URI/Field-II Interface Tools
DRAFT

Duke University

May 17, 2004

Contents

Chapter 1

Introduction

URI/Field is a software tool that simplifies simulation of the Siemens Antares scanner. URI/Field is an adjunct to the Siemens Antares Ultrasound Research Interface (URI) software option. The URI allows beam-summed RF data and associated beamforming parameters (header data) to be recorded for off-line processing. URI/Field performs matched simulations based on this header data. URI/Field is implemented entirely in Matlab and thus runs on a wide range of computer systems.

URI/Field relies on Jensens Field II software[1] to perform the acoustic simulations by the spatial-impulse response method. URI/Field provides a set of Matlab functions that automatically configure Field II to simulate apertures matched to transducer and scanner settings indicated in the URI header. Transmit and receive beams are specified which are steered and walked across the aperture as in the physical system, as opposed to the more common practice of simulating scanning by phantom translation. Functions for scan simulation, beam intensity estimation, envelope detection and scan conversion are included in the URI/Field package and may be called from the Matlab command line as desired.

Simulation data is divided into three logical structures based on its nature. A probe data structure contains fixed, transducer-specific information geometric and impulse-response data. A beamset data structure contains information related to beamformation parameters beam direction, origin, apodization and focal characteristics. Both of these structures may be generated automatically from a URI data file, or may be manually created to

[1] Available from http://www.es.oersted.dtu.dk/staff/jaj/field/

simulate arbitrary systems. The third structure, phantom, contains positions and amplitudes of scatterers to be imaged. Commands for creating simple wire target and lesion phantoms are included.

A graphical user interface (GUI) is provided to facilitate common tasks. Using the GUI, URI files may be loaded and apertures created automatically. Scans are simulated and beamplots calculated with simple menu commands. Graphical probe, beamset, and phantom editors are included for the creation or modification of the related data structure. Using these editors, modifications to a baseline scanner configuration may be simulated easily, or completely user-defined systems may be simulated. The GUI allows simulations to be launched as separate processes on the local machine, or as batch jobs on a remote machine. The GUI provides the advantage of reducing common simulation tasks to simple point-and-click operations. At the same time, an effort has been made to provide as much flexibility as possible with the GUI.

URI/Field provides tools for simplifying simulation of ultrasound systems, particularly the Siemens Antares equipped with the Siemens URI software. Powerful commands make it possible to describe an ultrasound system at a relatively high level and perform common simulation tasks easily. GUI editors simplify the inspection and modification of URI derived parameters and allow the specification of arbitrary scanners.

Chapter 2

Getting Started

Here's how to set up the URI/Field-II toolset and perform a trial simulation.

1. Copy the URI/Field-II tools into a convenient directory.
2. Launch Matlab
3. Ensure that the Matlab path includes the Field-II, URI-OPT, and URI/Field-II directories, using `path` and `addpath` as needed.
4. Run `scan_demo`
5. Select a URI datafile in the popup window
6. Simulation runs.

2.1 scan_demo.m

`scan_demo.m` is a sample Field II simulation using the URI/Field II tools.

```
% scan_demo.m demonstrates use of URI-Field II tools
% to simulate scanning
%

% Start Field II
%
field_init(0);
```

```
% Read URI file
%
[rf,head]=readFile;
pause(0);

% Extract probe and beam sequence data from header
%
[probe,beamset]=uf_parameters(head);

% create transmit and receive apertures
%
[tx,rx]=uf_make_xdc(probe);

% create a phantom
%
phantom=uf_phantom_grid(0.005,0.03,0.005,0,0.03,0.005);

% Pick the beamset you want to simulate
%
n_set=1;

% Select each vector in the beamset and calculate echo signal
%
for n_vector=1:beamset(n_set).no_beams,
        disp(sprintf('Processing Vector %d of %d',n_vector, ...
                beamset(n_set).no_beams));
        uf_set_beam(tx,rx,probe,beamset,n_set,n_vector);
        [v,t1]=calc_scat(tx,rx,phantom.position,phantom.amplitude);
        rfdata(1:length(v),n_vector)=v;
        start_times(n_vector)=t1;
        end;

% Create rf with equal t0's from rfdata
%
[rf,t0]=uf_time_eq(rfdata,start_times,probe.field_sample_freq);
td=length(probe.impulse_response)*2/probe.field_sample_freq;

% Close Field
```

```
%
field_end;

% Create envelope-detected, log-compressed dataset
%
envdb=uf_envelope(rf);

% Scan convert and create axes
%
[sc_img,ax,lat]=uf_scan_convert(envdb,probe,beamset(n_set),t0-td);

% Display the image
%
imagesc(lat,ax,sc_img,[-40 0]);axis image; colormap(gray);
```

2.2 Explaination of scan_demo.m

Stepping through this sample code is a good way to see how the URI/Field II tools work to perform a simulation.

The first command, texttfield_init(0); starts Field II. The URI/Field II tools rely on Field II to do the acoustic simulations. The zero argument suppresses the Field II startup graphic.

The next command block uses the URI-OPT [1] function `readFile` to load a URI data set into Matlab. The data is given in two parts, the rf echo data collected by the Antares, and the header data created by the scanner, describing the conditions under which each rf vector was acquired. It is this header data that allows us to describe the ultrasound system to Field II and perform simulations. The `pause(0)` is a zero-length pause that lets the file selection window close before proceeding.

The next command `[probe,beamset]=uf_parameters(head)` creates two data structures defining the ultrasound system. The first, *probe*, contains geometrical, impulse-response, and sampling frequency data about the transducer being simulated. The data in *probe* does not change over the course of a simulation; *probe* contains the fixed properties of the transducer.

[1] The URI-OPT toolbox is provided by the University of California, Davis, and is not explained in detail here. Brief descriptions of URI-OPT functions are given where necessary

The second, *beamset*, contains scanning parameter information, including the type of scan (e.g b-mode), pulse length, beam direction, and so on. The data in *beamset* describes the dynamic properties of the scan. In general *probe* describes fixed, physical properties of the system, while *beamset* describes aspects of the system controlled by the beamformer.

After extracting the system information, a call to `uf_make_xdc` returns handles to two Field II apertures, one transmit and one receive. The apertures are physically identical, but Field II uses two distinct handles to keep transmit and receive functions separate. All the necessary parameters for aperture creation are stored in *probe*.

Next, a simple point-target grid phantom with 5mm point spacing is created using the `uf_phantom_grid` function.

Each beamset may contain data for up to eight independent scan types, so we must select one to simulate. Here we choose n_set=1. n_set will be used as an index into *beamset* to select the scan to simulate.

The next block of commands performs the actual acoustic simulation. A `for` loops cycles through each of the beams in the selected beamset. There are `beamset(n_set).no_beams` many beams (vectors, a-lines) to simulate to create an image as the scanner would. For each beam in the set, `uf_set_beam` is used to select the particular vector to be simulated. `uf_set_beam` sets the (dynamic) focus, (dynamic) apodization, excitation pulse, beam origin and beam direction for the selected vector. These properties are set using data in the *probe* and *beamset* structures, and the corresponding Field II commands. The RF echo signal is calculated using the Field II command `calc_scat`. The echo signals are stored column-wise in the matrix *rfdata* and the time corresponding to the first sample of each echo signal is stored in *start_times*.

In general, each RF vector will have a different start time. The function `uf_time_eq` is used to shift the RF vectors into alignment with one another, adding leading zero-valued samples as necessary. `uf_time_eq` returns a matrix of aligned RF data and the start time of the aligned data in the scalar $t0$. The following line calculates the value td, which is the offset between zero time and the peak of the pulse, due to the convolution operations used in calculating the echo.

After shutting down Field II, a call to `uf_envelope` is used to envelope detect and logarithmically compress the echo data in preparation for scan conversion and display. The envelope of the signal is calculated from the magnitude of the Hilbert transform of the echo. The magnitude is scaled so all values are between 0 and 1, and then converted to a decibel scale,

$20\log_{10}(M)$. The peak value is 0dB.

The echo data is acquired with a variety of scan geometries, and the function `uf_scan_convert` is supplied to convert data from the scan coordinate system to rectilinear coordinates for display. Probe type (linear, phased, curvilinear) is taken from the *probe* data structure to determine the basic scan geometry, and information about beam spacing, position and direction is taken from the *beamset* structure. Note that the particular beamset being scan converted is specified, *beamset(n_set)*. The last argument is the start time of the data set to provide proper offset from the origin. In addition to returning the scan-converted data *out*, the axial and lateral coordinates of the image pixels are returned in the vectors *ax* and *lat* to allow proper scaling of the image.

The scan-converted image is displayed using Matlab's `imagesc` function. The axial and lateral image dimensions are used in conjunction with the `axis image` command to ensure the image is displayed with the proper aspect ratio. The grayscale selection vector is also supplied, in this case limiting the dynamic range of the image to -40 to 0 dB. This grayscale clipping is necessary to prevent a low-contrast image.

Chapter 3

Details

3.1 Sampling frequencies

The URI (scanner) RF data sample rate is 40MHz. This is a rather low sample rate for Field. By default, Field simulations are performed at a 100MHz sampling frequency.

The assumed sample frequency is stored in the *probe* data structure, as *probe.field_sample_freq*. Likewise, the URI sample frequency is given by *probe.uri_sample_freq*.

It may be that a different sample frequency is required for a particular simulation. The sample frequency in the *probe* structure may be set when it is created by `uf_parameters` by adding the optional argument *fs* to the argument list. For example, to set a 200MHz sampling frequency, the command would be `[probe,beamset]=uf_parameters(head,200e6)`.

This sets *probe.field_sample_freq* to 200×10^6Hz. In addition, it uses this sampling frequency when calculating the impulse response and transducer excitation signals to be used. It is important, therefore, that sampling frequencies other than the default be specified when `uf_parameters` is called.

The value in *probe.field_sample_freq* does not take effect (i.e., it's not passed to Field II) until the next call to `uf_make_xdc`. Field II supports only one sample frequency at a time.

The simulated signal may then be converted to the URI (scanner) sample frequency with the Matlab `resample` command, e.g.

```
RF40MHz = resample(RF100MHz,40,100);
```

Chapter 4

Description of Functions

All functions in the URI-Field II toolbox are prefixed by `uf_` to help keep their names from interfering with other functions and variables.

4.1 uf_parameters

uf_parameters.m Extracts transducer and beam-sequence information from a URI header. `uf_parameters` calls on probe description files located in the `probes` directory to supply geometry and impulse response information about the transducer specified in the URI header. See Chapter 5 for a description of the probe files.

```
[probe,beamset]=uf_parameters(head{,fs});
```

where

probe is a probe data structure, containing a geometric description of the transducer, as well as impulse response and sample frequency data.

beamset is a beamset data structure, containing information about the pulse sequences indicated in the URI header, including the number of sets, number of beams in each set, and the focus, pulse length, direction and apodization for each pulse in the sequence.

head is the URI header information returned by a call to readFile in the URI-OPT toolbox, e.g. `[rf,head]=readFile;`

fs is an optional argument used to specify the sampling frequency for the simulation. This is 100MHz by default. Values lower than 100MHz are not recommended. Note that setting the value of fs here only sets the value of probe.field_sampling_freq — Field becomes aware of this value when uf_make_xdc is called.

Side effects: None.

4.2 uf_make_xdc

uf_make_xdc.m Returns Matlab handles (pointers) to transmit and receive apertures based on the URI header transducer information.

```
[tx,rx]=uf_make_xdc(probe);
```

where

tx, rx — pointers to the transmit and receive apertures created by `uf_make_xdc`. These pointers are identical to those created by any of the FieldÎI transducer creation functions, e.g. `xdc_linear_array`.

probe is a probe data structure created by `uf_parameters` or other means.

Side effects:

- Sets Field II sampling frequency to the value in the **probe** structure.
- Sets impulse response of transmit and receive apertures to the values specified in the **probe** structure.
- Sets elevation focus of the transmit and receive apertures.

4.3 uf_set_beam

uf_set_beam.m Sets transmit and receive apertures to simulate a particular beam (vector) from the URI header beam sequences.

```
uf_set_beam(tx,rx,probe,beamset,n_set,n_vector);
```

where

tx, rx pointers to the transmit and receive apertures, created by uf_make_xdc.

probe is a probe data structure created by uf_parameters.

beamset is a beam sequence data structure created by uf_parameters.

n_set is an integer specifying the beam sequence of interest. `length(beamset)` will give the number of beam sequences described by a particular beam sequence structure, and will be between 1 and 8.

n_vector is an integer specifying the particular beam (vector) within a beam sequence to simulate.

Side effects:

- Sets focus, apodization (both possibly dynamic for receive aperture), beam origin, and excitation signal for the specified transmit and receive apertures.

4.4 uf_phantom_grid

uf_phantom_grid.m Creates a phantom data structure describing a wire-grid phantom for simulation. All dimensions are in meters, per the Field II convention.

```
[p]=uf_phantom_struct(AxialMin,AxialSpan,AxialSpacing,LateralOffset,LateralSpan,
```

where

AxialMin is the distance (m) from the transducer to the nearest point in the axial direction.

AxialSpan is the range axial distance (m) from the closest to the most distant scatterer

AxialSpacing is the seperation (m) between points in the axial direction.

LateralOffest is lateral distance (m) between the transducer center and the phantom center.

LateralSpan is the width of the phantom (m).

LateralSpacing is the separation (m) between points in the lateral direction.

Side effects: none.

4.5 uf_scan_convert

Takes a data set created by Field and resamples it onto a raster grid with the correct geometry. See scan_demo for an example of its use.

```
[out,ax,lat] = uf_scan_convert(in,probe,beamset,t0);
```

where

out is the scan-converted image.

ax is a vector of z or axial coordinates

lat is a vector of x or lateral coordinates.

in is the data to be scan-converted.

probe is the probe data structure used to create the data.

beamset is the *single* beamset used to create the data. If there are multiple sets in a beamset, one must be specified, e.g. beamset(2).

t0 is a scalar containing the start time of the echoes.

Side effects: none.

4.6 uf_time_eq

Creates a columnwise matrix of RF echo data with equal start times.

```
[out,st]=uf_time_eq(in,start_times,fs);
```

where

out is the matrix aligned echo data.

st is a scalar containing the start time

in is the data to be adjusted

start_times is a vector containing the start time of each column of *in*

fs is the sampling frequency of the data, usually it's probe.field_sample_freq.

Side effects: none.

4.7 uf_envelope

Performs envelope detection and log compression of RF data.

```
[out]=uf_envelope(in);
```

where

out is the envelope detected RF data on a decibel scale. Maximum value is 0dB.

in is a matrix of RF echo data to be processed, with one vector per column.

Side effects: none.

4.8 uf_beam_inten

Calculates ultrasound intensity at specified points in space.

```
[out]=uf_beam_inten(tx,x,y,z);
```

where

out is intensity of the ultrasound beam. out(i,j,k) is the intensity at point (x[i],y[j],z[k]). This intensity is calculated by summing the squared vector returned by the Field II function `calc_hp`.

tx is a Field II aperture pointer. This may be set up using the `uf_make_xdc` and `uf_set_beam` commands.

x,y,z are vectors of length 1 or greater which specify the points at which to calculate the field intensity.

Example:

```
field_init(0);
[rf,head]=readFile;
[probe,beamset]=uf_parameters(head);
[tx,rx]=uf_make_xdc(probe);
n_set=1;n_vector=round(beamset(n_set).no_beams/2);
uf_set_beam(tx,rx,probe,beamset,n_set,n_vector);
x=-0.01:0.0002:0.01;
y=0;
z=0.001:0.001:(beamset(n_set).tx_focus_range*2);
intensity=uf_beam_inten(tx,x,y,z);
imagesc(x,z,squeeze(intensity(:,1,:))');axis image;
field_end;
```

Side effects: none.

Chapter 5

Data Files and Structures

URI/FieldII makes use of several data structures for describing the transducer, scan geometry, and phantom.

5.1 Probe Structure

This structure contains all the geometric information about the aperture needed to specify a transducer using `xdc_focused_array`. This is the first of two structures created by `uf_parameters`. Unless otherwise specified, all dimensions are in meters and all frequencies are in Hertz.

The elements of this structure are

field_sample_freq — the sampling frequency, to be used by Field. `uf_make_xdec` makes a call to `set_field` with this value when invoked. This is the sampling frequency assumed for the impulse response and excitaiton pulses as well.

uri_sample_freq — the sampling frequency of the RF data supplied by the URI. This and field_sample_freq can be used to resample the Field-generated data to match the URI RF data.

no_elements — the number of elements in the array

height —the individual element dimension in the y dimension in the Field-II coordinate system.

kerf — the width of the gap between individual transducer elements.

width — the individual element dimension in the x dimension in the Field-II coordinate system.

elv_focus — the depth of the fixed elevational focus supplied by the transducer's lens.

probe_type — a string, either 'linear', 'curvilinear', or 'phased' describing the transducer type.

no_sub_x — the number of mathematical subdivisions of the transducer elements in the x direction used by Field for simulation. The subdivisions are purely for the purpose of simulation and do no correspond to any physical dicing of elements.

no_sub_y — the number of mathematical subdivisions of the transducer elements in the y direction used by Field for simulation. The subdivisions are purely for the purpose of simulation and do no correspond to any physical dicing of elements.

impulse_response — This may be either a row vector or a structure. If it is a row vector, it contains the impulse response to be used in simulation, interpreted at the sample frequency field_sample_freq. If it is a structure, it contains four fields: **f0**, the center frequency, in Hertz; **bw**, the bandwidth as a percentage of the center frequency; **phase**, the phase of the carrier relative to the envelope; and **wavetype**, the name of the function that describes the impulse response envelope, *e.g.* "gaussian".

5.2 Beamset Structure

This structure array contains all the information about the beams that were produced by the aperture to produce a set of beams. This is an array of anywhere from 1 to 8 structures. Elements a particular array are specified by beanset(n).*property*.

type — the type of scan this set is for, either 'B', 'C', 'D', or 'M' for B-mode, Color Doppler, PW Doppler, or M-mode.

no_beams — the number of beams (transducer firings, a-lines) in this set of beams.

origin — a *no_beams*x2 array specifying the x and z coordinates of the starting point of a beam.

direction — a *no_beams*x1 array specifying the angle of the direction in which a beam is fired, relative to the z-axis.

tx_focus_range — the depth of transmit focus. Constant for all beams in a set.

tx_f_num — the f/number of the transmit aperture. Constant for all beams in a set, though the transmit aperture may be truncated by the edge of the array.

tx_freq — the frequency of the excitation pulse.

tx_num_cycles — the (not necessarily integer) number of cycles in a excitation pulse.

prf — the pulse repetition frequency for this set.

tx_apod_type — 0 – rectangular apodization, 1 – hamming apodization

tx_excitation

is_dyn_focus — 0 – fixed receive focus, 1– dynamic receive focus

rx_focus_range — the focal depth if fixed receive focus is used, else 0.

rx_apod_type — 0 – rectangular apodization, 1 – hamming apodization

rx_f_num — the f/number of the receive aperture. The aperture is grown in element steps to maintain this f/number if aperture growth is enabled

aperture_growth — 0 – no receive aperture growth, 1 – fixed aperture size, determined by the fixed receive focal depth and receive f/number.

apex

steering_angle — the angle relative to the z axis the beam is steered, for rectilinear scans.

5.3 Phantom Structure

This structure, generated by uf_phantom*, contains the scatter positions and amplitudes for simulation phantoms. Its elements are

position — an m x 3 array of scatterer coordinates, (x, y, z).

amplitude — an m x 1 array of scatterer amplitudes.

5.4 Probe File

The URI header does not, by itself, contain all the information needed for Field II to perform a simulation. In particular, information about transducer geometry and impulse response are not contained in the URI header. URI/Field II supplies this data in text files which are read by `uf_parameters` as it generates the *probe* structure. Probes can be added to URI/Field II by adding a file to the probes directory. The file name should be that of the probe, as specified by the URI header, plus the extension '.txt'. Filenames should be all lower-case, *e.g.* `vf10-5.txt`.

The probe file for the VF10-5 linear array is shown in figure 5.1. The file contains two columns: the first lists the parameter name, the second, the parameter's value. The parameters may be listed in any order. Columns may be separated with spaces, tabs, or an equal sign (=). Comments may be inserted by prefacing a line with a percent sign (%), which causes all remaining characters on that line to be ignored. Unrecognized parameters will cause a warning to be printed identifying the spurious entry but otherwise ignored. Any entries on a line beyond the second column are ignored.

The parameter names are the same as the values they correspond to in the *probe* data structure. The parameters that are specified in the probe file are

no_elements — the number of elements in the array

height —the individual element dimension in the y dimension in the Field-II coordinate system, in meters.

kerf — the width of the gap between individual transducer elements, in meters.

width — the individual element dimension in the x dimension in the Field-II coordinate system, in meters.

elv_focus — the depth of the fixed elevational focus supplied by the transducer's lens, in meters.

probe_type — a string, either 'linear', 'curvilinear', or 'phased' describing the transducer type.

no_sub_x (no_sub_y) — the number of mathematical subdivisions of the transducer elements in the x (y) direction used by Field for simulation. The subdivisions are purely for the purpose of simulation and do not correspond to any physical dicing of elements.

f0 — the center frequency of the transducer, in Hertz

bw — the bandwidth of the transducer, specified as a percentage of the center frequency.

phase — the phase (in degrees) of the carrier relative to the pulse envelope

wavetype — the function describing the envelope of the impulse response. Currently "gaussian" is the only option.

```
%
% This file describes the VF10-5 probe
%

no_elements     192
height          0.005
kerf            0.00002
width           0.00018
elv_focus       0.025
probe_type      linear
f0              7000000
bw              53
wavetype        gaussian
phase           0
no_sub_x        1
no_sub_y        8
```

Figure 5.1: Probe file for the VF10-5 probe, stored as `vf10-5.txt` on disk.