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

CHAPTER

ONE

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 points[1], 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.

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

CHAPTER

TWO

Description of Matlab Functions

2.1 Functions list

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

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	

BFT user's guide	**bft_center_focus**

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	

BFT user's guide	**bft_dynamic_focus**

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_zx	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	

BFT user's guide	**bft_end**

Release all resources allocated by the beamforming toolbox.

USAGE:	`bft_end`
INPUT:	None
OUTPUT:	None

BFT user's guide	**bft_focus**

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 elelements 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 !!!

CHAPTER

THREE

Examples

3.1 Using Field II simulations

BFT user's guide	**Phased array B-mode image**

```
%PHASED_IMAGE Create phased array B-mode image with BFT.
%   This script creates a B-mode PSF line by line. Each line is
%   calculated using CALC_SCAT and CALC_SCAT_MULTY. The rf_data
%   from CALC_SCAT_MULTI is passed to  the beamforming toolbox,
%   and in the end the results are compared.
%
%   The function calls XDC_FOCUS, and BFT_FOCUS in order to set the
%   the delays.

% VERSION 1.0, 29 Feb. 2000, Svetoslav Nikolov

f0 = 4e6;                  %  Central frequency            [Hz]
fs = 100e6;                %  Sampling frequency            [Hz]
c = 1540;                  %  Speed of sound               [m/s]
no_elements = 64;          %  Number of elements in the transducer

lambda = c / f0;           % Wavelength                   [m]
pitch = lambda / 2;        % Pitch - center-to-center       [m]
width = .95*pitch;         % Width of the element           [m]
kerf = pitch - width;      % Inter-element spacing          [m]
height = 10/1000;          % Size in the Y direction        [m]


%  Define the impulse response of the transducer
impulse_response = sin(2*pi*f0*(0:1/fs:2/f0));
impulse_response = impulse_response.*hanning(length(impulse_response))';
excitation = impulse_response;

```

```
% Define the phantom

pht_pos = [0 0 20;
           0 0 30;
           0 0 40;
           0 0 50;
           0 0 60;
           0 0 70;
           0 0 80;
           0 0 90;
           ] / 1000;         % The position of the phantom
pht_amp = 20*ones(8,1);      % The amplitude of the back-scatter



% Define the focus
focus_r = [20;30;40;50;60;70;80;90] / 1000;
T = (focus_r-5/1000)/c *2;


% Initialize the program
field_init(0);
bft_init;


% Set some paramters
set_field(’c’, c);
bft_param(’c’, c);

set_field(’fs’, fs);
bft_param(’fs’, fs);



% Create some apertures.

xmt = xdc_linear_array(no_elements,width,height,kerf,1,1,[0 0 0]);
rcv = xdc_linear_array(no_elements,width,height,kerf,1,1,[0 0 0]);

xdc = bft_linear_array(no_elements, width, kerf);


% Set the impulse responses
xdc_impulse(rcv, impulse_response);
xdc_impulse(xmt, impulse_response);

xdc_excitation(xmt, excitation);



% Define and create the image
sector = 30 * pi / 180;
no_lines = 32;
d_theta = sector / (no_lines-1);
```

```
theta = -(no_lines-1) / 2 * d_theta;

Rmax = max(sqrt(pht_pos(:,1).^ 2 + pht_pos(:,2).^ 2  + pht_pos(:,3).^ 2)) + 15/1000;

no_rf_samples = ceil(2*Rmax/c * fs);
rf_line = zeros(no_rf_samples, 1);
bf_line = zeros(no_rf_samples, 1);

env_line = zeros(no_rf_samples, no_lines);
env_bf =  zeros(no_rf_samples, no_lines);


xmt_r = (max(focus_r) + min(focus_r) )/2;

for i = 1 : no_lines
  rf_line(:) = 0;
  disp(['Line no ' num2str(i)])

  focus = [sin(theta)*focus_r, zeros(length(focus_r),1), cos(theta)*focus_r];
  xmt_f = [sin(theta)*xmt_r, zeros(length(xmt_r),1), cos(theta)*xmt_r];
  xdc_center_focus(xmt,[0 0 0])
  xdc_center_focus(rcv,[0 0 0])
  bft_center_focus([0 0 0]);

  xdc_focus(xmt, 0, xmt_f);
  xdc_focus(rcv, T, focus);
  bft_focus(xdc, T, focus);

  % Beamform with Field II
  [rf_temp, t(i)] = calc_scat(xmt,rcv, pht_pos, pht_amp);

  % Beamform with BFT
  xdc_focus_times(rcv, 0, zeros(1,no_elements));
  [rf_data, start_t] = calc_scat_multi(xmt,rcv, pht_pos, pht_amp);
  rf_data = [zeros(300,no_elements); rf_data; zeros(300,no_elements)];

  start_t = start_t - 300  / fs;
  bf_temp = bft_beamform(start_t, rf_data);

  start_sample = t(i)*fs; no_temp_samples = length(rf_temp);

  rf_line(start_sample:start_sample+no_temp_samples-1) = rf_temp(1:no_temp_samples);
  env_line(:,i) = abs(hilbert(rf_line(:)));

  start_sample = floor(start_t*fs); no_temp_samples = length(bf_temp);
  bf_line(start_sample:start_sample+no_temp_samples-1) = bf_temp(1:no_temp_samples);
  env_bf(:,i) = abs(hilbert(bf_line(:)));
  theta = theta + d_theta;

end

% Release the allocated memory

field_end
```

```
bft_end

env_line = env_line / max(max(abs(env_line)));
env_bf = env_bf / max(max(abs(env_bf)));

figure;
subplot(1,2,1)
imagesc([-sector/2 sector/2]*180/pi,[0 Rmax]*1000,20*log10(env_line + 0.001))
axis('image')
xlabel('Angle [deg]');
ylabel('Axial distance [mm]')
title('Beamformed by Field II ');

subplot(1,2,2)
imagesc([-sector/2 sector/2]*180/pi,[0 Rmax]*1000,20*log10(env_bf + 0.001));
title('Beamformed by BFT');
xlabel('Angle [deg]');
ylabel('Axial distance [mm]')
axis('image')

colorbar
colormap(gray)

clc
disp([' ' 10 10 10 10 ]);
disp([9 '*****************************************************']);
disp([9 '*                                                   *']);
disp([9 '* The image beamformed by Field II is in "env_line" *']);
disp([9 '* The image beamformed by BFT is in "env_bf"        *']);
disp([9 '*                                                   *'])
disp([9 '*****************************************************']);
disp([' ' 10 10 ]);


```

BFT user's guide	**Dynamic focusing**

```
%PHASED_DYN_IMAGE Create a phased-array B-mode image, using the
%    commands for setting a dynamic focusing

%VERSION 1.0, 29 Feb 2000, Svetoslav Nikolov

f0 = 4e6;                 %  Central frequency          [Hz]
fs = 100e6;               %  Sampling frequency          [Hz]
c = 1540;                 %  Speed of sound             [m/s]
B = .35;                  %  Relative bandwith          [fraction]
```

```
no_elements = 64;        % Number of elements in the transducer

lambda = c / f0;         % Wavelength                    [m]
pitch = lambda / 2;      % Pitch - center-to-center      [m]
width = .95*pitch;       % Width of the element          [m]
kerf = pitch - width;    % Inter-element spacing         [m]
height = 10/1000;        % Size in the Y direction       [m]


% Define the impulse response of the transducer
impulse_response = sin(2*pi*f0*(0:1/fs:2/f0));
impulse_response = impulse_response.*hanning(length(impulse_response))';
excitation = impulse_response;

% Define the phantom

pht_pos = [0 0 20;
           0 0 30;
           0 0 40;
           0 0 50;
           0 0 60;
           0 0 70;
           0 0 80;] / 1000;        % The position of the phantom
pht_amp = 20*ones(7,1);      % The amplitude of the back-scatter

% Define the focus
focus_r = [20;30;40;50;60;70;80;90] / 1000;
T = (focus_r-5/1000)/c *2;

% Initialize the program
field_init(0);
bft_init;

% Set some paramters
set_field('c', c);
bft_param('c', c);

set_field('fs', fs);
bft_param('fs', fs);


% Create some apertures.

xmt = xdc_linear_array(no_elements,width,height,kerf,1,1,[0 0 0]);
rcv = xdc_linear_array(no_elements,width,height,kerf,1,1,[0 0 0]);

xdc = bft_linear_array(no_elements, width, kerf);


% Set the impulse responses
xdc_impulse(rcv, impulse_response);
xdc_impulse(xmt, impulse_response);

xdc_excitation(xmt, excitation);
```

```
% Set the apodization

xdc_apodization(xmt, 0, ones(1,no_elements))
xdc_apodization(rcv, 0, ones(1,no_elements))
bft_apodization(xdc, 0 , ones(1,no_elements))

% Define and create the image
sector = 30 * pi / 180;
no_lines = 32;
d_theta = sector / (no_lines-1);
theta = -(no_lines-1) / 2 * d_theta;

Rmax = max(sqrt(pht_pos(:,1).^ 2 + pht_pos(:,2).^ 2  + pht_pos(:,3).^ 2)) + 15/1000;

no_rf_samples = ceil(2*Rmax/c * fs);
rf_line = zeros(no_rf_samples, 1);
bf_line = zeros(no_rf_samples, 1);

env_line = zeros(no_rf_samples, no_lines);
env_bf =  zeros(no_rf_samples, no_lines);


xmt_r = (max(focus_r) + min(focus_r) )/2;
bf = cell(no_lines,1);
for i = 1 : no_lines
  rf_line(:) = 0;
  theta

  xmt_f = [sin(theta)*xmt_r, zeros(length(xmt_r),1), cos(theta)*xmt_r];
  xdc_center_focus(xmt,[0 0 0])
  xdc_center_focus(rcv,[0 0 0])
  bft_center_focus([0 0 0]);

  xdc_focus(xmt, 0, xmt_f);
  xdc_dynamic_focus(rcv, 0, theta, 0);

  % Beamform with Field II
  [rf_temp, t(i)] = calc_scat(xmt,rcv, pht_pos, pht_amp);

  % Beamform with BFT
  bft_dynamic_focus(xdc, theta, 0)
  xdc_focus_times(rcv, 0, zeros(1,no_elements));
  [rf_data, start_t] = calc_scat_multi(xmt,rcv, pht_pos, pht_amp);

  rf_data = [zeros(300,no_elements); rf_data; zeros(300,no_elements)];

  start_t = start_t - 300  / fs;
  bf_temp = bft_beamform(start_t, rf_data);

  start_sample = t(i)*fs; no_temp_samples = length(rf_temp);

  rf_line(start_sample:start_sample+no_temp_samples-1) = rf_temp(1:no_temp_samples);
```

```
  env_line(:,i) = abs(hilbert(rf_line(:)));

  start_sample = floor(start_t*fs); no_temp_samples = length(bf_temp);
  bfi = bf_temp;

  bf_line(start_sample:start_sample+no_temp_samples-1) = bf_temp(1:no_temp_samples);
  env_bf(:,i) = abs(hilbert(bf_line(:)));
  theta = theta + d_theta;

end

% Release the allocated memory

field_end
bft_end
env_line = env_line / max(max(abs(env_line)));
env_bf = env_bf / max(max(abs(env_bf)));

figure;
subplot(1,2,1)
imagesc([-sector/2 sector/2]*180/pi,[0 Rmax]*1000,20*log10(env_line + 0.001))
axis('image')
xlabel('Angle [deg]');
ylabel('Axial distance [mm]')
title('Beamformed by Field II ');

subplot(1,2,2)
imagesc([-sector/2 sector/2]*180/pi,[0 Rmax]*1000,20*log10(env_bf + 0.001));
title('Beamformed by BFT');
xlabel('Angle [deg]');
ylabel('Axial distance [mm]')
axis('image')

colorbar
colormap(gray)

clc
disp([' ' 10 10 10 10 ]);
disp([9 '*****************************************************']);
disp([9 '*                                                   *']);
disp([9 '* The image beamformed by Field II is in "env_line" *']);
disp([9 '* The image beamformed by BFT is in "env_bf"        *']);
disp([9 '*                                                   *'])
disp([9 '*****************************************************']);
disp([' ' 10 10 ]);
```

```
%SYNTHETIC Synthetic aperture beamforming with BFT
%


f0 = 4e6;               % Central frequency
fs = 100e6;             % Sampling frequency
c = 1540;               % Speed of sound
no_elements = 64;       % Number of elements in the transducer

lambda = c / f0;        % Wavelength
pitch = lambda / 2;     % Pitch - center-to-center
width = .95*pitch;      % Width of the element
kerf = pitch - width;   % Inter-element spacing
height = 10/1000;       % Size in the Y direction


% Define the impulse response of the transducer
impulse_response = sin(3*pi*f0*(0:1/fs:2/f0));
impulse_response = impulse_response.*hanning(length(impulse_response))';
excitation = sin(2*pi*f0*(0:1/fs:3/f0));

% Define the phantom

pht_pos = [0 0 40] / 1000;     % The position of the phantom

[m n] = size(pht_pos);
pht_amp = 20*ones(m,1);        % The amplitude of the back-scatter


% Define the focus

focus_r = [1:max(sqrt(pht_pos(:,1).^2 + pht_pos(:,2).^2 + pht_pos(:,3).^2))*1000]' / 1000;
T = (focus_r-.5/1000)/c *2;



% Initialize the program
field_init(0);
bft_init;

% Set some paramters
set_field('c', c);
bft_param('c', c);

set_field('fs', fs);
bft_param('fs', fs);
```

```


% Create some apertures.

xmt = xdc_linear_array(no_elements,width,height,kerf,1,1,[0 0 0]);
rcv = xdc_linear_array(no_elements,width,height,kerf,1,1,[0 0 0]);

xdc = bft_linear_array(no_elements, width, kerf);


% Set the impulse responses
xdc_impulse(rcv, impulse_response);
xdc_impulse(xmt, impulse_response);

xdc_excitation(xmt, excitation);


%  Define and create the image
sector = 60 * pi / 180;
no_lines = 64;
d_theta = sector / (no_lines-1);
theta = -(no_lines-1) / 2 * d_theta;

%  Set the delays for one whole image
%
bft_no_lines(no_lines);
for i = 1 : no_lines
  bft_apodization(xdc,0,hanning(no_elements)',i);
%  bft_sum_apodization(xdc,0,ones(1,no_elements),i);
  focus = [sin(theta)*focus_r, zeros(length(focus_r),1), cos(theta)*focus_r];
  bft_center_focus([0 0 0],i);
  bft_focus(xdc, T, focus,i);
  theta = theta + d_theta;
end


%
%  Allocate memory for the image
%
Rmax = max(sqrt(pht_pos(:,1).^ 2 + pht_pos(:,2).^ 2  + pht_pos(:,3).^ 2)) + 10/1000;
Rmin = min(sqrt(pht_pos(:,1).^ 2 + pht_pos(:,2).^ 2  + pht_pos(:,3).^ 2)) - 10/1000;
if (Rmin < 0) Rmin = 0; end;
Tmin = 2*Rmin / c; Tmax = 2*Rmax / c;
Smin = floor(Tmin * fs); Smax = ceil(Tmax * fs);

no_rf_samples = Smax - Smin + 1;

bf_image = zeros(no_rf_samples, no_lines);

%
% Make one low-resolution image at a time and sum them
%

xdc_focus_times(xmt,0,zeros(1,no_elements));
```

```
xdc_focus_times(rcv,0,zeros(1,no_elements));
for emission_no = 1:no_elements
  disp(['emission no: ' num2str(emission_no)]);
  xdc_apodization(xmt,0,[zeros(1,emission_no-1) 1 zeros(1, no_elements - emission_no)]);

  [scat, start_time] = calc_scat_multi (xmt, rcv, pht_pos, pht_amp);

  start_sample = floor(start_time * fs + 0.5);
  end_sample = start_sample + max(size(scat))-1;

  scat = [zeros(start_sample - Smin, no_elements); scat; zeros(Smax -
end_sample,no_elements)];

  start_time = Tmin;
  beamformed = bft_beamform(start_time,scat);
  bf_image = bft_add_image(bf_image, beamformed, emission_no, start_time);
end



% Release the allocated memory

field_end
bft_end



%
% Dispplay the image
%

bf_image = abs(hilbert(bf_image));                 % Envelope detection
bf_image = bf_image / max(max(bf_image));

figure;
imagesc([-sector/2 sector/2]*180/pi,[Rmin Rmax]*1000,20*log10(bf_image + 0.001))
axis('image')
xlabel('Angle [deg]');
ylabel('Axial distance [mm]')
title('Beamformed by BFT ');
clc
disp([' ' 10 10 10 10 ]);
disp([9 '****************************************************']);
disp([9 '*                                                  *']);
disp([9 '* The image beamformed by BFT is in "bf_image"     *']);
disp([9 '*                                                  *'])
disp([9 '****************************************************']);
disp([' ' 10 10 ]);
```