (mm)');
shading flat;
title('ASA Pressure (y=0)');
```

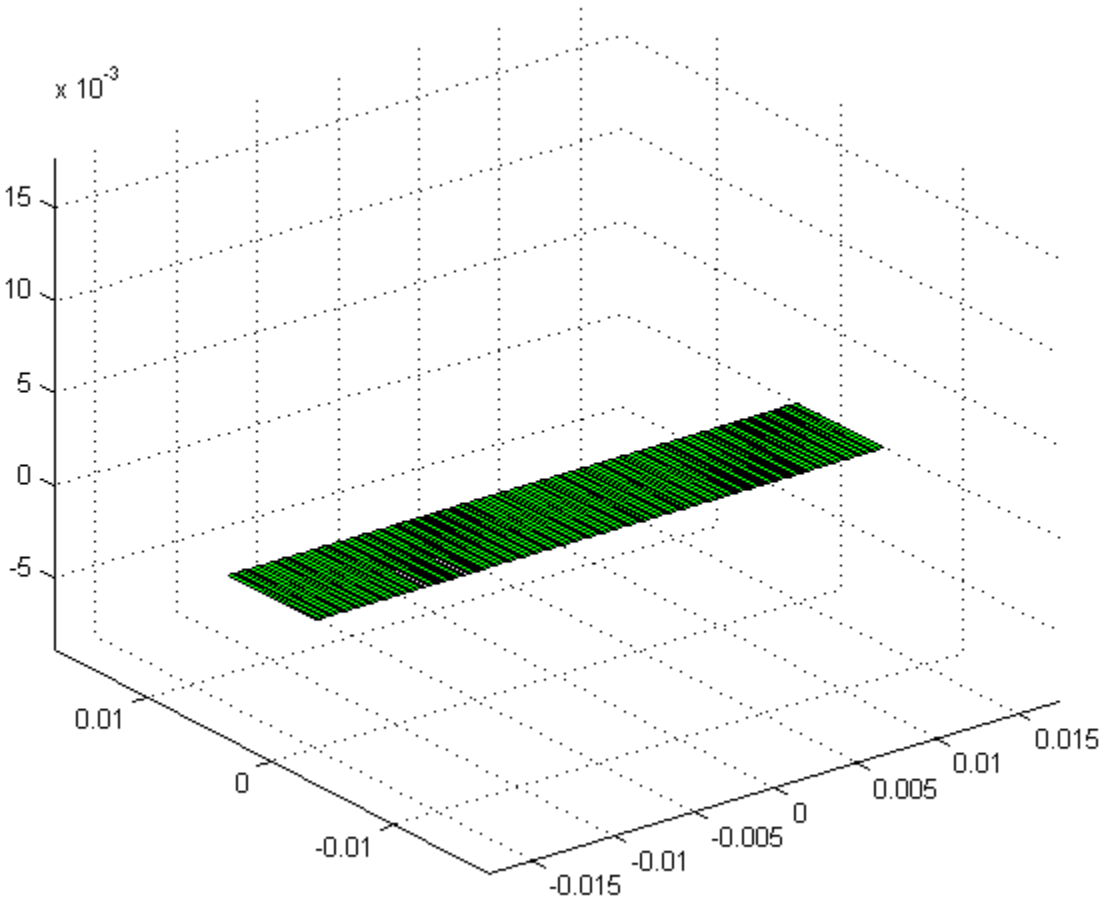

Figure 7: The transducer array used in this simulation.

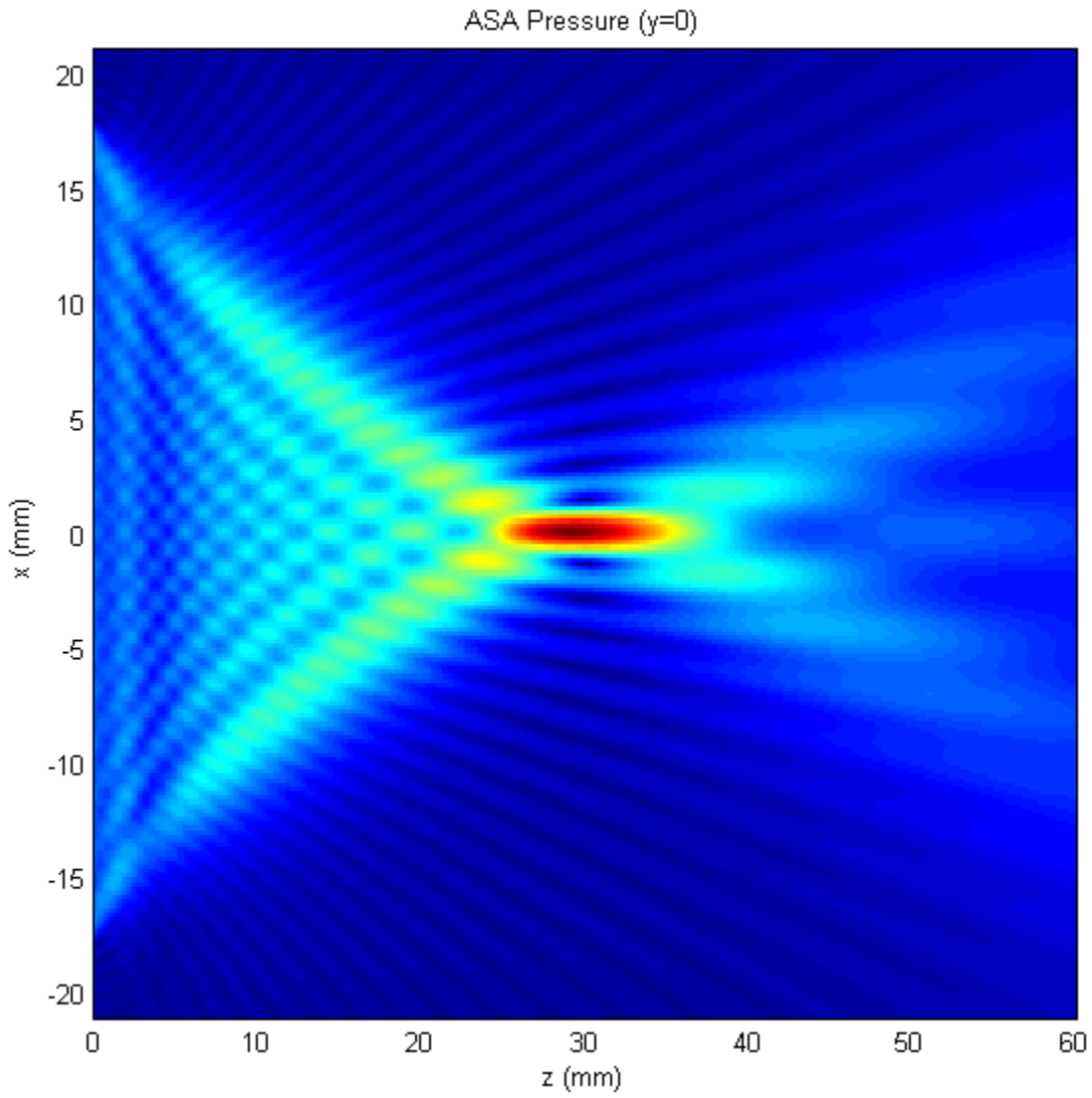

Figure 8: This simulation takes about 20 seconds to run on a 2.8GHz Intel Core i7 860. Compare that to about 3000 seconds to perform the same calculation with FNM.