P4-1-FAPI
Flash-angles pulse inversion imaging for P4-1 transducer (P4-1FAPI)
Code associated with journal article “Fast, low frequency plane-wave imaging for ultrasound contrast imaging” published in Ultrasound in Medicine and Biology, 2018. J. Kusunose, C.F. Caskey
DOI for most recent version of code:
http://doi.org/10.5281/zenodo.840560

Description:
This is a multi-angle plane-wave ultrasound contrast-imaging sequence specifically written and optimized for the P4-1 phased array transducer to be operated on the Verasonics Vantage Research Ultrasound System. The sequence is based upon the Flash-angles method, a plane-wave imaging method with multiple steering angles provided by Verasonics. Pulse-inversion was implemented onto the Flash-angles method to achieve contrast imaging, and variables were empirically optimized for microbubble imaging.

File name: SetUpP4_1FAPI.m

General Instructions

To use the code, you must have the Vantage Verasonics system, Matlab, as well as the package of matlab scripts provided through Verasonics installed onto your computer. Open matlab and set the Vantage folder (with all the Verasonics scripts) as your main folder. To use the P4-1 FAPI code as is, create a P4-1FAPI.mat file by running the setupP4-1FAPI script. Once the matfile is created and placed inside the folder called "matfiles", run VSX and use the P4-1FAPI code, while the appropriate transducer is attatched.

Several things can be modified with relative ease in the script to suite your needs. The following are some of the
* To change the depth of the *image window*, adjust the P.startDepth and P.endDepth. The numbers are represented in wavelength
* Change Na to adjusted the *number of acquisition angles* (currently set to 7). The range of acquisition angles (curretly set -30 to 30 degrees) can be modified by changing the total angle of coverage (set to 60 in the same section)
* Change the Trans.frequency in order to change the *sampling frequency*. THe sampling frequency will be that of 4x(Trans.frequency) in MHz.
* Change the TW (transmit wave) structure to modify the transmit pulse(s). A = *transmit frequency (MHz)*, C = *pulse length* (# of half cycles), D = polarity (-1 = inverted)
* In SeqControl, adjust the SeqControl(2) value, which is set to 200 (us), to change the inter-pulse time. Adjust the SeqControl(3) constant, which is set to 20000 (us) (or 50Hz), to change the image frame rate.

```
% Flash-angles pulse inversion imaging for P4-1 transducer (P4-1FAPI)
%
% Description:
% This is a multi-angle plane-wave ultrasound contrast-imaging
% sequence specifically written and optimized for the P4-1 phased
% array transducer to be operated on the Verasonics Vantage Research
% Ultrasound System. The sequence is based upon the Flash-angles
% method, a plane-wave imaging method with multiple steering angles
% provided by Verasonics. Pulse-inversion was implemented onto the
% Flash-angles method, and variables were optimized for microbubble
% imaging.
%
% File name: SetUpP4_1FAPI.m
%
% Copyright (c) [2017] [Jiro Kusunose]
%
% Permission is hereby granted, free of charge, to any person
% obtaining a copy of this software and associated documentation
% files (the "Software"), to deal in the Software without
% restriction, including without limitation the rights to use, copy,
% modify, merge, publish, distribute, sublicense, and/or sell copies
% of the Software, and to permit persons to whom the Software is
% furnished to do so, subject to the following conditions:
%
% The above copyright notice and this permission notice shall be
% included in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
% EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
% OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
% NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
% HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
% WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
% FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
% OTHER DEALINGS IN THE SOFTWARE.

clear all
P.startDepth = 12;
P.endDepth = 128;    % Acquisition depth in wavelengths

na = 7;      % Number of angles
if na > 1
    dtheta = (60*pi/180)/(na-1); startAngle = -60*pi/180/2;  % set
 dtheta to range over +/- 30 degrees.
else
    dtheta = 0; startAngle = 0;
end

% Specify system parameters.
Resource.Parameters.numTransmit = 128;  % number of transmit channels.
Resource.Parameters.numRcvChannels = 128;  % number of receive
 channels.
```

```
Resource.Parameters.speedOfSound = 1540;
Resource.Parameters.speedCorrectionFactor = 1.0;
Resource.Parameters.verbose = 2;
Resource.Parameters.initializeOnly = 0;
Resource.Parameters.connector = 1;
Resource.Parameters.simulateMode = 0;
%  Resource.Parameters.simulateMode = 1 forces simulate mode, even if
 hardware is present.
%  Resource.Parameters.simulateMode = 2 stops sequence and processes
 RcvData continuously.

% Specify Trans structure array.
Trans.name = 'P4-1';
Trans.frequency = 6.25;
Trans.units = 'wavelengths'; % Explicit declaration avoids warning
 message when selected by default
Trans = computeTrans(Trans);
Trans.maxHighVoltage = 50;  % set maximum high voltage limit for
 pulser supply.

P.theta = -pi/4;
P.rayDelta = 2*(-P.theta);
P.aperture = Trans.numelements*Trans.spacing; % P.aperture in
 wavelengths
P.radius = (P.aperture/2)/tan(-P.theta); % dist. to virt. apex

% Set up PData structure.
PData(1).PDelta = [0.875, 0, 0.5];
PData(1).Size(1) = 10 + ceil((P.endDepth-P.startDepth) /
PData(1).PDelta(3));
PData(1).Size(2) = 10 + ceil(2*(P.endDepth + P.radius) *sin(-P.theta)/
PData(1).PDelta(1));
PData(1).Size(3) = 1;
PData(1).Origin = [-(PData(1).Size(2)/2)
 *PData(1).PDelta(1),0,P.startDepth];
PData(1).Region = struct(...
            'Shape',struct('Name','SectorFT', ...
            'Position',[0,0,-P.radius], ...
            'z',P.startDepth, ...
            'r',P.radius+P.endDepth, ...
            'angle',P.rayDelta, ...
            'steer',0));

% Specify Resources.
Resource.RcvBuffer(1).datatype = 'int16';
Resource.RcvBuffer(1).rowsPerFrame = 2*na*4096;
Resource.RcvBuffer(1).colsPerFrame =
 Resource.Parameters.numRcvChannels;
Resource.RcvBuffer(1).numFrames = 10;     % 10 frames used for RF
 cineloop.
Resource.InterBuffer(1).numFrames = 1;    % one intermediate buffer
 defined but not used.
Resource.ImageBuffer(1).numFrames = 5;
Resource.DisplayWindow(1).Title = 'P4-1FlashAnglesPulseInversion';
Resource.DisplayWindow(1).pdelta = 0.35;
ScrnSize = get(0,'ScreenSize');
DwWidth = ceil(PData(1).Size(2)*PData(1).PDelta(1) /
Resource.DisplayWindow(1).pdelta);
DwHeight = ceil(PData(1).Size(1)*PData(1).PDelta(3) /
Resource.DisplayWindow(1).pdelta);
Resource.DisplayWindow(1).Position = [250,(ScrnSize(4)-(DwHeight
+150))/2, ...  % lower left corner position
                                      DwWidth, DwHeight];
Resource.DisplayWindow(1).ReferencePt = [PData(1).Origin(1),
 0,PData(1).Origin(3)];   % 2D imaging is in the X,Z plane
Resource.DisplayWindow(1).numFrames = 5;
Resource.DisplayWindow(1).AxesUnits = 'mm';
Resource.DisplayWindow.Colormap = gray(256);

% Specify TW structure array.

TW(1).type = 'parametric';
TW(1).Parameters = [1.8382,.67,2,1];   % A, B, C, D
% Inverted pulse
TW(2).type = 'parametric';
TW(2).Parameters = [1.8382,.67,2,-1];   % A, B, C, D

% Specify TX structure array.
TX = repmat(struct('waveform', 1, ...
                   'Origin', [0.0,0.0,0.0], ...
                   'focus', -P.radius, ...
                   'Steer', [0.0,0.0], ...
                   'Apod', ones(1,Trans.numelements), ...  % set
 TX.Apod for 96 elements
                   'Delay', zeros(1,Trans.numelements)), 1, 2*na);
% - Set event specific TX attributes.
for n = 1:na   % na transmit events
    TX(n).Steer = [(startAngle+(n-1)*dtheta),0.0];
    TX(n).Delay = computeTXDelays(TX(n));
    TX(n+na) = TX(n);
    TX(n+na).waveform = 2;
end

% Specify Receive structure arrays.
maxAcqLength = sqrt(P.aperture^2 + P.endDepth^2 -
 2*P.aperture*P.endDepth*cos(P.theta-pi/2)) - P.startDepth;
wlsPer128 = 128/(4*2); % wavelengths in 128 samples for 4
 samplesPerWave
Receive = repmat(struct('Apod', ones(1,Trans.numelements), ...
                        'startDepth', P.startDepth, ...
                        'endDepth', P.startDepth +
 wlsPer128*ceil(maxAcqLength/wlsPer128), ...
                        'TGC', 1, ...
                        'bufnum', 1, ...
                        'framenum', 1, ...
                        'acqNum', 1, ...
                        'sampleMode', 'NS200BW', ...
                        'mode', 0, ...
```

```
  'callMediaFunc',0),1,2*na*Resource.RcvBuffer(1).numFrames);

% - Set event specific Receive attributes.
for i = 1:Resource.RcvBuffer(1).numFrames
    Receive(2*na*(i-1)+1).callMediaFunc = 1;
    for j = 1:na
        Receive(2*na*(i-1)+j).framenum = i;
        Receive(2*na*(i-1)+j).acqNum = j;
        Receive(2*na*(i-1)+na+j).framenum = i;
        Receive(2*na*(i-1)+na+j).acqNum = j;
        Receive(2*na*(i-1)+na+j).mode = 1;
    end
end

% Specify TGC Waveform structure.
TGC.CntrlPts = [[91,590,651,710,769,675,716,665]];
TGC.rangeMax = P.endDepth;
TGC.Waveform = computeTGCWaveform(TGC);

% Specify Recon structure arrays.
Recon = struct('senscutoff', 0.45, ...
               'pdatanum', 1, ...
               'rcvBufFrame', -1, ...
               'IntBufDest', [1,1], ...
               'ImgBufDest', [1,-1], ...
               'RINums', 1:2*na);

% Define ReconInfo structures.
% We need na ReconInfo structures for na steering angles.
ReconInfo = repmat(struct('mode', 4, ...  % accumulate IQ data.
                   'txnum', 1, ...
                   'rcvnum', 1, ...
                   'regionnum', 1), 1, 2*na);
% - Set specific ReconInfo attributes.
if na>1
    ReconInfo(1).mode = 'replaceIQ';
    for j = 1:na  % For each row in the column
        ReconInfo(j).txnum = j;
        ReconInfo(j).rcvnum = j;
        ReconInfo(j+na).txnum = j+na;
        ReconInfo(j+na).rcvnum = j+na;
    end
    ReconInfo(2*na).mode = 'accumIQ_replaceIntensity';  % accumulate
 and detect
else
    ReconInfo(1).mode = 'replaceIntensity';
end


% Specify Process structure array.
pers = 20;
cmpFactor = 40;
Process(1).classname = 'Image';
Process(1).method = 'imageDisplay';
Process(1).Parameters = {'imgbufnum',1,...   % number of buffer to
 process.
                         'framenum',-1,...   % (-1 => lastFrame)
                         'pdatanum',1,...    % number of PData
 structure to use
                         'pgain',1.0,...     % pgain is image
 processing gain
                         'reject',2,...
                         'grainRemoval','none',...
                         'persistMethod','none',...
                         'persistLevel',pers,...
                         'interp','4pt',...      % method of
 interpolation (1=4pt interp)
                         'processMethod','none',...
                         'averageMethod','none',...
                         'compressMethod','power',...
                         'compressFactor',cmpFactor,...
                         'mappingMethod','full',...
                         'display',1,...      % display image after
 processing
                         'displayWindow',1};

% Specify SeqControl structure arrays.  Missing fields are set to
 NULL.
SeqControl(1).command = 'jump'; % jump back to start
SeqControl(1).argument = 1;
SeqControl(2).command = 'timeToNextAcq';  % time between each transmit
SeqControl(2).argument = 200;  % 200 us
SeqControl(3).command = 'timeToNextAcq';  % time between frames
SeqControl(3).argument = 20000 - (2*na-1)*200;  % 20 msec, or 50Hz FPS
SeqControl(4).command = 'returnToMatlab';
nsc = 5;

% Specify Event structure arrays.
n = 1;
for i = 1:Resource.RcvBuffer(1).numFrames
    for j = 1:na                    % Acquire frame
        Event(n).info = 'Acquire full aperture.';
        Event(n).tx = j;   % use "normal" TX structure.
        Event(n).rcv = 2*na*(i-1)+j;
        Event(n).recon = 0;      % no reconstruction.
        Event(n).process = 0;    % no processing
        Event(n).seqControl = 2; % break before next acquisition
        n = n+1;

        Event(n).info = 'Acquire full aperture.';
        Event(n).tx = j+na;   % use inverted TX structure.
        Event(n).rcv = 2*na*(i-1)+j+na;
        Event(n).recon = 0;      % no reconstruction.
        Event(n).process = 0;    % no processing
        Event(n).seqControl = 2; % break before next acquisition
        n = n+1;
    end
    Event(n-1).seqControl = [3,nsc]; % modify last event's seqCntrl:
 time between frames & transferToHostuse
       SeqControl(nsc).command = 'transferToHost';
       nsc = nsc + 1;

    Event(n).info = 'recon and process';
    Event(n).tx = 0;         % no transmit
    Event(n).rcv = 0;        % no rcv
    Event(n).recon = 1;      % reconstruction
    Event(n).process = 1;    % process
    Event(n).seqControl = 0;
    if floor(i/3) == i/3     % Exit to Matlab every 3rd frame
        Event(n).seqControl = 4;
    end
    n = n+1;
end

Event(n).info = 'Jump back';
Event(n).tx = 0;        % no TX
Event(n).rcv = 0;       % no Rcv
Event(n).recon = 0;     % no Recon
Event(n).process = 0;
Event(n).seqControl = 1;


% User specified UI Control Elements
% - Sensitivity Cutoff
UI(1).Control =  {'UserB7','Style','VsSlider','Label','Sens.
 Cutoff',...
                  'SliderMinMaxVal',[0,1.0,Recon(1).senscutoff],...
                  'SliderStep',[0.025,0.1],'ValueFormat','%1.3f'};
UI(1).Callback = text2cell('%SensCutoffCallback');

% - Range Change
MinMaxVal = [64,300,P.endDepth]; % default unit is wavelength
AxesUnit = 'wls';
if isfield(Resource.DisplayWindow(1),'AxesUnits')
 &&~isempty(Resource.DisplayWindow(1).AxesUnits)
    if strcmp(Resource.DisplayWindow(1).AxesUnits,'mm');
        AxesUnit = 'mm';
        MinMaxVal = MinMaxVal *
 (Resource.Parameters.speedOfSound/1000/Trans.frequency);
    end
end
UI(2).Control = {'UserA1','Style','VsSlider','Label',['Range
 (',AxesUnit,')'],...
                 'SliderMinMaxVal',MinMaxVal,'SliderStep',
[0.1,0.2],'ValueFormat','%3.0f'};
UI(2).Callback = text2cell('%RangeChangeCallback');

% Specify factor for converting sequenceRate to frameRate.
frameRateFactor = 3;

% Save all the structures to a .mat file.
```

```
save('MatFiles/P4-1FAPI');
disp(['P4-1FAPI']);

return


% **** Callback routines to be converted by text2cell function. ****
%SensCutoffCallback - Sensitivity cutoff change
ReconL = evalin('base', 'Recon');
for i = 1:size(ReconL,2)
    ReconL(i).senscutoff = UIValue;
end
assignin('base','Recon',ReconL);
Control = evalin('base','Control');
Control.Command = 'update&Run';
Control.Parameters = {'Recon'};
assignin('base','Control', Control);
return
%SensCutoffCallback

%RangeChangeCallback - Range change
simMode = evalin('base','Resource.Parameters.simulateMode');
% No range change if in simulate mode 2.
if simMode == 2
    set(hObject,'Value',evalin('base','P.endDepth'));
    return
end
Trans = evalin('base','Trans');
Resource = evalin('base','Resource');
scaleToWvl = Trans.frequency/(Resource.Parameters.speedOfSound/1000);

P = evalin('base','P');
P.endDepth = UIValue;
if isfield(Resource.DisplayWindow(1),'AxesUnits')
 &&~isempty(Resource.DisplayWindow(1).AxesUnits)
    if strcmp(Resource.DisplayWindow(1).AxesUnits,'mm');
        P.endDepth = UIValue*scaleToWvl;
    end
end
assignin('base','P',P);

PData = evalin('base','PData');
PData(1).Size(1) = 10 + ceil((P.endDepth-P.startDepth)/
PData(1).PDelta(3));
PData(1).Region = struct(...
            'Shape',struct('Name','SectorFT', ...
            'Position',[0,0,-P.radius], ...
            'z',P.startDepth, ...
            'r',P.radius+P.endDepth, ...
            'angle',P.rayDelta, ...
            'steer',0));
PData(1).Region = computeRegions(PData(1));
assignin('base','PData',PData);
```

```
evalin('base','Resource.DisplayWindow(1).Position(4)
 = ceil(PData(1).Size(1)*PData(1).PDelta(3) /
 Resource.DisplayWindow(1).pdelta);');
Receive = evalin('base', 'Receive');
maxAcqLength =
 sqrt(P.aperture^2+P.endDepth^2-2*P.aperture*P.endDepth*cos(P.theta-
pi/2))-P.startDepth;
wlsPer128 = 128/(4*2);
for i = 1:size(Receive,2)
    Receive(i).endDepth = P.startDepth + wlsPer128*ceil(maxAcqLength/
wlsPer128);
end
assignin('base','Receive',Receive);
evalin('base','TGC.rangeMax = P.endDepth;');
evalin('base','TGC.Waveform = computeTGCWaveform(TGC);');
evalin('base','if VDAS==1, Result = loadTgcWaveform(1); end');
Control = evalin('base','Control');
Control.Command = 'update&Run';
Control.Parameters =
 {'PData','InterBuffer','ImageBuffer','DisplayWindow','Receive','Recon'};
assignin('base','Control', Control);
assignin('base', 'action', 'displayChange');
return
%RangeChangeCallback
```

Published with MATLAB® R2015b