
Users' guide for the Beamformation Toolbox

Release 1.3

Svetoslav Ivanov Nikolov

Aug 31 2000

Department of Information Technology, Build. 344,
Technical University of Denmark
DK-2800 Lyngby, Denmark

CONTENTS

1	Introduction	1
2	Description of Matlab Functions	3
2.1	Functions list	3
2.2	Function description	4
3	Examples	11
3.1	Using Field II simulations	11

Introduction

This is the user guide to the Beamformation Toolbox. This toolbox is available only for Matlab under Linux, and is intended for processing raw RF data recorded by an ultrasound system. Typical applications include synthetic aperture focusing and beamformation of data recorded by experimental systems such as XTRA or the *Experimental Ultrasound System for Real Time Synthetic Aperture Focusing*. All the functions have calling conventions like **Field-II**.

The default behaviour of the toolbox is to beamform only one line at a time, also like in Field II. In this case the user can omit the last of the arguments in all the functions, which usually is *line_no*.

However, there are some differences between this toolbox and **Field-II**. For the purposes of the synthetic aperture focusing it is necessary to beamform a whole low-resolution image at every emission. This can be achieved by setting the number of simultaneously beamformed lines using the command `bft_no_lines`. Then each of the lines must be described before calling `bft_beamform`.

For the time-being there are some limitations :

- Two low-resolution images can not be added if the lines are specified as dynamically focused.

As it can be seen the library is fully functional when the focus lines are specified by a number of focal points¹, and for the case of dynamic focusing, low resolution images cannot be summed.

In the future the toolbox will try to support many different types of strange and wacky algorithms, so - stay tuned.

Svetoslav Ivanov Nikolov
February 29, 2000.

¹There is a speed difference when the apodization is set to *ones* or if no apodization is set at all.

Description of Matlab Functions

2.1 Functions list

<code>bft_add_image</code>	Add a low resolution to hi resolution image.
<code>bft_apodization</code>	Create a apodization time line.
<code>bft_beamform</code>	Beamform a number of scan-lines.
<code>bft_center_focus</code>	Set the center focus point for the focusing.
<code>bft_dynamic_focus</code>	Set dynamic focusing for a line.
<code>bft_end</code>	Release all resources, allocated by the beamforming toolbox.
<code>bft_focus</code>	Create a focus time line defined by focus points.
<code>bft_focus_times</code>	Create a focus time line defined by focus delays.
<code>bft_free_xdc</code>	Free the memory allocated for a transducer definition.
<code>bft_init</code>	Initialize the BeamForming Toolbox.
<code>bft_linear_array</code>	Create a linear array.
<code>bft_no_lines</code>	Set the number of lines that will be beamformed in parallel.
<code>bft_param</code>	Set a paramater of the BeamForming Toolbox.
<code>bft_sub_image</code>	Subtract one low-res image from high-res one.
<code>bft_sum_apodization</code>	Create a summation apodization time line.
<code>bft_sum_images</code>	Sum 2 low resolution images in 1 high resolution.
<code>bft_transducer</code>	Create a new transducer definition.

2.2 Function description

BFT user's guide

bft_add_image

Add a low resolution to high resolution image.

USAGE: [hi_res]=bft_add_image(hi_res,lo_res,element,start_time)
INPUT: hi_res High resolution RF image. One column per scan line.
 lo_res Low resolution RF image. One column per scan line.
 element Number of element, used to acquire the low resolution image.
 time Arrival time of the first sample of the RF lines. [sec]
OUTPUT: hi_res - The high resolution image

BFT user's guide

bft_apodization

Create an apodization time line.

USAGE: bft_apodization(xdc, times, values, line_no)
INPUT: xdc Pointer to a transducer aperture
 times Time after which the associated apodization is valid
 values Apodization values. Matrix with one row for each time value and a number of
 columns equal to the number of physical elements in the aperture.
 line_no Number of line. If skipped, line_no is assumed to be equal to '1'.
OUTPUT: None

BFT user's guide

bft_beamform

Beamform a number of scan-lines. The number of the simultaneously beamformed scan-lines is set by **bft_no_lines**. If **bft_no_lines** is not called, only one scan line will be beamformed.

USAGE: bf_lines = bft_beamform(time, rf_data)
INPUT: time The time of the first sampled value
 rf_data The recorded RF data. The number of columns is equal to the number of
 elements.
OUTPUT: bf_lines Matrix with the beamformed data. The number of rows of bf_lines is equal to the number
 of rows of rf_data. The number of columns is equal to the number of lines

Set the center focus point for the focusing. This point is used as a reference for calculating the focusing delay times and as a starting point for dynamic focusing.

USAGE: bft_center_focus(point, line_no)
INPUT: point The center point [x,y,z] [m]
 line_no Number of line. If omitted in the parameter list *line_no* is assumed equal to 1
OUTPUT: None

Set dynamic focusing for a line

USAGE: bft_dynamic_focus(xdc, dir_xz, dir_zy, line_no)
INPUT: xdc Pointer to the transducer aperture
 dir_xz Direction (angle) in radians for the dynamic focus. The direction is taken from
 the center for the focus of the transducer in the z-x plane.
 dir_zy Direction (angle) in radians for the dynamic focus. The direction is taken from
 the center for the focus of the transducer in the z-y plane.
 line_no Number of line. If skipped, *line_no* is assumed to be equal to '1'.
OUTPUT: None

Release all resources allocated by the beamforming toolbox.

USAGE: bft_end
INPUT: None
OUTPUT: None

Create a focus time line.

USAGE: `bft_focus(xdc, times, points, line_no)`
INPUT: `xdc` Pointer to aperture.
`times` Time after which the associated focus is valid
`points` Focus points. Vector with three columns (x,y,z) and one row for each field point.
`line_no` Number of line for which we set the focus. If skipped, `line_no` is assumed equal to '1'.
OUTPUT: none

BFT user's guide

bft_focus_times

Create a focus time line. The user supplies the delay times for each element.

USAGE: `bft_focus_times(xdc, times, delays, line_no)`
INPUT: `xdc` Pointer to a transducer aperture.
`times` Time after which the associated delay is valid.
`delays` Delay values. Matrix with one row for each time value and a number of columns equal to the number of physical elements in the aperture.
`line_no` Number of line. If skipped, `line_no` is assumed to be equal to 1.
OUTPUT: None

BFT user's guide

bft_free_xdc

Free the memory allocated for a transducer definition

USAGE: `bft_free_xdc(xdc)`
INPUT: `xdc` Pointer to the memory location returned by the function `bft_transducer`
OUTPUT: Nothing

BFT user's guide

bft_init

Initialize the BeamForming Toolbox. This command must be executed first in order to set some parameters and allocate the the necessary memory

USAGE: bft_init
INPUT: None
OUTPUT: None

BFT user's guide

bft_linear_array

Create a linear array aperture.

USAGE: xdc = bft_linear_array(no_elements, width, kerf)
 xdc = bft_linear_array(no_elements, pitch)
INPUT: no_elements Number of elements in the array
 pitch Distance between the centers of two elements [m]
 width Width in x-direction [m]
 kerf Distance between two elements [m]
 The function assumes that $kerf + width = pitch$
OUTPUT: xdc - Pointer to the allocated aperture.

BFT user's guide

bft_no_lines

Set the number of lines that will be beamformed in parallel. After calling **bft_init**, the number of lines that are beamformed in parallel is 1. If the user wants to beamform a whole image in one command, he/she must set the number of lines, and then specify the focal zones for each of the lines.

USAGE: bft_no_lines(no_lines)
INPUT: no_lines Number of lines beamformed in parallel
OUTPUT: None

BFT user's guide

bft_param

Set a parameter of the BeamForming Toolbox

USAGE: `bft_param(name, value)`

INPUT: `name` Name of the parameter (string). Currently supported:

name	Meaning	Default value	Unit
'c'	Speed of sound.	1540	m/s
'fs'	Sampling frequency	40,000,000	Hz

`value` New value for the parameter. Must be scalar.
OUTPUT: None

BFT user's guide

bft_sub_image

Subtract one low-res image from high-res one.

USAGE: `[hi_res] = bft_sub_image(hi_res, lo_res, element, start_time)`

INPUT: `hi_res` High resolution RF image. One column per scan line.
`lo_res` Low resolution RF image. One column per scan line.
`element` Number of element, used to acquire the low resolution image.
`start_time` Arrival time of the first sample of the RF lines.

OUTPUT: `hi_res` - The high resolution image.

BFT user's guide

bft_sum_apodization

Create a summation apodization time line. This function is used in the case that the individual low resolution images must be weighted during the summation

USAGE: `bft_sum_apodization(xdc, times, values, line_no)`

INPUT: `xdc` Pointer to a transducer aperture.
`times` Time after which the associated apodization is valid.
`values` Apodization values. Matrix with one row for each time value and a number of columns equal to the number of physical elements in the aperture.
`line_no` Number of line. If skipped, `line_no` is assumed to be equal to '1'.

OUTPUT: None

BFT user's guide

bft_sum_images

Sum 2 low resolution images in 1 high resolution.

USAGE: `[hi_res] = bft_sum_images(image1, ele1, image2, ele2, time)`
 INPUT: *image1* Matrix with the RF data for the image. The number of columns corresponds to the number of lines
 ele1 Number of emitting element used to obtain the image.
 image2 Matrix with the RF data for the image. The number of columns corresponding to the number of lines
 ele2 Number of emitting element used to obtain the image.
 time The arrival time of the first samples. The two images must be aligned in time

BFT user's guide

bft_transducer

Create a new transducer definition. The transducer definition is necessary for the calculation of the delays.

USAGE: `xdc = bft_transducer(centers)`
 INPUT: *centers* Matrix with the coordinates of the centers of the elements. It has 3 columns (x,y,z) and a number of rows equal to the number of elements. The coordinates are specified in [m]
 OUTPUT: *xdc* Pointer to the memory location with the transducer definition. Do not alter this value !!!

Examples

3.1 Using Field II simulations

BFT user's guide

Phased array B-mode image

```

1  %PHASED_IMAGE Create phased array B-mode image with BFT.
2  % This script creates a B-mode PSF line by line. Each line is
3  % calculated using CALC_SCAT and CALC_SCAT_MULTY. The rf_data
4  % from CALC_SCAT_MULTY is passed to the beamforming toolbox,
5  % and in the end the results are compared.
6  %
7  % The function calls XDC_FOCUS, and BFT_FOCUS in order to set the
8  % the delays.
9
10 % VERSION 1.0, 29 Feb. 2000, Svetoslav Nikolov
11
12 f0 = 4e6;           % Central frequency           [Hz]
13 fs = 100e6;         % Sampling frequency          [Hz]
14 c = 1540;           % Speed of sound              [m/s]
15 no_elements = 64;   % Number of elements in the transducer
16
17 lambda = c / f0;     % Wavelength                 [m]
18 pitch = lambda / 2;  % Pitch - center-to-center   [m]
19 width = .95*pitch;   % Width of the element        [m]
20 kerf = pitch - width; % Inter-element spacing      [m]
21 height = 10/1000;    % Size in the Y direction    [m]
22
23
24 % Define the impulse response of the transducer
25 impulse_response = sin(2*pi*f0*(0:1/fs:2/f0));
26 impulse_response = impulse_response.*hanning(length(impulse_response));
27 excitation = impulse_response;
28

```

```

29  % Define the phantom
30
31  pht_pos = [0 0 20;
32            0 0 30;
33            0 0 40;
34            0 0 50;
35            0 0 60;
36            0 0 70;
37            0 0 80;
38            0 0 90;
39            ] / 1000;          % The position of the phantom
40  pht_amp = 20*ones(8,1);      % The amplitude of the back-scatter
41
42
43
44  % Define the focus
45  focus_r = [20;30;40;50;60;70;80;90] / 1000;
46  T = (focus_r-5/1000)/c *2;
47
48
49  % Initialize the program
50  field_init(0);
51  bft_init;
52
53
54  % Set some paramters
55  set_field('c', c);
56  bft_param('c', c);
57
58  set_field('fs', fs);
59  bft_param('fs', fs);
60
61
62
63  % Create some apertures.
64
65  xmt = xdc_linear_array(no_elements,width,height,kerf,1,1,[0 0 0]);
66  rcv = xdc_linear_array(no_elements,width,height,kerf,1,1,[0 0 0]);
67
68  xdc = bft_linear_array(no_elements, width, kerf);
69
70
71  % Set the impulse responses
72  xdc_impulse(rcv, impulse_response);
73  xdc_impulse(xmt, impulse_response);
74
75  xdc_excitation(xmt, excitation);
76
77
78
79  % Define and create the image
80  sector = 30 * pi / 180;
81  no_lines = 32;
82  d_theta = sector / (no_lines-1);

```

```

83  theta = -(no_lines-1) / 2 * d_theta;
84
85  Rmax = max(sqrt(pht_pos(:,1).^ 2 + pht_pos(:,2).^ 2 + pht_pos(:,3).^ 2)) + 15/1000;
86
87  no_rf_samples = ceil(2*Rmax/c * fs);
88  rf_line = zeros(no_rf_samples, 1);
89  bf_line = zeros(no_rf_samples, 1);
90
91  env_line = zeros(no_rf_samples, no_lines);
92  env_bf = zeros(no_rf_samples, no_lines);
93
94
95  xmt_r = (max(focus_r) + min(focus_r))/2;
96
97  for i = 1 : no_lines
98      rf_line(:) = 0;
99      disp(['Line no ' num2str(i)])
100
101      focus = [sin(theta)*focus_r, zeros(length(focus_r),1), cos(theta)*focus_r];
102      xmt_f = [sin(theta)*xmt_r, zeros(length(xmt_r),1), cos(theta)*xmt_r];
103      xdc_center_focus(xmt,[0 0 0])
104      xdc_center_focus(rcv,[0 0 0])
105      bft_center_focus([0 0 0]);
106
107      xdc_focus(xmt, 0, xmt_f);
108      xdc_focus(rcv, T, focus);
109      bft_focus(xdc, T, focus);
110
111      % Beamform with Field II
112      [rf_temp, t(i)] = calc_scatt(xmt,rcv, pht_pos, pht_amp);
113
114      % Beamform with BFT
115      xdc_focus_times(rcv, 0, zeros(1,no_elements));
116      [rf_data, start_t] = calc_scatt_multi(xmt,rcv, pht_pos, pht_amp);
117      rf_data = [zeros(300,no_elements); rf_data; zeros(300,no_elements)];
118
119      start_t = start_t - 300 / fs;
120      bf_temp = bft_beamform(start_t, rf_data);
121
122      start_sample = t(i)*fs; no_temp_samples = length(rf_temp);
123
124      rf_line(start_sample:start_sample+no_temp_samples-1) = rf_temp(1:no_temp_samples);
125      env_line(:,i) = abs(hilbert(rf_line(:)));
126
127      start_sample = floor(start_t*fs); no_temp_samples = length(bf_temp);
128      bf_line(start_sample:start_sample+no_temp_samples-1) = bf_temp(1:no_temp_samples);
129      env_bf(:,i) = abs(hilbert(bf_line(:)));
130      theta = theta + d_theta;
131
132  end
133
134  % Release the allocated memory
135
136  field_end

```



```

137 bft_end
138
139 env_line = env_line / max(max(abs(env_line)));
140 env_bf = env_bf / max(max(abs(env_bf)));
141
142 figure;
143 subplot(1,2,1)
144 imagesc([-sector/2 sector/2]*180/pi,[0 Rmax]*1000,20*log10(env_line + 0.001))
145 axis('image')
146 xlabel('Angle [deg]');
147 ylabel('Axial distance [mm]')
148 title('Beamformed by Field II ');
149
150 subplot(1,2,2)
151 imagesc([-sector/2 sector/2]*180/pi,[0 Rmax]*1000,20*log10(env_bf + 0.001));
152 title('Beamformed by BFT');
153 xlabel('Angle [deg]');
154 ylabel('Axial distance [mm]')
155 axis('image')
156
157 colorbar
158 colormap(gray)
159
160 clc
161 disp([' ' 10 10 10 10 ]);
162 disp([9 '*****']);
163 disp([9 '****']);
164 disp([9 '** The image beamformed by Field II is in "env_line" '**]);
165 disp([9 '** The image beamformed by BFT is in "env_bf" '**]);
166 disp([9 '****']);
167 disp([9 '*****']);
168 disp([' ' 10 10 ]);
169
170

```

BFT user's guide

Dynamic focusing

```

1  %PHASED_DYN_IMAGE Create a phased-array B-mode image, using the
2  %  commands for setting a dynamic focusing
3
4  %VERSION 1.0, 29 Feb 2000, Svetoslav Nikolov
5
6  f0 = 4e6;           % Central frequency           [Hz]
7  fs = 100e6;         % Sampling frequency         [Hz]
8  c = 1540;           % Speed of sound              [m/s]
9  B = .35;            % Relative bandwidth          [fraction]

```

```

10 no_elements = 64;          % Number of elements in the transducer
11
12 lambda = c / f0;           % Wavelength [m]
13 pitch = lambda / 2;        % Pitch - center-to-center [m]
14 width = .95*pitch;         % Width of the element [m]
15 kerf = pitch - width;      % Inter-element spacing [m]
16 height = 10/1000;         % Size in the Y direction [m]
17
18
19 % Define the impulse response of the transducer
20 impulse_response = sin(2*pi*f0*(0:1/fs:2/f0));
21 impulse_response = impulse_response.*hanning(length(impulse_response));
22 excitation = impulse_response;
23
24 % Define the phantom
25
26 pht_pos = [0 0 20;
27            0 0 30;
28            0 0 40;
29            0 0 50;
30            0 0 60;
31            0 0 70;
32            0 0 80;] / 1000; % The position of the phantom
33 pht_amp = 20*ones(7,1);    % The amplitude of the back-scatter
34
35 % Define the focus
36 focus_r = [20;30;40;50;60;70;80;90] / 1000;
37 T = (focus_r-5/1000)/c *2;
38
39 % Initialize the program
40 field_init(0);
41 bft_init;
42
43 % Set some paramters
44 set_field('c', c);
45 bft_param('c', c);
46
47 set_field('fs', fs);
48 bft_param('fs', fs);
49
50
51 % Create some apertures.
52
53 xmt = xdc_linear_array(no_elements,width,height,kerf,1,1,[0 0 0]);
54 rcv = xdc_linear_array(no_elements,width,height,kerf,1,1,[0 0 0]);
55
56 xdc = bft_linear_array(no_elements, width, kerf);
57
58
59 % Set the impulse responses
60 xdc_impulse(rcv, impulse_response);
61 xdc_impulse(xmt, impulse_response);
62
63 xdc_excitation(xmt, excitation);

```

```

64
65
66 % Set the apodization
67
68 xdc_apodization(xmt, 0, ones(1,no_elements))
69 xdc_apodization(rcv, 0, ones(1,no_elements))
70 bft_apodization(xdc, 0, ones(1,no_elements))
71
72 % Define and create the image
73 sector = 30 * pi / 180;
74 no_lines = 32;
75 d_theta = sector / (no_lines-1);
76 theta = -(no_lines-1) / 2 * d_theta;
77
78 Rmax = max(sqrt(pht_pos(:,1).^ 2 + pht_pos(:,2).^ 2 + pht_pos(:,3).^ 2)) + 15/1000;
79
80 no_rf_samples = ceil(2*Rmax/c * fs);
81 rf_line = zeros(no_rf_samples, 1);
82 bf_line = zeros(no_rf_samples, 1);
83
84 env_line = zeros(no_rf_samples, no_lines);
85 env_bf = zeros(no_rf_samples, no_lines);
86
87
88 xmt_r = (max(focus_r) + min(focus_r) )/2;
89 bf = cell(no_lines,1);
90 for i = 1 : no_lines
91     rf_line(:) = 0;
92     theta
93
94     xmt_f = [sin(theta)*xmt_r, zeros(length(xmt_r),1), cos(theta)*xmt_r];
95     xdc_center_focus(xmt,[0 0 0])
96     xdc_center_focus(rcv,[0 0 0])
97     bft_center_focus([0 0 0]);
98
99     xdc_focus(xmt, 0, xmt_f);
100    xdc_dynamic_focus(rcv, 0, theta, 0);
101
102 % Beamform with Field II
103 [rf_temp, t(i)] = calc_scatt(xmt,rcv, pht_pos, pht_amp);
104
105 % Beamform with BFT
106 bft_dynamic_focus(xdc, theta, 0)
107 xdc_focus_times(rcv, 0, zeros(1,no_elements));
108 [rf_data, start_t] = calc_scatt_multi(xmt,rcv, pht_pos, pht_amp);
109
110 rf_data = [zeros(300,no_elements); rf_data; zeros(300,no_elements)];
111
112 start_t = start_t - 300 / fs;
113 bf_temp = bft_beamform(start_t, rf_data);
114
115 start_sample = t(i)*fs; no_temp_samples = length(rf_temp);
116
117 rf_line(start_sample:start_sample+no_temp_samples-1) = rf_temp(1:no_temp_samples);

```

```

118     env_line(:,i) = abs(hilbert(rf_line(:)));
119
120     start_sample = floor(start_t*fs); no_temp_samples = length(bf_temp);
121     bfi = bf_temp;
122
123     bf_line(start_sample:start_sample+no_temp_samples-1) = bf_temp(1:no_temp_samples);
124     env_bf(:,i) = abs(hilbert(bf_line(:)));
125     theta = theta + d_theta;
126
127 end
128
129 % Release the allocated memory
130
131 field_end
132 bft_end
133 env_line = env_line / max(max(abs(env_line)));
134 env_bf = env_bf / max(max(abs(env_bf)));
135
136 figure;
137 subplot(1,2,1)
138 imagesc([-sector/2 sector/2]*180/pi,[0 Rmax]*1000,20*log10(env_line + 0.001))
139 axis('image')
140 xlabel('Angle [deg]');
141 ylabel('Axial distance [mm]')
142 title('Beamformed by Field II ');
143
144 subplot(1,2,2)
145 imagesc([-sector/2 sector/2]*180/pi,[0 Rmax]*1000,20*log10(env_bf + 0.001));
146 title('Beamformed by BFT');
147 xlabel('Angle [deg]');
148 ylabel('Axial distance [mm]')
149 axis('image')
150
151 colorbar
152 colormap(gray)
153
154 clc
155 disp([' ' 10 10 10 10 ]);
156 disp([9 '*****']);
157 disp([9 '*                               *']);
158 disp([9 '* The image beamformed by Field II is in "env_line" *']);
159 disp([9 '* The image beamformed by BFT is in "env_bf" *']);
160 disp([9 '*                               *']);
161 disp([9 '*****']);
162 disp([' ' 10 10 ]);
163
164

```

```
1  %SYNTHETIC Synthetic aperture beamforming with BFT
2  %
3
4
5  f0 = 4e6;           % Central frequency
6  fs = 100e6;         % Sampling frequency
7  c = 1540;           % Speed of sound
8  no_elements = 64;   % Number of elements in the transducer
9
10 lambda = c / f0;    % Wavelength
11 pitch = lambda / 2;  % Pitch - center-to-center
12 width = .95*pitch;   % Width of the element
13 kerf = pitch - width; % Inter-element spacing
14 height = 10/1000;    % Size in the Y direction
15
16
17 % Define the impulse response of the transducer
18 impulse_response = sin(3*pi*f0*(0:1/fs:2/f0));
19 impulse_response = impulse_response.*hanning(length(impulse_response));
20 excitation = sin(2*pi*f0*(0:1/fs:3/f0));
21
22 % Define the phantom
23
24 pht_pos = [0 0 40] / 1000; % The position of the phantom
25
26 [m n] = size(pht_pos);
27 pht_amp = 20*ones(m,1);    % The amplitude of the back-scatter
28
29
30 % Define the focus
31
32 focus_r = [1:max(sqrt(pht_pos(:,1).^2 + pht_pos(:,2).^2 + pht_pos(:,3).^2))*1000]' / 1000;
33 T = (focus_r-.5/1000)/c *2;
34
35
36
37 % Initialize the program
38 field_init(0);
39 bft_init;
40
41 % Set some paramters
42 set_field('c', c);
43 bft_param('c', c);
44
45 set_field('fs', fs);
46 bft_param('fs', fs);
```

```

47
48
49 % Create some apertures.
50
51 xmt = xdc_linear_array(no_elements,width,height,kerf,1,1,[0 0 0]);
52 rcv = xdc_linear_array(no_elements,width,height,kerf,1,1,[0 0 0]);
53
54 xdc = bft_linear_array(no_elements, width, kerf);
55
56
57 % Set the impulse responses
58 xdc_impulse(rcv, impulse_response);
59 xdc_impulse(xmt, impulse_response);
60
61 xdc_excitation(xmt, excitation);
62
63
64 % Define and create the image
65 sector = 60 * pi / 180;
66 no_lines = 64;
67 d_theta = sector / (no_lines-1);
68 theta = -(no_lines-1) / 2 * d_theta;
69
70 % Set the delays for one whole image
71 %
72 bft_no_lines(no_lines);
73 for i = 1 : no_lines
74     bft_apodization(xdc,0,hanning(no_elements)',i);
75     % bft_sum_apodization(xdc,0,ones(1,no_elements),i);
76     focus = [sin(theta)*focus_r, zeros(length(focus_r),1), cos(theta)*focus_r];
77     bft_center_focus([0 0 0],i);
78     bft_focus(xdc, T, focus,i);
79     theta = theta + d_theta;
80 end
81
82
83 %
84 % Allocate memory for the image
85 %
86 Rmax = max(sqrt(pht_pos(:,1).^ 2 + pht_pos(:,2).^ 2 + pht_pos(:,3).^ 2)) + 10/1000;
87 Rmin = min(sqrt(pht_pos(:,1).^ 2 + pht_pos(:,2).^ 2 + pht_pos(:,3).^ 2)) - 10/1000;
88 if (Rmin < 0) Rmin = 0; end;
89 Tmin = 2*Rmin / c; Tmax = 2*Rmax / c;
90 Smin = floor(Tmin * fs); Smax = ceil(Tmax * fs);
91
92 no_rf_samples = Smax - Smin + 1;
93
94 bf_image = zeros(no_rf_samples, no_lines);
95
96 %
97 % Make one low-resolution image at a time and sum them
98 %
99
100 xdc_focus_times(xmt,0,zeros(1,no_elements));

```

```

101 xdc_focus_times(rcv,0,zeros(1,no_elements));
102 for emission_no = 1:no_elements
103     disp(['emission no: ' num2str(emission_no)]);
104     xdc_apodization(xmt,0,[zeros(1,emission_no-1) 1 zeros(1, no_elements - emission_no)]);
105
106     [scat, start_time] = calc_scat_multi (xmt, rcv, pht_pos, pht_amp);
107
108     start_sample = floor(start_time * fs + 0.5);
109     end_sample = start_sample + max(size(scat))-1;
110
111     scat = [zeros(start_sample - Smin, no_elements); scat; zeros(Smax -
end_sample,no_elements)];
112
113     start_time = Tmin;
114     beamformed = bft_beamform(start_time,scat);
115     bf_image = bft_add_image(bf_image, beamformed, emission_no, start_time);
116 end
117
118
119
120 % Release the allocated memory
121
122 field_end
123 bft_end
124
125
126
127 %
128 % Display the image
129 %
130
131 bf_image = abs(hilbert(bf_image)); % Envelope detection
132 bf_image = bf_image / max(max(bf_image));
133
134 figure;
135 imagesc([-sector/2 sector/2]*180/pi,[Rmin Rmax]*1000,20*log10(bf_image + 0.001))
136 axis('image')
137 xlabel('Angle [deg]');
138 ylabel('Axial distance [mm]')
139 title('Beamformed by BFT ');
140 clc
141 disp([' ' 10 10 10 10 ]);
142 disp([9 '*****']);
143 disp([9 '* * * * *']);
144 disp([9 '* The image beamformed by BFT is in "bf_image" *']);
145 disp([9 '* * * * *']);
146 disp([9 '*****']);
147 disp([' ' 10 10 ]);
148

```