```
function [ doa_meters, doa_samples, reliability ] = tdoa2(signal1_complex,
signal2_complex, rx_distance_diff, rx_distance, ...
                                                          smoothing_factor,
corr_type,  report_level, signal_bandwidth_khz, ...
                                                          ref_bandwidth_khz,
smoothing_factor_ref, interpol)
    % tdoa2 calculates the TDOA of two signals captured by two RXs
    %   output:
    %   doa_meters:             delay in meters (how much signal1 is later than
signal 2)
    %   doa_samples:            delay in samples
    %   reliability:            reliab. of the correlation, 0(bad)..1(good)
    %
    %   input:
    %   signal1:                        signal with length 3.6e6 from RX 1
    %   signal2:                        signal with length 3.6e6 from RX 2
    %   rx_distance_diff:       difference in distance in meters between two RX to
Ref (sign matters)
    %   rx_distance:        distance between RX1 and RX2 in meters (always
positive)
    %   smooting_factor:        for wideband signals
    %   corr_type:                      switch between abs and delta phase (abs:
0, delta phase: 1);
    %   report_level:           no reports: 0, show figures >0
    %   signal_bandwidth_khz bandwidth of FIR filter for signal filtering applied
to meas signal (400, 200, 40, 12, 0)
    %   interpol            interpolation factor (0 or 1 = no interpolation)
    %
    %   requirement: signal capture with 2 Msps:

    %   still open: correct valid signal generation for interpolation
    %               (currently: native valid signal is used, which gives an
approximation, which should be ok)


    % integrity checks
    if (length(signal1_complex) ~= 3.6e6)
        error ('Length of Signal 1 is unequal 3.6e6');
    end;

    if (length(signal2_complex) ~= 3.6e6)
        error ('Length of Signal 2 is unequal 3.6e6');
    end;


    % slice signal into three parts
    % 111111111111111111111111xxxxxxxxxxxxx2222222222222222222222xxx3333..
    % |-num_samples_per_slice-|
    % |-num_samples_per_freq+guard_interval-|-num_samples_per_slice-|
    % |-------2*num_samples_per_freq + % guard_interval----------------|..
```

```
    num_samples_per_freq = 1.2e6;
    num_samples_per_slice = 1e6;
    guard_interval = 200e3; % time to switch to a new frequency, fixed,
empirically determined
    sample_rate = 2e6;  % in Hz
    num_samples_total = num_samples_per_freq*3;

    signal11_complex = signal1_complex(1                                    :
num_samples_per_slice);
    signal12_complex = signal1_complex(num_samples_per_freq   + guard_interval :
num_samples_per_freq   + guard_interval + num_samples_per_slice - 1);
    signal13_complex = signal1_complex(2*num_samples_per_freq + guard_interval :
2*num_samples_per_freq + guard_interval + num_samples_per_slice - 1);

    signal21_complex = signal2_complex(1                                    :
num_samples_per_slice);
    signal22_complex = signal2_complex(num_samples_per_freq   + guard_interval :
num_samples_per_freq   + guard_interval + num_samples_per_slice - 1);
    signal23_complex = signal2_complex(2*num_samples_per_freq + guard_interval :
2*num_samples_per_freq + guard_interval + num_samples_per_slice - 1);

    %% Filter measurement to signal bandwidth

    disp('Filter measurement signal to actual bandwidth');

    signal12_complex_unfiltered = signal12_complex; % copy unfiltered signals for
display
    signal22_complex_unfiltered = signal22_complex;

    signal12_complex = filter_iq(signal12_complex, signal_bandwidth_khz);
    signal22_complex = filter_iq(signal22_complex, signal_bandwidth_khz);

    %% Filter Ref Signal
    signal11_complex = filter_iq(signal11_complex, ref_bandwidth_khz);
    signal13_complex = filter_iq(signal13_complex, ref_bandwidth_khz);
    signal21_complex = filter_iq(signal21_complex, ref_bandwidth_khz);
    signal23_complex = filter_iq(signal23_complex, ref_bandwidth_khz);

    disp(' ');


    if report_level > 2
        % display spectrum before and after filtering

        figure;
        subplot(2,1,1);
        spectrum1 = 10*log10(abs(fftshift(fft(signal12_complex_unfiltered))));
        spectrum1 = smooth(spectrum1, 201);

        spectrum2 = 10*log10(abs(fftshift(fft(signal22_complex_unfiltered))));
        spectrum2 = smooth(spectrum2, 201);
```

```
        freq_axis = -(length(spectrum1)/2) : 1 : ((length(spectrum1)/2)-1);
        plot(freq_axis, spectrum1, freq_axis, spectrum2);
        title('Measurement Signals, before filtering');
        grid;

        subplot(2,1,2);
        spectrum1 = 10*log10(abs(fftshift(fft(signal12_complex))));
        spectrum1 = smooth(spectrum1, 201);

        spectrum2 = 10*log10(abs(fftshift(fft(signal22_complex))));
        spectrum2 = smooth(spectrum2, 201);

        freq_axis = -(length(spectrum1)/2) : 1 : ((length(spectrum1)/2)-1);
        plot(freq_axis, spectrum1, freq_axis, spectrum2);
        title('Measurement Signals, after filtering');
        grid;
    end


    %% Correlation for slice 1 (ref)
    disp('CORRELATION CALCULATION DETAILS:');

    % native
    corr_signal_1 = correlate_iq(signal11_complex, signal21_complex, corr_type,
smoothing_factor_ref);
    corr1_reliability = corr_reliability( corr_signal_1 );

    [~, idx1] = max(corr_signal_1);
    delay1_native = idx1 - length(signal11_complex); % >0: signal1 later, <0
signal2 later

    % with interpolation
    if (interpol > 1)
        signal11_interp = interp(signal11_complex, interpol);
        signal12_interp = interp(signal21_complex, interpol);

        corr_signal_1_interp = correlate_iq(signal11_interp, signal12_interp,
corr_type, smoothing_factor_ref);
        [~, idx1_interp_i] = max(corr_signal_1_interp);
        idx1_interp = (idx1_interp_i-1) ./ interpol + (1/interpol);
        delay1_interp = idx1_interp - length(signal11_complex); % >0: signal1
later, <0 signal2 later
    else
        delay1_interp = 0;
    end

%%  zur Sicherheit = untuk antisipasi
%% +1 falls abgerundet wird = pembulatan kebawah
%% +2 wegen m glichem Frequenzdrif = frekuensi drift yang mungkin terjadi
```

```
    % Determining valid correlation area, such that: toa < distance of the two
RXes
    delay_mask = zeros(1, length(corr_signal_1));
    valid_samples_right = (rx_distance - rx_distance_diff ) / (3e8 / sample_rate);
 % number of valid samples right of idx2
    valid_samples_left = -(-rx_distance - rx_distance_diff ) / (3e8 /
sample_rate); % number of valid samples left of idx2

    for ii=1:valid_samples_right+1+2   % +1 zur Sicherheit, falls abgerundet wird
und +2 wegen m glichem Frequenzdrift
        delay_mask(idx1+ii) = 0.8;
    end

    for ii=1:valid_samples_left+1+2   % +1 zur Sicherheit, falls abgerundet wird
und +2 wegen m glichem Frequenzdrift
        delay_mask(idx1-ii) = 0.8;
    end

    delay_mask(idx1) = 0.7;


    %% Correlation for slice 2 (measure)
    corr_signal_2 = correlate_iq(signal12_complex, signal22_complex, corr_type,
smoothing_factor);

    %truncate to valid area
    corr_signal_2_valid = zeros(length(corr_signal_2),1)-1;
    corr_signal_2_valid(idx1) = corr_signal_2(idx1);

     for ii=1:valid_samples_right+1+2   % +1 zur Sicherheit, falls abgerundet wird
und +2 wegen m glichem Frequenzdrift
         corr_signal_2_valid(idx1+ii) = corr_signal_2(idx1+ii);
     end

     for ii=1:valid_samples_left+1+2   % +1 zur Sicherheit, falls abgerundet wird
und +2 wegen m glichem Frequenzdrift
         corr_signal_2_valid(idx1-ii) = corr_signal_2(idx1-ii);
     end

    %corr_signal_2_valid = corr_signal_2; % truncation abschalten

    corr2_reliability = corr_reliability(corr_signal_2_valid);

    [~, idx2] = max(corr_signal_2_valid);
    delay2_native = idx2 - length(signal12_complex); % >0: signal1 later, <0
signal2 later


    % with interpolation, noch ohne Valid
    if (interpol > 1)
        signal12_interp = interp(signal12_complex, interpol);
```

```
        signal22_interp = interp(signal22_complex, interpol);

        corr_signal_2_interp = correlate_iq(signal12_interp, signal22_interp,
corr_type, smoothing_factor);

        %truncate to valid area
        corr_signal_2_valid_interp = zeros(length(corr_signal_2_interp),1)-1;
        corr_signal_2_valid_interp(idx1_interp_i) =
corr_signal_2_interp(idx1_interp_i);

        for ii=1:interpol*(valid_samples_right+1+2)   % +1 zur Sicherheit, falls
abgerundet wird und +2 wegen m glichem Frequenzdrift
          corr_signal_2_valid_interp(idx1_interp_i+ii) =
corr_signal_2_interp(idx1_interp_i+ii);
        end

        for ii=1:interpol*(valid_samples_left+1+2)   % +1 zur Sicherheit, falls
abgerundet wird und +2 wegen m glichem Frequenzdrift
          corr_signal_2_valid_interp(idx1_interp_i-ii) =
corr_signal_2_interp(idx1_interp_i-ii);
        end

        [~, idx2_interp_i] = max(corr_signal_2_valid_interp);
        idx2_interp = (idx2_interp_i-1) ./ interpol + (1/interpol);
        delay2_interp = idx2_interp - length(signal12_complex); % >0: signal1
later, <0 signal2 later
    else
        delay2_interp = 0;
    end


    %% Correlation for slice 3 (ref check)
    %native
    corr_signal_3 = correlate_iq(signal13_complex, signal23_complex, corr_type,
smoothing_factor_ref);
    corr3_reliability = corr_reliability(corr_signal_3);

    [~, idx3] = max(corr_signal_3);
    delay3_native = idx3 - length(signal13_complex);

    % with interpolation
    if (interpol > 1)
        signal13_interp = interp(signal13_complex, interpol);
        signal23_interp = interp(signal23_complex, interpol);

        corr_signal_3_interp = correlate_iq(signal13_interp, signal23_interp,
corr_type, smoothing_factor_ref);
        [~, idx3_interp_i] = max(corr_signal_3_interp);
        idx3_interp = (idx3_interp_i-1) ./ interpol + (1/interpol);
        delay3_interp = idx3_interp - length(signal13_complex); % >0: signal1
later, <0 signal2 later
```

```
    else
        delay3_interp = 0;
    end



    %% Display Correlation Signals

    % display reference and reference check signal
    if (report_level > 0)
        figure('units','normalized','outerposition',[0 0 1 1]);
        subplot(4,2,1);

        if (interpol > 1)
            x_corr_interp = 1:(2*interpol*length(signal11_complex) - 1);
            x_corr_interp_plot = (x_corr_interp-1)./interpol +(1/interpol);
        end

        plot(1:length(corr_signal_1), corr_signal_1, 1:length(corr_signal_3),
corr_signal_3,...
            1:length(corr_signal_2), delay_mask, 'r', idx1,
corr_signal_1(idx1),'dr', idx3, corr_signal_3(idx3),'dr');

        title('REFERENCE SIGNAL: correlations, full span'); legend('ref (slice
1)','ref check (slice 3)', 'ref interp'); ylim([-0.1 1.1]); grid;

        win_len1 = 10000;
        win_len2 = 500;
        win_len3 = 50;

        %[c, idx] = max(corr_signal_1);
        idx = idx1;

        idx_left = idx-win_len1;
        if idx_left < 1
            idx_left = 1;
        end;
        idx_right = idx+win_len1;
        if idx_right > length(corr_signal_1)
            idx_right = length(corr_signal_1);
        end;

        subplot(4,2,3);
        plot(idx_left:idx_right, corr_signal_1(idx_left:idx_right), 'o-',
idx_left:idx_right, corr_signal_3(idx_left:idx_right), 'x-', idx_left:idx_right,
delay_mask(idx_left:idx_right), 'r', idx1, corr_signal_1(idx1),'dr', idx3,
corr_signal_3(idx3),'dr');
        title('Zoom 1 on max'); legend('ref (slice 1)','ref check (slice 3)');
ylim([-0.1 1.1]); xlim([idx_left idx_right]); grid;

        idx_left = idx-win_len2;
```

```
        if idx_left < 1
            idx_left = 1;
        end;
        idx_right = idx+win_len2;
        if idx_right > length(corr_signal_1)
            idx_right = length(corr_signal_1);
        end;

        subplot(4,2,5);
        plot(idx_left:idx_right, corr_signal_1(idx_left:idx_right),'o-',
idx_left:idx_right, corr_signal_3(idx_left:idx_right) ,'x-', idx_left:idx_right,
delay_mask(idx_left:idx_right), 'r', idx1, corr_signal_1(idx1),'dr', idx3,
corr_signal_3(idx3),'dr');
        title('Zoom 2 on max'); legend('ref (slice 1)','ref check (slice 3)');
ylim([-0.1 1.1]); xlim([idx_left idx_right]); grid;

        idx_left = idx-win_len3;
        if idx_left < 1
            idx_left = 1;
        end;
        idx_right = idx+win_len3;
        if idx_right > length(corr_signal_1)
            idx_right = length(corr_signal_1);
        end;

        subplot(4,2,7);

        if (interpol <= 1)
            plot(idx_left:idx_right, corr_signal_1(idx_left:idx_right),'o-',
idx_left:idx_right, corr_signal_3(idx_left:idx_right) ,'x-', ...
                idx_left:idx_right, delay_mask(idx_left:idx_right), 'r', idx1,
corr_signal_1(idx1),'dr', idx3, corr_signal_3(idx3),'dr');
        else
            plot(idx_left:idx_right, corr_signal_1(idx_left:idx_right),'o-',
idx_left:idx_right, corr_signal_3(idx_left:idx_right) ,'x-', ...
                x_corr_interp_plot((interpol*idx_left):(interpol*idx_right)),
corr_signal_1_interp((interpol*idx_left):(interpol*idx_right)), 'm.',...
                x_corr_interp_plot((interpol*idx_left):(interpol*idx_right)),
corr_signal_3_interp((interpol*idx_left):(interpol*idx_right)), 'c.',...
                idx_left:idx_right, delay_mask(idx_left:idx_right), 'r',
idx1_interp, corr_signal_1_interp(idx1_interp_i),'dr', idx3_interp,
corr_signal_3_interp(idx3_interp_i),'dr');
        end

        title('Zoom 3 on max'); legend('ref (slice 1)','ref check (slice 3)', 'ref
interp', 'ref check interp'); ylim([-0.1 1.1]); xlim([idx_left idx_right]); grid;


        % display measurement signal
        subplot(4,2,2);
        plot(1:length(corr_signal_2), corr_signal_2, 1:length(corr_signal_2),
```

```
delay_mask, 'r', idx2, corr_signal_2(idx2),'dr');
        title('MEASUREMENT SIGNAL: correlations, full span'); ylim([-0.1 1.1]);
grid;

        win_len1 = 10000;
        win_len2 = 500;
        win_len3 = 50;

        %[c, idx] = max(corr_signal_2);
        idx = idx2;

        idx_left = idx-win_len1;
        if idx_left < 1
            idx_left = 1;
        end;
        idx_right = idx+win_len1;
        if idx_right > length(corr_signal_2)
            idx_right = length(corr_signal_2);
        end;

        subplot(4,2,4);
        plot(idx_left:idx_right, corr_signal_2(idx_left:idx_right), 'o-',
idx_left:idx_right, delay_mask(idx_left:idx_right), 'r', idx2,
corr_signal_2(idx2),'dr');
        title('Zoom 1 on max'); ylim([-0.1 1.1]); xlim([idx_left idx_right]);
grid;

        idx_left = idx-win_len2;
        if idx_left < 1
            idx_left = 1;
        end;
        idx_right = idx+win_len2;
        if idx_right > length(corr_signal_2)
            idx_right = length(corr_signal_2);
        end;

        subplot(4,2,6);
        plot(idx_left:idx_right, corr_signal_2(idx_left:idx_right),'o-',
idx_left:idx_right, delay_mask(idx_left:idx_right), 'r', idx2,
corr_signal_2(idx2),'dr');
        title('Zoom 2 on max'); ylim([-0.1 1.1]); xlim([idx_left idx_right]);
grid;

        idx_left = idx-win_len3;
        if idx_left < 1
            idx_left = 1;
        end;
        idx_right = idx+win_len3;
        if idx_right > length(corr_signal_2)
            idx_right = length(corr_signal_2);
        end;
```

```
        subplot(4,2,8);

        if (interpol <= 1)
            plot(idx_left:idx_right, corr_signal_2(idx_left:idx_right),'o-',...
                idx_left:idx_right, delay_mask(idx_left:idx_right), 'r', idx2,
corr_signal_2(idx2),'dr');
        else
            plot(idx_left:idx_right, corr_signal_2(idx_left:idx_right),'o-',...
                x_corr_interp_plot((interpol*idx_left):(interpol*idx_right)),
corr_signal_2_interp((interpol*idx_left):(interpol*idx_right)), 'm.',...
                idx_left:idx_right, delay_mask(idx_left:idx_right), 'r',
idx2_interp, corr_signal_2_interp(idx2_interp_i),'dr');
        end

        title('Zoom 3 on max'); ylim([-0.1 1.1]); xlim([idx_left idx_right]);
grid;
    end;



    %% Calculate Correlation Results

    if (interpol <= 1)
        delay1 = delay1_native;
        delay2 = delay2_native;
        delay3 = delay3_native;
    else
        delay1 = delay1_interp;
        delay2 = delay2_interp;
        delay3 = delay3_interp;
    end


    if abs(delay1 - delay3) <= 2
        avg_delay13 = (delay1 + delay3) / 2; % this delay includes: 1) different
reception start time 2) ref signal delay due to different distances to ref
transmitter
    else
        disp('<strong>WARNING: BAD REFERENCE SIGNALS: ref delays differ by more
than 2 samples! </strong>');
        if corr1_reliability > corr3_reliability
            avg_delay13 = delay1;
            disp(['<strong>taking ref with higher reliability, i.e. ref
(reliability: ' num2str(corr1_reliability) '(ref) > ' num2str(corr3_reliability)
'(ref check) </strong>']);
        else
            avg_delay13 = delay3;
            disp(['<strong>taking ref with higher reliability, i.e. ref check
(reliability: ' num2str(corr1_reliability) '(ref) < ' num2str(corr3_reliability)
'(ref check) </strong>']);
```

```
        end

    end


    ref_signal_diff_samples = (rx_distance_diff / 3e8) * sample_rate; % known ref
signal delay in samples

    % doa_samples/_meters specifies how much signal1 is later than signal2
    doa_samples = delay2 - avg_delay13 + ref_signal_diff_samples; % time
difference of arrival without delays due to reception start time and ref
transmitter (desc. above)
    doa_meters = (doa_samples / sample_rate) * 3e8;


    reliability = min([corr3_reliability, corr2_reliability, corr1_reliability]);

    disp(' ');
    disp('CORRELATION RESULTS');

    disp(['raw delay1 (ref) (nativ/interp): ' int2str(delay1_native) ' / '
num2str(delay1_interp) ', reliability nativ (0..1): '
num2str(corr1_reliability)]);
    disp(['raw delay2 (measure) (nativ/interp): ' int2str(delay2_native) ' / '
num2str(delay2_interp) ', reliability nativ: ' num2str(corr2_reliability)]);
    disp(['raw delay3 (ref check) (nativ/interp): ' int2str(delay3_native) ' / '
num2str(delay3_interp) ', reliability nativ: ' num2str(corr3_reliability)]);
    disp(['merged delay of ref and ref check: ' num2str(avg_delay13) ]);
    disp(' ');

    disp(['specified distance difference to ref tx [m]: '
int2str(rx_distance_diff)]);
    disp(['specified distance difference to ref tx [samples]: '
num2str(ref_signal_diff_samples)]);
    disp(['specified distance between two RXes [m]: ' num2str(rx_distance)]);
    disp(' ');

    disp('FINAL RESULT');
    disp(['TDOA in samples: ' num2str(doa_samples) '(how much is signal1 later
than signal2)']);
    disp(['TDOA in distance [m]: ' num2str(doa_meters) ]);
    disp(['Total Reliability (min of all 3): ' num2str(reliability) ]);
    disp(' ');
end
```