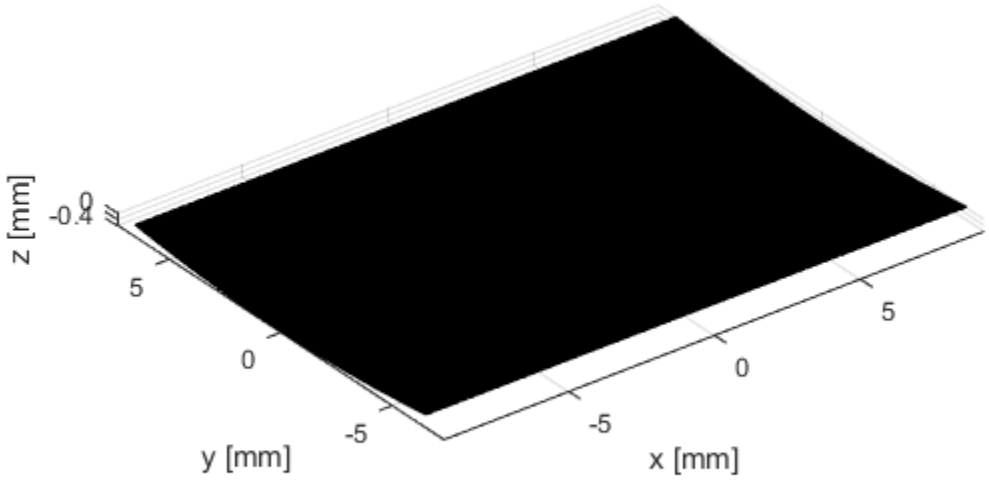
Table of Contents


```
close all;
clear all;

no_elements = 64;
pitch = 0.29e-3;
kerf = 0.025e-3;
width = pitch - kerf;
height=13e-3;
no_sub_x = 5;
no_sub_y = 30;
%focus = [0 0 60]/1000;
steeringAngle = -35;
Rfocus = 60e-3;
focusRange=60e-3;
focus =
 focusRange*([sin(steeringAngle*pi/180),0,cos(steeringAngle*pi/180)]);
c =1540;
field_init(0);

Th_tnx = xdc_focused_array(no_elements, width, height, kerf, Rfocus,
 no_sub_x, no_sub_y, focus);
Th_rcv = xdc_focused_array(no_elements, width, height, kerf, Rfocus,
 no_sub_x, no_sub_y, focus);
figure;
show_xdc_geir(Th_tnx, 1);
axis equal;
view(3);

%
fs = 100e6; %sampling freq (100Mhz)
f0 = 2.5e6; % transducer center freq (2.5Mhz)
t0 = 1/f0;
dt = 1/fs;  %sampling period
set_sampling(fs);

      *-------------------------------------------------------------*
```

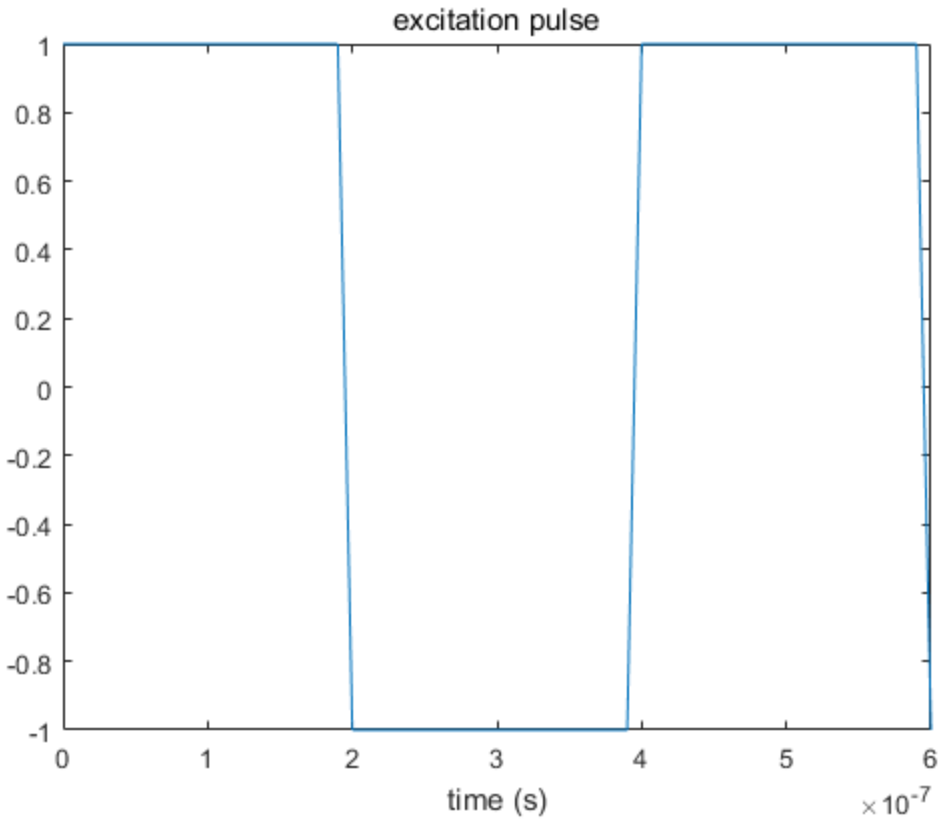
```
 *                                                                 *
 *                    F I E L D   I I                              *
 *                                                                 *
 *              Simulator for ultrasound systems                   *
 *                                                                 *
 *            Copyright by Joergen Arendt Jensen                   *
 *    Version 3.24, May 12, 2014 (Matlab 8.20 version)             *
 *                 Web-site: field-ii.dk                           *
 *                                                                 *
 *     This is citationware. Note the terms and conditions         *
 *     for use on the web-site at:                                 *
 *               field-ii.dk/?copyright.html                       *
 *  It is illegal to use this program, if the rules in the         *
 *  copyright statement is not followed.                           *
 *-----------------------------------------------------------------*
Read rectangular data for plotting....
Plots aperture with physical element number...
Warning:  Remember to set all pulses in apertures for the new sampling
 frequency
```

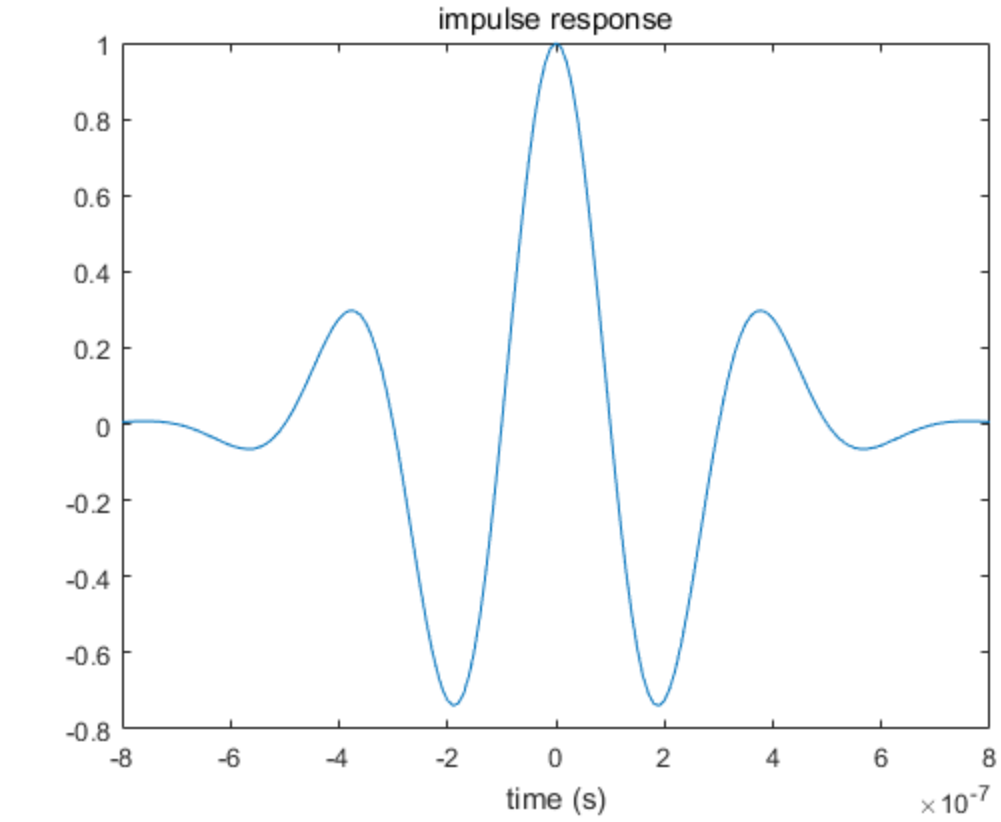


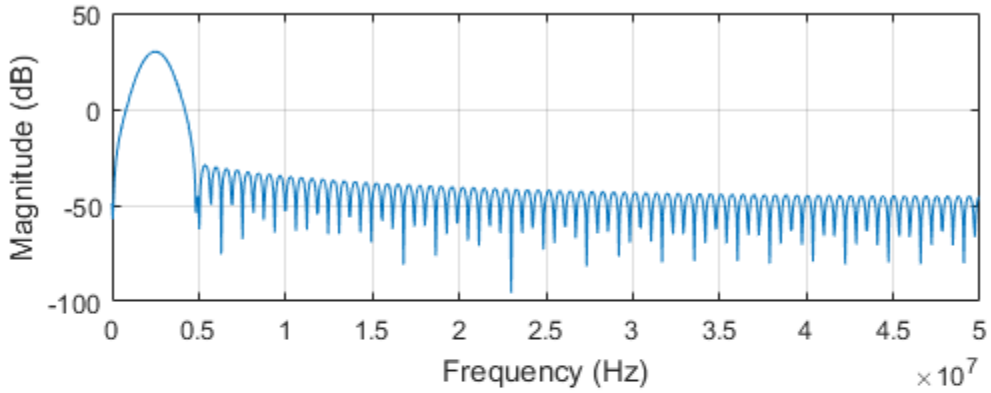
Impulse setup

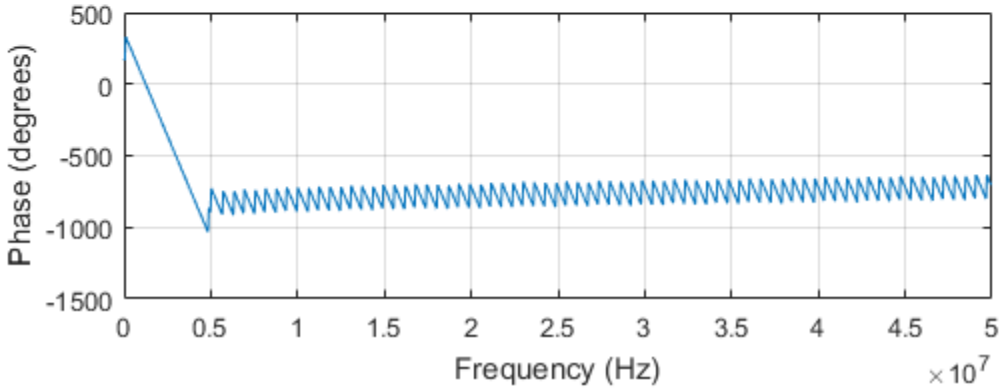
```
t_ir = -2/f0:1/fs:2/f0;
Bw = 0.6;
impulse_response = gauspuls(t_ir, f0, Bw);
xdc_impulse (Th_tnx, impulse_response);
```

```
xdc_impulse (Th_rcv, impulse_response);

figure;
excitation = square(2*pi*f0*(0:dt:1.5*t0));
plot(0:dt:1.5*t0, excitation);
xlabel("time (s)");
title("excitation pulse");
xdc_excitation(Th_tnx, excitation);
figure;
plot(t_ir, impulse_response);
xlabel("time (s)");
title("impulse response");
figure;
freqz(impulse_response,1,1024,fs);
```




impulse response
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-8
-6
-4
-2
0
2
4
6
8
time (s)
×10^-7


Magnitude (dB)
50
0
-50
-100
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Frequency (Hz)
×10^7


Phase (degrees)
500
0
-500
-1000
-1500
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Frequency (Hz)
×10^7

setup tx Apod

```
txApodWeights = ones(1, no_elements);
%txApodWeights = hanning(no_elements)';
%txApodWeights = tukeywin(no_elements, 0.3)';
%figure;
%stem(txApodWeights);
%xdc_apodization(Th_tnx, 0, txApodWeights);
```

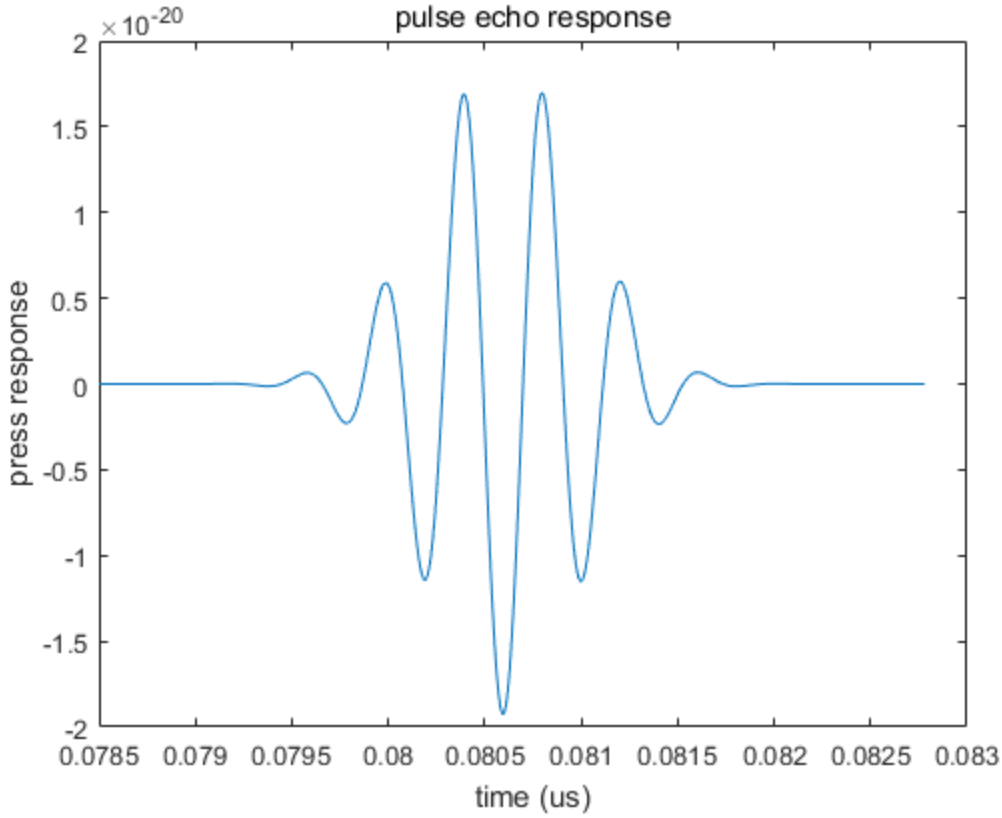
setup rx Apod

```
rxApodWeights = ones(1, no_elements);
%rxApodWeights = hanning(no_elements)';
% rxApodWeights = tukeywin(no_elements, 0.3)';
% figure;
% stem(rxApodWeights);
% xdc_apodization(Th_rcv, 0, rxApodWeights);

receiveAngle = -35;
xdc_dynamic_focus(Th_rcv, 0, receiveAngle*pi/180, 0);
```

pressure response from focus point

Nx = 81; Nz = 59; x0=linspace(-25e-3,25e-3,Nx); z0=linspace(5e-3,250e-3,Nz); [X,Z]=meshgrid(x0,z0); measure_point = [X(:), zeros(length(X(:)),1),Z(:)];

```
measure_point = focus;

[hhp_x0, t_start]=calc_hhp(Th_tnx, Th_rcv, measure_point);
figure;
tAx_hp = t_start+(0:length(hhp_x0)-1)/fs;
plot(tAx_hp*1000, hhp_x0);
title("pulse echo response");
xlabel("time (us)");
ylabel("press response");
```



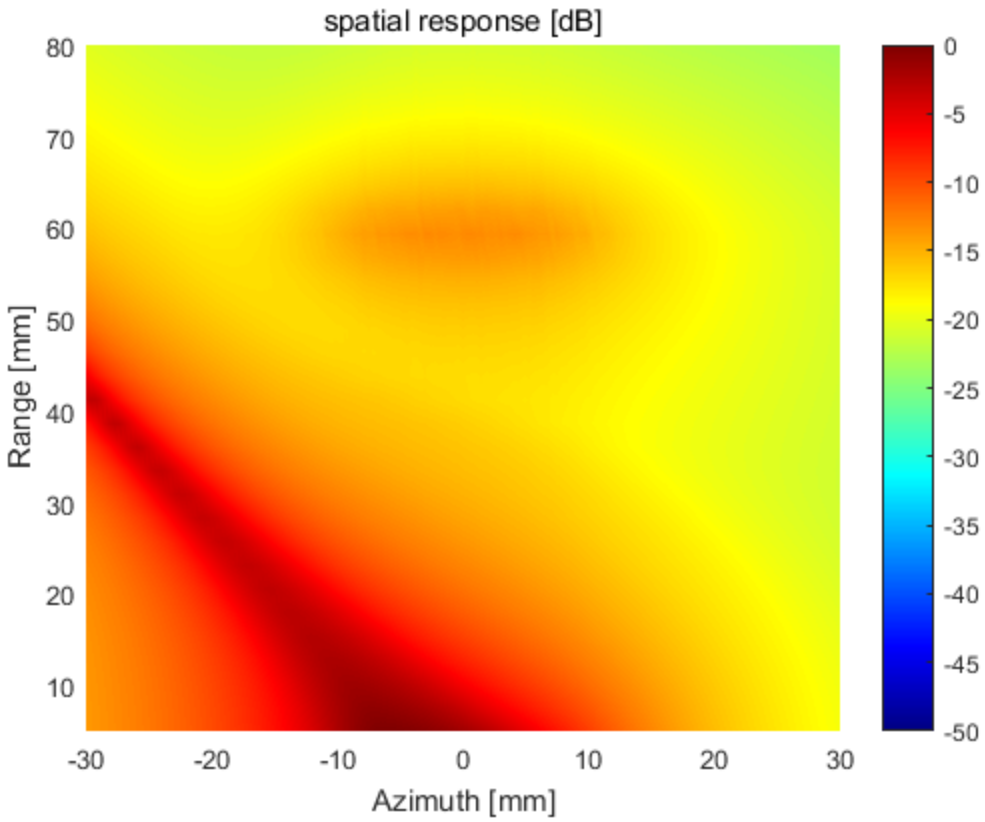
grid setup

```
Nx = 161; Nz = 30;
x0=linspace(-30e-3,30e-3,Nx)';
z0=linspace( 5e-3,80e-3,Nz)';
[X,Z]=meshgrid(x0,z0);
measure_point = [X(:), zeros(length(X(:)),1),Z(:)];
```

spatial response

```
[h_tx0, t_start]=calc_h(Th_tnx, measure_point);
figure;
bp = sqrt(mean(h_tx0.^2));
bp = reshape(bp, Nz, Nx);
bp = bp/max(bp(:));
pcolor(x0*1000,z0*1000, 20*log10(bp));
shading interp
title("spatial response [dB]")
xlabel('Azimuth [mm]');
ylabel('Range [mm]');
caxis([-50 0]); % Set dynamic range
colormap(jet(256));
colorbar
```

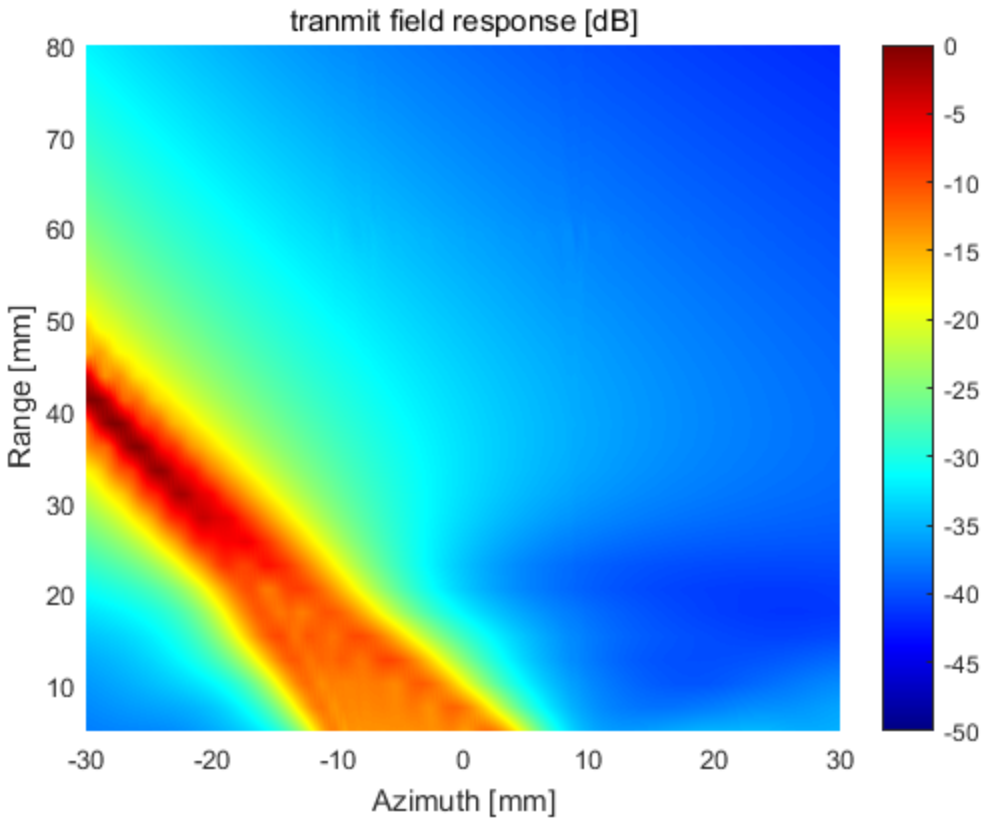
3 seconds used for the calculation



tranmit field response

```
[hp_tx0, t_start]=calc_hp(Th_tnx, measure_point);
figure;
bp = sqrt(mean(hp_tx0.^2));
bp = reshape(bp, Nz, Nx);
bp = bp/max(bp(:));
pcolor(x0*1000,z0*1000, 20*log10(bp));
shading interp
title("tranmit field response [dB]")
xlabel('Azimuth [mm]');
ylabel('Range [mm]');
caxis([-50 0]); % Set dynamic range
colormap(jet(256));
colorbar
```

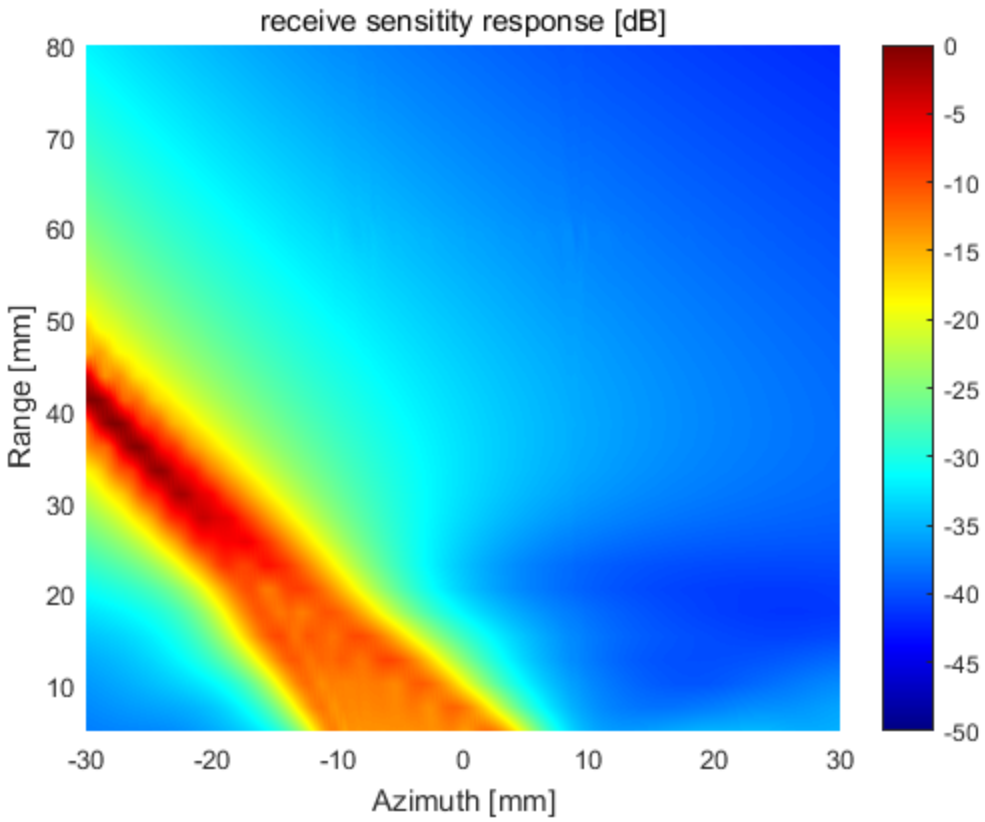
3 seconds used for the calculation



receive sensitity response

```
[hp_rv0, t_start]=calc_hp(Th_tnx, measure_point);
figure;
bp = sqrt(mean(hp_rv0.^2));
bp = reshape(bp, Nz, Nx);
bp = bp/max(bp(:));
pcolor(x0*1000,z0*1000, 20*log10(bp));
shading interp
title("receive sensitity response [dB]")
xlabel('Azimuth [mm]');
ylabel('Range [mm]');
caxis([-50 0]); % Set dynamic range
colormap(jet(256));
colorbar
```

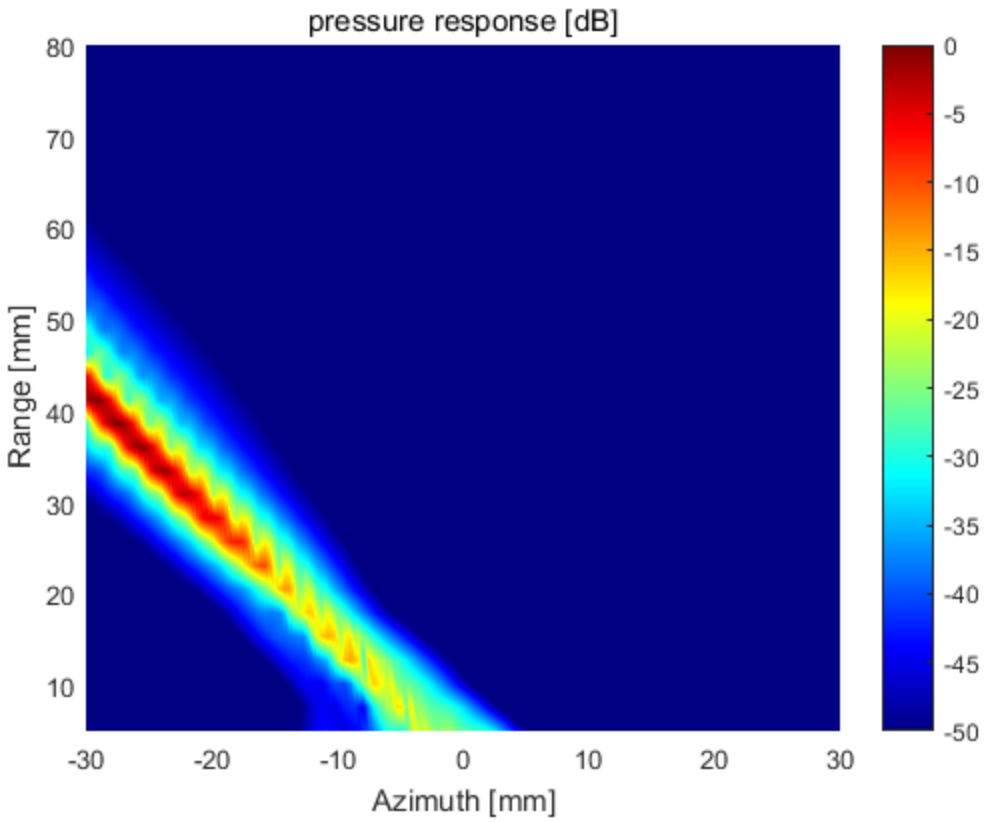
4 seconds used for the calculation



pressure response from XZ plane

```
[hhp_x0, t_start]=calc_hhp(Th_tnx, Th_rcv, measure_point);
figure;
bp = sqrt(mean(hhp_x0.^2));
bp = reshape(bp, Nz, Nx);
bp = bp/max(bp(:));
pcolor(x0*1000,z0*1000, 20*log10(bp));
shading interp
title("pressure response [dB]")
xlabel('Azimuth [mm]');
ylabel('Range [mm]');
caxis([-50 0]); % Set dynamic range
colormap(jet(256));
colorbar
```

```
  72.6 % performed (roughly  2 seconds remaining)
          8 seconds used for the calculation
```



pressure response from XZ plane based on local depth max

figure; bp = sqrt(mean(hhp_x0.^2)); bp = reshape(bp, Nz, Nx); bp = bp./repmat(max(bp')', 1,Nx); pcolor(x0*1000,z0*1000, 20*log10(bp)); shading interp title("pressure response [dB]") xlabel('Azimuth [mm]'); ylabel('Range [mm]'); caxis([-50 0]); % Set dynamic range colormap(jet(256)); colorbar

```
focalDepth = focus(3);
transmitApertureSize = no_elements*pitch - kerf;
receiveApertureSize = no_elements*pitch - kerf;
lambda = c/f0;

beamwidth = focalDepth/(transmitApertureSize
+receiveApertureSize)*lambda;

s_beamwidth = sprintf("-6db beamwidth = %0.2gmm",beamwidth*1000)
```

```
s_beamwidth =

    "-6db beamwidth = 0.82mm"
```

Published with MATLAB® R2017a